Universidad Politécnica de Cartagena

industriales
etsii UPCT

DESARROLLO DE PLANTILLAS INSTRUMENTADAS DE BAJO COSTE PARA MEDICIÓN DE LA PISADA

Titulación: Ingeniería Industrial
Intensificación:
Alumno/a: Antonio Miguel García Benítez
Director/a/s: Andrés Valverde Conesa

Cartagena, 9 de Octubre de 2014

I. MEMORIA

0. Índice

0.1. Índice de imágenes.

Nota: las imágenes que no tengan información de la fuente son propias (hechas por el autor del proyecto).

1. Objeto y antecedentes.

Este trabajo ha sido desarrollado por el alumno de Ingeniería Industrial D. Antonio Miguel García Benítez, D.N.I. 47.090.522 – T con el objetivo de obtener una calificación positiva en el trabajo y poder así obtener el título de Ingeniero Industrial, según las normas impuestas por la Universidad Politécnica de Cartagena.

Junto a este objetivo se desarrolla un objetivo paralelo, que es el de demostrar los conocimientos y aptitudes adquiridos durante los años de estudio. Para ello, se eligió este trabajo ya que es una mezcla de las diferentes materias y conocimientos que han sido cursados y obtenidos durante la carrera; no engloba a todas las materias porque sería imposible, pero sí es muy concreto con una de las ramas de la ingeniería actual, la electrónica.

Así mismo, este trabajo busca ofrecer una solución a un problema de carácter biomecánico mediante una aplicación de ingeniería. El problema en cuestión es analizar la pisada de una persona mediante un sistema electrónico que permita traspasar estos datos a un ordenador con el objetivo de su análisis posterior.

Con el tiempo, está siendo creciente la inclusión de la ingeniería en la vida. La ingeniería que antiguamente se usaba tan sólo en el ámbito industrial, poco a poco ha ido formando cada vez más parte de nuestras vidas, siendo ahora algo muy cotidiano. Usamos elementos de ingeniería desde el momento en que nos despertamos con nuestro radio reloj, usamos nuestra ducha o calentamos leche en un microondas. Poco a poco, la ingeniería también se va asentando en el campo de las ciencias de la salud, con el objetivo de mejorar éstas.

Por ejemplo, se están asentando servicios de diagnóstico automático para diversas enfermedades, se está comenzando a usar robots en quirófanos que no se ven sometidos a las limitaciones humanas (fatiga, cansancio,…) y otras múltiples aplicaciones.

Concretando en este proyecto, lo que se pretende es analizar la pisada del humano. Tiene dos vertientes este tipo de análisis:

- Por un lado, es muy usado en alta competición, ya que es muy necesario saber el corredor el tipo de pisada que tiene para poder desarrollar un calzado lo más acorde a su forma de pisar y mejorar su rendimiento tanto como sea posible, así como llegar a evitar lesiones.
- Por otro lado, para cierto tipo de enfermedades. Hay enfermedades que dejan expuestos los pies a múltiples riesgos, siendo el más grave la amputación. El análisis de esta pisada sirve para prevenir esta amputación, diagnosticando qué puntos son los más sensibles en el pie de un determinado enfermo y poder poner solución antes de llegar a tal extremo.

No es una novedad el deseo de analizar esta pisada, y por tanto, no es un proyecto pionero. Habrá aspectos que sí serán pioneros, en los que el proyecto marcará la diferencia con respecto al resto, si no, no tendría sentido llevarlo a cabo. Uno de estos aspectos principales es el coste del mismo, pues existen sistemas muy caros capaces de hacer lo mismo

que aquí se pretende; pero es obvio que para un centro de ortopedia pequeño, un laboratorio casero o similares casos, son sistemas que se quedan fuera del alcance. Por tanto, una de las bases de este trabajo, será el coste económico del proyecto, y el esfuerzo por reducirlo al máximo.

La memoria del presente trabajo tendrá la siguiente estructura; primero de todo, unos aspectos introductorios al mismo (Objetos, antecedentes, definiciones y abreviaturas), que servirán para introducir al lector en ciertos aspectos esenciales que debe conocer. Después, se pasa a hablar de la problemática en sí misma y de otras soluciones que existen con similar o idéntico objetivo. En un tercer bloque, se entraría en la discusión de los motivos que han llevado a elegir una solución u otra y el desarrollo de los aspectos necesarios de la misma, durante los apartados 4, 5 y 6. Se continuará con una descripción general del producto final; antes de llegar a los últimos apartados, que serán la muestra de los resultados obtenidos y las conclusiones, así como una discusión donde se comentarán las posibilidades de mejora y/o las futuras líneas de investigación para conseguir un producto lo más versátil posible.

