industriales
etsii

Escuela Técnica Superior de Ingeniería Industrial

Universidad Politécnica de Cartagena

UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Escuela Técnica Superior de Ingeniería Industrial

Sensorización de un robot paralelo para aplicaciones médicas

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

Autor: **Valdez Vidal, Laura**
Director: Miguel Almonacid Kroeger
Codirector: José Manuel Cano Izquierdo

Cartagena, mayo de 2018

UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Departamento de Ingeniería de Sistemas y Automática

SENSORIZACIÓN DE UN ROBOT PARALELO PARA APLICACIONES MÉDICAS

Trabajo Fin de Grado

Laura Valdez Vidal

Director: Miguel Almonacid Kroeger

Codirector: José Manuel Cano Izquierdo

2018

A mis Tutores y al Doctor César Salcedo, por fomentar
mi interés en aplicar la ingeniería en ramas
que antes desconocía.

A mi familia, en especial a mis Padres, por confiar
en mí y apoyarme siempre.

A mis Compañeros y Amigos, por recordarme que
soy capaz de conseguir lo que me proponga.

A tantos Profesores que me han enseñado
a ganarme las cosas con esfuerzo.

RESUMEN

El presente Trabajo de Fin de Grado tiene por objetivo la sensorización del robot paralelo TrueLock Hexapod usado en operaciones de Traumatología y Ortopedia. Para ello, se analizan las características de una variedad de sensores utilizando como electrónica de adquisición y procesamiento un microcontrolador Arduino Uno Rev 3.

Un requisito fundamental en el presente estudio, es que la sensorización del robot debe tener un mínimo impacto desde el punto de vista mecánico sobre el mismo.

Un análisis previo de las distintas características de los robots paralelos se desarrolla en los capítulos iniciales. En base al funcionamiento y prescripciones indicadas por los doctores en cuanto al uso del sistema, se ha desarrollado en "Processing" una interfaz que mediante comunicación inalámbrica con el robot permite conocer la posición de los actuadores en cada momento con una precisión milimétrica.

Finalmente, en los últimos capítulos se desarrollan los resultados de la integración de los sensores seleccionados y posteriormente, se indican los futuros trabajos en relación al presente robot para una futura solución integral que pueda ser utilizada en el campo de la medicina.

ABSTRACT

The aim of this Final Degree Project is the sensorization of the TrueLock Hexapod parallel robot used in Traumatology and Orthopedics operations. For this, the characteristics of a variety of sensors are analyzed using an Arduino One Rev 3 microcontroller as acquisition and processing electronics.

A fundamental requirement in the present study, is that the sensorization of the robot must have a minimum impact from the mechanical point of view on it.

A previous analysis of the different characteristics of the parallel robots is developed in the initial chapters. Based on the operation and prescriptions indicated by the doctors regarding the use of the system, an "interface" has been developed in "Processing" that, by means of wireless communication with the robot, allows to know the position of the actuators at every moment with millimeter precision.

Finally, in the last chapters the results of the integration of the selected sensors are developed and subsequently, future works are indicated in relation to the present robot for a future integral solution that can be used in the field of medicine.

Índice de Contenido

Índice de Figuras

Índice de Tablas

Capítulo 1

Introducción

1.1 Marco de Trabajo

El fin último del presente trabajo es conseguir la sensorización de un robot paralelo utilizado para aplicaciones médicas, conocido como "TrueLock Hexapod" (TL-Hex) desarrollado en "Texas Scottish Rite Hospital for Children" (TSRHC) en Dallas, Texas. Actualmente, los brazos de los que dispone este robot no son actuados, es decir, los distintos posicionamientos marcados por prescripción médica deben ser realizados de forma manual. La sensorización del mismo surge a partir de la necesidad de comprobar si los posicionamientos se están realizando de manera adecuada acorde con lo marcado en la prescripción médica.

Este Trabajo de Fin de Grado (TFG) no mantiene relación con ninguna línea de investigación previa del Departamento de Ingeniería de Sistemas y Automática de la Universidad Politécnica de Cartagena (UPCT). Por tanto, pretendemos que sirva como punto de inicio en esta nueva línea de investigación, se espera que en un futuro el sistema TL-Hex pueda ser totalmente automatizado.

Actualmente, el sistema TL-Hex se utiliza tanto en la Unidad de Reconstrucción Ósea de adultos como la de Ortopedia Infantil, ambas dependientes del servicio de Traumatología, del Hospital de La Arrixaca, Murcia.

En este proyecto contaremos con la ayuda y asesoramiento del jefe de la unidad de Ortopedia Infantil, Dr. César Salcedo. Además de utilizar el sistema TL-Hex, el Dr. Salcedo, ha dirigido intervenciones en las cuales se utiliza un Clavo Autoexpandible Intramedular Magnético, el cual con la ayuda de un imán externo permite elongar el hueso de forma controlada [1].

1.2 Motivación

La presencia de la electrónica en el día a día es indiscutible. Surge la necesidad de su aplicación en el sector médico, ya que ofrece: mayor precisión, mejora la productividad, permite actuar en entornos complejos y facilita el trabajo diario de los médicos, por ejemplo, mediante la medida de constantes vitales del paciente en tiempo real.

La cirugía ortopédica es ideal para la aplicación de sistemas robóticos, ya que entre otras mejoras permite: resultados más fiables y reproducibles, mayor exactitud en los trabajos de superficies óseas y mayor precisión espacial. En estos casos, el hueso es tratado como un objeto fijo, simplificando el ordenador el control del sistema robótico.

Varios estudios a corto plazo demuestran la fiabilidad de los robots en aplicaciones ortopédicas. Sin embargo, en general, no hay datos publicados a largo plazo debido a cuestiones de costo, capacitación y seguridad que deben ser abordadas antes de que estén ampliamente disponibles. Por tanto, la cirugía ortopédica asistida por robot todavía está en su infancia, pero tiene potencial para transformar la forma en que se realizan los procedimientos ortopédicos en el futuro [2].

Por otro lado, cabe destacar el interés incipiente en el departamento de Ingeniería de Sistemas y Automática de la UPCT por el sistema TL-Hex. Esto se debe a que la aplicación de la ingeniería en el campo de la medicina tiene un gran potencial, sin embargo, hasta el día de hoy no se había investigado en esta rama. Finalmente, además se pretende conocer y extender el interés por este tipo de investigaciones.

Finalmente, otra motivación de la realización de este TFG es poder aplicar los principios de la ingeniería y electrónica para conocer y mejorar la vida cotidiana de los usuarios del sistema TL-Hex. Debemos tener en cuenta que pretendemos que su uso sea más sencillo tanto para el médico como para el paciente, en ese sentido, todo lo que hagamos no debe interferir en el correcto funcionamiento actual del sistema.

1.3 Descripción

Para simplificar la corrección de la deformidad, el sistema actual consta de dos partes perfectamente diferenciadas: una *hardware* y una *software* [3]. Ambas se presentan a continuación y serán estudiadas en más detalle en posteriores capítulos:

Por un lado, el hardware, como podemos ver en la Figura 1.1, está formado por dos soportes externos circulares o semicirculares fijados a los huesos por medio de cables y clavos. Estos soportes están interconectados por medio de seis actuadores mecánicos, los cuales permiten un ajuste multiplanar.

Figura 1.1 Hardware sistema TL-Hex

Las posiciones de los soportes externos pueden ajustarse de forma "rápida" o "gradual en incrementos precisos" para realizar reposicionamientos del segmento óseo sobre las tres dimensiones del espacio.

Por otro lado, el software sirve de soporte para los cirujanos durante los procesos pre-operatorios, durante la operación y luego post-operación a los efectos de control de la plataforma.

Éste software cuenta con un módulo (Figura 1.2), diseñado para facilitar el *planning* de la corrección pre-operación y post-operación que permite subir imagines de rayos-X.

Figura 1.2 Software sistema TL-Hex

El software, entre otras cosas, nos permite realizar:

- Cálculo de medidas
- Plantillas para la pre-planificación
- Entrada de datos automática en el software TL-HEX

A continuación, en la Figura 1.3 se incluye un flujograma con el fin de explicar de forma sencilla el funcionamiento del sistema [4]:

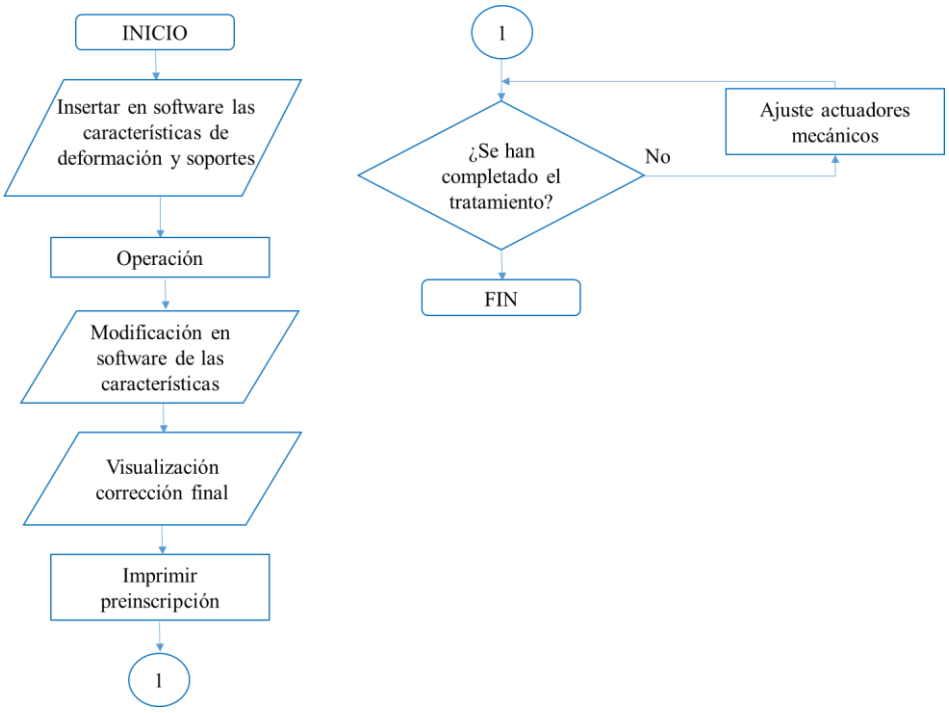


Figura 1.3 Flujograma funcionamiento sistema TL-Hex

Como queda reflejado en el flujograma anterior, el software sirve de ayuda al cirujano a la hora de la planificación para la corrección de la deformación. Permitiéndole imprimir una prescripción, la cual el paciente deberá seguir para realizar las modificaciones de las longitudes de los actuadores mecánicos. Además el cirujano realizará un seguimiento para comprobar que el paciente sigue adecuadamente la prescripción.

1.4 Objetivos Particulares del Trabajo

El objetivo principal del proyecto es conseguir la sensorización del robot TL-HEX, lo que facilitará al médico y al paciente, el posicionamiento del robot en cada instante solicitado. Esto nos lleva a la aparición de unos objetivos específicos, logros parciales que al ser alcanzados en conjunto nos permitirán alcanzar el objetivo final.

A continuación se detallan los objetivos específicos:

1. Estudiar y analizar los robots paralelos desde el punto de vista ingenieril y médico.
2. Recopilar información de sensores y requisitos necesarios para poder ser aplicados.
3. Realizar pruebas de calibración.
4. Seleccionar e instalar los sensores.
5. Comunicar inalámbricamente la información hacia una plataforma (IDE Arduino).
6. Programar una aplicación para el tratamiento de la información (IDE Arduino).
7. Programar una interfaz de visualización de la información tratada (Processing).
8. Realizar y evaluar las pruebas experimentales.

Cabe destacar que los objetivos han sufrido una pequeña variación con los indicados en la propuesta del proyecto. Esto se debe a que son resultado de requerimientos que han ido surgiendo o ampliaciones que se han incluido con el fin de mejorar el resultado final.

Capítulo 2

Estado del Arte

2.1 Robots Paralelos

Los robots paralelos se pueden definir como aquellos en los que el extremo final está unido a la base por una o más de una cadena cinemática en lazo cerrado. En este tipo de robots existen dos plataformas: una fija que forma la base y una móvil llamada plataforma móvil o efector final. Las articulaciones que conforman los brazos de estas estructuras pueden ser actuadores eléctricos o mecánicos [5]. En la Figura 2.1 podemos ver las distintas partes de un robot paralelo:

Figura 2.1 Estructura robot paralelo

En la actualidad, los robots paralelos están en constante crecimiento y son utilizados especialmente en determinadas aplicaciones en las que sus características ofrecen ventajas para resolver problemas para los que tienen limitaciones los robots serie.

A continuación se presentan algunas ventajas de los robots paralelos frente a los serie:

- Los accionamientos de potencia conectan directamente la base del robot al efector final. Debido a esto, los accionamientos de potencia sirven de elementos estructurales y actúan de manera simultánea, lo que les da la capacidad de manipular cargas muy superiores a su propio peso. Por tanto, la elevada relación carga/peso de estos mecanismos proporciona una alta eficiencia energética.
- Son mecanismos que ofrecen una alta rigidez y muy bajo peso.
- Presentan elevadas velocidades de operación, en comparación con otros tipos de estructuras robóticas.

2.2 Evolución Histórica de los Robots Paralelos

Fue el Dr. Eric Gough quien en 1947 invento la plataforma paralela más popular de todas. Diseñó un octaedro hexápodo con lados de longitud variable (Figura 2.2) como plataforma para la comprobación del comportamiento de los neumáticos de la empresa Dunlop bajo cargas aplicadas en diferentes ejes. De esta forma, intentaba simular el proceso de aterrizaje de un avión. En la actualidad existen multitud de plataformas basadas en este diseño y se conocen bajo el nombre de MAST (Multi-Axis Simulation Table) [5].

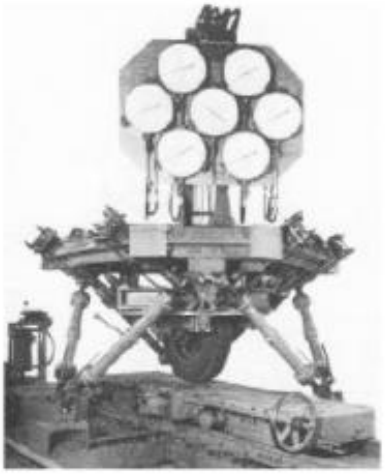


Figura 2.2 Plataforma paralela de Eric Gough

En 1965, Mr. Stewart presentó un artículo en el que describía una plataforma de movimiento de seis GDL destinada a trabajar como simulador de vuelo (Figura 2.3). Las diversas cadenas cinemáticas del mecanismo podían proveer los complejos movimientos de la cabina de un piloto. Contrariamente a la creencia general, el mecanismo de Stewart es diferente al presentado por

Gough. El artículo de Stewart tuvo y tiene una gran influencia en el mundo académico y se considera como uno de los primeros trabajos de análisis de plataformas paralelas.

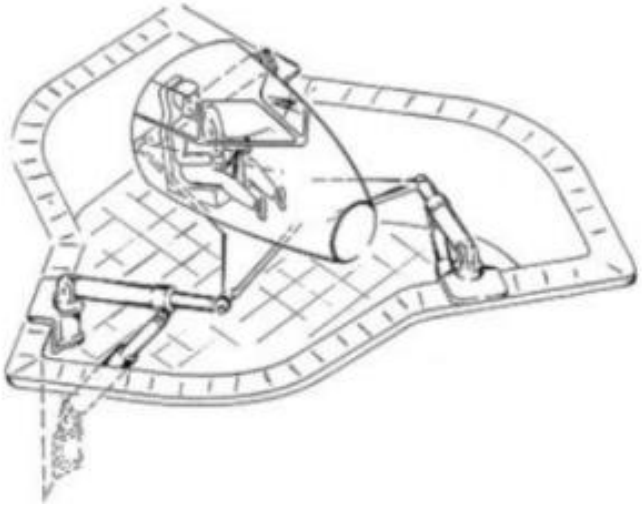


Figura 2.3 Plataforma de Stewart

Aunque ha habido una serie de estudios pioneros sobre modelos cinemáticos, fueron Fichter (1986) y Merlet (1990), los primeros que de forma rigurosa estudiaron la cinemática de los robots paralelos. Fichter derivó las ecuaciones cinemáticas de la plataforma de Stewart y formuló las ecuaciones dinámicas de una forma rudimentaria, despreciando la masa de los actuadores y la fricción de las articulaciones. Merlet consideró los aspectos de diseño de la plataforma de Stewart tratando las diferentes arquitecturas, dando las directrices para solucionar las ecuaciones cinemáticas y determinar el espacio de trabajo.

Merlet (1990) ha sido el autor que más profundamente ha estudiado las configuraciones estructurales de los robots paralelos. En principio, se pueden distinguir dos grupos, los robots planares y los espaciales.

Los mecanismos planares son aquellos en los que su movimiento se reduce al plano. Por tanto, pueden tener dos o tres grados de libertad, correspondientes al movimiento de traslación en el plano y a una rotación sobre un eje perpendicular al mismo.

Los robots espaciales son aquellos que evolucionan en todo el espacio tridimensional.

2.3 Estudios de Robots Paralelos en España

No estando aún a nivel industrial, pero en elevado estado de progreso en centros de investigación, se pueden citar los desarrollos que hacen uso de las capacidades de deformarse de los mecanismos paralelos [5]:

2.3.1 Robot TREPA

Las estructuras paralelas tienen por una parte, facilidad para realizar los movimientos complejos que requieren trepar o deslizarse por diferentes entornos; por otra, su gran capacidad de carga hace que puedan desplazar fácilmente, además de los objetos que se requieren para realizar las distintas tareas, su propio peso.

Un primer desarrollo de un robot trepador fue el TREPA I (Figura 2.4), destinado a tareas de mantenimiento y fumigación de palmeras (Figura 2.4).

Figura 2.4 Robot TREPA

El robot tiene dos anillos iguales con dispositivos de aprensión, utiliza accionamientos neumáticos y está dotado en cada anillo de un sistema de sensores a distancia en disposición radial. En su funcionamiento abarca el tronco y para instalarlo se puede abrir por su periferia.

2.3.2 Robot REMO

El robot REMO (Figura 2.5) basa su navegación en la deformación de su estructura que permite modificar la posición relativa de los impulsores y de las aletas de guiado. Esta deformación se realiza por una estructura paralela de seis GDL. Esta idea permite una gran capacidad de maniobrabilidad además (por el hecho de no tener casco y quedar reducido a su área estanca el mínimo volumen) de una posibilidad de alcanzar grandes profundidades.

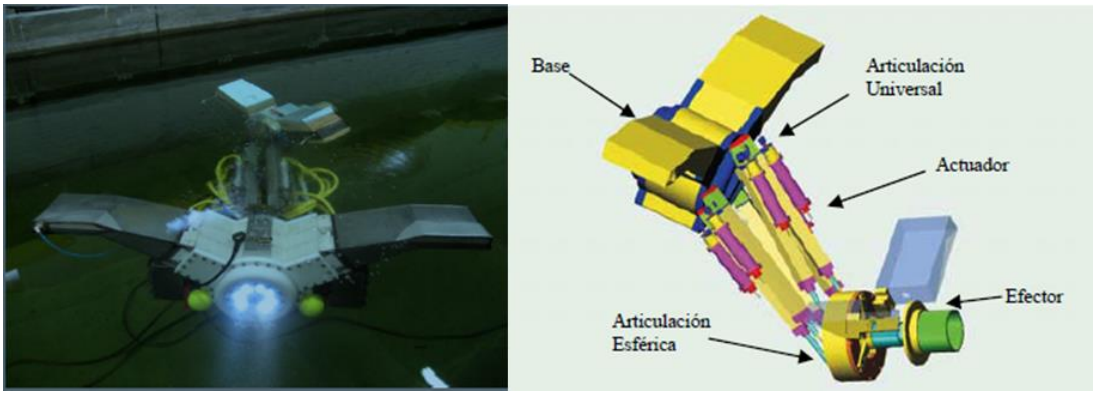


Figura 2.5 Robot REMO

2.4 Técnicas de Cirugía Ortopédica y Traumatología

En el Boletín Oficial del Estado de 7 de febrero de 2007 (número 33) se define la cirugía ortopédica y traumatológica como " Especialidad que incluye la prevención, la valoración clínica, el diagnóstico, el tratamiento quirúrgico y no quirúrgico y el seguimiento hasta el restablecimiento funcional definitivo, por los medios adecuados definidos por la «lex artis» de la comunidad de especialistas, de los procesos congénitos, traumáticos, infecciosos, tumorales, metabólicos, degenerativos y de las deformidades y trastornos funcionales adquiridos del aparato locomotor y de sus estructuras asociadas"[6].

A continuación, se realiza un breve recorrido sobre los principales avances de esta disciplina orientado concretamente a "los sistemas de fijación externa".

2.5 Evolución Histórica de los Sistemas de Fijación Externa

En 1912, Victtorio Putti fue nombrado director del Instituto de Rizzoli en Bolonia. Insistió en hacer una osteotomía con un mínimo trauma y un alargamiento gradual y controlado. Para ello diseñó un dispositivo, como el de la Figura 2.6, formado por dos agujas (una insertada en la región subtrocantérea y otra en los cóndilos) y una barra conectora para la distracción de segmentos óseos, que permitía controlar las fuerzas de tracción aplicadas. El apartado producía distracción continúa durante treinta días, consiguiendo alargamientos de hasta 8 cm. Sin embargo, no proporcionaba una adecuada estabilidad al no controlar bien las desviaciones axiales [7, 8].

Figura 2.6 Fijador externo monolateral de Putti

Una vez presentados los trabajos de Putti en 1921, Abbott y Crego modificaron la técnica en San Luis, en 1924. Desarrollaron un fijador monolateral, como el de la Figura 2.7, con pines proximales y distales a la osteotomía, que atravesaban el grosor completo de la tibia, y con una serie de varillas roscadas, a las que conectaban los cables, resultando un montaje estable.

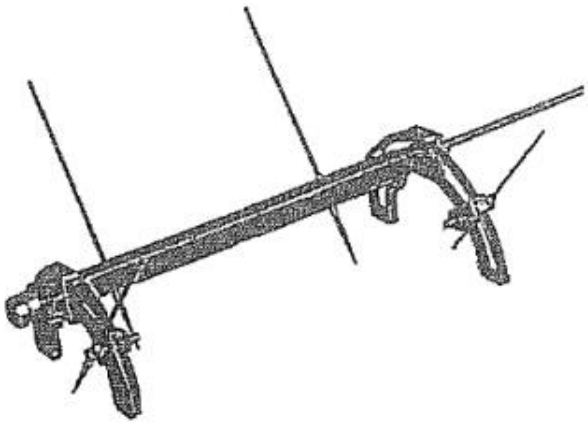


Figura 2.7 Fijador externo monolateral de Abbott y Crego

En 1930, llevaron a cabo la cirugía a setenta y tres pacientes, con múltiples complicaciones, entre las que se encontraban: deformidades en equino-valgo, luxación tibio-peronea, contracturas en flexión de la rodilla, limitación de la movilidad de la cadera, recurvatum o procurvatum de los segmentos tibiales distraídos, debilidad muscular, parálisis nerviosa, infecciones, etc…

Gavril A. Ilizarov, diseñó un fijador externo circular (Figura 2.8) que se conectaba al hueso mediante un sistema de agujas transfixiantes tensadas. Los anillos se interconectaban entre sí mediante barras roscadas. Con este sistema conseguía controlar las desviaciones en todos los planos del espacio. La primera vez que lo utilizó fue en 1951, como tratamiento de defectos óseos provocados por la tuberculosis.

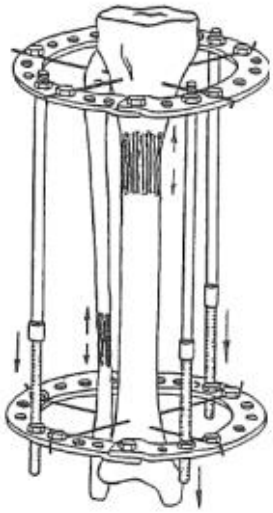


Figura 2.8 Fijador externo circular de Ilizarov

Las investigaciones que realizó Ilizarov lo llevaron al planteamiento del principio biológico de tensión- estrés que se basaba en la estimulación de la regeneración y crecimiento de los tejidos durante la distracción. Este principio lo explicó como: "*La tracción gradual sobre los tejidos vivos crea un estrés que puede estimular y mantener la regeneración y el crecimiento activo de ciertas estructuras tisulares. Los tejidos sometidos a una tracción lenta y constante son activados metabólicamente, un fenómeno caracterizado por la estimulación de las funciones celulares, tanto proliferativas como sintéticas. Estos procesos regenerativos dependen de una adecuada vascularización y del efecto estimulante del apoyo de peso*".

Actualmente, los principios del tratamiento del Ilizarov se mantienen:

- Gran calidad biológica del hueso regenerado debido al empleo de una corticotomía percutánea con mínimo daño al periostio y médula ósea.
- Periodo de latencia postoperatorio.
- Distracción total de 1 mm/día en varios pasos.

- Uso de compresión y distracción permitiendo carga completa de la extremidad.
- Uso de un fijador circular donde los fragmentos se sujetan mediante agujas de Kirschner tensadas, lo que permite al cirujano controlar los fragmentos en todos los planos y corregir deformidades.
- Desarrollo de transportes óseos para defectos del eje del hueso.
- Promover buena nutrición del hueso y la movilidad articular por medio de un sistema que permita la carga completa y la fisioterapia.

Charles Taylor, médico, y su hermano Harold, ingeniero, crearon en 1994 un software y un fijador externo hexápodo (Figura 2.9) capaz de corregir deformidades en los tres ejes del espacio, que se conoce actualmente como "Plataforma espacial de Taylor" Al año siguiente, se llevó acabo la primera operación utilizando este sistema y en 1997 se patentó el dispositivo [9].

Figura 2.9 Plataforma espacial de Taylor

2.6 Otras Aplicaciones de la Plataforma de Stewart en Medicina

2.6.1 "Tobillo de Rutgers"

El "Tobillo de Rutgers" es un componente de tele-rehabilitación con retroalimentación de fuerza virtual. A través de este sistema, los pacientes son capaces de hacer ejercicios en casa para la rehabilitación de la movilidad del tobillo mientras son supervisados remotamente por un terapeuta. El sistema genera fuerzas en seis GDL en respuesta a ejercicios basados en la realidad virtual que se ejecutan en un PC host.

La plataforma Stewart utiliza cilindros neumáticos de doble efecto, potenciómetros lineales como sensores de posición y un sensor de fuerza [10]. En la figura 2.10 podemos apreciar los distintos componentes mencionados.

Figura 2.10 Tobillo de Rutgers

2.6.2 Hexápodos en Miniatura

Un hexápodo en miniatura con seis GDL se utiliza como asistencia de guía para cirugía espinal. El sistema montado en hueso, llamado "SpineAssist", guía al cirujano para obtener la máxima precisión al colocar implantes destinados a estabilizar las vértebras en cirugía abierta y mínimamente invasiva. Dado que el robot está rígidamente unido al paciente, no hay necesidad de un sistema de coordenadas de seguimiento [11]. En la Figura 2.11 podemos ver el robot hexápodo "SpineAssist":

Figura 2.11 SpineAssist

Capítulo 3

Robot TL-Hex

3.1 Descripción

Como se mencionó anteriormente, en 1951 el profesor Gvril Ilizarov en Kurgán (Rusia), introdujo unos nuevos aparatos de fijación externa y técnicas para reducción de fracturas, alargamiento de miembros y corrección de deformaciones. Éstas técnicas revolucionaron la gestión de muchos problemas de reconstrucción previos. El aparato del profesor Ilizarov ha experimentado muchas modificaciones a lo largo de estos últimos sesenta años. Una de estas modificaciones es el "TrueLock Ring Fixation System" desarrollado en TSRHC en Dallas, Texas. Aunque hay mejoras significativas comparadas con el aparato original, el "TrueLock Ring Fixation System" conserva la mayoría de los principios originales y metodologías del Profesor Ilizarov [12].

El TL-Hex es un sistema hexápodo, diseñado en el hospital TSRHC, como un módulo tridimensional de reposición de segmento óseo para mejorar el sistema TrueLock desarrollado previamente. El sistema consiste en una serie de soportes externos circulares y semi-circulares afianzados a los huesos mediante tornillos óseos o agujas de Kirschner, todo ello interconectado mediante seis actuadores mecánicos de longitud variable. Esto permite un ajuste multi-planar de los soportes circulares e indirectamente de los segmentos óseos.

Este robot está basado en un robot paralelo, en particular, en una Plataforma de Stewart. Está clasificado como un robot espacial y dispone de una plataforma con una configuración 6-6, lo que significa que tiene seis puntos distintos de anclaje en el soporte circular inferior y seis puntos en el soporte circular superior. Dichos mecanismos pueden simular movimientos con seis GDL, esto es, tres movimientos lineales respecto a los ejes x, y, z (lateral, longitudinal y vertical) junto con otros tres rotacionales alrededor de los mismos ejes [13].

3.2 Elementos Constitutivos

En los siguientes apartados veremos de forma detallada los distintos elementos que componen el hardware del sistema TL-Hex así como sus características más importantes [12]:

3.2.1 Soportes Externos

Todos los soportes externos y placas de pie del TL-Hex están fabricados en aluminio y acero inoxidable, por tanto, son: ligeros, fuertes y muy estables. Tienen un espesor de 9,5 mm y están disponibles en una gran variedad de tamaños para permitir construcciones que se ajusten a las distintas situaciones clínicas.

Como se puede ver en la Figura 3.1, cada soporte y placa de pie tienen salientes angulares con dos orificios de montaje a los lados para acoplar los actuadores y un único tornillo de bloqueo para asegurarlos en su lugar.

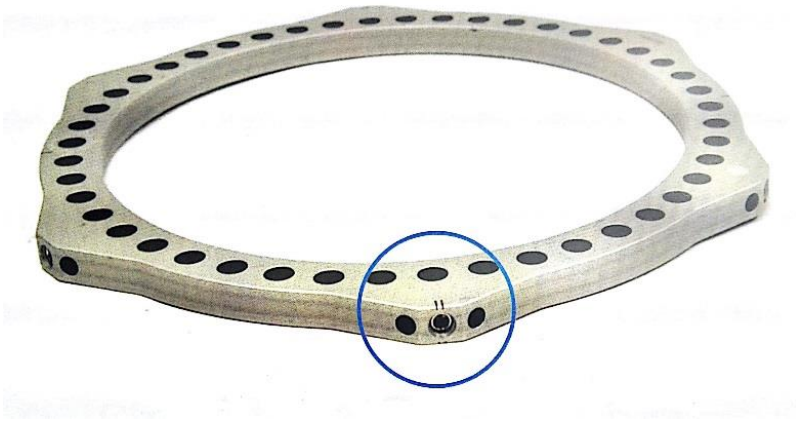


Figura 3.1 Saliente angular de soporte circular externo

Hay dos salientes angulares opuestos marcados con una doble línea en cada soporte circular, indicando los salientes anteroposterior (ver **ANEXO I**). Mientras que en los soportes 5/8, está marcado el saliente anterior. La orientación exacta de estos salientes es muy importante tanto en el montaje como para la correcta utilización del software. Además, cada soporte circular tiene dos líneas orientadas 90º relativas a los salientes anteroposterior para indicar la orientación mediolateral, lo que permite simplificar la alineación del marco de montaje.

En la Figura 3.2 podemos ver los salientes anteroposteriores (marcados en azul) con una doble línea y las marcas mediolaterales (marcados en rojo) con una línea.

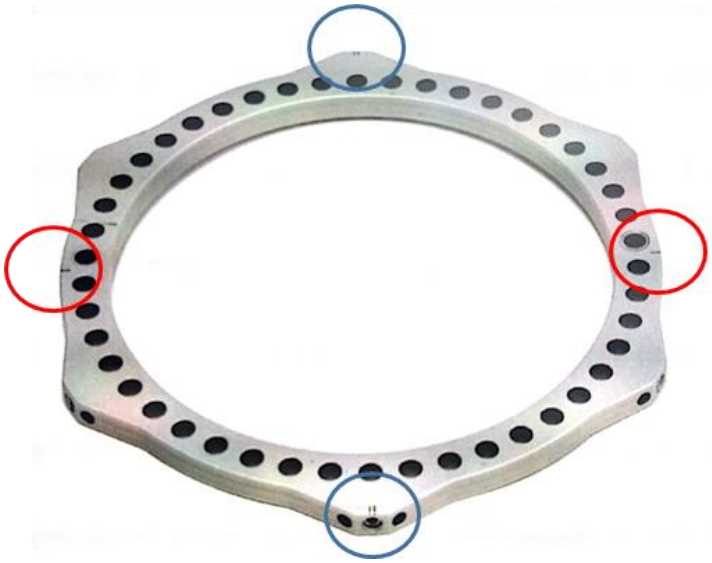


Figura 3.2 Salientes angulares anteroposteriores y marcas mediolaterales

3.2.2 Soportes Circulares

Los soportes circulares del TL-HEX están disponibles en 10 tamaños con diámetros internos de 100 mm, 120 mm, 140 mm, 160 mm, 180 mm, 200 mm, 220 mm, 240 mm, 280 mm y 300 mm.

Cada soporte tiene seis salientes angulares, lo que permite incorporar hasta 12 actuadores.

3.2.3 Soportes Circulares 5/8 y 3/8

Los soportes 5/8 y 3/8 están disponibles con los mismos diámetros internos que los soportes circulares. Siguiendo la Figura 3.3, cada soporte 5/8 tiene cinco salientes angulares, lo que permite incorporar hasta 10 actuadores.

Figura 3.3 Soporte circular 5/8

También se pueden obtener soportes circulares enganchando un anillo 3/8 a otro anillo de 5/8 usando dos tornillos y tuercas.

3.2.4 Placa de Pie

La placa de pie (Figura 3.4) tiene una base ancha, dos filas de agujeros de fijación y dos largas áreas de montaje para elementos adicionales, lo que permite mayor versatilidad para la fijación de cables y conexión de elementos adjuntos.

Está disponible en 6 tamaños, entre 120 mm y 220 mm. Además tiene tres salientes angulares y dos conjuntos de marcas de cuadrante, que encajan con las marcas encontradas en los soportes circulares.

Figura 3.4 Placa de pie

3.2.5 Actuadores Mecánicos “Struts”

Como puede verse en la Figura 3.5, los actuadores consisten en dos tubos telescópicos de aluminio, uno exterior (A) y uno interior (B), los cuales pueden bloquearse juntos en varias longitudes usando el perno de bloqueo lateral (C) y una arandela de sujeción (D).

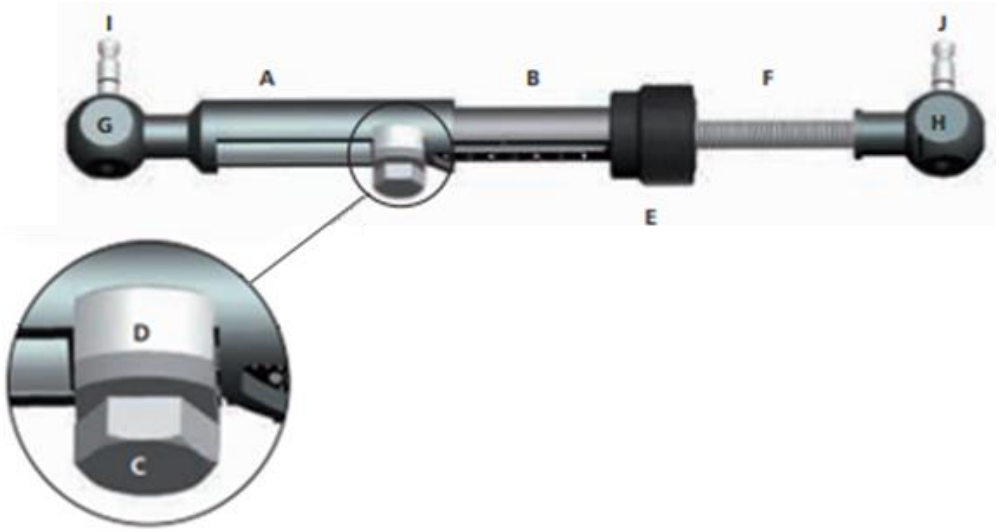


Figura 3.5 Actuador mecánico "Strut"

El tubo interior está unido a un cargador de muelles, alojado en un nudo de color negro ajustable (E), el cual va unido a una varilla roscada (F) de manera que la varilla se traslada respecto al tubo interior cuando el nudo es girado. Cada actuador tiene dos articulaciones (G) y (H), una en la base del tubo externo y la otra en el final de la varilla roscada. Cada articulación tiene un perno de montaje (I) y (J), los cuales pueden ser insertados en los orificios de montaje del soporte externo y mantenerse en su lugar con el tornillo de bloqueo.

Los actuadores mecánicos están disponibles en tres tamaños estándar (corto, medio y largo), existiendo un tamaño para casos especiales (ultracorto) permitiendo un ajuste desde los 45 mm hasta la máxima extensión de 318 mm. En la Tabla 3.1 podemos encontrar un resumen de los mismos:

Número Pieza	Descripción	Longitud Mínima (mm)	Longitud Máxima (mm)
50-10100	Ultracorto	45	101
50-10200	Corto	92	122
50-10300	Medio	114	184
50-10400	Largo	158	318

Tabla 3.1 Características actuadores mecánicos

Las longitudes de los actuadores pueden modificarse de forma "rápida" o "gradual en incrementos precisos" para realizar reposicionamientos del segmento óseo sobre las tres dimensiones del espacio.

El ajuste "rápido" se consigue aflojando el perno de bloqueo lateral, deslizando el tubo interno respecto al externo hasta la longitud deseada y apretando el perno de bloqueo. Este ajuste se indica mediante una marca naranja en el tubo externo (Figura 3.6), realizándose en incrementos de 1 mm.

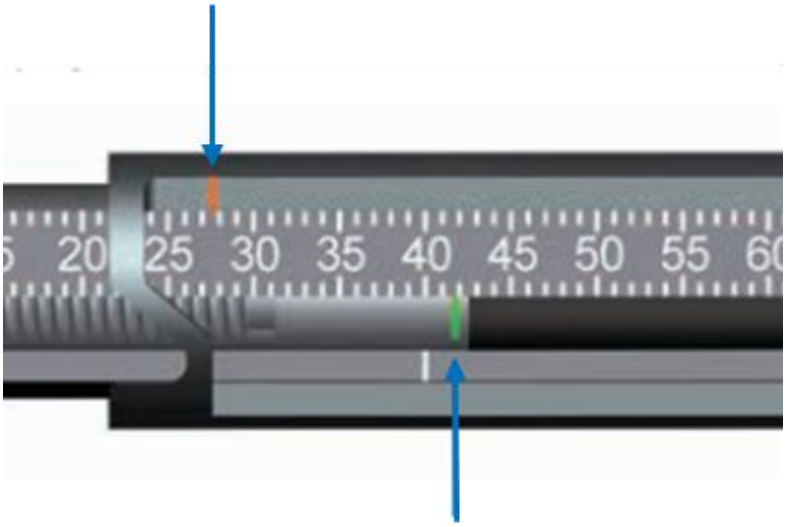


Figura 3.6 Marca de ajuste rápido (naranja) y marca de ajuste gradual (verde)

El ajuste "gradual en incrementos precisos" se consigue tirando y rotando el nudo ajustable resultado un notable "click" en cada incremento de 0,5 mm (Figura 3.7). Una flecha de referencia con un (+) localizada en la base del nudo de ajuste indica la dirección de giro para realizar el alargamiento del actuador. El ajuste gradual está indicado mediante la línea marcada en verde al final de la varilla roscada (Figura 3.6).

Figura 3.7 Ajuste gradual de los actuadores mecánicos

Hay dos tipos de clips de aluminio intercambiables: clips numerados para indicar el número del actuador y otros para indicar la dirección del ajuste gradual. Los clips se ponen en los actuadores y permanecen en el mismo lugar durante todo el tratamiento.

Los clips numerados, como los de la Figura 3.8, van desde 1 hasta 6 y un código de colores con rojo (1), naranja (2), amarillo (3), verde (4), azul (5), lila (6). Normalmente se unen a la ranura del final del tubo externo.

Figura 3.8 Clips numerados

Los clips direccionales, como los de la Figura 3.9, son universales para todos los actuadores. Tienen una flecha indicando en qué dirección se debe girar el nudo ajustable para conseguir el ajuste gradual deseado. Normalmente se unen a la ranura del extremo del actuador, después de la cirugía acorde con la prescripción.

Figura 3.9 Clips direccionales

3.2.6 Pasos a seguir en el montaje

A continuación se enumeran los pasos a seguir para el correcto ensamblaje del sistema:

1. Seleccionar dos soportes externos circulares, seis actuadores y seis clips numerados.
2. Utilizando una llave Allen, confirmar que los tornillos de bloqueo de los soportes se pueden retroceder hasta los topes.
3. Introducir los clips numerados en la ranura del final del tubo externo.
4. Posicionar el soporte circular proximal con el saliente marcado con doble línea hacia el cirujano, en él inserte el actuador #1 en el orificio de montaje izquierdo del saliente.
5. Confirmar que la línea de inserción de profundidad del perno queda completamente ocultada en el orificio de montaje.
6. Mientras se aguanta el actuador #1 en su lugar, insertar el perno de montaje del actuador #2 en el orificio derecho del mismo saliente, siguiendo la Figura 3.10:

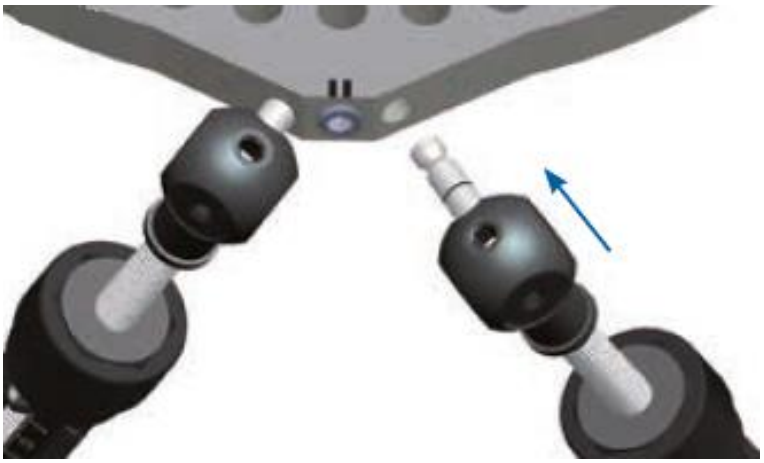


Figura 3.10 Inserción de actuadores en los orificios de montaje

7. Usando una llave Allen, ajustar parcialmente el tornillo de fijación para que se mantengan los actuadores en su sitio.
8. Girando hacia el sentido contrario de las agujas del reloj y saltándose un saliente, insertar los actuadores #3 y #4 siguiendo los pasos anteriores.
9. Repetir los ocho pasos anteriores para insertar los actuadores #5 y #6.
10. Confirmar que todos los actuadores están secuencialmente posicionados en sentido contrario de las agujas del reloj alrededor del soporte proximal.
11. Alinear el saliente de orientación del soporte distal con el saliente de orientación del soporte proximal.

12. Insertar los pernos de montaje de los extremos opuestos de los actuadores #2 y #3 en los orificios de montaje del siguiente saliente del soporte distal situado a la derecha del saliente del soporte proximal.
13. Siguiendo el mismo procedimiento que para el otro soporte, insertar el resto de actuadores.
14. Comprobar que el montaje final coincide con el de la Figura 3.11:

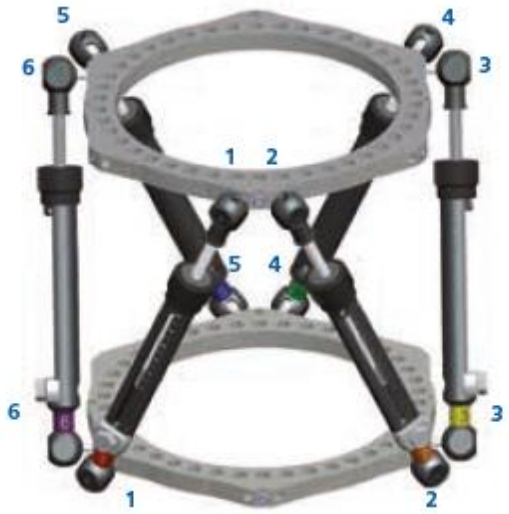


Figura 3.11 Montaje final TL-Hex

3.3 Patologías y Enfermedades

El robot TL-Hex se usa para corregir deformidades de los huesos causadas por una gran variedad de patologías y enfermedades tanto en niños como en adultos. A continuación se exponen algunos de los principales casos donde se utiliza [14]:

3.3.1 Deformaciones Congénitas

En los párrafos siguientes se mencionan algunas deformidades congénitas donde se utiliza el TL-Hex.

3.3.1.1 Enfermedad de Blount

La enfermedad de Blount, también conocida como "tibia vara", es un trastorno del crecimiento de la tibia provocando en la pierna un ángulo hacia adentro, como el de la Figura 3.12, en niños pequeños y adolescentes.

A diferencia de las piernas arqueadas o genu varo, que tienden a corregirse a medida que el niño crece, la enfermedad de Blount tiene un curso progresivo y empeora. Puede ocasionar un arqueamiento severo de las piernas y puede afectar a una o ambas piernas.

Figura 3.12 Enfermedad de Blount o Tibia Vara

3.3.1.2 Trastorno del Pie Zambo

El pie zambo, también conocido como pie equinovaro, es un defecto de nacimiento en el que el pie se encuentra invertido hacia dentro y hacia abajo (Figura 3.13). Cerca de la mitad de los casos son bilaterales, es decir, afecta a ambos pies.

Figura 3.13 Pie Zambo

3.3.1.3 Enfermedad de Raquitismo

El raquitismo es una condición que suaviza y debilita los huesos en los niños, debido a niveles bajos de calcio y fósforo en la sangre.

3.3.1.4 Genu Varo

El “Genu Varo” es una deformidad frecuente de las rodillas en los niños (Figura 3.14). Hasta los dos años de edad, puede producirse una doblez indolora en ambos lados y con frecuencia se arregla con el tiempo.

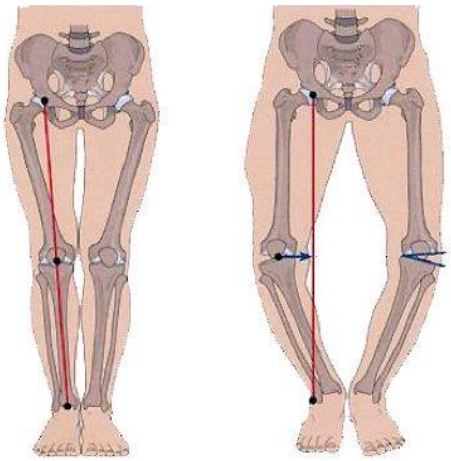


Figura 3.14 Genu Varo

3.3.1.5 Genu Valgo

El “Genu Valgo” es una deformidad caracterizada porque el muslo y la pierna se encuentran desviados, en el plano frontal, de tal manera que las rodillas se aproximan hacia la línea media, es decir, los talones de los pies están separados y las rodillas juntas (Figura 3.15).

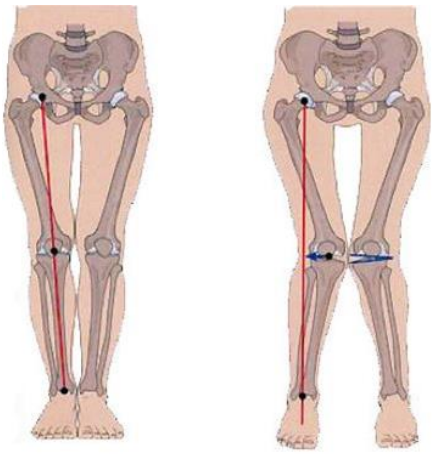


Figura 3.15 Genu Valgo

3.3.1.6 Artrogriposis Múltiple Congénita

Trastorno presente en el momento del nacimiento en el que los músculos y los tendones se acortan, limitando el movimiento articular, por lo que las extremidades pueden quedar "pegadas" en una posición.

Las extremidades del niño a menudo son delgadas, con músculos debilitados y articulaciones anormales que dan como resultado, entre otras, contracturas de la rodilla y enfermedades del pie.

3.3.2 Enfermedades del Pie y Tobillo

En los párrafos siguientes se mencionan afecciones de Pie y Tobillo donde se utiliza el TL-Hex.

3.3.2.1 Enfermedad de Charcot-Marie-Tooth

Grupo de trastornos neurológicos heredados que afectan a los nervios periféricos, es decir, aquellos que se encuentran fuera del cerebro y la médula espinal, y regulan las capacidades motoras y sensoriales de las extremidades.

Se pueden presentar deformidades en los pies, y las piernas pueden tener una apariencia de "botella de champagne invertida" a medida que se pierde masa muscular.

3.3.2.2 Pie Cavo

El pie cavo es una anomalía del pie arqueado. Las personas con esta condición colocan demasiado peso y esfuerzo sobre el metatarso y talón del pie cuando están de pie o caminando (Figura 3.16). El desarrollo de esta condición puede suceder a cualquier edad.

Figura 3.16 Pie Cavo

3.3.2.3 Pie plano

El arco del pie se baja o aplana, de forma que la superficie de la planta tiene contacto con el suelo (Figura 3.17). Puede desarrollarse como resultado de cambios degenerativos en articulaciones y/o ligamentos; esto tiende a suceder en pacientes mayores y personas que son muy pesadas.

Figura 3.17 Pie Plano

3.3.2.4 Artritis de tobillo

La artritis del tobillo es un endurecimiento de la articulación que restringe la movilidad y el uso.

El desgaste con el tiempo y la hinchazón en la articulación, así como las condiciones de trauma e inmunológicas, pueden provocar rigidez, dolor y una capacidad reducida para moverse.

3.3.3 Trauma y Deformidad Postraumática

En los párrafos siguientes se mencionan algunas afecciones traumáticas y deformidades postraumáticas donde se utiliza el TL-Hex.

3.3.3.1 Tratamientos de Fractura

Las fracturas abiertas con daño severo de los tejidos blandos y las fracturas complejas con altos niveles de inestabilidad requieren estabilización mecánica para permitir la alineación de los fragmentos mientras se produce la cicatrización.

3.3.3.2 Correcciones de Deformidades Postraumáticas

Las condiciones postraumáticas incluyen fracturas mal unidas, no unidas o infectadas y pueden dar como resultado una anatomía anormal que requiere corrección o tratamiento adicional.

Capítulo 4

Software TL-Hex

4.1 Descripción

El software, empezando por la fractura o por la deformación, es capaz de calcular una prescripción para que el cirujano la revise y la apruebe, indicando la dirección y el ajuste diario necesario en longitud para cada actuador con el fin de lograr los objetivos del tratamiento [4].

En los siguientes apartados se pretende realizar una breve introducción al uso del software con el objetivo de entender mejor el funcionamiento conjunto del sistema TL-Hex. No se hará una descripción detallada del mismo, ya que no está dentro de los objetivos del presente trabajo.

Toda la información está disponible y detallada en la "Guía de Software para el Usuario" en el siguiente enlace:

http://web.orthofix.com

4.2 Orientación del Hardware

Las siguientes reglas se aplican para cada montaje del TL-Hex con el objetivo de ser consistentes con las representaciones del software:

4.2.1 Soportes Circulares y Soportes Circulares 5/8

- El saliente angular de orientación del soporte siempre se coloca en la cara anterior de la extremidad.
- El saliente angular de orientación es del que salen los actuadores 1 y 2.
- Siempre está en la base proximal, independientemente de cual sea el segmento de referencia seleccionado.

Podemos comprobar lo enunciado en los puntos anteriores siguiendo la Figura 4.1:

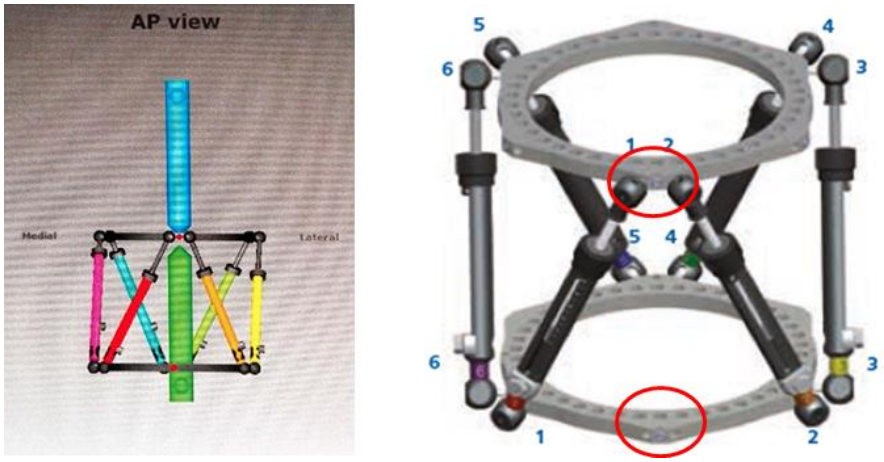


Figura 4.1 Orientación hardware

4.2.2 Soportes Circulares 5/8 abiertos medialmente

- En este caso, cuando no hay deformación rotacional, de acuerdo con el software, el saliente angular de orientación es aquel más cercano en el sentido contrario de las agujas desde del punto anterior de la extremidad.
- Hacer esta rotación siempre en sentido contrario de las agujas del reloj para ambas piernas, para la pierna izquierda hay un valor de rotación "externa" y para la pierna derecha hay un valor de rotación "interna".
- Los valores de las rotaciones se calculan automáticamente.

Podemos comprobar lo enunciado en los puntos anteriores siguiendo la Figura 4.2:

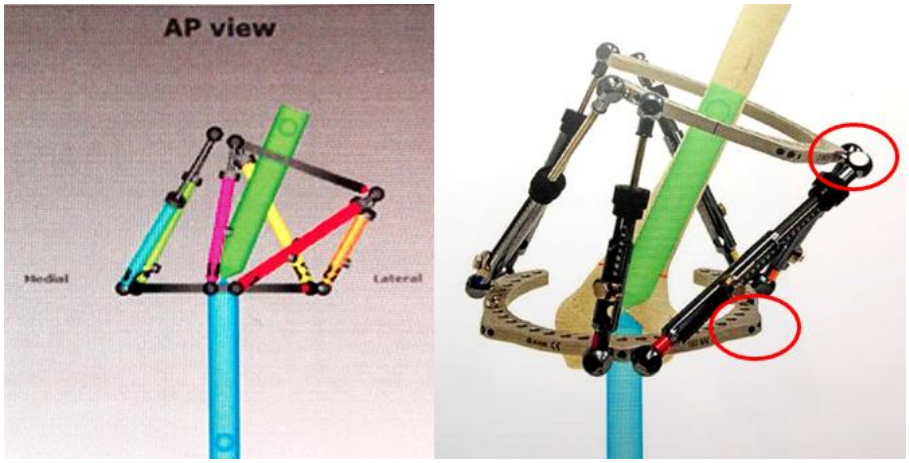


Figura 4.2 Orientación soportes circulares 5/8 abiertos medialmente

4.3 Nomenclatura

Segmento de referencia: Proximal o Distal

En la descripción de la fractura o deformación, uno de los segmentos del hueso es definido como "segmento de referencia" y el otro como el "segmento en movimiento". En el software, el "segmento de referencia" es el indicado en azul, mientras que el "segmento en movimiento" es el verde. El cirujano puede elegir como referencia, bien el segmento proximal o bien el distal.

Elegir la referencia "Proximal" significa:

- El hardware y la deformidad se orientan relativas a los ejes del segmento proximal.
- Los parámetros de deformación (cómo se ve el hueso) se deben describir en este sentido.
- La traslación y la rotación del segmento distal se describen en relación al segmento proximal.

Elegir la referencia "Distal" significa:

- El hardware y la deformidad se orientan relativas a los ejes del segmento distal.
- Los parámetros de deformación (cómo se ve el hueso) se deben describir en este sentido.
- La traslación y la rotación del segmento proximal se describen en relación al segmento distal.

En el caso de elegir el segmento de referencia como el distal, la traslación medial del segmento de hueso distal se describirá como traslación lateral porque el segmento proximal se estará

trasladando en relación al segmento distal (Figura 4.3). Para el caso de pies, el software sigue las mismas reglas de representación:

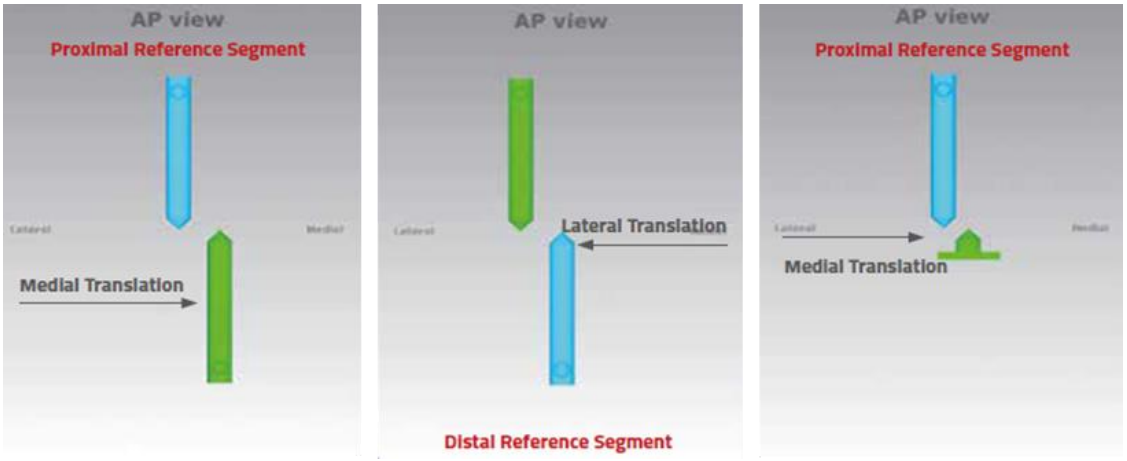


Figura 4.3 Orientación según el segmento de referencia

Es importante destacar que cambiar la referencia de distal a proximal cambiará la dirección de traslación y rotación en los planos, pero no cambiará los parámetros de longitud o angulares porque estos son matemáticamente independientes del punto de referencia. Para minimizar errores de medida en los rayos-x, el segmento más corto se debe escoger como el de referencia.

A continuación, en la Figura 4.4 se presentan tres imágenes del software. La primera es la vista AP (Anteroposterior), los rayos pasan en sentido anterior hacia el posterior de la extremidad. La segunda es la vista ML (Mediolateral), los rayos pasan en sentido medial hacia el lateral [15]. La tercera es la vista Axial que representa la vista que tendríamos si miramos desde arriba o desde debajo de la extremidad desde el segmento de referencia.

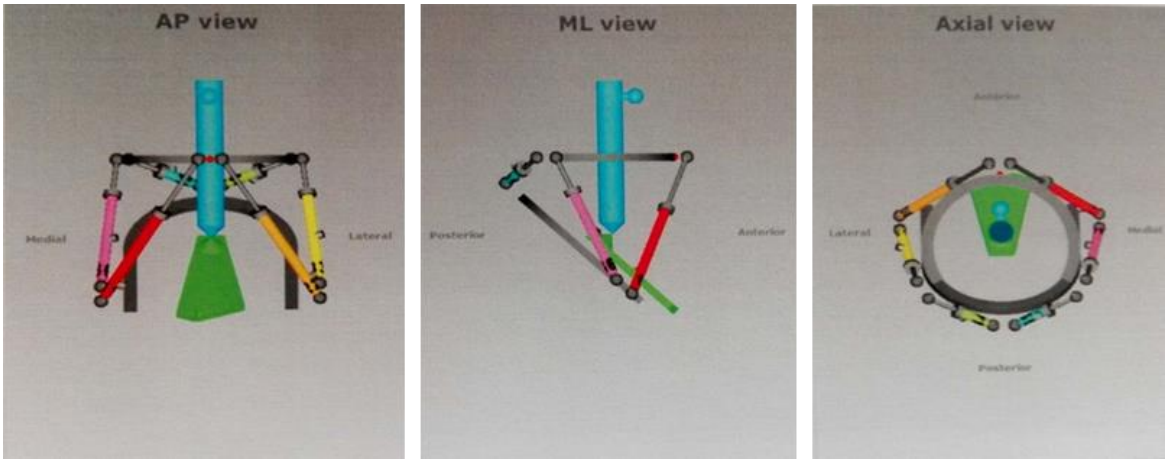


Figura 4.4 Vistas del Software TL-Hex

4.4 Planificación de Casos

4.4.1 Datos Caso

Los "datos del caso" (Figura 4.5) incluye el documento de identidad del paciente, número de caso, nombre del caso, lado anatómico (izquierda o derecha), Tipo de hueso (hueso largo o pie y tobillo), notas y día.

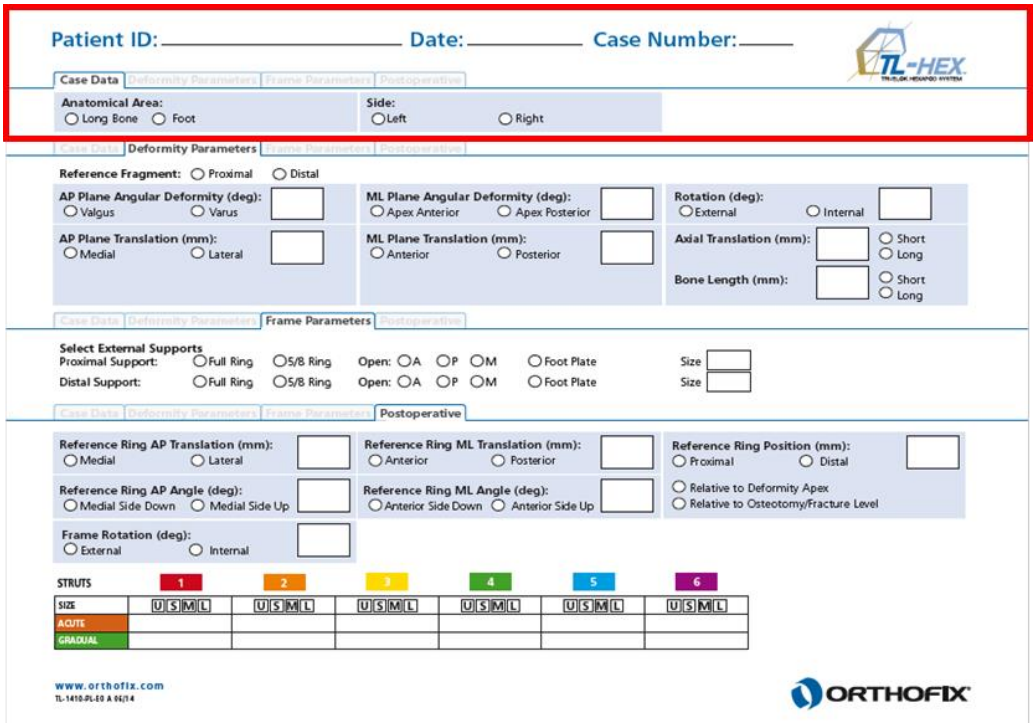


Figura 4.5 Datos del caso

El software necesita tres bloques de parámetros para calcular la corrección, los cuales podemos ver en la Tabla 4.1:

	Parámetros de Deformación	**Parámetros de Hardware**	**Parámetros de montaje**
Definición	Cómo es la deformación o fractura	El tamaño y la forma de los soportes que se van a usar	Dónde está el punto de referencia del soporte respecto a la fractura o deformación
Fuente	Rayos x AP y ML, valoración médica	Soportes (circulares o 5/8 más 3/8), soporte de pie, soportes 5/8 (en mm)	Posición, traslación y ángulo de los soportes
Pantalla Software	Parámetros de deformación	Parámetros de hardware	Montaje preopertario, parámetros sección o postopertario

Tabla 4.1 Parámetros necesarios para calcular corrección

4.4.2 Parámetros Deformación

A continuación, podemos ver los "Parámetros deformación" en la Figura 4.6. Lo primero es elegir el segmento de referencia. El cirujano es libre de elegir el segmento proximal o distal en función del escenario clínico.

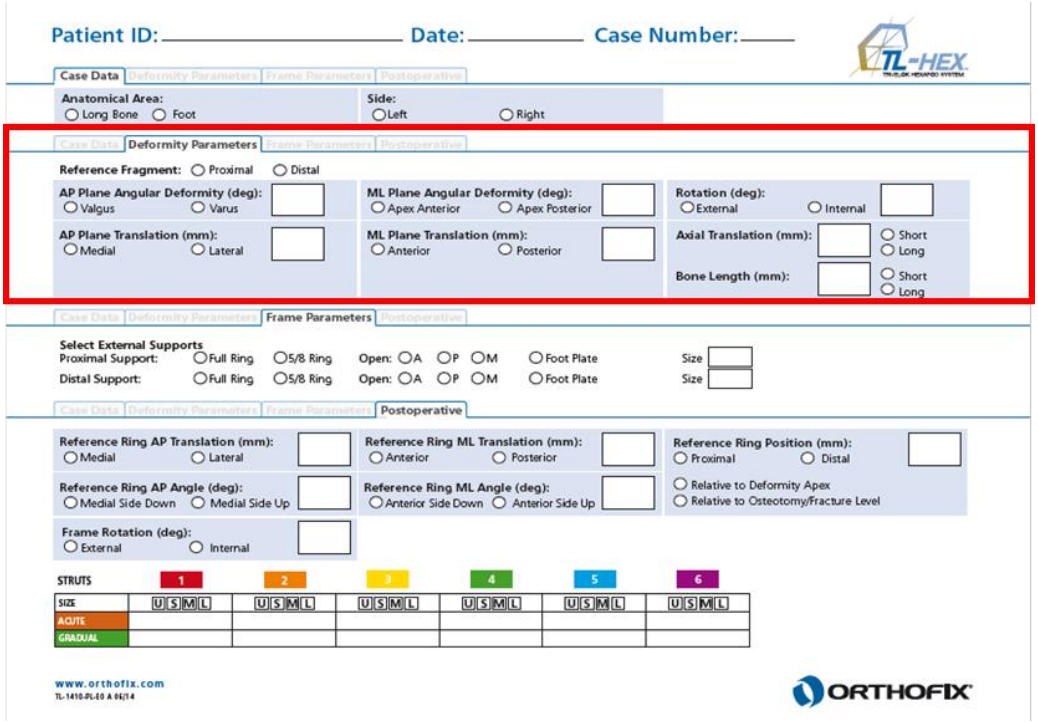


Figura 4.6 Parámetros de deformación

Deformación Angular (grados) plano AP: En el plano Coronal (Frontal) la deformación angular puede ser varus o valgus, dependiendo si el segmento distal esta desviado medial o lateralmente del segmento de referencia.

Traslación (mm) plano AP: En el plano Coronal, la traslación puede ser medial o lateral.

Deformación Angular (grados) plano ML: La deformación angular en el plano Sagital se describe como "apex anterior" o "apex posterior"

Traslación (mm) plano ML: En el plano Sagital, la traslación puede ser anterior o posterior.

Rotación (grados): La deformación angular en cualquier plano horizontal se puede describir como rotación interna o externa.

Traslación longitudinal en los ejes de la extremidad (mm): se describe en milímetros como corto o largo. La opción de corto se utiliza cuando el segmento del hueso en movimiento se traslada acercándose al segmento de referencia. La opción de largo se utiliza cuando el segmento del hueso en movimiento se traslada alejándose del segmento de referencia.

Longitud hueso (mm): se puede indicar si el hueso es largo o corto, además del desfase entre los segmentos (Figura 4.7).

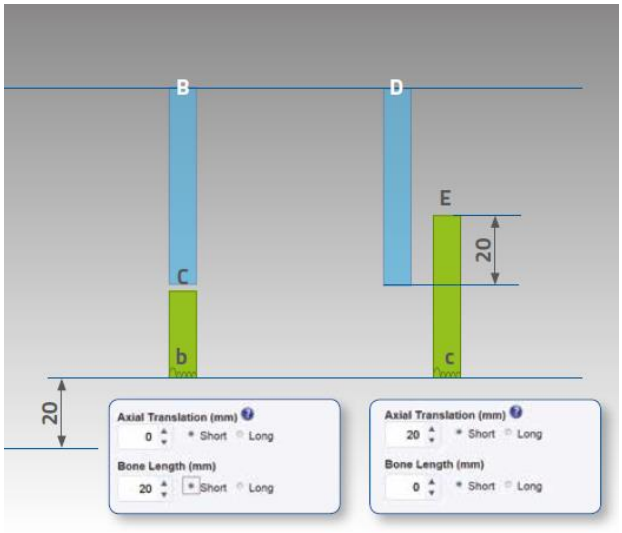


Figura 4.7 Longitud del hueso

4.4.3 Parámetros Hardware

Consiste en dos menús desplegables (Figura 4.8), uno para cada soporte distal y proximal y una sección opcional "Parámetros de Montaje Preoperatorios" que nos muestra los parámetros por defecto.

Para cada soporte distal o proximal, podemos elegir el tipo de soporte circular y el tamaño en milímetros.

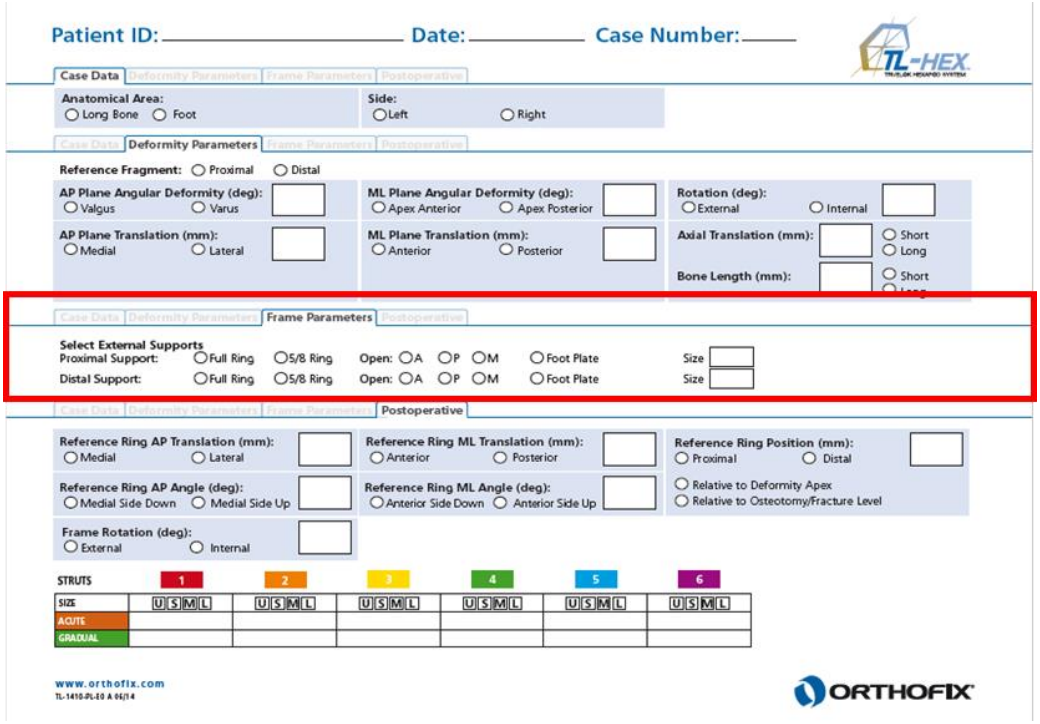


Figura 4.8 Parámetros hardware

4.4.4 Post-Operatorio

En esta pantalla (Figura 4.9) se introducen los parámetros del hardware después de la operación, con la finalidad de reproducir la posición exacta de los soportes y valores de los actuadores.

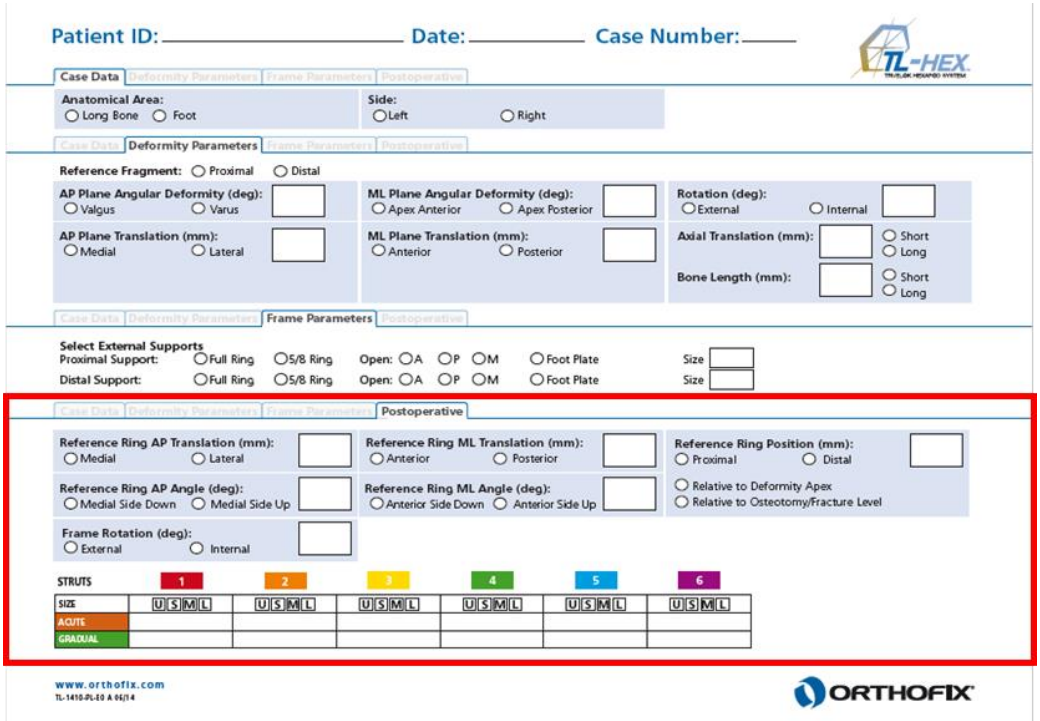


Figura 4.9 Post operatorio

Traslación del soporte de referencia (mm), vista AP: Posición del soporte de referencia en el plano coronal. Descrita como traslación medial o lateral del centro del soporte de referencia en relación al eje longitudinal del segmento de referencia del hueso.

Desviación angular del soporte de referencia (grados), vista AP: Este es el ángulo entre la orientación del soporte de referencia en el plano coronal y su proyección ortogonal al hueso de referencia en el eje longitudinal.

Traslación del soporte de referencia (mm), vista ML: Posición del soporte de referencia en el plano sagital. Descrita como traslación anterior o posterior del centro del soporte de referencia en relación al eje longitudinal del segmento de referencia del hueso.

Desviación angular del soporte de referencia (grados), vista ML: Este es el ángulo entre la orientación del soporte de referencia en el plano sagital y su proyección ortogonal al hueso de referencia en el eje longitudinal.

Posición del soporte de referencia (mm): Posición del soporte de referencia relativa al vértice de la deformación o nivel de la osteotomía o fractura. Descrito como la longitud sobre el eje longitudinal del hueso de referencia, proximal o distal desde el centro del soporte de referencia.

Rotación del soporte de referencia (grados): Este es el ángulo entre el plano sagital y el plano que pasa a través de la orientación del soporte de referencia en el plano axial.

Al final de la pantalla, se pueden introducir los parámetros de montaje de los seis actuadores. Cada actuador se describe mediante tres campos:

- Tamaño del actuador
- Longitud “Rápida” (mm) relativa a la marca naranja
- Longitud “Gradual” (mm) relativa a la marca verde

4.4.5 Corrección Final

La ventana de la Figura 4.10 nos proporciona la posición de los segmentos del hueso y del hardware al final del tratamiento. El software asume que, al final de la corrección de la deformación, los segmentos del hueso deben estar perfectamente alineados. Para el caso del pie, el software considera que el tratamiento finaliza cuando los dos segmentos están perpendiculares.

Sin embargo, el cirujano puede anular los valores por defecto e introducir los valores deseados al final del tratamiento de forma manual.

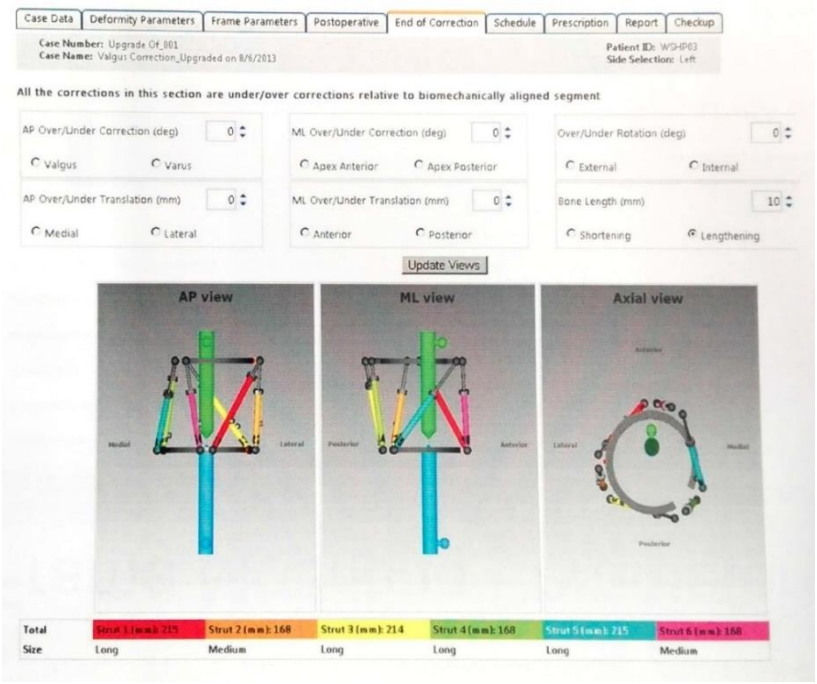


Figura 4.10 Corrección final

4.4.6 Programación

La ventana de la Figura 4.11 permite al cirujano introducir parámetros específicos sobre el movimiento del hueso durante el tratamiento. Incluyendo:

- Corrección diaria en mm/día
- Corrección angular diaria en grados/día
- Rotación diaria en grados/día
- Días de tratamiento

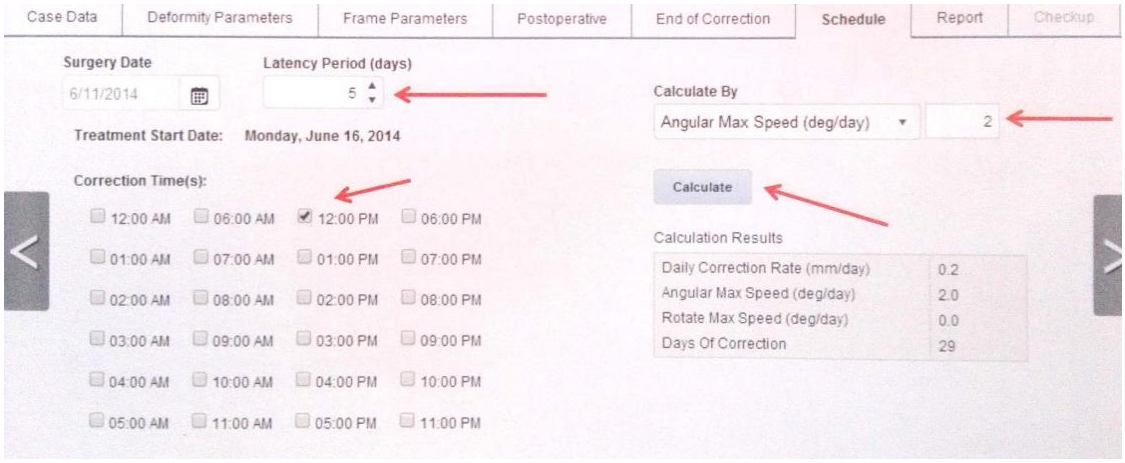


Figura 4.11 Programación

Los parámetros están correlacionados, es decir, el cirujano puede elegir introducir la "corrección angular" y la "rotación diaria" o los "días de tratamiento".