2. Definiciones y abreviaturas.

Dentro de este apartado, se comentarán ciertos conceptos básicos que se consideran de importante conocimiento para el lector del presente trabajo, así como de abreviaturas necesarias para aclarar posibles dudas.

2.1. Definiciones.

- **Adquisición de datos:** procedimiento por el que comunicamos un proceso con diversas variables, tanto analógicas como digitales, con un sistema de procesamiento que se encarga de analizar y trabajar con esta información.
- **Amperio:** es la unidad para la medida de intensidad por el Sistema Internacional.
- **Baudios:** El baudio es una unidad de medida, usada en telecomunicaciones, que representa el número de símbolos por segundo en un medio de transmisión digital. Cada símbolo puede codificar 1 o más bits dependiendo del esquema de modulación.
- **Ciclo de marcha (*gait cycle* en inglés):** es la unidad básica de medida en el análisis de la marcha. Empieza cuando un pie entra en contacto con el suelo y acaba cuando ese mismo pie vuelve a entrar en contacto con el suelo. Esos instantes son tomados como contacto inicial.
- **Circuito integrado:** es un circuito encapsulado de material semiconductor que contiene circuitos electrónicos de pequeño tamaño en su interior.
- **DIP**: dual in line package, simboliza un tipo de encapsulado de circuitos monolíticos que se basa en que tiene dos líneas de patillaje, en línea recta y paralelas, quedando siempre el mismo número de patillas a un lado que al otro.
- **Estrategia de control:** conjunto de pasos a seguir por un sistema procesador para lograr el control de una determinada variable o proceso.
- **Frecuencia:** es el número de operaciones por segundo que realiza un sistema electrónico.
- **Microcontrolador:** es un circuito integrado programable, capaz de ejecutar órdenes grabadas en una memoria ROM.
- **Resistor o resistencia:** Se denomina resistor al componente electrónico diseñado para introducir una resistencia eléctrica determinada entre dos puntos de un circuito eléctrico[1]. Su valor se mide en Ohmios.
- **Sensor:** elemento capaz de transformar una magnitud física y transformarla en una magnitud eléctrica, que pueda ser recogida por un sistema de adquisición de datos.
- **Voltímetro**: es un instrumento de medida usado para medir la tensión entre dos puntos.
- **Voltio**: es una unidad de tensión que expresa la diferencia de potencial entre dos puntos.

2.2. Abreviaturas.

- Ω = Ohmio, unidad fundamental de medida de resistencia.
- A = Amperio, unidad fundamental de medida de intensidad.

[1] http://es.wikipedia.org/wiki/Resistor

- PC = Personal Computer.
- PIC = microprocesador de la empresa *Microchip Technology*. Es el acrónimo de "*Peripheral Interface Controller".*
- V = Voltio, unidad fundamental de medida de tensión.
- Hz = Hercio, unidad inversa de segundo utilizada para hablar de frecuencias.

2.3. Software y hardware usados.

En este apartado se dará una breve descripción de los programas y elementos hardware que han intervenido en el desarrollo del trabajo, tanto a nivel de creación de la memoria como de creación del programa.

- **AutoCad**: programa de dibujo técnico de gran fama. Ha sido usado sobre todo en el diseño y distribución de los sensores en la plantilla y diseño del circuito de hardware.
- **SmartDraw**: programa gratuito capaz de formar, de una manera rápida y sencilla, diagramas de flujo. Fue usado para crear los diagramas de flujo de las estrategias de control.
- **Microsoft Word 2007:** procesador de textos. Usado para la redacción de la memoria.
- **PIC C Compiler:** éste es un software propietario de la empresa CSS para la escritura de programas para microcontroladores en lenguaje C, de alto nivel.
- **PICkit 2:** este software es el que controla el programador de MicroChip para poder introducir el programa diseñado en el PC en la memoria del controlador.
- **Proteus:** es un software de simulación electrónico, para poder crear circuitos virtuales y comprobar su funcionamiento como si se tratasen de circuitos reales; pudiendo realizar mediciones para comprobar su correcto funcionamiento.
- **NetBeans IDE 8:** es un entorno de desarrollo de programas, fundamentalmente en Java, pero que admite otro gran número de lenguajes. Es válido tanto como para programas de salida y entrada en modo texto como programas con GUI (interfaz de usuario gráfica).