- Comienzo del tratamiento
- Aplicar alargamiento/ acortamiento primero: programa el tratamiento en dos pasos, el primero de alargamiento o acortamiento axial y el segundo para la corrección de la deformación.

4.4.7 Informe/ Prescripción

Esta ventana nos muestra la programación para el ajuste de los actuadores resultante de los parámetros introducidos en los apartados anteriores.

La opción "Seleccionar la opción de impresión" permite elegir qué tipo de PDF imprimir: Prescripción, informe o BOM (Bill of Material) Listado de Materiales.

La "Prescripción" (Figura 4.12) muestra la programación que el paciente debe seguir para realizar el ajuste de los actuadores. El ajuste de los actuadores está representado por el número de "clicks" y puede ser positivo (si la longitud del actuador aumenta) o negativo (si la longitud del actuador disminuye) [16].

Dr. test test
Hospital Name
Hospital Street
Dallas Texas US 23443
123456789
0346789000

Print Date: lunedì 29 settembre 2014
Case Number: 2
Case Name: blount's disease
Patient ID: PATIENT A
Side: Left
Bone Type: Long Bone

Page 2

			Strut Adjustment in 'CLICKS' (a)						Strut Reference Length (b)					
			RED	ORANGE	YELLOW	GREEN	BLUE	PURPLE	RED	ORANGE	YELLOW	GREEN	BLUE	PURPLE
No	Day	Date-Time	1	2	3	4	5	6	1	2	3	4	5	6
11	gio	09/10/2014 08:00	0	+1	+4	+4	+3	0	5	72	60	12	64	1
12	gio	09/10/2014 20:00	-1	+1	+4	+4	+3	0	6	72	58	10	63	1
13	ven	10/10/2014 08:00	-1	+1	+3	+4	+2	0	6	71	56	8	62	1
14	ven	10/10/2014 20:00	-1	+2	+4	+5	+3	0	6	70	54	6	60	1
15	sab	11/10/2014 08:00	-1	+1	+4	+4	+2	0	7	70	52	4	60	1
16	sab	11/10/2014 20:00	-1	+1	+3	+4	+3	0	8	69	50	2	58	1
17	dom	12/10/2014 08:00	-1	+1	+4	0	+3	-1	8	68	48	79	56	2
18	dom	12/10/2014 20:00	-1	+1	+3	+4	+2	0	8	68	47	77	56	2
19	lun	13/10/2014 08:00	-1	+1	+4	+4	+3	0	9	68	45	75	54	2
20	lun	13/10/2014 20:00	-1	+1	+4	+5	+2	-1	10	67	43	73	53	2
21	mar	14/10/2014 08:00	-2	+1	+3	+4	+3	0	10	66	42	71	52	2
22	mar	14/10/2014 20:00	-1	+2	+4	+4	+3	0	11	66	40	69	50	2
23	mer	15/10/2014 08:00	-1	+1	+3	+4	+2	-1	12	65	38	67	49	2
24	mer	15/10/2014 20:00	-1	+1	+4	+4	+3	0	12	64	36	65	48	2
25	gio	16/10/2014 08:00	-1	+1	+3	+5	+2	-1	12	64	34	62	46	3
26	gio	16/10/2014 20:00	-1	+1	+4	+4	+3	0	13	64	32	60	45	3

Figura 4.12 Prescripción

Por tanto, la dirección de las flechas de los clips direccionales se colocan siguiendo la Figura 4.13, en función de si queremos una elongación (números positivos en la prescripción), las flechas de los clips deben apuntar a la misma dirección de la flecha de referencia en el nudo negro del actuador. Mientras que si se desea acortar (números negativos en la prescripción), el clip debe colocarse con las flechas en sentido contrario a la dirección de la flecha de referencia en el nudo negro del actuador.

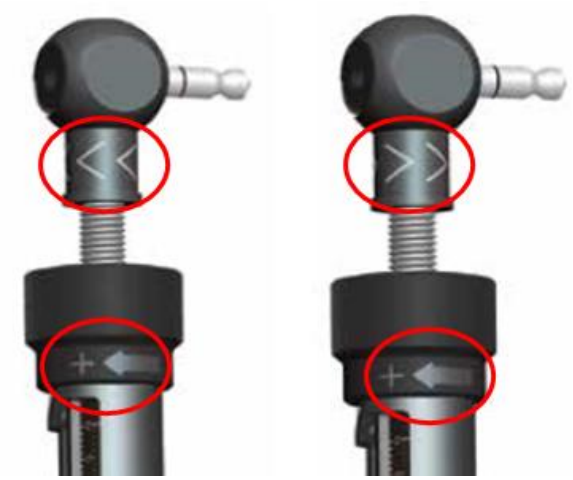


Figura 4.13 Ajuste gradual según la prescripción

Cambio en la dirección de ajuste de los actuadores

En la mayoría de los casos, la orientación de los clips direccionales es la misma durante todo el tratamiento. Pero en algunos casos con corrección de deformación rotacional, la dirección de ajuste de los actuadores puede cambiar de positivo a negativo o viceversa. En esta situación, el cirujano debe advertir al paciente y programar una visita.

Reajustar o intercambio del actuador

La fila estará sombreada de color azul cuando se requiera reajustar el actuador, o roja si se requiere un intercambio. El sombreado suave indica los días permitidos para realizar el reajuste o el intercambio; el sombreado fuerte indica el último día.

La opción “Imprimir BOM” genera una lista de materiales necesarios para cubrir el tratamiento. La opción “Imprimir Informe” genera un PDF para el cirujano, como el de la Figura 4.14, similar a la prescripción.

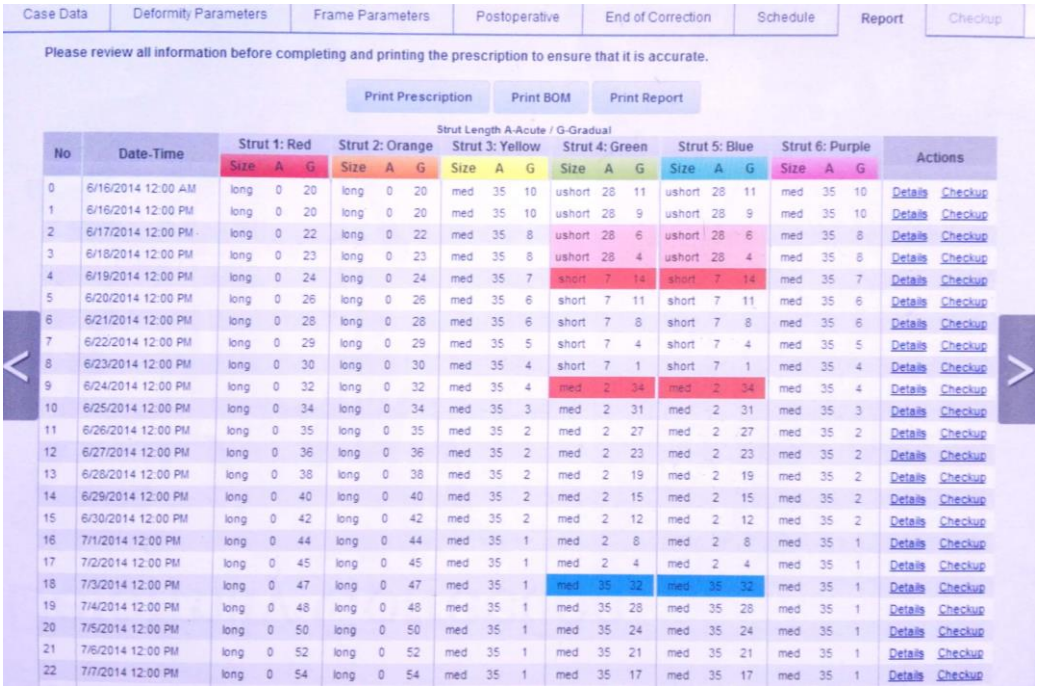


Figura 4.14 Informe

A diferencia de la prescripción, además hay una columna de "Acciones" que permite elegir: "Detalles", "Ajuste" y "Revisión".

Detalles: Se genera una ventana con las tres vistas de los segmentos del hueso junto al hardware para el día elegido, así como los valores de Rápido/Gradual de los actuadores.

Ajuste: Abre una ventana en la que podemos modificar los valores de longitud, tamaño y los valores de Rápido/Gradual del actuador seleccionado.

Revisión: Nos muestra La ventana de revisión.

4.4.8 Revisión

La ventana de la Figura 4.15 proporciona la posición de los segmentos óseos y del hardware con los correspondientes valores de ajuste de los actuadores para un día en particular. Aquí podemos "Crear un nuevo caso" si se da alguna de las siguientes situaciones:

- Cambios en los parámetros o ajuste de los actuadores.
- Reajuste o cambio de actuador que no estaba planificado.
- Se requiere una corrección residual.
- Comienzo de la siguiente fase del tratamiento.

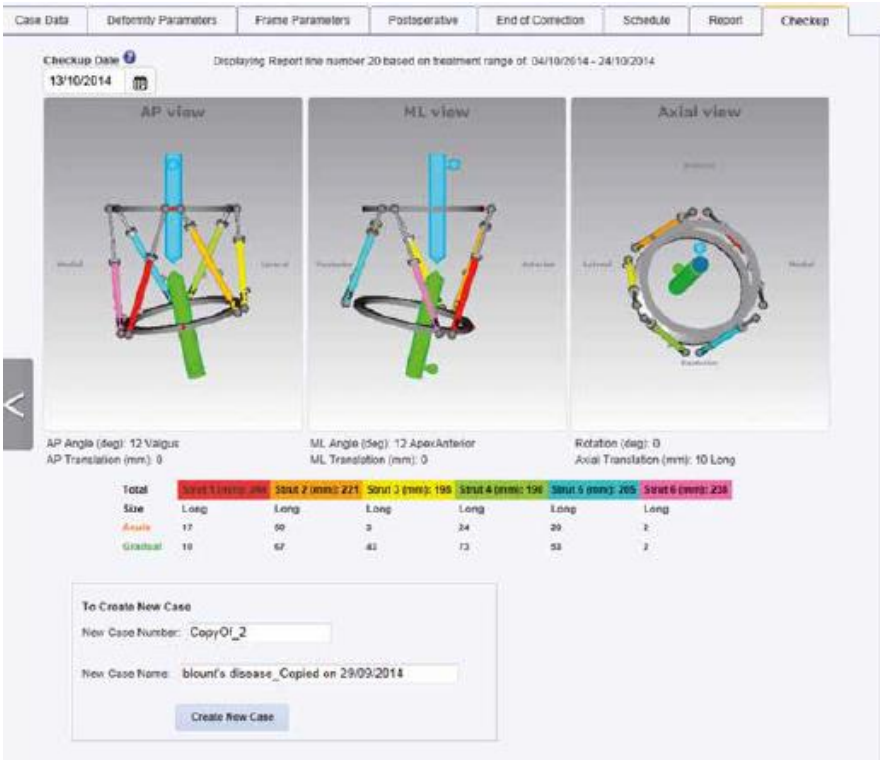


Figura 4.15 Revisión

Capítulo 5

Sensorización de los Actuadores Mecánicos

5.1 Introducción

En este capítulo nos centraremos en el análisis de los posibles sensores aptos para la sensorización del TL-Hex, así como los motivos por los que algunos han sido rechazados.

Empezaremos el capítulo enumerando los requisitos mínimos a cumplir. Seguiremos con el estudio de los sensores candidatos. Finalmente, expondremos los motivos por los que se han seleccionado sensores del tipo potenciómetro lineal.

5.2 Requisitos Mínimos

Junto al asesoramiento del Dr. Salcedo, se han fijado los siguientes requisitos:

1. **Solución de mínimo impacto**: no modificará el funcionamiento existente del actuador mecánico ni será necesario el diseño y mecanizado de uno nuevo.
2. **Precisión mínima de 1 mm**: Aunque el ajuste gradual se haga cada 0,5 mm, si miramos los valores de longitud de las tablas del actuador mecánico podemos comprobar que todas son valores enteros sin decimales. Debe ser capaz de medir longitudes entre 158 - 318 mm.
3. **Los datos admiten un retraso de presentación**: Debemos tener en cuenta que en algunos casos hará falta un software para tratamiento de datos como: promediados, cálculos de medianas, filtros paso bajo, etc…
4. **Sensores ligeros y poco voluminosos**: Éstos no pueden incorporar excesivo peso al sistema y por tanto, a la pierna del paciente.

5. **Fácil integración**: Es decir, capacidad de funcionar plenamente con la configuración actual.
6. **Seguridad**: Tanto para el paciente como para el equipo médico.
7. **Comunicación inalámbrica**: Los datos serán enviados a un PC sin necesidad de cableado.
8. **Medida del ajuste de precisión**: El ajuste rápido se suele fijar en el momento de la operación por el médico especialista. Normalmente cuando se necesita modificar el ajuste rápido es debido a un fallo y se hace necesaria la sustitución completa del actuador mecánico. Por tanto, el paciente manipula exclusivamente las longitudes graduales, las cuales serán de interés sensorizar.

A continuación se detallarán todos los sensores analizados junto con sus ventajas y desventajas. Además analizaremos qué requisitos podemos cumplir con cada sensor.

5.3 Sensores Estudiados

5.3.1 Potenciómetro Lineal

5.3.1.1 ¿Qué es y cómo funciona?

Un potenciómetro es un dispositivo conformado por dos resistencias en serie, las cuales poseen valores que pueden ser modificados por el usuario. Todos los tipos existentes comparten una característica: tienen 3 terminales (o patas) [17].

A nivel interno, la estructura de un potenciómetro se puede ver en la Figura 5.1:

Figura 5.1 Estructura de un potenciómetro

A partir del nodo que se forma entre las dos resistencias en serie tenemos un terminal, el cual normalmente será la pata del centro. Los potenciómetros que encontramos en el mercado vienen con un valor de resistencia determinado que han sido estandarizados siendo, por ejemplo 1KΩ, 5 KΩ, 10 KΩ, 50 KΩ, 100 KΩ, etc. Este valor de resistencia lo podemos medir entre los terminales 1 y 3 del potenciómetro.

A medida que movemos la perilla del potenciómetro físico, aumentamos un valor de una resistencia y reducimos el de la otra de forma tal que la suma de ambas siempre se mantendrá constante. Por eso al medir la resistencia en los terminales 1 y 3 el valor no varía, ya que la variación se da entre 1 y 2 y entre 2 y 3.

Por tanto, un potenciómetro puede funcionar como resistencia variable siempre y cuando se considere solamente 2 terminales consecutivos de los 3 que posee el dispositivo.

La capacidad de variar la resistencia entre 2 terminales y mantenerla entre sus extremos permite que los potenciómetros se utilicen como variadores de voltaje.

Esta variación de voltaje la podemos deducir a partir de la ecuación de divisor de tensión:

$$V_{out} = \frac{V_s R_2}{R_1 + R_2}$$

Donde:

- Vout es el voltaje en el nodo central
- VS es el voltaje de la fuente
- R1 y R2 son las 2 resistencias que forman el potenciómetro

5.3.1.2 Potenciómetros estudiados

En este apartado se detallan los dos tipos de potenciómetros lineales estudiados:

5.3.1.2.1 Potenciómetro deslizante

A continuación se adjunta en la Tabla 5.1 un resumen del potenciómetro elegido:

Características	Valor	Observaciones
Modelo	Alps RSAON1111	El promediado se realizará con la lectura de 1200 muestras del pin analógico de Arduino
Dimensiones (mm)	128x16x20	
Carrera (mm)	100	
Tipo	Lineal	
Resistencia	10kΩ	
Interfaz de datos	Analógica	
Pruebas		
Cantidad de datos adquiridos	100	
Precisión en la medida (mm)	1	
Coeficiente de determinación (R^2)	0,999 ≈ 1	
Filtrado	Promediado	

Tabla 5.1 Características potenciómetro deslizante

La hoja de características de este sensor puede encontrarse en la siguiente dirección:

http://www.docs-europe.electrocomponents.com

Nota: Los pines están mal numerados en la hoja de características, la conexión se realiza de la siguiente forma:

- Pin 1 se conecta a 5 V
- Pin 2 se conecta a la entrada analógica
- Pin 3 se conecta a tierra

En la Figura 5.2 se puede ver el potenciómetro deslizante Alps RSAON1111:

Figura 5.2 Potenciómetro deslizante Alps RSAON1111

En la figura 5.3 se adjunta el diagrama de conexionado necesario para utilizar este potenciómetro junto a un Arduino 1 Rev 3. Los diagramas de conexiones se realizarán utilizando el programa "Fritzing".

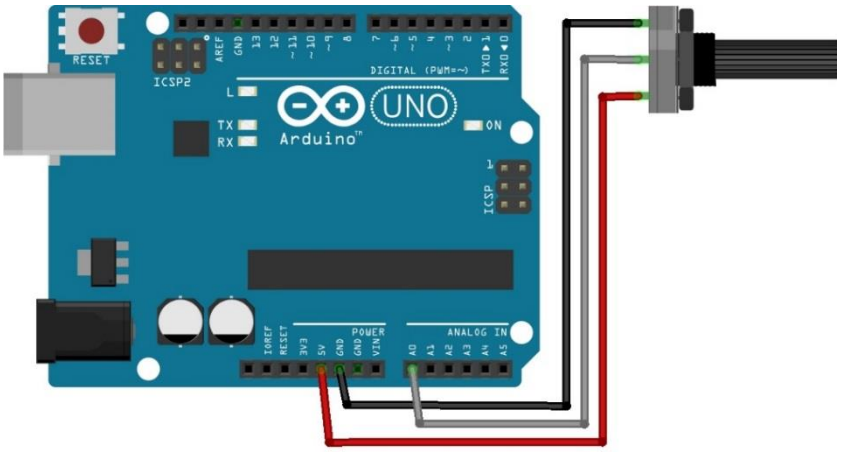


Figura 5.3 Diagrama conexionado potenciómetro deslizante

Por tanto, en la Figura 5.4 podemos ver cómo quedaría la conexión real entre el Arduino y el potenciómetro deslizante:

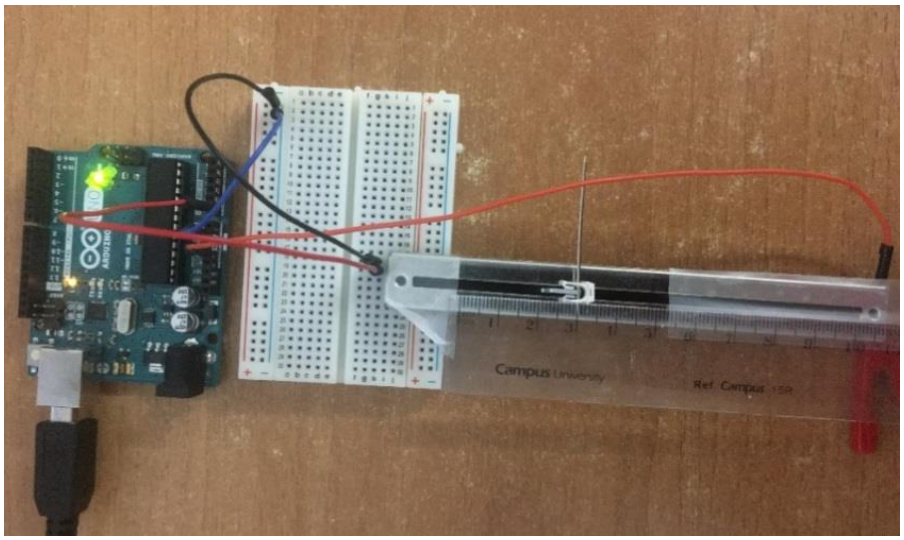


Figura 5.4 Conexión real potenciómetro deslizante

En esta figura además se ha fijado una regla para ir comprobando los resultados de las medidas que obtendremos posteriormente en el ordenador. En la Figura 5.5 se ha realizado un zoom para que se pueda apreciar mejor el método de comprobación de medidas:

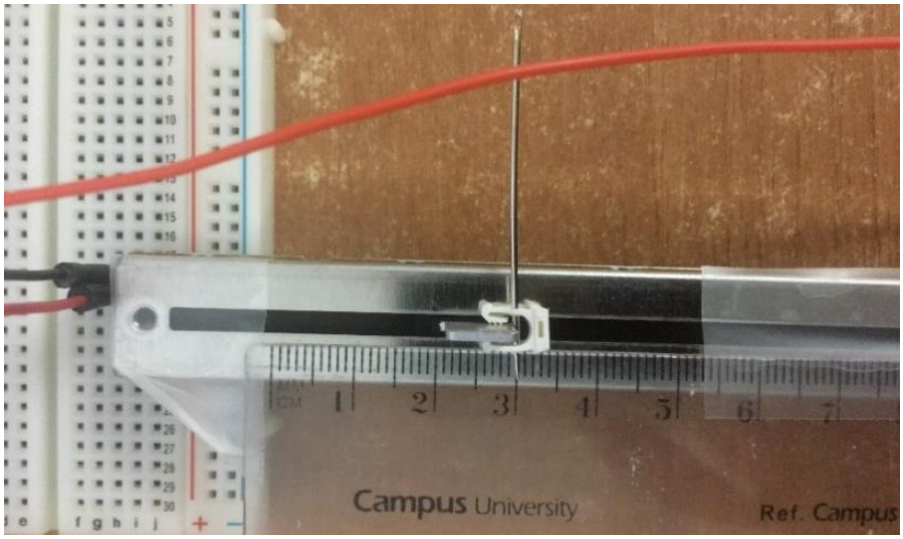


Figura 5.5 Comprobación medidas potenciómetro deslizante

5.3.1.2.2 Potenciómetro hilo

A continuación se adjunta en la Tabla 5.2 un resumen del potenciómetro elegido:

Características	Valor	Observaciones
Modelo	SP1-12	El promediado se realizará con la lectura de 1200 muestras del pin analógico de Arduino
Dimensiones (mm)	48,3x50,8x46,9	
Carrera (mm)	317	
Tipo	Lineal	
Resistencia	10kΩ	
Interfaz de datos	Analógica	
Pruebas		
Cantidad de datos adquiridos	200	
Precisión en la medida (mm)	1	
Coeficiente de determinación (R^2)	0,999 ≈ 1	
Filtrado	Promediado	

Tabla 5.2 Características potenciómetro de hilo

La hoja de características de este sensor puede encontrarse en la siguiente dirección:

http://www.docs-europe.electrocomponents.com

En la Figura 5.6 se puede ver el potenciómetro de hilo SP1-12 :

Figura 5.6 Potenciómetro de hilo SP1-12

El diagrama de conexionado necesario para utilizar este potenciómetro junto a un Arduino es el mismo que el del otro potenciómetro. Por tanto, en la Figura 5.7 podemos ver cómo quedaría la conexión real entre el Arduino y el potenciómetro de hilo:

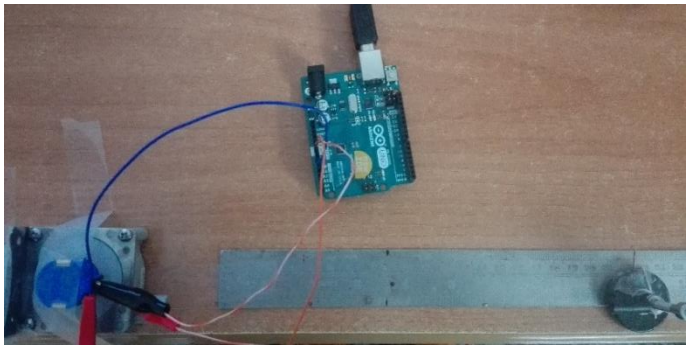


Figura 5.7 Conexión real potenciómetro de hilo

5.3.1.3 Problemas de inestabilidad y soluciones software

5.3.1.3.1 Conversor analógico-digital

En el caso de un Arduino Uno tenemos entradas analógicas con una resolución de 10 bits, es decir, unos valores enteros entre 0 y 1023. El valor de 0 voltios analógico es expresado en digital como B0000000000 (0) y el valor de 5V analógico es expresado en digital como B1111111111 (1023).

Por tanto, el primer código consistirá en la lectura del valor digital correspondiente al analógico obtenido en el pin A0:

```
long valor;

void setup() {
  //Inicializo la comunicación serial
  Serial.begin(9600);
}

void loop() {
  //Leemos el valor del pin A0
  valor = analogRead(0);
  //Imprimimos por el monitor serie
  Serial.print("El valor es = ");
  Serial.println(valor);
}
```

Posteriormente, se añadió una ecuación para convertir los valores digitales en voltaje:

```
long valor;
float voltaje;

void setup() {
  //Inicializo la comunicación serial
  Serial.begin(9600);
    }

void loop() {
  //Leemos el valor del pin A0
  valor = analogRead(0);
  //Imprimimos por el monitor serie
  Serial.println(".........");
  Serial.print("El valor es = ");
  Serial.println(valor);
  //Calculamos valor en voltios
  voltaje=valor*(5.0/1023.0);
  Serial.print("El valor en voltios es= ");
  Serial.println(voltaje);
  delay(950);
}
```

Al realizar la lectura de los valores que toman las distintas posiciones del potenciómetro, entre 0 y 1023, se halló un problema de inestabilidad. Observamos que el monitor serie no muestra datos estables cuando no movemos la patilla del potenciómetro.

A continuación se explica de forma teórica los motivos por los que ocurre y las distintas soluciones que se pueden adoptar.

Una señal eléctrica analógica es aquella en la que los valores de tensión o voltaje varían constantemente y pueden tomar cualquier valor.

Un microcontrolador es un circuito integrado programable, capaz de ejecutar las órdenes grabadas en su memoria, y que dispone de los tres elementos básicos de una microcomputadora: un procesador, memoria e interfaces. Sin embargo, no tiene capacidad para trabajar con señales analógicas, de modo que necesita convertir las señales analógicas en señales digitales para poder trabajar con ellas [18].

Un conversor analógico-digital (A/D) es un dispositivo electrónico capaz de convertir una señal analógica en un valor digital. El dispositivo establece una relación entre su entrada (señal analógica) y su salida (digital) dependiendo de su resolución. La resolución determina la precisión con la que se reproduce la señal original.

Esta resolución se puede saber, siempre y cuando conozcamos el valor máximo de la entrada a convertir y la cantidad máxima de la salida en dígitos binarios.

$$Resolución = \frac{+V_{ref}}{2^n} \;\; ; donde\ n\ son\ bits$$

Por tanto, la señal digital obtenida de una analógica tiene dos propiedades fundamentales:

- Valores: que se define entre 0 y 1. En nuestro caso es tecnología TTL (lógica transistor a transistor) 0 – 5 V.
- Resolución analógica: número de bits que usamos para representar con una notación digital una señal analógica.

La tarjeta Arduino Uno utiliza un conversor A/D de 10 bits y un voltaje de referencia de 5V. Aplicando la ecuación, obtenemos una resolución de 4,883 mV. Es decir, el error en las medidas de voltaje será de aproximadamente de unos 5 mV.

5.3.1.3.2 Variación significativa de tensión

El conversor A/D de Arduino mantiene en "memoria" el último valor presente en su entrada (voltios analógicos). Si la siguiente medida es mucho menor (es decir, no es del orden de magnitud de la anterior), la medida es inexacta. La tensión anterior debe "descargarse" antes de poder reconocer la siguiente tensión, si la diferencia de tensión es mucha no le da tiempo a alcanzar el nivel adecuado para que la medida sea exacta [19].

Por tanto, la solución adoptada consiste en programar la suma de 1200 lecturas para una determinada posición. Posteriormente se hará la división de la suma entre el contador, obteniendo así el valor del promediado.

```
float suma = 0;
int contador = 0;
float promedio = 0;
float voltaje = 0;

void setup() {
  Serial.begin(9600);
}

void loop() {
  do {
    //Suma todas las lecturas de A0
    suma += analogRead(0);
    contador++;
  } while (contador < 1200);
  //Cálculo del promedio
  promedio = (float) (suma / contador);
  //Pasa el valor del promedio a voltios
  voltaje = promedio * (5.0 / 1023.0);
..Serial.println(volatje);
..delay(1000);
  //Preparo las variables para el siguiente promediado
  suma = 0;
  contador = 0;
}
```

A partir de este último vimos que las lecturas ya eran muchísimo más estables, por lo que decidimos tomar los valores de tensión que se obtiene en cada variación milímetro a milímetro con ayuda de la regla de la Figura 5.5.

5.3.1.4 Software Arduino

5.3.1.4.1 Potenciómetro deslizante

Obtuvimos los resultados de la Tabla 5.3:

Longitud (mm)	Voltaje (V)	Longitud (mm)	Voltaje (V)	Longitud (mm)	Voltaje (V)
0	5	34	3,34	68	1,54
1	5	35	3,29	69	1,49
2	4,99	36	3,23	70	1,44
3	4,93	37	3,19	71	1,38
4	4,88	38	3,13	72	1,32
5	4,83	39	3,09	73	1,28
6	4,79	40	3,04	74	1,22
7	4,73	41	2,99	75	1,17
8	4,67	42	2,92	76	1,11
9	4,63	43	2,87	77	1,06
10	4,58	44	2,81	78	1
11	4,52	45	2,77	79	0,95
12	4,47	46	2,71	80	0,89
13	4,43	47	2,65	81	0,84
14	4,37	48	2,6	82	0,79
15	4,33	49	2,56	83	0,75
16	4,27	50	2,5	84	0,69
17	4,21	51	2,44	85	0,64
18	4,16	52	2,39	86	0,59
19	4,11	53	2,34	87	0,54
20	4,06	54	2,28	88	0,48
21	4,01	55	2,23	89	0,43
22	3,96	56	2,17	90	0,37
23	3,91	57	2,12	91	0,32
24	3,85	58	2,07	92	0,27
25	3,8	59	2,01	93	0,22
26	3,75	60	1,96	94	0,17
27	3,7	61	1,9	95	0,11
28	3,64	62	1,85	96	0,06
29	3,59	63	1,8	97	0,02
30	3,55	64	1,74	98	0
31	3,49	65	1,69	99	0
32	3,44	66	1,64	100	0
33	3,39	67	1,59		

Tabla 5.3 Voltaje asociado a cada posición de potenciómetro deslizante

Los datos de la tabla anterior se han representado en la Figura 5.8 con el objetivo de hallar la regresión lineal que nos permita relacionar los valores de voltaje con longitud en milímetros. Hemos representado Posición (mm) frente a Voltaje Leído (V):

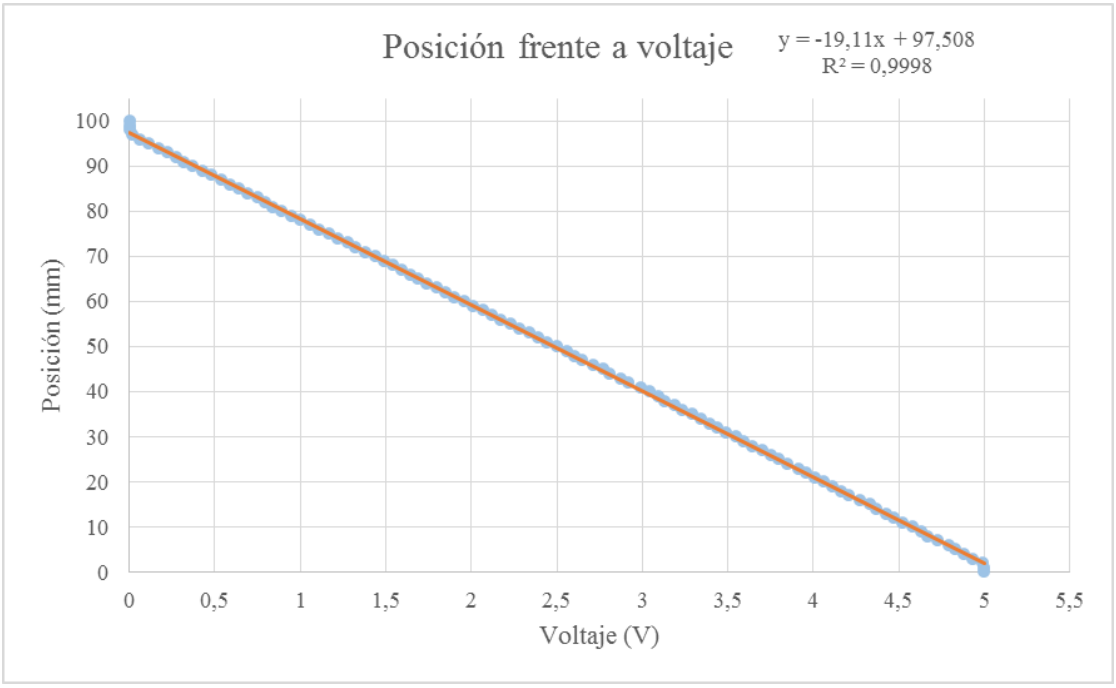


Figura 5.8 Posición frente a voltaje de potenciómetro deslizante

La ecuación de la recta que siguen los datos de la Tabla 5.3 es la siguiente:

$$y = -19{,}11x + 97{,}508$$

Con un valor R^2 de 0,9998

Valores de R^2 cercanos a la unidad nos indican que el ajuste por regresión lineal es bueno. Por tanto, utilizaremos esta ecuación para convertir el voltaje leído en A0 en valores de longitud en milímetros.

Para el potenciómetro deslizante el código final queda de la siguiente forma:

```
float suma = 0;
int contador = 0;
float promedio = 0;
float voltaje = 0;
int mm = 0;


void setup() {
  Serial.begin(9600);
}


void loop() {
```

```
  do {
    //Suma todas las lecturas de A0
    suma += analogRead(0);
    contador++;
  } while (contador < 1200);
  //Cálculo del promedio
  promedio = (float) (suma / contador);
  //Pasa el valor del promedio a voltios
  voltaje = promedio * (5.0 / 1023.0);
  //Valor en mm
  mm = voltaje * (-19.11) + 97.508;
  Serial.println(mm);
  delay(1000);
  //Preparo las variables para el siguiente promediado
  suma = 0;
  contador = 0;
}
```

5.3.1.4.2 Potenciómetro hilo

Obtuvimos los resultados de la Tabla 5.4:

Longitud (mm)	Voltaje (V)	Longitud (mm)	Voltaje (V)	Longitud (mm)	Voltaje (V)	Longitud (mm)	Voltaje (V)
158	1,07	200	1,36	242	1,65	284	1,95
159	1,08	201	1,37	243	1,66	285	1,96
160	1,08	202	1,37	244	1,67	286	1,96
161	1,09	203	1,38	245	1,67	287	1,97
162	1,1	204	1,38	246	1,68	288	1,98
163	1,11	205	1,39	247	1,69	289	1,98
164	1,11	206	1,39	248	1,7	290	1,99
165	1,12	207	1,4	249	1,71	291	1,99
166	1,13	208	1,41	250	1,71	292	2
167	1,14	209	1,42	251	1,72	293	2,01
168	1,14	210	1,43	252	1,73	294	2,01
169	1,15	211	1,43	253	1,73	295	2,02
170	1,15	212	1,44	254	1,74	296	2,02
171	1,16	213	1,45	255	1,75	297	2,03
172	1,17	214	1,45	256	1,75	298	2,04
173	1,18	215	1,46	257	1,76	299	2,04
174	1,18	216	1,47	258	1,77	300	2,05
175	1,19	217	1,48	259	1,77	301	2,06
176	1,2	218	1,48	260	1,78	302	2,06
177	1,21	219	1,49	261	1,78	303	2,07
178	1,21	220	1,5	262	1,79	304	2,07
179	1,22	221	1,5	263	1,8	305	2,08
180	1,22	222	1,51	264	1,8	306	2,09
181	1,23	223	1,52	265	1,81	307	2,09
182	1,24	224	1,52	266	1,82	308	2,1
183	1,24	225	1,53	267	1,83	309	2,11
184	1,25	226	1,54	268	1,83	310	2,11
185	1,26	227	1,54	269	1,84	311	2,12
186	1,26	228	1,55	270	1,85	312	2,12
187	1,27	229	1,56	271	1,86	313	2,13
188	1,28	230	1,57	272	1,86	314	2,14
189	1,28	231	1,58	273	1,87	315	2,14
190	1,29	232	1,58	274	1,88	316	2,15
191	1,3	233	1,59	275	1,88	317	2,16
192	1,31	234	1,59	276	1,89	318	2,16
193	1,31	235	1,6	277	1,9	319	2,17
194	1,32	236	1,61	278	1,91	320	2,18
195	1,33	237	1,61	279	1,91	321	2,18
196	1,33	238	1,62	280	1,92	322	2,19
197	1,34	239	1,63	281	1,93	323	2,19
198	1,35	240	1,64	282	1,94	324	2,2
199	1,35	241	1,65	283	1,94	325	2,21

Tabla 5.4 Voltaje asociado a cada posición de potenciómetro hilo

Los datos de la tabla anterior se han representado en la Figura 5.9, con el objetivo de hallar la regresión lineal que nos permita relacionar los valores de voltaje con longitud en milímetros. Hemos representado Posición (mm) frente a Voltaje Leído (V):

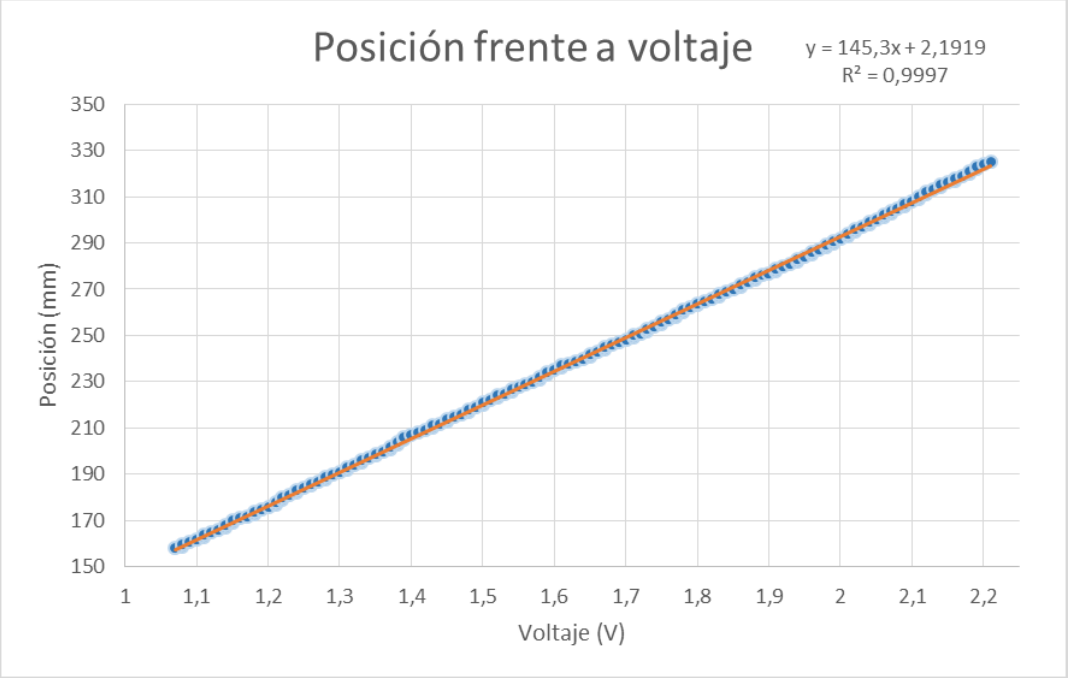


Figura 5.9 Posición frente a voltaje de potenciómetro hilo

La ecuación de la recta que siguen los datos de la Tabla 5.4 es la siguiente:

$$y = 145{,}3x - 2{,}1919$$

Con un valor R^2 de 0,9997

Para el potenciómetro de hilo el código final queda de la siguiente forma:

```
float suma = 0;
int contador = 0;
float promedio = 0;
float voltaje = 0;
int mm = 0;


void setup() {
  Serial.begin(9600);
}


void loop() {
  do {
    //Suma todas las lecturas de A1
    suma += analogRead(1);
    contador++;
  } while (contador < 1200);
```

```
  //Cálculo del promedio
  promedio = (float) (suma / contador);
  //Pasa el valor del promedio a voltios
  voltaje = promedio * (5.0 / 1023.0);
  //Valor en mm
  mm = voltaje * (145.3) + 2.1919;
  Serial.println(mm);
  delay(1000);
  //Preparo las variables para el siguiente promediado
  suma = 0;
  contador = 0;
}
```

5.3.1.5 Conclusiones

Los potenciómetros lineales tanto deslizante como de hilo cumplen el requisito de precisión. Sin embargo, el potenciómetro de hilo, a priori, supone no cumplir los requisitos de ser poco voluminoso y de fácil integración en el sistema TL-Hex. Se ha considerado no necesario hacer un análisis de errores, ya que los errores de precisión de ambos potenciómetros son despreciables.

Para el caso del potenciómetro deslizante, encontramos que en ambos extremos hay una serie de medidas en las que no se detectan correctamente las variaciones de resistencia. Lo indicado ocurre en los tres últimos milímetros desde 9,8 hasta 10 cm, los cuales no suponen un problema ya que el máximo alcance del ajuste gradual es de 8 cm. Esto último ocurre presumiblemente debido a las distintas calidades de fabricación de los potenciómetros.

5.3.2 Sensor basado en los detectores LIDAR

5.3.2.1 ¿Qué es y cómo funciona?

El principio en el que se basan estos sensores es emitir una pequeña luz desde una superficie (integrado donde se encuentra el emisor y el receptor) y medir el tiempo que tarda en retornar. La velocidad de la luz es de alrededor de 3·108 m/s, lo que parecer ser casi instantáneo.

La ecuación para calcular la distancia que ha recorrido el haz de luz desde el emisor hasta el objeto es la siguiente:

$$distancia = \frac{1}{2}(velocidadluz * tiempovuelo)$$

Los instrumentos que integran la tecnología LIDAR emiten pulsos de haz de luz desde un emisor, algunos llegando a alcanzar los 150.000 pulsos por segundo. Un sensor integrado en el instrumento mide la cantidad de tiempo que le requiere a cada pulso retornar. La luz se transmite con una velocidad constante y conocida, por tanto, el instrumento LIDAR puede calcular la distancia entre el mismo y el objeto con gran precisión [20].

5.3.2.2 Bus I2C

Este tipo de sensor utiliza para la transmisión de datos a Arduino un bus de comunicaciones en serie, conocido como "I2C". Su nombre viene de "Inter-Integrated Circuit" [21]. La principal característica es que utiliza dos líneas para transmitir la información: una para datos (SDA) y otra para la señal de reloj (SCL). También es necesaria una tercera línea, pero esta sólo es la masa.

En el bus cada dispositivo dispone de una dirección, que se emplea para acceder al mismo de forma individual. En general, cada dispositivo conectado al bus debe tener una dirección única. Si tenemos varios dispositivos similares tendremos que cambiar las direcciones.

El bus I2C tiene una arquitectura de tipo maestro-esclavo. El dispositivo maestro inicia la comunicación con los esclavos, y puede mandar o recibir datos de los esclavos. Los esclavos no pueden iniciar la comunicación, ni hablar entre sí directamente [22].

El bus I2C es síncrono. El maestro proporciona una señal de reloj, que mantiene sincronizados a todos los dispositivos del bus. No hay necesidad de que cada dispositivo tenga su propio reloj, de tener que acordar velocidades de transmisión ni mecanismos para mantener la transmisión sincronizada.

5.3.2.3 Funcionamiento del bus I2C

Para poder realizar la comunicación con un solo cable de datos, el bus I2C emplea una trama amplia. La comunicación consta de:

- 7 bits para la dirección del dispositivo esclavo
- 1 bit restante para indicar si queremos enviar o recibir información
- 1 bit de validación
- 1 o más bytes para datos enviados o recibidos del esclavo
- 1 bit de validación

En la Figura 5.10 se pueden apreciar los bits de los que consta la comunicación:

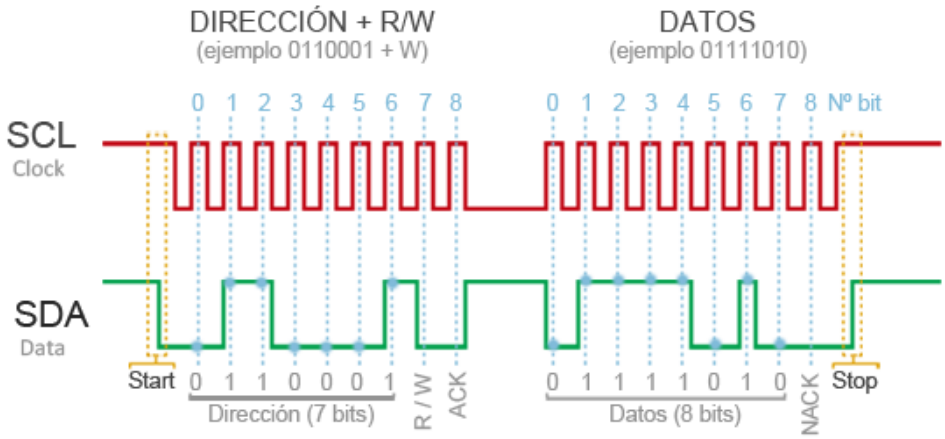


Figura 5.10 Trama del bus I2C

La velocidad estándar de transmisión es de 100Mhz, por lo que es bastante reducida.

5.3.2.4 Sensores LIDAR estudiados

5.3.2.4.1 Adafruit VL53L0X

A continuación en la Tabla 5.5, se pueden ver las principales características de este sensor:

Características	Valor	Observaciones
Modelo	Adafruit VL53L0X	Este sensor se ve muy afectado por el ruido, por lo que necesita un tratamiento de datos del tipo filtrado. El software actual calcula el valor de la mediana de un conjunto de 34 lecturas previamente filtradas por un paso bajo.
Dimensiones (mm)	21x18x2,8	
Alcance (mm)	50 a 1200	
Tipo	LIDAR	
Voltaje alimentación (V)	3 a 5	
Interfaz de datos	I2C	
Pruebas		
Cantidad de datos adquiridos	500	
Precisión en la medida (mm)	± 3	
Filtrado	Mediana y paso bajo	

Tabla 5.5 Características sensor Adafruit VL53L0X

La hoja de características de este sensor puede encontrarse en la siguiente dirección:

https://www.cdn-learn.adafruit.com

A continuación se adjunta en la Figura 5.11 una imagen del sensor estudiado:

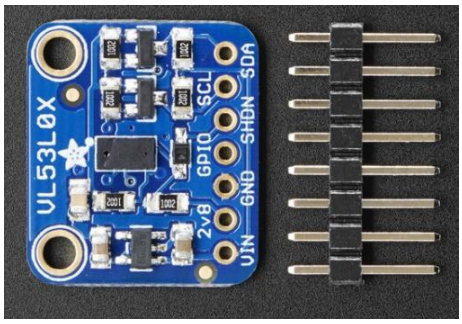


Figura 5.11 Sensor Adafruit VL53L0X

En la Figura 5.12 se adjunta el diagrama de conexionado necesario para utilizar este sensor:

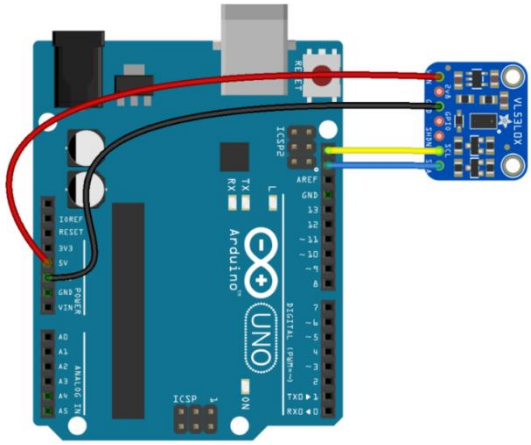


Figura 5.12 Conexionado sensor Adafruit VL53L0X

5.3.2.4.2 Adafruit VL6180X

A continuación en la Tabla 5.6, se pueden ver las principales características de este sensor:

Características	Valor	Observaciones
Modelo	Adafruit VL6180X	Este sensor se ve muy afectado por el ruido, por lo que necesita un tratamiento de datos del tipo filtrado. El software actual calcula el valor de la mediana de un conjunto de 34 lecturas previamente filtradas por un paso bajo.
Dimensiones (mm)	12x10x2	
Alcance (mm)	5 a 100	
Tipo	LIDAR	
Voltaje alimentación (V)	3 a 5	
Interfaz de datos	I2C	
Pruebas		
Cantidad de datos adquiridos	300	
Precisión en la medida (mm)	± 3	
Filtrado	Mediana y paso bajo	

Tabla 5.6 Características sensor Adafruit VL6180X

La hoja de características de este sensor puede encontrarse en la siguiente dirección:

https://www.cdn-learn.adafruit.com

A continuación se adjunta en la Figura 5.13 una imagen del sensor estudiado:

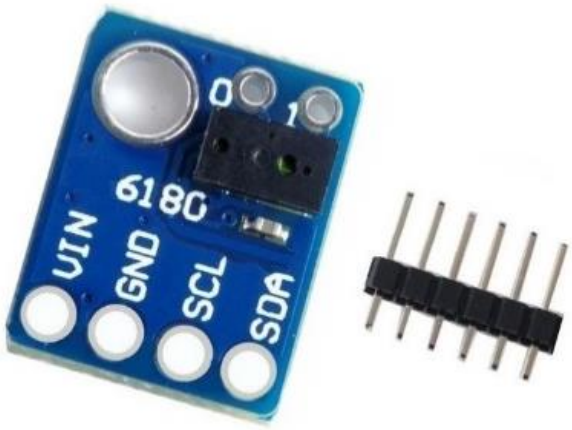


Figura 5.13 Sensor Adafruit VL6180X

El diagrama de conexionado necesario para utilizar este sensor junto a un Arduino 1 Rev 3 coincide con el de la Figura 5.12:

5.3.2.5 Filtrado de la señal y software Arduino

Una vez tenemos el sensor conectado y la librería del mismo instalada en el IDE de programación de Arduino, cargamos el programa de ejemplo del sensor que deseamos probar.

Para el sensor VL53L0X:

https://www.github.com

Para el sensor VL6180X:

https://www.cdn-learn.adafruit.com

5.3.2.5.1 Ruido

Al arrancarlo nos dimos cuenta que la lectura para una distancia fija oscilaba mucho, debido a un problema presente en cualquier sistema electrónico conocido como "Ruido".

Se entiende por ruido a señales de alta frecuencia y carácter aleatorio que interfieren con la señal de interés. La calidad de los sensores influye fuertemente en la cantidad de ruido en la medición, sin embargo, todos los sensores electrónicos presentan ruido.

Dado que las muestras tomadas con carácter individual no son fiables, se deduce que es necesario realizar un muestreo múltiple, es decir, adquirir varias veces la medición del sensor. Posteriormente, tendremos que aplicar algún tipo de algoritmo para combinar las muestras y obtener una medición con mayor precisión que las muestras individuales.

5.3.2.5.2 Filtro paso bajo exponencial (EMA)

En este apartado vamos a centrarnos en el filtro exponencial EMA "Exponential Moving Average". Consiste en obtener un valor filtrado a partir de una medición mediante la aplicación de la siguiente expresión [23]:

$$A_n = \alpha M + (1 - \alpha) A_{n-1}$$

Siendo An el valor filtrado, An-1 el valor filtrado anterior, M es el valor muestreado de la señal a filtrar y alfa es un factor entre 0 y 1.

El resultado de un filtro exponencial EMA es una señal suavizada donde la cantidad de suavizado depende del factor alfa, como analizaremos posteriormente.

Podemos emplear el filtro exponencial como un filtro paso bajo, es decir, un algoritmo que deja pasar las componentes frecuenciales inferiores a una frecuencia de corte. Por tanto, podemos emplear el filtro paso bajo para eliminar el ruido de alta frecuencia superpuesto a la señal.

5.3.2.5.2.1 Influencia del factor alfa

El factor alfa condiciona el comportamiento del filtro exponencial y está relacionado con la frecuencia de corte del filtro. De forma cuantitativa:

- Un valor alfa = 1 proporciona la señal sin filtrar, ya que prescinde del efecto del filtrado que proporciona la medición anterior.
- Un valor de alfa = 0 provoca que el valor filtrado siempre sea 0, ya que prescinde de la información nueva que aporta la medición al sistema

Disminuir el factor alfa aumenta el suavizado de la señal, pero a costa de introducir consecuencias también negativas. Podríamos eliminar componentes frecuenciales que realmente nos fueran de interés, clasificando como ruido algo que realmente era una variación real de la señal.

Por otro lado, disminuir el factor alfa aumenta el tiempo de respuesta del sistema, es decir, el tiempo que tarda el sistema en estabilizarse ante una entrada constante. Lo que se traduce en la introducción de un retraso entre la señal original y la señal filtrada.

5.3.2.5.2.2 Motivos por los que el filtro EMA es capaz de filtrar el ruido

A continuación vamos a demostrar matemáticamente, el por qué este filtro consigue filtrar el ruido de una señal.

El filtro exponencial EMA es un caso particular de media de pesos ponderados, es decir, un cálculo de media en la que cada uno de los elementos Xi dispone de un factor Wi que pondera su efecto en el cálculo global de la media:

$$M_{ponderada} = \frac{\sum(w_i * x_i)}{\sum(w_i)}$$

Partimos de una expresión genérica para obtener una serie de valores filtrados Ai de una serie de mediciones Mi, sujeta a dos parámetros alfa y beta:

$$A_i = \alpha M_i + \beta A_{i-1}$$

Analizando lo que ocurre al emplear esta expresión a la serie de mediciones:

$$A_0 = \alpha M_0$$

$$A_1 = \alpha M_1 + \beta A_0 = \alpha M_1 + \beta \alpha M_0$$

$$A_2 = \alpha M_2 + \beta A_1 = \alpha M_1 + \beta \alpha M_1 + \beta^2 \alpha M_0$$

Que generalizando nos queda:

$$A_{n+1} = \alpha M_{n+1} + \beta A_n = \alpha \sum_{t=0}^{n} (\beta^t M_{n-t})$$

Por tanto, el próximo valor filtrado es la suma de todas las mediciones anteriores ponderados por un factor exponencial beta y escalado con un factor alfa. La media calcula el efecto de todas las mediciones anteriores, aunque el factor beta hace que su efecto sea reducido.

Las restricciones de que los valores de alfa y beta tienen que estar entre 0 y 1 se deben a que:

Por un lado, se deduce que salvo que todas las mediciones sean igual a 0, cualquier factor beta mayor que 1 provocará una señal que terminará tendiendo a infinito. Por otro lado, valores de beta negativos producirán soluciones oscilantes.

Para entender mejor esta relación, supongamos que introducimos como entrada un valor constante C. Es razonable esperar que el valor filtrado obtenido sea igualmente C:

$$C = \alpha \sum_{t=0}^{n} (\beta^t C) = \alpha C \sum_{t=0}^{n} (\beta^t)$$

Siendo beta < 1 la expresión anterior se simplifica:

$$C = \alpha C \frac{1}{1-\beta}$$

Que finalmente, se reduce a:

$$\beta = 1 - \alpha$$

Con lo que las restricciones sobre alfa y beta quedan explicadas. Hemos demostrado que la expresión general del filtro EMA es en realidad una media ponderada cuyos factores de ponderación siguen una progresión exponencial de parámetro beta, y que para que el sistema sea estable beta debe estar comprendido entre 0 y 1.

La librería que implementa este filtro paso bajo exponencial la podemos encontrar en el siguiente enlace:

https://www.github.com

5.3.2.5.3 Filtro mediana móvil rápido

El filtro exponencial EMA es sencillo y eficiente de implementar pero tiene la desventaja de que es poco robusto/estable ante la aparición de puntos espurios (puntos que desvían mucho del valor real) [24].

Esto es debido que se basa en la media como estimador de tendencia, siendo este poco robusto. Un único punto anómalo puede provocar una gran desviación en la media arruinando toda una serie de mediciones.

En este apartado nos centraremos en estudiar un filtro de mediana móvil, siendo la mediana un estimador de tendencia más robusto que la media.

El filtro de mediana móvil define una ventana de N elementos, que recoge las últimas N mediciones. El valor del filtro es la mediana de los valores contenidos en la ventana.

Aumentar el tamaño de la ventana aumenta el suavizado de la señal pero añade un retraso ya que es necesario "esperar" al muestreo de más puntos. Además se prefieren tamaños de ventana impares para evitar tener que promediar entre dos muestras, lo cual eliminaría parte de la capacidad del filtrado.

Este tipo de filtros son muy empleados en sistemas de adquisición de datos y captación de sensores para la eliminación de ruido, por ejemplo, en giroscopios y acelerómetros.

La principal desventaja al emplear la mediana es que computacionalmente es un cálculo más costoso. Por otro lado, únicamente puede devolver puntos muestreados, por lo que perdemos la capacidad de submuestreo que podríamos obtener con otros filtros.

La librería que implementa este filtro mediana móvil la podemos encontrar en el siguiente enlace:

https://www.github.com

5.3.2.6 Pruebas de filtrado en entorno simulado

El objetivo de este apartado es el de comprobar la eficacia de los filtros mencionados anteriormente antes de implementarlos en el actuador mecánico. El objetivo es verificar que realizan el filtrado de forma correcta, para posteriormente integrarlos en los sensores.

5.3.2.6.1 Filtro paso bajo exponencial (EMA)

A continuación se incluye el código utilizando el filtro paso bajo exponencial. Éste programa simula la lectura de 34 valores de una misma distancia y devuelve el valor después de aplicar el filtro paso bajo con un alfa de valor 0,6:

```
#include "SingleEMAFilterLib.h"
 //Toma de datos simulados
float values[] = {134, 133, 132, 135, 138, 132, 131, 135, 134, 136, 133, 134, 137, 130, 133, 134, 132, 135, 131, 130,134, 133, 132, 135, 138, 132, 131, 135, 132, 131, 135, 134, 136, 133 };
size_t valuesLength = sizeof(values) / sizeof(values[0]);
int getMeasure()
{
	size_t static index = 0;
	index++;
	return values[index - 1];
}
SingleEMAFilter<float> singleEMAFilter(0.6);

void setup() {
	Serial.begin(9600);
	for (size_t iCount = 0; iCount < valuesLength; iCount++)
	{
		// Obtener medicion simulada
		float rawMeasure = getMeasure();
		// Calcular filtro
		singleEMAFilter.AddValue(rawMeasure);
		// Mostrar resultados
		Serial.print(rawMeasure);
		Serial.print(",\t");
		Serial.println(singleEMAFilter.GetLowPass());
	}
}

void loop()
{
}
```

Ejecutando este código obtenemos los siguientes resultados de la Tabla 5.7:

dato simulado (mm)	paso bajo (mm)	dato simulado (mm)	paso bajo (mm)
134	80,4	135	134,04
133	111,96	131	132,22
132	123,98	130	130,89
135	130,59	134	132,75
138	135,04	133	132,9
132	133,21	132	132,36
131	121,89	135	133,94
135	133,75	138	136,38
134	133,9	132	133,75
136	135,1	131	132,1
133	133,86	135	133,84
134	133,95	132	132,74
137	135,78	131	131,69
130	132,31	135	133,68
133	132,72	134	133,87
134	133,49	136	135,15
132	132,6	133	133,86

Tabla 5.7 Filtrado paso bajo de datos simulados

Siguiendo la tabla anterior, el valor de la distancia medida oscila alrededor de 133 mm, encontrándonos casos en los que incluso puede alcanzar los 135 mm. Sin embargo, como hemos mencionado en el apartado "5.3.2.5.3 Filtro mediana móvil" esto se debe a puntos espurios.

5.3.2.6.2 Filtro mediana móvil rápido

A continuación se incluye el código utilizando el filtro mediana móvil rápido. Éste programa simula la lectura de 34 valores de una misma distancia y devuelve el valor después de aplicar el filtro:

```
#include "MedianFilterLib.h"
//Toma de datos simulados
float values[] = {134, 133, 132, 135, 138, 132, 131, 135, 134, 136, 133, 134, 137, 130, 133, 134, 132, 135, 131, 130,134, 133, 132, 135, 138, 132, 131, 135, 132, 131, 135, 134, 136, 133 };
size_t valuesLength = sizeof(values) / sizeof(values[0]);
int getMeasure()
{
	size_t static index = 0;
	index++;
	return values[index - 1];
}
MedianFilter<float> medianFilter(5);

void setup()
```

```
{
	Serial.begin(9600);
	float timeMean = 0;
	for (size_t iCount = 0; iCount < valuesLength; iCount++)
	{
		float rawMeasure = getMeasure();
		unsigned long timeCount = micros();
		float median = medianFilter.AddValue(rawMeasure);
		timeCount = micros() - timeCount;
		timeMean += timeCount;
		Serial.print(rawMeasure);
		Serial.print(",");
		Serial.println(median);
	}
	Serial.println(timeMean / valuesLength);
}

void loop()
{
}
```

Ejecutando este código obtenemos los siguientes resultados de la Tabla 5.8:

dato simulado (mm)	mediana (mm)	dato simulado (mm)	mediana (mm)
134	134	135	133
133	134	131	133
132	133	130	132
135	134	134	132
138	134	133	133
132	133	132	132
131	132	135	133
135	135	138	134
134	134	132	133
136	134	131	132
133	134	135	135
134	134	132	132
137	134	131	132
130	134	135	132
133	133	134	134
134	134	136	134
132	133	133	134

Tabla 5.8 Filtrado mediana de datos simulados

Siguiendo la tabla anterior, el valor de la distancia medida se estabiliza en 134 mm. Este filtro, a diferencia del "Filtro paso bajo exponencial" es más robusto ante la aparición de puntos espurios.

5.3.2.6.3 Combinación filtro mediana móvil junto a paso bajo exponencial

Sin embargo, vamos a comprobar si es posible mejorar el comportamiento combinando el filtro de mediana móvil con el filtro EMA, teniendo en cuenta que el factor de ambos filtros se va a acumular. Utilizaremos un tamaño de ventana de 5 y un alfa de 0,3 y comprobaremos si el retraso de la señal es muy grande.

El código combinación de ambos filtros para un entorno simulado es el siguiente:

```
#include "MedianFilterLib.h"
#include "SingleEMAFilterLib.h"
//Toma de datos simulados
float values[] = {134, 133, 132, 135, 138, 132, 131, 135, 134, 136, 133, 134, 137, 130, 133, 134,
132, 135, 131, 130,134, 133, 132, 135, 138, 132, 131, 135, 132, 131, 135, 134, 136, 133 };
size_t valuesLength = sizeof(values) / sizeof(values[0]);
int getMeasure()
{
  size_t static index = 0;
  index++;
  return values[index - 1];
}
MedianFilter<float> medianFilter(5); //Utiliza un tamaño de ventana 5
SingleEMAFilter<float> singleEMAFilter(0.3);

void setup()
{
  Serial.begin(9600);
  float timeMean = 0;
  for (size_t iCount = 0; iCount < valuesLength; iCount++)
  {
    float rawMeasure = getMeasure();
    unsigned long timeCount = micros();
    float median = medianFilter.AddValue(rawMeasure);
    timeCount = micros() - timeCount;
    timeMean += timeCount;
    Serial.print(rawMeasure);
    Serial.print(",");
    Serial.print(median);
    singleEMAFilter.AddValue(median);
    Serial.print(",\t");
    Serial.println(singleEMAFilter.GetLowPass());
  }
}

void loop()
{
}
```

Ejecutando este código obtenemos los siguientes resultados de la Tabla 5.9:

Dato simulado (mm)	Tiempo dato (s)	Mediana (mm)	Tiempo mediana (s)	Paso bajo (mm)	Tiempo paso bajo (s)
134	0,001	134	0,001	40,2	0,003
133	0,03	134	0,056	68,34	0,084
132	0,112	133	0,139	87,74	0,167
135	0,196	134	0,223	101,62	0,252
138	0,281	134	0,308	111,33	0,337
132	0,366	133	0,393	117,83	0,423
131	0,452	132	0,479	122,08	0,508
135	0,537	135	0,564	125,96	0,593
134	0,622	134	0,649	128,37	0,678
136	0,708	134	0,735	130,06	0,764
133	0,793	134	0,82	131,24	0,849
134	0,878	134	0,905	132,07	0,934
137	0,963	134	0,99	132,65	1,02
130	1,05	134	1,078	133,05	1,108
133	1,138	133	1,166	133,04	1,196
134	1,226	134	1,255	133,33	1,285
132	1,315	133	1,343	133,23	1,373
135	1,403	133	1,431	133,16	1,462
131	1,492	133	1,52	133,11	1,55
130	1,58	132	1,608	132,78	1,638
134	1,668	132	1,697	132,54	1,727
133	1,757	133	1,785	132,68	1,815
132	1,845	132	1,873	132,48	1,904
135	1,934	133	1,962	132,63	1,992
138	2,022	134	2,05	133,04	2,08
132	2,11	133	2,139	133,03	2,169
131	2,199	132	2,227	132,71	2,257
135	2,287	135	2,315	133,41	2,346
132	2,376	132	2,404	132,98	2,434
131	2,464	132	2,492	132,69	2,522
135	2,552	132	2,581	132,48	2,611
134	2,641	134	2,669	132,94	2,699
136	2,729	134	2,757	133,26	2,788
133	2,818	134	2,846	133,48	2,876

Tabla 5.9 Filtrado mediana y paso bajo de datos simulados

Los cuales podemos representar gráficamente en la Figura 5.14 para interpretar los resultados más fácilmente:

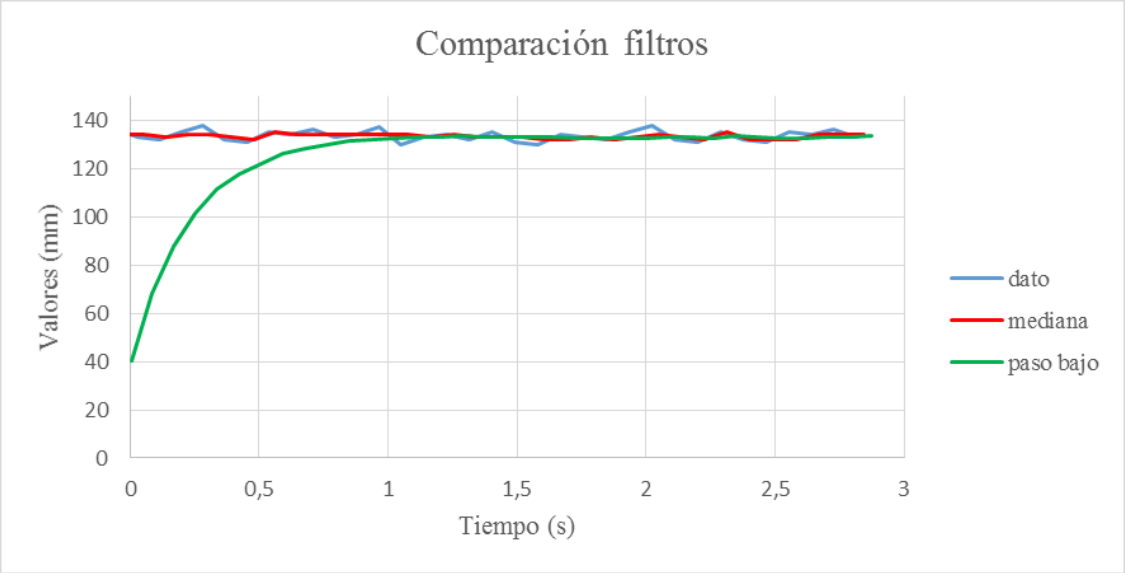


Figura 5.14 Comparación de filtros en entorno simulado

Queda comprobada que la combinación de un filtro mediana con un tamaño de ventana de 5 junto con el filtro EMA con un alfa de 0,3 aporta la eliminación de ruido y espurios de la mediana junto con la suavidad y submuestro del filtro EMA, a la vez que el retraso en obtener los valores es aceptable.

En la Tabla 5.10 podemos ver un resumen sobre el filtrado:

Combinación filtros	**Resultados**
Muestreo múltiple:	34 datos para obtener una medida
Filtro paso bajo:	suaviza la señal eliminando componentes de alta frecuencia
Filtro mediana:	elimina el ruido
Tiempo en obtener lectura aceptable:	1,55 segundos
Retraso computacional total:	2,876 segundos

Tabla 5.10 Resumen del filtrado en entorno simulado

5.3.2.7 Integración sensores en el actuador mecánico

Disponemos de varias posibilidades a la hora de integrar estos sensores en el actuador mecánico. Las primeras que se pensaron son las que se exponen a continuación:

5.3.2.7.1 Adafruit VL53L0X

A continuación, en la Figura 5.15 se adjunta una imagen de las plataformas diseñadas con ayuda del programa "Solid Works". Este diseño es necesario para poder enviarlo al software de la impresora 3D utilizada para la impresión de las plataformas que sujetan la electrónica y la pantalla reflectora.

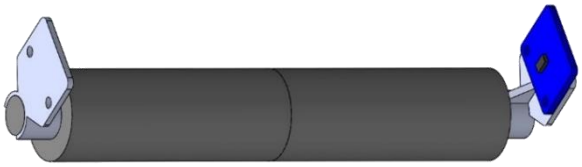


Figura 5.15 Diseño plataformas para sensor Adafruit VL53L0X

Por lo tanto, quedaría como se ve en la Figura 5.16:

Figura 5.16 Integración plataformas y sensor Adafruit VL53L0X

5.3.2.7.2 Adafruit VL6180X

Debido a que las dimensiones de este módulo son bastante más reducidas que la del VL53L0X, se ha pensado en la posibilidad de integrarlo "dentro" del actuador mecánico. Con lo que se pretende poder realizar la medición del ajuste rápido, esto implica que necesitaremos otro sensor para poder realizar la medición del ajuste de precisión.

La primera prueba que se realizó consistió en comprobar si este sensor nos devuelve medidas que tengan sentido estando entubado, ya que como hemos dicho anteriormente, el haz de luz tiene forma de "cono" por lo que se podría reflejar contra el tubo y darnos medidas incoherentes.

En la Figura 5.17 se ve cómo se realizó esta primera prueba:

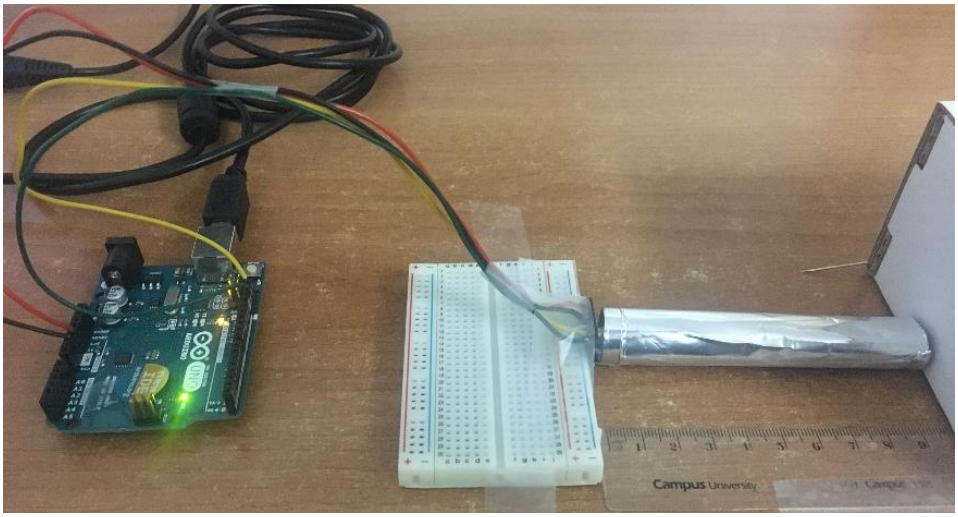


Figura 5.17 Prueba de sensor Adafruit VL6180X entubado

Después de cargar el programa y realizar varias mediciones para distintas distancias, concluimos que el sensor entubado funcionaba adecuadamente.

Posteriormente, se realizó una segunda prueba, esta consistió en integrar el sensor en un actuador mecánico del sistema TL-Hex. Por lo que se ha eliminado un pequeño trozo de la rebaba del actuador, como se puede ver en la Figura 5.18:

Figura 5.18 Integración sensor Adafruit VL6180X en actuador mecánico

Al igual que el sensor VL53L0X, este también necesita una pantalla reflectora, preferiblemente blanca como la de la Figura 5.19:

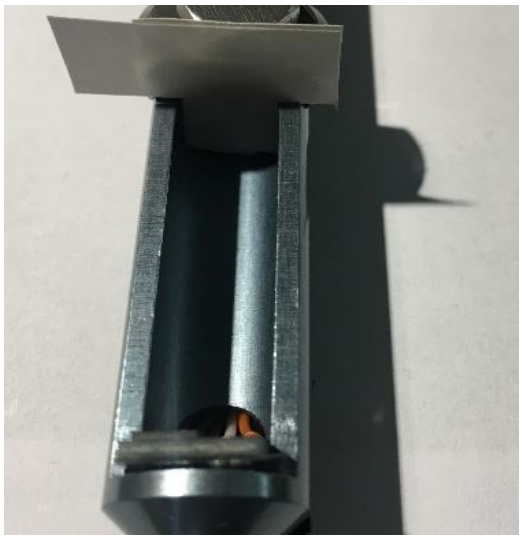


Figura 5.19 Pantalla reflectora de sensor Adafruit VL6180X

Al igual que con el sensor VL53L0X, cuanto mayor sea la superficie reflectora mayor precisión obtendremos en las medidas.

5.3.2.8 Pruebas de filtrado en entorno real

A diferencia del apartado de "Pruebas de filtrado en entorno simulado" este lo dividiremos en dos bloques, cada uno correspondiente a un sensor. Esto se debe a que cada sensor necesita implementar una librería específica.