3. Estado de la materia y soluciones existentes.

3.1. Discusión sobre el estudio de la pisada.

El estudio de la pisada se puede afrontar desde dos puntos de vista fundamentales. Por un lado, de forma estática o a bajas frecuencias (andando), que nos aporta información acerca de los mecanismos de mantenimiento del equilibrio, y la biomecánica del miembro inferior durante la marcha. Podólogos y fisioterapeutas suelen encargarse de medir esta pisada estática en sus pacientes, no es nuevo que el pisar mal provoca dolores en la espalda, en la cadera; un desgaste pronunciado en ciertas zonas del calzado, etc.; en algún tipo de pacientes una mala forma de pisar puede derivar en problemas mucho más graves, concretamente, se hablará del caso de pacientes con diabetes, enfermedad muy común en la actualidad. Los pacientes con diabetes suelen desarrollar daños en su sistema nervioso, siendo esta patología denominada neuropatía. Es una enfermedad del sistema nervioso que suele degenerar en pérdidas de sensibilidad, siendo una de las más usuales la pérdida de sensibilidad en piernas y pies. Esto provoca alteraciones en la forma de andar del paciente, provocando zonas que soportarán mucho más esfuerzo que otras, que pueden derivar en úlceras plantares en un primer momento; e incluso a la amputación como última opción.

Por otro lado, el estudio de la pisada de forma dinámica, esto es, andando o en carrera, es muy importante desde el punto de vista de la prevención de lesiones y de la mejora de rendimiento. Históricamente es posible remontarse muy atrás para hallar evidencias del intento de este análisis, no siendo hasta el siglo XX cuando se han desarrollado las técnicas más avanzadas, evidentemente, pero la intención de averiguar por qué ya existía. Tanto las técnicas modernas como el interés de la gente en correr tanto a nivel profesional como a nivel de aficionado han desarrollado muchísimo la investigación en este ámbito.

De acuerdo con Hanks G. A. (1982)[2], entre la mitad y un cuarto de corredores (profesionales y aficionados) sufren a lo largo del año alguna lesión que les obliga a cambiar sus hábitos de correr o afectan a su rendimiento. Por lo tanto, no es algo que se pueda omitir, sino que es un aspecto muy importante en la industria.

La pisada se compone de cuatro fases que será necesario conocer bien para posibilitar el análisis satisfactorio de la misma. Es necesario que a la hora de desarrollar el proyecto estas fases se conozcan para así poder desarrollar un elemento fiable y con fundamentos teóricos.

Estas fases son:

1. Fase de impacto.
2. Fase de soporte y postura media.
3. Fase de despegue y propulsión.
4. Fase de vuelo y recuperación.

En la pisada óptima, durante cada una de las fases se tiene una concreta distribución de presiones; así por ejemplo, en la fase de impacto la distribución esperada sería muy elevada

[2] Hanks and Kalenak, “Running Injuries.”

en la parte posterior del pie y nula en la parte anterior (se pisa primero con el talón). No obstante, esto no es siempre así, pues según la velocidad a la que se corra, la distribución óptima será una u otra. Por tanto, es necesario distinguir en qué tipo de carrera se está observando.

3.1.1. Tipos de carrera.

La clave para distinguir entre los tipos de carrera está en el ciclo de marcha, que ya se introdujo en el apartado de definiciones y abreviaturas (punto 2.1). Según la forma y la tendencia de este ciclo, se puede distinguir entre tres tipos de carrera.

Andar.

Es el tipo de carrera más lento. Al andar, en un ciclo de marcha existen periodos de doble apoyo (ambos pies en el suelo), que es el rasgo más característico. Esto se produce porque la fase de apoyo toma más del 50 % del ciclo de marcha, por lo que se superponen ambos apoyos en algún instante. La evolución normal de la pisada al andar es, como se aprecia en la imagen 1:

- En el contacto inicial (IC) se apoya primeramente el talón del pie.
- Se pasa a la fase mid – stance (postura media), donde la distribución de presiones se iguala a lo largo de la planta de pie.
- Se pasa la presión hacia la parte posterior de pie (Heel Fit, HF) en la tercera fase.
- Por último, la fase de despegue, que en la imagen se corresponde con "*Toe Off (TO)*".