5.3.2.8.1 Adafruit VL53L0X

Una vez integrado el sensor en el actuador (ver Figura 5.16), procederemos a la lectura de distancias en el entorno REAL, ya que para un entorno simulado funciona adecuadamente. La modificación del código para que realice 34 lecturas y posteriormente haga los filtrados de mediana y paso bajo es el siguiente:

```
#include "Adafruit_VL53L0X.h"
#include "MedianFilterLib.h"
#include "SingleEMAFilterLib.h"
Adafruit_VL53L0X lox = Adafruit_VL53L0X();
float values[34];
int i;
size_t valuesLength = sizeof(values) / sizeof(values[0]);
int getMeasure()
{
  size_t static index = 0;
  index++;
  return values[index - 1];
}
MedianFilter<int> medianFilter(5); //Utiliza un tamaño de ventana 5
SingleEMAFilter<float> singleEMAFilter(0.3); //Utiliza un alpha de 0.3

void setup() {
  Serial.begin(9600);
  // wait until serial port opens for native USB devices
  while (! Serial) {
    delay(1);
  }
  Serial.println("Adafruit VL53L0X test");
  if (!lox.begin()) {
    Serial.println(F("Failed to boot VL53L0X"));
    while(1);
  }
  // power
  Serial.println(F("VL53L0X API Simple Ranging example\n\n"));
  VL53L0X_RangingMeasurementData_t measure;
   for(i=0; i<34;i++){
  lox.rangingTest(&measure, false); // pass in 'true' to get debug data printout!
  if (measure.RangeStatus != 4) {  // phase failures have incorrect data
     Serial.println(measure.RangeMilliMeter);
```

```
    values[i]= measure.RangeMilliMeter;
    }
    else {
    Serial.println(" out of range ");
      }
  }
  delay(100);
////// FILTRO MEDIANA Y MEDIA ////////
   Serial.println("------FILTRADO------");
  float timeMean = 0;
  for (size_t iCount = 0; iCount < valuesLength; iCount++)
  {
    float rawMeasure = getMeasure();
    unsigned long timeCount = micros();
    float median = medianFilter.AddValue(rawMeasure);
    timeCount = micros() - timeCount;
    timeMean += timeCount;
    Serial.print(rawMeasure);
    Serial.print(" , ");
    Serial.print(median);
    singleEMAFilter.AddValue(median);
    Serial.print(" , \t");
    Serial.println(singleEMAFilter.GetLowPass());
  }
}

void loop()
{
}
```

Vamos a realizar la lectura para una posición fija conocida, siendo ésta 230 mm. El objetivo de este apartado es analizar qué valor nos devuelve el sensor, el error respecto de la lectura, el retraso computacional total…

En la Tabla 5.11 podemos ver los resultados obtenidos para una posición de 230 mm:

Lectura (mm)	tiempo lectura (s)	Mediana (mm)	Tiempo mediana (s)	Paso bajo (mm)	Tiempo paso bajo (s)
232	1,882	232	1,882	69,6	1,907
230	1,932	232	1,96	118,32	1,988
233	2,013	232	2,041	152,42	2,069
230	2,094	232	2,122	176,3	2,15
232	2,175	232	2,203	193,01	2,231
236	2,256	232	2,284	204,71	2,312
231	2,337	232	2,366	212,89	2,394
230	2,419	231	2,447	218,33	2,475
234	2,5	232	2,528	222,43	2,556
232	2,581	232	2,609	225,3	2,637
233	2,662	232	2,69	227,31	2,718
232	2,743	232	2,771	228,72	2,799
234	2,824	233	2,852	230	2,88
233	2,905	233	2,933	230,9	2,961
231	2,986	233	3,014	231,53	3,043
233	3,068	233	3,096	231,97	3,124
232	3,149	233	3,177	232,28	3,205
228	3,23	232	3,258	232,2	3,286
232	3,311	232	3,339	232,14	3,367
230	3,392	232	3,42	232,1	3,448
232	3,473	232	3,501	232,07	3,529
233	3,554	232	3,582	232,05	3,61
233	3,635	232	3,663	232,03	3,692
234	3,716	233	3,745	232,32	3,773
234	3,798	233	3,826	232,53	3,854
237	3,879	234	3,907	232,97	3,935
233	3,96	234	3,988	233,28	4,016
232	4,041	234	4,069	233,49	4,097
232	4,122	233	4,15	233,35	4,178
230	4,203	232	4,231	232,94	4,259
232	4,284	232	4,312	232,66	4,34
234	4,365	232	4,394	232,46	4,422
233	4,447	232	4,475	232,32	4,503
232	4,528	232	4,556	232,23	4,584

Tabla 5.11 Prueba de filtrado entorno real sensor Adafruit VL53L0X

Para que la interpretación sea más sencilla, realizaremos una representación gráfica de la tabla anterior obteniendo la Figura 5.20:

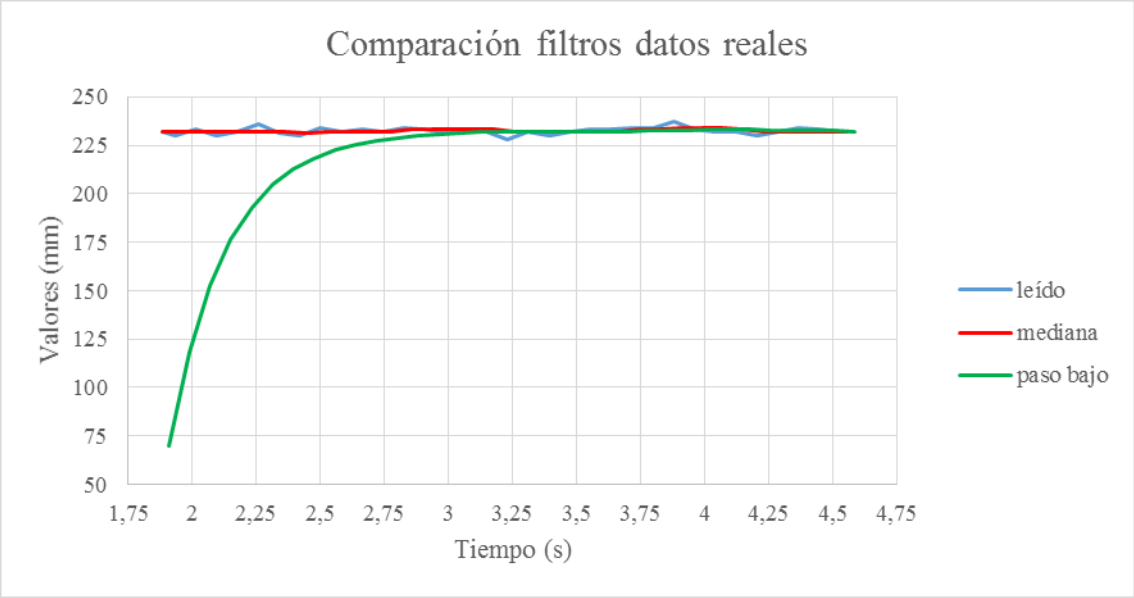


Figura 5.20 Comparación de filtrado entorno real sensor Adafruit VL53L0X

Las conclusiones obtenidas se pueden ver en la Tabla 5.12:

Combinación filtros	Resultados
Muestreo múltiple:	34 datos para obtener una medida
Filtro paso bajo:	suaviza la señal eliminando componentes de alta frecuencia
Filtro mediana:	elimina el ruido
Posición real:	230 mm
Posición filtrada:	232,23 mm
Error absoluto:	2,23 mm
Error relativo:	0,969
Tiempo obtener una lectura aceptable:	3,286 segundos
Retraso computacional total:	4,584 segundos

Tabla 5.12 Resumen de filtrado entorno real sensor Adafruit VL53L0X

5.3.2.8.2 Adafruit VL6180X

Una vez integrado el sensor, procederemos a la lectura de distancias en el entorno REAL, ya que para un entorno simulado funciona adecuadamente. La modificación del código para que realice 34 lecturas y posteriormente hago los filtrados de mediana y paso bajo es el siguiente:

```
#include <Wire.h>
#include "Adafruit_VL6180X.h"
#include "MedianFilterLib.h"
```

```
#include "SingleEMAFilterLib.h"
Adafruit_VL6180X vl = Adafruit_VL6180X();
int values[34];
int i;
size_t valuesLength = sizeof(values) / sizeof(values[0]);
int getMeasure()
{
  size_t static index = 0;
  index++;
  return values[index - 1];
}
MedianFilter<int> medianFilter(5);
SingleEMAFilter<float> singleEMAFilter(0.3); //Utiliza un alpha de 0.3

void setup() {
  Serial.begin(9600);
  // wait for serial port to open on native usb devices
  while (!Serial) {
    delay(1);
  }
  Serial.println("Adafruit VL6180x test!");
  if (! vl.begin()) {
    Serial.println("Failed to find sensor");
    while (1);
  }
  Serial.println("Sensor found!");
   for(i=0; i<34;i++){
  uint8_t range = vl.readRange();
  uint8_t status = vl.readRangeStatus();
  if (status == VL6180X_ERROR_NONE) {
    values[i]= range;
    Serial.println(values[i]);
    delay(1000);
  }
      else {
  // Some error occurred, print it out!
  if  ((status >= VL6180X_ERROR_SYSERR_1) && (status <=
VL6180X_ERROR_SYSERR_5)) {
    Serial.println("System error");
  }
  else if (status == VL6180X_ERROR_ECEFAIL) {
    Serial.println("ECE failure");
  }
  else if (status == VL6180X_ERROR_NOCONVERGE) {
    Serial.println("No convergence");
  }
  else if (status == VL6180X_ERROR_RANGEIGNORE) {
    Serial.println("Ignoring range");
  }
  else if (status == VL6180X_ERROR_SNR) {
    Serial.println("Signal/Noise error");
  }
  else if (status == VL6180X_ERROR_RAWUFLOW) {
```

```
    Serial.println("Raw reading underflow");
   }
   else if (status == VL6180X_ERROR_RAWOFLOW) {
    Serial.println("Raw reading overflow");
   }
   else if (status == VL6180X_ERROR_RANGEUFLOW) {
    Serial.println("Range reading underflow");
   }
   else if (status == VL6180X_ERROR_RANGEOFLOW) {
    Serial.println("Range reading overflow");
   }
   delay(50);
      }
   }
 ////// FILTRO MEDIANA Y MEDIA ////////
    Serial.println("------FILTRADO------");
   float timeMean = 0;
   for (size_t iCount = 0; iCount < 34; iCount++)
   {
    float rawMeasure = getMeasure();
    unsigned long timeCount = micros();
    float median = medianFilter.AddValue(rawMeasure);
    timeCount = micros() - timeCount;
    timeMean += timeCount;
    Serial.print(rawMeasure);
    Serial.print(" , ");
    Serial.print(median);
    singleEMAFilter.AddValue(median);
    Serial.print(" , \t");
    Serial.println(singleEMAFilter.GetLowPass());
   }
}

void loop()
{
}
```

Los resultados obtenidos para diferentes distancias son los de la Tabla 5.13 a continuación:

marca (mm)	Pie de rey (mm)	Leído (mm)	Error absoluto	Error relativo (%)
0	fuera de rango	fuera de rango	fuera de rango	fuera de rango
5	fuera de rango	fuera de rango	fuera de rango	fuera de rango
10	7	10,16	-3,16	-45,143
15	11	8,89	2,11	19,182
20	17	13,01	3,99	23,471
25	21	19,14	1,86	8,857
30	26	24,15	1,85	7,115
35	31	29,91	1,09	3,516
40	35,5	33,09	2,41	6,789
45	41	40,99	0,01	0,024
50	46	43,32	2,68	5,826
55	51	48,74	2,26	4,431
60	56	52,77	3,23	5,768
65	61	58,45	2,55	4,18
70	66	63	3	4,545
75	71	67	4	5,634
80	76	72,33	3,67	4,829

Tabla 5.13 Prueba de filtrado entorno real sensor Adafruit VL6180X

Los valores sombreados de la tabla anterior son los que consideraremos como fuera de rango, ya que nos ofrecen una lectura incoherente o con error muy grande.

A continuación, en la Figura 5.21 representamos gráficamente los resultados de la Tabla 5.13 anterior, a fin de comprobar si existe una tendencia lineal entre ambas variables:

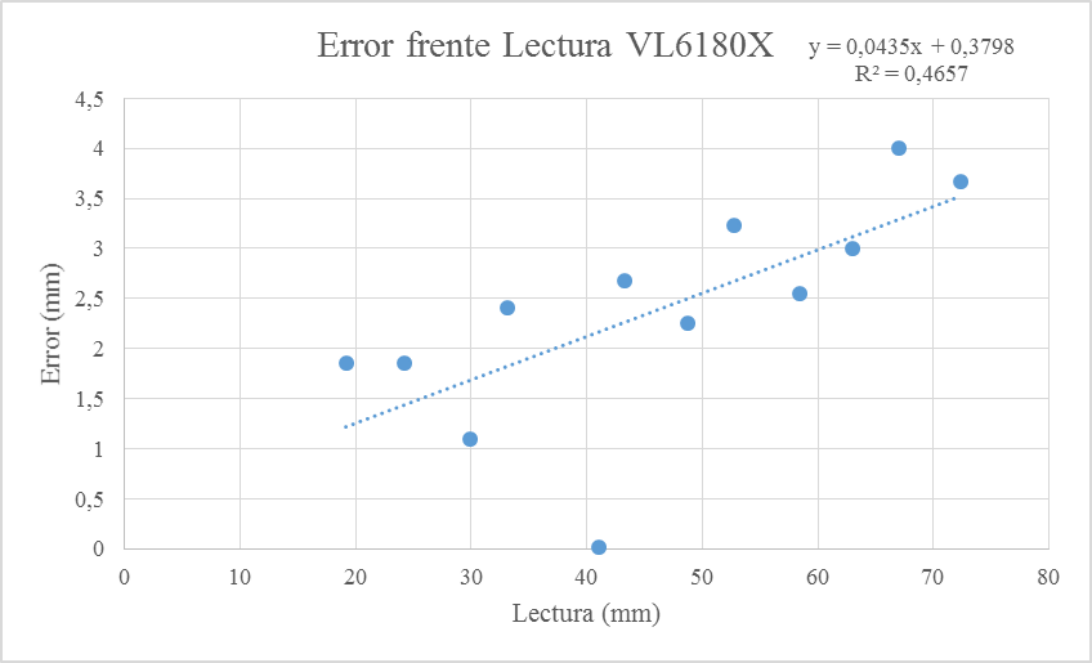


Figura 5.21 Error frente a Lectura sensor VL6180X

Con esta gráfica podemos comprobar que parece que existe una tendencia lineal entre las variables error absoluto y lectura. Sin embargo, podemos observar que además de eliminar los datos fuera de rango debemos eliminar el dato atípico para ver si mejora el valor de R^2 de 0,4657.

Eliminando el dato atípico obtenemos la Figura 5.22:

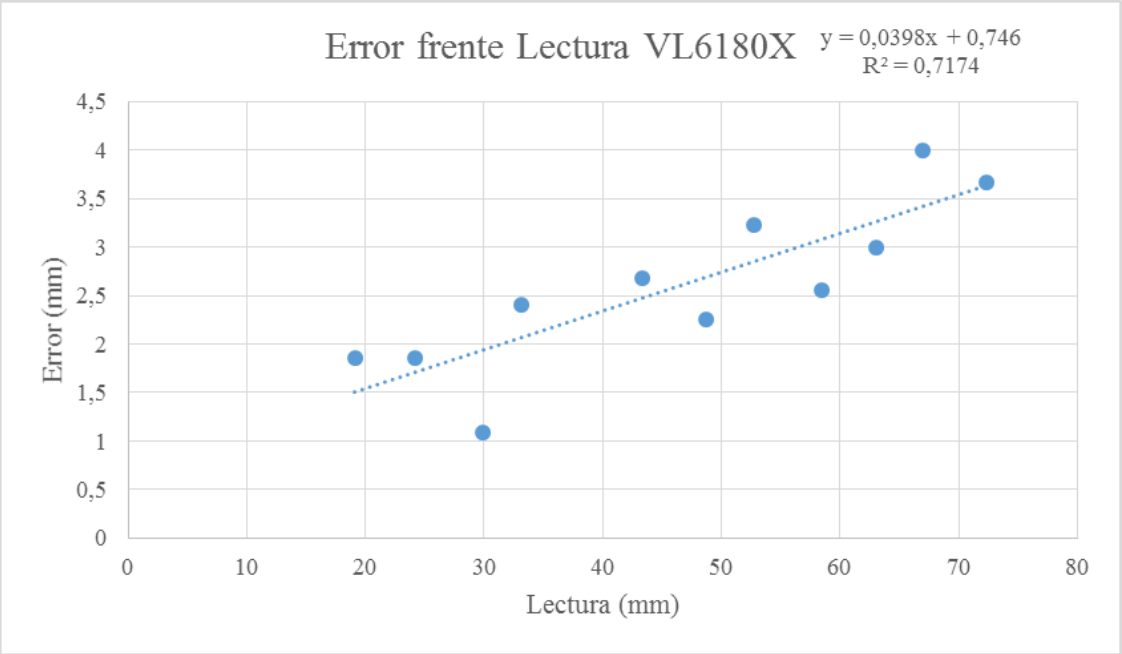


Figura 5.22 Error frente a Lectura sensor VL6180X sin dato atípico

Podemos comprobar que sí existe una relación lineal entre las variables. Al eliminar el dato atípico el valor de R^2 mejora a 0,7174 lo que nos permitiría validar la hipótesis de tendencia lineal. Con la ecuación de la recta del gráfico podemos corregir el error en la lectura, modificando el código Arduino.

5.3.2.9 Problemas y soluciones

5.3.2.9.1 Adafruit VL53L0X

Los problemas asociados a este sensor se detallan a continuación:

5.3.2.9.1.1 Tamaño de la pantalla

Como hemos visto anteriormente, este tipo de sensor necesita una superficie en la cual rebotar (preferiblemente de color blanca). En la Figura 5.23, podemos ver el sensor junto a la pantalla mencionada:

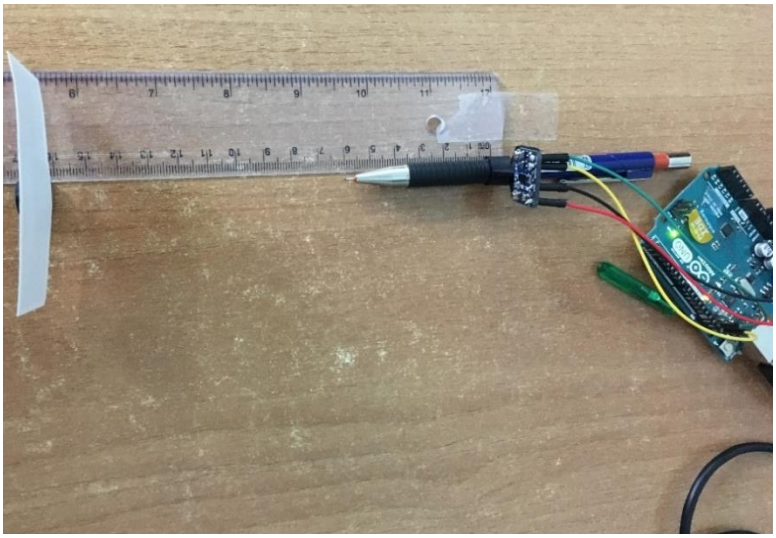


Figura 5.23 Pantalla sensor Adafruit VL53L0X

En una primera aproximación se pensó que el tamaño de la pantalla podría ser menor de la dimensión 5x5 cm y que el sensor funcionaría adecuadamente. Sin embargo, como se puede ver en la Tabla 5.14 esta hipótesis es errónea:

pantalla (cm)	Rango detección (mm)	Leído (mm)
2x2	50	fuera de rango
	80	fuera de rango
5x3,5	30	32-33
	40	fuera de rango
	50	fuera de rango
	100	fuera de rango
	110	109-110
	120	115-118
	150	143-144

Tabla 5.14 Primera prueba tamaño pantalla Adafruit VL53L0X

Con estos tamaños de pantalla no obtenemos una medida adecuada de las longitudes. Como segunda aproximación, se aumentó considerablemente el tamaño de la pantalla y se volvió a realizar la toma de datos. En la Tabla 5.15 podemos ver los resultados:

Pantalla 9x6,5 (cm)			Pantalla 9x6,5 (cm)		
Lectura regla (mm)	Leído (mm)	error absoluto (mm)	Lectura regla (mm)	Leído (mm)	error absoluto (mm)
135	135	0	215	222	-7
140	139	1	220	228	-8
145	146	-1	225	231	-6
150	149	1	230	236	-6
155	155	0	235	240	-5
160	161	-1	240	247	-7
165	166	-1	245	252	-7
170	169	1	250	257	-7
175	176	-1	255	263	-8
180	181	-1	260	270	-10
185	188	-3	265	274	-9
190	192	-2	270	282	-12
195	197	-2	275	291	-16
200	204	-4	280	296	-16
205	216	-11	285	304	-19
210	218	-8			

Tabla 5.15 Segunda prueba tamaño pantalla Adafruit VL53L0X

Podemos observar que hay un dato sombreado en verde, esto se debe a que a partir de esa medida (20 cm) nos dimos cuenta que el error aumenta de forma considerable y que a esa distancia lo más probable es que el haz de luz rebotase contra alguna superficie y se perdiesen medidas. Esto implica que este sensor es muy difícil de integrar en un sistema donde constantemente puede estar muy cercano a otra superficie, ya sea otro actuador mecánico o con alguno de los anillos del sistema TL-Hex.

Para asegurarnos, se decidió colocar el actuador mecánico de forma que el haz de luz no pudiese incidir contra otras superficies cercanas. Se volvió a realizar la toma de medida de aquellas que hemos considerado incorrectas y los resultados son los de la siguiente Tabla 5.16:

Pantalla 9x6,5 (cm)				
Lectura regla (mm)	**Leído (mm)**	**error absoluto (mm)**	**nueva lectura (mm)**	**error absoluto (mm)**
200	204	-4	202	-2
205	216	-11	206	-1
210	218	-8	212	-2
215	222	-7	216	-1
220	228	-8	221	-1
225	231	-6	223	2
230	236	-6	229	1
235	240	-5	232	3
240	247	-7	236	4
245	252	-7	240	5
250	257	-7	246	4
255	263	-8	251	4
260	270	-10	258	2
265	274	-9	259	6
270	282	-12	264	6
275	291	-16	270	5
280	296	-16	273	7
285	304	-19	276	9

Tabla 5.16 Tercera prueba tamaño pantalla Adafruit VL53L0X

Podemos comprobar que el error disminuye considerablemente. Sin embargo, errores mayores de medio centímetro siguen siendo no aceptables para elegir como válido este sensor.

5.3.2.9.1.2 Amplitud del haz emisor y receptor

Se decidió analizar la forma del haz de luz, siguiendo Figura 5.24 extraída de la hoja de características del sensor:

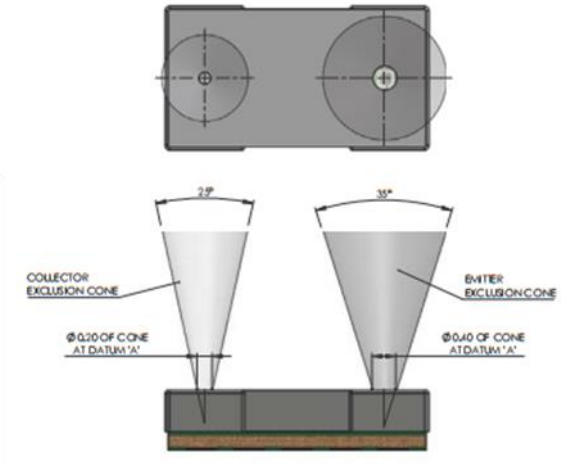


Figura 5.24 Cono haz luz Adafruit VL53L0X

Lo más importante a destacar es que el "cono" del haz de luz aumenta considerablemente con la distancia, es decir, cuanto mayor sea la distancia que deseamos medir, mayor debe ser la pantalla. Específicamente, el emisor tiene un cono de 35^{O} y el emisor de 25^{O}.

Como solución, se decidió aumentar aún más el tamaño de la pantalla. Recordemos que la última era de 9x6,5 cm lo que ya empieza a ser un tamaño considerable. Por tanto, utilizamos una pantalla de 15x9 cm para esta última prueba y representamos los resultados en la Tabla 5.17:

Lectura regla (mm)	Pantalla 9x6,5 cm		Pantalla 15x9 (cm)	
	Leído (mm)	error absoluto (mm)	Leído (mm)	error absoluto (mm)
135	135	0	134	1
140	139	1	141	-1
145	146	-1	145	0
150	149	1	148	2
155	155	0	156	-1
160	161	-1	163	-3
165	166	-1	164	1
170	169	1	169	1
175	176	-1	174	1
180	181	-1	180	0
185	188	-3	185	0
190	192	-2	191	-1
195	197	-2	197	-2
200	202	-2	200	0
205	206	-1	204	1
210	212	-2	209	1
215	216	-1	213	2
220	221	-1	219	1
225	223	2	224	1
230	229	1	227	3
235	232	3	233	2
240	236	4	240	0
245	240	5	243	2
250	246	4	249	1
255	251	4	252	3
260	258	2	261	-1
265	259	6	264	1
270	264	6	266	4
275	270	5	271	4
280	273	7	277	3
285	276	9	283	2

Tabla 5.17 Cuarta prueba tamaño pantalla Adafruit VL53L0X

Se ha decidido incluir los valores medidos para una pantalla de 9x6,5 cm (Tabla 5.16). De esta forma podemos comparar más fácilmente los errores absolutos obtenidos con ambas pantallas. Como esperábamos, al aumentar el tamaño de la pantalla el error disminuye considerablemente, ahora el error absoluto máximo está alrededor de 4 mm mientras que antes era de 9 mm.

A continuación en la Figura 5.25 se incluye una representación gráfica del error frente a distancia, con el fin de hallar si existe o no una correlación lineal entre ambas variables.

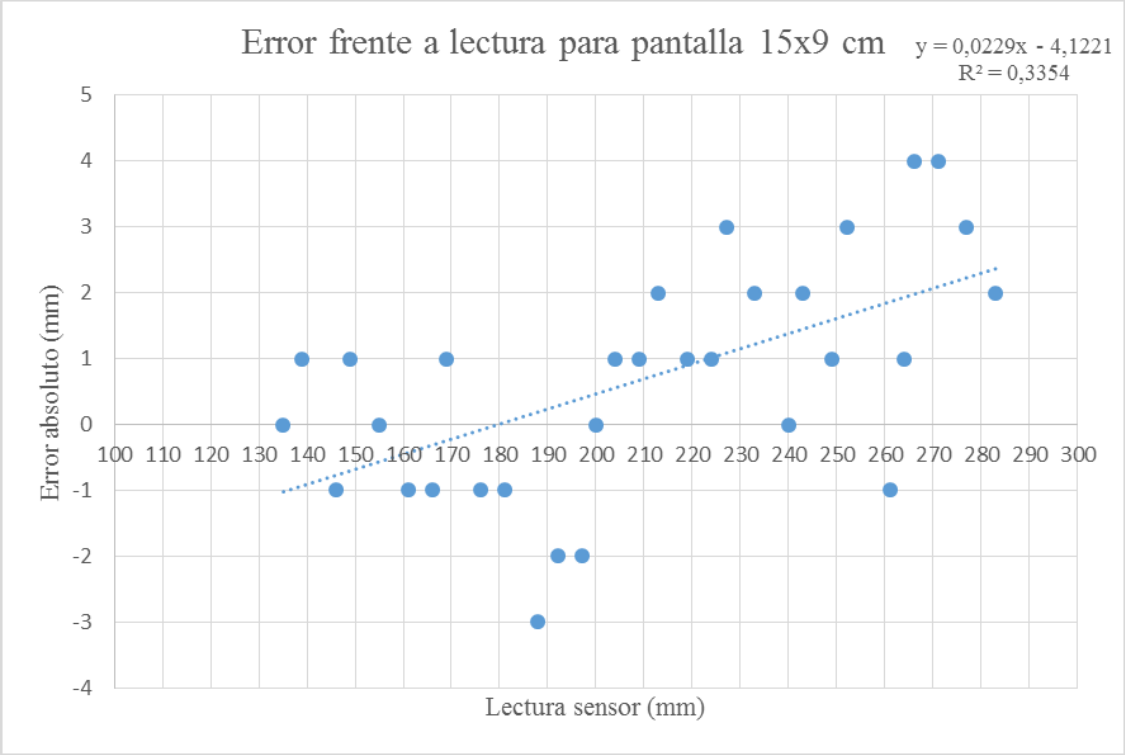


Figura 5.25 Ajuste lineal error frente a lectura sensor Adafruit VL53L0X

Podemos deducir que los datos medidos no se ajustan a una tendencia lineal, ya que el valor de R^2 es de 0,3354. Lo que implica que no somos capaces de realizar una corrección del error precisa para obtener la distancia real.

Por otro lado, se intentó ajustar a un polinomio de grado 4 como en la Figura 5.26:

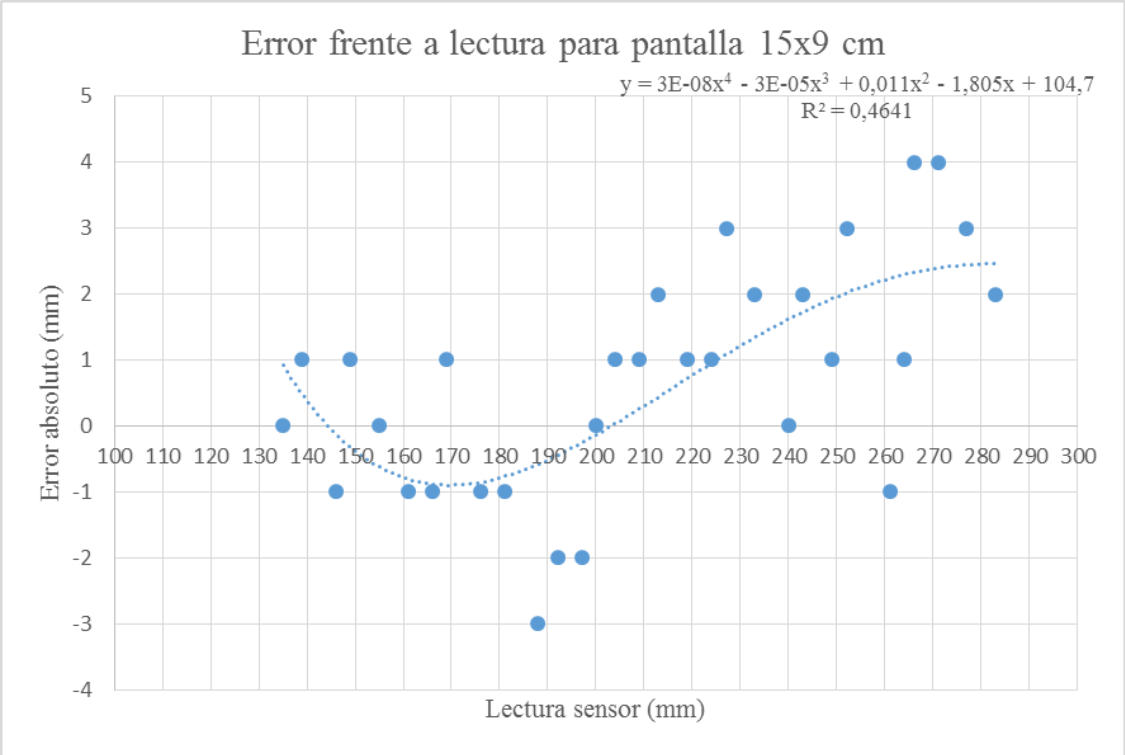


Figura 5.26 Ajuste polinómico error frente a lectura sensor Adafruit VL53L0X

En este caso el valor de R^2 es de 0,4641 que a pesar de ser mejor que el anterior, sigue sin ser válido ya que se considera que los datos ajustan bien a una recta para valores de R^2 comprendidos entre 0,7 y 1.

Éstas últimas figuras solo nos sirven para darnos cuenta que a partir de distancias mayores de 20 cm el valor del error oscila entre ±3 y ±4 mm.

5.3.2.9.2 Adafruit VL6180X

Los problemas asociados a este sensor se detallan a continuación:

5.3.2.9.2.1 Dirección bus I2C

Como hemos comentado anteriormente, este tipo de sensores trabajan con el bus I2C de Arduino. A este bus pueden conectarse más de un dispositivo, el problema es que para poder controlar varios sensores las direcciones del bus I2C no pueden coincidir. La dirección de este sensor es la 0x29 y **NO** se puede modificar. Se han realizado pruebas modificando la dirección del bus del sensor, y a pesar de conseguir modificar la dirección cuando realizamos la lectura de mediciones obtenemos valores que no son coherentes

Esto quiere decir que si queremos utilizar este sensor integrado en cada actuador mecánico del TL-HEX, necesitaríamos seis sensores que comparten la misma dirección del bus I2C. Para solucionar este problema, necesitaríamos integrar un multiplexor del bus I2C .

El multiplexor seleccionado sería el TCA9548A, mostrado en la Figura 5.27:

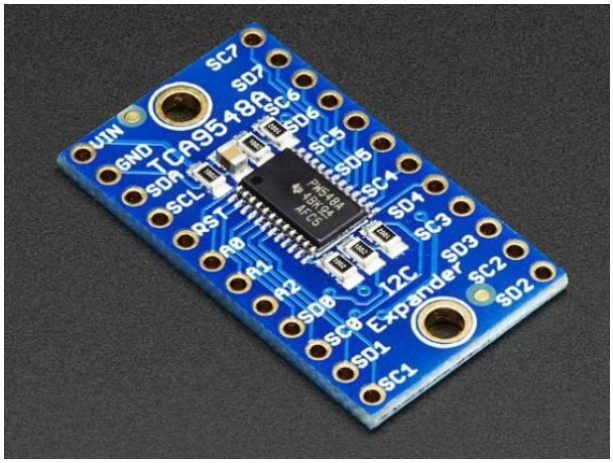


Figura 5.27 Multiplexor TCA9548A

La hoja de características de este multiplexor puede encontrarse en la siguiente dirección:

https://www.cdn-shop.adafruit.com

Este multiplexor nos permite conectar hasta 8 dispositivos que compartan la misma dirección del bus I2C.

El conexionado con Arduino se realiza siguiendo la Figura 5.28:

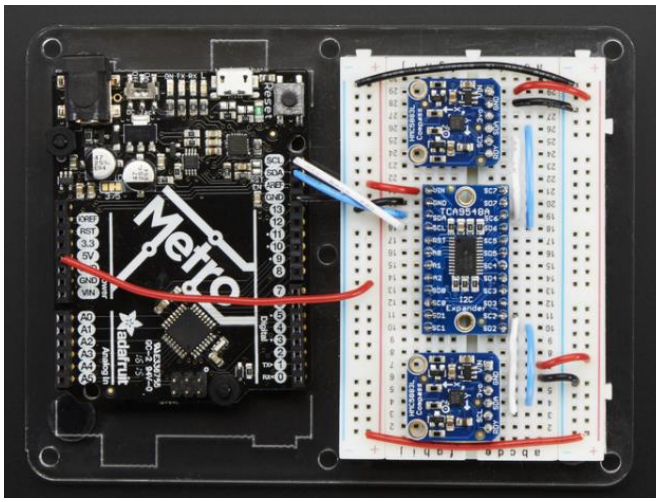


Figura 5.28 Conexionado multiplexor TCA9548A

Como podemos ver en la figura anterior, simplemente debemos realizar las siguientes conexiones:

1. Conectar los pines SCL y SDA del multiplexor con los equivalentes de Arduino
2. Conectar Vin del multiplexor con 5 V
3. Conectar GND del multiplexor con GND de Arduino
4. Conectar los pines SCL y SDA del sensor con SCL1 y SDA1 del multiplexor
5. Conectar los pines GND y Vin con los correspondientes

5.3.2.9.2.2 Pérdida de valores

Tras integrar el sensor VL6180X en el actuador, procedimos a realizar las primeras pruebas. El segundo problema que encontramos al poner en marcha el sensor, es que para posiciones de ajuste rápido menores de 10 el sensor está fuera de rango y nos da valores incoherentes.

Esto quiere decir que con la integración del sensor en el actuador debemos aceptar un rango de pérdida de mediciones, el cual está representado en la Tabla 5.18:

	Longitud total (mm)= 158+A+(80-G) ; siendo A=Rápido y G= Gradual																										
Gradual /Rápido	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80
0	184	183	182	181	180	179	178	177	176	175	174	173	172	171	170	169	168	167	166	165	164	163	162	161	160	159	158
1	185	184	183	182	181	180	179	178	177	176	175	174	173	172	171	170	169	168	167	166	165	164	163	162	161	160	159
2	186	185	184	183	182	181	180	179	178	177	176	175	174	173	172	171	170	169	168	167	166	165	164	163	162	161	160
3	187	186	185	184	183	182	181	180	179	178	177	176	175	174	173	172	171	170	169	168	167	166	165	164	163	162	161
4	188	187	186	185	184	183	182	181	180	179	178	177	176	175	174	173	172	171	170	169	168	167	166	165	164	163	162
5	189	188	187	186	185	184	183	182	181	180	179	178	177	176	175	174	173	172	171	170	169	168	167	166	165	164	163
6	190	189	188	187	186	185	184	183	182	181	180	179	178	177	176	175	174	173	172	171	170	169	168	167	166	165	164
7	191	190	189	188	187	186	185	184	183	182	181	180	179	178	177	176	175	174	173	172	171	170	169	168	167	166	165
8	192	191	190	189	188	187	186	185	184	183	182	181	180	179	178	177	176	175	174	173	172	171	170	169	168	167	166
9	193	192	191	190	189	188	187	186	185	184	183	182	181	180	179	178	177	176	175	174	173	172	171	170	169	168	167
10	194	193	192	191	190	189	188	187	186	185	184	183	182	181	180	179	178	177	176	175	174	173	172	171	170	169	168
11	195	194	193	192	191	190	189	188	187	186	185	184	183	182	181	180	179	178	177	176	175	174	173	172	171	170	169
12	196	195	194	193	192	191	190	189	188	187	186	185	184	183	182	181	180	179	178	177	176	175	174	173	172	171	170
13	197	196	195	194	193	192	191	190	189	188	187	186	185	184	183	182	181	180	179	178	177	176	175	174	173	172	171
14	198	197	196	195	194	193	192	191	190	189	188	187	186	185	184	183	182	181	180	179	178	177	176	175	174	173	172
15	199	198	197	196	195	194	193	192	191	190	189	188	187	186	185	184	183	182	181	180	179	178	177	176	175	174	173
16	200	199	198	197	196	195	194	193	192	191	190	189	188	187	186	185	184	183	182	181	180	179	178	177	176	175	174
17	201	200	199	198	197	196	195	194	193	192	191	190	189	188	187	186	185	184	183	182	181	180	179	178	177	176	175
18	202	201	200	199	198	197	196	195	194	193	192	191	190	189	188	187	186	185	184	183	182	181	180	179	178	177	176
19	203	202	201	200	199	198	197	196	195	194	193	192	191	190	189	188	187	186	185	184	183	182	181	180	179	178	177
20	204	203	202	201	200	199	198	197	196	195	194	193	192	191	190	189	188	187	186	185	184	183	182	181	180	179	178
21	205	204	203	202	201	200	199	198	197	196	195	194	193	192	191	190	189	188	187	186	185	184	183	182	181	180	179
22	206	205	204	203	202	201	200	199	198	197	196	195	194	193	192	191	190	189	188	187	186	185	184	183	182	181	180
23	207	206	205	204	203	202	201	200	199	198	197	196	195	194	193	192	191	190	189	188	187	186	185	184	183	182	181
24	208	207	206	205	204	203	202	201	200	199	198	197	196	195	194	193	192	191	190	189	188	187	186	185	184	183	182
25	209	208	207	206	205	204	203	202	201	200	199	198	197	196	195	194	193	192	191	190	189	188	187	186	185	184	183
26	210	209	208	207	206	205	204	203	202	201	200	199	198	197	196	195	194	193	192	191	190	189	188	187	186	185	184
27	211	210	209	208	207	206	205	204	203	202	201	200	199	198	197	196	195	194	193	192	191	190	189	188	187	186	185

Gradual
Rápido
Valores que se pierden
Valor mínimo que se puede medir
Solapamiento con el strut mediano

Tabla 5.18 Valores medibles Adafruit VL6180X

De la tabla anterior podemos deducir que el rango de posiciones del actuador que no vamos a ser capaces de medir es [158 – 178] mm.

5.3.2.10 Conclusiones

5.3.2.10.1 Adafruit VL53L0X

Este tipo de sensor necesita un filtrado de la señal, lo que implica que el software es bastante más complejo e introduce un retardo a la hora de visualizar el valor de la medida de entre 5 y 10 segundos para este tipo de tarjeta Arduino.

Que la pantalla y el sensor sean desmontables, supone una ventaja, ya que en lugar de necesitar seis sensores, podríamos utilizar sólo un sensor. Sin embargo, la integración en el sistema TL-Hex también supone una gran desventaja, ya que como hemos visto en el apartado anterior, para obtener medidas con un error alrededor de 3-4 mm se necesita una pantalla muy voluminosa (15x9,5 cm).

No hay posibilidad de reducir más el error, ya que éste no sigue una tendencia lineal ni polinómica. Es más, si volvemos a realizar la toma de datos obtenemos una recta o polinomio de grado 4 distintos. Lo que nos lleva a la conclusión de que éste sensor no ofrece una lectura de datos estables para esta aplicación.

Por todo lo indicado, el sensor VL53L0X ha sido descartado por no ser útil para nuestra aplicación.

5.3.2.10.2 Adafruit VL6180X

Este sensor cumple el requisito de ser poco voluminoso y de "fácil" integración en el sistema TL-Hex. Sin embargo, tiene una par de inconvenientes que nos llevaron a descartar el sensor.

El primer inconveniente es que se necesitaría otro sensor para la medida del ajuste gradual. Además necesitaríamos hasta seis sensores que compartan la misma dirección del bus, por tanto necesitaríamos el multiplexor TCA9548A mencionado anteriormente. A esto hay que sumarle otros seis sensores para la medida del ajuste de gradual, lo que finalmente supone que necesitaríamos un total de doce sensores.

El segundo inconveniente es que perdemos unos 2 cm en las lecturas del ajuste rápido. Aunque debemos destacar que podemos no contar con él ya que lo que nos interesa es la medición del ajuste gradual. La medida del ajuste rápido se deja para trabajos futuros.

5.3.3 Sensor Ultrasonidos

5.3.3.1 ¿Qué es y cómo funciona?

Los ultrasonidos son sonido a una frecuencia mayor que la máxima audible por el oído humano. Ésta comienza desde unos 16 Hz y tiene un límite superior aproximadamente de 20 KHz, mientras que con el sensor vamos a utilizar sonido con una frecuencia de 40 KHz [25, 26].

El funcionamiento básico de los ultrasonidos como medidores de distancia se muestra en la Figura 5.29, donde se tiene un transmisor que emite un pulso de ultrasonido que rebota sobre un determinado objeto y la reflexión de ese pulso es detectada por un receptor.

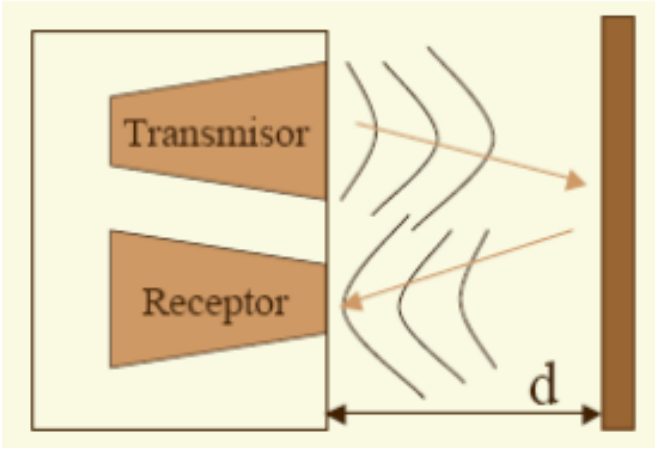


Figura 5.29 Funcionamiento sensor ultrasonidos

La mayoría de los sensores por ultrasonidos se basan en la emisión de un pulso de ultrasonido cuyo campo de acción es de forma cónica. Midiendo el tiempo que transcurre entre la emisión del sonido y la percepción del eco se puede establecer la distancia a la que se encuentra el obstáculo que ha producido la reflexión de la onda sonora, mediante la fórmula:

$$e = \frac{1}{2}vt$$

Donde e es el espacio recorrido, v es la velocidad del sonido en el aire y t es el tiempo transcurrido entre la emisión y la recepción del pulso.

5.3.3.2 Sensor ultrasonidos estudiado

A continuación se adjunta en la Tabla 5.19 un resumen del sensor por ultrasonidos elegido:

Características	**Valor**	**Observaciones**
Modelo	Ultra Sonic range measurement module	Este sensor es ideal para aplicaciones en la que la precisión requerida es de 1 cm. Para nuestro caso concreto no es válido.
Dimensiones	43x20x15 mm	
Alcance	400 cm	
Tipo	Ultrasonidos	
Voltaje alimentación	5 V	
Interfaz de datos	Pin digital	
Pruebas		
Cantidad de datos adquiridos	37	
Precisión en la medida	1 cm	
Coeficiente determinación (R^2)	0,0437	
Filtrado	Mediana	

Tabla 5.19 Características sensor ultrasonidos

La hoja de características de este sensor puede encontrarse en la siguiente dirección:

http://www.fritzing.org

En la Figura 5.30 se puede ver el sensor por ultrasonidos estudiado:

Figura 5.30 Sensor ultrasonidos

En la Figura 5.31 se adjunta el diagrama de conexionado necesario:

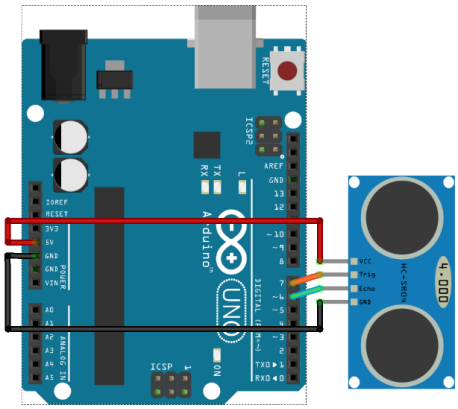


Figura 5.31 Conexionado sensor ultrasonidos

Por tanto, en la Figura 5.32 podemos ver cómo quedaría la conexión real entre el Arduino y el sensor por ultrasonidos:

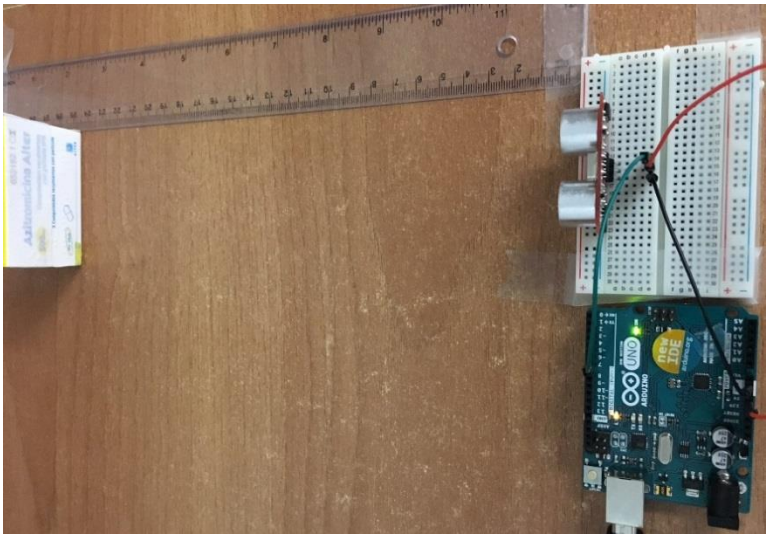


Figura 5.32 Conexionado real sensor ultrasonidos

5.3.3.3 Problemas

Entre los diversos factores que alteran las lecturas que se realizan con este sensor cabe destacar [26]:

- El campo de actuación del pulso que se emite desde un transductor de ultrasonido tiene forma cónica. El eco que se recibe como respuesta a la reflexión del sonido indica la presencia del objeto más cercano que se encuentra dentro del cono acústico y no especifica en ningún momento la localización angular del mismo. No podemos descartar la probabilidad de que el eco se produzca por un objeto en la periferia del eje central.

  En la Figura 5.33 extraída de la hoja de características del sensor podemos ver que existe una incertidumbre angular en la toma de datos. Además la detección más precisa será en la que el objeto reflector se encuentre dentro del campo de acción de 30^{O}.

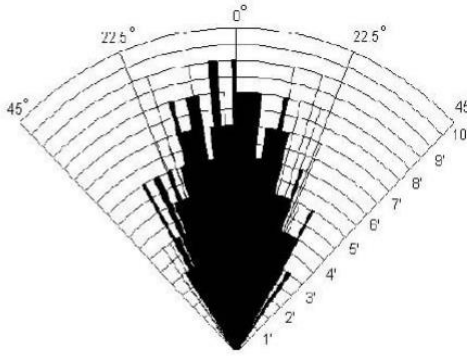


Figura 5.33 Campo de actuación por ultrasonidos

- Tras la emisión del ultrasonido se espera un determinado tiempo (tiempo de relajación) aproximado de unos 50 ms a que las vibraciones en el sensor desaparezcan y esté preparado para recibir el eco producido por el obstáculo. Esto implica que existe una distancia mínima a partir de la cual el sensor mide con precisión.

  En la Figura 5.34 extraída de la hoja de características del sensor podemos deducir cual será la distancia mínima capaz de detectar el sensor:

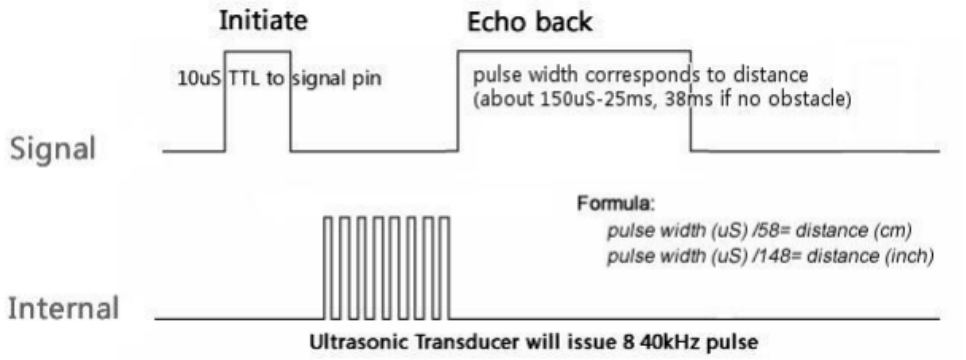


Figura 5.34 Funcionamiento interno sensor ultrasonidos

El ancho de pulso mínimo tiene un valor de 150 μs, por lo que la distancia mínima será:

$$distancia\ mínima = \frac{150}{58} = 2{,}58\ \approx 3cm$$

5.3.3.4 Software Arduino

El siguiente código mide el ancho de pulso en microsegundos y luego lo divide entre 58 (fórmula en la Figura 5.35). Este valor lo almacena en una posición de un array hasta un total de 21 posiciones. Posteriormente, aplica el filtro mediana móvil y devuelve el valor filtrado por pantalla:

```
#include "Arduino.h"
#include "MedianFilterLib.h"
double values[21];
int i;
size_t valuesLength = sizeof(values) / sizeof(values[0]);
double getMeasure()
{
  size_t static index = 0;
  index++;
  return values[index - 1];
}
class Ultrasonic
{
    public:
    Ultrasonic(int pin);
    void DistanceMeasure(void);
     private:
    int _pin=7;//pin number of Arduino that is connected with SIG pin of Ultrasonic Ranger.
    double duration;// the Pulse time received;
};
Ultrasonic::Ultrasonic(int pin)
{
    _pin = pin;
}
void Ultrasonic::DistanceMeasure(void)
{
for(i=0; i<21;i++){
    pinMode(_pin, OUTPUT);
    digitalWrite(_pin, LOW);
    delayMicroseconds(2);
    digitalWrite(_pin, HIGH);
    delayMicroseconds(5);
    digitalWrite(_pin,LOW);
    pinMode(_pin,INPUT);
     duration = pulseIn(_pin,HIGH);
    Serial.print("Pulse width in uS: ");
    Serial.println(duration);
    //pasa el valor de uS a mm:
    Serial.print("distance to the box: ");
    double a=duration/58;
    Serial.print(a);
    Serial.println(" cm");
    values[i]= a;
    Serial.print("El valor de la posicion "); Serial.print(i);Serial.print(" es:");
    Serial.println(values[i]);
```

```
   delay(1000);
}
}
MedianFilter<double> medianFilter(5);
Ultrasonic ultrasonic(7);

void setup() {
Serial.begin(9600);
ultrasonic.DistanceMeasure();
delay(2000);
// Filtro MEDIANA
float timeMean = 0;
  for (size_t iCount = 0; iCount < valuesLength; iCount++)
  {
    double rawMeasure = getMeasure();
    unsigned long timeCount = micros();
    double median = medianFilter.AddValue(rawMeasure);
    timeCount = micros() - timeCount;
    timeMean += timeCount;
    Serial.print(rawMeasure);
    Serial.print(",");
    Serial.println(median);
  }
  Serial.println(timeMean / valuesLength);
}

void loop()
{
}
```

Ejecutando este código obtenemos los siguientes resultados de la Tabla 5.20:

regla (mm)	leído(mm)	error absoluto (mm)	regla (mm)	leído(mm)	error absoluto (mm)
140	141,9	1,9	235	238,3	3,3
145	146,2	1,2	240	244	4
150	150,7	0,7	245	249,6	4,6
155	155,3	0,3	250	253,6	3,6
160	161,4	1,4	255	255,7	0,7
165	165,8	0,8	260	258,9	1,1
170	173,7	3,7	265	264,2	0,8
175	176,1	1,1	270	269,2	0,8
180	183,2	3,2	275	273,8	1,2
185	191,8	6,8	280	279,7	0,3
190	195,4	5,4	285	283,8	1,2
195	201	6	290	288,3	1,7
200	201,1	1,1	295	293,9	1,1
205	204,6	0,4	300	298,5	1,5
210	215,4	5,4	305	307,3	2,3
215	223,4	8,4	310	310,2	0,2
220	220,7	0,7	315	315,3	0,3
225	226,6	1,6	320	320,3	0,3
230	231,1	1,1			

Tabla 5.20 Prueba de medida sensor ultrasonidos

A primera vista, podemos destacar que el error en las medidas es mayor que en el caso de los sensores LIDAR.

Por otro lado, a pesar de que hayamos intentado tomar medidas con precisiones de 1 mm, este sensor no está pensado para ser utilizado en aplicaciones que requieran tales precisiones. Además de que a la hora de ejecutar el programa este a veces daba lecturas bastante incorrectas que han sido suprimidas.

Los valores de la tabla anterior los podemos representar gráficamente para facilitar la interpretación de los resultados, Figura 5.35:

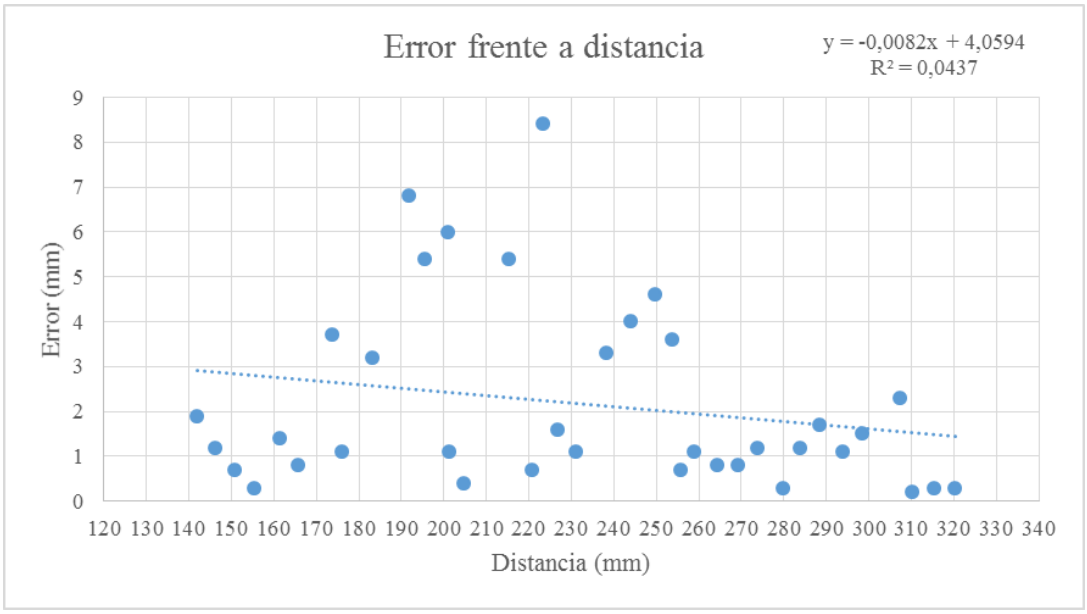


Figura 5.35 Error frente a distancia sensor ultrasondios

Por último, simplemente observando la figura anterior destacar que las variables representadas no se ajustan a una tendencia lineal, siendo el valor de R^2 de 0,0437.

5.3.3.5 Conclusiones

Este sensor no es válido para nuestra aplicación ya que no podemos conseguir precisiones menores de 1 cm. Además, cuando intentamos obtener lecturas con precisiones de milímetro, éste sensor se vuelve inestable ya que no está diseñado para tal fin.

Por otro lado, tampoco se ajusta a una tendencia lineal lo que nos impide intentar corregir el error producido en las lecturas.

5.3.4 Sensor de Efecto Hall

5.3.4.1 ¿Qué es y cómo funciona?

Un sensor Hall es un dispositivo que nos permite realizar mediciones de campo magnético.

Una ventaja importante es que realizan la medición a distancia, sin necesidad de contacto físico. Aunque su alcance es limitado (típicamente pocos centímetros) esto supone que apenas presentan desgaste mecánico. Además son inmunes a ruidos y polvo, lo que los convierte en sensores fiables y duraderos.

En general, encontramos dos tipos de sensores Hall:

- Analógicos. Generan una salida proporcional a la intensidad de campo magnético. Empleados para medir la intensidad de un campo magnético.
- Digitales. Proporcionan un valor Alto en presencia de campo magnético, y bajo en ausencia del mismo. Por tanto, son empleados para detectar la existencia de campos magnéticos.

Nosotros nos centraremos en los sensores Hall de tipo analógico, ya que nos interesa medir la intensidad de campo magnético proporcional a diferentes distancias.

Su principio de funcionamiento es el efecto Hall. Al hacer circular una corriente eléctrica a lo largo de un semiconductor en presencia de un campo magnético, los electrones son desviados por efecto del campo magnético, dando lugar a una tensión perpendicular a la corriente y al campo magnético. En la Figura 5.36 podemos ver gráficamente este comportamiento:

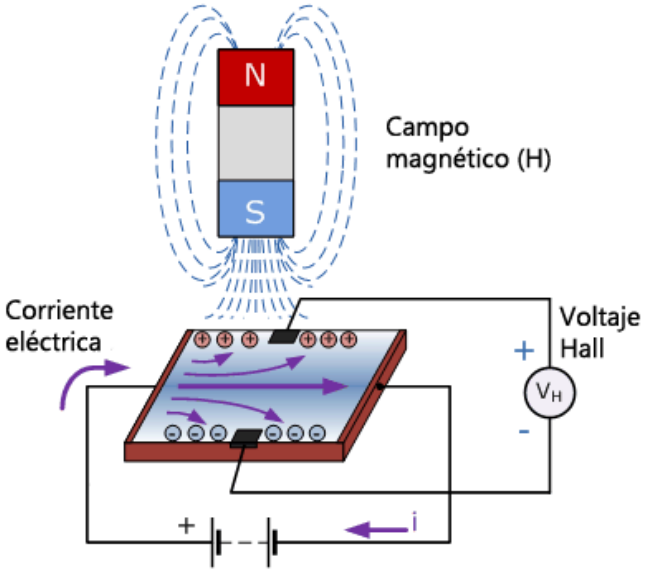


Figura 5.36 Funcionamiento sensor de efecto Hall

Midiendo esta tensión originada por el efecto Hall pretendemos hallar la relación existente entre el voltaje y la distancia a la que se encuentra el imán que previamente habremos colocado al objeto que deseamos conocer a que distancia se encuentra [27].

5.3.4.2 Sensor efecto hall estudiado

A continuación se adjunta en la Tabla 5.21 un resumen del sensor elegido:

Características	Valor	Observaciones
Modelo	CYL49E	Este tipo de sensor ha sido descartado porque no pensamos que sea el más adecuado para nuestra aplicación.
Dimensiones	4,06x1,5x5,8 mm	
Máxima distancia detección	indeterminada	
Tipo	Lineal	
Sensibilidad	14-18 mV/mT	
Interfaz de datos	Analógica	

Tabla 5.21 Características sensor efecto Hall

La hoja de características de este sensor puede encontrarse en la siguiente dirección:

http://www.hallsensors.de

En la Figura 5.37 se puede ver el sensor de efecto Hall CYL49E:

Figura 5.37 Sensor de efecto Hall CYL49E

En la Figura 5.38 se adjunta el diagrama de conexionado necesario para utilizar este sensor junto a un Arduino:

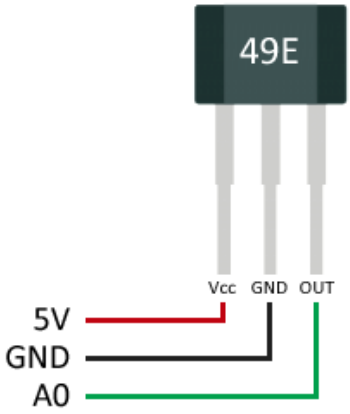


Figura 5.38 Conexionado sensor efecto Hall

Estos sensores Hall incorporan la electrónica necesaria para dar respuesta de tensión lineal en el rango de -100 a 100 mT. Los circuitos están diseñados para minimizar el ruido de la señal, por lo que no es necesario filtrado externo [27].

De la hoja de características podemos obtener la gráfica de la Figura 5.39:

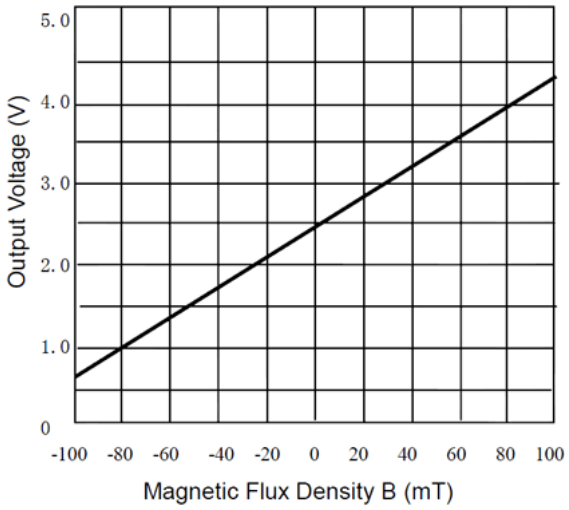


Figura 5.39 Voltaje frente a densidad de campo magnético

Interpolando en la gráfica anterior podemos obtener la siguiente expresión para la respuesta en tensión respecto al flujo magnético medido por el sensor:

$$V = 0{,}0188 \cdot B + 2{,}5$$

5.3.4.3 Software Arduino

Para poder obtener la distancia a la que se encuentra el imán, en primer lugar mediríamos el voltaje mediante una entrada analógica de Arduino. A continuación, convertiríamos este voltaje en distancia siguiendo un proceso análogo al caso del potenciómetro.

```
const int pinHall = A0;

void setup() {
  pinMode(pinHall, INPUT);
  Serial.begin(9600);
}

void loop() {
  //media de 10 medidas para filtrar ruido
  long measure = 0;
  for(int i = 0; i < 10; i++){
     int value =
     measure += analogRead(pinHall);
  }
  measure /= 10;
  //cálculo del voltaje en mV
  float outputV = measure * 5000.0 / 1023;
  Serial.print("Output Voltaje = ");
  Serial.print(outputV);
  Serial.print(" mV   ");
```

5.3.4.4 Problemas y conclusiones

A pesar de tener toda la documentación necesaria para realizar las pruebas con este sensor, se ha decidido descartarlo directamente. No pensamos que pueda cumplir los requisitos de precisión ni que pueda detectar un imán a una distancia de 8 cm.

5.3.5 Sensor Láser de Triangulación

5.3.5.1 ¿Qué es y cómo funciona?

Este tipo de sensores también son conocidos como "telémetro láser". Este dispositivo emite un rayo láser que rebota contra un objeto y es capaz de medir el tiempo que transcurre desde que el rayo abandona el dispositivo y vuelve a regresar.

En este apartado nos centraremos en explicar sólo cómo funciona el sensor seleccionado, debido a que es el único compatible con Arduino que integra tecnología láser.

El módulo seleccionado utiliza triangulación óptica (Figura 5.40) para hacer el cálculo de las distancias, utilizando trigonometría simple entre el punto central de la luz láser, la cámara y el objeto.

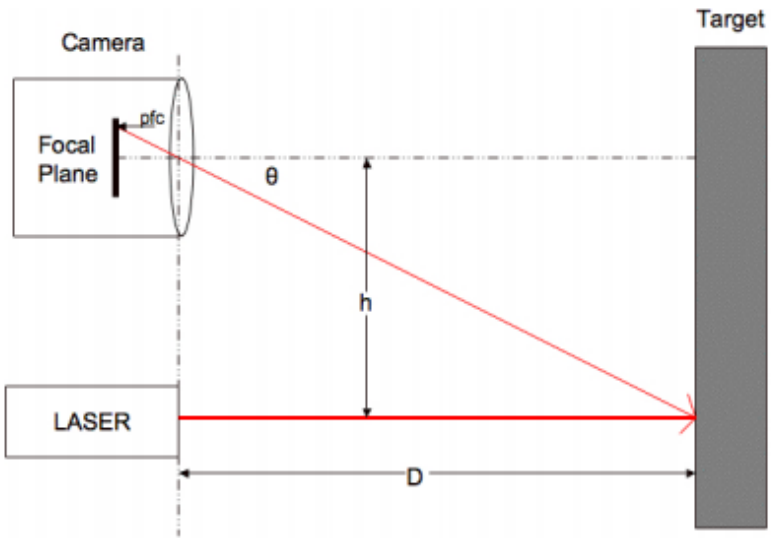


Figura 5.40 Funcionamiento sensor láser de triangulación

Refiriéndonos a la Figura anterior, un diodo láser emite un punto sobre el objeto al que queremos hallar la distancia. El valor de h está fijado, es la distancia entre el centro del punto láser y la cámara (de unos 78 mm en el módulo estudiado). Cuando la distancia a la que nos encontramos del objeto cambia, también lo hace el ángulo Θ y el valor pfc (pixeles desde el centro), que es el número de pixeles entre el punto central de la cámara y el punto láser.

Cuando el objeto se acerca, el valor de pfc y de Θ aumentan. Mientras que si el objeto se aleja, estos valores se aproximan a 0. Si el ángulo Θ es conocido, entonces podemos usar trigonometría para calcular la distancia D:

$$D = h / \tan\theta$$

La Figura 5.41 muestra la relación entre pfc (eje de las x) y el ángulo Θ (eje de las y) para el caso de nuestro módulo.

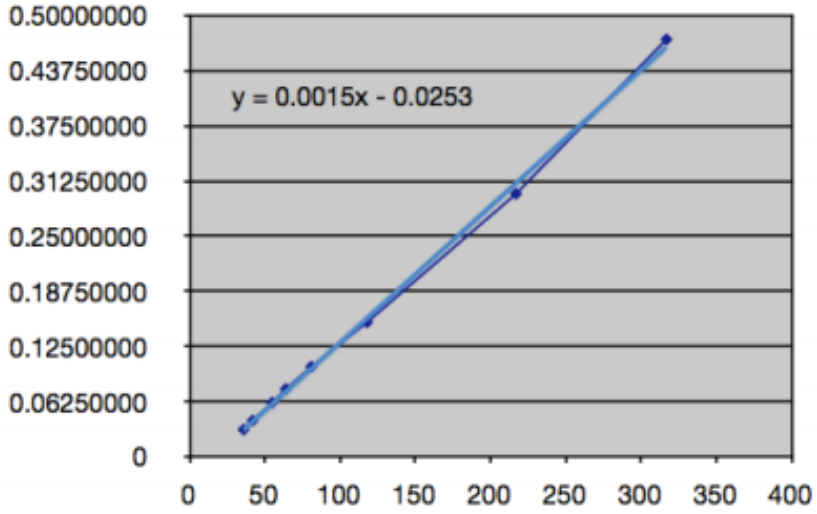


Figura 5.41 Pfc frente a ángulo Θ

De esta gráfica podemos obtener la regresión lineal que mejor ajusta (recta azul clarita) para los valores medidos (recta azul oscuro), siendo:

$$y = 0{,}0015x - 0{,}0253$$

De esta forma ya tenemos establecida la relación entre pfc y el ángulo Θ.

5.3.5.2 Sensor láser estudiado

A continuación se adjunta en la Tabla 5.22 un resumen del sensor elegido:

Características	Valor	Observaciones
Modelo	Parallax Laser Range Finder (LRF)	Módulo compacto con cámara CMOS y diodo láser. Cálculo de distancias por triangulación óptica. Descartado por ser voluminoso
Dimensiones	10,05x 3,95x 1,7 cm	
Rango de medidas óptimo	15–122 cm	
Máxima distancia detección	2,4 m	
Tipo	Láser	
Error precisión	< 5%	
Interfaz de datos	Puerto serie	

Tabla 5.22 Características sensor láser de triangulación

La hoja de características de este sensor puede encontrarse en la siguiente dirección:

https://www.parallax.com

En la Figura 5.42 se puede ver el sensor "Parallax Laser Range Finder (LRF)":

Figura 5.42 Sensor "Parallax Laser Range Finder"

En la Figura 5.43 se adjunta el diagrama de conexionado necesario para utilizar este sensor junto a Arduino:

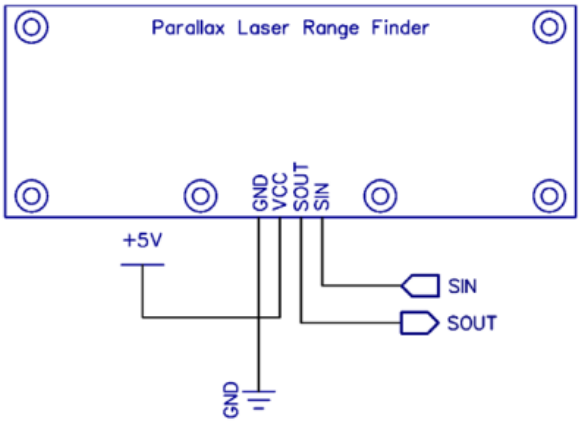


Figura 5.43 Conexionado sensor láser de triangulación

5.3.5.3 Problemas y conclusiones

Este sensor ha sido descartado por ser muy voluminoso, lo que conlleva una difícil integración en el sistema TL-Hex. Además que el objetivo al que queremos hallar la distancia debe ser lo suficientemente grande como para que pueda incidir el rayo láser y el punto central de la cámara (recordemos que están espaciados entre si 78 mm).

5.4 Selección de Sensores

Finalmente, los sensores seleccionados son los potenciómetros de tipo lineal, ya que no presentan el inconveniente del ruido y el retraso de presentación de datos es despreciable comparado con los sensores LIDAR. Además integrar el LIDAR VL6180X entubado para medir el ajuste GRADUAL era complejo y no podemos admitir un retraso de presentación de casi 10 segundos cada vez que queramos ver un valor de medida en la interfaz.

5.5 Montaje e Integración de los Sensores en los Actuadores

En las Figuras 5.44 y 5.45, se pueden ver distintas perspectivas del potenciómetro deslizante integrado en el actuador 4 del TL-Hex:

Figura 5.44 Primera perspectiva integración potenciómetro deslizante en el TL-Hex

Figura 5.45 Segunda perspectiva integración potenciómetro deslizante en el TL-Hex

Se puede ver en más detalle en las Figuras 5.46, 5.47, 5.48:

Figura 5.46 Vista posterior de integración potenciómetro deslizante

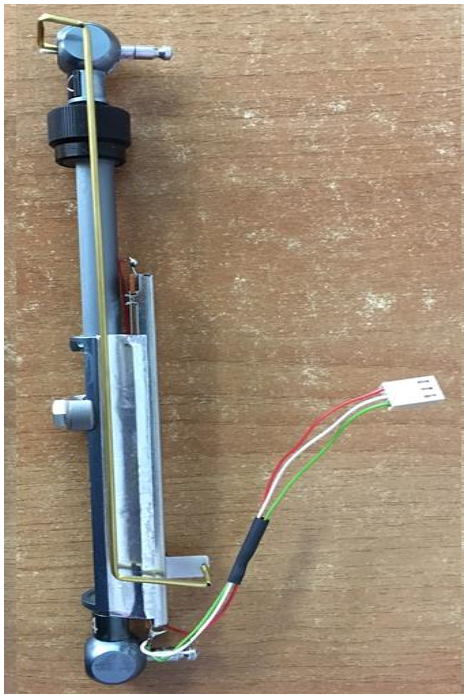


Figura 5.47 Vista lateral de integración potenciómetro deslizante

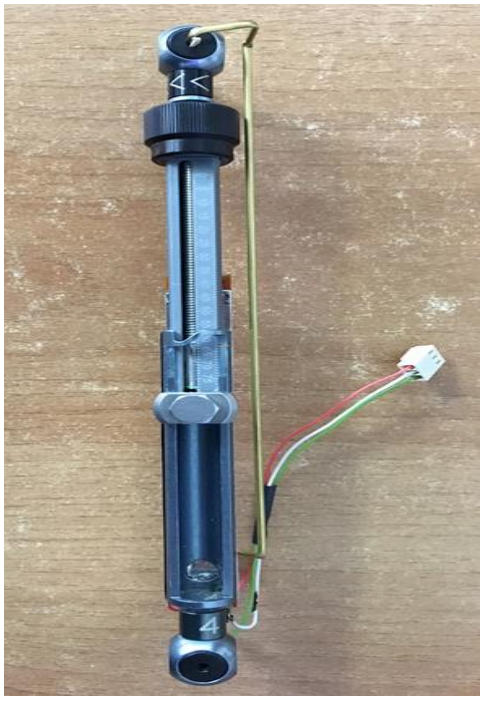


Figura 5.48 Vista frontal de integración potenciómetro deslizante

Mientras que en la Figura 5.49 podemos ver el potenciómetro de hilo integrado en el actuador 5 del TL-Hex:

Figura 5.49 Integración potenciómetro hilo en el TL-Hex

Se puede ver en más detalle en la Figura 5.50:

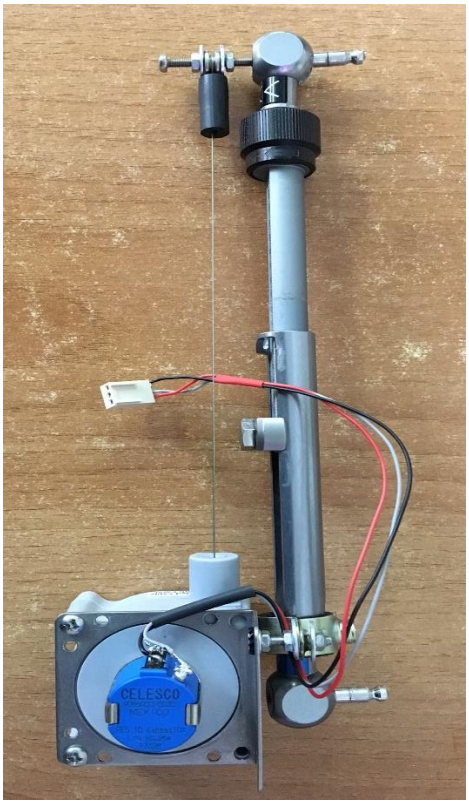


Figura 5.50 Detalle de integración potenciómetro hilo

Por otro lado, a fin de completar la información sobre la integración de los potenciómetros en el sistema TL-HEX , se analizó el peso adicional que suponía integrar cada potenciómetro.

Lo primero fue hallar el peso del actuador mecánico, según la Figura 5.51:

Figura 5.51 Peso actuador mecánico

Como podemos ver su peso es de 96.9 g.

El peso de la varilla metálica que se necesita para la integración del potenciómetro deslizante lo podemos ver en la Figura 5.52:

Figura 5.52 Peso varilla metálica

Como podemos ver su peso es de 3 g.

Mientras que el peso del potenciómetro deslizante es el indicado en la Figura 5.53:

Figura 5.53 Peso potenciómetro deslizante

Como podemos ver su peso es de 22.2 g.

El peso del potenciómetro deslizante integrado en el actuador lo podemos ver en la Figura 5.54:

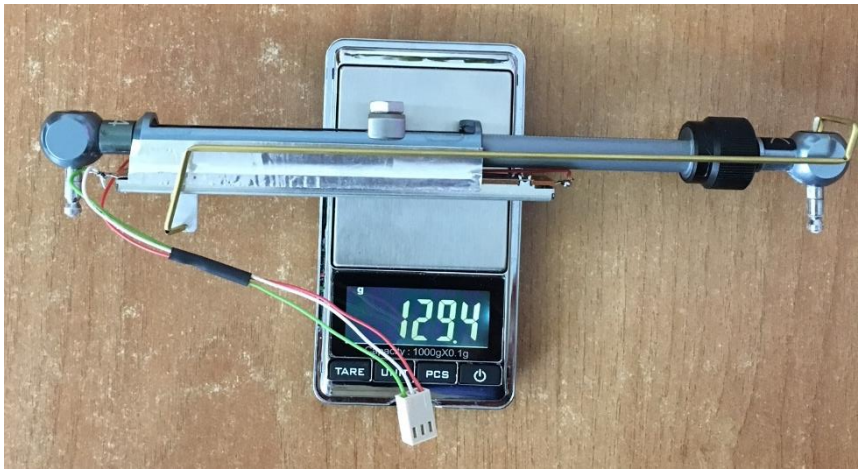


Figura 5.54 Peso integración potenciómetro deslizante

Como podemos el peso con el potenciómetro deslizante se incrementa a 129.4 g.

Por último, el peso del potenciómetro de hilo integrado en el actuador se puede ver en la Figura 5.55:

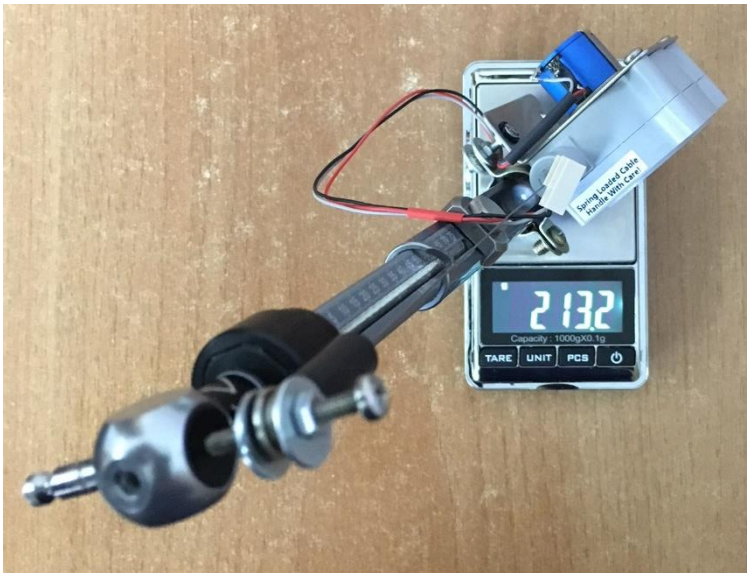


Figura 5.55 Peso integración potenciómetro de hilo

A modo de resumen, se adjunta en la Tabla 5.23 los pesos indicados en las figuras anteriores:

Tipo Integración	Peso (g)	Incremento peso (%)
Actuador	96,9	0
Actuador y potenciómetro deslizante	129,4	33,539
Actuador y potenciómetro hilo	213,2	120,021

Tabla 5.23 Peso según el tipo de integración

Como conclusión de la tabla anterior, podemos decir que la integración del potenciómetro deslizante supone un incremento de peso bastante inferior al que supone integrar el potenciómetro de hilo. Si en futuros trabajos se decide realizar la integración de seis potenciómetros, esto será un factor a tener en cuenta. El peso total de los seis actuadores sin ningún potenciómetro es de 581,4 g. El peso total de los seis actuadores con un potenciómetro deslizante integrado en cada uno sería de 776,4 g; mientras que si se decide integrar los potenciómetros de hilo el peso asciende a 1279,2 g.

Adicionalmente debemos tener en cuenta que el volumen que adiciona el potenciómetro deslizante es muy inferior al del potenciómetro de hilo algo que lo hace aún una mejor opción para la integración futura.

Capítulo 6

Desarrollo de la Interfaz para Lectura y Comunicación

6.1 Introducción

En este capítulo se incluye toda la información relacionada con el hardware y el software que permite la comunicación inalámbrica con los sensores y el desarrollo de una aplicación para su lectura.

El hardware utilizado es una tarjeta Arduino Uno Rev 3, una plataforma de hardware libre, basada en una placa con un microcontrolador y un software que consiste en un entorno de desarrollo (IDE) que implementa el lenguaje de programación de Arduino [28].

La aplicación para la lectura de datos se desarrolló en "Processing". Éste es un lenguaje de programación y entorno de desarrollo integrado de código abierto basado en Java que cuenta con modificaciones para simplificar la programación [29].

La comunicación inalámbrica se hará mediante "Bluetooth".

6.2 Comunicación Bluetooth

La idea de integrar una conexión Bluetooth en nuestro proyecto nos abre unas posibilidades enormes, ya que hoy en día todos disponemos de un teléfono móvil en el bolsillo que incluirá un sistema operativo como Android o Apple IOS.

Pretendemos poder visualizar el estado de los actuadores mecánicos desde el móvil o el laptop.

6.2.1 Las redes Bluetooth

Los dispositivos Bluetooth pueden actuar como Maestro “Masters” o como Esclavos “Slaves”. La diferencia es que un esclavo solo puede conectarse a un maestro y a nadie más; mientras que un maestro, puede conectarse a varios esclavos para solicitar y recibir información de todos ellos (hasta un máximo de siete) [30].

Cada uno de los dispositivos que se identifican vía Bluetooth presenta una dirección única de 48 bits y además un nombre de dispositivo que nos sirve para identificarlo. Habitualmente, también incluyen un PIN de conexión o número de identificación que debe teclearse para conseguir acceso al mismo.

6.2.2 Módulos Bluetooth disponibles para Arduino

Los más frecuente son los módulos HC-05 y HC-06 que están disponibles independientes o en modo *shield*.

El hardware de ambos módulos es el mismo, pero el software que incorporan es diferente. Como se puede ver en la Figura 6.1, el módulo de conexión se monta sobre un soporte que tiene una diferencia obvia, el número de pines.

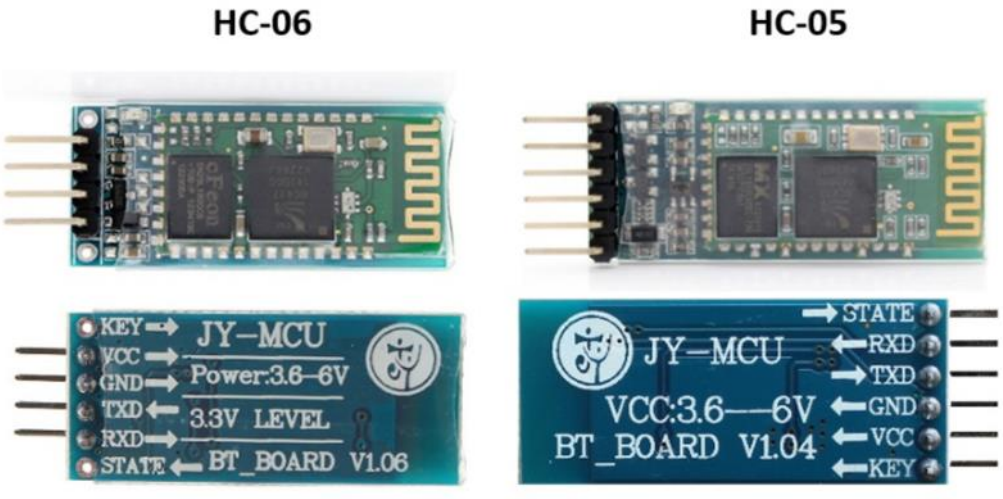


Figura 6.1 Módulos bluetooth

El modelo HC-06 dispone de 4 pines, un juego reducido de instrucciones a las que atiende y solo puede actuar como esclavo. Mientras que el módulo HC-05 dispone de 6 pines, puede actuar como maestro o esclavo y atiende a un mayor número de órdenes.

Estos módulos aceptan órdenes del tipo "AT+Orden", donde AT es el comando especifico de atención que significa que a continuación viene una orden de programación. Sirven para configurar el módulo Bluetooth a través de un microcontrolador (en nuestro caso Arduino), un ordenador o con cualquier dispositivo que posea una comunicación serie (Tx/Rx). Estas instrucciones permiten cambiar los baudios del módulo, el PIN, el nombre, etc… Según las especificaciones, el tiempo que se tiene que respetar entre el envió de un comando AT y otro tiene que ser de 1 segundo. Si se envía un comando AT y en menos de un segundo se envía otro, el módulo no devuelve respuesta.

6.2.3 Módulo HC-06

Finalmente, se decidió utilizar este módulo bluetooth debido a que era más sencillo de utilizar y no nos dió los inconvenientes relacionados con el HC-05. En la Tabla 6.1 podemos ver las características del módulo HC-06:

Características	Valor	Observaciones
Modelo	HC-06	Se ha seleccionado este módulo por ser más sencillo de utilizar y porque necesitamos un esclavo.
Rango de voltaje	3.6 a 6 V	
Dimensiones (cm)	1.7 x 4	
Tipo	Bluetooth v2.0 + EDR (Enhanced Data Rate)	
Alcance (m)	5 a 10	
Consumo (mA)	30 a 40	
Velocidad	Asincrónica: 2 Mbps (max.)/160 kbps	
	Sincrónica: 1 Mbps/1 Mbps	
Pruebas		
Cantidad de datos tomados	100	

Tabla 6.1 Características módulo bluetooth HC-06

Podemos encontrar información más detallada en el siguiente enlace:

https://www.hobbytronics.co.za

6.2.3.1 Conexión con Arduino

La conexión entre el módulo HC-06 y Arduino se realiza siguiendo la Figura 6.2:

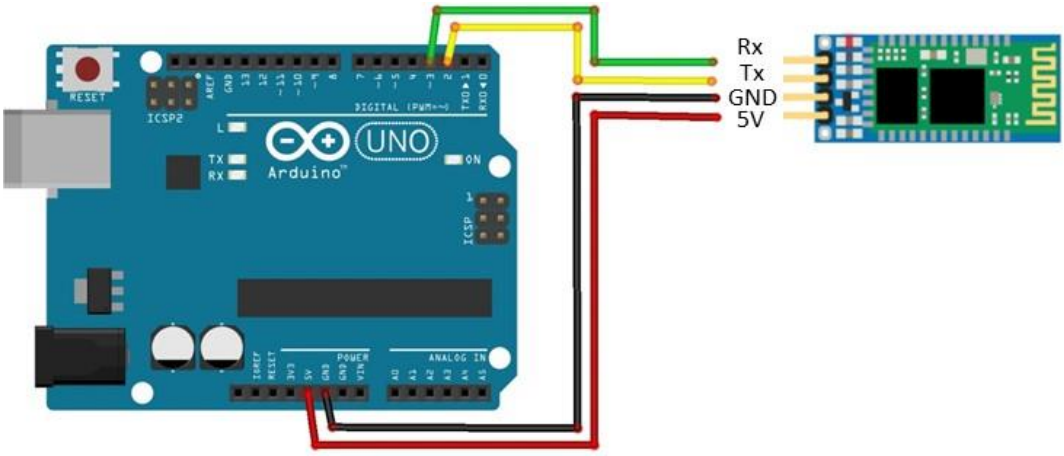


Figura 6.2 Conexionado módulo HC-06

NOTA: Esta conexión SÓLO es necesaria cuando deseemos configurar el bluetooth. Una vez configurado el Tx del módulo (cable amarillo) podrá conectarse al Rx de la placa Arduino; mientras que el Rx del módulo (cable verde) podrá conectarse al Tx de la plaza Arduino. Cada vez que se actualice el software Arduino (no el de programación del módulo HC-06) hay que desconectar estos dos últimos cables ya que los pines Tx y Rx de la placa Arduino se utilizan en la conexión serie con el pc.

Al hacer esta conexión, el LED del módulo HC-06, parpadea continuamente. Esto indica que no está pareado o vinculado.

Cuando conectamos algo al módulo, esta luz se quedará fija y es la forma de saber si hay conexión o no.

6.2.4 Comandos AT

A continuación en la Tabla 6.2 se incluyen los comandos AT capaces de reconocer el módulo hc-06:

Comando	Respuesta	Comentario
AT	OK	Test de comunicación
AT+VERSION	OKlinvorV1.8	Versión del Firmware
AT+NAMEmyBTmodule	OKsetname	Fija el nombre del módulo a "myBTmodule"
AT+PIN6789	OKsetPIN	Fija el PIN a 6789
AT+BAUD1	OK1200	Fija la velocidad a 1200 Baudios
AT+BAUD2	OK2400	Fija la velocidad a 2400 Baudios
AT+BAUD3	OK4800	Fija la velocidad a 4800 Baudios
AT+BAUD4	OK9600	Fija la velocidad a 9600 Baudios
AT+BAUD5	OK19200	Fija la velocidad a 19200 Baudios
AT+BAUD6	OK38400	Fija la velocidad a 38400 Baudios
AT+BAUD7	OK57600	Fija la velocidad a 57600 Baudios
AT+BAUD8	OK115200	Fija la velocidad a 115200 Baudios
AT+BAUD9	OK230400	Fija la velocidad a 230400 Baudios
AT+BAUDA	OK460800	Fija la velocidad a 460800 Baudios
AT+BAUDB	OK921600	Fija la velocidad a 921600 Baudios
AT+BAUDC	OK1382400	Fija la velocidad a 1382400 Baudios

Tabla 6.2 Comandos AT módulo HC-06

6.2.5 Software de configuración

Como el módulo bluetooth es básicamente un nodo bluetooth conectado a una interface serie, podríamos en principio conectar los pines RX y Tx a los equivalentes de Arduino en los pines 0 y 1 digitales. Sin embargo, se ha preferido evitar este modo porque los pines 0 y 1 se utilizan en la comunicación serie de Arduino con el PC a través del USB y por tanto, si los usamos para comunicar con el módulo, perderíamos la conexión con el PC.

Por ello se prefiere destinar otro par de pines cualesquiera a la transmisión, aunque para ello tenemos que importar una librería que habilite la comunicación serie con otros pines como es la librería SoftwareSerial [31].

Para ello importamos la librería que viene de serie en el IDE y creamos un nuevo objeto serie llamado BTserial conectado a los pines 2 y 3:

```
#include <SoftwareSerial.h>
SoftwareSerial BTserial(2, 3); // RX | TX
// Connect the HC-06 TX to the Arduino RX on pin 2.
// Connect the HC-06 RX to the Arduino TX on pin 3 through a voltage divider.

void setup()
{
  Serial.begin(9600);
  Serial.println("Enter AT commands:");

  // HC-06 default serial speed is 9600
  BTserial.begin(9600);
}

void loop()
{
  // Keep reading from HC-06 and send to Arduino Serial Monitor
  if (BTserial.available())
  {
    Serial.write(BTserial.read());
  }
  // Keep reading from Arduino Serial Monitor and send to HC-06
  if (Serial.available())
  {
    BTserial.write(Serial.read());
  }
}
```

IMPORTANTE: Utilizando los comandos AT se han fijado las siguientes características para el módulo bluetooth:

Nombre: TLHEX

Pin: 6789

6.2.6 Pruebas de Conexión

6.2.6.1 Dispositivo Android

Como el módulo HC-06 actúa únicamente como esclavo, la conexión debe ser iniciada por otro dispositivo (en este caso un teléfono móvil). A continuación en este apartado se detallan los pasos a seguir para conectarnos con un dispositivo Android.

Se hará uso de una App de software libre llamada “Bluetooth Terminal” disponible en Google Play.

Antes de poder conectar el HC-06 necesitamos parearlo. Por tanto, seguiremos los siguientes pasos:

1. Alimentar el módulo hc-06. El pin led parpadeará rápidamente.
2. Abrir Ajustes en el dispositivo Android y seleccionar Bluetooth.
3. Seleccionar el módulo en la lista, en nuestro caso llamado "TLHEX".
4. Introducir el pin, recordemos que es "6789"

Después de varias pruebas, se concluyó que con la versión 6 de Android no es posible encontrar el módulo bluetooth.

6.2.6.2 PC o laptop

Debido a que no podemos realizar la prueba de bluetooth con un dispositivo Android versión 6.0, utilizaremos un programa hiperterminal llamado "PuTTy" desde nuestro PC para realizar la verificación de que el módulo funciona correctamente [32, 33]. La trayectoria de la información puede verse en la Figura 6.3:

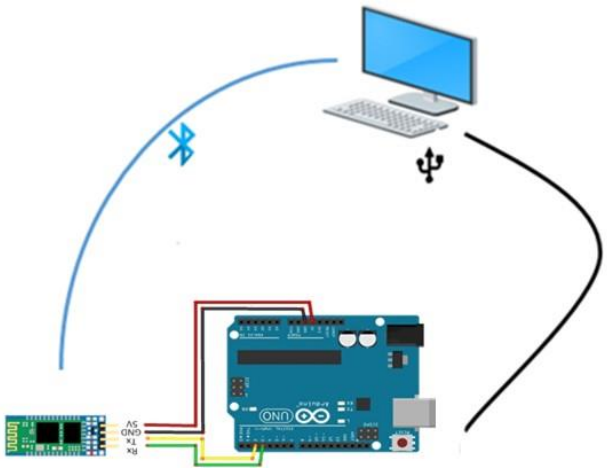


Figura 6.3 Utilización módulo HC-06 junto a PuTTY

Siguiendo la figura anterior podemos deducir el recorrido de la información:

- Escribiendo en el IDE de Arduino la información irá por comunicación serie mediante USB a la placa Arduino y desde ahí será enviada por medio de bluetooth a la ventana de PuTTy.
- Escribiendo en la ventana de PuTTy la información irá por comunicación bluetooth a la placa Arduino y desde ahí será enviada por comunicación serie USB al IDE de Arduino.

Los pasos a seguir con Windows son los siguientes:

1. Conectamos nuestro Arduino junto con el módulo bluetooth y abrimos el monitor serie en el IDE de Arduino. El LED del bluetooth parpadeará.
2. En el icono de "bluetooth" de nuestro escritorio, pinchamos botón derecho y "mostrar dispositivos"
3. Seleccionamos "agregar dispositivo"
4. De la lista que aparece elegiremos el módulo al que le hemos llamado "TLHEX"
5. A continuación, nos pedirá el pin: 6789
6. Una vez realizados los pasos anteriores, volvemos al icono de "bluetooth" de nuestro escritorio y seleccionamos "abrir configuración".
7. En nuestro caso, en la pestaña "Puertos COM" nos aparece la siguiente Figura 6.4:

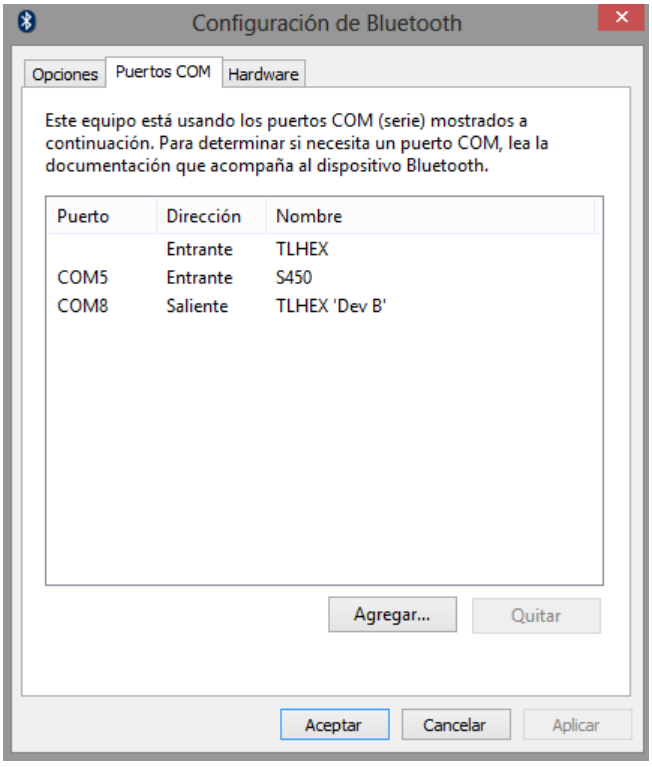


Figura 6.4 Configuración bluetooth para Windows

Como podemos ver, la dirección saliente del dispositivo TLHEX es el puerto COM8.

8. Abrimos el programa PuTTy, en la ventana que nos aparece debemos completar los campos siguiendo la Figura 6.5:

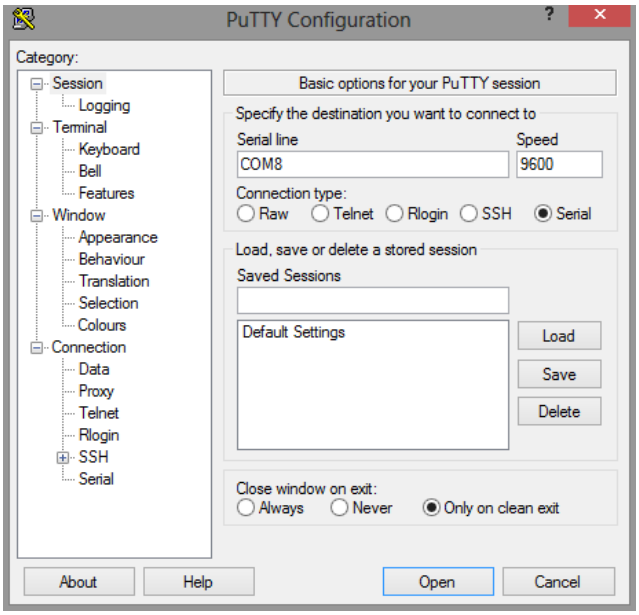


Figura 6.5 Configuración de PuTTY

Nota: Para nuestro caso, hemos comprobado que se trataba del puerto COM8. Éste podría variar.

9. Una vez hayamos pulsado "Open", el LED dejará de parpadear indicando que se ha establecido la conexión correctamente. Tendremos las pantallas de la Figura 6.6:

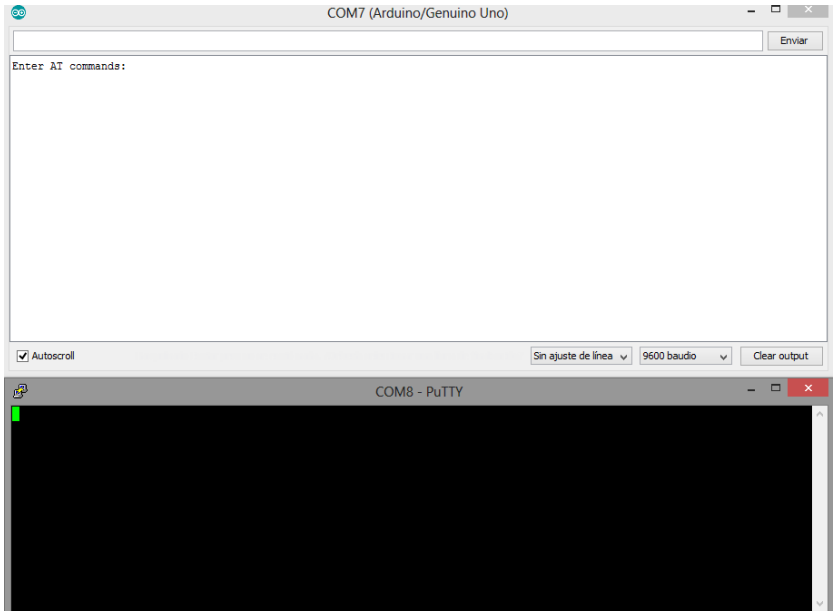


Figura 6.6 Pantallas para envío de información

Ahora para comprobar, escribiremos en el IDE de Arduino y veremos que aparece en la ventana PuTTy (Figura 6.7).

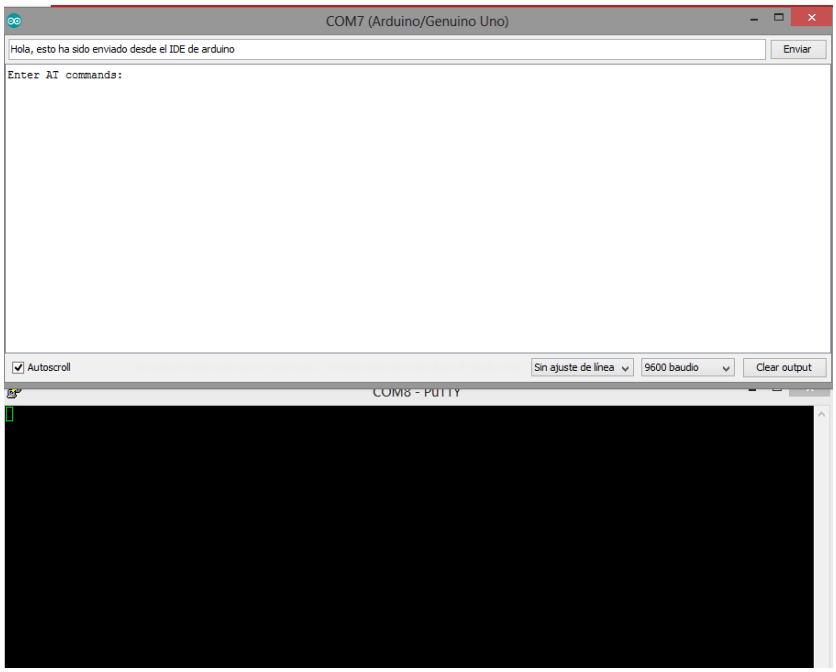


Figura 6.7 Envío de información desde IDE Arduino a PuTTY

Al darle a intro, lo vemos en la ventana de PuTTy (Figura 6.8):

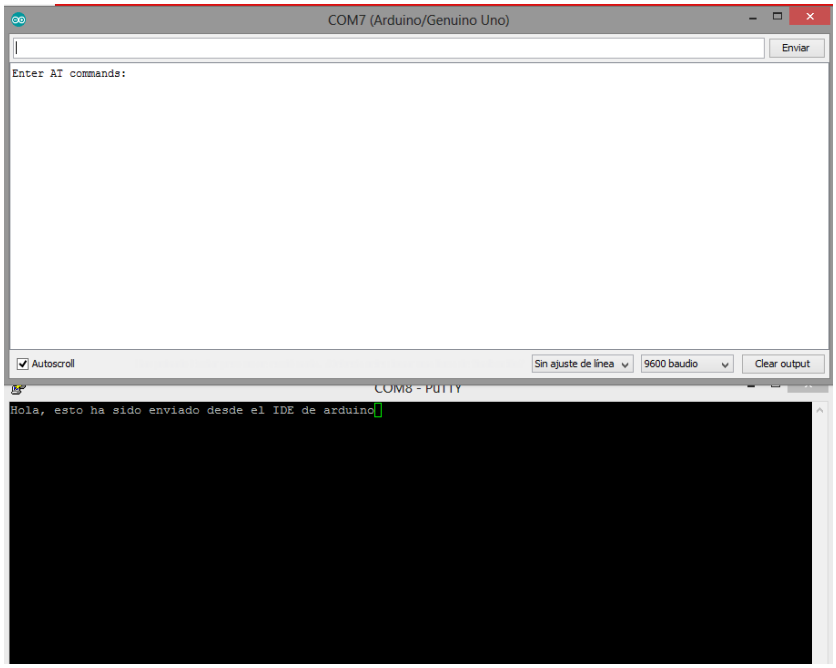


Figura 6.8 Visualización en PuTTY de la información

De forma inversa, podemos escribir en la ventana PuTTy y aparecerá en el IDE de Arduino (Figura 6.9):

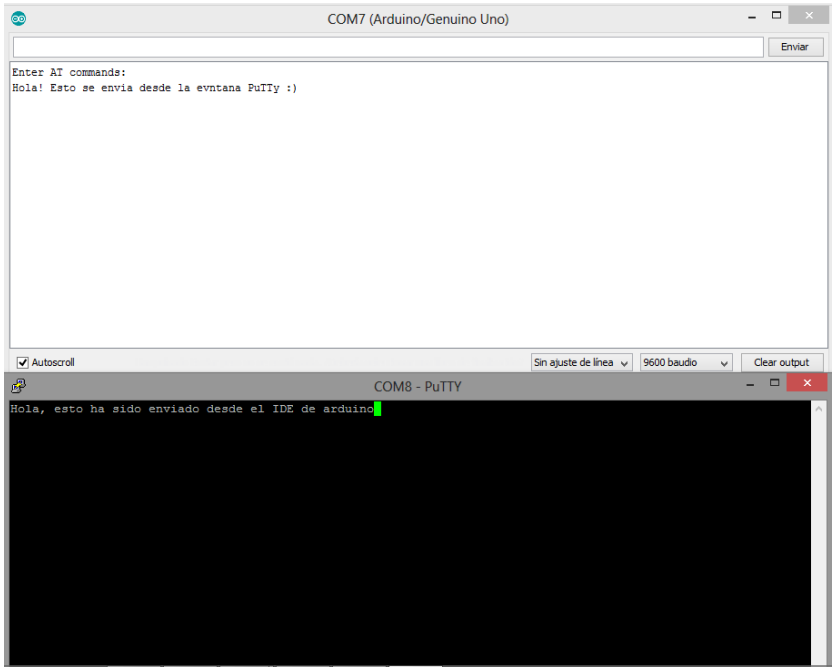


Figura 6.9 Envío de información desde PuTTY a IDE Arduino

De esta forma queda comprobado que el módulo bluetooth HC-06 funciona de forma adecuada, no pudiendo realizarse la conexión con el dispositivo móvil por problemas de incompatibilidad.

6.2.6.2.1 Potenciómetro de hilo

En este apartado nos centraremos en comprobar el envío de lecturas realizadas por el potenciómetro de hilo a la ventana del hipperterminal PuTTy, realizaremos la conexión de la Figura 6.10:

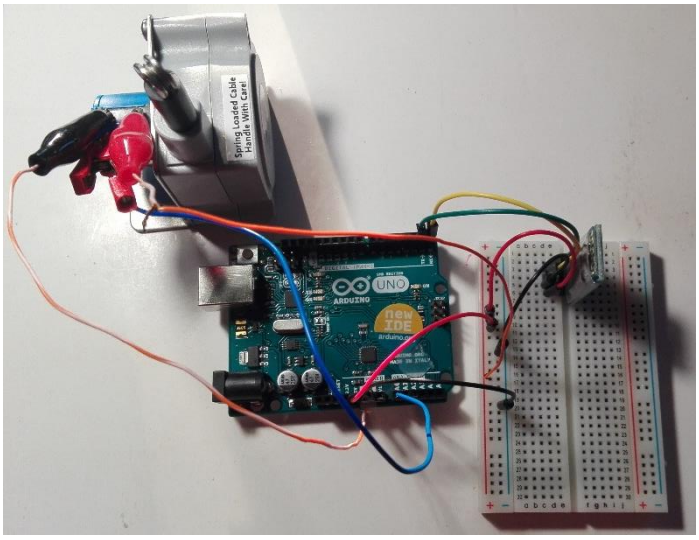


Figura 6.10 Conexionado potenciómetro de hilo junto a módulo HC-06

El programa Arduino necesario para este caso se explica a continuación. Es importante destacar que al cargar el programa es preferiblemente que no se haya realizado aun la conexión sobre la placa ya que puede dar errores.

El siguiente código consiste en que si pulsamos la tecla "1" nos devuelve el valor del promedio de las 1200 medidas tomadas (es una ampliación del código visto en el apartado del "Software Arduino" del potenciómetro de hilo) y luego nos muestra ese valor en la ventana PuTTy. Mientras que si pulsamos la tecla "0" nos muestra la frase "ha escogido no realizar lectura". Este código es una primera aproximación al definitivo, más adelante necesitará una seria de pequeñas modificaciones.

```
float suma = 0;
long start;
int contador = 0;
float promedio = 0;
float voltaje = 0;
int mm = 0;
int c;

void setup() {
  Serial.begin(9600);
}

void loop() {
```

```
 if(Serial.available()){
 c=Serial.read();

 switch(c){

   case '1':
   Serial.println("la lectura en mm es: ");
 do {
   //Suma todas las lecturas de A1
   suma += analogRead(1);
   contador++;
 } while (contador < 1200);
 //Cálculo del promedio
 promedio = (float) (suma / contador);
 //Pasa el valor del promedio a voltios
 voltaje = promedio * (5.0 / 1023.0);
 //Valor en mm
 mm = voltaje * (145.3) + 2.1919;
 Serial.println(mm);
 delay(1000);
 //Preparo las variables para el siguiente promediado
 suma = 0;
 contador = 0;
    break;

    case '0':
    Serial.println("Ha escogido no realizar lectura");
    break;
 }
 }
}
```

Una vez cargado el programa podemos abrir la ventana PuTTy y observar el valor de las lecturas (Figura 6.11):

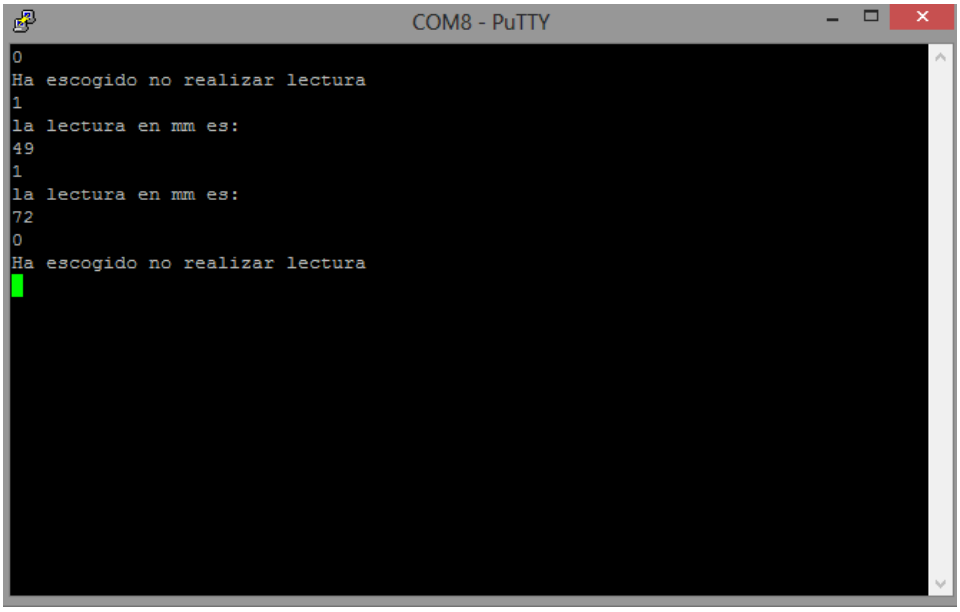


Figura 6.11 Prueba envío de medición utilizando el potenciómetro hilo

6.3 Entorno de Desarrollo

Los sistemas de visualización de datos tienen su origen en los SCADA (Supervisoy Control and Data Acquisition), se empezaron a usar en la industria en los años 70 para poder monitorizar y controlar los diferentes procesos productivos de la industria cómodamente desde una sala de control [34].

Para poder visualizar datos desde Arduino existen tres diferentes alternativas, podemos utilizar: Processing, Matlab o Labview. Hemos elegido Processing porque es sencillo encontrar ejemplos y tutoriales en la red, además es un software libre y de código abierto.

Es muy común usar Processing con Arduino para crear un entorno de visualización de datos mucho más atractivo visualmente y con más posibilidades que si usamos el Monitor Serial del IDE de Arduino.

Cabe destacar que Matlab es una alternativa con mucho potencial, ya que nos permite conectarnos con Arduino tanto por Firmata como a través del puerto serie. Como gran inconveniente, tiene la imposibilidad de seguir varios hilos de ejecución (al igual que Processing) [35].

Por último, siguiendo con Labview, destacar su particular lenguaje de programación, que debe dominarse para poder desarrollar una aplicación con él y que la única forma de comunicarse con

Arduino es mediante Firmata. Como ventaja, éste si permite la ejecución en paralelo de varios bucles.

Han sido utilizados dos entornos de programación distintos para el desarrollo del software necesario para realizar la lectura de los sensores instalados en los actuadores del TL-Hex. Por un lado, se ha utilizado el IDE de Arduino que es indispensable para cargar programas en el microcontrolador; por otro lado, se ha utilizado Processing.

El IDE de Arduino está basado en Processing (Figura 6.12), solo cambia el botón PLAY que sirve para arrancar el entorno creado con el programa, y el botón STOP que sirve para pararlo [36].

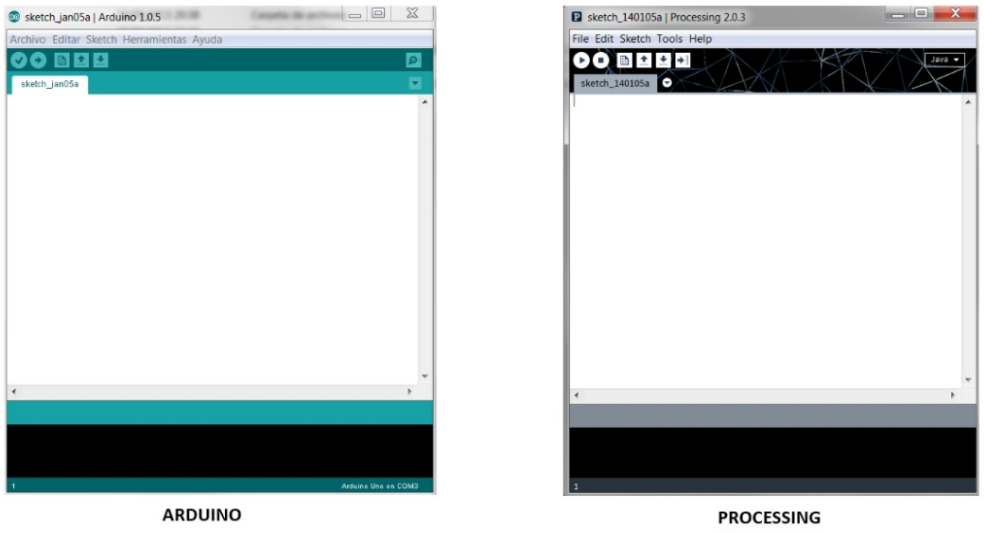


Figura 6.12 Comparación Arduino y Processing

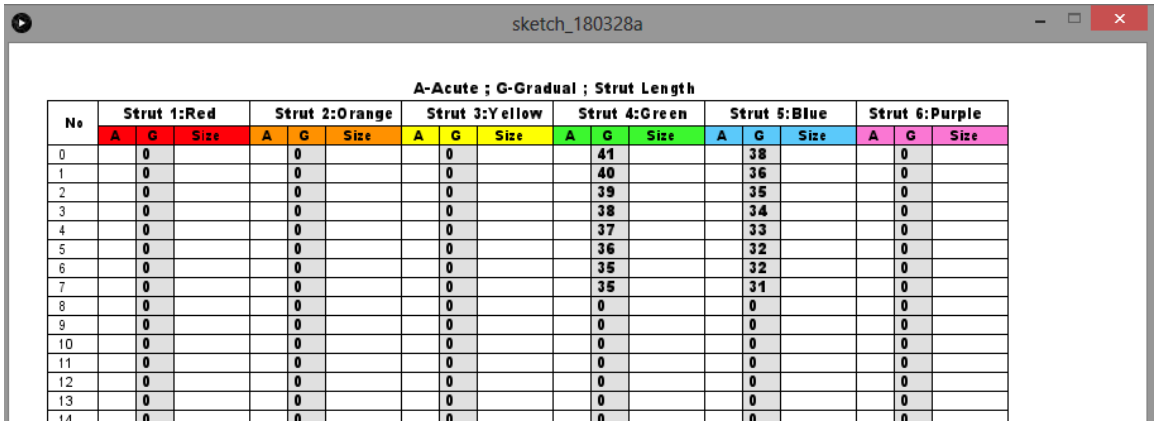
Existen dos modos de programar Arduino junto a Processing. El primero, es utilizando la librería para Processing de Arduino y Firmata; el segundo, es utilizando la comunicación por puerto serie. Nosotros hemos utilizado el segundo modo, consiguiendo una comunicación estable y sin interferencias en el puerto serie, esto nos permite aprovechar el potencial del microcontrolador para realizar los cálculos necesarios [37, 38].

6.4 Software Desarrollado

El software desarrollado completo consta de dos partes claramente diferenciadas.

La primera parte es la encargada de la lectura de los sensores, de esta parte se encarga Arduino. Una vez que Arduino realiza los cálculos necesarios y tiene el valor del ajuste gradual, lo almacena en una variable llamada "G". Esta variable la calcula, la escribe en el puerto serie y la sobrescribe cada vez que se ejecuta el "void loop".

La segunda parte es la encargada de la presentación gráfica del ajuste gradual, desarrollada con Processing. Como podemos ver en la Figura 6.13, nos hemos basado en el "informe" para el médico del Software TL-Hex. Para visualizar el valor del ajuste gradual, lo que debemos hacer es pulsar con el ratón sobre la casilla de la columna "G" correspondiente del actuador que deseamos saber la medida. Esta interfaz lo que hace es leer lo que hay en el puerto serie y almacenarlo en una variable que luego pinta en la celda seleccionada.



No	Strut 1:Red			Strut 2:Orange			Strut 3:Yellow			Strut 4:Green			Strut 5:Blue			Strut 6:Purple		
	A	G	Size	A	G	Size	A	G	Size	A	G	Size	A	G	Size	A	G	Size
0		0			0			0			41			38			0	
1		0			0			0			40			36			0	
2		0			0			0			39			35			0	
3		0			0			0			38			34			0	
4		0			0			0			37			33			0	
5		0			0			0			36			32			0	
6		0			0			0			35			32			0	
7		0			0			0			35			31			0	
8		0			0			0			0			0			0	
9		0			0			0			0			0			0	
10		0			0			0			0			0			0	
11		0			0			0			0			0			0	
12		0			0			0			0			0			0	
13		0			0			0			0			0			0	

Figura 6.13 Software para visualización posición gradual

Para simplificar la comprensión de ambas partes, en los dos siguientes apartados se encuentran los diagramas de flujo y una explicación de los mismos. En los ANEXOS indicados podremos encontrar ambos códigos completamente desarrollados.

6.4.1 Arduino

El código del siguiente diagrama de flujo se encuentra en el **ANEXO II.**

La estructura de un programa Arduino consta de dos partes bien diferenciadas (Figura 6.14): Void Setup, donde se ejecutan un conjunto de sentencias solo una vez; Void Loop, se ejecutan en bucle un conjunto de sentencias. En la primera se encuentran las instrucciones de inicialización, mientras que en la segunda el programa en concreto que queremos ejecutar.

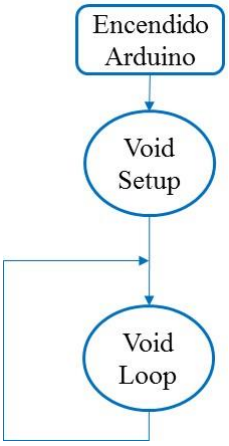


Figura 6.14 Flujograma de estructura programa Arduino

El conjunto de sentencias que se encuentra en cada bloque se pueden ver en los flujogramas de la Figura 6.15:

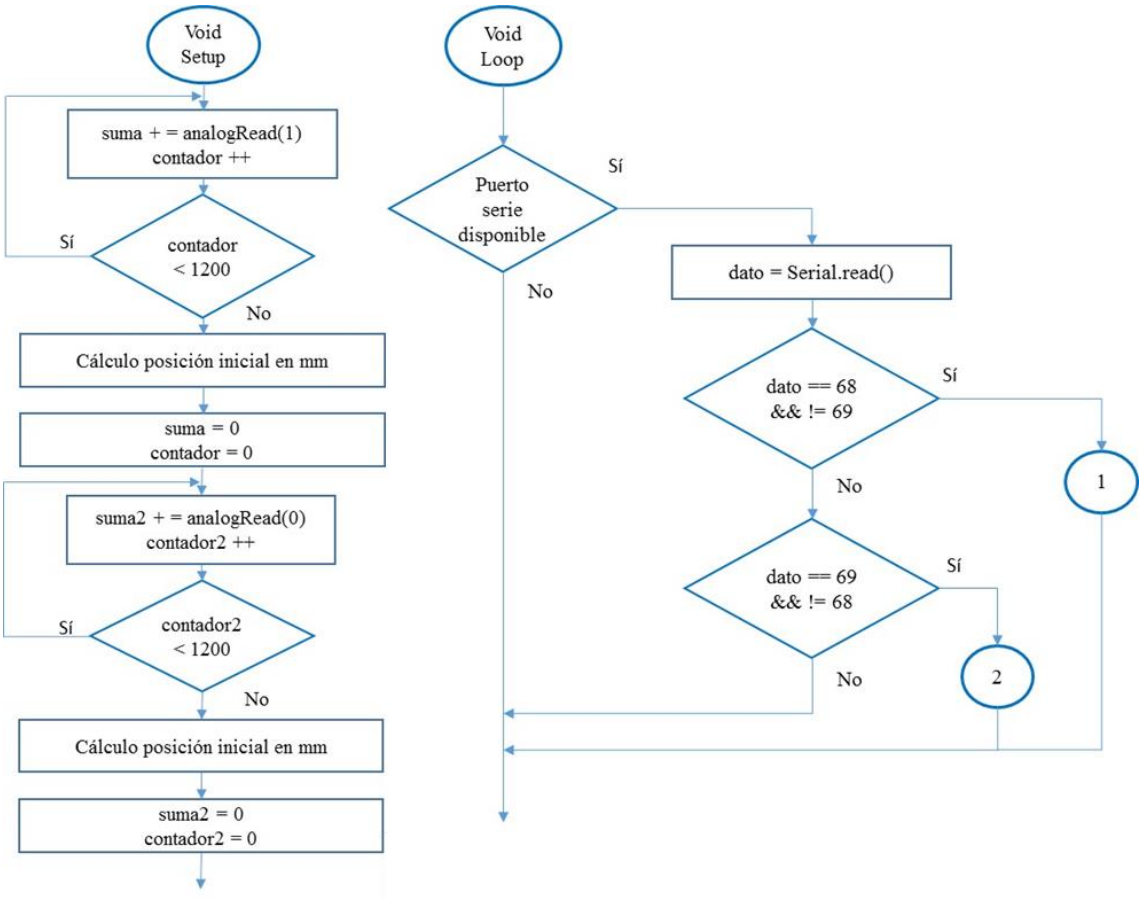


Figura 6.15 Flujogramas programa Arduino parte 1

En el Void Setup, ajustaremos la velocidad de comunicación con el puerto serie a 9600 baudios y además realizaremos la primera lectura de la posición inicial. Siguiendo el diagrama de flujo, se ejecutan las siguientes sentencias:

Bucle "do-while": en este realizamos la lectura del pin analógico A1 donde se encuentra conectado el potenciómetro deslizante. Realizamos la lectura y la acumulamos en una variable "suma" hasta que realicemos un total de 1200 lecturas.

Cálculo posición inicial: este está formado por el siguiente bloque de sentencias, las cuales han sido explicadas en el apartado 5.3.1 Potenciómetro lineal.

```
//Cálculo del promedio
promedio = (float) (suma / contador);
//Pasa el valor del promedio a voltios
voltaje = promedio * (5.0 / 1023.0);
//Valor en mm
inicio = voltaje * (-19.11) + 97.508;
```

Por último, se reinician las variables "suma" y "contador" para poder utilizarlas en el siguiente promediado.

A continuación se realiza lo mismo pero para el potenciómetro deslizante conectado en el pin A0.

En el Void Loop, realizaremos el cálculo de la posición gradual y la escribiremos en el puerto serie. Para ello, se ejecutan las siguientes sentencias:

"if - else": Esta sentencia se ejecuta para comprobar si hay algún dato disponible en el puerto serie. Es decir, estamos esperando recibir desde la interfaz desarrollada en "Processing" una "D" o una "E" en código ASCII, 68 o 69 respectivamente.

Si hay un dato disponible lo almacena en una variable "dato"

"if – else": Si dato es 68 y distinto de 69, se ejecutan el conjunto de sentencias del conector "1". Mientras que si no se cumple esta condición, entramos en otro "if – else" que comprueba que dato sea 69 y distinto de 68, para ejecutar el conjunto de sentencias del conector "2".

En los conectores "1" y "2" se vuelve a realizar la lectura de la posición actual en mm. En este caso, entramos en un bucle "do – while", en cual ejecutaremos las sentencias en su interior mientras que no recibamos el dato de parada. El dato de parada será volver a recibir una "D" desde la interfaz para el caso del potenciómetro deslizante, o una "E" para el potenciómetro de hilo (Figura 6.16).

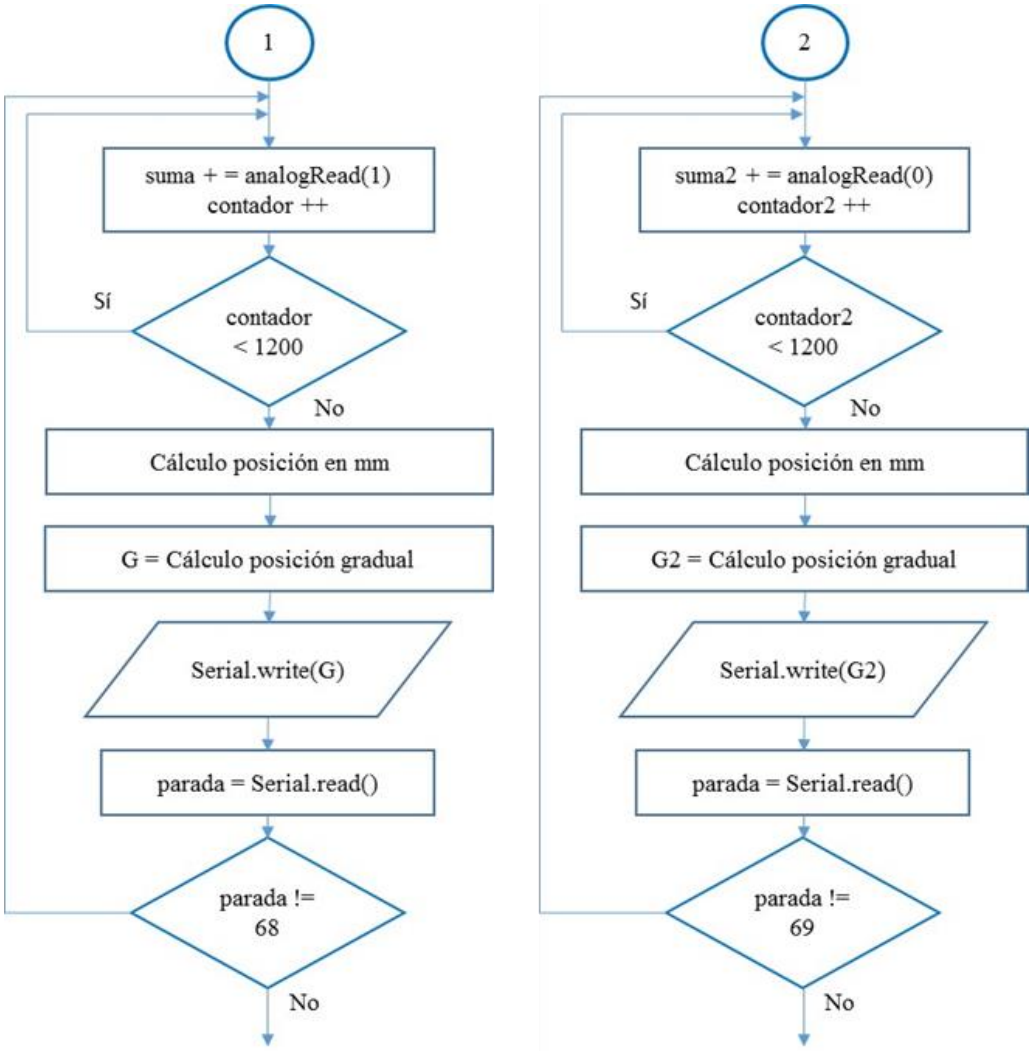


Figura 6.16 Flujogramas programa Arduino parte 2

En este caso, cuando realizamos la lectura de la posición actual en mm se almacena en una variable llamada "mm". Para el cálculo de la posición gradual, se ejecutan las siguientes ecuaciones:

```
//Diferencia posición actual e inicial
    medida = mm - inicio;
// valor de la posición gradual en mm
    G = -(abs(medida)) + 80;
```

Posteriormente, se escribe en formato de un byte en el puerto serie.

6.4.2 Processing

El código del siguiente diagrama de flujo se encuentra en el **ANEXO III.**

En este caso, se adjunta el diagrama de flujo de una parte del código completo. Esto se debe a que este proceso se repite de forma análoga hasta un total de 155 veces.

La estructura de un programa Processing es muy similar a uno Arduino (Figura 6.17), consta de dos partes bien diferenciadas: Void Setup y Void Draw, lo que equivale al Void Loop de Arduino. En la primera se encuentran las instrucciones de inicialización, mientras que en la segunda el programa en concreto que queremos ejecutar.

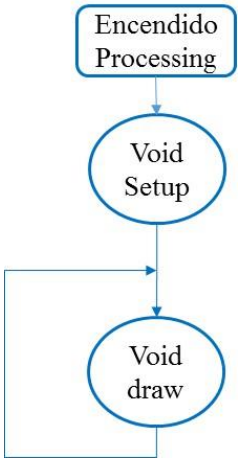


Figura 6.17 Flujograma de estructura programa Processing

El conjunto de sentencias que se encuentra en cada bloque se pueden ver en los flujogramas a continuación:

En el Void Setup (Figura 6.18), ajustaremos la velocidad de comunicación con el puerto serie a 9600 baudios, tiene que ser la misma que la ajustada en Arduino para que pueda existir comunicación. Además ejecutaremos la instrucción "InitLayout()" la cual contiene el conjunto de sentencias necesarias para definir completamente las columnas de introducción de texto, en este caso son las situadas debajo de "A" de Acute y "Size" de tamaño del actuador.

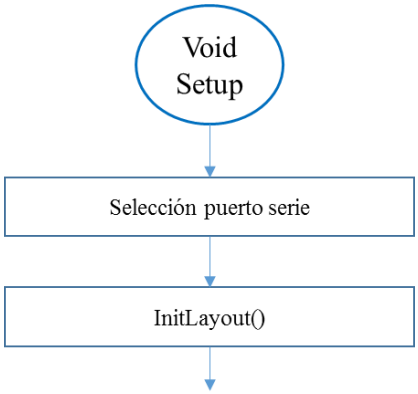


Figura 6.18 Flujogramas programa Processing parte 1

En el Void Draw (Figura 6.19), por el puerto serie enviaremos un carácter a Arduino para que sepa qué sensor queremos leer, posteriormente, realizaremos la lectura del puerto serie para recibir la lectura del sensor y por último, pintaremos por pantalla ese valor. Para ello, se ejecutan las siguientes sentencias:

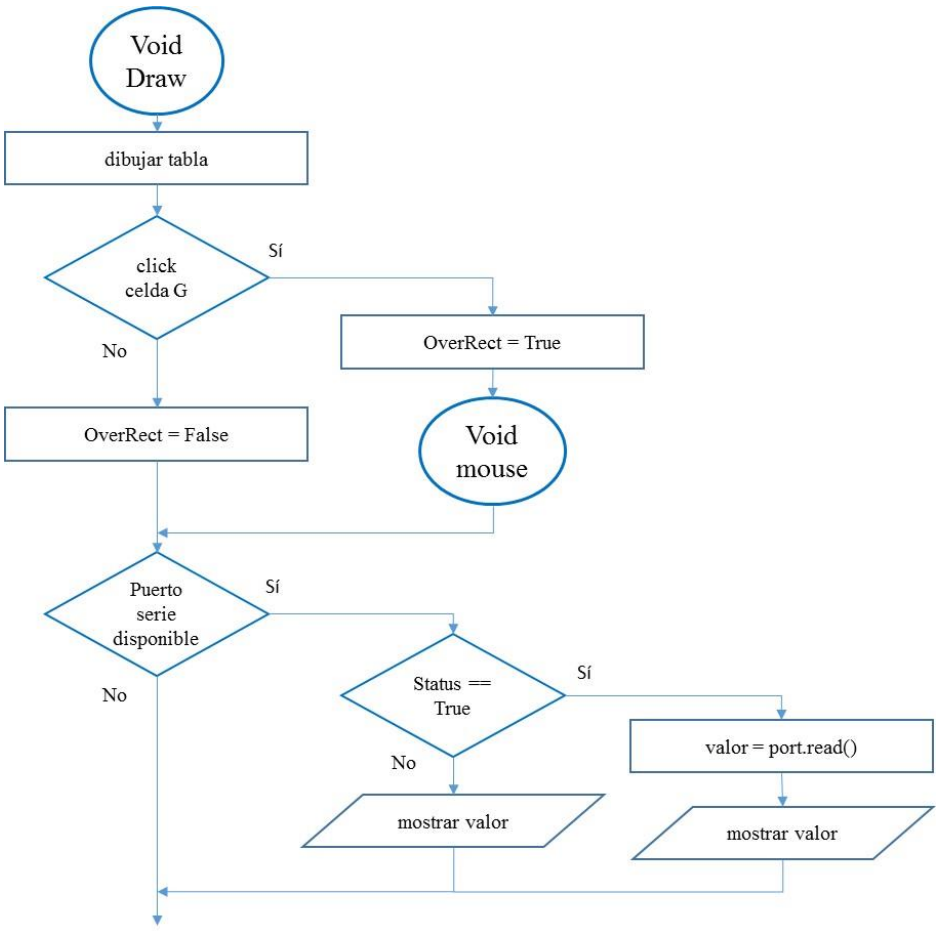


Figura 6.19 Flujogramas programa Processing parte 2

Lo primero, será dibujar el diseño completo de la interfaz, esto está formado por: las columnas correspondientes a los días del tratamiento y las columnas de cada actuador divididas en “A” de Acute, “G” de Gradual y “Size” del tamaño del actuador.

“if - else”: Este sirve para comprobar si hemos pulsado con el ratón sobre la celda “G” de la que queremos saber la medida. En el caso de que sea no, se define la variable OverRect como falsa, de lo contrario se definirá como verdadera y entraremos en el conector “Void mouse” forma abreviada de “Void mousePressed()”.

Posteriormente, entraremos en otro “if - else” en cual se comprueba si el puerto serie está disponible. De ser cierto esto, nos encontramos con otro “if - else” en cual verificamos si la variable "status" es cierta, de cumplirse esto, realizamos la lectura del puerto serie y lo almacenamos en una variable llamada “valor”. Por último, pintaremos pantalla la variable “valor” en la celda donde habíamos pulsado con el ratón. De ser falsa la variable “status”, mostramos “valor” por pantalla.

Una vez estamos dentro del conector “Void MousePressed” (Figura 6.20) se realizará el siguiente conjunto de sentencias:

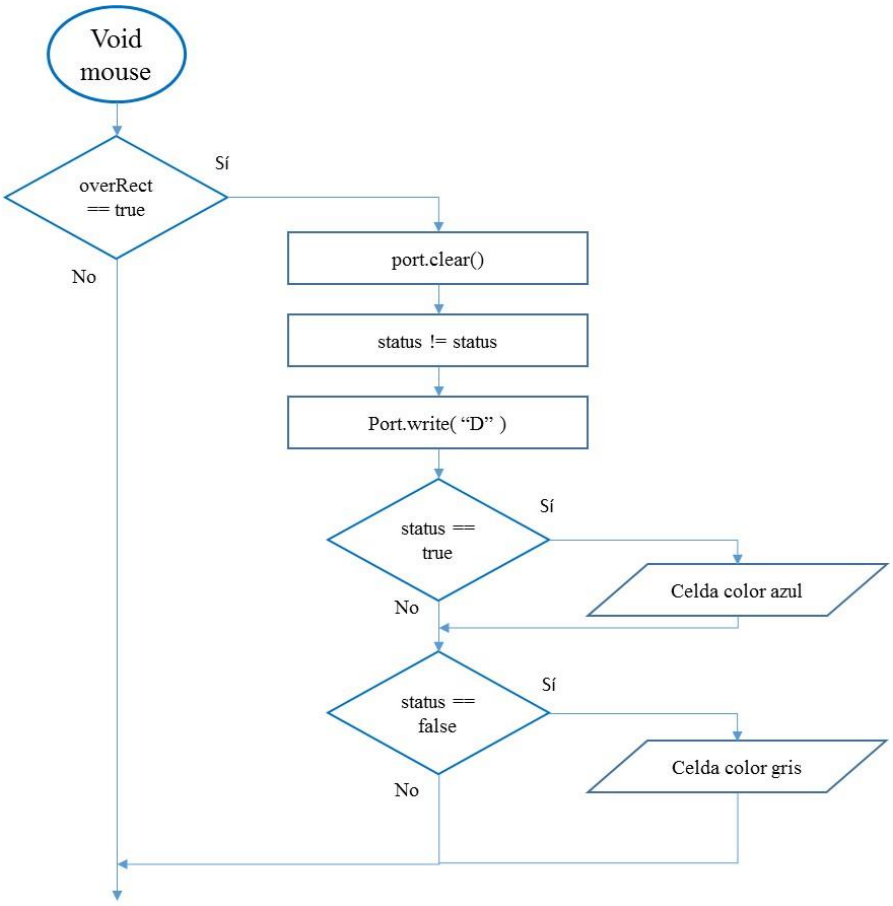


Figura 6.20 Flujogramas programa Processing parte 3

Lo primero, será comprobar si la variable "OverRect" es cierta mediante un "if - else", de ser así se borrará todo lo que hay en el puerto serie con la función "port.clear()", luego se cambiará el estado de la variable "status" con "status != status" y por último, se enviará una letra por el puerto serie con la función "Port.write("D")". La letra enviada seguirá el siguiente criterio:

- Letra "A" si queremos saber la medida "G" del actuador 1
- Letra "B" si queremos saber la medida "G" del actuador 2
- Letra "C" si queremos saber la medida "G" del actuador 3
- Letra "D" si queremos saber la medida "G" del actuador 4
- Letra "E" si queremos saber la medida "G" del actuador 5
- Letra "F" si queremos saber la medida "G" del actuador 6

Para terminar, tenemos dos "if - else" con el fin de comprobar si la variable "status" es cierta o falsa. En el caso de ser cierta, el color de la celda seleccionada se pondrá de color azul; mientras que si es falsa, el color de la celda permanecerá gris.

6.5 Descripción del HMI

A continuación se detallan los pasos a seguir para realizar una correcta interacción con la interfaz y el sistema TL-Hex:

1. Presionar el botón de encendido situado en la parte posterior de la caja negra del TL-Hex y esperar unos segundos.
2. Abrir el sketch de Processing donde se encuentra el código de programación de la interfaz y darle al "play".
3. Se nos abrirá una ventana con la interfaz lista para comenzar a usar.
4. Introducimos manualmente los valores de Acute y Size en el día correspondiente al actuador que deseamos realizar la medición.
5. Hacemos "Click" con el ratón sobre la celda "G" que deseamos medir, esta se pondrá de color azul indicando que esta lista para recibir información.
6. Siguiendo la prescripción proporcionada por el médico, realizaremos el ajuste manual del actuador según el número de "clicks" indicados
7. A la vez que realicemos este ajuste, podremos ver por pantalla la variación a tiempo real que está sufriendo el ajuste gradual.
8. Una vez que hemos realizado el ajuste y vemos el valor esperado en la interfaz, volvemos a hacer "click" con el ratón sobre la celda "G" ahora esta se volverá de color gris, indicando que el sensor ya no hará más lecturas sobre ese actuador.

9. Repetir todos los pasos anteriores hasta completar el tratamiento.
10. Una vez completado el tratamiento, presionar el botón de apagado situado en la parte posterior de la caja negra del TL-Hex.

6.6 Montaje e Integración en el TL-Hex

Finalmente, podemos ver la conexión con todos los elementos en la Figura 6.21:

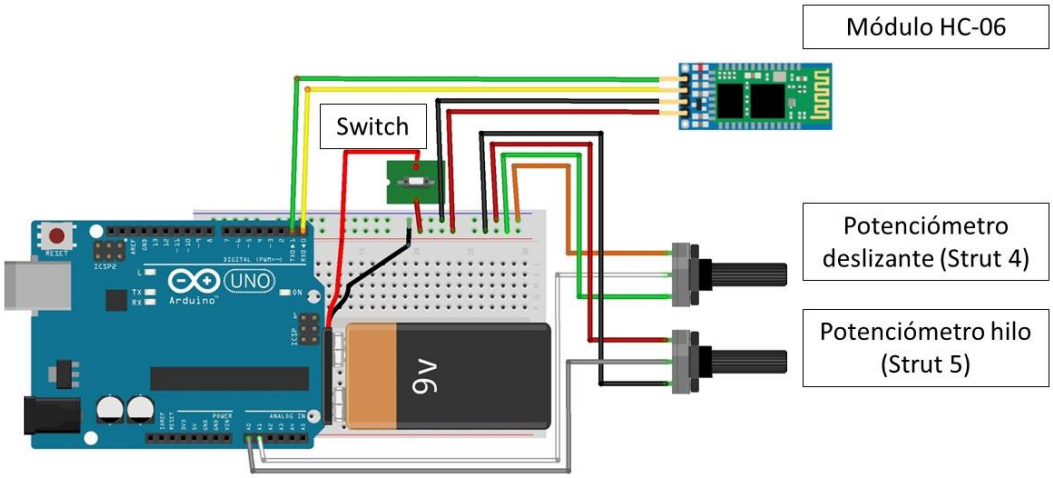


Figura 6.21 Conexionado final

Mientras que el conexionado final (integrado en una caja negra sobre el TL-Hex) podemos verlo en la Figura 6.22 a continuación:

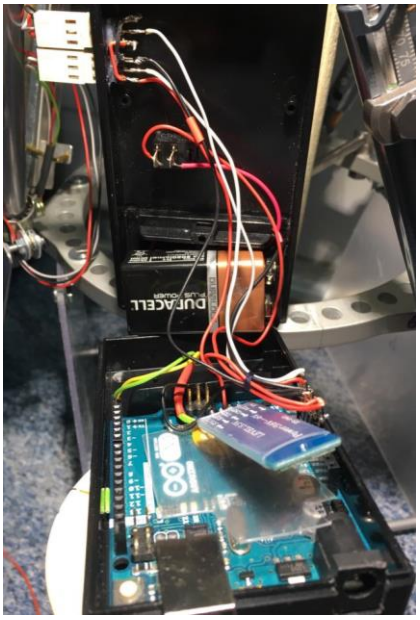


Figura 6.22 Conexionado final real

Capítulo 7

Prueba Experimental y Resultados

7.1 Introducción

En este capítulo se muestra una aplicación más cercana a la realidad del hardware y la interfaz desarrollados. El Dr. Salcedo realizó la instalación de una maqueta de un fémur izquierdo y nos proporcionó la "prescripción" que se les entrega a los pacientes, para el tratamiento consistente en una elongación de fémur.

El fin de este capítulo es verificar que las medidas de los sensores tengan concordancia con la de la prescripción y lo que marca el actuador. Por otro lado, también se hará un análisis estadístico de ANOVA (*Analysis Of Variance*) para validar si existen diferencias significativas.

7.2 Prueba Experimental

Como primera prueba del sistema, con el asesoramiento del Dr. Salcedo, se ha optado por comenzar con un caso sencillo de "elongación de fémur izquierdo". En la Figura 7.1 se puede ver el hueso antes y en la Figura 7.2 después del tratamiento:

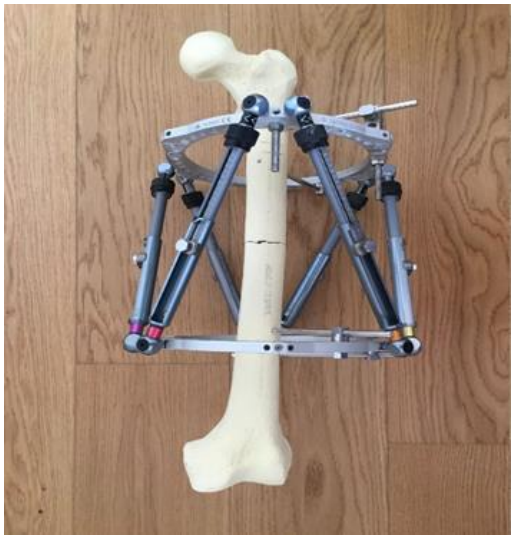


Figura 7.1 Fémur izquierdo antes de realizar la elongación

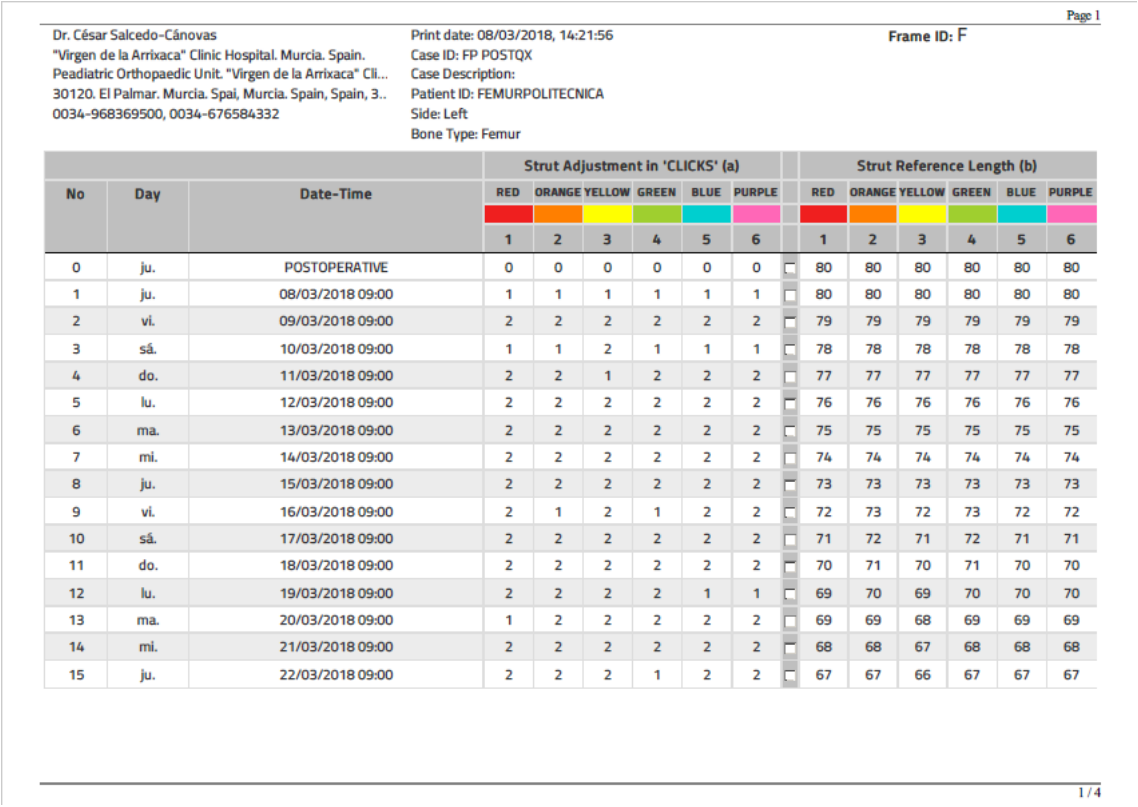
Figura 7.2 Fémur izquierdo después de realizar la elongación

Este tipo de operaciones es común entre niños y adolescentes, aunque también puede darse en adultos. Las ventajas que nos permite el TL-Hex respecto a otros sistemas son:

- Alargamiento óseo.
- Corrección simultánea de la deformación (no es posible con el monolateral).
- Control de la compresión.
- Corrección mediante software.

Los pacientes a los que se les somete a este tipo de sistema son los que sufren una dismetría e hipometrías. Además se puede aplicar en pacientes que simultáneamente tengan una deformación. El mejor momento para realizar este tipo de intervenciones es durante la adolescencia, ya que la regeneración ósea es buena y existe una cierta implicación por parte del paciente [39].

El alargamiento que se va a llevar acabo es de 4,5 cm durante 51 días de tratamiento. A continuación, en las Figuras 7.3, 7.4, 7.5 y 7.6 se adjunta la prescripción proporcionada por el Dr. Salcedo.



Page 1

Dr. César Salcedo-Cánovas
"Virgen de la Arrixaca" Clinic Hospital. Murcia. Spain.
Peadiatric Orthopaedic Unit. "Virgen de la Arrixaca" Cli...
30120. El Palmar. Murcia. Spai, Murcia. Spain, Spain, 3..
0034-968369500, 0034-676584332

Print date: 08/03/2018, 14:21:56
Case ID: FP POSTQX
Case Description:
Patient ID: FEMURPOLITECNICA
Side: Left
Bone Type: Femur

Frame ID: F

			Strut Adjustment in 'CLICKS' (a)							Strut Reference Length (b)					
No	Day	Date-Time	RED	ORANGE	YELLOW	GREEN	BLUE	PURPLE		RED	ORANGE	YELLOW	GREEN	BLUE	PURPLE
			1	2	3	4	5	6		1	2	3	4	5	6
0	ju.	POSTOPERATIVE	0	0	0	0	0	0	☐	80	80	80	80	80	80
1	ju.	08/03/2018 09:00	1	1	1	1	1	1	☐	80	80	80	80	80	80
2	vi.	09/03/2018 09:00	2	2	2	2	2	2	☐	79	79	79	79	79	79
3	sá.	10/03/2018 09:00	1	1	2	1	1	1	☐	78	78	78	78	78	78
4	do.	11/03/2018 09:00	2	2	1	2	2	2	☐	77	77	77	77	77	77
5	lu.	12/03/2018 09:00	2	2	2	2	2	2	☐	76	76	76	76	76	76
6	ma.	13/03/2018 09:00	2	2	2	2	2	2	☐	75	75	75	75	75	75
7	mi.	14/03/2018 09:00	2	2	2	2	2	2	☐	74	74	74	74	74	74
8	ju.	15/03/2018 09:00	2	2	2	2	2	2	☐	73	73	73	73	73	73
9	vi.	16/03/2018 09:00	2	1	2	1	2	2	☐	72	73	72	73	72	72
10	sá.	17/03/2018 09:00	2	2	2	2	2	2	☐	71	72	71	72	71	71
11	do.	18/03/2018 09:00	2	2	2	2	2	2	☐	70	71	70	71	70	70
12	lu.	19/03/2018 09:00	2	2	2	2	1	1	☐	69	70	69	70	70	70
13	ma.	20/03/2018 09:00	1	2	2	2	2	2	☐	69	69	68	69	69	69
14	mi.	21/03/2018 09:00	2	2	2	2	2	2	☐	68	68	67	68	68	68
15	ju.	22/03/2018 09:00	2	2	2	1	2	2	☐	67	67	66	67	67	67

1 / 4

Figura 7.3 Prescripción página 1

Dr. César Salcedo-Cánovas
"Virgen de la Arrixaca" Clinic Hospital. Murcia. Spain.
Peadiatric Orthopaedic Unit. "Virgen de la Arrixaca" Cli...
30120. El Palmar. Murcia. Spai, Murcia. Spain, Spain, 3..
0034-968369500, 0034-676584332

Print date: 08/03/2018, 14:21:56
Case ID: FP POSTQX
Case Description:
Patient ID: FEMURPOLITECNICA
Side: Left
Bone Type: Femur

Frame ID: F

No	Day	Date-Time	Strut Adjustment in 'CLICKS' (a)						Strut Reference Length (b)					
			RED	ORANGE	YELLOW	GREEN	BLUE	PURPLE	RED	ORANGE	YELLOW	GREEN	BLUE	PURPLE
			1	2	3	4	5	6	1	2	3	4	5	6
16	vi.	23/03/2018 09:00	2	1	2	2	2	2	66	66	65	66	66	66
17	sá.	24/03/2018 09:00	2	2	2	2	2	2	65	65	64	65	65	65
18	do.	25/03/2018 09:00	2	2	1	2	2	2	64	64	64	64	64	64
19	lu.	26/03/2018 09:00	2	2	2	2	2	2	63	63	63	63	63	63
20	ma.	27/03/2018 09:00	2	2	2	2	2	2	62	62	62	62	62	62
21	mi.	28/03/2018 09:00	2	2	2	2	2	1	61	61	61	61	61	61
22	ju.	29/03/2018 09:00	2	2	2	1	2	2	60	60	60	61	60	60
23	vi.	30/03/2018 09:00	2	2	2	2	1	2	59	59	59	60	59	59
24	sá.	31/03/2018 09:00	1	1	2	2	2	2	58	59	58	59	58	58
25	do.	01/04/2018 09:00	2	2	2	2	2	2	57	58	57	58	57	57
26	lu.	02/04/2018 09:00	2	2	2	2	2	2	56	57	56	57	56	56
27	ma.	03/04/2018 09:00	2	2	2	2	2	2	55	56	55	56	55	55
28	mi.	04/04/2018 09:00	2	2	2	2	2	2	54	55	54	55	54	54
29	ju.	05/04/2018 09:00	2	2	2	2	2	2	53	54	53	54	53	53
30	vi.	06/04/2018 09:00	2	2	2	1	2	2	52	53	52	53	52	52
31	sá.	07/04/2018 09:00	2	2	2	2	2	2	51	52	51	52	51	51

Figura 7.4 Prescripción página 2

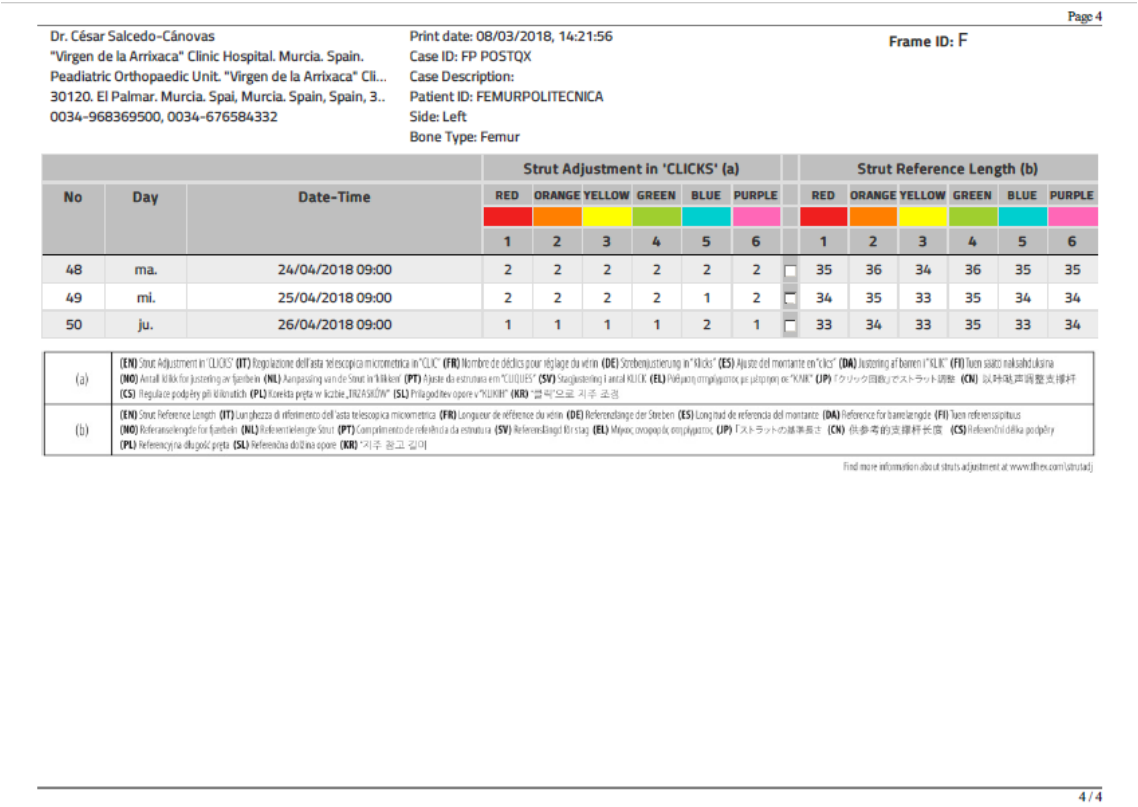
Dr. César Salcedo-Cánovas
"Virgen de la Arrixaca" Clinic Hospital. Murcia. Spain.
Peadiatric Orthopaedic Unit. "Virgen de la Arrixaca" Cli...
30120. El Palmar. Murcia. Spai, Murcia. Spain, Spain, 3..
0034-968369500, 0034-676584332

Print date: 08/03/2018, 14:21:56
Case ID: FP POSTQX
Case Description:
Patient ID: FEMURPOLITECNICA
Side: Left
Bone Type: Femur

Frame ID: F

No	Day	Date-Time	Strut Adjustment in 'CLICKS' (a)						Strut Reference Length (b)					
			RED	ORANGE	YELLOW	GREEN	BLUE	PURPLE	RED	ORANGE	YELLOW	GREEN	BLUE	PURPLE
			1	2	3	4	5	6	1	2	3	4	5	6
32	do.	08/04/2018 09:00	2	1	2	2	2	1	50	51	50	51	50	51
33	lu.	09/04/2018 09:00	2	2	2	2	2	2	49	50	49	50	49	50
34	ma.	10/04/2018 09:00	2	2	2	2	2	2	48	49	48	49	48	49
35	mi.	11/04/2018 09:00	2	2	1	2	1	2	47	48	47	48	48	48
36	ju.	12/04/2018 09:00	2	2	2	2	2	2	46	47	46	47	47	47
37	vi.	13/04/2018 09:00	1	2	2	2	2	2	46	46	45	46	46	46
38	sá.	14/04/2018 09:00	2	2	2	1	2	2	45	45	44	46	45	45
39	do.	15/04/2018 09:00	2	2	2	2	2	2	44	44	43	45	44	44
40	lu.	16/04/2018 09:00	2	2	2	2	2	2	43	43	42	44	43	43
41	ma.	17/04/2018 09:00	2	1	2	2	2	2	42	43	41	43	42	42
42	mi.	18/04/2018 09:00	2	2	2	2	2	2	41	42	40	42	41	41
43	ju.	19/04/2018 09:00	2	2	2	2	2	2	40	41	39	41	40	40
44	vi.	20/04/2018 09:00	2	2	2	2	2	1	39	40	38	40	39	39
45	sá.	21/04/2018 09:00	2	2	2	2	2	2	38	39	37	39	38	38
46	do.	22/04/2018 09:00	2	2	2	2	2	2	37	38	36	38	37	37
47	lu.	23/04/2018 09:00	2	2	2	1	2	2	36	37	35	37	36	36

Figura 7.5 Prescripción página 3



Page 4

Dr. César Salcedo-Cánovas
"Virgen de la Arrixaca" Clinic Hospital. Murcia. Spain.
Peadiatric Orthopaedic Unit. "Virgen de la Arrixaca" Cli...
30120. El Palmar. Murcia. Spai, Murcia. Spain, Spain, 3..
0034-968369500, 0034-676584332

Print date: 08/03/2018, 14:21:56
Case ID: FP POSTQX
Case Description:
Patient ID: FEMURPOLITECNICA
Side: Left
Bone Type: Femur

Frame ID: F

No	Day	Date-Time	Strut Adjustment in 'CLICKS' (a)						Strut Reference Length (b)					
			RED	ORANGE	YELLOW	GREEN	BLUE	PURPLE	RED	ORANGE	YELLOW	GREEN	BLUE	PURPLE
			1	2	3	4	5	6	1	2	3	4	5	6
48	ma.	24/04/2018 09:00	2	2	2	2	2	2	35	36	34	36	35	35
49	mi.	25/04/2018 09:00	2	2	2	2	1	2	34	35	33	35	34	34
50	ju.	26/04/2018 09:00	1	1	1	1	2	1	33	34	33	35	33	34

(a) **(EN)** Strut Adjustment in 'CLICKS' **(IT)** Regolazione dell'asta telescopica micrometrica in "CLIC" **(FR)** Nombre de déclics pour réglage du vérin **(DE)** Strebenjustierung in "Klicks" **(ES)** Ajuste del montante en "clics" **(DA)** Justering af banen i "KLIK" **(FI)** Tuen säätö naksahduksina **(NO)** Antall klikk for justering av fjæreben **(NL)** Aanpassing van de Strut in 'klikken' **(PT)** Ajuste da estrutura em "CLIQUES" **(SV)** Stagjustering i antal KLICK **(EL)** Ρύθμιση απομάγρυνσης με μέτρηση σε "ΚΛΙΚ" **(JP)** 「クリック回数」でのストラット調整 **(CN)** 以种"咔声"调整支撑杆 **(CS)** Regulace podpěry při kliknutích **(PL)** Korekta pręta w liczbie „TRZASKÓW" **(SL)** Prilagoditev opore v "KLIKIH" **(KR)** "클릭"으로 지주 조정

(b) **(EN)** Strut Reference Length **(IT)** Lunghezza di riferimento dell'asta telescopica micrometrica **(FR)** Longueur de référence du vérin **(DE)** Referenzlänge der Streben **(ES)** Longitud de referencia del montante **(DA)** Reference for barrelængde **(FI)** Tuen referenssipituus **(NO)** Referanselengde for fjæreben **(NL)** Referentielengte Strut **(PT)** Comprimento de referência da estrutura **(SV)** Referenslängd Kl strut **(EL)** Μήκος αναφοράς απομάγρυνσης **(JP)** 「ストラットの基準長さ **(CN)** 供参考的支撑杆长度 **(CS)** Referenční délka podpěry **(PL)** Referencyjna długość pręta **(SL)** Referenčna dolžina opore **(KR)** '지주 참고 길이'

Find more information about struts adjustment at www.ilhex.com/strutadj

4 / 4

Figura 7.6 Prescripción página 4

En las siguientes Figuras 7.7 y 7.8 se muestran dos perspectivas de los sensores y electrónica de adquisición y envío de los datos utilizado en las pruebas finales de los sensores:

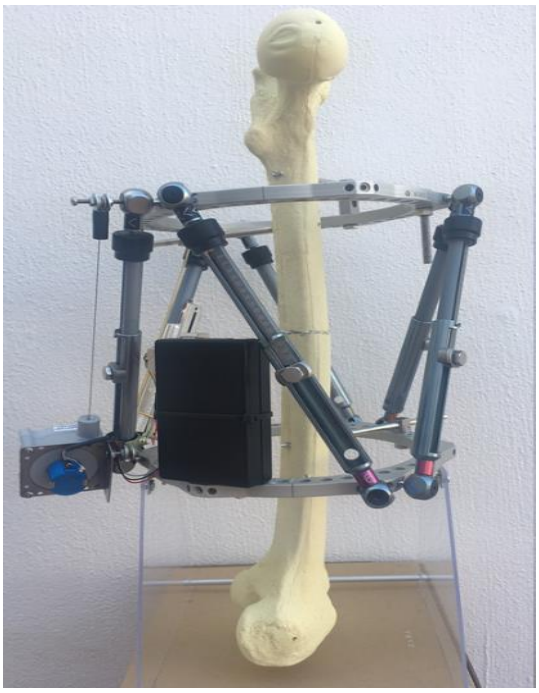


Figura 7.7 Primera perspectiva de integración final electrónica y sensores

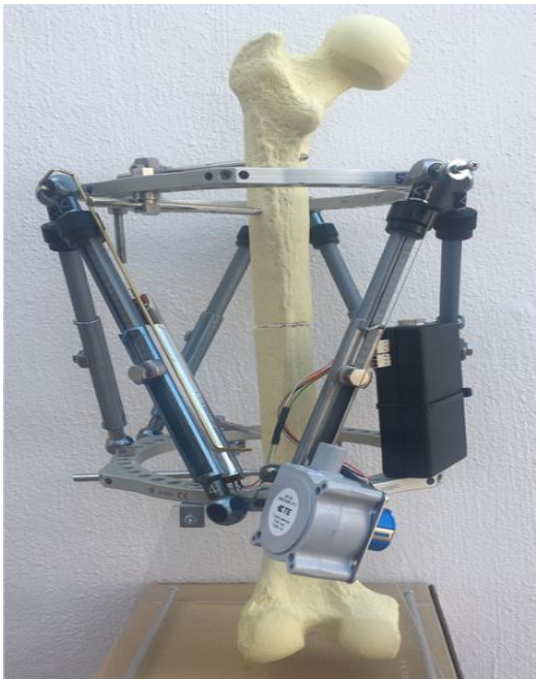


Figura 7.8 Segunda perspectiva de integración final electrónica y sensores

7.3 Resultados

En la Tablas 7.1, podemos ver los resultados que se obtienen para los actuadores 4 y 5. Cada tabla se divide en tres columnas que corresponden a los valores de la medida gradual en mm de:

- Prescripción: es el valor que esperamos obtener al realizar el ajuste por “clicks”.
- Interfaz: es el valor que obtenemos desde Arduino utilizando los sensores.
- Actuador: es el valor que marca la regleta del actuador.