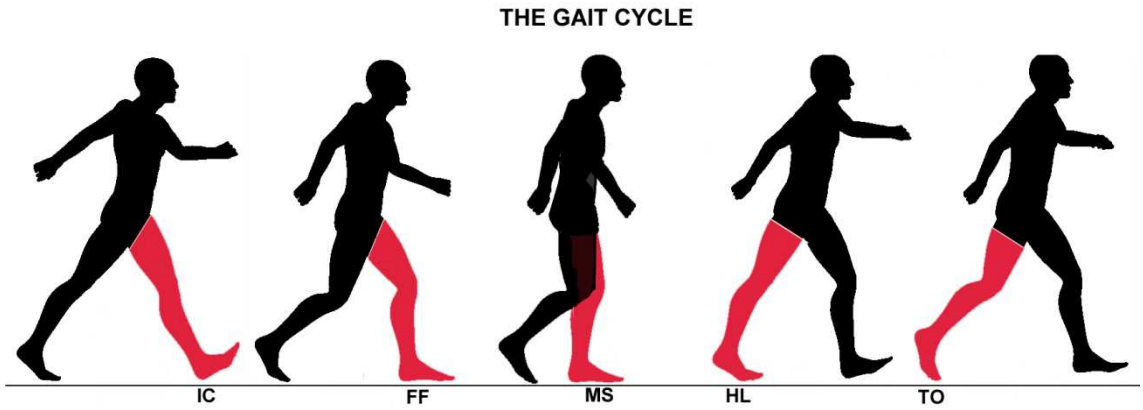


Figura 1: posición de la pierna durante un ciclo de marcha al andar. Fuente:

Correr.

Es el tipo de velocidad intermedia. El límite entre andar y correr se da cuando periodos de apoyo doble (los dos pies están el suelo en algún instante) dan paso a periodos "flotantes" (ningún pie está en contacto con el suelo) al principio y al final del ciclo de marcha. Esto es debido a que el apoyo ya no consume el 50% del tiempo del ciclo de marcha, por lo que en ningún momento se apoyan los dos pies. En la imagen 2 se puede apreciar cuál es la sucesión más usual:

- En el contacto inicial (IC) se debe apoyar la parte central del pie de forma que soporte unas presiones similares en toda la planta, así como en la segunda fase "mid- stance" (postura media).
- En el tercer instante, el despegue (Take off) lo que sucede es que la presión es soportada en la punta del pie, y comienza el "balanceo" (swing) que se corresponde con la fase de vuelo y recuperación (en la imagen, esta fase está desgranada en tres posturas diferentes):

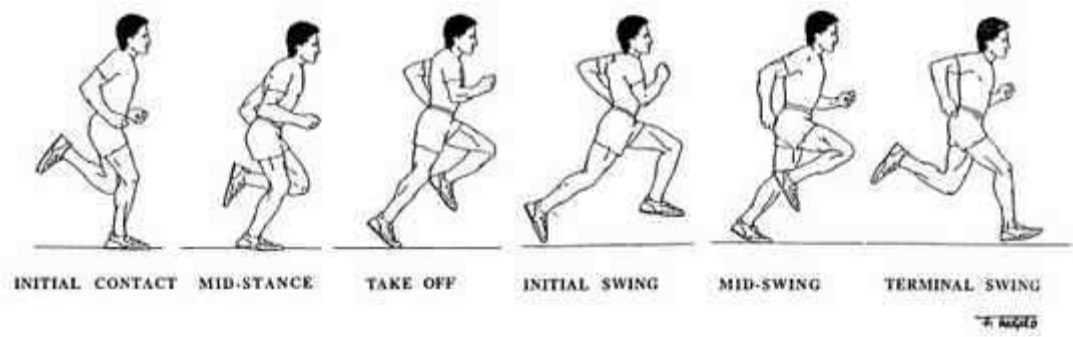


Figura 2: posición de la pierna durante un ciclo de marcha al correr. Fuente:

Sprint.

Es el tipo de máxima velocidad. Cuando se pasa de correr a sprint, el contacto inicial cambia de la parte media del pie que se comentó en el apartado anterior, a la parte anterior del mismo, haciendo que el pie permanezca menos tiempo en contacto con el suelo. Esto es debido a los diferentes objetivos entre sprintar y correr. El sprint se usa para correr una distancia en el menor tiempo posible; mientras que correr es un ejercicio con el objetivo de recorrer largas distancias, sin ser prioritario el tiempo.

Para el sprint, el cuerpo y sus partes se mueven tan rápido como pueden a través de la carrera; mientras que al correr, el cuerpo se mueve en función de la energía que pueda demandar la carrera.

3.2. Soluciones existentes.

En una primera instancia, lo que se hizo fue comprobar las posibles soluciones que existían para analizar la pisada de la misma forma que se quería conseguir con el desarrollo del presente trabajo. Esto ofreció una serie de soluciones que fueron muy útiles de investigar, tanto para saber las cosas que se debían evitar como para adquirir ideas que luego pudieran ser convenientes.

Se encontraron muy diferentes opciones, solo serán destacadas algunas de ellas, bien por sus similitudes con lo que se pretende conseguir o bien por su uso comercial y extendido.