Los valores resaltados corresponden a medidas que difieren de lo previsto en la prescripción.

STRUT 4			
Día	Preinscripción (mm)	Interfaz (mm)	Actuador (mm)
0	80	80	80
1	80	80	80
2	79	79	79
3	78	78	78
4	77	77	77
5	76	76	76
6	75	75	75
7	74	74	74
8	73	73	73
9	73	73	73
10	72	72	72
11	71	71	71
12	70	70	70
13	69	69	69
14	68	68	68
15	67	67	67
16	66	67	65
17	65	66	64
18	64	65	63
19	63	64	62
20	62	63	61
21	61	62	60
22	61	61	60
23	60	60	60
24	59	59	59
25	58	58	58
26	57	57	57
27	56	56	56
28	55	55	55
29	54	54	54
30	53	54	53
31	52	53	51
32	51	52	50
33	50	51	49
34	49	50	48
35	48	49	47
36	47	48	46
37	46	47	45
38	46	46	45
39	45	45	45
40	44	44	44
41	43	43	43
42	42	42	42
43	41	41	41
44	40	40	40
45	39	39	39
46	38	38	38
47	37	37	37
48	36	36	36
49	35	35	35
50	35	35	35

STRUT 5			
Día	Preinscripción (mm)	Interfaz (mm)	Actuador (mm)
0	80	80	80
1	80	80	80
2	79	79	79
3	78	78	78
4	77	77	77
5	76	76	76
6	75	75	75
7	74	74	74
8	73	73	73
9	72	72	72
10	71	71	71
11	70	70	70
12	70	70	70
13	69	69	69
14	68	68	68
15	67	67	67
16	66	66	66
17	65	65	65
18	64	64	64
19	63	62	63
20	62	61	62
21	61	60	61
22	60	59	60
23	59	59	59
24	58	58	58
25	57	56	57
26	56	55	56
27	55	54	55
28	54	53	54
29	53	52	54
30	52	51	52
31	51	50	51
32	50	49	51
33	49	48	50
34	48	47	49
35	48	46	48
36	47	45	47
37	46	44	46
38	45	43	45
39	44	42	44
40	43	41	43
41	42	42	42
42	41	39	41
43	40	38	40
44	39	36	39
45	38	35	38
46	37	34	37
47	36	33	36
48	35	32	35
49	34	32	34
50	33	31	33

Figura 7.9 Resultados prueba experimental

Aunque por simple observación parece que los resultados son bastante buenos, vamos a realizar un análisis estadístico para corroborarlo.

El análisis que se ha elegido es ANOVA de medidas repetidas, ya que se tratan de variables dependientes, es decir, se realiza la misma medida bajo tres condiciones distintas [40]. Es una prueba estadística que sirve para comprar las medias que se obtienen de distintas valoraciones (más de dos, sino sería una T student) sobre un mismo grupo.

Las hipótesis que vamos a contrastar son las siguientes:

$$H_0: \mu_1 = \mu_2 = \mu_3$$

$$H_1: \mu_i \neq \mu_j$$

Utilizando ANOVA queremos contrastar la hipótesis nula de que las medias entre las condiciones prescripción, interfaz y actuador coinciden, es decir, que no hay diferencias significativas entre los dispositivos.

Para el análisis se ha utilizado el programa R junto a RStudio. Las líneas de código que debemos escribir son las siguientes:

```
boxplot(strut4)
datos=c(strut4$PRE,strut4$INTER,strut4$ACT)
disp=c(rep("PRE",51),rep("INTER",51),rep("ACT",51))
conjunto=data.frame(datos,disp)
modelo=lm(datos~disp,data=conjunto)
anova(modelo)
```

7.3.1 Actuador cuatro

Lo primero que haremos será analizar los "Diagramas de cajas y bigotes" para saber si es correcto aplicar ANOVA o si debemos realizar otra prueba estadística distinta. En la Figura 7.7 podemos ver los diagramas obtenidos para los distintos dispositivos en el actuador 4:

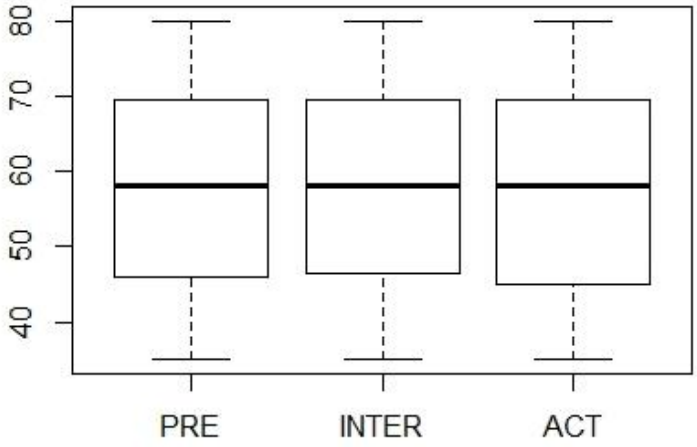


Figura 7.10 Diagramas de cajas y bigotes de actuador 4

De los diagramas de cajas y bigotes podemos deducir que se trata de distribuciones bastante simétricas, ya que el valor de la mediana se encuentra centrado entre el primer y el tercer cuartil y además la longitud de los bigotes son prácticamente iguales. No se aprecian valores atípicos [41].

Por último, este gráfico sirve también para comparar varias distribuciones. En la figura tenemos tres distribuciones, aparentemente normales y con medianas prácticamente iguales. Como queremos hacer un contraste de hipótesis sobre la igualdad de sus medias, primero tenemos que saber si sus varianzas son iguales, es decir, si existe homocedasticidad para saber si podemos aplicar ANOVA. Observando las tres distribuciones, vemos que la amplitud de la caja y de los bigotes es prácticamente la misma en los tres casos, pudiendo asumir igualdad de varianzas [42]. Por tanto, podremos aplicar ANOVA de medidas repetidas para el análisis:

El resultado que obtenemos de aplicar ANOVA utilizando RStudio es el siguiente:

```
Analysis of Variance Table

Response: datos
           Df  Sum Sq Mean Sq F value Pr(>F)
disp        2     8.2   4.124  0.0217 0.9786
Residuals 150 28539.0 190.260
```

La primera columna son los grados de libertad del modelo y de los residuos, que se obtienen con las ecuaciones:

Grados de libertad del modelo (disp):

$$Gl_M = K - 1 = 2$$

Siendo k el número de dispositivos/condiciones =3

Grados de libertad de los residuos (Residuals):

Grados de libertad totales:

$$Gl_T = N - 1 = 152$$

Con N= 51 muestras x 3 condiciones = 153

Entonces:

$$Gl_R = Gl_T - Gl_M = 150$$

La segunda columna son las sumas de cuadrados del modelo y de los residuos, que se obtienen con las ecuaciones:

Suma de cuadrados modelo:

$$SSM = \sum n_k(\overline{x_k} - x_{total})^2$$

Donde los términos de la ecuación son:

n_k: Número de muestras en cada condición = 51

$\overline{x_k}$: Media de cada condición

x_{total}: Media total de las condiciones

Suma de cuadrados de los residuos:

$$SSR = \sum {S_k}^2(n_k - 1)$$

${S_k}^2$: Varianza de cada condición

n_k: Número de muestras en cada condición

La tercera columna son las medias cuadráticas que se obtienen de dividir la suma de cuadrados entre los grados de libertad:

$$MS_M = \frac{SS_M}{Gl_M}$$

$$MS_R = \frac{SS_R}{Gl_R}$$

La cuarta columna es el valor del **estadístico F** que se obtiene de dividir la media cuadrática del modelo entre la de los residuos.

$$F = \frac{MS_M}{MS_R}$$

Y por último, el **Pvalor**.

El valor del estadístico F obtenido es de 0.0217 mientras que el teórico obtenido de la tabla F de Snedecor $F_{0,095}(2/150) = 3{,}0563$. El Pvalor es de 0,9786, lo que significa que es muchísimo mayor que 0,05. Por tanto, al ser el valor de F menor que el teórico y el Pvalor grande, se acepta la hipótesis nula, es decir, no existen diferencias significativas entre prescripción, interfaz y actuador.

7.3.2 Actuador cinco

En la Figura 7.8 podemos ver los diagramas obtenidos para los distintos dispositivos en el actuador 5:

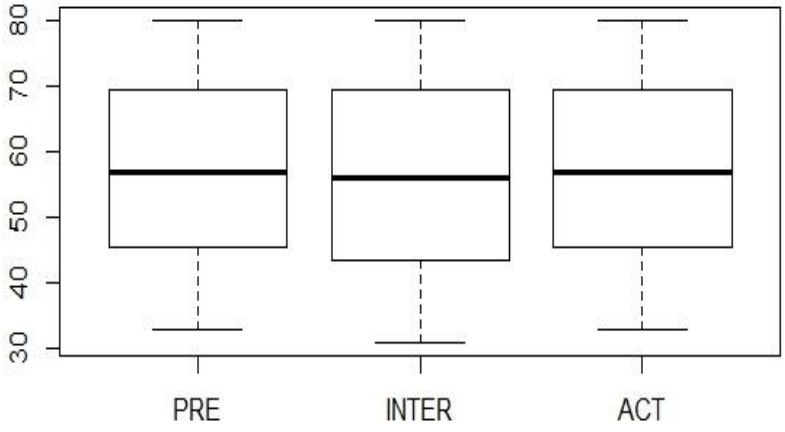


Figura 7.11 Diagramas de cajas y bigotes de actuador 5

Las conclusiones obtenidas para este actuador son las mismas que para el anterior. Las tres distribuciones son aparentemente normales con medianas prácticamente iguales. Además existe homocedasticidad, al comprobar que la amplitud de la caja y de los bigotes es prácticamente la misma en los tres casos.

El resultado que obtenemos de aplicar ANOVA utilizando RStudio es el siguiente:

```
Analysis of Variance Table

Response: datos
           Df  Sum Sq Mean Sq F value Pr(>F)
disp        2    34.2  17.078  0.0819 0.9214
Residuals 150 31263.4 208.422
```

El valor del estadístico F obtenido es de 0.0819 mientras que el teórico obtenido de la tabla F de Snedecor $F_{0,095}(2/150) = 3{,}0563$. El Pvalor es de 0,9786; lo que significa que es muchísimo mayor que 0,05. Por tanto, al ser el valor de F menor que el teórico y el Pvalor grande, se acepta la hipótesis nula, es decir, no existen diferencias significativas entre prescripción, interfaz y actuador.

Capítulo 8

Conclusiones y Trabajos Futuros

8.1 Conclusiones

Se ha cumplido el fin último del TFG, conseguir la sensorización de los actuadores mecánicos del sistema TL-Hex. Se ha demostrado que un sensor de tipo mecánico, como los potenciómetros, son los más adecuados para conocer con precisión y mediante una electrónica de prototipado no industrial las posiciones de los actuadores. Los potenciómetros fueron también la solución utilizada en el sistema del "tobillo de Rutgers".

La integración actual corresponde a un primer prototipo para pruebas, el cual se puede mejorar en el futuro integrando los potenciómetros deslizantes en el proceso de mecanizado del actuador.

También, la interfaz desarrollada en Processing ha quedado funcionando correctamente. Los datos enviados desde las distintas posiciones de los actuadores (con precisión de un milímetro) son adquiridos y procesados por la tarjeta Arduino y enviados vía Bluetooth al PC donde la interfaz muestra los datos. Además, en un futuro, la misma podrá permitir al médico comprobar desde su propia consulta si el paciente está siguiendo correctamente la prescripción.

Algo importante a destacar es que mediante el presente trabajo ha quedado demostrada la posibilidad de sensorizar el sistema TL-Hex. Esto permite visualizar en pantalla de manera precisa, la posición de los actuadores mecánicos en cada momento, sin necesidad de mirar las escalas de los mismos, facilitando en toda circunstancia no sólo la labor de los médicos sino fundamentalmente a los pacientes. Estos pacientes en el futuro y mediante una interfaz apropiada podrán con seguridad interactuar con el sistema siguiendo la prescripción médica.

También se ha dejado preparada la interfaz para que en un futuro, si se desea, se pueda hacer una ampliación de la misma sensorizando también el ajuste rápido de los actuadores.

En cualquier caso, con mínimo impacto mecánico del sistema y utilizando los potenciómetros deslizantes con una tarjeta procesadora adecuada se puede lograr la sensorización del ajuste gradual de todos los actuadores simultáneamente.

8.2 Aspectos a Mejorar

A continuación se detallan los principales aspectos a mejorar del trabajo:

8.2.1 Ampliar búsqueda de sensores

Como resultado del rápido desarrollo de la tecnología, en un futuro, podemos esperar encontrar sensores de dimensiones reducidas que puedan satisfacer las condiciones de precisión de nuestro proyecto. Siempre pensando en sensores que no requieran cambios mecánicos en los actuadores y permitiendo mantener las funcionalidades del sistema actual.

8.2.2 Consumo de Arduino

Al tratarse de un primer prototipo, este proyecto no está preparado para tener aplicación en un entorno real. Debido principalmente al consumo del microcontrolador Arduino utilizado para el desarrollo del mismo.

Actualmente, el software desarrollado en el IDE de Arduino para el cálculo de las posiciones necesita estar en funcionamiento continuo durante los días que dure el tratamiento.

Para conocer el consumo de nuestro Arduino [43] lo que debemos hacer es medir su consumo en vacío. Con ayuda de un polímetro se obtienen aproximadamente unos 46 mA. La pila de 9V utilizada nos ofrece aproximadamente 300 mAh.

Cabe destacar que el consumo en nuestro caso será mayor, debido al programa cargado. Arduino como tarjeta de prototipado es una tarjeta procesadora excepcional aunque desde el punto de vista de la eficiencia energética tiene algunas limitaciones.

Por tanto, podemos concluir que utilizar un Arduino Uno junto a una pila de 9 V no es la solución más adecuada como solución definitiva, por ello proponemos las siguientes alternativas:

Reemplazar el Arduino Uno por un Arduino Nano, ya que su consumo en vacío es menor y de aproximadamente 15 mA. Ello favorecerá que la integración en el sistema TL-Hex sea menos

voluminosa (ya que el tamaño del Arduino Nano es la mitad que el Arduino Uno) y compacta haciendo también más eficiente la solución desde el punto de vista de los pacientes.

Por otro lado, se propone sustituir la pila de 9V por otra que nos pueda ofrecer más mAh. Una primera solución podría ser incorporar una "Powerbank" utilizada hoy en día para cargar nuestros móviles.

Pero podemos ir más allá utilizando el modo "Sleep" de Arduino, esto ocurre de manera similar en los portátiles que cuando no los utilizamos entran en un modo bajo consumo [44].

Por ejemplo, se pueden encontrar referencias en la red donde utilizando el modo "dormir" en un Arduino Nano durante ocho segundos y manteniéndolo despierto durante dos segundos (momento en el que realiza el programa cargado en el Void Loop) obtenemos un consumo reducido a unos 4,8 mA aproximadamente.

8.2.3 Mejorar el software

Por otro lado, se plantea la mejora del software de la interfaz. Actualmente, cuando cerramos la ventana de la interfaz gráfica y la volvemos a abrir, podemos comprobar que hemos perdido el valor de las medidas que habíamos realizado el día anterior. Cada casilla de posición gradual tiene asignada una variable en el software de Processing, lo que se propone es que esas variables se guarden en memoria y no se borren al cerrar el programa.

8.3 Trabajos Futuros

El sistema ideal, desde el punto de vista médico sería aquel que realizase los ajustes de los actuadores durante un día completo de forma progresiva, en lugar de realizar un ajuste a una hora determinada del día.

En este sentido estamos hablando de motorizar y sensorizar actuadores que progresivamente siguieran la prescripción indicada por los médicos. Desde el punto de vista ingenieril, implicaría hacer funcionar un sistema durante todo el día y durante varios días seguidos sin interrupción. Visto los avances en cuanto a las baterías actuales, hoy en día, implicaría quizás hacer que el paciente ponga "a cargar" su sistema TL-Hex en algunos periodos determinados lo que sería una pequeña restricción.

Quizás lo más importante en el futuro, sería seguir dando pequeños pasos que permitan demostrar a la comunidad médica la posibilidad y beneficios que son capaces de proporcionar los robots controlados por ordenadores.

Acrónimos

A	Ajuste rápido "Acute"
A/D	Analógico/Digital
ANOVA	Análisis de Varianza"Analysis Of Variance"
AP	Anteroposterior
APP	Programa "Application"
AT	Atención
BOM	Lista De Materiales “Bill Of Material”
COM	comunicación
EDR	Velocidad de Datos Mejorada "Enhanced Data Rate"
EMA	Promediado Móvil Exponencial "Exponential Moving Average"
G	Ajuste gradual "Gradual"
GDL	Grados De Libertad
GND	Tierra "Ground"
HMI	Interfaz Humano Máquina "Human Machine Interface"
I2C	Inter-Integrated Circuit
IDE	Entorno de Desarrollo Integrado "Integrated Development Environment"
IOS	Sistema Operativo Iphone "Iphone Operating System"
LED	Diodo Emisor de Luz "light-emitting diode"
LIDAR	Detección de Luz y Medida o Detección de Imágenes por Láser y Medida"Light Detection and Ranging o Laser Imaging Detection and Ranging"
LRF	Buscador de Medida Láser Parallax "Parallax Laser Range Finder"
MAST	Multi-Axis Simulation Table
ML	Mediolateral
PC	Computadora Personal "personal computer"
PDF	Formato de Documento Portátil "Portable Document Format"
PIN	Número de Identificación Personal "Personal Identification Number"
Rx	Recepción, recibir o receptor
SCADA	Control de Supervisión y Adquisición de Datos "Supervisoy Control and Data Acquisition"
SCL	Línea de Reloj "Serial Clock Line"
SDA	Línea de Datos "Serial Data Line"
TFG	Trabajo de Fin de Grado
TL-Hex	Truelock Hexapod
TSRHC	Texas Scottish Rite Hospital for Children
Tx	Transmisión o transmitir
UPCT	Universidad Politécnica de Cartagena
USB	Bus Universal en Serie "Universal Serial Bus"

ANEXO I

Terminología Anatómica

El listado actual de términos anatómicos se creó y revisó en el año 1998 por el Comité Federativo de Terminología Anatómica (FCAT)

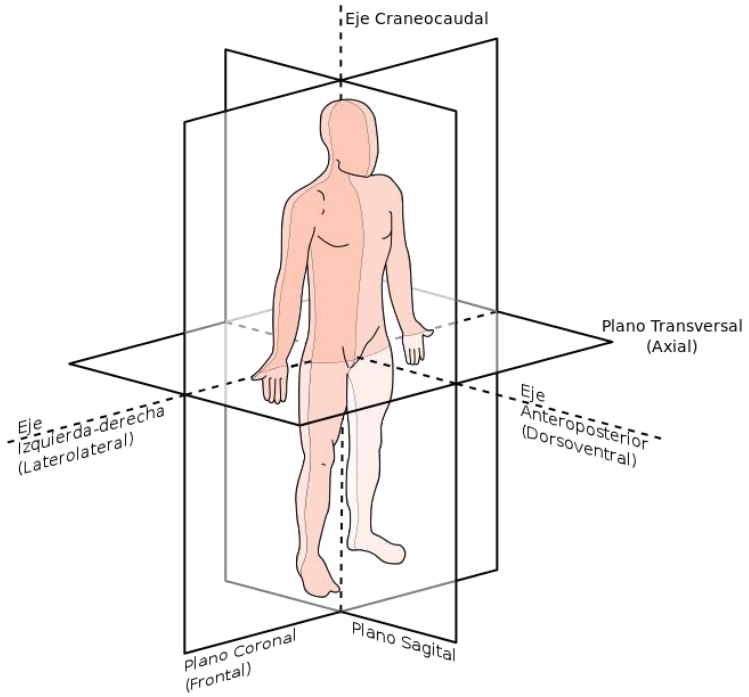
Términos de relación y comparación

Son los términos que definen la situación relativa entre las estructuras anatómicas:

- Anterior y posterior: hacia adelante o hacia atrás respectivamente (también se les llama ventral o dorsal).
- Lateral y medial: Alejar o acercarse del plano sagital medio.
- Superior e inferior: ubicado sobre o debajo de alguna estructura.
- Distal y proximal: Más alejado o cercano a alguna estructura determinada.
- Superficial y profundo: Más cerca o más lejos de la superficie.
- Externo e interno: Más alejado o más cercano al centro de un órgano.

Planimetría

Para la descripción y orientación de las partes del cuerpo, se usan cortes que seccionan el cuerpo en diferentes partes, llamados "Planos". Estos planos pueden verse en la figura a continuación:



- Planos sagitales: son verticales y van de sentido anterior a posterior.
  - Plano sagital medio: es el plano sagital que divide el cuerpo en mitad izquierda y mitad derecha.
  - Plano sagital paramedial: están paralelos al plano sagital medio.
- Planos coronales: son verticales, van de lateral a lateral y dividen el cuerpo en anterior y posterior.
- Planos horizontales: son planos transversales, perpendiculares a los verticales. Dividen el cuerpo en superior e inferior.

Para información más detallada visitar los enlaces:

https://es.wikiversity.org

https://periodicosalud.com

ANEXO II

Código desarrollado Arduino

```
// Trabajo de Fin de Grado. Laura Valdez Vidal GITI UPCT.
// Última modificación: 26/04/2018

float suma = 0;
int contador = 0;
float promedio = 0;
float voltaje = 0;
long mm = 0;
long inicio = 0;
long medida = 0;
long G = 0;

float suma2 = 0;
int contador2 = 0;
float promedio2 = 0;
float voltaje2 = 0;
long mm2 = 0;
long inicio2 = 0;
long medida2 = 0;
long G2 = 0;

byte parada;

void setup() {
  Serial.begin(9600);
  do {
    //Suma todas las lecturas de A1 POT.DESLIZANTE
    suma += analogRead(1);
    contador++;
  } while (contador < 1200);
  //Cálculo del promedio
  promedio = (float) (suma / contador);
  //Pasa el valor del promedio a voltios
  voltaje = promedio * (5.0 / 1023.0);
  //Valor en mm
  inicio = voltaje * (-19.11) + 97.508;
  //Preparo las variables para el siguiente promediado
  suma = 0;
  contador = 0;

  do {
    //Suma todas las lecturas de A0 POT.HILO
    suma2 += analogRead(0);
    contador2++;
```

```
  } while (contador2 < 1200);
  //Cálculo del promedio
  promedio2 = (float) (suma2 / contador2);
  //Pasa el valor del promedio a voltios
  voltaje2 = promedio2 * (5.0 / 1023.0);
  //Valor en mm
  inicio2 = voltaje2 * (145.3) + 2.1919;
  //Preparo las variables para el siguiente promediado
  suma2 = 0;
  contador2 = 0;
}

void loop() {
  if (Serial.available() > 0)
  {
    byte dato = Serial.read();

    //Esperamos la letra D (ASCII 68) del strut 4
    if (dato == 68 && dato != 69)
    {
      do {

        do {
          //Suma todas las lecturas de A1 POT.DESLIZANTA
          suma += analogRead(1);
          contador++;
        } while (contador < 1200);
        //Cálculo del promedio
        promedio = (float) (suma / contador);
        //Pasa el valor del promedio a voltios
        voltaje = promedio * (5.0 / 1023.0);
        //Valor en mm
        mm = voltaje * (-19.11) + 97.508;
        //Diferencia posición actual e inicial
        medida = mm - inicio;
        // valor de la posición gradual en mm
        G = -(abs(medida)) + 80;
        Serial.write(G);
        delay(25);

        //Comprobamos la parada
        parada = Serial.read();
        delay(100);
        //Preparo las variables para el siguiente promediado
        suma = 0;
        contador = 0;
      } while (parada != 68);
    }

    //Esperamos la letra E (ASCII 69) del strut 5
    else if (dato == 69 && dato != 68)
    {
      do {
```

```
      do {
        //Suma todas las lecturas de A0 POT. HILO
        suma2 += analogRead(0);
        contador2++;
      } while (contador2 < 1200);
      //Cálculo del promedio
      promedio2 = (float) (suma2 / contador2);
      //Pasa el valor del promedio a voltios
      voltaje2 = promedio2 * (5.0 / 1023.0);
      //Valor en mm
      mm2 = voltaje2 * (145.3) + 2.1919;
      //Diferencia posición actual e inicial
      medida2 = mm2 - inicio2;
      // valor de la posición gradual en mm
      G2 = -medida2 + 80;
      Serial.write(G2);
      delay(25);

      parada = Serial.read();
      delay(100);
      //Preparo las variables para el siguiente promediado
      suma2 = 0;
      contador2 = 0;
    } while (parada != 69);
  }
 }
}
```

ANEXO III

Código desarrollado Processing

Primera pestaña

```
// Trabajo Fin de Grado. Laura Valdez Vidal GITI UPCT.
// Última modificación: 26/04/2018

ArrayList<TEXTBOX> textboxes = new ArrayList<TEXTBOX>();
import processing.serial.*;
Serial port;
int bquad=30;
int aquad=15;
//////////////////////////VARIABLES RECTANGULOS LECTURA G//////////////////////////
//////COLUMNA STRUT 1//////
//CELDAS//
int xquad01=100;
int yquad01=80;
int yquad11=95;
int yquad21=110;
int yquad31=125;
int yquad41=140;
int yquad51=155;
int yquad61=170;
int yquad71=185;
int yquad81=200;
int yquad91=215;
int yquad101=230;
int yquad111=245;
int yquad121=260;
int yquad131=275;
int yquad141=290;
int yquad151=305;
int yquad161=320;
int yquad171=335;
int yquad181=350;
int yquad191=365;
int yquad201=380;
int yquad211=395;
int yquad221=410;
int yquad231=425;
int yquad241=440;
int yquad251=455;
int yquad261=470;
int yquad271=485;
int yquad281=500;
int yquad291=515;
int yquad301=530;
```

```
int yquad311=545;
//VARIABLES BOOLEAN//
boolean overRect01=false;
boolean overRect11=false;
boolean overRect21=false;
boolean overRect31=false;
boolean overRect41=false;
boolean overRect51=false;
boolean overRect61=false;
boolean overRect71=false;
boolean overRect81=false;
boolean overRect91=false;
boolean overRect101=false;
boolean overRect111=false;
boolean overRect121=false;
boolean overRect131=false;
boolean overRect141=false;
boolean overRect151=false;
boolean overRect161=false;
boolean overRect171=false;
boolean overRect181=false;
boolean overRect191=false;
boolean overRect201=false;
boolean overRect211=false;
boolean overRect221=false;
boolean overRect231=false;
boolean overRect241=false;
boolean overRect251=false;
boolean overRect261=false;
boolean overRect271=false;
boolean overRect281=false;
boolean overRect291=false;
boolean overRect301=false;
boolean overRect311=false;
//Colores de los botones//
int red01=224;
int green01=224;
int blue01=224;
int red11=224;
int green11=224;
int blue11=224;
int red21=224;
int green21=224;
int blue21=224;
int red31=224;
int green31=224;
int blue31=224;
int red41=224;
int green41=224;
int blue41=224;
int red51=224;
int green51=224;
int blue51=224;
```

```
int red61=224;
int green61=224;
int blue61=224;
int red71=224;
int green71=224;
int blue71=224;
int red81=224;
int green81=224;
int blue81=224;
int red91=224;
int green91=224;
int blue91=224;
int red101=224;
int green101=224;
int blue101=224;
int red111=224;
int green111=224;
int blue111=224;
int red121=224;
int green121=224;
int blue121=224;
int red131=224;
int green131=224;
int blue131=224;
int red141=224;
int green141=224;
int blue141=224;
int red151=224;
int green151=224;
int blue151=224;
int red161=224;
int green161=224;
int blue161=224;
int red171=224;
int green171=224;
int blue171=224;
int red181=224;
int green181=224;
int blue181=224;
int red191=224;
int green191=224;
int blue191=224;
int red201=224;
int green201=224;
int blue201=224;
int red211=224;
int green211=224;
int blue211=224;
int red221=224;
int green221=224;
int blue221=224;
int red231=224;
int green231=224;
```

```
int blue231=224;
int red241=224;
int green241=224;
int blue241=224;
int red251=224;
int green251=224;
int blue251=224;
int red261=224;
int green261=224;
int blue261=224;
int red271=224;
int green271=224;
int blue271=224;
int red281=224;
int green281=224;
int blue281=224;
int red291=224;
int green291=224;
int blue291=224;
int red301=224;
int green301=224;
int blue301=224;
int red311=224;
int green311=224;
int blue311=224;
//VARIABLES BOOLEAN ESTADOS//
boolean status01=false;
boolean status11=false;
boolean status21=false;
boolean status31=false;
boolean status41=false;
boolean status51=false;
boolean status61=false;
boolean status71=false;
boolean status81=false;
boolean status91=false;
boolean status101=false;
boolean status111=false;
boolean status121=false;
boolean status131=false;
boolean status141=false;
boolean status151=false;
boolean status161=false;
boolean status171=false;
boolean status181=false;
boolean status191=false;
boolean status201=false;
boolean status211=false;
boolean status221=false;
boolean status231=false;
boolean status241=false;
boolean status251=false;
boolean status261=false;
```

```
boolean status271=false;
boolean status281=false;
boolean status291=false;
boolean status301=false;
boolean status311=false;
//VARIABLES LECTURA PUERTO SERIE//
int valor01;
int valor11;
int valor21;
int valor31;
int valor41;
int valor51;
int valor61;
int valor71;
int valor81;
int valor91;
int valor101;
int valor111;
int valor121;
int valor131;
int valor141;
int valor151;
int valor161;
int valor171;
int valor181;
int valor191;
int valor201;
int valor211;
int valor221;
int valor231;
int valor241;
int valor251;
int valor261;
int valor271;
int valor281;
int valor291;
int valor301;
int valor311;
//////COLUMNA STRUT 2//////
//CELDAS//
int xquad02=220;
int yquad02=80;
int yquad12=95;
int yquad22=110;
int yquad32=125;
int yquad42=140;
int yquad52=155;
int yquad62=170;
int yquad72=185;
int yquad82=200;
int yquad92=215;
int yquad102=230;
int yquad112=245;
```

```
int yquad122=260;
int yquad132=275;
int yquad142=290;
int yquad152=305;
int yquad162=320;
int yquad172=335;
int yquad182=350;
int yquad192=365;
int yquad202=380;
int yquad212=395;
int yquad222=410;
int yquad232=425;
int yquad242=440;
int yquad252=455;
int yquad262=470;
int yquad272=485;
int yquad282=500;
int yquad292=515;
int yquad302=530;
int yquad312=545;
//VARIABLES BOOLEAN//
boolean overRect02=false;
boolean overRect12=false;
boolean overRect22=false;
boolean overRect32=false;
boolean overRect42=false;
boolean overRect52=false;
boolean overRect62=false;
boolean overRect72=false;
boolean overRect82=false;
boolean overRect92=false;
boolean overRect102=false;
boolean overRect112=false;
boolean overRect122=false;
boolean overRect132=false;
boolean overRect142=false;
boolean overRect152=false;
boolean overRect162=false;
boolean overRect172=false;
boolean overRect182=false;
boolean overRect192=false;
boolean overRect202=false;
boolean overRect212=false;
boolean overRect222=false;
boolean overRect232=false;
boolean overRect242=false;
boolean overRect252=false;
boolean overRect262=false;
boolean overRect272=false;
boolean overRect282=false;
boolean overRect292=false;
boolean overRect302=false;
boolean overRect312=false;
```

```
//Colores de los botones//
int red02=224;
int green02=224;
int blue02=224;
int red12=224;
int green12=224;
int blue12=224;
int red22=224;
int green22=224;
int blue22=224;
int red32=224;
int green32=224;
int blue32=224;
int red42=224;
int green42=224;
int blue42=224;
int red52=224;
int green52=224;
int blue52=224;
int red62=224;
int green62=224;
int blue62=224;
int red72=224;
int green72=224;
int blue72=224;
int red82=224;
int green82=224;
int blue82=224;
int red92=224;
int green92=224;
int blue92=224;
int red102=224;
int green102=224;
int blue102=224;
int red112=224;
int green112=224;
int blue112=224;
int red122=224;
int green122=224;
int blue122=224;
int red132=224;
int green132=224;
int blue132=224;
int red142=224;
int green142=224;
int blue142=224;
int red152=224;
int green152=224;
int blue152=224;
int red162=224;
int green162=224;
int blue162=224;
int red172=224;
```

```
int green172=224;
int blue172=224;
int red182=224;
int green182=224;
int blue182=224;
int red192=224;
int green192=224;
int blue192=224;
int red202=224;
int green202=224;
int blue202=224;
int red212=224;
int green212=224;
int blue212=224;
int red222=224;
int green222=224;
int blue222=224;
int red232=224;
int green232=224;
int blue232=224;
int red242=224;
int green242=224;
int blue242=224;
int red252=224;
int green252=224;
int blue252=224;
int red262=224;
int green262=224;
int blue262=224;
int red272=224;
int green272=224;
int blue272=224;
int red282=224;
int green282=224;
int blue282=224;
int red292=224;
int green292=224;
int blue292=224;
int red302=224;
int green302=224;
int blue302=224;
int red312=224;
int green312=224;
int blue312=224;
//VARIABLES BOOLEAN ESTADOS//
boolean status02=false;
boolean status12=false;
boolean status22=false;
boolean status32=false;
boolean status42=false;
boolean status52=false;
boolean status62=false;
boolean status72=false;
```

```
boolean status82=false;
boolean status92=false;
boolean status102=false;
boolean status112=false;
boolean status122=false;
boolean status132=false;
boolean status142=false;
boolean status152=false;
boolean status162=false;
boolean status172=false;
boolean status182=false;
boolean status192=false;
boolean status202=false;
boolean status212=false;
boolean status222=false;
boolean status232=false;
boolean status242=false;
boolean status252=false;
boolean status262=false;
boolean status272=false;
boolean status282=false;
boolean status292=false;
boolean status302=false;
boolean status312=false;
//VARIABLES LECTURA PUERTO SERIE//
int valor02;
int valor12;
int valor22;
int valor32;
int valor42;
int valor52;
int valor62;
int valor72;
int valor82;
int valor92;
int valor102;
int valor112;
int valor122;
int valor132;
int valor142;
int valor152;
int valor162;
int valor172;
int valor182;
int valor192;
int valor202;
int valor212;
int valor222;
int valor232;
int valor242;
int valor252;
int valor262;
int valor272;
```

```
int valor282;
int valor292;
int valor302;
int valor312;
//////COLUMNA STRUT 3//////
//CELDAS//
int xquad03=340;
int yquad03=80;
int yquad13=95;
int yquad23=110;
int yquad33=125;
int yquad43=140;
int yquad53=155;
int yquad63=170;
int yquad73=185;
int yquad83=200;
int yquad93=215;
int yquad103=230;
int yquad113=245;
int yquad123=260;
int yquad133=275;
int yquad143=290;
int yquad153=305;
int yquad163=320;
int yquad173=335;
int yquad183=350;
int yquad193=365;
int yquad203=380;
int yquad213=395;
int yquad223=410;
int yquad233=425;
int yquad243=440;
int yquad253=455;
int yquad263=470;
int yquad273=485;
int yquad283=500;
int yquad293=515;
int yquad303=530;
int yquad313=545;
//VARIABLES BOOLEAN//
boolean overRect03=false;
boolean overRect13=false;
boolean overRect23=false;
boolean overRect33=false;
boolean overRect43=false;
boolean overRect53=false;
boolean overRect63=false;
boolean overRect73=false;
boolean overRect83=false;
boolean overRect93=false;
boolean overRect103=false;
boolean overRect113=false;
boolean overRect123=false;
```

```
boolean overRect133=false;
boolean overRect143=false;
boolean overRect153=false;
boolean overRect163=false;
boolean overRect173=false;
boolean overRect183=false;
boolean overRect193=false;
boolean overRect203=false;
boolean overRect213=false;
boolean overRect223=false;
boolean overRect233=false;
boolean overRect243=false;
boolean overRect253=false;
boolean overRect263=false;
boolean overRect273=false;
boolean overRect283=false;
boolean overRect293=false;
boolean overRect303=false;
boolean overRect313=false;
//Colores de los botones//
int red03=224;
int green03=224;
int blue03=224;
int red13=224;
int green13=224;
int blue13=224;
int red23=224;
int green23=224;
int blue23=224;
int red33=224;
int green33=224;
int blue33=224;
int red43=224;
int green43=224;
int blue43=224;
int red53=224;
int green53=224;
int blue53=224;
int red63=224;
int green63=224;
int blue63=224;
int red73=224;
int green73=224;
int blue73=224;
int red83=224;
int green83=224;
int blue83=224;
int red93=224;
int green93=224;
int blue93=224;
int red103=224;
int green103=224;
int blue103=224;
```

```
int red113=224;
int green113=224;
int blue113=224;
int red123=224;
int green123=224;
int blue123=224;
int red133=224;
int green133=224;
int blue133=224;
int red143=224;
int green143=224;
int blue143=224;
int red153=224;
int green153=224;
int blue153=224;
int red163=224;
int green163=224;
int blue163=224;
int red173=224;
int green173=224;
int blue173=224;
int red183=224;
int green183=224;
int blue183=224;
int red193=224;
int green193=224;
int blue193=224;
int red203=224;
int green203=224;
int blue203=224;
int red213=224;
int green213=224;
int blue213=224;
int red223=224;
int green223=224;
int blue223=224;
int red233=224;
int green233=224;
int blue233=224;
int red243=224;
int green243=224;
int blue243=224;
int red253=224;
int green253=224;
int blue253=224;
int red263=224;
int green263=224;
int blue263=224;
int red273=224;
int green273=224;
int blue273=224;
int red283=224;
int green283=224;
```

```
int blue283=224;
int red293=224;
int green293=224;
int blue293=224;
int red303=224;
int green303=224;
int blue303=224;
int red313=224;
int green313=224;
int blue313=224;
//VARIABLES BOOLEAN ESTADOS//
boolean status03=false;
boolean status13=false;
boolean status23=false;
boolean status33=false;
boolean status43=false;
boolean status53=false;
boolean status63=false;
boolean status73=false;
boolean status83=false;
boolean status93=false;
boolean status103=false;
boolean status113=false;
boolean status123=false;
boolean status133=false;
boolean status143=false;
boolean status153=false;
boolean status163=false;
boolean status173=false;
boolean status183=false;
boolean status193=false;
boolean status203=false;
boolean status213=false;
boolean status223=false;
boolean status233=false;
boolean status243=false;
boolean status253=false;
boolean status263=false;
boolean status273=false;
boolean status283=false;
boolean status293=false;
boolean status303=false;
boolean status313=false;
//VARIABLES LECTURA PUERTO SERIE//
int valor03;
int valor13;
int valor23;
int valor33;
int valor43;
int valor53;
int valor63;
int valor73;
int valor83;
```

```
int valor93;
int valor103;
int valor113;
int valor123;
int valor133;
int valor143;
int valor153;
int valor163;
int valor173;
int valor183;
int valor193;
int valor203;
int valor213;
int valor223;
int valor233;
int valor243;
int valor253;
int valor263;
int valor273;
int valor283;
int valor293;
int valor303;
int valor313;
//////COLUMNA STRUT 4//////
//CELDAS//
int xquad04=460;
int yquad04=80;
int yquad14=95;
int yquad24=110;
int yquad34=125;
int yquad44=140;
int yquad54=155;
int yquad64=170;
int yquad74=185;
int yquad84=200;
int yquad94=215;
int yquad104=230;
int yquad114=245;
int yquad124=260;
int yquad134=275;
int yquad144=290;
int yquad154=305;
int yquad164=320;
int yquad174=335;
int yquad184=350;
int yquad194=365;
int yquad204=380;
int yquad214=395;
int yquad224=410;
int yquad234=425;
int yquad244=440;
int yquad254=455;
int yquad264=470;
```

```
int yquad274=485;
int yquad284=500;
int yquad294=515;
int yquad304=530;
int yquad314=545;
//VARIABLES BOOLEAN//
boolean overRect04=false;
boolean overRect14=false;
boolean overRect24=false;
boolean overRect34=false;
boolean overRect44=false;
boolean overRect54=false;
boolean overRect64=false;
boolean overRect74=false;
boolean overRect84=false;
boolean overRect94=false;
boolean overRect104=false;
boolean overRect114=false;
boolean overRect124=false;
boolean overRect134=false;
boolean overRect144=false;
boolean overRect154=false;
boolean overRect164=false;
boolean overRect174=false;
boolean overRect184=false;
boolean overRect194=false;
boolean overRect204=false;
boolean overRect214=false;
boolean overRect224=false;
boolean overRect234=false;
boolean overRect244=false;
boolean overRect254=false;
boolean overRect264=false;
boolean overRect274=false;
boolean overRect284=false;
boolean overRect294=false;
boolean overRect304=false;
boolean overRect314=false;
//Colores de los botones//
int red04=224;
int green04=224;
int blue04=224;
int red14=224;
int green14=224;
int blue14=224;
int red24=224;
int green24=224;
int blue24=224;
int red34=224;
int green34=224;
int blue34=224;
int red44=224;
int green44=224;
```

```
int blue44=224;
int red54=224;
int green54=224;
int blue54=224;
int red64=224;
int green64=224;
int blue64=224;
int red74=224;
int green74=224;
int blue74=224;
int red84=224;
int green84=224;
int blue84=224;
int red94=224;
int green94=224;
int blue94=224;
int red104=224;
int green104=224;
int blue104=224;
int red114=224;
int green114=224;
int blue114=224;
int red124=224;
int green124=224;
int blue124=224;
int red134=224;
int green134=224;
int blue134=224;
int red144=224;
int green144=224;
int blue144=224;
int red154=224;
int green154=224;
int blue154=224;
int red164=224;
int green164=224;
int blue164=224;
int red174=224;
int green174=224;
int blue174=224;
int red184=224;
int green184=224;
int blue184=224;
int red194=224;
int green194=224;
int blue194=224;
int red204=224;
int green204=224;
int blue204=224;
int red214=224;
int green214=224;
int blue214=224;
int red224=224;
```

```
int green224=224;
int blue224=224;
int red234=224;
int green234=224;
int blue234=224;
int red244=224;
int green244=224;
int blue244=224;
int red254=224;
int green254=224;
int blue254=224;
int red264=224;
int green264=224;
int blue264=224;
int red274=224;
int green274=224;
int blue274=224;
int red284=224;
int green284=224;
int blue284=224;
int red294=224;
int green294=224;
int blue294=224;
int red304=224;
int green304=224;
int blue304=224;
int red314=224;
int green314=224;
int blue314=224;
//VARIABLES BOOLEAN ESTADOS//
boolean status04=false;
boolean status14=false;
boolean status24=false;
boolean status34=false;
boolean status44=false;
boolean status54=false;
boolean status64=false;
boolean status74=false;
boolean status84=false;
boolean status94=false;
boolean status104=false;
boolean status114=false;
boolean status124=false;
boolean status134=false;
boolean status144=false;
boolean status154=false;
boolean status164=false;
boolean status174=false;
boolean status184=false;
boolean status194=false;
boolean status204=false;
boolean status214=false;
boolean status224=false;
```

```
boolean status234=false;
boolean status244=false;
boolean status254=false;
boolean status264=false;
boolean status274=false;
boolean status284=false;
boolean status294=false;
boolean status304=false;
boolean status314=false;
//VARIABLES LECTURA PUERTO SERIE//
int valor04;
int valor14;
int valor24;
int valor34;
int valor44;
int valor54;
int valor64;
int valor74;
int valor84;
int valor94;
int valor104;
int valor114;
int valor124;
int valor134;
int valor144;
int valor154;
int valor164;
int valor174;
int valor184;
int valor194;
int valor204;
int valor214;
int valor224;
int valor234;
int valor244;
int valor254;
int valor264;
int valor274;
int valor284;
int valor294;
int valor304;
int valor314;
//////COLUMNA STRUT 5//////
//CELDAS//
int xquad05=580;
int yquad05=80;
int yquad15=95;
int yquad25=110;
int yquad35=125;
int yquad45=140;
int yquad55=155;
int yquad65=170;
int yquad75=185;
```

```
int yquad85=200;
int yquad95=215;
int yquad105=230;
int yquad115=245;
int yquad125=260;
int yquad135=275;
int yquad145=290;
int yquad155=305;
int yquad165=320;
int yquad175=335;
int yquad185=350;
int yquad195=365;
int yquad205=380;
int yquad215=395;
int yquad225=410;
int yquad235=425;
int yquad245=440;
int yquad255=455;
int yquad265=470;
int yquad275=485;
int yquad285=500;
int yquad295=515;
int yquad305=530;
int yquad315=545;
//VARIABLES BOOLEAN//
boolean overRect05=false;
boolean overRect15=false;
boolean overRect25=false;
boolean overRect35=false;
boolean overRect45=false;
boolean overRect55=false;
boolean overRect65=false;
boolean overRect75=false;
boolean overRect85=false;
boolean overRect95=false;
boolean overRect105=false;
boolean overRect115=false;
boolean overRect125=false;
boolean overRect135=false;
boolean overRect145=false;
boolean overRect155=false;
boolean overRect165=false;
boolean overRect175=false;
boolean overRect185=false;
boolean overRect195=false;
boolean overRect205=false;
boolean overRect215=false;
boolean overRect225=false;
boolean overRect235=false;
boolean overRect245=false;
boolean overRect255=false;
boolean overRect265=false;
boolean overRect275=false;
```

```
boolean overRect285=false;
boolean overRect295=false;
boolean overRect305=false;
boolean overRect315=false;
//Colores de los botones//
int red05=224;
int green05=224;
int blue05=224;
int red15=224;
int green15=224;
int blue15=224;
int red25=224;
int green25=224;
int blue25=224;
int red35=224;
int green35=224;
int blue35=224;
int red45=224;
int green45=224;
int blue45=224;
int red55=224;
int green55=224;
int blue55=224;
int red65=224;
int green65=224;
int blue65=224;
int red75=224;
int green75=224;
int blue75=224;
int red85=224;
int green85=224;
int blue85=224;
int red95=224;
int green95=224;
int blue95=224;
int red105=224;
int green105=224;
int blue105=224;
int red115=224;
int green115=224;
int blue115=224;
int red125=224;
int green125=224;
int blue125=224;
int red135=224;
int green135=224;
int blue135=224;
int red145=224;
int green145=224;
int blue145=224;
int red155=224;
int green155=224;
int blue155=224;
```

```
int red165=224;
int green165=224;
int blue165=224;
int red175=224;
int green175=224;
int blue175=224;
int red185=224;
int green185=224;
int blue185=224;
int red195=224;
int green195=224;
int blue195=224;
int red205=224;
int green205=224;
int blue205=224;
int red215=224;
int green215=224;
int blue215=224;
int red225=224;
int green225=224;
int blue225=224;
int red235=224;
int green235=224;
int blue235=224;
int red245=224;
int green245=224;
int blue245=224;
int red255=224;
int green255=224;
int blue255=224;
int red265=224;
int green265=224;
int blue265=224;
int red275=224;
int green275=224;
int blue275=224;
int red285=224;
int green285=224;
int blue285=224;
int red295=224;
int green295=224;
int blue295=224;
int red305=224;
int green305=224;
int blue305=224;
int red315=224;
int green315=224;
int blue315=224;
//VARIABLES BOOLEAN ESTADOS//
boolean status05=false;
boolean status15=false;
boolean status25=false;
boolean status35=false;
```

```
boolean status45=false;
boolean status55=false;
boolean status65=false;
boolean status75=false;
boolean status85=false;
boolean status95=false;
boolean status105=false;
boolean status115=false;
boolean status125=false;
boolean status135=false;
boolean status145=false;
boolean status155=false;
boolean status165=false;
boolean status175=false;
boolean status185=false;
boolean status195=false;
boolean status205=false;
boolean status215=false;
boolean status225=false;
boolean status235=false;
boolean status245=false;
boolean status255=false;
boolean status265=false;
boolean status275=false;
boolean status285=false;
boolean status295=false;
boolean status305=false;
boolean status315=false;
//VARIABLES LECTURA PUERTO SERIE//
int valor05;
int valor15;
int valor25;
int valor35;
int valor45;
int valor55;
int valor65;
int valor75;
int valor85;
int valor95;
int valor105;
int valor115;
int valor125;
int valor135;
int valor145;
int valor155;
int valor165;
int valor175;
int valor185;
int valor195;
int valor205;
int valor215;
int valor225;
int valor235;
```

```
int valor245;
int valor255;
int valor265;
int valor275;
int valor285;
int valor295;
int valor305;
int valor315;
//////COLUMNA STRUT 6//////
//CELDAS//
int xquad06=700;
int yquad06=80;
int yquad16=95;
int yquad26=110;
int yquad36=125;
int yquad46=140;
int yquad56=155;
int yquad66=170;
int yquad76=185;
int yquad86=200;
int yquad96=215;
int yquad106=230;
int yquad116=245;
int yquad126=260;
int yquad136=275;
int yquad146=290;
int yquad156=305;
int yquad166=320;
int yquad176=335;
int yquad186=350;
int yquad196=365;
int yquad206=380;
int yquad216=395;
int yquad226=410;
int yquad236=425;
int yquad246=440;
int yquad256=455;
int yquad266=470;
int yquad276=485;
int yquad286=500;
int yquad296=515;
int yquad306=530;
int yquad316=545;
//VARIABLES BOOLEAN//
boolean overRect06=false;
boolean overRect16=false;
boolean overRect26=false;
boolean overRect36=false;
boolean overRect46=false;
boolean overRect56=false;
boolean overRect66=false;
boolean overRect76=false;
boolean overRect86=false;
```

```
boolean overRect96=false;
boolean overRect106=false;
boolean overRect116=false;
boolean overRect126=false;
boolean overRect136=false;
boolean overRect146=false;
boolean overRect156=false;
boolean overRect166=false;
boolean overRect176=false;
boolean overRect186=false;
boolean overRect196=false;
boolean overRect206=false;
boolean overRect216=false;
boolean overRect226=false;
boolean overRect236=false;
boolean overRect246=false;
boolean overRect256=false;
boolean overRect266=false;
boolean overRect276=false;
boolean overRect286=false;
boolean overRect296=false;
boolean overRect306=false;
boolean overRect316=false;
//Colores de los botones//
int red06=224;
int green06=224;
int blue06=224;
int red16=224;
int green16=224;
int blue16=224;
int red26=224;
int green26=224;
int blue26=224;
int red36=224;
int green36=224;
int blue36=224;
int red46=224;
int green46=224;
int blue46=224;
int red56=224;
int green56=224;
int blue56=224;
int red66=224;
int green66=224;
int blue66=224;
int red76=224;
int green76=224;
int blue76=224;
int red86=224;
int green86=224;
int blue86=224;
int red96=224;
int green96=224;
```

```
int blue96=224;
int red106=224;
int green106=224;
int blue106=224;
int red116=224;
int green116=224;
int blue116=224;
int red126=224;
int green126=224;
int blue126=224;
int red136=224;
int green136=224;
int blue136=224;
int red146=224;
int green146=224;
int blue146=224;
int red156=224;
int green156=224;
int blue156=224;
int red166=224;
int green166=224;
int blue166=224;
int red176=224;
int green176=224;
int blue176=224;
int red186=224;
int green186=224;
int blue186=224;
int red196=224;
int green196=224;
int blue196=224;
int red206=224;
int green206=224;
int blue206=224;
int red216=224;
int green216=224;
int blue216=224;
int red226=224;
int green226=224;
int blue226=224;
int red236=224;
int green236=224;
int blue236=224;
int red246=224;
int green246=224;
int blue246=224;
int red256=224;
int green256=224;
int blue256=224;
int red266=224;
int green266=224;
int blue266=224;
int red276=224;
```

```
int green276=224;
int blue276=224;
int red286=224;
int green286=224;
int blue286=224;
int red296=224;
int green296=224;
int blue296=224;
int red306=224;
int green306=224;
int blue306=224;
int red316=224;
int green316=224;
int blue316=224;
//VARIABLES BOOLEAN ESTADOS//
boolean status06=false;
boolean status16=false;
boolean status26=false;
boolean status36=false;
boolean status46=false;
boolean status56=false;
boolean status66=false;
boolean status76=false;
boolean status86=false;
boolean status96=false;
boolean status106=false;
boolean status116=false;
boolean status126=false;
boolean status136=false;
boolean status146=false;
boolean status156=false;
boolean status166=false;
boolean status176=false;
boolean status186=false;
boolean status196=false;
boolean status206=false;
boolean status216=false;
boolean status226=false;
boolean status236=false;
boolean status246=false;
boolean status256=false;
boolean status266=false;
boolean status276=false;
boolean status286=false;
boolean status296=false;
boolean status306=false;
boolean status316=false;
//VARIABLES LECTURA PUERTO SERIE//
int valor06;
int valor16;
int valor26;
int valor36;
int valor46;
```

```
int valor56;
int valor66;
int valor76;
int valor86;
int valor96;
int valor106;
int valor116;
int valor126;
int valor136;
int valor146;
int valor156;
int valor166;
int valor176;
int valor186;
int valor196;
int valor206;
int valor216;
int valor226;
int valor236;
int valor246;
int valor256;
int valor266;
int valor276;
int valor286;
int valor296;
int valor306;
int valor316;
///Estado de la medición///
String texto="No está realizando\nmedición";

void setup() {
  println(Serial.list());
  port=new Serial(this, Serial.list()[2], 9600);
  size(900, 700);
  InitLayout();
}

void draw() {
  background(255);
  int n=0;
  //Diseño primera columna
  for (int y=80; y<555; y=y+15)
  {
    stroke(0);
    fill(#FFFFFF);
    rect(30, y, 40, 15);
    if (n<32)
    {
      fill(0);
      PFont a=loadFont("ArialMT-11.vlw");
      textFont(a, 10);
      text(n, 40, y+12);
    }
```

```
  n++;
}
stroke(0);
fill(#FFFFFF);
rect(30, 45, 40, 35);
fill(0);
PFont f=loadFont("Arial-Black-11.vlw");
textFont(f, 10);
text("No", 45, 67);
fill(0);
textFont(f, 12);
text("A-Acute ; G-Gradual ; Strut Length", 320, 40);
//Diseño columna strut 1
stroke(0);
fill(#FFFFFF);
rect(70, 45, 120, 20);
fill(0);
textFont(f, 12);
text("Strut 1:Red", 90, 60);
for (int y=80; y<555; y=y+15)
{
  stroke(0);
  fill(#FFFFFF);
  rect(70, y, 30, 15);
}
stroke(0);
fill(#FF0009);
rect(70, 65, 30, 15);
fill(0);
textFont(f, 10);
text("A", 80, 77);
for (int y=80; y<555; y=y+15)
{
  stroke(0);
  fill(#FFFFFF);
  rect(100, y, 30, 15);
}
stroke(0);
fill(#FF0009);
rect(100, 65, 30, 15);
fill(0);
textFont(f, 10);
text("G", 110, 77);
for (int y=80; y<555; y=y+15)
{
  stroke(0);
  fill(#FFFFFF);
  rect(130, y, 60, 15);
}
stroke(0);
fill(#FF0009);
rect(130, 65, 60, 15);
fill(0);
```

```
textFont(f, 10);
text("Size", 145, 77);
//Diseño columna strut 2
stroke(0);
fill(#FFFFFF);
rect(190, 45, 120, 20);
fill(0);
textFont(f, 12);
text("Strut 2:Orange", 210, 60);
for (int y=80; y<555; y=y+15)
{
  stroke(0);
  fill(#FFFFFF);
  rect(190, y, 30, 15);
}
stroke(0);
fill(#FF9100);
rect(190, 65, 30, 15);
fill(0);
textFont(f, 10);
text("A", 200, 77);
for (int y=80; y<555; y=y+15)
{
  stroke(0);
  fill(#FFFFFF);
  rect(220, y, 30, 15);
}
stroke(0);
fill(#FF9100);
rect(220, 65, 30, 15);
fill(0);
textFont(f, 10);
text("G", 230, 77);
for (int y=80; y<555; y=y+15)
{
  stroke(0);
  fill(#FFFFFF);
  rect(250, y, 60, 15);
}
stroke(0);
fill(#FF9100);
rect(250, 65, 60, 15);
fill(0);
textFont(f, 10);
text("Size", 265, 77);
//Diseño columna strut 3
stroke(0);
fill(#FFFFFF);
rect(310, 45, 120, 20);
fill(0);
textFont(f, 12);
text("Strut 3:Yellow", 330, 60);
for (int y=80; y<555; y=y+15)
```

```
{
  stroke(0);
  fill(#FFFFFF);
  rect(310, y, 30, 15);
}
stroke(0);
fill(#FEFF03);
rect(310, 65, 30, 15);
fill(0);
textFont(f, 10);
text("A", 320, 77);
for (int y=80; y<555; y=y+15)
{
  stroke(0);
  fill(#FFFFFF);
  rect(340, y, 30, 15);
}
stroke(0);
fill(#FEFF03);
rect(340, 65, 30, 15);
fill(0);
textFont(f, 10);
text("G", 350, 77);
for (int y=80; y<555; y=y+15)
{
  stroke(0);
  fill(#FFFFFF);
  rect(370, y, 60, 15);
}
stroke(0);
fill(#FEFF03);
rect(370, 65, 60, 15);
fill(0);
textFont(f, 10);
text("Size", 385, 77);
//Diseño columna strut 4
stroke(0);
fill(#FFFFFF);
rect(430, 45, 120, 20);
fill(0);
textFont(f, 12);
text("Strut 4:Green", 450, 60);
for (int y=80; y<555; y=y+15)
{
  stroke(0);
  fill(#FFFFFF);
  rect(430, y, 30, 15);
}
stroke(0);
fill(#3CF72F);
rect(430, 65, 30, 15);
fill(0);
textFont(f, 10);
```

```
text("A", 440, 77);
for (int y=80; y<555; y=y+15)
{
  stroke(0);
  fill(#FFFFFF);
  rect(460, y, 30, 15);
}
stroke(0);
fill(#3CF72F);
rect(460, 65, 30, 15);
fill(0);
textFont(f, 10);
text("G", 470, 77);
for (int y=80; y<555; y=y+15)
{
  stroke(0);
  fill(#FFFFFF);
  rect(490, y, 60, 15);
}
stroke(0);
fill(#3CF72F);
rect(490, 65, 60, 15);
fill(0);
textFont(f, 10);
text("Size", 505, 77);
//Diseño columna strut 5
stroke(0);
fill(#FFFFFF);
rect(550, 45, 120, 20);
fill(0);
textFont(f, 12);
text("Strut 5:Blue", 570, 60);
for (int y=80; y<555; y=y+15)
{
  stroke(0);
  fill(#FFFFFF);
  rect(550, y, 30, 15);
}
stroke(0);
fill(#5BC9FA);
rect(550, 65, 30, 15);
fill(0);
textFont(f, 10);
text("A", 560, 77);
for (int y=80; y<555; y=y+15)
{
  stroke(0);
  fill(#FFFFFF);
  rect(580, y, 30, 15);
}
stroke(0);
fill(#5BC9FA);
rect(580, 65, 30, 15);
```

```
fill(0);
textFont(f, 10);
text("G", 590, 77);
for (int y=80; y<555; y=y+15)
{
  stroke(0);
  fill(#FFFFFF);
  rect(610, y, 60, 15);
}
stroke(0);
fill(#5BC9FA);
rect(610, 65, 60, 15);
fill(0);
textFont(f, 10);
text("Size", 625, 77);
//Diseño columna strut 6
stroke(0);
fill(#FFFFFF);
rect(670, 45, 120, 20);
fill(0);
textFont(f, 12);
text("Strut 6:Purple", 680, 60);
for (int y=80; y<555; y=y+15)
{
  stroke(0);
  fill(#FFFFFF);
  rect(670, y, 30, 15);
}
stroke(0);
fill(#FA77D3);
rect(670, 65, 30, 15);
fill(0);
textFont(f, 10);
text("A", 680, 77);
for (int y=80; y<555; y=y+15)
{
  stroke(0);
  fill(#FFFFFF);
  rect(700, y, 30, 15);
}
stroke(0);
fill(#FA77D3);
rect(700, 65, 30, 15);
fill(0);
textFont(f, 10);
text("G", 710, 77);
for (int y=80; y<555; y=y+15)
{
  stroke(0);
  fill(#FFFFFF);
  rect(730, y, 60, 15);
}
stroke(0);
```

```
fill(#FA77D3);
rect(730, 65, 60, 15);
fill(0);
textFont(f, 10);
text("Size", 745, 77);
///TEXTBOX///
for (TEXTBOX t : textboxes) {
  t.DRAW();
}
//LECTURA PUERTO SERIE//
////// COLUMNA G STRUT 1 ///////
if (mouseX > xquad01 && mouseX < xquad01+bquad &&
  mouseY > yquad01 && mouseY < yquad01+aquad)
{
  overRect01=true;
  stroke(70, 255, 0);
} else
{
  overRect01=false;
  stroke(0, 0, 0);
}
fill(red01, green01, blue01);
rect(xquad01, yquad01, bquad, aquad);
fill(0, 0, 0);
PFont b=loadFont("Arial-Black-18.vlw");
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status01==true)
  {
    valor01=port.read();
    delay(250);
    textFont(f, 12);
    text(valor01, 105, 92);
  }
  if (status01==false)
  {
    textFont(f, 12);
    text(valor01, 105, 92);
  }
}
if (mouseX > xquad01 && mouseX < xquad01+bquad &&
  mouseY > yquad11 && mouseY < yquad11+aquad)
{
  overRect11=true;
  stroke(70, 255, 0);
} else
{
  overRect11=false;
  stroke(0, 0, 0);
}
fill(red11, green11, blue11);
```

```
rect(xquad01, yquad11, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
 if (status11==true)
 {
  valor11=port.read();
  delay(250);
  textFont(f, 12);
  text(valor11, 105, 107);
 }
 if (status11==false)
 {
  textFont(f, 12);
  text(valor11, 105, 107);
 }
}
if (mouseX > xquad01 && mouseX < xquad01+bquad &&
 mouseY > yquad21 && mouseY < yquad21+aquad)
{
 overRect21=true;
 stroke(70, 255, 0);
} else
{
 overRect21=false;
 stroke(0, 0, 0);
}
fill(red21, green21, blue21);
rect(xquad01, yquad21, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
 if (status21==true)
 {
  valor21=port.read();
  delay(250);
  textFont(f, 12);
  text(valor21, 105, 122);
 }
 if (status21==false)
 {
  textFont(f, 12);
  text(valor21, 105, 122);
 }
}
if (mouseX > xquad01 && mouseX < xquad01+bquad &&
 mouseY > yquad31 && mouseY < yquad31+aquad)
{
 overRect31=true;
```

```
  stroke(70, 255, 0);
} else
{
  overRect31=false;
  stroke(0, 0, 0);
}
fill(red31, green31, blue31);
rect(xquad01, yquad31, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status31==true)
  {
    valor31=port.read();
    delay(250);
    textFont(f, 12);
    text(valor31, 105, 137);
  }
  if (status31==false)
  {
    textFont(f, 12);
    text(valor31, 105, 137);
  }
}
if (mouseX > xquad01 && mouseX < xquad01+bquad &&
  mouseY > yquad41 && mouseY < yquad41+aquad)
{
  overRect41=true;
  stroke(70, 255, 0);
} else
{
  overRect41=false;
  stroke(0, 0, 0);
}
fill(red41, green41, blue41);
rect(xquad01, yquad41, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status41==true)
  {
    valor41=port.read();
    delay(250);
    textFont(f, 12);
    text(valor41, 105, 152);
  }
  if (status41==false)
  {
    textFont(f, 12);
```

```
   text(valor41, 105, 152);
  }
}
if (mouseX > xquad01 && mouseX < xquad01+bquad &&
  mouseY > yquad51 && mouseY < yquad51+aquad)
{
  overRect51=true;
  stroke(70, 255, 0);
} else
{
  overRect51=false;
  stroke(0, 0, 0);
}
fill(red51, green51, blue51);
rect(xquad01, yquad51, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status51==true)
  {
    valor51=port.read();
    delay(250);
    textFont(f, 12);
    text(valor51, 105, 167);
  }
  if (status51==false)
  {
    textFont(f, 12);
    text(valor51, 105, 167);
  }
}
if (mouseX > xquad01 && mouseX < xquad01+bquad &&
  mouseY > yquad61 && mouseY < yquad61+aquad)
{
  overRect61=true;
  stroke(70, 255, 0);
} else
{
  overRect61=false;
  stroke(0, 0, 0);
}
fill(red61, green61, blue61);
rect(xquad01, yquad61, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status61==true)
  {
    valor61=port.read();
```

```
   delay(250);
   textFont(f, 12);
   text(valor61, 105, 182);
  }
  if (status61==false)
  {
   textFont(f, 12);
   text(valor61, 105, 182);
  }
}
if (mouseX > xquad01 && mouseX < xquad01+bquad &&
  mouseY > yquad71 && mouseY < yquad71+aquad)
{
  overRect71=true;
  stroke(70, 255, 0);
} else
{
  overRect71=false;
  stroke(0, 0, 0);
}
fill(red71, green71, blue71);
rect(xquad01, yquad71, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status71==true)
  {
   valor71=port.read();
   delay(250);
   textFont(f, 12);
   text(valor71, 105, 197);
  }
  if (status71==false)
  {
   textFont(f, 12);
   text(valor71, 105, 197);
  }
}
if (mouseX > xquad01 && mouseX < xquad01+bquad &&
  mouseY > yquad81 && mouseY < yquad81+aquad)
{
  overRect81=true;
  stroke(70, 255, 0);
} else
{
  overRect81=false;
  stroke(0, 0, 0);
}
fill(red81, green81, blue81);
rect(xquad01, yquad81, bquad, aquad);
fill(0, 0, 0);
```

```
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
 if (status81==true)
 {
  valor81=port.read();
  delay(250);
  textFont(f, 12);
  text(valor81, 105, 212);
 }
 if (status81==false)
 {
  textFont(f, 12);
  text(valor81, 105, 212);
 }
}
if (mouseX > xquad01 && mouseX < xquad01+bquad &&
 mouseY > yquad91 && mouseY < yquad91+aquad)
{
 overRect91=true;
 stroke(70, 255, 0);
} else
{
 overRect91=false;
 stroke(0, 0, 0);
}
fill(red91, green91, blue91);
rect(xquad01, yquad91, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
 if (status91==true)
 {
  valor91=port.read();
  delay(250);
  textFont(f, 12);
  text(valor91, 105, 227);
 }
 if (status91==false)
 {
  textFont(f, 12);
  text(valor91, 105, 227);
 }
}
if (mouseX > xquad01 && mouseX < xquad01+bquad &&
 mouseY > yquad101 && mouseY < yquad101+aquad)
{
 overRect101=true;
 stroke(70, 255, 0);
} else
```

```
{
 overRect101=false;
 stroke(0, 0, 0);
}
fill(red101, green101, blue101);
rect(xquad01, yquad101, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
 if (status101==true)
 {
  valor101=port.read();
  delay(250);
  textFont(f, 12);
  text(valor101, 105, 242);
 }
 if (status101==false)
 {
  textFont(f, 12);
  text(valor101, 105, 242);
 }
}
if (mouseX > xquad01 && mouseX < xquad01+bquad &&
 mouseY > yquad111 && mouseY < yquad111+aquad)
{
 overRect111=true;
 stroke(70, 255, 0);
} else
{
 overRect111=false;
 stroke(0, 0, 0);
}
fill(red111, green111, blue111);
rect(xquad01, yquad111, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
 if (status111==true)
 {
  valor111=port.read();
  delay(250);
  textFont(f, 12);
  text(valor111, 105, 257);
 }
 if (status111==false)
 {
  textFont(f, 12);
  text(valor111, 105, 257);
 }
```

```
}
if (mouseX > xquad01 && mouseX < xquad01+bquad &&
  mouseY > yquad121 && mouseY < yquad121+aquad)
{
  overRect121=true;
  stroke(70, 255, 0);
} else
{
  overRect121=false;
  stroke(0, 0, 0);
}
fill(red121, green121, blue121);
rect(xquad01, yquad121, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status121==true)
  {
    valor121=port.read();
    delay(250);
    textFont(f, 12);
    text(valor121, 105, 272);
  }
  if (status121==false)
  {
    textFont(f, 12);
    text(valor121, 105, 272);
  }
}
if (mouseX > xquad01 && mouseX < xquad01+bquad &&
  mouseY > yquad131 && mouseY < yquad131+aquad)
{
  overRect131=true;
  stroke(70, 255, 0);
} else
{
  overRect131=false;
  stroke(0, 0, 0);
}
fill(red131, green131, blue131);
rect(xquad01, yquad131, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status131==true)
  {
    valor131=port.read();
    delay(250);
    textFont(f, 12);
```

```
   text(valor131, 105, 287);
  }
  if (status131==false)
  {
   textFont(f, 12);
   text(valor131, 105, 287);
  }
}
if (mouseX > xquad01 && mouseX < xquad01+bquad &&
  mouseY > yquad141 && mouseY < yquad141+aquad)
{
  overRect141=true;
  stroke(70, 255, 0);
} else
{
  overRect141=false;
  stroke(0, 0, 0);
}
fill(red141, green141, blue141);
rect(xquad01, yquad141, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status141==true)
  {
   valor141=port.read();
   delay(250);
   textFont(f, 12);
   text(valor141, 105, 302);
  }
  if (status141==false)
  {
   textFont(f, 12);
   text(valor141, 105, 302);
  }
}
if (mouseX > xquad01 && mouseX < xquad01+bquad &&
  mouseY > yquad151 && mouseY < yquad151+aquad)
{
  overRect151=true;
  stroke(70, 255, 0);
} else
{
  overRect151=false;
  stroke(0, 0, 0);
}
fill(red151, green151, blue151);
rect(xquad01, yquad151, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
```

```
if (port.available()>0)
{
 if (status151==true)
 {
  valor151=port.read();
  delay(250);
  textFont(f, 12);
  text(valor151, 105, 317);
 }
 if (status151==false)
 {
  textFont(f, 12);
  text(valor151, 105, 317);
 }
}
if (mouseX > xquad01 && mouseX < xquad01+bquad &&
 mouseY > yquad161 && mouseY < yquad161+aquad)
{
 overRect161=true;
 stroke(70, 255, 0);
} else
{
 overRect161=false;
 stroke(0, 0, 0);
}
fill(red161, green161, blue161);
rect(xquad01, yquad161, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
 if (status161==true)
 {
  valor161=port.read();
  delay(250);
  textFont(f, 12);
  text(valor161, 105, 332);
 }
 if (status161==false)
 {
  textFont(f, 12);
  text(valor161, 105, 332);
 }
}
if (mouseX > xquad01 && mouseX < xquad01+bquad &&
 mouseY > yquad171 && mouseY < yquad171+aquad)
{
 overRect171=true;
 stroke(70, 255, 0);
} else
{
 overRect171=false;
```

```
  stroke(0, 0, 0);
}
fill(red171, green171, blue171);
rect(xquad01, yquad171, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status171==true)
  {
    valor171=port.read();
    delay(250);
    textFont(f, 12);
    text(valor171, 105, 347);
  }
  if (status171==false)
  {
    textFont(f, 12);
    text(valor171, 105, 347);
  }
}
if (mouseX > xquad01 && mouseX < xquad01+bquad &&
  mouseY > yquad181 && mouseY < yquad181+aquad)
{
  overRect181=true;
  stroke(70, 255, 0);
} else
{
  overRect181=false;
  stroke(0, 0, 0);
}
fill(red181, green181, blue181);
rect(xquad01, yquad181, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status181==true)
  {
    valor181=port.read();
    delay(250);
    textFont(f, 12);
    text(valor181, 105, 362);
  }
  if (status181==false)
  {
    textFont(f, 12);
    text(valor181, 105, 362);
  }
}
if (mouseX > xquad01 && mouseX < xquad01+bquad &&
```

```
 mouseY > yquad191 && mouseY < yquad191+aquad)
{
 overRect191=true;
 stroke(70, 255, 0);
} else
{
 overRect191=false;
 stroke(0, 0, 0);
}
fill(red191, green191, blue191);
rect(xquad01, yquad191, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
 if (status191==true)
 {
  valor191=port.read();
  delay(250);
  textFont(f, 12);
  text(valor191, 105, 377);
 }
 if (status191==false)
 {
  textFont(f, 12);
  text(valor191, 105, 377);
 }
}
if (mouseX > xquad01 && mouseX < xquad01+bquad &&
 mouseY > yquad201 && mouseY < yquad201+aquad)
{
 overRect201=true;
 stroke(70, 255, 0);
} else
{
 overRect201=false;
 stroke(0, 0, 0);
}
fill(red201, green201, blue201);
rect(xquad01, yquad201, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
 if (status201==true)
 {
  valor201=port.read();
  delay(250);
  textFont(f, 12);
  text(valor201, 105, 392);
 }
```

```
  if (status201==false)
  {
    textFont(f, 12);
    text(valor201, 105, 392);
  }
}
if (mouseX > xquad01 && mouseX < xquad01+bquad &&
  mouseY > yquad211 && mouseY < yquad211+aquad)
{
  overRect211=true;
  stroke(70, 255, 0);
} else
{
  overRect211=false;
  stroke(0, 0, 0);
}
fill(red211, green211, blue211);
rect(xquad01, yquad211, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status211==true)
  {
    valor211=port.read();
    delay(250);
    textFont(f, 12);
    text(valor211, 105, 407);
  }
  if (status211==false)
  {
    textFont(f, 12);
    text(valor211, 105, 407);
  }
}
if (mouseX > xquad01 && mouseX < xquad01+bquad &&
  mouseY > yquad221 && mouseY < yquad221+aquad)
{
  overRect221=true;
  stroke(70, 255, 0);
} else
{
  overRect221=false;
  stroke(0, 0, 0);
}
fill(red221, green221, blue221);
rect(xquad01, yquad221, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
```

```
  if (status221==true)
  {
    valor221=port.read();
    delay(250);
    textFont(f, 12);
    text(valor221, 105, 422);
  }
  if (status221==false)
  {
    textFont(f, 12);
    text(valor221, 105, 422);
  }
}
if (mouseX > xquad01 && mouseX < xquad01+bquad &&
  mouseY > yquad231 && mouseY < yquad231+aquad)
{
  overRect231=true;
  stroke(70, 255, 0);
} else
{
  overRect231=false;
  stroke(0, 0, 0);
}
fill(red231, green231, blue231);
rect(xquad01, yquad231, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status231==true)
  {
    valor231=port.read();
    delay(250);
    textFont(f, 12);
    text(valor231, 105, 437);
  }
  if (status231==false)
  {
    textFont(f, 12);
    text(valor231, 105, 437);
  }
}
if (mouseX > xquad01 && mouseX < xquad01+bquad &&
  mouseY > yquad241 && mouseY < yquad241+aquad)
{
  overRect241=true;
  stroke(70, 255, 0);
} else
{
  overRect241=false;
  stroke(0, 0, 0);
}
```

```
fill(red241, green241, blue241);
rect(xquad01, yquad241, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
 if (status241==true)
 {
  valor241=port.read();
  delay(250);
  textFont(f, 12);
  text(valor241, 105, 452);
 }
 if (status241==false)
 {
  textFont(f, 12);
  text(valor241, 105, 452);
 }
}
if (mouseX > xquad01 && mouseX < xquad01+bquad &&
 mouseY > yquad251 && mouseY < yquad251+aquad)
{
 overRect251=true;
 stroke(70, 255, 0);
} else
{
 overRect251=false;
 stroke(0, 0, 0);
}
fill(red251, green251, blue251);
rect(xquad01, yquad251, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
 if (status251==true)
 {
  valor251=port.read();
  delay(250);
  textFont(f, 12);
  text(valor251, 105, 467);
 }
 if (status251==false)
 {
  textFont(f, 12);
  text(valor251, 105, 467);
 }
}
if (mouseX > xquad01 && mouseX < xquad01+bquad &&
 mouseY > yquad261 && mouseY < yquad261+aquad)
{
```

```
  overRect261=true;
  stroke(70, 255, 0);
} else
{
  overRect261=false;
  stroke(0, 0, 0);
}
fill(red261, green261, blue261);
rect(xquad01, yquad261, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status261==true)
  {
    valor261=port.read();
    delay(250);
    textFont(f, 12);
    text(valor261, 105, 482);
  }
  if (status261==false)
  {
    textFont(f, 12);
    text(valor261, 105, 482);
  }
}
if (mouseX > xquad01 && mouseX < xquad01+bquad &&
  mouseY > yquad271 && mouseY < yquad271+aquad)
{
  overRect271=true;
  stroke(70, 255, 0);
} else
{
  overRect271=false;
  stroke(0, 0, 0);
}
fill(red271, green271, blue271);
rect(xquad01, yquad271, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status271==true)
  {
    valor271=port.read();
    delay(250);
    textFont(f, 12);
    text(valor271, 105, 497);
  }
  if (status271==false)
  {
```

```
    textFont(f, 12);
    text(valor271, 105, 497);
  }
}
if (mouseX > xquad01 && mouseX < xquad01+bquad &&
  mouseY > yquad281 && mouseY < yquad281+aquad)
{
  overRect281=true;
  stroke(70, 255, 0);
} else
{
  overRect281=false;
  stroke(0, 0, 0);
}
fill(red281, green281, blue281);
rect(xquad01, yquad281, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status281==true)
  {
    valor281=port.read();
    delay(250);
    textFont(f, 12);
    text(valor281, 105, 512);
  }
  if (status281==false)
  {
    textFont(f, 12);
    text(valor281, 105, 512);
  }
}
if (mouseX > xquad01 && mouseX < xquad01+bquad &&
  mouseY > yquad291 && mouseY < yquad291+aquad)
{
  overRect291=true;
  stroke(70, 255, 0);
} else
{
  overRect291=false;
  stroke(0, 0, 0);
}
fill(red291, green291, blue291);
rect(xquad01, yquad291, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status291==true)
  {
```

```
    valor291=port.read();
    delay(250);
    textFont(f, 12);
    text(valor291, 105, 527);
  }
  if (status291==false)
  {
    textFont(f, 12);
    text(valor291, 105, 527);
  }
}
if (mouseX > xquad01 && mouseX < xquad01+bquad &&
  mouseY > yquad301 && mouseY < yquad301+aquad)
{
  overRect301=true;
  stroke(70, 255, 0);
} else
{
  overRect301=false;
  stroke(0, 0, 0);
}
fill(red301, green301, blue301);
rect(xquad01, yquad301, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status301==true)
  {
    valor301=port.read();
    delay(250);
    textFont(f, 12);
    text(valor301, 105, 542);
  }
  if (status301==false)
  {
    textFont(f, 12);
    text(valor301, 105, 542);
  }
}
if (mouseX > xquad01 && mouseX < xquad01+bquad &&
  mouseY > yquad311 && mouseY < yquad311+aquad)
{
  overRect311=true;
  stroke(70, 255, 0);
} else
{
  overRect311=false;
  stroke(0, 0, 0);
}
fill(red311, green311, blue311);
rect(xquad01, yquad311, bquad, aquad);
```

```
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
 if (status311==true)
 {
  valor311=port.read();
  delay(250);
  textFont(f, 12);
  text(valor311, 105, 557);
 }
 if (status311==false)
 {
  textFont(f, 12);
  text(valor311, 105, 557);
 }
}
////// COLUMNA G STRUT 2 ///////
if (mouseX > xquad02 && mouseX < xquad02+bquad &&
 mouseY > yquad02 && mouseY < yquad02+aquad)
{
 overRect02=true;
 stroke(70, 255, 0);
} else
{
 overRect02=false;
 stroke(0, 0, 0);
}
fill(red02, green02, blue02);
rect(xquad02, yquad02, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
 if (status02==true)
 {
  valor02=port.read();
  delay(250);
  textFont(f, 12);
  text(valor02, 225, 92);
 }
 if (status02==false)
 {
  textFont(f, 12);
  text(valor02, 225, 92);
 }
}
if (mouseX > xquad02 && mouseX < xquad02+bquad &&
 mouseY > yquad12 && mouseY < yquad12+aquad)
{
 overRect12=true;
```

```
  stroke(70, 255, 0);
} else
{
  overRect12=false;
  stroke(0, 0, 0);
}
fill(red12, green12, blue12);
rect(xquad02, yquad12, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status12==true)
  {
    valor12=port.read();
    delay(250);
    textFont(f, 12);
    text(valor12, 225, 107);
  }
  if (status12==false)
  {
    textFont(f, 12);
    text(valor12, 225, 107);
  }
}
if (mouseX > xquad02 && mouseX < xquad02+bquad &&
  mouseY > yquad22 && mouseY < yquad22+aquad)
{
  overRect22=true;
  stroke(70, 255, 0);
} else
{
  overRect22=false;
  stroke(0, 0, 0);
}
fill(red22, green22, blue22);
rect(xquad02, yquad22, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status22==true)
  {
    valor22=port.read();
    delay(250);
    textFont(f, 12);
    text(valor22, 225, 122);
  }
  if (status22==false)
  {
    textFont(f, 12);
```

```
   text(valor22, 225, 122);
  }
}
if (mouseX > xquad02 && mouseX < xquad02+bquad &&
  mouseY > yquad32 && mouseY < yquad32+aquad)
{
  overRect32=true;
  stroke(70, 255, 0);
} else
{
  overRect32=false;
  stroke(0, 0, 0);
}
fill(red32, green32, blue32);
rect(xquad02, yquad32, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status32==true)
  {
    valor32=port.read();
    delay(250);
    textFont(f, 12);
    text(valor32, 225, 137);
  }
  if (status32==false)
  {
    textFont(f, 12);
    text(valor32, 225, 137);
  }
}
if (mouseX > xquad02 && mouseX < xquad02+bquad &&
  mouseY > yquad42 && mouseY < yquad42+aquad)
{
  overRect42=true;
  stroke(70, 255, 0);
} else
{
  overRect42=false;
  stroke(0, 0, 0);
}
fill(red42, green42, blue42);
rect(xquad02, yquad42, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status42==true)
  {
    valor42=port.read();
```

```
    delay(250);
    textFont(f, 12);
    text(valor42, 225, 152);
   }
  if (status42==false)
  {
    textFont(f, 12);
    text(valor42, 225, 152);
  }
}
if (mouseX > xquad02 && mouseX < xquad02+bquad &&
  mouseY > yquad52 && mouseY < yquad52+aquad)
{
  overRect52=true;
  stroke(70, 255, 0);
} else
{
  overRect52=false;
  stroke(0, 0, 0);
}
fill(red52, green52, blue52);
rect(xquad02, yquad52, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status52==true)
  {
    valor52=port.read();
    delay(250);
    textFont(f, 12);
    text(valor52, 225, 167);
  }
  if (status52==false)
  {
    textFont(f, 12);
    text(valor52, 225, 167);
  }
}
if (mouseX > xquad02 && mouseX < xquad02+bquad &&
  mouseY > yquad62 && mouseY < yquad62+aquad)
{
  overRect62=true;
  stroke(70, 255, 0);
} else
{
  overRect62=false;
  stroke(0, 0, 0);
}
fill(red62, green62, blue62);
rect(xquad02, yquad62, bquad, aquad);
fill(0, 0, 0);
```

```
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
 if (status62==true)
 {
  valor62=port.read();
  delay(250);
  textFont(f, 12);
  text(valor62, 225, 182);
 }
 if (status62==false)
 {
  textFont(f, 12);
  text(valor62, 225, 182);
 }
}
if (mouseX > xquad02 && mouseX < xquad02+bquad &&
 mouseY > yquad72 && mouseY < yquad72+aquad)
{
 overRect72=true;
 stroke(70, 255, 0);
} else
{
 overRect72=false;
 stroke(0, 0, 0);
}
fill(red72, green72, blue72);
rect(xquad02, yquad72, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
 if (status72==true)
 {
  valor72=port.read();
  delay(250);
  textFont(f, 12);
  text(valor72, 225, 197);
 }
 if (status72==false)
 {
  textFont(f, 12);
  text(valor72, 225, 197);
 }
}
if (mouseX > xquad02 && mouseX < xquad02+bquad &&
 mouseY > yquad82 && mouseY < yquad82+aquad)
{
 overRect82=true;
 stroke(70, 255, 0);
} else
```

```
{
 overRect82=false;
 stroke(0, 0, 0);
}
fill(red82, green82, blue82);
rect(xquad02, yquad82, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
 if (status82==true)
 {
  valor82=port.read();
  delay(250);
  textFont(f, 12);
  text(valor82, 225, 212);
 }
 if (status82==false)
 {
  textFont(f, 12);
  text(valor82, 225, 212);
 }
}
if (mouseX > xquad02 && mouseX < xquad02+bquad &&
 mouseY > yquad92 && mouseY < yquad92+aquad)
{
 overRect92=true;
 stroke(70, 255, 0);
} else
{
 overRect92=false;
 stroke(0, 0, 0);
}
fill(red92, green92, blue92);
rect(xquad02, yquad92, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
 if (status92==true)
 {
  valor92=port.read();
  delay(250);
  textFont(f, 12);
  text(valor92, 225, 227);
 }
 if (status92==false)
 {
  textFont(f, 12);
  text(valor92, 225, 227);
 }
```

```
}
if (mouseX > xquad02 && mouseX < xquad02+bquad &&
  mouseY > yquad102 && mouseY < yquad102+aquad)
{
  overRect102=true;
  stroke(70, 255, 0);
} else
{
  overRect102=false;
  stroke(0, 0, 0);
}
fill(red102, green102, blue102);
rect(xquad02, yquad102, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status102==true)
  {
    valor102=port.read();
    delay(250);
    textFont(f, 12);
    text(valor102, 225, 242);
  }
  if (status102==false)
  {
    textFont(f, 12);
    text(valor102, 225, 242);
  }
}
if (mouseX > xquad02 && mouseX < xquad02+bquad &&
  mouseY > yquad112 && mouseY < yquad112+aquad)
{
  overRect112=true;
  stroke(70, 255, 0);
} else
{
  overRect112=false;
  stroke(0, 0, 0);
}
fill(red112, green112, blue112);
rect(xquad02, yquad112, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status112==true)
  {
    valor112=port.read();
    delay(250);
    textFont(f, 12);
```

```
   text(valor112, 225, 257);
  }
  if (status112==false)
  {
   textFont(f, 12);
   text(valor112, 225, 257);
  }
}
if (mouseX > xquad02 && mouseX < xquad02+bquad &&
  mouseY > yquad122 && mouseY < yquad122+aquad)
{
  overRect122=true;
  stroke(70, 255, 0);
} else
{
  overRect122=false;
  stroke(0, 0, 0);
}
fill(red122, green122, blue122);
rect(xquad02, yquad122, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status122==true)
  {
   valor122=port.read();
   delay(250);
   textFont(f, 12);
   text(valor122, 225, 272);
  }
  if (status122==false)
  {
   textFont(f, 12);
   text(valor122, 225, 272);
  }
}
if (mouseX > xquad02 && mouseX < xquad02+bquad &&
  mouseY > yquad132 && mouseY < yquad132+aquad)
{
  overRect132=true;
  stroke(70, 255, 0);
} else
{
  overRect132=false;
  stroke(0, 0, 0);
}
fill(red132, green132, blue132);
rect(xquad02, yquad132, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
```

```
if (port.available()>0)
{
 if (status132==true)
 {
  valor132=port.read();
  delay(250);
  textFont(f, 12);
  text(valor132, 225, 287);
 }
 if (status132==false)
 {
  textFont(f, 12);
  text(valor132, 225, 287);
 }
}
if (mouseX > xquad02 && mouseX < xquad02+bquad &&
 mouseY > yquad142 && mouseY < yquad142+aquad)
{
 overRect142=true;
 stroke(70, 255, 0);
} else
{
 overRect142=false;
 stroke(0, 0, 0);
}
fill(red142, green142, blue132);
rect(xquad02, yquad142, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
 if (status142==true)
 {
  valor142=port.read();
  delay(250);
  textFont(f, 12);
  text(valor142, 225, 302);
 }
 if (status142==false)
 {
  textFont(f, 12);
  text(valor142, 225, 302);
 }
}
if (mouseX > xquad02 && mouseX < xquad02+bquad &&
 mouseY > yquad152 && mouseY < yquad152+aquad)
{
 overRect152=true;
 stroke(70, 255, 0);
} else
{
 overRect152=false;
```

```
  stroke(0, 0, 0);
}
fill(red152, green152, blue152);
rect(xquad02, yquad152, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status152==true)
  {
    valor152=port.read();
    delay(250);
    textFont(f, 12);
    text(valor152, 225, 317);
  }
  if (status152==false)
  {
    textFont(f, 12);
    text(valor152, 225, 317);
  }
}
if (mouseX > xquad02 && mouseX < xquad02+bquad &&
  mouseY > yquad162 && mouseY < yquad162+aquad)
{
  overRect162=true;
  stroke(70, 255, 0);
} else
{
  overRect162=false;
  stroke(0, 0, 0);
}
fill(red162, green162, blue162);
rect(xquad02, yquad162, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status162==true)
  {
    valor162=port.read();
    delay(250);
    textFont(f, 12);
    text(valor162, 225, 332);
  }
  if (status162==false)
  {
    textFont(f, 12);
    text(valor162, 225, 332);
  }
}
if (mouseX > xquad02 && mouseX < xquad02+bquad &&
```

```
  mouseY > yquad172 && mouseY < yquad172+aquad)
{
  overRect172=true;
  stroke(70, 255, 0);
} else
{
  overRect172=false;
  stroke(0, 0, 0);
}
fill(red172, green172, blue172);
rect(xquad02, yquad172, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status172==true)
  {
    valor172=port.read();
    delay(250);
    textFont(f, 12);
    text(valor172, 225, 347);
  }
  if (status172==false)
  {
    textFont(f, 12);
    text(valor172, 225, 347);
  }
}
if (mouseX > xquad02 && mouseX < xquad02+bquad &&
  mouseY > yquad182 && mouseY < yquad182+aquad)
{
  overRect182=true;
  stroke(70, 255, 0);
} else
{
  overRect182=false;
  stroke(0, 0, 0);
}
fill(red182, green182, blue182);
rect(xquad02, yquad182, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status182==true)
  {
    valor182=port.read();
    delay(250);
    textFont(f, 12);
    text(valor182, 225, 362);
  }
```

```
  if (status182==false)
  {
    textFont(f, 12);
    text(valor182, 225, 362);
  }
}
if (mouseX > xquad02 && mouseX < xquad02+bquad &&
  mouseY > yquad192 && mouseY < yquad192+aquad)
{
  overRect192=true;
  stroke(70, 255, 0);
} else
{
  overRect192=false;
  stroke(0, 0, 0);
}
fill(red192, green192, blue192);
rect(xquad02, yquad192, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status192==true)
  {
    valor192=port.read();
    delay(250);
    textFont(f, 12);
    text(valor192, 225, 377);
  }
  if (status192==false)
  {
    textFont(f, 12);
    text(valor192, 225, 377);
  }
}
if (mouseX > xquad02 && mouseX < xquad02+bquad &&
  mouseY > yquad202 && mouseY < yquad202+aquad)
{
  overRect202=true;
  stroke(70, 255, 0);
} else
{
  overRect202=false;
  stroke(0, 0, 0);
}
fill(red202, green202, blue202);
rect(xquad02, yquad202, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
```

```
  if (status202==true)
  {
    valor202=port.read();
    delay(250);
    textFont(f, 12);
    text(valor202, 225, 392);
  }
  if (status202==false)
  {
    textFont(f, 12);
    text(valor202, 225, 392);
  }
}
if (mouseX > xquad02 && mouseX < xquad02+bquad &&
  mouseY > yquad212 && mouseY < yquad212+aquad)
{
  overRect212=true;
  stroke(70, 255, 0);
} else
{
  overRect212=false;
  stroke(0, 0, 0);
}
fill(red212, green212, blue212);
rect(xquad02, yquad212, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status212==true)
  {
    valor212=port.read();
    delay(250);
    textFont(f, 12);
    text(valor212, 225, 407);
  }
  if (status212==false)
  {
    textFont(f, 12);
    text(valor212, 225, 407);
  }
}
if (mouseX > xquad02 && mouseX < xquad02+bquad &&
  mouseY > yquad222 && mouseY < yquad222+aquad)
{
  overRect222=true;
  stroke(70, 255, 0);
} else
{
  overRect222=false;
  stroke(0, 0, 0);
}
```

```
fill(red222, green222, blue222);
rect(xquad02, yquad222, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
 if (status222==true)
 {
  valor222=port.read();
  delay(250);
  textFont(f, 12);
  text(valor222, 225, 422);
 }
 if (status222==false)
 {
  textFont(f, 12);
  text(valor222, 225, 422);
 }
}
if (mouseX > xquad02 && mouseX < xquad02+bquad &&
 mouseY > yquad232 && mouseY < yquad232+aquad)
{
 overRect232=true;
 stroke(70, 255, 0);
} else
{
 overRect232=false;
 stroke(0, 0, 0);
}
fill(red232, green232, blue232);
rect(xquad02, yquad232, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
 if (status232==true)
 {
  valor232=port.read();
  delay(250);
  textFont(f, 12);
  text(valor232, 225, 437);
 }
 if (status232==false)
 {
  textFont(f, 12);
  text(valor232, 225, 437);
 }
}
if (mouseX > xquad02 && mouseX < xquad02+bquad &&
 mouseY > yquad242 && mouseY < yquad242+aquad)
{
```

```
  overRect242=true;
  stroke(70, 255, 0);
} else
{
  overRect242=false;
  stroke(0, 0, 0);
}
fill(red242, green242, blue242);
rect(xquad02, yquad242, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status242==true)
  {
    valor242=port.read();
    delay(250);
    textFont(f, 12);
    text(valor242, 225, 452);
  }
  if (status242==false)
  {
    textFont(f, 12);
    text(valor242, 225, 452);
  }
}
if (mouseX > xquad02 && mouseX < xquad02+bquad &&
  mouseY > yquad252 && mouseY < yquad252+aquad)
{
  overRect252=true;
  stroke(70, 255, 0);
} else
{
  overRect252=false;
  stroke(0, 0, 0);
}
fill(red252, green252, blue252);
rect(xquad02, yquad252, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status252==true)
  {
    valor252=port.read();
    delay(250);
    textFont(f, 12);
    text(valor252, 225, 467);
  }
  if (status252==false)
  {
```

```
    textFont(f, 12);
    text(valor252, 225, 467);
  }
}
if (mouseX > xquad02 && mouseX < xquad02+bquad &&
  mouseY > yquad262 && mouseY < yquad262+aquad)
{
  overRect262=true;
  stroke(70, 255, 0);
} else
{
  overRect262=false;
  stroke(0, 0, 0);
}
fill(red262, green262, blue262);
rect(xquad02, yquad262, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status262==true)
  {
    valor262=port.read();
    delay(250);
    textFont(f, 12);
    text(valor262, 225, 482);
  }
  if (status262==false)
  {
    textFont(f, 12);
    text(valor262, 225, 482);
  }
}
if (mouseX > xquad02 && mouseX < xquad02+bquad &&
  mouseY > yquad272 && mouseY < yquad272+aquad)
{
  overRect272=true;
  stroke(70, 255, 0);
} else
{
  overRect272=false;
  stroke(0, 0, 0);
}
fill(red272, green272, blue272);
rect(xquad02, yquad272, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status272==true)
  {
```

```
    valor272=port.read();
    delay(250);
    textFont(f, 12);
    text(valor272, 225, 497);
  }
  if (status272==false)
  {
    textFont(f, 12);
    text(valor272, 225, 497);
  }
}
if (mouseX > xquad02 && mouseX < xquad02+bquad &&
  mouseY > yquad282 && mouseY < yquad282+aquad)
{
  overRect282=true;
  stroke(70, 255, 0);
} else
{
  overRect282=false;
  stroke(0, 0, 0);
}
fill(red282, green282, blue282);
rect(xquad02, yquad282, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status282==true)
  {
    valor282=port.read();
    delay(250);
    textFont(f, 12);
    text(valor282, 225, 512);
  }
  if (status282==false)
  {
    textFont(f, 12);
    text(valor282, 225, 512);
  }
}
if (mouseX > xquad02 && mouseX < xquad02+bquad &&
  mouseY > yquad292 && mouseY < yquad292+aquad)
{
  overRect292=true;
  stroke(70, 255, 0);
} else
{
  overRect292=false;
  stroke(0, 0, 0);
}
fill(red292, green292, blue292);
rect(xquad02, yquad292, bquad, aquad);
```

```
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
 if (status292==true)
 {
  valor292=port.read();
  delay(250);
  textFont(f, 12);
  text(valor292, 225, 527);
 }
 if (status292==false)
 {
  textFont(f, 12);
  text(valor292, 225, 527);
 }
}
if (mouseX > xquad02 && mouseX < xquad02+bquad &&
 mouseY > yquad302 && mouseY < yquad302+aquad)
{
 overRect302=true;
 stroke(70, 255, 0);
} else
{
 overRect302=false;
 stroke(0, 0, 0);
}
fill(red302, green302, blue302);
rect(xquad02, yquad302, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
 if (status302==true)
 {
  valor302=port.read();
  delay(250);
  textFont(f, 12);
  text(valor302, 225, 542);
 }
 if (status302==false)
 {
  textFont(f, 12);
  text(valor302, 225, 542);
 }
}
if (mouseX > xquad02 && mouseX < xquad02+bquad &&
 mouseY > yquad312 && mouseY < yquad312+aquad)
{
 overRect312=true;
 stroke(70, 255, 0);
```

```
} else
{
  overRect312=false;
  stroke(0, 0, 0);
}
fill(red312, green312, blue312);
rect(xquad02, yquad312, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status312==true)
  {
    valor312=port.read();
    delay(250);
    textFont(f, 12);
    text(valor312, 225, 557);
  }
  if (status312==false)
  {
    textFont(f, 12);
    text(valor312, 225, 557);
  }
}
////// COLUMNA G STRUT 3 ///////
if (mouseX > xquad03 && mouseX < xquad03+bquad &&
  mouseY > yquad03 && mouseY < yquad03+aquad)
{
  overRect03=true;
  stroke(70, 255, 0);
} else
{
  overRect03=false;
  stroke(0, 0, 0);
}
fill(red03, green03, blue03);
rect(xquad03, yquad03, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status03==true)
  {
    valor03=port.read();
    delay(250);
    textFont(f, 12);
    text(valor03, 345, 92);
  }
  if (status03==false)
  {
    textFont(f, 12);
```

```
   text(valor03, 345, 92);
  }
}
if (mouseX > xquad03 && mouseX < xquad03+bquad &&
  mouseY > yquad13 && mouseY < yquad13+aquad)
{
  overRect13=true;
  stroke(70, 255, 0);
} else
{
  overRect13=false;
  stroke(0, 0, 0);
}
fill(red13, green13, blue13);
rect(xquad03, yquad13, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status13==true)
  {
    valor13=port.read();
    delay(250);
    textFont(f, 12);
    text(valor13, 345, 107);
  }
  if (status13==false)
  {
    textFont(f, 12);
    text(valor13, 345, 107);
  }
}
if (mouseX > xquad03 && mouseX < xquad03+bquad &&
  mouseY > yquad23 && mouseY < yquad23+aquad)
{
  overRect23=true;
  stroke(70, 255, 0);
} else
{
  overRect23=false;
  stroke(0, 0, 0);
}
fill(red23, green23, blue23);
rect(xquad03, yquad23, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status23==true)
  {
    valor23=port.read();
```

```
   delay(250);
   textFont(f, 12);
   text(valor23, 345, 122);
  }
  if (status23==false)
  {
   textFont(f, 12);
   text(valor23, 345, 122);
  }
}
if (mouseX > xquad03 && mouseX < xquad03+bquad &&
  mouseY > yquad33 && mouseY < yquad33+aquad)
{
  overRect33=true;
  stroke(70, 255, 0);
} else
{
  overRect33=false;
  stroke(0, 0, 0);
}
fill(red33, green33, blue33);
rect(xquad03, yquad33, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status33==true)
  {
   valor33=port.read();
   delay(250);
   textFont(f, 12);
   text(valor33, 345, 137);
  }
  if (status33==false)
  {
   textFont(f, 12);
   text(valor33, 345, 137);
  }
}
if (mouseX > xquad03 && mouseX < xquad03+bquad &&
  mouseY > yquad43 && mouseY < yquad43+aquad)
{
  overRect43=true;
  stroke(70, 255, 0);
} else
{
  overRect43=false;
  stroke(0, 0, 0);
}
fill(red43, green43, blue43);
rect(xquad03, yquad43, bquad, aquad);
fill(0, 0, 0);
```

```
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
 if (status43==true)
 {
  valor43=port.read();
  delay(250);
  textFont(f, 12);
  text(valor43, 345, 152);
 }
 if (status43==false)
 {
  textFont(f, 12);
  text(valor43, 345, 152);
 }
}
if (mouseX > xquad03 && mouseX < xquad03+bquad &&
 mouseY > yquad53 && mouseY < yquad53+aquad)
{
 overRect53=true;
 stroke(70, 255, 0);
} else
{
 overRect53=false;
 stroke(0, 0, 0);
}
fill(red53, green53, blue53);
rect(xquad03, yquad53, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
 if (status53==true)
 {
  valor53=port.read();
  delay(250);
  textFont(f, 12);
  text(valor53, 345, 167);
 }
 if (status53==false)
 {
  textFont(f, 12);
  text(valor53, 345, 167);
 }
}
if (mouseX > xquad03 && mouseX < xquad03+bquad &&
 mouseY > yquad63 && mouseY < yquad63+aquad)
{
 overRect63=true;
 stroke(70, 255, 0);
} else
```

```
{
 overRect63=false;
 stroke(0, 0, 0);
}
fill(red63, green63, blue63);
rect(xquad03, yquad63, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
 if (status63==true)
 {
  valor63=port.read();
  delay(250);
  textFont(f, 12);
  text(valor63, 345, 182);
 }
 if (status63==false)
 {
  textFont(f, 12);
  text(valor63, 345, 182);
 }
}
if (mouseX > xquad03 && mouseX < xquad03+bquad &&
 mouseY > yquad73 && mouseY < yquad73+aquad)
{
 overRect73=true;
 stroke(70, 255, 0);
} else
{
 overRect73=false;
 stroke(0, 0, 0);
}
fill(red73, green73, blue73);
rect(xquad03, yquad73, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
 if (status73==true)
 {
  valor73=port.read();
  delay(250);
  textFont(f, 12);
  text(valor73, 345, 197);
 }
 if (status73==false)
 {
  textFont(f, 12);
  text(valor73, 345, 197);
 }
```

```
}
if (mouseX > xquad03 && mouseX < xquad03+bquad &&
  mouseY > yquad83 && mouseY < yquad83+aquad)
{
  overRect83=true;
  stroke(70, 255, 0);
} else
{
  overRect83=false;
  stroke(0, 0, 0);
}
fill(red83, green83, blue83);
rect(xquad03, yquad83, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status83==true)
  {
    valor83=port.read();
    delay(250);
    textFont(f, 12);
    text(valor83, 345, 212);
  }
  if (status83==false)
  {
    textFont(f, 12);
    text(valor83, 345, 212);
  }
}
if (mouseX > xquad03 && mouseX < xquad03+bquad &&
  mouseY > yquad93 && mouseY < yquad93+aquad)
{
  overRect93=true;
  stroke(70, 255, 0);
} else
{
  overRect93=false;
  stroke(0, 0, 0);
}
fill(red93, green93, blue93);
rect(xquad03, yquad93, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status93==true)
  {
    valor93=port.read();
    delay(250);
    textFont(f, 12);
```

```
   text(valor93, 345, 227);
  }
  if (status93==false)
  {
   textFont(f, 12);
   text(valor93, 345, 227);
  }
}
if (mouseX > xquad03 && mouseX < xquad03+bquad &&
  mouseY > yquad103 && mouseY < yquad103+aquad)
{
  overRect103=true;
  stroke(70, 255, 0);
} else
{
  overRect103=false;
  stroke(0, 0, 0);
}
fill(red103, green103, blue103);
rect(xquad03, yquad103, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status103==true)
  {
   valor103=port.read();
   delay(250);
   textFont(f, 12);
   text(valor103, 345, 242);
  }
  if (status103==false)
  {
   textFont(f, 12);
   text(valor103, 345, 242);
  }
}
if (mouseX > xquad03 && mouseX < xquad03+bquad &&
  mouseY > yquad113 && mouseY < yquad113+aquad)
{
  overRect113=true;
  stroke(70, 255, 0);
} else
{
  overRect113=false;
  stroke(0, 0, 0);
}
fill(red113, green113, blue113);
rect(xquad03, yquad113, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
```

```
if (port.available()>0)
{
 if (status113==true)
 {
  valor113=port.read();
  delay(250);
  textFont(f, 12);
  text(valor113, 345, 257);
 }
 if (status113==false)
 {
  textFont(f, 12);
  text(valor113, 345, 257);
 }
}
if (mouseX > xquad03 && mouseX < xquad03+bquad &&
 mouseY > yquad123 && mouseY < yquad123+aquad)
{
 overRect123=true;
 stroke(70, 255, 0);
} else
{
 overRect123=false;
 stroke(0, 0, 0);
}
fill(red123, green123, blue123);
rect(xquad03, yquad123, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
 if (status123==true)
 {
  valor123=port.read();
  delay(250);
  textFont(f, 12);
  text(valor123, 345, 272);
 }
 if (status123==false)
 {
  textFont(f, 12);
  text(valor123, 345, 272);
 }
}
if (mouseX > xquad03 && mouseX < xquad03+bquad &&
 mouseY > yquad133 && mouseY < yquad133+aquad)
{
 overRect133=true;
 stroke(70, 255, 0);
} else
{
 overRect133=false;
```

```
  stroke(0, 0, 0);
}
fill(red133, green133, blue133);
rect(xquad03, yquad133, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status133==true)
  {
    valor133=port.read();
    delay(250);
    textFont(f, 12);
    text(valor133, 345, 287);
  }
  if (status133==false)
  {
    textFont(f, 12);
    text(valor133, 345, 287);
  }
}
if (mouseX > xquad03 && mouseX < xquad03+bquad &&
  mouseY > yquad143 && mouseY < yquad143+aquad)
{
  overRect143=true;
  stroke(70, 255, 0);
} else
{
  overRect143=false;
  stroke(0, 0, 0);
}
fill(red143, green143, blue143);
rect(xquad03, yquad143, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status143==true)
  {
    valor143=port.read();
    delay(250);
    textFont(f, 12);
    text(valor143, 345, 302);
  }
  if (status143==false)
  {
    textFont(f, 12);
    text(valor143, 345, 302);
  }
}
if (mouseX > xquad03 && mouseX < xquad03+bquad &&
```

```
  mouseY > yquad153 && mouseY < yquad153+aquad)
{
  overRect153=true;
  stroke(70, 255, 0);
} else
{
  overRect153=false;
  stroke(0, 0, 0);
}
fill(red153, green153, blue153);
rect(xquad03, yquad153, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status153==true)
  {
    valor153=port.read();
    delay(250);
    textFont(f, 12);
    text(valor153, 345, 317);
  }
  if (status153==false)
  {
    textFont(f, 12);
    text(valor153, 345, 317);
  }
}
if (mouseX > xquad03 && mouseX < xquad03+bquad &&
  mouseY > yquad163 && mouseY < yquad163+aquad)
{
  overRect163=true;
  stroke(70, 255, 0);
} else
{
  overRect163=false;
  stroke(0, 0, 0);
}
fill(red163, green163, blue163);
rect(xquad03, yquad163, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status163==true)
  {
    valor163=port.read();
    delay(250);
    textFont(f, 12);
    text(valor163, 345, 332);
  }
```

```
  if (status163==false)
  {
    textFont(f, 12);
    text(valor163, 345, 332);
  }
}
if (mouseX > xquad03 && mouseX < xquad03+bquad &&
  mouseY > yquad173 && mouseY < yquad173+aquad)
{
  overRect173=true;
  stroke(70, 255, 0);
} else
{
  overRect173=false;
  stroke(0, 0, 0);
}
fill(red173, green173, blue173);
rect(xquad03, yquad173, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status173==true)
  {
    valor173=port.read();
    delay(250);
    textFont(f, 12);
    text(valor173, 345, 347);
  }
  if (status173==false)
  {
    textFont(f, 12);
    text(valor173, 345, 347);
  }
}
if (mouseX > xquad03 && mouseX < xquad03+bquad &&
  mouseY > yquad183 && mouseY < yquad183+aquad)
{
  overRect183=true;
  stroke(70, 255, 0);
} else
{
  overRect183=false;
  stroke(0, 0, 0);
}
fill(red183, green183, blue183);
rect(xquad03, yquad183, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
```

```
  if (status183==true)
  {
    valor183=port.read();
    delay(250);
    textFont(f, 12);
    text(valor183, 345, 362);
  }
  if (status183==false)
  {
    textFont(f, 12);
    text(valor183, 345, 362);
  }
}
if (mouseX > xquad03 && mouseX < xquad03+bquad &&
  mouseY > yquad193 && mouseY < yquad193+aquad)
{
  overRect193=true;
  stroke(70, 255, 0);
} else
{
  overRect193=false;
  stroke(0, 0, 0);
}
fill(red193, green193, blue193);
rect(xquad03, yquad193, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status193==true)
  {
    valor193=port.read();
    delay(250);
    textFont(f, 12);
    text(valor193, 345, 377);
  }
  if (status193==false)
  {
    textFont(f, 12);
    text(valor193, 345, 377);
  }
}
if (mouseX > xquad03 && mouseX < xquad03+bquad &&
  mouseY > yquad203 && mouseY < yquad203+aquad)
{
  overRect203=true;
  stroke(70, 255, 0);
} else
{
  overRect203=false;
  stroke(0, 0, 0);
}
```

```
fill(red203, green203, blue203);
rect(xquad03, yquad203, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
 if (status203==true)
 {
  valor203=port.read();
  delay(250);
  textFont(f, 12);
  text(valor203, 345, 392);
 }
 if (status203==false)
 {
  textFont(f, 12);
  text(valor203, 345, 392);
 }
}
if (mouseX > xquad03 && mouseX < xquad03+bquad &&
 mouseY > yquad213 && mouseY < yquad213+aquad)
{
 overRect213=true;
 stroke(70, 255, 0);
} else
{
 overRect213=false;
 stroke(0, 0, 0);
}
fill(red213, green213, blue213);
rect(xquad03, yquad213, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
 if (status213==true)
 {
  valor213=port.read();
  delay(250);
  textFont(f, 12);
  text(valor213, 345, 407);
 }
 if (status213==false)
 {
  textFont(f, 12);
  text(valor213, 345, 407);
 }
}
if (mouseX > xquad03 && mouseX < xquad03+bquad &&
 mouseY > yquad223 && mouseY < yquad223+aquad)
{
```

```
  overRect223=true;
  stroke(70, 255, 0);
} else
{
  overRect223=false;
  stroke(0, 0, 0);
}
fill(red223, green223, blue223);
rect(xquad03, yquad223, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status223==true)
  {
    valor223=port.read();
    delay(250);
    textFont(f, 12);
    text(valor223, 345, 422);
  }
  if (status223==false)
  {
    textFont(f, 12);
    text(valor223, 345, 422);
  }
}
if (mouseX > xquad03 && mouseX < xquad03+bquad &&
  mouseY > yquad233 && mouseY < yquad233+aquad)
{
  overRect233=true;
  stroke(70, 255, 0);
} else
{
  overRect233=false;
  stroke(0, 0, 0);
}
fill(red233, green233, blue233);
rect(xquad03, yquad233, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status233==true)
  {
    valor233=port.read();
    delay(250);
    textFont(f, 12);
    text(valor233, 345, 437);
  }
  if (status233==false)
  {
```

```
    textFont(f, 12);
    text(valor233, 345, 437);
  }
}
if (mouseX > xquad03 && mouseX < xquad03+bquad &&
  mouseY > yquad243 && mouseY < yquad243+aquad)
{
  overRect243=true;
  stroke(70, 255, 0);
} else
{
  overRect243=false;
  stroke(0, 0, 0);
}
fill(red243, green243, blue243);
rect(xquad03, yquad243, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status243==true)
  {
    valor243=port.read();
    delay(250);
    textFont(f, 12);
    text(valor243, 345, 452);
  }
  if (status243==false)
  {
    textFont(f, 12);
    text(valor243, 345, 452);
  }
}
if (mouseX > xquad03 && mouseX < xquad03+bquad &&
  mouseY > yquad253 && mouseY < yquad253+aquad)
{
  overRect253=true;
  stroke(70, 255, 0);
} else
{
  overRect253=false;
  stroke(0, 0, 0);
}
fill(red253, green253, blue253);
rect(xquad03, yquad253, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status253==true)
  {
```

```
    valor253=port.read();
    delay(250);
    textFont(f, 12);
    text(valor253, 345, 467);
  }
  if (status253==false)
  {
    textFont(f, 12);
    text(valor253, 345, 467);
  }
}
if (mouseX > xquad03 && mouseX < xquad03+bquad &&
  mouseY > yquad263 && mouseY < yquad263+aquad)
{
  overRect263=true;
  stroke(70, 255, 0);
} else
{
  overRect263=false;
  stroke(0, 0, 0);
}
fill(red263, green263, blue263);
rect(xquad03, yquad263, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status263==true)
  {
    valor263=port.read();
    delay(250);
    textFont(f, 12);
    text(valor263, 345, 482);
  }
  if (status263==false)
  {
    textFont(f, 12);
    text(valor263, 345, 482);
  }
}
if (mouseX > xquad03 && mouseX < xquad03+bquad &&
  mouseY > yquad273 && mouseY < yquad273+aquad)
{
  overRect273=true;
  stroke(70, 255, 0);
} else
{
  overRect273=false;
  stroke(0, 0, 0);
}
fill(red273, green273, blue273);
rect(xquad03, yquad273, bquad, aquad);
```

```
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
 if (status273==true)
 {
  valor273=port.read();
  delay(250);
  textFont(f, 12);
  text(valor273, 345, 497);
 }
 if (status273==false)
 {
  textFont(f, 12);
  text(valor273, 345, 497);
 }
}
if (mouseX > xquad03 && mouseX < xquad03+bquad &&
 mouseY > yquad283 && mouseY < yquad283+aquad)
{
 overRect283=true;
 stroke(70, 255, 0);
} else
{
 overRect283=false;
 stroke(0, 0, 0);
}
fill(red283, green283, blue283);
rect(xquad03, yquad283, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
 if (status283==true)
 {
  valor283=port.read();
  delay(250);
  textFont(f, 12);
  text(valor283, 345, 512);
 }
 if (status283==false)
 {
  textFont(f, 12);
  text(valor283, 345, 512);
 }
}
if (mouseX > xquad03 && mouseX < xquad03+bquad &&
 mouseY > yquad293 && mouseY < yquad293+aquad)
{
 overRect293=true;
 stroke(70, 255, 0);
```

```
} else
{
 overRect293=false;
 stroke(0, 0, 0);
}
fill(red293, green293, blue293);
rect(xquad03, yquad293, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
 if (status293==true)
 {
  valor293=port.read();
  delay(250);
  textFont(f, 12);
  text(valor293, 345, 527);
 }
 if (status293==false)
 {
  textFont(f, 12);
  text(valor293, 345, 527);
 }
}

if (mouseX > xquad03 && mouseX < xquad03+bquad &&
 mouseY > yquad303 && mouseY < yquad303+aquad)
{
 overRect303=true;
 stroke(70, 255, 0);
} else
{
 overRect303=false;
 stroke(0, 0, 0);
}
fill(red303, green303, blue303);
rect(xquad03, yquad303, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
 if (status303==true)
 {
  valor303=port.read();
  delay(250);
  textFont(f, 12);
  text(valor303, 345, 542);
 }
 if (status303==false)
 {
  textFont(f, 12);
```

```
   text(valor303, 345, 542);
  }
}
if (mouseX > xquad03 && mouseX < xquad03+bquad &&
  mouseY > yquad313 && mouseY < yquad313+aquad)
{
  overRect313=true;
  stroke(70, 255, 0);
} else
{
  overRect313=false;
  stroke(0, 0, 0);
}
fill(red313, green313, blue313);
rect(xquad03, yquad313, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status313==true)
  {
    valor313=port.read();
    delay(250);
    textFont(f, 12);
    text(valor313, 345, 557);
  }
  if (status313==false)
  {
    textFont(f, 12);
    text(valor313, 345, 557);
  }
}
////// COLUMNA G STRUT 4 ///////
if (mouseX > xquad04 && mouseX < xquad04+bquad &&
  mouseY > yquad04 && mouseY < yquad04+aquad)
{
  overRect04=true;
  stroke(70, 255, 0);
} else
{
  overRect04=false;
  stroke(0, 0, 0);
}
fill(red04, green04, blue04);
rect(xquad04, yquad04, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status04==true)
  {
```

```
    valor04=port.read();
    delay(250);
    textFont(f, 12);
    text(valor04, 465, 92);
  }
  if (status04==false)
  {
    textFont(f, 12);
    text(valor04, 465, 92);
  }
}
if (mouseX > xquad04 && mouseX < xquad04+bquad &&
  mouseY > yquad14 && mouseY < yquad14+aquad)
{
  overRect14=true;
  stroke(70, 255, 0);
} else
{
  overRect14=false;
  stroke(0, 0, 0);
}
fill(red14, green14, blue14);
rect(xquad04, yquad14, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status14==true)
  {
    valor14=port.read();
    delay(250);
    textFont(f, 12);
    text(valor14, 465, 107);
  }
  if (status14==false)
  {
    textFont(f, 12);
    text(valor14, 465, 107);
  }
}
if (mouseX > xquad04 && mouseX < xquad04+bquad &&
  mouseY > yquad24 && mouseY < yquad24+aquad)
{
  overRect24=true;
  stroke(70, 255, 0);
} else
{
  overRect24=false;
  stroke(0, 0, 0);
}
fill(red24, green24, blue24);
rect(xquad04, yquad24, bquad, aquad);
```

```
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
 if (status24==true)
 {
  valor24=port.read();
  delay(250);
  textFont(f, 12);
  text(valor24, 465, 122);
 }
 if (status24==false)
 {
  textFont(f, 12);
  text(valor24, 465, 122);
 }
}
if (mouseX > xquad04 && mouseX < xquad04+bquad &&
 mouseY > yquad34 && mouseY < yquad34+aquad)
{
 overRect34=true;
 stroke(70, 255, 0);
} else
{
 overRect34=false;
 stroke(0, 0, 0);
}
fill(red34, green34, blue34);
rect(xquad04, yquad34, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
 if (status34==true)
 {
  valor34=port.read();
  delay(250);
  textFont(f, 12);
  text(valor34, 465, 137);
 }
 if (status34==false)
 {
  textFont(f, 12);
  text(valor34, 465, 137);
 }
}
if (mouseX > xquad04 && mouseX < xquad04+bquad &&
 mouseY > yquad44 && mouseY < yquad44+aquad)
{
 overRect44=true;
 stroke(70, 255, 0);
```

```
} else
{
 overRect44=false;
 stroke(0, 0, 0);
}
fill(red44, green44, blue44);
rect(xquad04, yquad44, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
 if (status44==true)
 {
  valor44=port.read();
  delay(250);
  textFont(f, 12);
  text(valor44, 465, 152);
 }
 if (status44==false)
 {
  textFont(f, 12);
  text(valor44, 465, 152);
 }
}
if (mouseX > xquad04 && mouseX < xquad04+bquad &&
 mouseY > yquad54 && mouseY < yquad54+aquad)
{
 overRect54=true;
 stroke(70, 255, 0);
} else
{
 overRect54=false;
 stroke(0, 0, 0);
}
fill(red54, green54, blue54);
rect(xquad04, yquad54, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
 if (status54==true)
 {
  valor54=port.read();
  delay(250);
  textFont(f, 12);
  text(valor54, 465, 167);
 }
 if (status54==false)
 {
  textFont(f, 12);
  text(valor54, 465, 167);
```

```
  }
}
if (mouseX > xquad04 && mouseX < xquad04+bquad &&
  mouseY > yquad64 && mouseY < yquad64+aquad)
{
  overRect64=true;
  stroke(70, 255, 0);
} else
{
  overRect64=false;
  stroke(0, 0, 0);
}
fill(red64, green64, blue64);
rect(xquad04, yquad64, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status64==true)
  {
    valor64=port.read();
    delay(250);
    textFont(f, 12);
    text(valor64, 465, 182);
  }
  if (status64==false)
  {
    textFont(f, 12);
    text(valor64, 465, 182);
  }
}
if (mouseX > xquad04 && mouseX < xquad04+bquad &&
  mouseY > yquad74 && mouseY < yquad74+aquad)
{
  overRect74=true;
  stroke(70, 255, 0);
} else
{
  overRect74=false;
  stroke(0, 0, 0);
}
fill(red74, green74, blue74);
rect(xquad04, yquad74, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status74==true)
  {
    valor74=port.read();
    delay(250);
```

```
   textFont(f, 12);
   text(valor74, 465, 197);
  }
  if (status74==false)
  {
   textFont(f, 12);
   text(valor74, 465, 197);
  }
}
if (mouseX > xquad04 && mouseX < xquad04+bquad &&
  mouseY > yquad84 && mouseY < yquad84+aquad)
{
  overRect84=true;
  stroke(70, 255, 0);
} else
{
  overRect84=false;
  stroke(0, 0, 0);
}
fill(red84, green84, blue84);
rect(xquad04, yquad84, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status84==true)
  {
   valor84=port.read();
   delay(250);
   textFont(f, 12);
   text(valor84, 465, 212);
  }
  if (status84==false)
  {
   textFont(f, 12);
   text(valor84, 465, 212);
  }
}
if (mouseX > xquad04 && mouseX < xquad04+bquad &&
  mouseY > yquad94 && mouseY < yquad94+aquad)
{
  overRect94=true;
  stroke(70, 255, 0);
} else
{
  overRect94=false;
  stroke(0, 0, 0);
}
fill(red94, green94, blue94);
rect(xquad04, yquad94, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
```

```
text(texto, 250, 600);
if (port.available()>0)
{
 if (status94==true)
 {
  valor94=port.read();
  delay(250);
  textFont(f, 12);
  text(valor94, 465, 227);
 }
 if (status94==false)
 {
  textFont(f, 12);
  text(valor94, 465, 227);
 }
}
if (mouseX > xquad04 && mouseX < xquad04+bquad &&
 mouseY > yquad104 && mouseY < yquad104+aquad)
{
 overRect104=true;
 stroke(70, 255, 0);
} else
{
 overRect104=false;
 stroke(0, 0, 0);
}
fill(red104, green104, blue104);
rect(xquad04, yquad104, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
 if (status104==true)
 {
  valor104=port.read();
  delay(250);
  textFont(f, 12);
  text(valor104, 465, 242);
 }
 if (status104==false)
 {
  textFont(f, 12);
  text(valor104, 465, 242);
 }
}
if (mouseX > xquad04 && mouseX < xquad04+bquad &&
 mouseY > yquad114 && mouseY < yquad114+aquad)
{
 overRect114=true;
 stroke(70, 255, 0);
} else
{
```

```
  overRect114=false;
  stroke(0, 0, 0);
}
fill(red114, green114, blue114);
rect(xquad04, yquad114, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status114==true)
  {
    valor114=port.read();
    delay(250);
    textFont(f, 12);
    text(valor114, 465, 257);
  }
  if (status114==false)
  {
    textFont(f, 12);
    text(valor114, 465, 257);
  }
}
if (mouseX > xquad04 && mouseX < xquad04+bquad &&
  mouseY > yquad124 && mouseY < yquad124+aquad)
{
  overRect124=true;
  stroke(70, 255, 0);
} else
{
  overRect124=false;
  stroke(0, 0, 0);
}
fill(red124, green124, blue124);
rect(xquad04, yquad124, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status124==true)
  {
    valor124=port.read();
    delay(250);
    textFont(f, 12);
    text(valor124, 465, 272);
  }
  if (status124==false)
  {
    textFont(f, 12);
    text(valor124, 465, 272);
  }
}
```

```
if (mouseX > xquad04 && mouseX < xquad04+bquad &&
  mouseY > yquad134 && mouseY < yquad134+aquad)
{
  overRect134=true;
  stroke(70, 255, 0);
} else
{
  overRect134=false;
  stroke(0, 0, 0);
}
fill(red134, green134, blue134);
rect(xquad04, yquad134, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status134==true)
  {
    valor134=port.read();
    delay(250);
    textFont(f, 12);
    text(valor134, 465, 287);
  }
  if (status134==false)
  {
    textFont(f, 12);
    text(valor134, 465, 287);
  }
}
if (mouseX > xquad04 && mouseX < xquad04+bquad &&
  mouseY > yquad144 && mouseY < yquad144+aquad)
{
  overRect144=true;
  stroke(70, 255, 0);
} else
{
  overRect144=false;
  stroke(0, 0, 0);
}
fill(red144, green144, blue144);
rect(xquad04, yquad144, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status144==true)
  {
    valor144=port.read();
    delay(250);
    textFont(f, 12);
    text(valor144, 465, 302);
```

```
  }
  if (status144==false)
  {
    textFont(f, 12);
    text(valor144, 465, 302);
  }
}
if (mouseX > xquad04 && mouseX < xquad04+bquad &&
  mouseY > yquad154 && mouseY < yquad154+aquad)
{
  overRect154=true;
  stroke(70, 255, 0);
} else
{
  overRect154=false;
  stroke(0, 0, 0);
}
fill(red154, green154, blue154);
rect(xquad04, yquad154, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status154==true)
  {
    valor154=port.read();
    delay(250);
    textFont(f, 12);
    text(valor154, 465, 317);
  }
  if (status154==false)
  {
    textFont(f, 12);
    text(valor154, 465, 317);
  }
}
if (mouseX > xquad04 && mouseX < xquad04+bquad &&
  mouseY > yquad164 && mouseY < yquad164+aquad)
{
  overRect164=true;
  stroke(70, 255, 0);
} else
{
  overRect164=false;
  stroke(0, 0, 0);
}
fill(red164, green164, blue164);
rect(xquad04, yquad164, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
```

```
{
 if (status164==true)
 {
  valor164=port.read();
  delay(250);
  textFont(f, 12);
  text(valor164, 465, 332);
 }
 if (status164==false)
 {
  textFont(f, 12);
  text(valor164, 465, 332);
 }
}
if (mouseX > xquad04 && mouseX < xquad04+bquad &&
 mouseY > yquad174 && mouseY < yquad174+aquad)
{
 overRect174=true;
 stroke(70, 255, 0);
} else
{
 overRect174=false;
 stroke(0, 0, 0);
}
fill(red174, green174, blue174);
rect(xquad04, yquad174, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
 if (status174==true)
 {
  valor174=port.read();
  delay(250);
  textFont(f, 12);
  text(valor174, 465, 347);
 }
 if (status174==false)
 {
  textFont(f, 12);
  text(valor174, 465, 347);
 }
}
if (mouseX > xquad04 && mouseX < xquad04+bquad &&
 mouseY > yquad184 && mouseY < yquad184+aquad)
{
 overRect184=true;
 stroke(70, 255, 0);
} else
{
 overRect184=false;
 stroke(0, 0, 0);
```

```
}
fill(red184, green184, blue184);
rect(xquad04, yquad184, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
 if (status184==true)
 {
  valor184=port.read();
  delay(250);
  textFont(f, 12);
  text(valor184, 465, 362);
 }
 if (status184==false)
 {
  textFont(f, 12);
  text(valor184, 465, 362);
 }
}
if (mouseX > xquad04 && mouseX < xquad04+bquad &&
 mouseY > yquad194 && mouseY < yquad194+aquad)
{
 overRect194=true;
 stroke(70, 255, 0);
} else
{
 overRect194=false;
 stroke(0, 0, 0);
}
fill(red194, green194, blue194);
rect(xquad04, yquad194, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
 if (status194==true)
 {
  valor194=port.read();
  delay(250);
  textFont(f, 12);
  text(valor194, 465, 377);
 }
 if (status194==false)
 {
  textFont(f, 12);
  text(valor194, 465, 377);
 }
}
if (mouseX > xquad04 && mouseX < xquad04+bquad &&
 mouseY > yquad204 && mouseY < yquad204+aquad)
```

```
{
 overRect204=true;
 stroke(70, 255, 0);
} else
{
 overRect204=false;
 stroke(0, 0, 0);
}
fill(red204, green204, blue204);
rect(xquad04, yquad204, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
 if (status204==true)
 {
  valor204=port.read();
  delay(250);
  textFont(f, 12);
  text(valor204, 465, 392);
 }
 if (status204==false)
 {
  textFont(f, 12);
  text(valor204, 465, 392);
 }
}
if (mouseX > xquad04 && mouseX < xquad04+bquad &&
 mouseY > yquad214 && mouseY < yquad214+aquad)
{
 overRect214=true;
 stroke(70, 255, 0);
} else
{
 overRect214=false;
 stroke(0, 0, 0);
}
fill(red214, green214, blue214);
rect(xquad04, yquad214, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
 if (status214==true)
 {
  valor214=port.read();
  delay(250);
  textFont(f, 12);
  text(valor214, 465, 407);
 }
 if (status214==false)
```

```
  {
   textFont(f, 12);
   text(valor214, 465, 407);
  }
}
if (mouseX > xquad04 && mouseX < xquad04+bquad &&
 mouseY > yquad224 && mouseY < yquad224+aquad)
{
 overRect224=true;
 stroke(70, 255, 0);
} else
{
 overRect224=false;
 stroke(0, 0, 0);
}
fill(red224, green224, blue224);
rect(xquad04, yquad224, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
 if (status224==true)
 {
  valor224=port.read();
  delay(250);
  textFont(f, 12);
  text(valor224, 465, 422);
 }
 if (status224==false)
 {
  textFont(f, 12);
  text(valor224, 465, 422);
 }
}
if (mouseX > xquad04 && mouseX < xquad04+bquad &&
 mouseY > yquad234 && mouseY < yquad234+aquad)
{
 overRect234=true;
 stroke(70, 255, 0);
} else
{
 overRect234=false;
 stroke(0, 0, 0);
}
fill(red234, green234, blue234);
rect(xquad04, yquad234, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
 if (status234==true)
```

```
  {
   valor234=port.read();
   delay(250);
   textFont(f, 12);
   text(valor234, 465, 437);
  }
  if (status234==false)
  {
   textFont(f, 12);
   text(valor234, 465, 437);
  }
}
if (mouseX > xquad04 && mouseX < xquad04+bquad &&
  mouseY > yquad244 && mouseY < yquad244+aquad)
{
  overRect244=true;
  stroke(70, 255, 0);
} else
{
  overRect244=false;
  stroke(0, 0, 0);
}
fill(red244, green244, blue244);
rect(xquad04, yquad244, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status244==true)
  {
   valor244=port.read();
   delay(250);
   textFont(f, 12);
   text(valor244, 465, 452);
  }
  if (status244==false)
  {
   textFont(f, 12);
   text(valor244, 465, 452);
  }
}
if (mouseX > xquad04 && mouseX < xquad04+bquad &&
  mouseY > yquad254 && mouseY < yquad254+aquad)
{
  overRect254=true;
  stroke(70, 255, 0);
} else
{
  overRect254=false;
  stroke(0, 0, 0);
}
fill(red254, green254, blue254);
```

```
rect(xquad04, yquad254, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
 if (status254==true)
 {
  valor254=port.read();
  delay(250);
  textFont(f, 12);
  text(valor254, 465, 467);
 }
 if (status254==false)
 {
  textFont(f, 12);
  text(valor254, 465, 467);
 }
}
if (mouseX > xquad04 && mouseX < xquad04+bquad &&
 mouseY > yquad264 && mouseY < yquad264+aquad)
{
 overRect264=true;
 stroke(70, 255, 0);
} else
{
 overRect264=false;
 stroke(0, 0, 0);
}
fill(red264, green264, blue264);
rect(xquad04, yquad264, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
 if (status264==true)
 {
  valor264=port.read();
  delay(250);
  textFont(f, 12);
  text(valor264, 465, 482);
 }
 if (status264==false)
 {
  textFont(f, 12);
  text(valor264, 465, 482);
 }
}
if (mouseX > xquad04 && mouseX < xquad04+bquad &&
 mouseY > yquad274 && mouseY < yquad274+aquad)
{
 overRect274=true;
```

```
  stroke(70, 255, 0);
} else
{
  overRect274=false;
  stroke(0, 0, 0);
}
fill(red274, green274, blue274);
rect(xquad04, yquad274, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status274==true)
  {
    valor274=port.read();
    delay(250);
    textFont(f, 12);
    text(valor274, 465, 497);
  }
  if (status274==false)
  {
    textFont(f, 12);
    text(valor274, 465, 497);
  }
}
if (mouseX > xquad04 && mouseX < xquad04+bquad &&
  mouseY > yquad284 && mouseY < yquad284+aquad)
{
  overRect284=true;
  stroke(70, 255, 0);
} else
{
  overRect284=false;
  stroke(0, 0, 0);
}
fill(red284, green284, blue284);
rect(xquad04, yquad284, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status284==true)
  {
    valor284=port.read();
    delay(250);
    textFont(f, 12);
    text(valor284, 465, 512);
  }
  if (status284==false)
  {
    textFont(f, 12);
```

```
   text(valor284, 465, 512);
  }
}
if (mouseX > xquad04 && mouseX < xquad04+bquad &&
  mouseY > yquad294 && mouseY < yquad294+aquad)
{
  overRect294=true;
  stroke(70, 255, 0);
} else
{
  overRect294=false;
  stroke(0, 0, 0);
}
fill(red294, green294, blue294);
rect(xquad04, yquad294, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status294==true)
  {
    valor294=port.read();
    delay(250);
    textFont(f, 12);
    text(valor294, 465, 527);
  }
  if (status294==false)
  {
    textFont(f, 12);
    text(valor294, 465, 527);
  }
}
if (mouseX > xquad04 && mouseX < xquad04+bquad &&
  mouseY > yquad304 && mouseY < yquad304+aquad)
{
  overRect304=true;
  stroke(70, 255, 0);
} else
{
  overRect304=false;
  stroke(0, 0, 0);
}
fill(red304, green304, blue304);
rect(xquad04, yquad304, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status304==true)
  {
    valor304=port.read();
```

```
    delay(250);
    textFont(f, 12);
    text(valor304, 465, 542);
   }
   if (status304==false)
   {
    textFont(f, 12);
    text(valor304, 465, 542);
   }
}
if (mouseX > xquad04 && mouseX < xquad04+bquad &&
  mouseY > yquad314 && mouseY < yquad314+aquad)
{
  overRect314=true;
  stroke(70, 255, 0);
} else
{
  overRect314=false;
  stroke(0, 0, 0);
}
fill(red314, green314, blue314);
rect(xquad04, yquad314, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status314==true)
  {
    valor314=port.read();
    delay(250);
    textFont(f, 12);
    text(valor314, 465, 557);
   }
   if (status314==false)
   {
    textFont(f, 12);
    text(valor314, 465, 557);
   }
}
////// COLUMNA G STRUT 5 ///////
if (mouseX > xquad05 && mouseX < xquad05+bquad &&
  mouseY > yquad05 && mouseY < yquad05+aquad)
{
  overRect05=true;
  stroke(70, 255, 0);
} else
{
  overRect05=false;
  stroke(0, 0, 0);
}
fill(red05, green05, blue05);
rect(xquad05, yquad05, bquad, aquad);
```

```
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
 if (status05==true)
 {
  valor05=port.read();
  delay(250);
  textFont(f, 12);
  text(valor05, 585, 92);
 }
 if (status05==false)
 {
  textFont(f, 12);
  text(valor05, 585, 92);
 }
}
if (mouseX > xquad05 && mouseX < xquad05+bquad &&
 mouseY > yquad15 && mouseY < yquad15+aquad)
{
 overRect15=true;
 stroke(70, 255, 0);
} else
{
 overRect15=false;
 stroke(0, 0, 0);
}
fill(red15, green15, blue15);
rect(xquad05, yquad15, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
 if (status15==true)
 {
  valor15=port.read();
  delay(250);
  textFont(f, 12);
  text(valor15, 585, 107);
 }
 if (status15==false)
 {
  textFont(f, 12);
  text(valor15, 585, 107);
 }
}
if (mouseX > xquad05 && mouseX < xquad05+bquad &&
 mouseY > yquad25 && mouseY < yquad25+aquad)
{
 overRect25=true;
 stroke(70, 255, 0);
```

```
} else
{
 overRect25=false;
 stroke(0, 0, 0);
}
fill(red25, green25, blue25);
rect(xquad05, yquad25, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
 if (status25==true)
 {
  valor25=port.read();
  delay(250);
  textFont(f, 12);
  text(valor25, 585, 122);
 }
 if (status25==false)
 {
  textFont(f, 12);
  text(valor25, 585, 122);
 }
}
if (mouseX > xquad05 && mouseX < xquad05+bquad &&
 mouseY > yquad35 && mouseY < yquad35+aquad)
{
 overRect35=true;
 stroke(70, 255, 0);
} else
{
 overRect35=false;
 stroke(0, 0, 0);
}
fill(red35, green35, blue35);
rect(xquad05, yquad35, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
 if (status35==true)
 {
  valor35=port.read();
  delay(250);
  textFont(f, 12);
  text(valor35, 585, 137);
 }
 if (status35==false)
 {
  textFont(f, 12);
  text(valor35, 585, 137);
```

```
  }
}
if (mouseX > xquad05 && mouseX < xquad05+bquad &&
  mouseY > yquad45 && mouseY < yquad45+aquad)
{
  overRect45=true;
  stroke(70, 255, 0);
} else
{
  overRect45=false;
  stroke(0, 0, 0);
}
fill(red45, green45, blue45);
rect(xquad05, yquad45, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status45==true)
  {
    valor45=port.read();
    delay(250);
    textFont(f, 12);
    text(valor45, 585, 152);
  }
  if (status45==false)
  {
    textFont(f, 12);
    text(valor45, 585, 152);
  }
}
if (mouseX > xquad05 && mouseX < xquad05+bquad &&
  mouseY > yquad55 && mouseY < yquad55+aquad)
{
  overRect55=true;
  stroke(70, 255, 0);
} else
{
  overRect55=false;
  stroke(0, 0, 0);
}
fill(red55, green55, blue55);
rect(xquad05, yquad55, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status55==true)
  {
    valor55=port.read();
    delay(250);
```

```
   textFont(f, 12);
   text(valor55, 585, 167);
  }
  if (status55==false)
  {
   textFont(f, 12);
   text(valor55, 585, 167);
  }
}
if (mouseX > xquad05 && mouseX < xquad05+bquad &&
  mouseY > yquad65 && mouseY < yquad65+aquad)
{
  overRect65=true;
  stroke(70, 255, 0);
} else
{
  overRect65=false;
  stroke(0, 0, 0);
}
fill(red65, green65, blue65);
rect(xquad05, yquad65, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status65==true)
  {
   valor65=port.read();
   delay(250);
   textFont(f, 12);
   text(valor65, 585, 182);
  }
  if (status65==false)
  {
   textFont(f, 12);
   text(valor65, 585, 182);
  }
}
if (mouseX > xquad05 && mouseX < xquad05+bquad &&
  mouseY > yquad75 && mouseY < yquad75+aquad)
{
  overRect75=true;
  stroke(70, 255, 0);
} else
{
  overRect75=false;
  stroke(0, 0, 0);
}
fill(red75, green75, blue75);
rect(xquad05, yquad75, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
```

```
text(texto, 250, 600);
if (port.available()>0)
{
 if (status75==true)
 {
  valor75=port.read();
  delay(250);
  textFont(f, 12);
  text(valor75, 585, 197);
 }
 if (status75==false)
 {
  textFont(f, 12);
  text(valor75, 585, 197);
 }
}
if (mouseX > xquad05 && mouseX < xquad05+bquad &&
 mouseY > yquad85 && mouseY < yquad85+aquad)
{
 overRect85=true;
 stroke(70, 255, 0);
} else
{
 overRect85=false;
 stroke(0, 0, 0);
}
fill(red85, green85, blue85);
rect(xquad05, yquad85, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
 if (status85==true)
 {
  valor85=port.read();
  delay(250);
  textFont(f, 12);
  text(valor85, 585, 212);
 }
 if (status85==false)
 {
  textFont(f, 12);
  text(valor85, 585, 212);
 }
}
if (mouseX > xquad05 && mouseX < xquad05+bquad &&
 mouseY > yquad95 && mouseY < yquad95+aquad)
{
 overRect95=true;
 stroke(70, 255, 0);
} else
{
```

```
  overRect95=false;
  stroke(0, 0, 0);
}
fill(red95, green95, blue95);
rect(xquad05, yquad95, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status95==true)
  {
    valor95=port.read();
    delay(250);
    textFont(f, 12);
    text(valor95, 585, 227);
  }
  if (status95==false)
  {
    textFont(f, 12);
    text(valor95, 585, 227);
  }
}
if (mouseX > xquad05 && mouseX < xquad05+bquad &&
  mouseY > yquad105 && mouseY < yquad105+aquad)
{
  overRect105=true;
  stroke(70, 255, 0);
} else
{
  overRect105=false;
  stroke(0, 0, 0);
}
fill(red105, green105, blue105);
rect(xquad05, yquad105, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status105==true)
  {
    valor105=port.read();
    delay(250);
    textFont(f, 12);
    text(valor105, 585, 242);
  }
  if (status105==false)
  {
    textFont(f, 12);
    text(valor105, 585, 242);
  }
}
```

```
if (mouseX > xquad05 && mouseX < xquad05+bquad &&
  mouseY > yquad115 && mouseY < yquad115+aquad)
{
  overRect115=true;
  stroke(70, 255, 0);
} else
{
  overRect115=false;
  stroke(0, 0, 0);
}
fill(red115, green115, blue115);
rect(xquad05, yquad115, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status115==true)
  {
    valor115=port.read();
    delay(250);
    textFont(f, 12);
    text(valor115, 585, 257);
  }
  if (status115==false)
  {
    textFont(f, 12);
    text(valor115, 585, 257);
  }
}
if (mouseX > xquad05 && mouseX < xquad05+bquad &&
  mouseY > yquad125 && mouseY < yquad125+aquad)
{
  overRect125=true;
  stroke(70, 255, 0);
} else
{
  overRect125=false;
  stroke(0, 0, 0);
}
fill(red125, green125, blue125);
rect(xquad05, yquad125, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status125==true)
  {
    valor125=port.read();
    delay(250);
    textFont(f, 12);
    text(valor125, 585, 272);
```

```
  }
  if (status125==false)
  {
    textFont(f, 12);
    text(valor125, 585, 272);
  }
}
if (mouseX > xquad05 && mouseX < xquad05+bquad &&
  mouseY > yquad135 && mouseY < yquad135+aquad)
{
  overRect135=true;
  stroke(70, 255, 0);
} else
{
  overRect135=false;
  stroke(0, 0, 0);
}
fill(red135, green135, blue135);
rect(xquad05, yquad135, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status135==true)
  {
    valor135=port.read();
    delay(250);
    textFont(f, 12);
    text(valor135, 585, 287);
  }
  if (status135==false)
  {
    textFont(f, 12);
    text(valor135, 585, 287);
  }
}
if (mouseX > xquad05 && mouseX < xquad05+bquad &&
  mouseY > yquad145 && mouseY < yquad145+aquad)
{
  overRect145=true;
  stroke(70, 255, 0);
} else
{
  overRect145=false;
  stroke(0, 0, 0);
}
fill(red145, green145, blue145);
rect(xquad05, yquad145, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
```

```
{
 if (status145==true)
 {
  valor145=port.read();
  delay(250);
  textFont(f, 12);
  text(valor145, 585, 302);
 }
 if (status145==false)
 {
  textFont(f, 12);
  text(valor145, 585, 302);
 }
}
if (mouseX > xquad05 && mouseX < xquad05+bquad &&
 mouseY > yquad155 && mouseY < yquad155+aquad)
{
 overRect155=true;
 stroke(70, 255, 0);
} else
{
 overRect155=false;
 stroke(0, 0, 0);
}
fill(red155, green155, blue155);
rect(xquad05, yquad155, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
 if (status155==true)
 {
  valor155=port.read();
  delay(250);
  textFont(f, 12);
  text(valor155, 585, 317);
 }
 if (status155==false)
 {
  textFont(f, 12);
  text(valor155, 585, 317);
 }
}
if (mouseX > xquad05 && mouseX < xquad05+bquad &&
 mouseY > yquad165 && mouseY < yquad165+aquad)
{
 overRect165=true;
 stroke(70, 255, 0);
} else
{
 overRect165=false;
 stroke(0, 0, 0);
```

```
}
fill(red165, green165, blue165);
rect(xquad05, yquad165, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
 if (status165==true)
 {
  valor165=port.read();
  delay(250);
  textFont(f, 12);
  text(valor165, 585, 332);
 }
 if (status165==false)
 {
  textFont(f, 12);
  text(valor165, 585, 332);
 }
}
if (mouseX > xquad05 && mouseX < xquad05+bquad &&
 mouseY > yquad175 && mouseY < yquad175+aquad)
{
 overRect175=true;
 stroke(70, 255, 0);
} else
{
 overRect175=false;
 stroke(0, 0, 0);
}
fill(red175, green175, blue175);
rect(xquad05, yquad175, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
 if (status175==true)
 {
  valor175=port.read();
  delay(250);
  textFont(f, 12);
  text(valor175, 585, 347);
 }
 if (status175==false)
 {
  textFont(f, 12);
  text(valor175, 585, 347);
 }
}
if (mouseX > xquad05 && mouseX < xquad05+bquad &&
 mouseY > yquad185 && mouseY < yquad185+aquad)
```

```
{
 overRect185=true;
 stroke(70, 255, 0);
} else
{
 overRect185=false;
 stroke(0, 0, 0);
}
fill(red185, green185, blue185);
rect(xquad05, yquad185, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
 if (status185==true)
 {
  valor185=port.read();
  delay(250);
  textFont(f, 12);
  text(valor185, 585, 362);
 }
 if (status185==false)
 {
  textFont(f, 12);
  text(valor185, 585, 362);
 }
}
if (mouseX > xquad05 && mouseX < xquad05+bquad &&
 mouseY > yquad195 && mouseY < yquad195+aquad)
{
 overRect195=true;
 stroke(70, 255, 0);
} else
{
 overRect195=false;
 stroke(0, 0, 0);
}
fill(red195, green195, blue195);
rect(xquad05, yquad195, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
 if (status195==true)
 {
  valor195=port.read();
  delay(250);
  textFont(f, 12);
  text(valor195, 585, 377);
 }
 if (status195==false)
```

```
  {
   textFont(f, 12);
   text(valor195, 585, 377);
  }
}
if (mouseX > xquad05 && mouseX < xquad05+bquad &&
  mouseY > yquad205 && mouseY < yquad205+aquad)
{
  overRect205=true;
  stroke(70, 255, 0);
} else
{
  overRect205=false;
  stroke(0, 0, 0);
}
fill(red205, green205, blue205);
rect(xquad05, yquad205, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status205==true)
  {
    valor205=port.read();
    delay(250);
    textFont(f, 12);
    text(valor205, 585, 392);
  }
  if (status205==false)
  {
    textFont(f, 12);
    text(valor205, 585, 392);
  }
}
if (mouseX > xquad05 && mouseX < xquad05+bquad &&
  mouseY > yquad215 && mouseY < yquad215+aquad)
{
  overRect215=true;
  stroke(70, 255, 0);
} else
{
  overRect215=false;
  stroke(0, 0, 0);
}
fill(red215, green215, blue215);
rect(xquad05, yquad215, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status215==true)
```

```
  {
   valor215=port.read();
   delay(250);
   textFont(f, 12);
   text(valor215, 585, 407);
  }
  if (status215==false)
  {
   textFont(f, 12);
   text(valor215, 585, 407);
  }
}
if (mouseX > xquad05 && mouseX < xquad05+bquad &&
  mouseY > yquad225 && mouseY < yquad225+aquad)
{
  overRect225=true;
  stroke(70, 255, 0);
} else
{
  overRect225=false;
  stroke(0, 0, 0);
}
fill(red225, green225, blue225);
rect(xquad05, yquad225, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status225==true)
  {
   valor225=port.read();
   delay(250);
   textFont(f, 12);
   text(valor225, 585, 422);
  }
  if (status225==false)
  {
   textFont(f, 12);
   text(valor225, 585, 422);
  }
}
if (mouseX > xquad05 && mouseX < xquad05+bquad &&
  mouseY > yquad235 && mouseY < yquad235+aquad)
{
  overRect235=true;
  stroke(70, 255, 0);
} else
{
  overRect235=false;
  stroke(0, 0, 0);
}
fill(red235, green235, blue235);
```

```
rect(xquad05, yquad235, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status235==true)
  {
    valor235=port.read();
    delay(250);
    textFont(f, 12);
    text(valor235, 585, 437);
  }
  if (status235==false)
  {
    textFont(f, 12);
    text(valor235, 585, 437);
  }
}
if (mouseX > xquad05 && mouseX < xquad05+bquad &&
  mouseY > yquad245 && mouseY < yquad245+aquad)
{
  overRect245=true;
  stroke(70, 255, 0);
} else
{
  overRect245=false;
  stroke(0, 0, 0);
}
fill(red245, green245, blue245);
rect(xquad05, yquad245, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status245==true)
  {
    valor245=port.read();
    delay(250);
    textFont(f, 12);
    text(valor245, 585, 452);
  }
  if (status245==false)
  {
    textFont(f, 12);
    text(valor245, 585, 452);
  }
}
if (mouseX > xquad05 && mouseX < xquad05+bquad &&
  mouseY > yquad255 && mouseY < yquad255+aquad)
{
  overRect255=true;
```

```
  stroke(70, 255, 0);
} else
{
  overRect255=false;
  stroke(0, 0, 0);
}
fill(red255, green255, blue255);
rect(xquad05, yquad255, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status255==true)
  {
    valor255=port.read();
    delay(250);
    textFont(f, 12);
    text(valor255, 585, 467);
  }
  if (status255==false)
  {
    textFont(f, 12);
    text(valor255, 585, 467);
  }
}
if (mouseX > xquad05 && mouseX < xquad05+bquad &&
  mouseY > yquad265 && mouseY < yquad265+aquad)
{
  overRect265=true;
  stroke(70, 255, 0);
} else
{
  overRect265=false;
  stroke(0, 0, 0);
}
fill(red265, green265, blue265);
rect(xquad05, yquad265, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status265==true)
  {
    valor265=port.read();
    delay(250);
    textFont(f, 12);
    text(valor265, 585, 482);
  }
  if (status265==false)
  {
    textFont(f, 12);
```

```
   text(valor265, 585, 482);
  }
}
if (mouseX > xquad05 && mouseX < xquad05+bquad &&
  mouseY > yquad275 && mouseY < yquad275+aquad)
{
  overRect275=true;
  stroke(70, 255, 0);
} else
{
  overRect275=false;
  stroke(0, 0, 0);
}
fill(red275, green275, blue275);
rect(xquad05, yquad275, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status275==true)
  {
    valor275=port.read();
    delay(250);
    textFont(f, 12);
    text(valor275, 585, 497);
  }
  if (status275==false)
  {
    textFont(f, 12);
    text(valor275, 585, 497);
  }
}
if (mouseX > xquad05 && mouseX < xquad05+bquad &&
  mouseY > yquad285 && mouseY < yquad285+aquad)
{
  overRect285=true;
  stroke(70, 255, 0);
} else
{
  overRect285=false;
  stroke(0, 0, 0);
}
fill(red285, green285, blue285);
rect(xquad05, yquad285, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status285==true)
  {
    valor285=port.read();
```

```
    delay(250);
    textFont(f, 12);
    text(valor285, 585, 512);
   }
  if (status285==false)
  {
    textFont(f, 12);
    text(valor285, 585, 512);
  }
}
if (mouseX > xquad05 && mouseX < xquad05+bquad &&
  mouseY > yquad295 && mouseY < yquad295+aquad)
{
  overRect295=true;
  stroke(70, 255, 0);
} else
{
  overRect295=false;
  stroke(0, 0, 0);
}
fill(red295, green295, blue295);
rect(xquad05, yquad295, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status295==true)
  {
    valor295=port.read();
    delay(250);
    textFont(f, 12);
    text(valor295, 585, 527);
  }
  if (status295==false)
  {
    textFont(f, 12);
    text(valor295, 585, 527);
  }
}
if (mouseX > xquad05 && mouseX < xquad05+bquad &&
  mouseY > yquad305 && mouseY < yquad305+aquad)
{
  overRect305=true;
  stroke(70, 255, 0);
} else
{
  overRect305=false;
  stroke(0, 0, 0);
}
fill(red305, green305, blue305);
rect(xquad05, yquad305, bquad, aquad);
fill(0, 0, 0);
```

```
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
 if (status305==true)
 {
  valor305=port.read();
  delay(250);
  textFont(f, 12);
  text(valor305, 585, 542);
 }
 if (status305==false)
 {
  textFont(f, 12);
  text(valor305, 585, 542);
 }
}
if (mouseX > xquad05 && mouseX < xquad05+bquad &&
 mouseY > yquad315 && mouseY < yquad315+aquad)
{
 overRect315=true;
 stroke(70, 255, 0);
} else
{
 overRect315=false;
 stroke(0, 0, 0);
}
fill(red315, green315, blue315);
rect(xquad05, yquad315, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
 if (status315==true)
 {
  valor315=port.read();
  delay(250);
  textFont(f, 12);
  text(valor315, 585, 557);
 }
 if (status315==false)
 {
  textFont(f, 12);
  text(valor315, 585, 557);
 }
}
////// COLUMNA G STRUT 6 ///////
if (mouseX > xquad06 && mouseX < xquad06+bquad &&
 mouseY > yquad06 && mouseY < yquad06+aquad)
{
 overRect06=true;
 stroke(70, 255, 0);
```

```
} else
{
  overRect06=false;
  stroke(0, 0, 0);
}
fill(red06, green06, blue06);
rect(xquad06, yquad06, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status06==true)
  {
    valor06=port.read();
    delay(250);
    textFont(f, 12);
    text(valor06, 705, 92);
  }
  if (status06==false)
  {
    textFont(f, 12);
    text(valor06, 705, 92);
  }
}
if (mouseX > xquad06 && mouseX < xquad06+bquad &&
  mouseY > yquad16 && mouseY < yquad16+aquad)
{
  overRect16=true;
  stroke(70, 255, 0);
} else
{
  overRect16=false;
  stroke(0, 0, 0);
}
fill(red16, green16, blue16);
rect(xquad06, yquad16, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status16==true)
  {
    valor16=port.read();
    delay(250);
    textFont(f, 12);
    text(valor16, 705, 107);
  }
  if (status16==false)
  {
    textFont(f, 12);
    text(valor16, 705, 107);
```

```
  }
}
if (mouseX > xquad06 && mouseX < xquad06+bquad &&
  mouseY > yquad26 && mouseY < yquad26+aquad)
{
  overRect26=true;
  stroke(70, 255, 0);
} else
{
  overRect26=false;
  stroke(0, 0, 0);
}
fill(red26, green26, blue26);
rect(xquad06, yquad26, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status26==true)
  {
    valor26=port.read();
    delay(250);
    textFont(f, 12);
    text(valor26, 705, 122);
  }
  if (status26==false)
  {
    textFont(f, 12);
    text(valor26, 705, 122);
  }
}
if (mouseX > xquad06 && mouseX < xquad06+bquad &&
  mouseY > yquad36 && mouseY < yquad36+aquad)
{
  overRect36=true;
  stroke(70, 255, 0);
} else
{
  overRect36=false;
  stroke(0, 0, 0);
}
fill(red36, green36, blue36);
rect(xquad06, yquad36, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status36==true)
  {
    valor36=port.read();
    delay(250);
```

```
   textFont(f, 12);
   text(valor36, 705, 137);
  }
  if (status35==false)
  {
   textFont(f, 12);
   text(valor36, 705, 137);
  }
}
if (mouseX > xquad06 && mouseX < xquad06+bquad &&
  mouseY > yquad46 && mouseY < yquad46+aquad)
{
  overRect46=true;
  stroke(70, 255, 0);
} else
{
  overRect46=false;
  stroke(0, 0, 0);
}
fill(red46, green46, blue46);
rect(xquad06, yquad46, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status46==true)
  {
   valor46=port.read();
   delay(250);
   textFont(f, 12);
   text(valor46, 705, 152);
  }
  if (status46==false)
  {
   textFont(f, 12);
   text(valor46, 705, 152);
  }
}
if (mouseX > xquad06 && mouseX < xquad06+bquad &&
  mouseY > yquad56 && mouseY < yquad56+aquad)
{
  overRect56=true;
  stroke(70, 255, 0);
} else
{
  overRect56=false;
  stroke(0, 0, 0);
}
fill(red56, green56, blue56);
rect(xquad06, yquad56, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
```

```
text(texto, 250, 600);
if (port.available()>0)
{
 if (status56==true)
 {
  valor56=port.read();
  delay(250);
  textFont(f, 12);
  text(valor56, 705, 167);
 }
 if (status56==false)
 {
  textFont(f, 12);
  text(valor56, 705, 167);
 }
}
if (mouseX > xquad06 && mouseX < xquad06+bquad &&
 mouseY > yquad66 && mouseY < yquad66+aquad)
{
 overRect66=true;
 stroke(70, 255, 0);
} else
{
 overRect66=false;
 stroke(0, 0, 0);
}
fill(red66, green66, blue66);
rect(xquad06, yquad66, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
 if (status66==true)
 {
  valor66=port.read();
  delay(250);
  textFont(f, 12);
  text(valor66, 705, 182);
 }
 if (status66==false)
 {
  textFont(f, 12);
  text(valor66, 705, 182);
 }
}
if (mouseX > xquad06 && mouseX < xquad06+bquad &&
 mouseY > yquad76 && mouseY < yquad76+aquad)
{
 overRect76=true;
 stroke(70, 255, 0);
} else
{
```

```
  overRect76=false;
  stroke(0, 0, 0);
}
fill(red76, green76, blue76);
rect(xquad06, yquad76, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status76==true)
  {
    valor76=port.read();
    delay(250);
    textFont(f, 12);
    text(valor76, 705, 197);
  }
  if (status76==false)
  {
    textFont(f, 12);
    text(valor76, 705, 197);
  }
}
if (mouseX > xquad06 && mouseX < xquad06+bquad &&
  mouseY > yquad86 && mouseY < yquad86+aquad)
{
  overRect86=true;
  stroke(70, 255, 0);
} else
{
  overRect86=false;
  stroke(0, 0, 0);
}
fill(red86, green86, blue86);
rect(xquad06, yquad86, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status86==true)
  {
    valor86=port.read();
    delay(250);
    textFont(f, 12);
    text(valor86, 705, 212);
  }
  if (status86==false)
  {
    textFont(f, 12);
    text(valor86, 705, 212);
  }
}
```

```
if (mouseX > xquad06 && mouseX < xquad06+bquad &&
  mouseY > yquad96 && mouseY < yquad96+aquad)
{
  overRect96=true;
  stroke(70, 255, 0);
} else
{
  overRect96=false;
  stroke(0, 0, 0);
}
fill(red96, green96, blue96);
rect(xquad06, yquad96, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status96==true)
  {
    valor96=port.read();
    delay(250);
    textFont(f, 12);
    text(valor96, 705, 227);
  }
  if (status96==false)
  {
    textFont(f, 12);
    text(valor96, 705, 227);
  }
}
if (mouseX > xquad06 && mouseX < xquad06+bquad &&
  mouseY > yquad106 && mouseY < yquad106+aquad)
{
  overRect106=true;
  stroke(70, 255, 0);
} else
{
  overRect106=false;
  stroke(0, 0, 0);
}
fill(red106, green106, blue106);
rect(xquad06, yquad106, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status106==true)
  {
    valor106=port.read();
    delay(250);
    textFont(f, 12);
    text(valor106, 705, 242);
```

```
  }
  if (status106==false)
  {
    textFont(f, 12);
    text(valor106, 705, 242);
  }
}
if (mouseX > xquad06 && mouseX < xquad06+bquad &&
  mouseY > yquad116 && mouseY < yquad116+aquad)
{
  overRect116=true;
  stroke(70, 255, 0);
} else
{
  overRect116=false;
  stroke(0, 0, 0);
}
fill(red116, green116, blue116);
rect(xquad06, yquad116, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status116==true)
  {
    valor116=port.read();
    delay(250);
    textFont(f, 12);
    text(valor116, 705, 257);
  }
  if (status116==false)
  {
    textFont(f, 12);
    text(valor116, 705, 257);
  }
}
if (mouseX > xquad06 && mouseX < xquad06+bquad &&
  mouseY > yquad126 && mouseY < yquad126+aquad)
{
  overRect126=true;
  stroke(70, 255, 0);
} else
{
  overRect126=false;
  stroke(0, 0, 0);
}
fill(red126, green126, blue126);
rect(xquad06, yquad126, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
```

```
{
 if (status126==true)
 {
  valor126=port.read();
  delay(250);
  textFont(f, 12);
  text(valor126, 705, 272);
 }
 if (status126==false)
 {
  textFont(f, 12);
  text(valor126, 705, 272);
 }
}
if (mouseX > xquad06 && mouseX < xquad06+bquad &&
 mouseY > yquad136 && mouseY < yquad136+aquad)
{
 overRect136=true;
 stroke(70, 255, 0);
} else
{
 overRect136=false;
 stroke(0, 0, 0);
}
fill(red136, green136, blue136);
rect(xquad06, yquad136, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
 if (status136==true)
 {
  valor136=port.read();
  delay(250);
  textFont(f, 12);
  text(valor136, 705, 287);
 }
 if (status136==false)
 {
  textFont(f, 12);
  text(valor136, 705, 287);
 }
}
if (mouseX > xquad06 && mouseX < xquad06+bquad &&
 mouseY > yquad146 && mouseY < yquad146+aquad)
{
 overRect146=true;
 stroke(70, 255, 0);
} else
{
 overRect146=false;
 stroke(0, 0, 0);
```

```
}
fill(red146, green146, blue146);
rect(xquad06, yquad146, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
 if (status146==true)
 {
  valor146=port.read();
  delay(250);
  textFont(f, 12);
  text(valor146, 705, 302);
 }
 if (status146==false)
 {
  textFont(f, 12);
  text(valor146, 705, 302);
 }
}
if (mouseX > xquad06 && mouseX < xquad06+bquad &&
 mouseY > yquad156 && mouseY < yquad156+aquad)
{
 overRect156=true;
 stroke(70, 255, 0);
} else
{
 overRect156=false;
 stroke(0, 0, 0);
}
fill(red156, green156, blue156);
rect(xquad06, yquad156, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
 if (status156==true)
 {
  valor156=port.read();
  delay(250);
  textFont(f, 12);
  text(valor156, 705, 317);
 }
 if (status156==false)
 {
  textFont(f, 12);
  text(valor156, 705, 317);
 }
}
if (mouseX > xquad06 && mouseX < xquad06+bquad &&
 mouseY > yquad166 && mouseY < yquad166+aquad)
```

```
{
 overRect166=true;
 stroke(70, 255, 0);
} else
{
 overRect166=false;
 stroke(0, 0, 0);
}
fill(red166, green166, blue166);
rect(xquad06, yquad166, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
 if (status166==true)
 {
  valor166=port.read();
  delay(250);
  textFont(f, 12);
  text(valor166, 705, 332);
 }
 if (status166==false)
 {
  textFont(f, 12);
  text(valor166, 705, 332);
 }
}
if (mouseX > xquad06 && mouseX < xquad06+bquad &&
 mouseY > yquad176 && mouseY < yquad176+aquad)
{
 overRect176=true;
 stroke(70, 255, 0);
} else
{
 overRect176=false;
 stroke(0, 0, 0);
}
fill(red176, green176, blue176);
rect(xquad06, yquad176, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
 if (status176==true)
 {
  valor176=port.read();
  delay(250);
  textFont(f, 12);
  text(valor176, 705, 347);
 }
 if (status176==false)
```

```
  {
   textFont(f, 12);
   text(valor176, 705, 347);
  }
}
if (mouseX > xquad06 && mouseX < xquad06+bquad &&
 mouseY > yquad186 && mouseY < yquad186+aquad)
{
 overRect186=true;
 stroke(70, 255, 0);
} else
{
 overRect186=false;
 stroke(0, 0, 0);
}
fill(red186, green186, blue186);
rect(xquad06, yquad186, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
 if (status186==true)
 {
  valor186=port.read();
  delay(250);
  textFont(f, 12);
  text(valor186, 705, 362);
 }
 if (status186==false)
 {
  textFont(f, 12);
  text(valor186, 705, 362);
 }
}
if (mouseX > xquad06 && mouseX < xquad06+bquad &&
 mouseY > yquad196 && mouseY < yquad196+aquad)
{
 overRect196=true;
 stroke(70, 255, 0);
} else
{
 overRect196=false;
 stroke(0, 0, 0);
}
fill(red196, green196, blue196);
rect(xquad06, yquad196, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
 if (status196==true)
```

```
  {
   valor196=port.read();
   delay(250);
   textFont(f, 12);
   text(valor196, 705, 377);
  }
  if (status196==false)
  {
   textFont(f, 12);
   text(valor196, 705, 377);
  }
}
if (mouseX > xquad06 && mouseX < xquad06+bquad &&
  mouseY > yquad206 && mouseY < yquad206+aquad)
{
  overRect206=true;
  stroke(70, 255, 0);
} else
{
  overRect206=false;
  stroke(0, 0, 0);
}
fill(red206, green206, blue206);
rect(xquad06, yquad206, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status206==true)
  {
   valor206=port.read();
   delay(250);
   textFont(f, 12);
   text(valor206, 705, 392);
  }
  if (status206==false)
  {
   textFont(f, 12);
   text(valor206, 705, 392);
  }
}
if (mouseX > xquad06 && mouseX < xquad06+bquad &&
  mouseY > yquad216 && mouseY < yquad216+aquad)
{
  overRect216=true;
  stroke(70, 255, 0);
} else
{
  overRect216=false;
  stroke(0, 0, 0);
}
fill(red216, green216, blue216);
```

```
rect(xquad06, yquad216, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
 if (status216==true)
 {
  valor216=port.read();
  delay(250);
  textFont(f, 12);
  text(valor216, 705, 407);
 }
 if (status216==false)
 {
  textFont(f, 12);
  text(valor216, 705, 407);
 }
}
if (mouseX > xquad06 && mouseX < xquad06+bquad &&
 mouseY > yquad226 && mouseY < yquad226+aquad)
{
 overRect226=true;
 stroke(70, 255, 0);
} else
{
 overRect226=false;
 stroke(0, 0, 0);
}
fill(red226, green226, blue226);
rect(xquad06, yquad226, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
 if (status226==true)
 {
  valor226=port.read();
  delay(250);
  textFont(f, 12);
  text(valor226, 705, 422);
 }
 if (status226==false)
 {
  textFont(f, 12);
  text(valor226, 705, 422);
 }
}
if (mouseX > xquad06 && mouseX < xquad06+bquad &&
 mouseY > yquad236 && mouseY < yquad236+aquad)
{
 overRect236=true;
```

```
  stroke(70, 255, 0);
} else
{
  overRect236=false;
  stroke(0, 0, 0);
}
fill(red236, green236, blue236);
rect(xquad06, yquad236, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status236==true)
  {
    valor236=port.read();
    delay(250);
    textFont(f, 12);
    text(valor236, 705, 437);
  }
  if (status236==false)
  {
    textFont(f, 12);
    text(valor236, 705, 437);
  }
}
if (mouseX > xquad06 && mouseX < xquad06+bquad &&
  mouseY > yquad246 && mouseY < yquad246+aquad)
{
  overRect246=true;
  stroke(70, 255, 0);
} else
{
  overRect246=false;
  stroke(0, 0, 0);
}
fill(red246, green246, blue246);
rect(xquad06, yquad246, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status246==true)
  {
    valor246=port.read();
    delay(250);
    textFont(f, 12);
    text(valor246, 705, 452);
  }
  if (status246==false)
  {
    textFont(f, 12);
```

```
   text(valor246, 705, 452);
  }
}
if (mouseX > xquad06 && mouseX < xquad06+bquad &&
  mouseY > yquad256 && mouseY < yquad256+aquad)
{
  overRect256=true;
  stroke(70, 255, 0);
} else
{
  overRect256=false;
  stroke(0, 0, 0);
}
fill(red256, green256, blue256);
rect(xquad06, yquad256, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status256==true)
  {
    valor256=port.read();
    delay(250);
    textFont(f, 12);
    text(valor256, 705, 467);
  }
  if (status256==false)
  {
    textFont(f, 12);
    text(valor256, 705, 467);
  }
}
if (mouseX > xquad06 && mouseX < xquad06+bquad &&
  mouseY > yquad266 && mouseY < yquad266+aquad)
{
  overRect266=true;
  stroke(70, 255, 0);
} else
{
  overRect266=false;
  stroke(0, 0, 0);
}
fill(red266, green266, blue266);
rect(xquad06, yquad266, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status266==true)
  {
    valor266=port.read();
```

```
   delay(250);
   textFont(f, 12);
   text(valor266, 705, 482);
  }
  if (status266==false)
  {
   textFont(f, 12);
   text(valor266, 705, 482);
  }
}
if (mouseX > xquad06 && mouseX < xquad06+bquad &&
  mouseY > yquad276 && mouseY < yquad276+aquad)
{
  overRect276=true;
  stroke(70, 255, 0);
} else
{
  overRect276=false;
  stroke(0, 0, 0);
}
fill(red276, green276, blue276);
rect(xquad06, yquad276, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
  if (status276==true)
  {
   valor276=port.read();
   delay(250);
   textFont(f, 12);
   text(valor276, 705, 497);
  }
  if (status276==false)
  {
   textFont(f, 12);
   text(valor276, 705, 497);
  }
}
if (mouseX > xquad06 && mouseX < xquad06+bquad &&
  mouseY > yquad286 && mouseY < yquad286+aquad)
{
  overRect286=true;
  stroke(70, 255, 0);
} else
{
  overRect286=false;
  stroke(0, 0, 0);
}
fill(red286, green286, blue286);
rect(xquad06, yquad286, bquad, aquad);
fill(0, 0, 0);
```

```
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
 if (status286==true)
 {
  valor286=port.read();
  delay(250);
  textFont(f, 12);
  text(valor286, 705, 512);
 }
 if (status286==false)
 {
  textFont(f, 12);
  text(valor286, 705, 512);
 }
}
if (mouseX > xquad06 && mouseX < xquad06+bquad &&
 mouseY > yquad296 && mouseY < yquad296+aquad)
{
 overRect296=true;
 stroke(70, 255, 0);
} else
{
 overRect296=false;
 stroke(0, 0, 0);
}
fill(red296, green296, blue296);
rect(xquad06, yquad296, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
 if (status296==true)
 {
  valor296=port.read();
  delay(250);
  textFont(f, 12);
  text(valor296, 705, 527);
 }
 if (status296==false)
 {
  textFont(f, 12);
  text(valor296, 705, 527);
 }
}
if (mouseX > xquad06 && mouseX < xquad06+bquad &&
 mouseY > yquad306 && mouseY < yquad306+aquad)
{
 overRect306=true;
 stroke(70, 255, 0);
} else
```

```
{
 overRect306=false;
 stroke(0, 0, 0);
}
fill(red306, green306, blue306);
rect(xquad06, yquad306, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
 if (status306==true)
 {
  valor306=port.read();
  delay(250);
  textFont(f, 12);
  text(valor306, 705, 542);
 }
 if (status306==false)
 {
  textFont(f, 12);
  text(valor306, 705, 542);
 }
}
if (mouseX > xquad06 && mouseX < xquad06+bquad &&
 mouseY > yquad316 && mouseY < yquad316+aquad)
{
 overRect316=true;
 stroke(70, 255, 0);
} else
{
 overRect316=false;
 stroke(0, 0, 0);
}
fill(red316, green316, blue316);
rect(xquad06, yquad316, bquad, aquad);
fill(0, 0, 0);
textFont(b, 18);
text(texto, 250, 600);
if (port.available()>0)
{
 if (status316==true)
 {
  valor316=port.read();
  delay(250);
  textFont(f, 12);
  text(valor316, 705, 557);
 }
 if (status316==false)
 {
  textFont(f, 12);
  text(valor316, 705, 557);
 }
```

```
  }
}

void mousePressed() {
  for (TEXTBOX t : textboxes) {
    t.PRESSED(mouseX, mouseY);
  }
  //// DATOS PUERTO SERIE STRUT 1/////
  if (overRect01==true)
  {
    port.clear();
    status01=!status01;
    port.write("A");
    if (status01==true)
    {
      //Color del boton verde
      red01=120;
      green01=247;
      blue01=208;
      texto="Medición\nStrut 1";
    }
    if (status01==false)
    {
      //Color del boton gris
      red01=224;
      green01=224;
      blue01=224;
      texto="No está realizando\nmedición";
    }
  }
  if (overRect11==true)
  {
    port.clear();
    status11=!status11;
    port.write("A");
    if (status11==true)
    {
      //Color del boton verde
      red11=120;
      green11=247;
      blue11=208;
      texto="Medición\nStrut 1";
    }
    if (status11==false)
    {
      //Color del boton gris
      red11=224;
      green11=224;
      blue11=224;
      texto="No está realizando\nmedición";
    }
  }
  if (overRect21==true)
```

```
{
  port.clear();
  status21=!status21;
  port.write("A");
  if (status21==true)
  {
    //Color del boton verde
    red21=120;
    green21=247;
    blue21=208;
    texto="Medición\nStrut 1";
  }
  if (status21==false)
  {
    //Color del boton gris
    red21=224;
    green21=224;
    blue21=224;
    texto="No está realizando\nmedición";
  }
}
if (overRect31==true)
{
  port.clear();
  status31=!status31;
  port.write("A");
  if (status31==true)
  {
    //Color del boton verde
    red31=120;
    green31=247;
    blue31=208;
    texto="Medición\nStrut 1";
  }
  if (status31==false)
  {
    //Color del boton gris
    red31=224;
    green31=224;
    blue31=224;
    texto="No está realizando\nmedición";
  }
}
if (overRect41==true)
{
  port.clear();
  status41=!status41;
  port.write("A");
  if (status41==true)
  {
    //Color del boton verde
    red41=120;
    green41=247;
```

```
    blue41=208;
    texto="Medición\nStrut 1";
   }
   if (status41==false)
   {
    //Color del boton gris
    red41=224;
    green41=224;
    blue41=224;
    texto="No está realizando\nmedición";
   }
  }
  if (overRect51==true)
  {
   port.clear();
   status51=!status51;
   port.write("A");
   if (status51==true)
   {
    //Color del boton verde
    red51=120;
    green51=247;
    blue51=208;
    texto="Medición\nStrut 1";
   }
   if (status51==false)
   {
    //Color del boton gris
    red51=224;
    green51=224;
    blue51=224;
    texto="No está realizando\nmedición";
   }
  }
  if (overRect61==true)
  {
   port.clear();
   status61=!status61;
   port.write("A");
   if (status61==true)
   {
    //Color del boton verde
    red61=120;
    green61=247;
    blue61=208;
    texto="Medición\nStrut 1";
   }
   if (status61==false)
   {
    //Color del boton gris
    red61=224;
    green61=224;
    blue61=224;
```

```
    texto="No está realizando\nmedición";
   }
  }
  if (overRect71==true)
  {
   port.clear();
   status71=!status71;
   port.write("A");
   if (status71==true)
   {
    //Color del boton verde
    red71=120;
    green71=247;
    blue71=208;
    texto="Medición\nStrut 1";
   }
   if (status71==false)
   {
    //Color del boton gris
    red71=224;
    green71=224;
    blue71=224;
    texto="No está realizando\nmedición";
   }
  }
  if (overRect81==true)
  {
   port.clear();
   status81=!status81;
   port.write("A");
   if (status81==true)
   {
    //Color del boton verde
    red81=120;
    green81=247;
    blue81=208;
    texto="Medición\nStrut 1";
   }
   if (status81==false)
   {
    //Color del boton gris
    red81=224;
    green81=224;
    blue81=224;
    texto="No está realizando\nmedición";
   }
  }
  if (overRect91==true)
  {
   port.clear();
   status91=!status91;
   port.write("A");
   if (status91==true)
```

```
  {
   //Color del boton verde
   red91=120;
   green91=247;
   blue91=208;
   texto="Medición\nStrut 1";
  }
  if (status91==false)
  {
   //Color del boton gris
   red91=224;
   green91=224;
   blue91=224;
   texto="No está realizando\nmedición";
  }
 }
 if (overRect101==true)
 {
  port.clear();
  status101=!status101;
  port.write("A");
  if (status101==true)
  {
   //Color del boton verde
   red101=120;
   green101=247;
   blue101=208;
   texto="Medición\nStrut 1";
  }
  if (status101==false)
  {
   //Color del boton gris
   red101=224;
   green101=224;
   blue101=224;
   texto="No está realizando\nmedición";
  }
 }
 if (overRect111==true)
 {
  port.clear();
  status111=!status111;
  port.write("A");
  if (status111==true)
  {
   //Color del boton verde
   red111=120;
   green111=247;
   blue111=208;
   texto="Medición\nStrut 1";
  }
  if (status111==false)
  {
```

```
    //Color del boton gris
    red111=224;
    green111=224;
    blue111=224;
    texto="No está realizando\nmedición";
   }
}
if (overRect121==true)
{
  port.clear();
  status121=!status121;
  port.write("A");
  if (status121==true)
  {
    //Color del boton verde
    red121=120;
    green121=247;
    blue121=208;
    texto="Medición\nStrut 1";
  }
  if (status121==false)
  {
    //Color del boton gris
    red121=224;
    green121=224;
    blue121=224;
    texto="No está realizando\nmedición";
  }
}
if (overRect131==true)
{
  port.clear();
  status131=!status131;
  port.write("A");
  if (status131==true)
  {
    //Color del boton verde
    red131=120;
    green131=247;
    blue131=208;
    texto="Medición\nStrut 1";
  }
  if (status131==false)
  {
    //Color del boton gris
    red131=224;
    green131=224;
    blue131=224;
    texto="No está realizando\nmedición";
  }
}
if (overRect141==true)
{
```

```
  port.clear();
  status141=!status141;
  port.write("A");
  if (status141==true)
  {
    //Color del boton verde
    red141=120;
    green141=247;
    blue141=208;
    texto="Medición\nStrut 1";
  }
  if (status141==false)
  {
    //Color del boton gris
    red141=224;
    green141=224;
    blue141=224;
    texto="No está realizando\nmedición";
  }
}
if (overRect151==true)
{
  port.clear();
  status151=!status151;
  port.write("A");
  if (status151==true)
  {
    //Color del boton verde
    red151=120;
    green151=247;
    blue151=208;
    texto="Medición\nStrut 1";
  }
  if (status151==false)
  {
    //Color del boton gris
    red151=224;
    green151=224;
    blue151=224;
    texto="No está realizando\nmedición";
  }
}
if (overRect161==true)
{
  port.clear();
  status161=!status161;
  port.write("A");
  if (status161==true)
  {
    //Color del boton verde
    red161=120;
    green161=247;
    blue161=208;
```

```
    texto="Medición\nStrut 1";
   }
   if (status161==false)
   {
    //Color del boton gris
    red161=224;
    green161=224;
    blue161=224;
    texto="No está realizando\nmedición";
   }
  }
 if (overRect171==true)
 {
  port.clear();
  status171=!status171;
  port.write("A");
  if (status171==true)
  {
   //Color del boton verde
   red171=120;
   green171=247;
   blue171=208;
   texto="Medición\nStrut 1";
  }
  if (status171==false)
  {
   //Color del boton gris
   red171=224;
   green171=224;
   blue171=224;
   texto="No está realizando\nmedición";
  }
 }
 if (overRect181==true)
 {
  port.clear();
  status181=!status181;
  port.write("A");
  if (status181==true)
  {
   //Color del boton verde
   red181=120;
   green181=247;
   blue181=208;
   texto="Medición\nStrut 1";
  }
  if (status181==false)
  {
   //Color del boton gris
   red181=224;
   green181=224;
   blue181=224;
   texto="No está realizando\nmedición";
```

```
  }
 }
 if (overRect191==true)
 {
  port.clear();
  status191=!status191;
  port.write("A");
  if (status191==true)
  {
   //Color del boton verde
   red191=120;
   green191=247;
   blue191=208;
   texto="Medición\nStrut 1";
  }
  if (status191==false)
  {
   //Color del boton gris
   red191=224;
   green191=224;
   blue191=224;
   texto="No está realizando\nmedición";
  }
 }
 if (overRect201==true)
 {
  port.clear();
  status201=!status201;
  port.write("A");
  if (status201==true)
  {
   //Color del boton verde
   red201=120;
   green201=247;
   blue201=208;
   texto="Medición\nStrut 1";
  }
  if (status201==false)
  {
   //Color del boton gris
   red201=224;
   green201=224;
   blue201=224;
   texto="No está realizando\nmedición";
  }
 }
 if (overRect211==true)
 {
  port.clear();
  status211=!status211;
  port.write("A");
  if (status211==true)
  {
```

```
    //Color del boton verde
    red211=120;
    green211=247;
    blue211=208;
    texto="Medición\nStrut 1";
   }
   if (status211==false)
   {
    //Color del boton gris
    red211=224;
    green211=224;
    blue211=224;
    texto="No está realizando\nmedición";
   }
  }
  if (overRect221==true)
  {
   port.clear();
   status221=!status221;
   port.write("A");
   if (status221==true)
   {
    //Color del boton verde
    red221=120;
    green221=247;
    blue221=208;
    texto="Medición\nStrut 1";
   }
   if (status221==false)
   {
    //Color del boton gris
    red221=224;
    green221=224;
    blue221=224;
    texto="No está realizando\nmedición";
   }
  }
  if (overRect231==true)
  {
   port.clear();
   status231=!status231;
   port.write("A");
   if (status231==true)
   {
    //Color del boton verde
    red231=120;
    green231=247;
    blue231=208;
    texto="Medición\nStrut 1";
   }
   if (status231==false)
   {
    //Color del boton gris
```

```
    red231=224;
    green231=224;
    blue231=224;
    texto="No está realizando\nmedición";
   }
  }
  if (overRect241==true)
  {
   port.clear();
   status241=!status241;
   port.write("A");
   if (status241==true)
   {
    //Color del boton verde
    red241=120;
    green241=247;
    blue241=208;
    texto="Medición\nStrut 1";
   }
   if (status241==false)
   {
    //Color del boton gris
    red241=224;
    green241=224;
    blue241=224;
    texto="No está realizando\nmedición";
   }
  }
  if (overRect251==true)
  {
   port.clear();
   status251=!status251;
   port.write("A");
   if (status251==true)
   {
    //Color del boton verde
    red251=120;
    green251=247;
    blue251=208;
    texto="Medición\nStrut 1";
   }
   if (status251==false)
   {
    //Color del boton gris
    red251=224;
    green251=224;
    blue251=224;
    texto="No está realizando\nmedición";
   }
  }
  if (overRect261==true)
  {
   port.clear();
```

```
  status261=!status261;
  port.write("A");
  if (status261==true)
  {
    //Color del boton verde
    red261=120;
    green261=247;
    blue261=208;
    texto="Medición\nStrut 1";
  }
  if (status261==false)
  {
    //Color del boton gris
    red261=224;
    green261=224;
    blue261=224;
    texto="No está realizando\nmedición";
  }
}
if (overRect271==true)
{
  port.clear();
  status271=!status271;
  port.write("A");
  if (status271==true)
  {
    //Color del boton verde
    red271=120;
    green271=247;
    blue271=208;
    texto="Medición\nStrut 1";
  }
  if (status271==false)
  {
    //Color del boton gris
    red271=224;
    green271=224;
    blue271=224;
    texto="No está realizando\nmedición";
  }
}
if (overRect281==true)
{
  port.clear();
  status281=!status281;
  port.write("A");
  if (status281==true)
  {
    //Color del boton verde
    red281=120;
    green281=247;
    blue281=208;
    texto="Medición\nStrut 1";
```

```
  }
  if (status281==false)
  {
    //Color del boton gris
    red281=224;
    green281=224;
    blue281=224;
    texto="No está realizando\nmedición";
  }
}
if (overRect291==true)
{
  port.clear();
  status291=!status291;
  port.write("A");
  if (status291==true)
  {
    //Color del boton verde
    red291=120;
    green291=247;
    blue291=208;
    texto="Medición\nStrut 1";
  }
  if (status291==false)
  {
    //Color del boton gris
    red291=224;
    green291=224;
    blue291=224;
    texto="No está realizando\nmedición";
  }
}
if (overRect301==true)
{
  port.clear();
  status301=!status301;
  port.write("A");
  if (status301==true)
  {
    //Color del boton verde
    red301=120;
    green301=247;
    blue301=208;
    texto="Medición\nStrut 1";
  }
  if (status301==false)
  {
    //Color del boton gris
    red301=224;
    green301=224;
    blue301=224;
    texto="No está realizando\nmedición";
  }
```

```
}
if (overRect311==true)
{
  port.clear();
  status311=!status311;
  port.write("A");
  if (status311==true)
  {
    //Color del boton verde
    red311=120;
    green311=247;
    blue311=208;
    texto="Medición\nStrut 1";
  }
  if (status311==false)
  {
    //Color del boton gris
    red311=224;
    green311=224;
    blue311=224;
    texto="No está realizando\nmedición";
  }
}
//// DATOS PUERTO SERIE STRUT 2/////
if (overRect02==true)
{
  port.clear();
  status02=!status02;
  port.write("B");
  if (status02==true)
  {
    //Color del boton verde
    red02=120;
    green02=247;
    blue02=208;
    texto="Medición\nStrut 2";
  }
  if (status02==false)
  {
    //Color del boton gris
    red02=224;
    green02=224;
    blue02=224;
    texto="No está realizando\nmedición";
  }
}
if (overRect12==true)
{
  port.clear();
  status12=!status12;
  port.write("B");
  if (status12==true)
  {
```

```
    //Color del boton verde
    red12=120;
    green12=247;
    blue12=208;
    texto="Medición\nStrut 2";
   }
   if (status12==false)
   {
    //Color del boton gris
    red12=224;
    green12=224;
    blue12=224;
    texto="No está realizando\nmedición";
   }
  }
  if (overRect22==true)
  {
   port.clear();
   status22=!status22;
   port.write("B");
   if (status22==true)
   {
    //Color del boton verde
    red22=120;
    green22=247;
    blue22=208;
    texto="Medición\nStrut 2";
   }
   if (status22==false)
   {
    //Color del boton gris
    red22=224;
    green22=224;
    blue22=224;
    texto="No está realizando\nmedición";
   }
  }
  if (overRect32==true)
  {
   port.clear();
   status32=!status32;
   port.write("B");
   if (status32==true)
   {
    //Color del boton verde
    red32=120;
    green32=247;
    blue32=208;
    texto="Medición\nStrut 2";
   }
   if (status32==false)
   {
    //Color del boton gris
```

```
    red32=224;
    green32=224;
    blue32=224;
    texto="No está realizando\nmedición";
   }
  }
  if (overRect42==true)
  {
   port.clear();
   status42=!status42;
   port.write("B");
   if (status42==true)
   {
    //Color del boton verde
    red42=120;
    green42=247;
    blue42=208;
    texto="Medición\nStrut 2";
   }
   if (status42==false)
   {
    //Color del boton gris
    red42=224;
    green42=224;
    blue42=224;
    texto="No está realizando\nmedición";
   }
  }
  if (overRect52==true)
  {
   port.clear();
   status52=!status52;
   port.write("B");
   if (status52==true)
   {
    //Color del boton verde
    red52=120;
    green52=247;
    blue52=208;
    texto="Medición\nStrut 2";
   }
   if (status52==false)
   {
    //Color del boton gris
    red52=224;
    green52=224;
    blue52=224;
    texto="No está realizando\nmedición";
   }
  }
  if (overRect62==true)
  {
   port.clear();
```

```
  status62=!status62;
  port.write("B");
  if (status62==true)
  {
    //Color del boton verde
    red62=120;
    green62=247;
    blue62=208;
    texto="Medición\nStrut 2";
  }
  if (status62==false)
  {
    //Color del boton gris
    red62=224;
    green62=224;
    blue62=224;
    texto="No está realizando\nmedición";
  }
 }
 if (overRect72==true)
 {
  port.clear();
  status72=!status72;
  port.write("B");
  if (status72==true)
  {
    //Color del boton verde
    red72=120;
    green72=247;
    blue72=208;
    texto="Medición\nStrut 2";
  }
  if (status72==false)
  {
    //Color del boton gris
    red72=224;
    green72=224;
    blue72=224;
    texto="No está realizando\nmedición";
  }
 }
 if (overRect82==true)
 {
  port.clear();
  status82=!status82;
  port.write("B");
  if (status82==true)
  {
    //Color del boton verde
    red82=120;
    green82=247;
    blue82=208;
    texto="Medición\nStrut 2";
```

```
  }
  if (status82==false)
  {
    //Color del boton gris
    red82=224;
    green82=224;
    blue82=224;
    texto="No está realizando\nmedición";
  }
}
if (overRect92==true)
{
  port.clear();
  status92=!status92;
  port.write("B");
  if (status92==true)
  {
    //Color del boton verde
    red92=120;
    green92=247;
    blue92=208;
    texto="Medición\nStrut 2";
  }
  if (status92==false)
  {
    //Color del boton gris
    red92=224;
    green92=224;
    blue92=224;
    texto="No está realizando\nmedición";
  }
}
if (overRect102==true)
{
  port.clear();
  status102=!status102;
  port.write("B");
  if (status102==true)
  {
    //Color del boton verde
    red102=120;
    green102=247;
    blue102=208;
    texto="Medición\nStrut 2";
  }
  if (status102==false)
  {
    //Color del boton gris
    red102=224;
    green102=224;
    blue102=224;
    texto="No está realizando\nmedición";
  }
```

```
}
if (overRect112==true)
{
  port.clear();
  status112=!status112;
  port.write("B");
  if (status112==true)
  {
    //Color del boton verde
    red112=120;
    green112=247;
    blue112=208;
    texto="Medición\nStrut 2";
  }
  if (status112==false)
  {
    //Color del boton gris
    red112=224;
    green112=224;
    blue112=224;
    texto="No está realizando\nmedición";
  }
}
if (overRect122==true)
{
  port.clear();
  status122=!status122;
  port.write("B");
  if (status122==true)
  {
    //Color del boton verde
    red122=120;
    green122=247;
    blue122=208;
    texto="Medición\nStrut 2";
  }
  if (status122==false)
  {
    //Color del boton gris
    red122=224;
    green122=224;
    blue122=224;
    texto="No está realizando\nmedición";
  }
}
if (overRect132==true)
{
  port.clear();
  status132=!status132;
  port.write("B");
  if (status132==true)
  {
    //Color del boton verde
```

```
    red132=120;
    green132=247;
    blue132=208;
    texto="Medición\nStrut 2";
   }
   if (status132==false)
   {
    //Color del boton gris
    red132=224;
    green132=224;
    blue132=224;
    texto="No está realizando\nmedición";
   }
  }
  if (overRect142==true)
  {
   port.clear();
   status142=!status142;
   port.write("B");
   if (status142==true)
   {
    //Color del boton verde
    red142=120;
    green142=247;
    blue142=208;
    texto="Medición\nStrut 2";
   }
   if (status142==false)
   {
    //Color del boton gris
    red142=224;
    green142=224;
    blue142=224;
    texto="No está realizando\nmedición";
   }
  }
  if (overRect152==true)
  {
   port.clear();
   status152=!status152;
   port.write("B");
   if (status152==true)
   {
    //Color del boton verde
    red152=120;
    green152=247;
    blue152=208;
    texto="Medición\nStrut 2";
   }
   if (status152==false)
   {
    //Color del boton gris
    red152=224;
```

```
    green152=224;
    blue152=224;
    texto="No está realizando\nmedición";
   }
  }
  if (overRect162==true)
  {
   port.clear();
   status162=!status162;
   port.write("B");
   if (status162==true)
   {
    //Color del boton verde
    red162=120;
    green162=247;
    blue162=208;
    texto="Medición\nStrut 2";
   }
   if (status162==false)
   {
    //Color del boton gris
    red162=224;
    green162=224;
    blue162=224;
    texto="No está realizando\nmedición";
   }
  }
  if (overRect172==true)
  {
   port.clear();
   status172=!status172;
   port.write("B");
   if (status172==true)
   {
    //Color del boton verde
    red172=120;
    green172=247;
    blue172=208;
    texto="Medición\nStrut 2";
   }
   if (status172==false)
   {
    //Color del boton gris
    red172=224;
    green172=224;
    blue172=224;
    texto="No está realizando\nmedición";
   }
  }
  if (overRect182==true)
  {
   port.clear();
   status182=!status182;
```

```
  port.write("B");
  if (status182==true)
  {
    //Color del boton verde
    red182=120;
    green182=247;
    blue182=208;
    texto="Medición\nStrut 2";
  }
  if (status182==false)
  {
    //Color del boton gris
    red182=224;
    green182=224;
    blue182=224;
    texto="No está realizando\nmedición";
  }
}
if (overRect192==true)
{
  port.clear();
  status192=!status192;
  port.write("B");
  if (status192==true)
  {
    //Color del boton verde
    red192=120;
    green192=247;
    blue192=208;
    texto="Medición\nStrut 2";
  }
  if (status192==false)
  {
    //Color del boton gris
    red192=224;
    green192=224;
    blue192=224;
    texto="No está realizando\nmedición";
  }
}
if (overRect202==true)
{
  port.clear();
  status202=!status202;
  port.write("B");
  if (status202==true)
  {
    //Color del boton verde
    red202=120;
    green202=247;
    blue202=208;
    texto="Medición\nStrut 2";
  }
```

```
  if (status202==false)
  {
    //Color del boton gris
    red202=224;
    green202=224;
    blue202=224;
    texto="No está realizando\nmedición";
  }
}
if (overRect212==true)
{
  port.clear();
  status212=!status212;
  port.write("B");
  if (status212==true)
  {
    //Color del boton verde
    red212=120;
    green212=247;
    blue212=208;
    texto="Medición\nStrut 2";
  }
  if (status212==false)
  {
    //Color del boton gris
    red212=224;
    green212=224;
    blue212=224;
    texto="No está realizando\nmedición";
  }
}
if (overRect222==true)
{
  port.clear();
  status222=!status222;
  port.write("B");
  if (status222==true)
  {
    //Color del boton verde
    red222=120;
    green222=247;
    blue222=208;
    texto="Medición\nStrut 2";
  }
  if (status222==false)
  {
    //Color del boton gris
    red222=224;
    green222=224;
    blue222=224;
    texto="No está realizando\nmedición";
  }
}
```

```
if (overRect232==true)
{
  port.clear();
  status232=!status232;
  port.write("B");
  if (status232==true)
  {
    //Color del boton verde
    red232=120;
    green232=247;
    blue232=208;
    texto="Medición\nStrut 2";
  }
  if (status232==false)
  {
    //Color del boton gris
    red232=224;
    green232=224;
    blue232=224;
    texto="No está realizando\nmedición";
  }
}
if (overRect242==true)
{
  port.clear();
  status242=!status242;
  port.write("B");
  if (status242==true)
  {
    //Color del boton verde
    red242=120;
    green242=247;
    blue242=208;
    texto="Medición\nStrut 2";
  }
  if (status242==false)
  {
    //Color del boton gris
    red242=224;
    green242=224;
    blue242=224;
    texto="No está realizando\nmedición";
  }
}
if (overRect252==true)
{
  port.clear();
  status252=!status252;
  port.write("B");
  if (status252==true)
  {
    //Color del boton verde
    red252=120;
```

```
    green252=247;
    blue252=208;
    texto="Medición\nStrut 2";
   }
   if (status252==false)
   {
    //Color del boton gris
    red252=224;
    green252=224;
    blue252=224;
    texto="No está realizando\nmedición";
   }
  }
  if (overRect262==true)
  {
   port.clear();
   status262=!status262;
   port.write("B");
   if (status262==true)
   {
    //Color del boton verde
    red262=120;
    green262=247;
    blue262=208;
    texto="Medición\nStrut 2";
   }
   if (status262==false)
   {
    //Color del boton gris
    red262=224;
    green262=224;
    blue262=224;
    texto="No está realizando\nmedición";
   }
  }
  if (overRect272==true)
  {
   port.clear();
   status272=!status272;
   port.write("B");
   if (status272==true)
   {
    //Color del boton verde
    red272=120;
    green272=247;
    blue272=208;
    texto="Medición\nStrut 2";
   }
   if (status272==false)
   {
    //Color del boton gris
    red272=224;
    green272=224;
```

```
    blue272=224;
    texto="No está realizando\nmedición";
   }
  }
 if (overRect282==true)
 {
  port.clear();
  status282=!status282;
  port.write("B");
  if (status282==true)
  {
    //Color del boton verde
    red282=120;
    green282=247;
    blue282=208;
    texto="Medición\nStrut 2";
  }
  if (status282==false)
  {
    //Color del boton gris
    red282=224;
    green282=224;
    blue282=224;
    texto="No está realizando\nmedición";
  }
 }
 if (overRect292==true)
 {
  port.clear();
  status292=!status292;
  port.write("B");
  if (status292==true)
  {
    //Color del boton verde
    red292=120;
    green292=247;
    blue292=208;
    texto="Medición\nStrut 2";
  }
  if (status292==false)
  {
    //Color del boton gris
    red292=224;
    green292=224;
    blue292=224;
    texto="No está realizando\nmedición";
  }
 }
 if (overRect302==true)
 {
  port.clear();
  status302=!status302;
  port.write("B");
```

```
 if (status302==true)
 {
  //Color del boton verde
  red302=120;
  green302=247;
  blue302=208;
  texto="Medición\nStrut 2";
 }
 if (status302==false)
 {
  //Color del boton gris
  red302=224;
  green302=224;
  blue302=224;
  texto="No está realizando\nmedición";
 }
}
if (overRect312==true)
{
 port.clear();
 status312=!status312;
 port.write("B");
 if (status312==true)
 {
  //Color del boton verde
  red312=120;
  green312=247;
  blue312=208;
  texto="Medición\nStrut 2";
 }
 if (status312==false)
 {
  //Color del boton gris
  red312=224;
  green312=224;
  blue312=224;
  texto="No está realizando\nmedición";
 }
}
//// DATOS PUERTO SERIE STRUT 3/////
if (overRect03==true)
{
 port.clear();
 status03=!status03;
 port.write("C");
 if (status03==true)
 {
  //Color del boton verde
  red03=120;
  green03=247;
  blue03=208;
  texto="Medición\nStrut 3";
 }
```

```
  if (status03==false)
  {
    //Color del boton gris
    red03=224;
    green03=224;
    blue03=224;
    texto="No está realizando\nmedición";
  }
}
if (overRect13==true)
{
  port.clear();
  status13=!status13;
  port.write("C");
  if (status13==true)
  {
    //Color del boton verde
    red13=120;
    green13=247;
    blue13=208;
    texto="Medición\nStrut 3";
  }
  if (status13==false)
  {
    //Color del boton gris
    red13=224;
    green13=224;
    blue13=224;
    texto="No está realizando\nmedición";
  }
}
if (overRect23==true)
{
  port.clear();
  status23=!status23;
  port.write("C");
  if (status23==true)
  {
    //Color del boton verde
    red23=120;
    green23=247;
    blue23=208;
    texto="Medición\nStrut 3";
  }
  if (status23==false)
  {
    //Color del boton gris
    red23=224;
    green23=224;
    blue23=224;
    texto="No está realizando\nmedición";
  }
}
```

```
if (overRect33==true)
{
 port.clear();
 status33=!status33;
 port.write("C");
 if (status33==true)
 {
  //Color del boton verde
  red33=120;
  green33=247;
  blue33=208;
  texto="Medición\nStrut 3";
 }
 if (status33==false)
 {
  //Color del boton gris
  red33=224;
  green33=224;
  blue33=224;
  texto="No está realizando\nmedición";
 }
}
if (overRect43==true)
{
 port.clear();
 status43=!status43;
 port.write("C");
 if (status43==true)
 {
  //Color del boton verde
  red43=120;
  green43=247;
  blue43=208;
  texto="Medición\nStrut 3";
 }
 if (status43==false)
 {
  //Color del boton gris
  red43=224;
  green43=224;
  blue43=224;
  texto="No está realizando\nmedición";
 }
}
if (overRect53==true)
{
 port.clear();
 status53=!status53;
 port.write("C");
 if (status53==true)
 {
  //Color del boton verde
  red53=120;
```

```
    green53=247;
    blue53=208;
    texto="Medición\nStrut 3";
   }
   if (status53==false)
   {
    //Color del boton gris
    red53=224;
    green53=224;
    blue53=224;
    texto="No está realizando\nmedición";
   }
  }
  if (overRect63==true)
  {
   port.clear();
   status63=!status63;
   port.write("C");
   if (status63==true)
   {
    //Color del boton verde
    red63=120;
    green63=247;
    blue63=208;
    texto="Medición\nStrut 3";
   }
   if (status63==false)
   {
    //Color del boton gris
    red63=224;
    green63=224;
    blue63=224;
    texto="No está realizando\nmedición";
   }
  }
  if (overRect73==true)
  {
   port.clear();
   status73=!status73;
   port.write("C");
   if (status73==true)
   {
    //Color del boton verde
    red73=120;
    green73=247;
    blue73=208;
    texto="Medición\nStrut 3";
   }
   if (status73==false)
   {
    //Color del boton gris
    red73=224;
    green73=224;
```

```
    blue73=224;
    texto="No está realizando\nmedición";
   }
  }
 if (overRect83==true)
 {
  port.clear();
  status83=!status83;
  port.write("C");
  if (status83==true)
  {
    //Color del boton verde
    red83=120;
    green83=247;
    blue83=208;
    texto="Medición\nStrut 3";
  }
  if (status83==false)
  {
    //Color del boton gris
    red83=224;
    green83=224;
    blue83=224;
    texto="No está realizando\nmedición";
  }
 }
 if (overRect93==true)
 {
  port.clear();
  status93=!status93;
  port.write("C");
  if (status93==true)
  {
    //Color del boton verde
    red93=120;
    green93=247;
    blue93=208;
    texto="Medición\nStrut 3";
  }
  if (status93==false)
  {
    //Color del boton gris
    red93=224;
    green93=224;
    blue93=224;
    texto="No está realizando\nmedición";
  }
 }
 if (overRect103==true)
 {
  port.clear();
  status103=!status103;
  port.write("C");
```

```
  if (status103==true)
  {
    //Color del boton verde
    red103=120;
    green103=247;
    blue103=208;
    texto="Medición\nStrut 3";
  }
  if (status103==false)
  {
    //Color del boton gris
    red103=224;
    green103=224;
    blue103=224;
    texto="No está realizando\nmedición";
  }
}
if (overRect113==true)
{
  port.clear();
  status113=!status113;
  port.write("C");
  if (status113==true)
  {
    //Color del boton verde
    red113=120;
    green113=247;
    blue113=208;
    texto="Medición\nStrut 3";
  }
  if (status113==false)
  {
    //Color del boton gris
    red113=224;
    green113=224;
    blue113=224;
    texto="No está realizando\nmedición";
  }
}
if (overRect123==true)
{
  port.clear();
  status123=!status123;
  port.write("C");
  if (status123==true)
  {
    //Color del boton verde
    red123=120;
    green123=247;
    blue123=208;
    texto="Medición\nStrut 3";
  }
  if (status123==false)
```

```
  {
    //Color del boton gris
    red123=224;
    green123=224;
    blue123=224;
    texto="No está realizando\nmedición";
  }
}
if (overRect133==true)
{
  port.clear();
  status133=!status133;
  port.write("C");
  if (status133==true)
  {
    //Color del boton verde
    red133=120;
    green133=247;
    blue133=208;
    texto="Medición\nStrut 3";
  }
  if (status133==false)
  {
    //Color del boton gris
    red133=224;
    green133=224;
    blue133=224;
    texto="No está realizando\nmedición";
  }
}
if (overRect143==true)
{
  port.clear();
  status143=!status143;
  port.write("C");
  if (status143==true)
  {
    //Color del boton verde
    red143=120;
    green143=247;
    blue143=208;
    texto="Medición\nStrut 3";
  }
  if (status143==false)
  {
    //Color del boton gris
    red143=224;
    green143=224;
    blue143=224;
    texto="No está realizando\nmedición";
  }
}
if (overRect153==true)
```

```
{
  port.clear();
  status153=!status153;
  port.write("C");
  if (status153==true)
  {
    //Color del boton verde
    red153=120;
    green153=247;
    blue153=208;
    texto="Medición\nStrut 3";
  }
  if (status153==false)
  {
    //Color del boton gris
    red153=224;
    green153=224;
    blue153=224;
    texto="No está realizando\nmedición";
  }
}
if (overRect163==true)
{
  port.clear();
  status163=!status163;
  port.write("C");
  if (status163==true)
  {
    //Color del boton verde
    red163=120;
    green163=247;
    blue163=208;
    texto="Medición\nStrut 3";
  }
  if (status163==false)
  {
    //Color del boton gris
    red163=224;
    green163=224;
    blue163=224;
    texto="No está realizando\nmedición";
  }
}
if (overRect173==true)
{
  port.clear();
  status173=!status173;
  port.write("C");
  if (status173==true)
  {
    //Color del boton verde
    red173=120;
    green173=247;
```

```
    blue173=208;
    texto="Medición\nStrut 3";
   }
   if (status173==false)
   {
    //Color del boton gris
    red173=224;
    green173=224;
    blue173=224;
    texto="No está realizando\nmedición";
   }
  }
  if (overRect183==true)
  {
   port.clear();
   status183=!status183;
   port.write("C");
   if (status183==true)
   {
    //Color del boton verde
    red183=120;
    green183=247;
    blue183=208;
    texto="Medición\nStrut 3";
   }
   if (status183==false)
   {
    //Color del boton gris
    red183=224;
    green183=224;
    blue183=224;
    texto="No está realizando\nmedición";
   }
  }
  if (overRect193==true)
  {
   port.clear();
   status193=!status193;
   port.write("C");
   if (status193==true)
   {
    //Color del boton verde
    red193=120;
    green193=247;
    blue193=208;
    texto="Medición\nStrut 3";
   }
   if (status193==false)
   {
    //Color del boton gris
    red193=224;
    green193=224;
    blue193=224;
```

```
    texto="No está realizando\nmedición";
   }
  }
  if (overRect203==true)
  {
   port.clear();
   status203=!status203;
   port.write("C");
   if (status203==true)
   {
    //Color del boton verde
    red203=120;
    green203=247;
    blue203=208;
    texto="Medición\nStrut 3";
   }
   if (status203==false)
   {
    //Color del boton gris
    red203=224;
    green203=224;
    blue203=224;
    texto="No está realizando\nmedición";
   }
  }
  if (overRect213==true)
  {
   port.clear();
   status213=!status213;
   port.write("C");
   if (status213==true)
   {
    //Color del boton verde
    red213=120;
    green213=247;
    blue213=208;
    texto="Medición\nStrut 3";
   }
   if (status213==false)
   {
    //Color del boton gris
    red213=224;
    green213=224;
    blue213=224;
    texto="No está realizando\nmedición";
   }
  }
  if (overRect223==true)
  {
   port.clear();
   status223=!status223;
   port.write("C");
   if (status223==true)
```

```
  {
   //Color del boton verde
   red223=120;
   green223=247;
   blue223=208;
   texto="Medición\nStrut 3";
  }
  if (status223==false)
  {
   //Color del boton gris
   red223=224;
   green223=224;
   blue223=224;
   texto="No está realizando\nmedición";
  }
 }
 if (overRect233==true)
 {
  port.clear();
  status233=!status233;
  port.write("C");
  if (status233==true)
  {
   //Color del boton verde
   red233=120;
   green233=247;
   blue233=208;
   texto="Medición\nStrut 3";
  }
  if (status233==false)
  {
   //Color del boton gris
   red233=224;
   green233=224;
   blue233=224;
   texto="No está realizando\nmedición";
  }
 }
 if (overRect243==true)
 {
  port.clear();
  status243=!status243;
  port.write("C");
  if (status243==true)
  {
   //Color del boton verde
   red243=120;
   green243=247;
   blue243=208;
   texto="Medición\nStrut 3";
  }
  if (status243==false)
  {
```

```
    //Color del boton gris
    red243=224;
    green243=224;
    blue243=224;
    texto="No está realizando\nmedición";
   }
}
if (overRect253==true)
{
  port.clear();
  status253=!status253;
  port.write("C");
  if (status253==true)
  {
    //Color del boton verde
    red253=120;
    green253=247;
    blue253=208;
    texto="Medición\nStrut 3";
  }
  if (status253==false)
  {
    //Color del boton gris
    red253=224;
    green253=224;
    blue253=224;
    texto="No está realizando\nmedición";
  }
}
if (overRect263==true)
{
  port.clear();
  status263=!status263;
  port.write("C");
  if (status263==true)
  {
    //Color del boton verde
    red263=120;
    green263=247;
    blue263=208;
    texto="Medición\nStrut 3";
  }
  if (status263==false)
  {
    //Color del boton gris
    red263=224;
    green263=224;
    blue263=224;
    texto="No está realizando\nmedición";
  }
}
if (overRect273==true)
{
```

```
  port.clear();
  status273=!status273;
  port.write("C");
  if (status273==true)
  {
    //Color del boton verde
    red273=120;
    green273=247;
    blue273=208;
    texto="Medición\nStrut 3";
  }
  if (status273==false)
  {
    //Color del boton gris
    red273=224;
    green273=224;
    blue273=224;
    texto="No está realizando\nmedición";
  }
}
if (overRect283==true)
{
  port.clear();
  status283=!status283;
  port.write("C");
  if (status283==true)
  {
    //Color del boton verde
    red283=120;
    green283=247;
    blue283=208;
    texto="Medición\nStrut 3";
  }
  if (status283==false)
  {
    //Color del boton gris
    red283=224;
    green283=224;
    blue283=224;
    texto="No está realizando\nmedición";
  }
}
if (overRect293==true)
{
  port.clear();
  status293=!status293;
  port.write("C");
  if (status293==true)
  {
    //Color del boton verde
    red293=120;
    green293=247;
    blue293=208;
```

```
    texto="Medición\nStrut 3";
   }
   if (status293==false)
   {
    //Color del boton gris
    red293=224;
    green293=224;
    blue293=224;
    texto="No está realizando\nmedición";
   }
  }
  if (overRect303==true)
  {
   port.clear();
   status303=!status303;
   port.write("C");
   if (status303==true)
   {
    //Color del boton verde
    red303=120;
    green303=247;
    blue303=208;
    texto="Medición\nStrut 3";
   }
   if (status303==false)
   {
    //Color del boton gris
    red303=224;
    green303=224;
    blue303=224;
    texto="No está realizando\nmedición";
   }
  }
  if (overRect313==true)
  {
   port.clear();
   status313=!status313;
   port.write("C");
   if (status313==true)
   {
    //Color del boton verde
    red313=120;
    green313=247;
    blue313=208;
    texto="Medición\nStrut 3";
   }
   if (status313==false)
   {
    //Color del boton gris
    red313=224;
    green313=224;
    blue313=224;
    texto="No está realizando\nmedición";
```

```
  }
}
//// DATOS PUERTO SERIE STRUT 4/////
if (overRect04==true)
{
  port.clear();
  status04=!status04;
  port.write("D");
  if (status04==true)
  {
    //Color del boton verde
    red04=120;
    green04=247;
    blue04=208;
    texto="Medición\nStrut 4";
  }
  if (status04==false)
  {
    //Color del boton gris
    red04=224;
    green04=224;
    blue04=224;
    texto="No está realizando\nmedición";
  }
}
if (overRect14==true)
{
  port.clear();
  status14=!status14;
  port.write("D");
  if (status14==true)
  {
    //Color del boton verde
    red14=120;
    green14=247;
    blue14=208;
    texto="Medición\nStrut 4";
  }
  if (status14==false)
  {
    //Color del boton gris
    red14=224;
    green14=224;
    blue14=224;
    texto="No está realizando\nmedición";
  }
}
if (overRect24==true)
{
  port.clear();
  status24=!status24;
  port.write("D");
  if (status24==true)
```

```
  {
   //Color del boton verde
   red24=120;
   green24=247;
   blue24=208;
   texto="Medición\nStrut 4";
  }
  if (status24==false)
  {
   //Color del boton gris
   red24=224;
   green24=224;
   blue24=224;
   texto="No está realizando\nmedición";
  }
 }
 if (overRect34==true)
 {
  port.clear();
  status34=!status34;
  port.write("D");
  if (status34==true)
  {
   //Color del boton verde
   red34=120;
   green34=247;
   blue34=208;
   texto="Medición\nStrut 4";
  }
  if (status34==false)
  {
   //Color del boton gris
   red34=224;
   green34=224;
   blue34=224;
   texto="No está realizando\nmedición";
  }
 }
 if (overRect44==true)
 {
  port.clear();
  status44=!status44;
  port.write("D");
  if (status44==true)
  {
   //Color del boton verde
   red44=120;
   green44=247;
   blue44=208;
   texto="Medición\nStrut 4";
  }
  if (status44==false)
  {
```

```
    //Color del boton gris
    red44=224;
    green44=224;
    blue44=224;
    texto="No está realizando\nmedición";
   }
}
if (overRect54==true)
{
  port.clear();
  status54=!status54;
  port.write("D");
  if (status54==true)
  {
    //Color del boton verde
    red54=120;
    green54=247;
    blue54=208;
    texto="Medición\nStrut 4";
  }
  if (status54==false)
  {
    //Color del boton gris
    red54=224;
    green54=224;
    blue54=224;
    texto="No está realizando\nmedición";
  }
}
if (overRect64==true)
{
  port.clear();
  status64=!status64;
  port.write("D");
  if (status64==true)
  {
    //Color del boton verde
    red64=120;
    green64=247;
    blue64=208;
    texto="Medición\nStrut 4";
  }
  if (status64==false)
  {
    //Color del boton gris
    red64=224;
    green64=224;
    blue64=224;
    texto="No está realizando\nmedición";
  }
}
if (overRect74==true)
{
```

```
  port.clear();
  status74=!status74;
  port.write("D");
  if (status74==true)
  {
    //Color del boton verde
    red74=120;
    green74=247;
    blue74=208;
    texto="Medición\nStrut 4";
  }
  if (status74==false)
  {
    //Color del boton gris
    red74=224;
    green74=224;
    blue74=224;
    texto="No está realizando\nmedición";
  }
}
if (overRect84==true)
{
  port.clear();
  status84=!status84;
  port.write("D");
  if (status84==true)
  {
    //Color del boton verde
    red84=120;
    green84=247;
    blue84=208;
    texto="Medición\nStrut 4";
  }
  if (status84==false)
  {
    //Color del boton gris
    red84=224;
    green84=224;
    blue84=224;
    texto="No está realizando\nmedición";
  }
}
if (overRect94==true)
{
  port.clear();
  status94=!status94;
  port.write("D");
  if (status94==true)
  {
    //Color del boton verde
    red94=120;
    green94=247;
    blue94=208;
```

```
    texto="Medición\nStrut 4";
   }
   if (status94==false)
   {
    //Color del boton gris
    red94=224;
    green94=224;
    blue94=224;
    texto="No está realizando\nmedición";
   }
  }
 if (overRect104==true)
 {
  port.clear();
  status104=!status104;
  port.write("D");
  if (status104==true)
  {
   //Color del boton verde
   red104=120;
   green104=247;
   blue104=208;
   texto="Medición\nStrut 4";
  }
  if (status104==false)
  {
   //Color del boton gris
   red104=224;
   green104=224;
   blue104=224;
   texto="No está realizando\nmedición";
  }
 }
 if (overRect114==true)
 {
  port.clear();
  status114=!status114;
  port.write("D");
  if (status114==true)
  {
   //Color del boton verde
   red114=120;
   green114=247;
   blue114=208;
   texto="Medición\nStrut 4";
  }
  if (status114==false)
  {
   //Color del boton gris
   red114=224;
   green114=224;
   blue114=224;
   texto="No está realizando\nmedición";
```

```
  }
 }
 if (overRect124==true)
 {
  port.clear();
  status124=!status124;
  port.write("D");
  if (status124==true)
  {
   //Color del boton verde
   red124=120;
   green124=247;
   blue124=208;
   texto="Medición\nStrut 4";
  }
  if (status124==false)
  {
   //Color del boton gris
   red124=224;
   green124=224;
   blue124=224;
   texto="No está realizando\nmedición";
  }
 }
 if (overRect134==true)
 {
  port.clear();
  status134=!status134;
  port.write("D");
  if (status134==true)
  {
   //Color del boton verde
   red134=120;
   green134=247;
   blue134=208;
   texto="Medición\nStrut 4";
  }
  if (status134==false)
  {
   //Color del boton gris
   red134=224;
   green134=224;
   blue134=224;
   texto="No está realizando\nmedición";
  }
 }
 if (overRect144==true)
 {
  port.clear();
  status144=!status144;
  port.write("D");
  if (status144==true)
  {
```

```
    //Color del boton verde
    red144=120;
    green144=247;
    blue144=208;
    texto="Medición\nStrut 4";
   }
   if (status144==false)
   {
    //Color del boton gris
    red144=224;
    green144=224;
    blue144=224;
    texto="No está realizando\nmedición";
   }
  }
  if (overRect154==true)
  {
   port.clear();
   status154=!status154;
   port.write("D");
   if (status154==true)
   {
    //Color del boton verde
    red154=120;
    green154=247;
    blue154=208;
    texto="Medición\nStrut 4";
   }
   if (status154==false)
   {
    //Color del boton gris
    red154=224;
    green154=224;
    blue154=224;
    texto="No está realizando\nmedición";
   }
  }
  if (overRect164==true)
  {
   port.clear();
   status164=!status164;
   port.write("D");
   if (status164==true)
   {
    //Color del boton verde
    red164=120;
    green164=247;
    blue164=208;
    texto="Medición\nStrut 4";
   }
   if (status164==false)
   {
    //Color del boton gris
```

```
    red164=224;
    green164=224;
    blue164=224;
    texto="No está realizando\nmedición";
   }
  }
  if (overRect174==true)
  {
   port.clear();
   status174=!status174;
   port.write("D");
   if (status174==true)
   {
    //Color del boton verde
    red174=120;
    green174=247;
    blue174=208;
    texto="Medición\nStrut 4";
   }
   if (status174==false)
   {
    //Color del boton gris
    red174=224;
    green174=224;
    blue174=224;
    texto="No está realizando\nmedición";
   }
  }
  if (overRect184==true)
  {
   port.clear();
   status184=!status184;
   port.write("D");
   if (status184==true)
   {
    //Color del boton verde
    red184=120;
    green184=247;
    blue184=208;
    texto="Medición\nStrut 4";
   }
   if (status184==false)
   {
    //Color del boton gris
    red184=224;
    green184=224;
    blue184=224;
    texto="No está realizando\nmedición";
   }
  }
  if (overRect194==true)
  {
   port.clear();
```

```
  status194=!status194;
  port.write("D");
  if (status194==true)
  {
    //Color del boton verde
    red194=120;
    green194=247;
    blue194=208;
    texto="Medición\nStrut 4";
  }
  if (status194==false)
  {
    //Color del boton gris
    red194=224;
    green194=224;
    blue194=224;
    texto="No está realizando\nmedición";
  }
}
if (overRect204==true)
{
  port.clear();
  status204=!status204;
  port.write("D");
  if (status204==true)
  {
    //Color del boton verde
    red204=120;
    green204=247;
    blue204=208;
    texto="Medición\nStrut 4";
  }
  if (status204==false)
  {
    //Color del boton gris
    red204=224;
    green204=224;
    blue204=224;
    texto="No está realizando\nmedición";
  }
}
if (overRect214==true)
{
  port.clear();
  status214=!status214;
  port.write("D");
  if (status214==true)
  {
    //Color del boton verde
    red214=120;
    green214=247;
    blue214=208;
    texto="Medición\nStrut 4";
```

```
  }
  if (status214==false)
  {
    //Color del boton gris
    red214=224;
    green214=224;
    blue214=224;
    texto="No está realizando\nmedición";
  }
}
if (overRect224==true)
{
  port.clear();
  status224=!status224;
  port.write("D");
  if (status224==true)
  {
    //Color del boton verde
    red224=120;
    green224=247;
    blue224=208;
    texto="Medición\nStrut 4";
  }
  if (status224==false)
  {
    //Color del boton gris
    red224=224;
    green224=224;
    blue224=224;
    texto="No está realizando\nmedición";
  }
}
if (overRect234==true)
{
  port.clear();
  status234=!status234;
  port.write("D");
  if (status234==true)
  {
    //Color del boton verde
    red234=120;
    green234=247;
    blue234=208;
    texto="Medición\nStrut 4";
  }
  if (status234==false)
  {
    //Color del boton gris
    red234=224;
    green234=224;
    blue234=224;
    texto="No está realizando\nmedición";
  }
```

```
}
if (overRect244==true)
{
  port.clear();
  status244=!status244;
  port.write("D");
  if (status244==true)
  {
    //Color del boton verde
    red244=120;
    green244=247;
    blue244=208;
    texto="Medición\nStrut 4";
  }
  if (status244==false)
  {
    //Color del boton gris
    red244=224;
    green244=224;
    blue244=224;
    texto="No está realizando\nmedición";
  }
}
if (overRect254==true)
{
  port.clear();
  status254=!status254;
  port.write("D");
  if (status253==true)
  {
    //Color del boton verde
    red254=120;
    green254=247;
    blue254=208;
    texto="Medición\nStrut 4";
  }
  if (status254==false)
  {
    //Color del boton gris
    red254=224;
    green254=224;
    blue254=224;
    texto="No está realizando\nmedición";
  }
}
if (overRect264==true)
{
  port.clear();
  status264=!status264;
  port.write("D");
  if (status264==true)
  {
    //Color del boton verde
```

```
    red264=120;
    green264=247;
    blue264=208;
    texto="Medición\nStrut 4";
   }
   if (status264==false)
   {
    //Color del boton gris
    red264=224;
    green264=224;
    blue264=224;
    texto="No está realizando\nmedición";
   }
  }
  if (overRect274==true)
  {
   port.clear();
   status274=!status274;
   port.write("D");
   if (status274==true)
   {
    //Color del boton verde
    red274=120;
    green274=247;
    blue274=208;
    texto="Medición\nStrut 4";
   }
   if (status274==false)
   {
    //Color del boton gris
    red274=224;
    green274=224;
    blue274=224;
    texto="No está realizando\nmedición";
   }
  }
  if (overRect284==true)
  {
   port.clear();
   status284=!status284;
   port.write("D");
   if (status284==true)
   {
    //Color del boton verde
    red284=120;
    green284=247;
    blue284=208;
    texto="Medición\nStrut 4";
   }
   if (status284==false)
   {
    //Color del boton gris
    red284=224;
```

```
    green284=224;
    blue284=224;
    texto="No está realizando\nmedición";
   }
  }
  if (overRect294==true)
  {
   port.clear();
   status294=!status294;
   port.write("D");
   if (status294==true)
   {
    //Color del boton verde
    red294=120;
    green294=247;
    blue294=208;
    texto="Medición\nStrut 4";
   }
   if (status294==false)
   {
    //Color del boton gris
    red294=224;
    green294=224;
    blue294=224;
    texto="No está realizando\nmedición";
   }
  }
  if (overRect304==true)
  {
   port.clear();
   status304=!status304;
   port.write("D");
   if (status304==true)
   {
    //Color del boton verde
    red304=120;
    green304=247;
    blue304=208;
    texto="Medición\nStrut 4";
   }
   if (status304==false)
   {
    //Color del boton gris
    red304=224;
    green304=224;
    blue304=224;
    texto="No está realizando\nmedición";
   }
  }
  if (overRect314==true)
  {
   port.clear();
   status314=!status314;
```

```
  port.write("D");
  if (status314==true)
  {
    //Color del boton verde
    red314=120;
    green314=247;
    blue314=208;
    texto="Medición\nStrut 4";
  }
  if (status314==false)
  {
    //Color del boton gris
    red314=224;
    green314=224;
    blue314=224;
    texto="No está realizando\nmedición";
  }
}
//// DATOS PUERTO SERIE STRUT 5/////
if (overRect05==true)
{
  port.clear();
  status05=!status05;
  port.write("E");
  if (status05==true)
  {
    //Color del boton verde
    red05=120;
    green05=247;
    blue05=208;
    texto="Medición\nStrut 5";
  }
  if (status05==false)
  {
    //Color del boton gris
    red05=224;
    green05=224;
    blue05=224;
    texto="No está realizando\nmedición";
  }
}
if (overRect15==true)
{
  port.clear();
  status15=!status15;
  port.write("E");
  if (status15==true)
  {
    //Color del boton verde
    red15=120;
    green15=247;
    blue15=208;
    texto="Medición\nStrut 5";
```

```
  }
  if (status15==false)
  {
    //Color del boton gris
    red15=224;
    green15=224;
    blue15=224;
    texto="No está realizando\nmedición";
  }
}
if (overRect25==true)
{
  port.clear();
  status25=!status25;
  port.write("E");
  if (status25==true)
  {
    //Color del boton verde
    red25=120;
    green25=247;
    blue25=208;
    texto="Medición\nStrut 5";
  }
  if (status25==false)
  {
    //Color del boton gris
    red25=224;
    green25=224;
    blue25=224;
    texto="No está realizando\nmedición";
  }
}
if (overRect35==true)
{
  port.clear();
  status35=!status35;
  port.write("E");
  if (status35==true)
  {
    //Color del boton verde
    red35=120;
    green35=247;
    blue35=208;
    texto="Medición\nStrut 5";
  }
  if (status35==false)
  {
    //Color del boton gris
    red35=224;
    green35=224;
    blue35=224;
    texto="No está realizando\nmedición";
  }
```

```
}
if (overRect45==true)
{
  port.clear();
  status45=!status45;
  port.write("E");
  if (status45==true)
  {
    //Color del boton verde
    red45=120;
    green45=247;
    blue45=208;
    texto="Medición\nStrut 5";
  }
  if (status45==false)
  {
    //Color del boton gris
    red45=224;
    green45=224;
    blue45=224;
    texto="No está realizando\nmedición";
  }
}
if (overRect55==true)
{
  port.clear();
  status55=!status55;
  port.write("E");
  if (status54==true)
  {
    //Color del boton verde
    red55=120;
    green55=247;
    blue55=208;
    texto="Medición\nStrut 5";
  }
  if (status55==false)
  {
    //Color del boton gris
    red55=224;
    green55=224;
    blue55=224;
    texto="No está realizando\nmedición";
  }
}
if (overRect65==true)
{
  port.clear();
  status65=!status65;
  port.write("E");
  if (status65==true)
  {
    //Color del boton verde
```

```
    red65=120;
    green65=247;
    blue65=208;
    texto="Medición\nStrut 5";
   }
   if (status65==false)
   {
    //Color del boton gris
    red65=224;
    green65=224;
    blue65=224;
    texto="No está realizando\nmedición";
   }
  }
  if (overRect75==true)
  {
   port.clear();
   status75=!status75;
   port.write("E");
   if (status75==true)
   {
    //Color del boton verde
    red75=120;
    green75=247;
    blue75=208;
    texto="Medición\nStrut 5";
   }
   if (status75==false)
   {
    //Color del boton gris
    red75=224;
    green75=224;
    blue75=224;
    texto="No está realizando\nmedición";
   }
  }
  if (overRect85==true)
  {
   port.clear();
   status85=!status85;
   port.write("E");
   if (status85==true)
   {
    //Color del boton verde
    red85=120;
    green85=247;
    blue85=208;
    texto="Medición\nStrut 5";
   }
   if (status85==false)
   {
    //Color del boton gris
    red85=224;
```

```
    green85=224;
    blue85=224;
    texto="No está realizando\nmedición";
   }
  }
  if (overRect95==true)
  {
   port.clear();
   status95=!status95;
   port.write("E");
   if (status95==true)
   {
    //Color del boton verde
    red95=120;
    green95=247;
    blue95=208;
    texto="Medición\nStrut 5";
   }
   if (status95==false)
   {
    //Color del boton gris
    red95=224;
    green95=224;
    blue95=224;
    texto="No está realizando\nmedición";
   }
  }
  if (overRect105==true)
  {
   port.clear();
   status105=!status105;
   port.write("E");
   if (status105==true)
   {
    //Color del boton verde
    red105=120;
    green105=247;
    blue105=208;
    texto="Medición\nStrut 5";
   }
   if (status105==false)
   {
    //Color del boton gris
    red105=224;
    green105=224;
    blue105=224;
    texto="No está realizando\nmedición";
   }
  }
  if (overRect115==true)
  {
   port.clear();
   status115=!status115;
```

```
  port.write("E");
  if (status115==true)
  {
    //Color del boton verde
    red115=120;
    green115=247;
    blue115=208;
    texto="Medición\nStrut 5";
  }
  if (status115==false)
  {
    //Color del boton gris
    red115=224;
    green115=224;
    blue115=224;
    texto="No está realizando\nmedición";
  }
}
if (overRect125==true)
{
  port.clear();
  status125=!status125;
  port.write("E");
  if (status125==true)
  {
    //Color del boton verde
    red125=120;
    green125=247;
    blue125=208;
    texto="Medición\nStrut 5";
  }
  if (status125==false)
  {
    //Color del boton gris
    red125=224;
    green125=224;
    blue125=224;
    texto="No está realizando\nmedición";
  }
}
if (overRect135==true)
{
  port.clear();
  status135=!status135;
  port.write("E");
  if (status135==true)
  {
    //Color del boton verde
    red135=120;
    green135=247;
    blue135=208;
    texto="Medición\nStrut 5";
  }
```

```
  if (status135==false)
  {
    //Color del boton gris
    red135=224;
    green135=224;
    blue135=224;
    texto="No está realizando\nmedición";
  }
}
if (overRect145==true)
{
  port.clear();
  status145=!status145;
  port.write("E");
  if (status145==true)
  {
    //Color del boton verde
    red145=120;
    green145=247;
    blue145=208;
    texto="Medición\nStrut 5";
  }
  if (status145==false)
  {
    //Color del boton gris
    red145=224;
    green145=224;
    blue145=224;
    texto="No está realizando\nmedición";
  }
}
if (overRect155==true)
{
  port.clear();
  status155=!status155;
  port.write("E");
  if (status155==true)
  {
    //Color del boton verde
    red155=120;
    green155=247;
    blue155=208;
    texto="Medición\nStrut 5";
  }
  if (status155==false)
  {
    //Color del boton gris
    red155=224;
    green155=224;
    blue155=224;
    texto="No está realizando\nmedición";
  }
}
```

```
if (overRect165==true)
{
 port.clear();
 status165=!status165;
 port.write("E");
 if (status165==true)
 {
  //Color del boton verde
  red165=120;
  green165=247;
  blue165=208;
  texto="Medición\nStrut 5";
 }
 if (status165==false)
 {
  //Color del boton gris
  red165=224;
  green165=224;
  blue165=224;
  texto="No está realizando\nmedición";
 }
}
if (overRect175==true)
{
 port.clear();
 status175=!status175;
 port.write("E");
 if (status175==true)
 {
  //Color del boton verde
  red175=120;
  green175=247;
  blue175=208;
  texto="Medición\nStrut 5";
 }
 if (status175==false)
 {
  //Color del boton gris
  red175=224;
  green175=224;
  blue175=224;
  texto="No está realizando\nmedición";
 }
}
if (overRect185==true)
{
 port.clear();
 status185=!status185;
 port.write("E");
 if (status185==true)
 {
  //Color del boton verde
  red185=120;
```

```
    green185=247;
    blue185=208;
    texto="Medición\nStrut 5";
   }
   if (status185==false)
   {
    //Color del boton gris
    red185=224;
    green185=224;
    blue185=224;
    texto="No está realizando\nmedición";
   }
 }
 if (overRect195==true)
 {
  port.clear();
  status195=!status195;
  port.write("E");
  if (status195==true)
  {
    //Color del boton verde
    red195=120;
    green195=247;
    blue195=208;
    texto="Medición\nStrut 5";
  }
  if (status195==false)
  {
    //Color del boton gris
    red195=224;
    green195=224;
    blue195=224;
    texto="No está realizando\nmedición";
  }
 }
 if (overRect205==true)
 {
  port.clear();
  status205=!status205;
  port.write("E");
  if (status205==true)
  {
    //Color del boton verde
    red205=120;
    green205=247;
    blue205=208;
    texto="Medición\nStrut 5";
  }
  if (status205==false)
  {
    //Color del boton gris
    red205=224;
    green205=224;
```

```
    blue205=224;
    texto="No está realizando\nmedición";
   }
  }
 if (overRect215==true)
 {
  port.clear();
  status215=!status215;
  port.write("E");
  if (status215==true)
  {
    //Color del boton verde
    red215=120;
    green215=247;
    blue215=208;
    texto="Medición\nStrut 5";
  }
  if (status215==false)
  {
    //Color del boton gris
    red215=224;
    green215=224;
    blue215=224;
    texto="No está realizando\nmedición";
  }
 }
 if (overRect225==true)
 {
  port.clear();
  status225=!status225;
  port.write("E");
  if (status225==true)
  {
    //Color del boton verde
    red225=120;
    green225=247;
    blue225=208;
    texto="Medición\nStrut 5";
  }
  if (status225==false)
  {
    //Color del boton gris
    red225=224;
    green225=224;
    blue225=224;
    texto="No está realizando\nmedición";
  }
 }
 if (overRect235==true)
 {
  port.clear();
  status235=!status235;
  port.write("E");
```

```
 if (status235==true)
 {
  //Color del boton verde
  red235=120;
  green235=247;
  blue235=208;
  texto="Medición\nStrut 5";
 }
 if (status235==false)
 {
  //Color del boton gris
  red235=224;
  green235=224;
  blue235=224;
  texto="No está realizando\nmedición";
 }
}
if (overRect245==true)
{
 port.clear();
 status245=!status245;
 port.write("E");
 if (status245==true)
 {
  //Color del boton verde
  red245=120;
  green245=247;
  blue245=208;
  texto="Medición\nStrut 5";
 }
 if (status245==false)
 {
  //Color del boton gris
  red245=224;
  green245=224;
  blue245=224;
  texto="No está realizando\nmedición";
 }
}
if (overRect255==true)
{
 port.clear();
 status255=!status255;
 port.write("E");
 if (status255==true)
 {
  //Color del boton verde
  red255=120;
  green255=247;
  blue255=208;
  texto="Medición\nStrut 5";
 }
 if (status255==false)
```

```
  {
    //Color del boton gris
    red255=224;
    green255=224;
    blue255=224;
    texto="No está realizando\nmedición";
  }
}
if (overRect265==true)
{
  port.clear();
  status265=!status265;
  port.write("E");
  if (status265==true)
  {
    //Color del boton verde
    red265=120;
    green265=247;
    blue265=208;
    texto="Medición\nStrut 5";
  }
  if (status265==false)
  {
    //Color del boton gris
    red265=224;
    green265=224;
    blue265=224;
    texto="No está realizando\nmedición";
  }
}
if (overRect275==true)
{
  port.clear();
  status275=!status275;
  port.write("E");
  if (status275==true)
  {
    //Color del boton verde
    red275=120;
    green275=247;
    blue275=208;
    texto="Medición\nStrut 5";
  }
  if (status275==false)
  {
    //Color del boton gris
    red275=224;
    green275=224;
    blue275=224;
    texto="No está realizando\nmedición";
  }
}
if (overRect285==true)
```

```
{
  port.clear();
  status285=!status285;
  port.write("E");
  if (status285==true)
  {
    //Color del boton verde
    red285=120;
    green285=247;
    blue285=208;
    texto="Medición\nStrut 5";
  }
  if (status285==false)
  {
    //Color del boton gris
    red285=224;
    green285=224;
    blue285=224;
    texto="No está realizando\nmedición";
  }
}
if (overRect295==true)
{
  port.clear();
  status295=!status295;
  port.write("E");
  if (status295==true)
  {
    //Color del boton verde
    red295=120;
    green295=247;
    blue295=208;
    texto="Medición\nStrut 5";
  }
  if (status295==false)
  {
    //Color del boton gris
    red295=224;
    green295=224;
    blue295=224;
    texto="No está realizando\nmedición";
  }
}
if (overRect305==true)
{
  port.clear();
  status305=!status305;
  port.write("E");
  if (status305==true)
  {
    //Color del boton verde
    red305=120;
    green305=247;
```

```
    blue305=208;
    texto="Medición\nStrut 5";
   }
   if (status305==false)
   {
    //Color del boton gris
    red305=224;
    green305=224;
    blue305=224;
    texto="No está realizando\nmedición";
   }
  }
  if (overRect315==true)
  {
   port.clear();
   status315=!status315;
   port.write("E");
   if (status315==true)
   {
    //Color del boton verde
    red315=120;
    green315=247;
    blue315=208;
    texto="Medición\nStrut 5";
   }
   if (status315==false)
   {
    //Color del boton gris
    red315=224;
    green315=224;
    blue315=224;
    texto="No está realizando\nmedición";
   }
  }
  //// DATOS PUERTO SERIE STRUT 6/////
  if (overRect06==true)
  {
   port.clear();
   status06=!status06;
   port.write("F");
   if (status06==true)
   {
    //Color del boton verde
    red06=120;
    green06=247;
    blue06=208;
    texto="Medición\nStrut 6";
   }
   if (status06==false)
   {
    //Color del boton gris
    red06=224;
    green06=224;
```

```
    blue06=224;
    texto="No está realizando\nmedición";
   }
  }
 if (overRect16==true)
 {
  port.clear();
  status16=!status16;
  port.write("F");
  if (status16==true)
  {
    //Color del boton verde
    red16=120;
    green16=247;
    blue16=208;
    texto="Medición\nStrut 6";
  }
  if (status16==false)
  {
    //Color del boton gris
    red16=224;
    green16=224;
    blue16=224;
    texto="No está realizando\nmedición";
  }
 }
 if (overRect26==true)
 {
  port.clear();
  status26=!status26;
  port.write("F");
  if (status26==true)
  {
    //Color del boton verde
    red26=120;
    green26=247;
    blue26=208;
    texto="Medición\nStrut 6";
  }
  if (status26==false)
  {
    //Color del boton gris
    red26=224;
    green26=224;
    blue26=224;
    texto="No está realizando\nmedición";
  }
 }
 if (overRect36==true)
 {
  port.clear();
  status36=!status36;
  port.write("F");
```

```
  if (status36==true)
  {
    //Color del boton verde
    red36=120;
    green36=247;
    blue36=208;
    texto="Medición\nStrut 6";
  }
  if (status36==false)
  {
    //Color del boton gris
    red36=224;
    green36=224;
    blue36=224;
    texto="No está realizando\nmedición";
  }
}
if (overRect46==true)
{
  port.clear();
  status46=!status46;
  port.write("F");
  if (status46==true)
  {
    //Color del boton verde
    red46=120;
    green46=247;
    blue46=208;
    texto="Medición\nStrut 6";
  }
  if (status46==false)
  {
    //Color del boton gris
    red46=224;
    green46=224;
    blue46=224;
    texto="No está realizando\nmedición";
  }
}
if (overRect56==true)
{
  port.clear();
  status56=!status56;
  port.write("F");
  if (status56==true)
  {
    //Color del boton verde
    red56=120;
    green56=247;
    blue56=208;
    texto="Medición\nStrut 6";
  }
  if (status56==false)
```

```
  {
    //Color del boton gris
    red56=224;
    green56=224;
    blue56=224;
    texto="No está realizando\nmedición";
  }
}
if (overRect66==true)
{
  port.clear();
  status66=!status66;
  port.write("F");
  if (status66==true)
  {
    //Color del boton verde
    red66=120;
    green66=247;
    blue66=208;
    texto="Medición\nStrut 6";
  }
  if (status66==false)
  {
    //Color del boton gris
    red66=224;
    green66=224;
    blue66=224;
    texto="No está realizando\nmedición";
  }
}
if (overRect76==true)
{
  port.clear();
  status76=!status76;
  port.write("F");
  if (status76==true)
  {
    //Color del boton verde
    red76=120;
    green76=247;
    blue76=208;
    texto="Medición\nStrut 6";
  }
  if (status76==false)
  {
    //Color del boton gris
    red76=224;
    green76=224;
    blue76=224;
    texto="No está realizando\nmedición";
  }
}
if (overRect86==true)
```

```
{
  port.clear();
  status86=!status86;
  port.write("F");
  if (status85==true)
  {
    //Color del boton verde
    red86=120;
    green86=247;
    blue86=208;
    texto="Medición\nStrut 6";
  }
  if (status86==false)
  {
    //Color del boton gris
    red86=224;
    green86=224;
    blue86=224;
    texto="No está realizando\nmedición";
  }
}
if (overRect96==true)
{
  port.clear();
  status96=!status96;
  port.write("F");
  if (status96==true)
  {
    //Color del boton verde
    red96=120;
    green96=247;
    blue96=208;
    texto="Medición\nStrut 6";
  }
  if (status96==false)
  {
    //Color del boton gris
    red96=224;
    green96=224;
    blue96=224;
    texto="No está realizando\nmedición";
  }
}
if (overRect106==true)
{
  port.clear();
  status106=!status106;
  port.write("F");
  if (status106==true)
  {
    //Color del boton verde
    red106=120;
    green106=247;
```

```
    blue106=208;
    texto="Medición\nStrut 6";
   }
   if (status106==false)
   {
    //Color del boton gris
    red106=224;
    green106=224;
    blue106=224;
    texto="No está realizando\nmedición";
   }
  }
  if (overRect116==true)
  {
   port.clear();
   status116=!status116;
   port.write("F");
   if (status116==true)
   {
    //Color del boton verde
    red116=120;
    green116=247;
    blue116=208;
    texto="Medición\nStrut 6";
   }
   if (status116==false)
   {
    //Color del boton gris
    red116=224;
    green116=224;
    blue116=224;
    texto="No está realizando\nmedición";
   }
  }
  if (overRect126==true)
  {
   port.clear();
   status126=!status126;
   port.write("F");
   if (status126==true)
   {
    //Color del boton verde
    red126=120;
    green126=247;
    blue126=208;
    texto="Medición\nStrut 6";
   }
   if (status126==false)
   {
    //Color del boton gris
    red126=224;
    green126=224;
    blue126=224;
```

```
    texto="No está realizando\nmedición";
   }
  }
  if (overRect136==true)
  {
   port.clear();
   status136=!status136;
   port.write("F");

   if (status136==true)
   {
    //Color del boton verde
    red136=120;
    green136=247;
    blue136=208;
    texto="Medición\nStrut 6";
   }
   if (status136==false)
   {
    //Color del boton gris
    red136=224;
    green136=224;
    blue136=224;
    texto="No está realizando\nmedición";
   }
  }
  if (overRect146==true)
  {
   port.clear();
   status146=!status146;
   port.write("F");
   if (status146==true)
   {
    //Color del boton verde
    red146=120;
    green146=247;
    blue146=208;
    texto="Medición\nStrut 6";
   }
   if (status146==false)
   {
    //Color del boton gris
    red146=224;
    green146=224;
    blue146=224;
    texto="No está realizando\nmedición";
   }
  }
  if (overRect156==true)
  {
   port.clear();
   status156=!status156;
   port.write("F");
```

```
 if (status156==true)
 {
  //Color del boton verde
  red156=120;
  green156=247;
  blue156=208;
  texto="Medición\nStrut 6";
 }
 if (status156==false)
 {
  //Color del boton gris
  red156=224;
  green156=224;
  blue156=224;
  texto="No está realizando\nmedición";
 }
}
if (overRect166==true)
{
 port.clear();
 status166=!status166;
 port.write("F");
 if (status165==true)
 {
  //Color del boton verde
  red166=120;
  green166=247;
  blue166=208;
  texto="Medición\nStrut 6";
 }
 if (status166==false)
 {
  //Color del boton gris
  red166=224;
  green166=224;
  blue166=224;
  texto="No está realizando\nmedición";
 }
}
if (overRect176==true)
{
 port.clear();
 status176=!status176;
 port.write("F");
 if (status176==true)
 {
  //Color del boton verde
  red176=120;
  green176=247;
  blue176=208;
  texto="Medición\nStrut 6";
 }
 if (status176==false)
```

```
  {
    //Color del boton gris
    red176=224;
    green176=224;
    blue176=224;
    texto="No está realizando\nmedición";
  }
}
if (overRect186==true)
{
  port.clear();
  status186=!status186;
  port.write("F");
  if (status186==true)
  {
    //Color del boton verde
    red186=120;
    green186=247;
    blue186=208;
    texto="Medición\nStrut 6";
  }
  if (status186==false)
  {
    //Color del boton gris
    red186=224;
    green186=224;
    blue186=224;
    texto="No está realizando\nmedición";
  }
}
if (overRect196==true)
{
  port.clear();
  status196=!status196;
  port.write("F");
  if (status196==true)
  {
    //Color del boton verde
    red196=120;
    green196=247;
    blue196=208;
    texto="Medición\nStrut 6";
  }
  if (status196==false)
  {
    //Color del boton gris
    red196=224;
    green196=224;
    blue196=224;
    texto="No está realizando\nmedición";
  }
}
if (overRect206==true)
```

```
{
 port.clear();
 status206=!status206;
 port.write("F");
 if (status206==true)
 {
  //Color del boton verde
  red206=120;
  green206=247;
  blue206=208;
  texto="Medición\nStrut 6";
 }
 if (status206==false)
 {
  //Color del boton gris
  red206=224;
  green206=224;
  blue206=224;
  texto="No está realizando\nmedición";
 }
}
if (overRect216==true)
{
 port.clear();
 status216=!status216;
 port.write("F");
 if (status216==true)
 {
  //Color del boton verde
  red216=120;
  green216=247;
  blue216=208;
  texto="Medición\nStrut 6";
 }
 if (status216==false)
 {
  //Color del boton gris
  red216=224;
  green216=224;
  blue216=224;
  texto="No está realizando\nmedición";
 }
}
if (overRect226==true)
{
 port.clear();
 status226=!status226;
 port.write("F");
 if (status226==true)
 {
  //Color del boton verde
  red226=120;
  green226=247;
```

```
    blue226=208;
    texto="Medición\nStrut 6";
   }
   if (status226==false)
   {
    //Color del boton gris
    red226=224;
    green226=224;
    blue226=224;
    texto="No está realizando\nmedición";
   }
  }
  if (overRect236==true)
  {
   port.clear();
   status236=!status236;
   port.write("F");
   if (status236==true)
   {
    //Color del boton verde
    red236=120;
    green236=247;
    blue236=208;
    texto="Medición\nStrut 6";
   }
   if (status236==false)
   {
    //Color del boton gris
    red236=224;
    green236=224;
    blue236=224;
    texto="No está realizando\nmedición";
   }
  }
  if (overRect246==true)
  {
   port.clear();
   status246=!status246;
   port.write("F");
   if (status246==true)
   {
    //Color del boton verde
    red246=120;
    green246=247;
    blue246=208;
    texto="Medición\nStrut 6";
   }
   if (status246==false)
   {
    //Color del boton gris
    red246=224;
    green246=224;
    blue246=224;
```

```
    texto="No está realizando\nmedición";
   }
  }
  if (overRect256==true)
  {
   port.clear();
   status256=!status256;
   port.write("F");
   if (status256==true)
   {
    //Color del boton verde
    red256=120;
    green256=247;
    blue256=208;
    texto="Medición\nStrut 6";
   }
   if (status256==false)
   {
    //Color del boton gris
    red256=224;
    green256=224;
    blue256=224;
    texto="No está realizando\nmedición";
   }
  }
  if (overRect266==true)
  {
   port.clear();
   status266=!status266;
   port.write("F");
   if (status266==true)
   {
    //Color del boton verde
    red266=120;
    green266=247;
    blue266=208;
    texto="Medición\nStrut 6";
   }
   if (status266==false)
   {
    //Color del boton gris
    red266=224;
    green266=224;
    blue266=224;
    texto="No está realizando\nmedición";
   }
  }
  if (overRect276==true)
  {
   port.clear();
   status276=!status276;
   port.write("F");
   if (status276==true)
```

```
  {
   //Color del boton verde
   red276=120;
   green276=247;
   blue276=208;
   texto="Medición\nStrut 6";
  }
  if (status276==false)
  {
   //Color del boton gris
   red276=224;
   green276=224;
   blue276=224;
   texto="No está realizando\nmedición";
  }
 }
 if (overRect286==true)
 {
  port.clear();
  status286=!status286;
  port.write("F");
  if (status286==true)
  {
   //Color del boton verde
   red286=120;
   green286=247;
   blue286=208;
   texto="Medición\nStrut 6";
  }
  if (status286==false)
  {
   //Color del boton gris
   red286=224;
   green286=224;
   blue286=224;
   texto="No está realizando\nmedición";
  }
 }
 if (overRect296==true)
 {
  port.clear();
  status296=!status296;
  port.write("F");
  if (status296==true)
  {
   //Color del boton verde
   red296=120;
   green296=247;
   blue296=208;
   texto="Medición\nStrut 6";
  }
  if (status296==false)
  {
```

```
      //Color del boton gris
      red296=224;
      green296=224;
      blue296=224;
      texto="No está realizando\nmedición";
    }
  }
  if (overRect306==true)
  {
    port.clear();
    status306=!status306;
    port.write("F");
    if (status306==true)
    {
      //Color del boton verde
      red306=120;
      green306=247;
      blue306=208;
      texto="Medición\nStrut 6";
    }
    if (status306==false)
    {
      //Color del boton gris
      red306=224;
      green306=224;
      blue306=224;
      texto="No está realizando\nmedición";
    }
  }
  if (overRect316==true)
  {
    port.clear();
    status316=!status316;
    port.write("F");
    if (status316==true)
    {
      //Color del boton verde
      red316=120;
      green316=247;
      blue316=208;
      texto="Medición\nStrut 6";
    }
    if (status316==false)
    {
      //Color del boton gris
      red316=224;
      green316=224;
      blue316=224;
      texto="No está realizando\nmedición";
    }
  }
}
```

```
void keyPressed() {
  for (TEXTBOX t : textboxes) {
    t.KEYPRESSED(key, keyCode);
  }
}

void InitLayout() {
  ///Strut 1///
  //Columna A//
  TEXTBOX receiver01 = new TEXTBOX(71, 81, 29, 14);
  textboxes.add(receiver01);
  TEXTBOX receiver11 = new TEXTBOX(71, 96, 29, 14);
  textboxes.add(receiver11);
  TEXTBOX receiver21 = new TEXTBOX(71, 111, 29, 14);
  textboxes.add(receiver21);
  TEXTBOX receiver31 = new TEXTBOX(71, 126, 29, 14);
  textboxes.add(receiver31);
  TEXTBOX receiver41 = new TEXTBOX(71, 141, 29, 14);
  textboxes.add(receiver41);
  TEXTBOX receiver51 = new TEXTBOX(71, 156, 29, 14);
  textboxes.add(receiver51);
  TEXTBOX receiver61 = new TEXTBOX(71, 171, 29, 14);
  textboxes.add(receiver61);
  TEXTBOX receiver71 = new TEXTBOX(71, 186, 29, 14);
  textboxes.add(receiver71);
  TEXTBOX receiver81 = new TEXTBOX(71, 201, 29, 14);
  textboxes.add(receiver81);
  TEXTBOX receiver91 = new TEXTBOX(71, 216, 29, 14);
  textboxes.add(receiver91);
  TEXTBOX receiver101 = new TEXTBOX(71, 231, 29, 14);
  textboxes.add(receiver101);
  TEXTBOX receiver111 = new TEXTBOX(71, 246, 29, 14);
  textboxes.add(receiver111);
  TEXTBOX receiver121 = new TEXTBOX(71, 261, 29, 14);
  textboxes.add(receiver121);
  TEXTBOX receiver131 = new TEXTBOX(71, 276, 29, 14);
  textboxes.add(receiver131);
  TEXTBOX receiver141 = new TEXTBOX(71, 291, 29, 14);
  textboxes.add(receiver141);
  TEXTBOX receiver151 = new TEXTBOX(71, 306, 29, 14);
  textboxes.add(receiver151);
  TEXTBOX receiver161 = new TEXTBOX(71, 321, 29, 14);
  textboxes.add(receiver161);
  TEXTBOX receiver171 = new TEXTBOX(71, 336, 29, 14);
  textboxes.add(receiver171);
  TEXTBOX receiver181 = new TEXTBOX(71, 351, 29, 14);
  textboxes.add(receiver181);
  TEXTBOX receiver191 = new TEXTBOX(71, 366, 29, 14);
  textboxes.add(receiver191);
  TEXTBOX receiver201 = new TEXTBOX(71, 381, 29, 14);
  textboxes.add(receiver201);
  TEXTBOX receiver211 = new TEXTBOX(71, 396, 29, 14);
  textboxes.add(receiver211);
```

```
TEXTBOX receiver221 = new TEXTBOX(71, 411, 29, 14);
textboxes.add(receiver221);
TEXTBOX receiver231 = new TEXTBOX(71, 426, 29, 14);
textboxes.add(receiver231);
TEXTBOX receiver241 = new TEXTBOX(71, 441, 29, 14);
textboxes.add(receiver241);
TEXTBOX receiver251 = new TEXTBOX(71, 456, 29, 14);
textboxes.add(receiver251);
TEXTBOX receiver261 = new TEXTBOX(71, 471, 29, 14);
textboxes.add(receiver261);
TEXTBOX receiver271 = new TEXTBOX(71, 486, 29, 14);
textboxes.add(receiver271);
TEXTBOX receiver281 = new TEXTBOX(71, 501, 29, 14);
textboxes.add(receiver281);
TEXTBOX receiver291 = new TEXTBOX(71, 516, 29, 14);
textboxes.add(receiver291);
TEXTBOX receiver301 = new TEXTBOX(71, 531, 29, 14);
textboxes.add(receiver301);
TEXTBOX receiver311 = new TEXTBOX(71, 546, 29, 14);
textboxes.add(receiver311);
//Columna Size//
TEXTBOX receiver01S = new TEXTBOX(131, 81, 59, 14);
textboxes.add(receiver01S);
TEXTBOX receiver11S = new TEXTBOX(131, 96, 59, 14);
textboxes.add(receiver11S);
TEXTBOX receiver21S = new TEXTBOX(131, 111, 59, 14);
textboxes.add(receiver21S);
TEXTBOX receiver31S = new TEXTBOX(131, 126, 59, 14);
textboxes.add(receiver31S);
TEXTBOX receiver41S = new TEXTBOX(131, 141, 59, 14);
textboxes.add(receiver41S);
TEXTBOX receiver51S = new TEXTBOX(131, 156, 59, 14);
textboxes.add(receiver51S);
TEXTBOX receiver61S = new TEXTBOX(131, 171, 59, 14);
textboxes.add(receiver61S);
TEXTBOX receiver71S = new TEXTBOX(131, 186, 59, 14);
textboxes.add(receiver71S);
TEXTBOX receiver81S = new TEXTBOX(131, 201, 59, 14);
textboxes.add(receiver81S);
TEXTBOX receiver91S = new TEXTBOX(131, 216, 59, 14);
textboxes.add(receiver91S);
TEXTBOX receiver101S = new TEXTBOX(131, 231, 59, 14);
textboxes.add(receiver101S);
TEXTBOX receiver111S = new TEXTBOX(131, 246, 59, 14);
textboxes.add(receiver111S);
TEXTBOX receiver121S = new TEXTBOX(131, 261, 59, 14);
textboxes.add(receiver121S);
TEXTBOX receiver131S = new TEXTBOX(131, 276, 59, 14);
textboxes.add(receiver131S);
TEXTBOX receiver141S = new TEXTBOX(131, 291, 59, 14);
textboxes.add(receiver141S);
TEXTBOX receiver151S = new TEXTBOX(131, 306, 59, 14);
textboxes.add(receiver151S);
```

```
TEXTBOX receiver161S = new TEXTBOX(131, 321, 59, 14);
textboxes.add(receiver161S);
TEXTBOX receiver171S = new TEXTBOX(131, 336, 59, 14);
textboxes.add(receiver171S);
TEXTBOX receiver181S = new TEXTBOX(131, 351, 59, 14);
textboxes.add(receiver181S);
TEXTBOX receiver191S = new TEXTBOX(131, 366, 59, 14);
textboxes.add(receiver191S);
TEXTBOX receiver201S = new TEXTBOX(131, 381, 59, 14);
textboxes.add(receiver201S);
TEXTBOX receiver211S = new TEXTBOX(131, 396, 59, 14);
textboxes.add(receiver211S);
TEXTBOX receiver221S = new TEXTBOX(131, 411, 59, 14);
textboxes.add(receiver221S);
TEXTBOX receiver231S = new TEXTBOX(131, 426, 59, 14);
textboxes.add(receiver231S);
TEXTBOX receiver241S = new TEXTBOX(131, 441, 59, 14);
textboxes.add(receiver241S);
TEXTBOX receiver251S = new TEXTBOX(131, 456, 59, 14);
textboxes.add(receiver251S);
TEXTBOX receiver261S = new TEXTBOX(131, 471, 59, 14);
textboxes.add(receiver261S);
TEXTBOX receiver271S = new TEXTBOX(131, 486, 59, 14);
textboxes.add(receiver271S);
TEXTBOX receiver281S = new TEXTBOX(131, 501, 59, 14);
textboxes.add(receiver281S);
TEXTBOX receiver291S = new TEXTBOX(131, 516, 59, 14);
textboxes.add(receiver291S);
TEXTBOX receiver301S = new TEXTBOX(131, 531, 59, 14);
textboxes.add(receiver301S);
TEXTBOX receiver311S = new TEXTBOX(131, 546, 59, 14);
textboxes.add(receiver311S);
///Strut 2///
//Columna A//
TEXTBOX receiver02 = new TEXTBOX(191, 81, 29, 14);
textboxes.add(receiver02);
TEXTBOX receiver12 = new TEXTBOX(191, 96, 29, 14);
textboxes.add(receiver12);
TEXTBOX receiver22 = new TEXTBOX(191, 111, 29, 14);
textboxes.add(receiver22);
TEXTBOX receiver32 = new TEXTBOX(191, 126, 29, 14);
textboxes.add(receiver32);
TEXTBOX receiver42 = new TEXTBOX(191, 141, 29, 14);
textboxes.add(receiver42);
TEXTBOX receiver52 = new TEXTBOX(191, 156, 29, 14);
textboxes.add(receiver52);
TEXTBOX receiver62 = new TEXTBOX(191, 171, 29, 14);
textboxes.add(receiver62);
TEXTBOX receiver72 = new TEXTBOX(191, 186, 29, 14);
textboxes.add(receiver72);
TEXTBOX receiver82 = new TEXTBOX(191, 201, 29, 14);
textboxes.add(receiver82);
TEXTBOX receiver92 = new TEXTBOX(191, 216, 29, 14);
```

```
textboxes.add(receiver92);
TEXTBOX receiver102 = new TEXTBOX(191, 231, 29, 14);
textboxes.add(receiver102);
TEXTBOX receiver112 = new TEXTBOX(191, 246, 29, 14);
textboxes.add(receiver112);
TEXTBOX receiver122 = new TEXTBOX(191, 261, 29, 14);
textboxes.add(receiver122);
TEXTBOX receiver132 = new TEXTBOX(191, 276, 29, 14);
textboxes.add(receiver132);
TEXTBOX receiver142 = new TEXTBOX(191, 291, 29, 14);
textboxes.add(receiver142);
TEXTBOX receiver152 = new TEXTBOX(191, 306, 29, 14);
textboxes.add(receiver152);
TEXTBOX receiver162 = new TEXTBOX(191, 321, 29, 14);
textboxes.add(receiver162);
TEXTBOX receiver172 = new TEXTBOX(191, 336, 29, 14);
textboxes.add(receiver172);
TEXTBOX receiver182 = new TEXTBOX(191, 351, 29, 14);
textboxes.add(receiver182);
TEXTBOX receiver192 = new TEXTBOX(191, 366, 29, 14);
textboxes.add(receiver192);
TEXTBOX receiver202 = new TEXTBOX(191, 381, 29, 14);
textboxes.add(receiver202);
TEXTBOX receiver212 = new TEXTBOX(191, 396, 29, 14);
textboxes.add(receiver212);
TEXTBOX receiver222 = new TEXTBOX(191, 411, 29, 14);
textboxes.add(receiver222);
TEXTBOX receiver232 = new TEXTBOX(191, 426, 29, 14);
textboxes.add(receiver232);
TEXTBOX receiver242 = new TEXTBOX(191, 441, 29, 14);
textboxes.add(receiver242);
TEXTBOX receiver252 = new TEXTBOX(191, 456, 29, 14);
textboxes.add(receiver252);
TEXTBOX receiver262 = new TEXTBOX(191, 471, 29, 14);
textboxes.add(receiver262);
TEXTBOX receiver272 = new TEXTBOX(191, 486, 29, 14);
textboxes.add(receiver272);
TEXTBOX receiver282 = new TEXTBOX(191, 501, 29, 14);
textboxes.add(receiver282);
TEXTBOX receiver292 = new TEXTBOX(191, 516, 29, 14);
textboxes.add(receiver292);
TEXTBOX receiver302 = new TEXTBOX(191, 531, 29, 14);
textboxes.add(receiver302);
TEXTBOX receiver312 = new TEXTBOX(191, 546, 29, 14);
textboxes.add(receiver312);
//Columna Size//
TEXTBOX receiver02S = new TEXTBOX(251, 81, 59, 14);
textboxes.add(receiver02S);
TEXTBOX receiver12S = new TEXTBOX(251, 96, 59, 14);
textboxes.add(receiver12S);
TEXTBOX receiver22S = new TEXTBOX(251, 111, 59, 14);
textboxes.add(receiver22S);
TEXTBOX receiver32S = new TEXTBOX(251, 126, 59, 14);
```

```
textboxes.add(receiver32S);
TEXTBOX receiver42S = new TEXTBOX(251, 141, 59, 14);
textboxes.add(receiver42S);
TEXTBOX receiver52S = new TEXTBOX(251, 156, 59, 14);
textboxes.add(receiver52S);
TEXTBOX receiver62S = new TEXTBOX(251, 171, 59, 14);
textboxes.add(receiver62S);
TEXTBOX receiver72S = new TEXTBOX(251, 186, 59, 14);
textboxes.add(receiver72S);
TEXTBOX receiver82S = new TEXTBOX(251, 201, 59, 14);
textboxes.add(receiver82S);
TEXTBOX receiver92S = new TEXTBOX(251, 216, 59, 14);
textboxes.add(receiver92S);
TEXTBOX receiver102S = new TEXTBOX(251, 231, 59, 14);
textboxes.add(receiver102S);
TEXTBOX receiver112S = new TEXTBOX(251, 246, 59, 14);
textboxes.add(receiver112S);
TEXTBOX receiver122S = new TEXTBOX(251, 261, 59, 14);
textboxes.add(receiver122S);
TEXTBOX receiver132S = new TEXTBOX(251, 276, 59, 14);
textboxes.add(receiver132S);
TEXTBOX receiver142S = new TEXTBOX(251, 291, 59, 14);
textboxes.add(receiver142S);
TEXTBOX receiver152S = new TEXTBOX(251, 306, 59, 14);
textboxes.add(receiver152S);
TEXTBOX receiver162S = new TEXTBOX(251, 321, 59, 14);
textboxes.add(receiver162S);
TEXTBOX receiver172S = new TEXTBOX(251, 336, 59, 14);
textboxes.add(receiver172S);
TEXTBOX receiver182S = new TEXTBOX(251, 351, 59, 14);
textboxes.add(receiver182S);
TEXTBOX receiver192S = new TEXTBOX(251, 366, 59, 14);
textboxes.add(receiver192S);
TEXTBOX receiver202S = new TEXTBOX(251, 381, 59, 14);
textboxes.add(receiver202S);
TEXTBOX receiver212S = new TEXTBOX(251, 396, 59, 14);
textboxes.add(receiver212S);
TEXTBOX receiver222S = new TEXTBOX(251, 411, 59, 14);
textboxes.add(receiver222S);
TEXTBOX receiver232S = new TEXTBOX(251, 426, 59, 14);
textboxes.add(receiver232S);
TEXTBOX receiver242S = new TEXTBOX(251, 441, 59, 14);
textboxes.add(receiver242S);
TEXTBOX receiver252S = new TEXTBOX(251, 456, 59, 14);
textboxes.add(receiver252S);
TEXTBOX receiver262S = new TEXTBOX(251, 471, 59, 14);
textboxes.add(receiver262S);
TEXTBOX receiver272S = new TEXTBOX(251, 486, 59, 14);
textboxes.add(receiver272S);
TEXTBOX receiver282S = new TEXTBOX(251, 501, 59, 14);
textboxes.add(receiver282S);
TEXTBOX receiver292S = new TEXTBOX(251, 516, 59, 14);
textboxes.add(receiver292S);
```

```
TEXTBOX receiver302S = new TEXTBOX(251, 531, 59, 14);
textboxes.add(receiver302S);
TEXTBOX receiver312S = new TEXTBOX(251, 546, 59, 14);
textboxes.add(receiver312S);
///Strut 3///
//Columna A//
TEXTBOX receiver03 = new TEXTBOX(311, 81, 29, 14);
textboxes.add(receiver03);
TEXTBOX receiver13 = new TEXTBOX(311, 96, 29, 14);
textboxes.add(receiver13);
TEXTBOX receiver23 = new TEXTBOX(311, 111, 29, 14);
textboxes.add(receiver23);
TEXTBOX receiver33 = new TEXTBOX(311, 126, 29, 14);
textboxes.add(receiver33);
TEXTBOX receiver43 = new TEXTBOX(311, 141, 29, 14);
textboxes.add(receiver43);
TEXTBOX receiver53 = new TEXTBOX(311, 156, 29, 14);
textboxes.add(receiver53);
TEXTBOX receiver63 = new TEXTBOX(311, 171, 29, 14);
textboxes.add(receiver63);
TEXTBOX receiver73 = new TEXTBOX(311, 186, 29, 14);
textboxes.add(receiver73);
TEXTBOX receiver83 = new TEXTBOX(311, 201, 29, 14);
textboxes.add(receiver83);
TEXTBOX receiver93 = new TEXTBOX(311, 216, 29, 14);
textboxes.add(receiver93);
TEXTBOX receiver103 = new TEXTBOX(311, 231, 29, 14);
textboxes.add(receiver103);
TEXTBOX receiver113 = new TEXTBOX(311, 246, 29, 14);
textboxes.add(receiver113);
TEXTBOX receiver123 = new TEXTBOX(311, 261, 29, 14);
textboxes.add(receiver123);
TEXTBOX receiver133 = new TEXTBOX(311, 276, 29, 14);
textboxes.add(receiver133);
TEXTBOX receiver143 = new TEXTBOX(311, 291, 29, 14);
textboxes.add(receiver143);
TEXTBOX receiver153 = new TEXTBOX(311, 306, 29, 14);
textboxes.add(receiver153);
TEXTBOX receiver163 = new TEXTBOX(311, 321, 29, 14);
textboxes.add(receiver163);
TEXTBOX receiver173 = new TEXTBOX(311, 336, 29, 14);
textboxes.add(receiver173);
TEXTBOX receiver183 = new TEXTBOX(311, 351, 29, 14);
textboxes.add(receiver183);
TEXTBOX receiver193 = new TEXTBOX(311, 366, 29, 14);
textboxes.add(receiver193);
TEXTBOX receiver203 = new TEXTBOX(311, 381, 29, 14);
textboxes.add(receiver203);
TEXTBOX receiver213 = new TEXTBOX(311, 396, 29, 14);
textboxes.add(receiver213);
TEXTBOX receiver223 = new TEXTBOX(311, 411, 29, 14);
textboxes.add(receiver223);
TEXTBOX receiver233 = new TEXTBOX(311, 426, 29, 14);
```

```
textboxes.add(receiver233);
TEXTBOX receiver243 = new TEXTBOX(311, 441, 29, 14);
textboxes.add(receiver243);
TEXTBOX receiver253 = new TEXTBOX(311, 456, 29, 14);
textboxes.add(receiver253);
TEXTBOX receiver263 = new TEXTBOX(311, 471, 29, 14);
textboxes.add(receiver263);
TEXTBOX receiver273 = new TEXTBOX(311, 486, 29, 14);
textboxes.add(receiver273);
TEXTBOX receiver283 = new TEXTBOX(311, 501, 29, 14);
textboxes.add(receiver283);
TEXTBOX receiver293 = new TEXTBOX(311, 516, 29, 14);
textboxes.add(receiver293);
TEXTBOX receiver303 = new TEXTBOX(311, 531, 29, 14);
textboxes.add(receiver303);
TEXTBOX receiver313 = new TEXTBOX(311, 546, 29, 14);
textboxes.add(receiver313);
//Columna Size//
TEXTBOX receiver03S = new TEXTBOX(371, 81, 59, 14);
textboxes.add(receiver03S);
TEXTBOX receiver13S = new TEXTBOX(371, 96, 59, 14);
textboxes.add(receiver13S);
TEXTBOX receiver23S = new TEXTBOX(371, 111, 59, 14);
textboxes.add(receiver23S);
TEXTBOX receiver33S = new TEXTBOX(371, 126, 59, 14);
textboxes.add(receiver33S);
TEXTBOX receiver43S = new TEXTBOX(371, 141, 59, 14);
textboxes.add(receiver43S);
TEXTBOX receiver53S = new TEXTBOX(371, 156, 59, 14);
textboxes.add(receiver53S);
TEXTBOX receiver63S = new TEXTBOX(371, 171, 59, 14);
textboxes.add(receiver63S);
TEXTBOX receiver73S = new TEXTBOX(371, 186, 59, 14);
textboxes.add(receiver73S);
TEXTBOX receiver83S = new TEXTBOX(371, 201, 59, 14);
textboxes.add(receiver83S);
TEXTBOX receiver93S = new TEXTBOX(371, 216, 59, 14);
textboxes.add(receiver93S);
TEXTBOX receiver103S = new TEXTBOX(371, 231, 59, 14);
textboxes.add(receiver103S);
TEXTBOX receiver113S = new TEXTBOX(371, 246, 59, 14);
textboxes.add(receiver113S);
TEXTBOX receiver123S = new TEXTBOX(371, 261, 59, 14);
textboxes.add(receiver123S);
TEXTBOX receiver133S = new TEXTBOX(371, 276, 59, 14);
textboxes.add(receiver133S);
TEXTBOX receiver143S = new TEXTBOX(371, 291, 59, 14);
textboxes.add(receiver143S);
TEXTBOX receiver153S = new TEXTBOX(371, 306, 59, 14);
textboxes.add(receiver153S);
TEXTBOX receiver163S = new TEXTBOX(371, 321, 59, 14);
textboxes.add(receiver163S);
TEXTBOX receiver173S = new TEXTBOX(371, 336, 59, 14);
```

```
textboxes.add(receiver173S);
TEXTBOX receiver183S = new TEXTBOX(371, 351, 59, 14);
textboxes.add(receiver183S);
TEXTBOX receiver193S = new TEXTBOX(371, 366, 59, 14);
textboxes.add(receiver193S);
TEXTBOX receiver203S = new TEXTBOX(371, 381, 59, 14);
textboxes.add(receiver203S);
TEXTBOX receiver213S = new TEXTBOX(371, 396, 59, 14);
textboxes.add(receiver213S);
TEXTBOX receiver223S = new TEXTBOX(371, 411, 59, 14);
textboxes.add(receiver223S);
TEXTBOX receiver233S = new TEXTBOX(371, 426, 59, 14);
textboxes.add(receiver233S);
TEXTBOX receiver243S = new TEXTBOX(371, 441, 59, 14);
textboxes.add(receiver243S);
TEXTBOX receiver253S = new TEXTBOX(371, 456, 59, 14);
textboxes.add(receiver253S);
TEXTBOX receiver263S = new TEXTBOX(371, 471, 59, 14);
textboxes.add(receiver263S);
TEXTBOX receiver273S = new TEXTBOX(371, 486, 59, 14);
textboxes.add(receiver273S);
TEXTBOX receiver283S = new TEXTBOX(371, 501, 59, 14);
textboxes.add(receiver283S);
TEXTBOX receiver293S = new TEXTBOX(371, 516, 59, 14);
textboxes.add(receiver293S);
TEXTBOX receiver303S = new TEXTBOX(371, 531, 59, 14);
textboxes.add(receiver303S);
TEXTBOX receiver313S = new TEXTBOX(371, 546, 59, 14);
textboxes.add(receiver313S);
///Strut 4///
//Columna A//
TEXTBOX receiver04 = new TEXTBOX(431, 81, 29, 14);
textboxes.add(receiver04);
TEXTBOX receiver14 = new TEXTBOX(431, 96, 29, 14);
textboxes.add(receiver14);
TEXTBOX receiver24 = new TEXTBOX(431, 111, 29, 14);
textboxes.add(receiver24);
TEXTBOX receiver34 = new TEXTBOX(431, 126, 29, 14);
textboxes.add(receiver34);
TEXTBOX receiver44 = new TEXTBOX(431, 141, 29, 14);
textboxes.add(receiver44);
TEXTBOX receiver54 = new TEXTBOX(431, 156, 29, 14);
textboxes.add(receiver54);
TEXTBOX receiver64 = new TEXTBOX(431, 171, 29, 14);
textboxes.add(receiver64);
TEXTBOX receiver74 = new TEXTBOX(431, 186, 29, 14);
textboxes.add(receiver74);
TEXTBOX receiver84 = new TEXTBOX(431, 201, 29, 14);
textboxes.add(receiver84);
TEXTBOX receiver94 = new TEXTBOX(431, 216, 29, 14);
textboxes.add(receiver94);
TEXTBOX receiver104 = new TEXTBOX(431, 231, 29, 14);
textboxes.add(receiver104);
```

```
TEXTBOX receiver114 = new TEXTBOX(431, 246, 29, 14);
textboxes.add(receiver114);
TEXTBOX receiver124 = new TEXTBOX(431, 261, 29, 14);
textboxes.add(receiver124);
TEXTBOX receiver134 = new TEXTBOX(431, 276, 29, 14);
textboxes.add(receiver134);
TEXTBOX receiver144 = new TEXTBOX(431, 291, 29, 14);
textboxes.add(receiver144);
TEXTBOX receiver154 = new TEXTBOX(431, 306, 29, 14);
textboxes.add(receiver154);
TEXTBOX receiver164 = new TEXTBOX(431, 321, 29, 14);
textboxes.add(receiver164);
TEXTBOX receiver174 = new TEXTBOX(431, 336, 29, 14);
textboxes.add(receiver174);
TEXTBOX receiver184 = new TEXTBOX(431, 351, 29, 14);
textboxes.add(receiver184);
TEXTBOX receiver194 = new TEXTBOX(431, 366, 29, 14);
textboxes.add(receiver194);
TEXTBOX receiver204 = new TEXTBOX(431, 381, 29, 14);
textboxes.add(receiver204);
TEXTBOX receiver214 = new TEXTBOX(431, 396, 29, 14);
textboxes.add(receiver214);
TEXTBOX receiver224 = new TEXTBOX(431, 411, 29, 14);
textboxes.add(receiver224);
TEXTBOX receiver234 = new TEXTBOX(431, 426, 29, 14);
textboxes.add(receiver234);
TEXTBOX receiver244 = new TEXTBOX(431, 441, 29, 14);
textboxes.add(receiver244);
TEXTBOX receiver254 = new TEXTBOX(431, 456, 29, 14);
textboxes.add(receiver254);
TEXTBOX receiver264 = new TEXTBOX(431, 471, 29, 14);
textboxes.add(receiver264);
TEXTBOX receiver274 = new TEXTBOX(431, 486, 29, 14);
textboxes.add(receiver274);
TEXTBOX receiver284 = new TEXTBOX(431, 501, 29, 14);
textboxes.add(receiver284);
TEXTBOX receiver294 = new TEXTBOX(431, 516, 29, 14);
textboxes.add(receiver294);
TEXTBOX receiver304 = new TEXTBOX(431, 531, 29, 14);
textboxes.add(receiver304);
TEXTBOX receiver314 = new TEXTBOX(431, 546, 29, 14);
textboxes.add(receiver314);
//Columna Size//
TEXTBOX receiver04S = new TEXTBOX(491, 81, 59, 14);
textboxes.add(receiver04S);
TEXTBOX receiver14S = new TEXTBOX(491, 96, 59, 14);
textboxes.add(receiver14S);
TEXTBOX receiver24S = new TEXTBOX(491, 111, 59, 14);
textboxes.add(receiver24S);
TEXTBOX receiver34S = new TEXTBOX(491, 126, 59, 14);
textboxes.add(receiver34S);
TEXTBOX receiver44S = new TEXTBOX(491, 141, 59, 14);
textboxes.add(receiver44S);
```

```
TEXTBOX receiver54S = new TEXTBOX(491, 156, 59, 14);
textboxes.add(receiver54S);
TEXTBOX receiver64S = new TEXTBOX(491, 171, 59, 14);
textboxes.add(receiver64S);
TEXTBOX receiver74S = new TEXTBOX(491, 186, 59, 14);
textboxes.add(receiver74S);
TEXTBOX receiver84S = new TEXTBOX(491, 201, 59, 14);
textboxes.add(receiver84S);
TEXTBOX receiver94S = new TEXTBOX(491, 216, 59, 14);
textboxes.add(receiver94S);
TEXTBOX receiver104S = new TEXTBOX(491, 231, 59, 14);
textboxes.add(receiver104S);
TEXTBOX receiver114S = new TEXTBOX(491, 246, 59, 14);
textboxes.add(receiver114S);
TEXTBOX receiver124S = new TEXTBOX(491, 261, 59, 14);
textboxes.add(receiver124S);
TEXTBOX receiver134S = new TEXTBOX(491, 276, 59, 14);
textboxes.add(receiver134S);
TEXTBOX receiver144S = new TEXTBOX(491, 291, 59, 14);
textboxes.add(receiver144S);
TEXTBOX receiver154S = new TEXTBOX(491, 306, 59, 14);
textboxes.add(receiver154S);
TEXTBOX receiver164S = new TEXTBOX(491, 321, 59, 14);
textboxes.add(receiver164S);
TEXTBOX receiver174S = new TEXTBOX(491, 336, 59, 14);
textboxes.add(receiver174S);
TEXTBOX receiver184S = new TEXTBOX(491, 351, 59, 14);
textboxes.add(receiver184S);
TEXTBOX receiver194S = new TEXTBOX(491, 366, 59, 14);
textboxes.add(receiver194S);
TEXTBOX receiver204S = new TEXTBOX(491, 381, 59, 14);
textboxes.add(receiver204S);
TEXTBOX receiver214S = new TEXTBOX(491, 396, 59, 14);
textboxes.add(receiver214S);
TEXTBOX receiver224S = new TEXTBOX(491, 411, 59, 14);
textboxes.add(receiver224S);
TEXTBOX receiver234S = new TEXTBOX(491, 426, 59, 14);
textboxes.add(receiver234S);
TEXTBOX receiver244S = new TEXTBOX(491, 441, 59, 14);
textboxes.add(receiver244S);
TEXTBOX receiver254S = new TEXTBOX(491, 456, 59, 14);
textboxes.add(receiver254S);
TEXTBOX receiver264S = new TEXTBOX(491, 471, 59, 14);
textboxes.add(receiver264S);
TEXTBOX receiver274S = new TEXTBOX(491, 486, 59, 14);
textboxes.add(receiver274S);
TEXTBOX receiver284S = new TEXTBOX(491, 501, 59, 14);
textboxes.add(receiver284S);
TEXTBOX receiver294S = new TEXTBOX(491, 516, 59, 14);
textboxes.add(receiver294S);
TEXTBOX receiver304S = new TEXTBOX(491, 531, 59, 14);
textboxes.add(receiver304S);
TEXTBOX receiver314S = new TEXTBOX(491, 546, 59, 14);
```

```
textboxes.add(receiver314S);
///Strut 5///
//Columna A//
TEXTBOX receiver05 = new TEXTBOX(551, 81, 29, 14);
textboxes.add(receiver05);
TEXTBOX receiver15 = new TEXTBOX(551, 96, 29, 14);
textboxes.add(receiver15);
TEXTBOX receiver25 = new TEXTBOX(551, 111, 29, 14);
textboxes.add(receiver25);
TEXTBOX receiver35 = new TEXTBOX(551, 126, 29, 14);
textboxes.add(receiver35);
TEXTBOX receiver45 = new TEXTBOX(551, 141, 29, 14);
textboxes.add(receiver45);
TEXTBOX receiver55 = new TEXTBOX(551, 156, 29, 14);
textboxes.add(receiver55);
TEXTBOX receiver65 = new TEXTBOX(551, 171, 29, 14);
textboxes.add(receiver65);
TEXTBOX receiver75 = new TEXTBOX(551, 186, 29, 14);
textboxes.add(receiver75);
TEXTBOX receiver85 = new TEXTBOX(551, 201, 29, 14);
textboxes.add(receiver85);
TEXTBOX receiver95 = new TEXTBOX(551, 216, 29, 14);
textboxes.add(receiver95);
TEXTBOX receiver105 = new TEXTBOX(551, 231, 29, 14);
textboxes.add(receiver105);
TEXTBOX receiver115 = new TEXTBOX(551, 246, 29, 14);
textboxes.add(receiver115);
TEXTBOX receiver125 = new TEXTBOX(551, 261, 29, 14);
textboxes.add(receiver125);
TEXTBOX receiver135 = new TEXTBOX(551, 276, 29, 14);
textboxes.add(receiver135);
TEXTBOX receiver145 = new TEXTBOX(551, 291, 29, 14);
textboxes.add(receiver145);
TEXTBOX receiver155 = new TEXTBOX(551, 306, 29, 14);
textboxes.add(receiver155);
TEXTBOX receiver165 = new TEXTBOX(551, 321, 29, 14);
textboxes.add(receiver165);
TEXTBOX receiver175 = new TEXTBOX(551, 336, 29, 14);
textboxes.add(receiver175);
TEXTBOX receiver185 = new TEXTBOX(551, 351, 29, 14);
textboxes.add(receiver185);
TEXTBOX receiver195 = new TEXTBOX(551, 366, 29, 14);
textboxes.add(receiver195);
TEXTBOX receiver205 = new TEXTBOX(551, 381, 29, 14);
textboxes.add(receiver205);
TEXTBOX receiver215 = new TEXTBOX(551, 396, 29, 14);
textboxes.add(receiver215);
TEXTBOX receiver225 = new TEXTBOX(551, 411, 29, 14);
textboxes.add(receiver225);
TEXTBOX receiver235 = new TEXTBOX(551, 426, 29, 14);
textboxes.add(receiver235);
TEXTBOX receiver245 = new TEXTBOX(551, 441, 29, 14);
textboxes.add(receiver245);
```

```
TEXTBOX receiver255 = new TEXTBOX(551, 456, 29, 14);
textboxes.add(receiver255);
TEXTBOX receiver265 = new TEXTBOX(551, 471, 29, 14);
textboxes.add(receiver265);
TEXTBOX receiver275 = new TEXTBOX(551, 486, 29, 14);
textboxes.add(receiver275);
TEXTBOX receiver285 = new TEXTBOX(551, 501, 29, 14);
textboxes.add(receiver285);
TEXTBOX receiver295 = new TEXTBOX(551, 516, 29, 14);
textboxes.add(receiver295);
TEXTBOX receiver305 = new TEXTBOX(551, 531, 29, 14);
textboxes.add(receiver305);
TEXTBOX receiver315 = new TEXTBOX(551, 546, 29, 14);
textboxes.add(receiver315);
//Columna Size//
TEXTBOX receiver05S = new TEXTBOX(611, 81, 59, 14);
textboxes.add(receiver05S);
TEXTBOX receiver15S = new TEXTBOX(611, 96, 59, 14);
textboxes.add(receiver15S);
TEXTBOX receiver25S = new TEXTBOX(611, 111, 59, 14);
textboxes.add(receiver25S);
TEXTBOX receiver35S = new TEXTBOX(611, 126, 59, 14);
textboxes.add(receiver35S);
TEXTBOX receiver45S = new TEXTBOX(611, 141, 59, 14);
textboxes.add(receiver45S);
TEXTBOX receiver55S = new TEXTBOX(611, 156, 59, 14);
textboxes.add(receiver55S);
TEXTBOX receiver65S = new TEXTBOX(611, 171, 59, 14);
textboxes.add(receiver65S);
TEXTBOX receiver75S = new TEXTBOX(611, 186, 59, 14);
textboxes.add(receiver75S);
TEXTBOX receiver85S = new TEXTBOX(611, 201, 59, 14);
textboxes.add(receiver85S);
TEXTBOX receiver95S = new TEXTBOX(611, 216, 59, 14);
textboxes.add(receiver95S);
TEXTBOX receiver105S = new TEXTBOX(611, 231, 59, 14);
textboxes.add(receiver105S);
TEXTBOX receiver115S = new TEXTBOX(611, 246, 59, 14);
textboxes.add(receiver115S);
TEXTBOX receiver125S = new TEXTBOX(611, 261, 59, 14);
textboxes.add(receiver125S);
TEXTBOX receiver135S = new TEXTBOX(611, 276, 59, 14);
textboxes.add(receiver135S);
TEXTBOX receiver145S = new TEXTBOX(611, 291, 59, 14);
textboxes.add(receiver145S);
TEXTBOX receiver155S = new TEXTBOX(611, 306, 59, 14);
textboxes.add(receiver155S);
TEXTBOX receiver165S = new TEXTBOX(611, 321, 59, 14);
textboxes.add(receiver165S);
TEXTBOX receiver175S = new TEXTBOX(611, 336, 59, 14);
textboxes.add(receiver175S);
TEXTBOX receiver185S = new TEXTBOX(611, 351, 59, 14);
textboxes.add(receiver185S);
```

```
TEXTBOX receiver195S = new TEXTBOX(611, 366, 59, 14);
textboxes.add(receiver195S);
TEXTBOX receiver205S = new TEXTBOX(611, 381, 59, 14);
textboxes.add(receiver205S);
TEXTBOX receiver215S = new TEXTBOX(611, 396, 59, 14);
textboxes.add(receiver215S);
TEXTBOX receiver225S = new TEXTBOX(611, 411, 59, 14);
textboxes.add(receiver225S);
TEXTBOX receiver235S = new TEXTBOX(611, 426, 59, 14);
textboxes.add(receiver235S);
TEXTBOX receiver245S = new TEXTBOX(611, 441, 59, 14);
textboxes.add(receiver245S);
TEXTBOX receiver255S = new TEXTBOX(611, 456, 59, 14);
textboxes.add(receiver255S);
TEXTBOX receiver265S = new TEXTBOX(611, 471, 59, 14);
textboxes.add(receiver265S);
TEXTBOX receiver275S = new TEXTBOX(611, 486, 59, 14);
textboxes.add(receiver275S);
TEXTBOX receiver285S = new TEXTBOX(611, 501, 59, 14);
textboxes.add(receiver285S);
TEXTBOX receiver295S = new TEXTBOX(611, 516, 59, 14);
textboxes.add(receiver295S);
TEXTBOX receiver305S = new TEXTBOX(611, 531, 59, 14);
textboxes.add(receiver305S);
TEXTBOX receiver315S = new TEXTBOX(611, 546, 59, 14);
textboxes.add(receiver315S);
///Strut 5///
//Columna A//
TEXTBOX receiver06 = new TEXTBOX(671, 81, 29, 14);
textboxes.add(receiver06);
TEXTBOX receiver16 = new TEXTBOX(671, 96, 29, 14);
textboxes.add(receiver16);
TEXTBOX receiver26 = new TEXTBOX(671, 111, 29, 14);
textboxes.add(receiver26);
TEXTBOX receiver36 = new TEXTBOX(671, 126, 29, 14);
textboxes.add(receiver36);
TEXTBOX receiver46 = new TEXTBOX(671, 141, 29, 14);
textboxes.add(receiver46);
TEXTBOX receiver56 = new TEXTBOX(671, 156, 29, 14);
textboxes.add(receiver56);
TEXTBOX receiver66 = new TEXTBOX(671, 171, 29, 14);
textboxes.add(receiver66);
TEXTBOX receiver76 = new TEXTBOX(671, 186, 29, 14);
textboxes.add(receiver76);
TEXTBOX receiver86 = new TEXTBOX(671, 201, 29, 14);
textboxes.add(receiver86);
TEXTBOX receiver96 = new TEXTBOX(671, 216, 29, 14);
textboxes.add(receiver96);
TEXTBOX receiver106 = new TEXTBOX(671, 231, 29, 14);
textboxes.add(receiver106);
TEXTBOX receiver116 = new TEXTBOX(671, 246, 29, 14);
textboxes.add(receiver116);
TEXTBOX receiver126 = new TEXTBOX(671, 261, 29, 14);
```

```
textboxes.add(receiver126);
TEXTBOX receiver136 = new TEXTBOX(671, 276, 29, 14);
textboxes.add(receiver136);
TEXTBOX receiver146 = new TEXTBOX(671, 291, 29, 14);
textboxes.add(receiver146);
TEXTBOX receiver156 = new TEXTBOX(671, 306, 29, 14);
textboxes.add(receiver156);
TEXTBOX receiver166 = new TEXTBOX(671, 321, 29, 14);
textboxes.add(receiver166);
TEXTBOX receiver176 = new TEXTBOX(671, 336, 29, 14);
textboxes.add(receiver176);
TEXTBOX receiver186 = new TEXTBOX(671, 351, 29, 14);
textboxes.add(receiver186);
TEXTBOX receiver196 = new TEXTBOX(671, 366, 29, 14);
textboxes.add(receiver196);
TEXTBOX receiver206 = new TEXTBOX(671, 381, 29, 14);
textboxes.add(receiver206);
TEXTBOX receiver216 = new TEXTBOX(671, 396, 29, 14);
textboxes.add(receiver216);
TEXTBOX receiver226 = new TEXTBOX(671, 411, 29, 14);
textboxes.add(receiver226);
TEXTBOX receiver236 = new TEXTBOX(671, 426, 29, 14);
textboxes.add(receiver236);
TEXTBOX receiver246 = new TEXTBOX(671, 441, 29, 14);
textboxes.add(receiver246);
TEXTBOX receiver256 = new TEXTBOX(671, 456, 29, 14);
textboxes.add(receiver256);
TEXTBOX receiver266 = new TEXTBOX(671, 471, 29, 14);
textboxes.add(receiver266);
TEXTBOX receiver276 = new TEXTBOX(671, 486, 29, 14);
textboxes.add(receiver276);
TEXTBOX receiver286 = new TEXTBOX(671, 501, 29, 14);
textboxes.add(receiver286);
TEXTBOX receiver296 = new TEXTBOX(671, 516, 29, 14);
textboxes.add(receiver296);
TEXTBOX receiver306 = new TEXTBOX(671, 531, 29, 14);
textboxes.add(receiver306);
TEXTBOX receiver316 = new TEXTBOX(671, 546, 29, 14);
textboxes.add(receiver316);
//Columna Size//
TEXTBOX receiver06S = new TEXTBOX(731, 81, 59, 14);
textboxes.add(receiver06S);
TEXTBOX receiver16S = new TEXTBOX(731, 96, 59, 14);
textboxes.add(receiver16S);
TEXTBOX receiver26S = new TEXTBOX(731, 111, 59, 14);
textboxes.add(receiver26S);
TEXTBOX receiver36S = new TEXTBOX(731, 126, 59, 14);
textboxes.add(receiver36S);
TEXTBOX receiver46S = new TEXTBOX(731, 141, 59, 14);
textboxes.add(receiver46S);
TEXTBOX receiver56S = new TEXTBOX(731, 156, 59, 14);
textboxes.add(receiver56S);
TEXTBOX receiver66S = new TEXTBOX(731, 171, 59, 14);
```

```
  textboxes.add(receiver66S);
  TEXTBOX receiver76S = new TEXTBOX(731, 186, 59, 14);
  textboxes.add(receiver76S);
  TEXTBOX receiver86S = new TEXTBOX(731, 201, 59, 14);
  textboxes.add(receiver86S);
  TEXTBOX receiver96S = new TEXTBOX(731, 216, 59, 14);
  textboxes.add(receiver96S);
  TEXTBOX receiver106S = new TEXTBOX(731, 231, 59, 14);
  textboxes.add(receiver106S);
  TEXTBOX receiver116S = new TEXTBOX(731, 246, 59, 14);
  textboxes.add(receiver116S);
  TEXTBOX receiver126S = new TEXTBOX(731, 261, 59, 14);
  textboxes.add(receiver126S);
  TEXTBOX receiver136S = new TEXTBOX(731, 276, 59, 14);
  textboxes.add(receiver136S);
  TEXTBOX receiver146S = new TEXTBOX(731, 291, 59, 14);
  textboxes.add(receiver146S);
  TEXTBOX receiver156S = new TEXTBOX(731, 306, 59, 14);
  textboxes.add(receiver156S);
  TEXTBOX receiver166S = new TEXTBOX(731, 321, 59, 14);
  textboxes.add(receiver166S);
  TEXTBOX receiver176S = new TEXTBOX(731, 336, 59, 14);
  textboxes.add(receiver176S);
  TEXTBOX receiver186S = new TEXTBOX(731, 351, 59, 14);
  textboxes.add(receiver186S);
  TEXTBOX receiver196S = new TEXTBOX(731, 366, 59, 14);
  textboxes.add(receiver196S);
  TEXTBOX receiver206S = new TEXTBOX(731, 381, 59, 14);
  textboxes.add(receiver206S);
  TEXTBOX receiver216S = new TEXTBOX(731, 396, 59, 14);
  textboxes.add(receiver216S);
  TEXTBOX receiver226S = new TEXTBOX(731, 411, 59, 14);
  textboxes.add(receiver226S);
  TEXTBOX receiver236S = new TEXTBOX(731, 426, 59, 14);
  textboxes.add(receiver236S);
  TEXTBOX receiver246S = new TEXTBOX(731, 441, 59, 14);
  textboxes.add(receiver246S);
  TEXTBOX receiver256S = new TEXTBOX(731, 456, 59, 14);
  textboxes.add(receiver256S);
  TEXTBOX receiver266S = new TEXTBOX(731, 471, 59, 14);
  textboxes.add(receiver266S);
  TEXTBOX receiver276S = new TEXTBOX(731, 486, 59, 14);
  textboxes.add(receiver276S);
  TEXTBOX receiver286S = new TEXTBOX(731, 501, 59, 14);
  textboxes.add(receiver286S);
  TEXTBOX receiver296S = new TEXTBOX(731, 516, 59, 14);
  textboxes.add(receiver296S);
  TEXTBOX receiver306S = new TEXTBOX(731, 531, 59, 14);
  textboxes.add(receiver306S);
  TEXTBOX receiver316S = new TEXTBOX(731, 546, 59, 14);
  textboxes.add(receiver316S);
}
```

Segunda pestaña

```
public class TEXTBOX {
  public int X = 0, Y = 0, H = 35, W = 200;
  public int TEXTSIZE = 11;
  // COLORS
  public color Background = color(#FFFFFF);
  public color Foreground = color(0, 0, 0);
  public color BackgroundSelected = color(#EDE1E1);
  public color Border = color(30, 30, 30);
  public boolean BorderEnable = false;
  public int BorderWeight = 1;
  public String Text = "";
  public int TextLength = 0;
  private boolean selected = false;
  TEXTBOX() {
    // CREATE OBJECT DEFAULT TEXTBOX
  }

  TEXTBOX(int x, int y, int w, int h) {
    X = x; Y = y; W = w; H = h;
  }

  void DRAW() {
    // DRAWING THE BACKGROUND
    if (selected) {
      fill(BackgroundSelected);
    } else {
      fill(Background);
    }
    if (BorderEnable) {
      strokeWeight(BorderWeight);
      stroke(Border);
    } else {
      noStroke();
    }
    rect(X, Y, W, H);
    // DRAWING THE TEXT ITSELF
    fill(Foreground);
    textSize(TEXTSIZE);
    text(Text, X + 3 + (textWidth("a") / 2), Y + TEXTSIZE);
  }
  // IF THE KEYCODE IS ENTER RETURN 1
  // ELSE RETURN 0
  boolean KEYPRESSED(char KEY, int KEYCODE) {
    if (selected) {
      if (KEYCODE == (int)BACKSPACE) {
        BACKSPACE();
      } else if (KEYCODE == 32) {
        // SPACE
        addText(' ');
      } else if (KEYCODE == (int)ENTER) {
```

```
          return true;
        } else {
          // CHECK IF THE KEY IS A LETTER OR A NUMBER
          boolean isKeyCapitalLetter = (KEY >= 'A' && KEY <= 'Z');
          boolean isKeySmallLetter = (KEY >= 'a' && KEY <= 'z');
          boolean isKeyNumber = (KEY >= '0' && KEY <= '9');
          if (isKeyCapitalLetter || isKeySmallLetter || isKeyNumber) {
            addText(KEY);
          }
        }
      }
      return false;
    }
    private void addText(char text) {
      // IF THE TEXT WIDHT IS IN BOUNDARIES OF THE TEXTBOX
      if (textWidth(Text + text) < W+5) {
        Text += text;
        TextLength++;
      }
    }

    private void BACKSPACE() {
      if (TextLength - 1 >= 0) {
        Text = Text.substring(0, TextLength - 1);
        TextLength--;
      }
    }
    // FUNCTION FOR TESTING IF THE POINT IS
    // OVER THE TEXTBOX
    private boolean overBox(int x, int y) {
      if (x >= X && x <= X + W) {
        if (y >= Y && y <= Y + H) {
          return true;
        }
      }
      return false;
    }

    void PRESSED(int x, int y) {
      if (overBox(x, y)) {
        selected = true;
      } else {
        selected = false;
      }
    }
}
```

Bibliografía

[1]	Laverdad[en línea][consulta: 8 enero 2018]
	Disponible en: http://www.laverdad.es/murcia/arrixaca-incorpora-tecnica-20180108003227-ntvo.html
[2]	Researchgate[en línea][consulta: 12 octubre 2017].
	Disponible en: https://www.researchgate.net/publication/8454173_Robot-Assisted_Orthopedic_Surgery
[3]	Tlhex[en línea][consulta: 18 abril 2018]
	Disponible en: http://www.tlhex.com/#product-information
[4]	Tlhex[en línea][consulta: 19 abril 2018]
	Disponible en: http://web.orthofix.com/Products/Products/TL-HEX/TL-1412-PL-E0.pdf
[5]	ARACIL, Rafael; SALTARÉN, Roque; SABATER, José; REINOSO, Oscar; Robots paralelos: Máquinas con un pasado para una robótica del futuro[en línea] ISSN: 1697-7912. Vol.3, Núm. 1, Enero 2006, pp. 16-28[Consulta: 1 mayo 2018]
	Disponible en: https://www.researchgate.net/publication/28141949_Robots_Paralelos_Maquinas_con_un_Pasado_para_una_Robotica_del_Futuro
[6]	BOE-A-2007-2648: por la que se aprueba y publica el programa formativo de la especialidad de Cirugía Ortopédica y Traumatología. [en línea][consulta: 15 abril 2018]
	Disponible en: http://www.boe.es/buscar/doc.php?id=BOE-A-2007-2648
[7]	Idus.us[en línea][29 abril 2018]
	Disponible en: https://idus.us.es/xmlui/bitstream/handle/11441/41205/TESIS%20DOCTORAL%20Osteog%C3%A9nesis%20por%20Distracci%C3%B3n%20con%20Mantenimiento%20de%20la%20actividad%20motriz.pdf?sequence=1
[8]	Pdfs.semanticscholar[en línea][29 abril 2018]
	Disponible en: https://pdfs.semanticscholar.org/c9d3/08b9ffbe61e8e42acf874c1f78d3d120805e.pdf
[9]	Dovepress[en línea][29 abril 2018]
	Disponible en: https://www.dovepress.com/clinical-utility-of-the-taylor-spatial-frame-for-limb-deformities-peer-reviewed-fulltext-article-ORR
[10]	GIRONE, M; BURDEA, G; BOUZIT, M; POPESCU, V; DEUTSCH, J.E. A Stewart Platform-Based System for Ankle Telerehabilitation[en línea]. Marzo 2011, Volumen 10, Tema 2, 203-212.
	Disponible en: https://link.springer.com/article/10.1023%2FA%3A1008938121020?LI=true
[11]	Micromo[en línea][Consulta: 12 octubre 2017]
	Disponible en: https://www.micromo.com/applications/medical-lab-automation-equipment/application-case-study-micro-precise-surgery-assistant
[12]	Orthofix[en línea][Consulta: 19 abril 2018]
	Disponible en: http://web.orthofix.com/Pages/IFU.aspx
[13]	Redeweb[en línea][consulta: 12 octubre 2017]
	Disponible en: http://www.redeweb.com/articulos/instrumentacion/diseno-y-construccion-de-un-prototipo-de-plataforma-stewart/

[14]	Limbhealing[en línea][consulta: 19 abril 2018]
	Disponible en: https://limbhealing.com/tl-hex/
[15]	Wikiversity[en línea][Consulta: 22 abril 2018]
	Disponible en: https://es.wikiversity.org/wiki/Terminolog%C3%ADa_anat%C3%B3mica
[16]	Youtube[en línea][Consulta: 23 abril 2018]
	Disponible en : https://www.youtube.com/watch?v=XeuTWjfCvYc
[17]	Panamahitek[en línea][Consulta: 21 octube 2017]
	Disponible en: http://panamahitek.com/que-es-y-como-funciona-un-potenciometro/
[18]	Playground[en línea][Consulta: 21 octube 2017]
	Disponible en: https://playground.arduino.cc/ArduinoNotebookTraduccion/Appendix4
[19]	Ahoramededicoalcacharreo[en línea][Consulta: 21 octube 2017]
	Disponible en: http://ahoramededicoalcacharreo.blogspot.com.es/2014/10/analogread-con-arduino-error-al-leer.html
[20]	Lidar-uk[en línea][Consulta: 21 octubre 2017]
	Disponible en: http://www.lidar-uk.com/how-lidar-works/
[21]	Aprendiendoarduino[en línea][Consulta: 22 octubre 2017]
	Disponible en: https://aprendiendoarduino.wordpress.com/2016/11/14/bus-i2ctwi/
[22]	Luisllamas[en línea][Consulta: 22 octubre 2017]
	Disponible en: https://www.luisllamas.es/arduino-i2c/
[23]	Luisllamas[en línea][Consulta: 22 octubre 2017]
	Disponible en: https://www.luisllamas.es/arduino-paso-bajo-exponencial/
[24]	Luisllamas[en línea][Consulta: 22 octubre 2017]
	Disponible en: https://www.luisllamas.es/arduino-filtro-mediana-rapido/
[25]	Wiki.seeed[en línea][Consulta: 24 octubre 2017]
	Disponible en: http://wiki.seeed.cc/Ultra_Sonic_range_measurement_module/
[26]	Picmania[en línea][Consulta: 24 octubre 2017]
	Disponible en: http://picmania.garcia-cuervo.net/recursos/redpictutorials/sensores/sensores_de_distancias_con_ultrasonidos.pdf
[27]	Luisllamas[en línea][Consulta: 25 octubre 2017]
	Disponible en: https://www.luisllamas.es/medir-campos-magneticos-arduino-hall-49e/
[28]	Aprendiendoarduino[en línea][Consulta: 23 abril 2018]
	Disponible en: https://aprendiendoarduino.wordpress.com/2016/09/25/que-es-arduino/
[29]	Diymakers[en línea][Consulta: 10 marzo 2018]
	Disponible en: http://diymakers.es/arduino-processing-primeros-pasos/
[30]	Prometec[en línea][Consulta: 10 marzo 2018]
	Disponible en: https://www.prometec.net/bt-hc06/
[31]	Martyncurrey[en línea][Consulta: 10 marzo 2018]
	Disponible en: http://www.martyncurrey.com/arduino-and-hc-06-zs-040/
[32]	Polaridad[en línea][Consulta: 10 marzo 2018]
	Disponible en: https://polaridad.es/usb-serie-terminal-putty/
[33]	Prometec[en línea][Consulta: 10 marzo 2018]
	Disponible en: https://www.prometec.net/pc-bt/
[34]	Academia.edu[en línea][Consulta: 24 abril 2018]
	Disponible en: http://www.academia.edu/9182998/Marduino
[35]	VILLANUEVA, Sergio; Diseño de un sistema de captura y procesamiento de señales. UPV. [en línea][Consulta: 24 abril 2018]
	Disponible en: https://riunet.upv.es/bitstream/handle/10251/53674/TFG-Dise%C3%B1o%20de%20un%20sistema%20de%20captura%20y%20procesamiento%20de%20se%C3%B1ales_1435782166888636165294520398573 8.pdf?sequence=2

[36]	Diymakers[en línea][Consulta: 24 abril 2018]
	Disponible en: http://diymakers.es/arduino-processing-primeros-pasos/
[37]	Playground[en línea][Consulta: 24 abril 2018]
	Disponible en: https://playground.arduino.cc/Interfacing/Processing
[38]	Aprendiendoarduino[en línea][Consulta: 24 abril 2018]
	Disponible en: https://aprendiendoarduino.wordpress.com/2016/03/06/firmata/
[39]	IV Curso de Fijación Externa Circular. Murcia 26-27 de abril de 2018.
[40]	Innovación Educativa Universitat de Valéncia [en línea][Consulta: 24 abril 2018]
	Disponible en: https://www.youtube.com/watch?v=MCthqJGsngE
[41]	qvision[en línea][Consulta: 24 abril 2018]
	Disponible en: http://www.qvision.es/blogs/manuel-rodriguez/2015/03/30/interpretacion-de-los-graficos-de-caja-en-el-analisis-descriptivo-e-inferencial/
[42]	Anestesiar[en línea][Consulta: 24 abril 2018]
	Disponible en: https://anestesiar.org/2015/una-caja-con-bigotes-el-grafico-de-caja/
[43]	Prometec[en línea][Consulta: 30 abril 2018]
	Disponible en: https://www.prometec.net/consumos-arduino/
[44]	Prometec[en línea][Consulta: 30 abril 2018]
	Disponible en: https://www.prometec.net/el-modo-sleep-en-arduino/