3.2.1. Plantillas para mediciones a baja frecuencia (caminar).

Desde la Universidad de los Andes, en Colombia, se desarrolló una plantilla con el objetivo de medir ambulatoriamente la pisada[3]. En el proyecto trabajaron los departamentos de Ingeniería Biomédica y el Centro de Microelectrónica de la citada universidad. La siguiente cita textual corresponde al resumen del proyecto:

[3] Torres et al., "DESARROLLO DE UN SISTEMA ELECTRÓNICO PARA MEDIR AMBULATORIAMENTE PRESIONES EN LA PLANTA DEL PIE INSENSIBLE."

"Con el fin de medir ambulatoriamente las presiones en 8 puntos estratégicos de la planta del pie humano, en pacientes que sufren de pie insensible, se diseñó un sistema de telemetría el cual consta de 2 plantillas instrumentadas encargadas de medir continuamente los valores de presión presentados en cada una de las plantas de los pies y transmitirlos mediante un enlace de comunicación inalámbrico a un sistema portátil encargado de recibir y almacenar las mediciones, para posteriormente descargarlas a una computadora en cual se pueden hacer los distintos diagnósticos respectivos por parte de los médicos especialistas.

Las principales restricciones en el desarrollo del proyecto fueron; el costo del sistema total junto con el tamaño y el consumo de potencia de las plantillas instrumentadas. Las pruebas realizadas mostraron una operación confiable a una velocidad de transmisión de 5 Khz en las bandas de los 315 y 433 Mhz durante 8 horas de operación continua."

Como ya se ha introducido al principio del apartado, de los dos objetivos primordiales que existen para enfocar el tema, esta solución sería englobada desde el primer punto de vista tratado; es decir, sería una solución con propósito médico y a baja frecuencia de pisada (andando). Las características esenciales y diferenciadoras con el resto de soluciones serían:

- Es un sistema portable, es decir, el objetivo no es medir la pisada en tiempo real, si no que almacena los datos para después ser leídos desde un ordenador.
- Deberá ser alimentado por baterías para aguantar las 8 horas de operación mientras el paciente realiza actividades usuales.
- Cada plantilla cuenta con 8 sensores dispuestos en las zonas que aguantarán más presión.
- Sensores Flexiforce 201, de tipo piezorresistivo.
- Cuenta con un convertidor analógico – digital de 8 bits de resolución.
- Presenta dos plantillas y un módulo común para el almacenaje de los datos, como se puede ver en la figura 3.

Los resultados del citado proyecto, en palabras de sus creadores, son que presentan un sistema capaz de medir las presiones encontradas en las plantas de los pies durante una actividad normal de 8 horas trabajando de forma continua. Las pruebas actuales les han llevado a afirmar que es un sistema fiable, ergonómico y de resultados reproducibles, de bajo coste en comparación con otros métodos comerciales. Por último, se están realizando pruebas para la validación médica del dispositivo, con pacientes reales, por parte del grupo de especialistas del Laboratorio de Marcha del Instituto de Ortopedia Infantil Roosevelt, en Colombia.

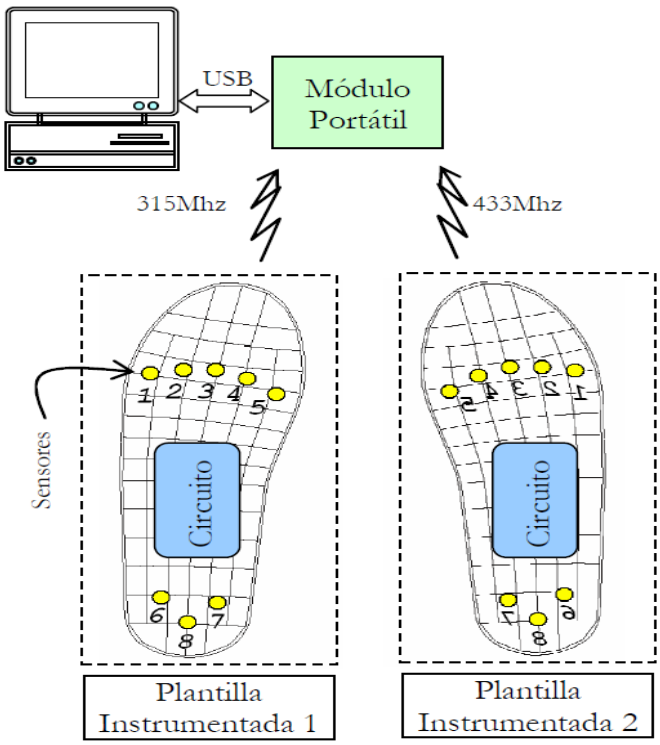


Figura 3: esquema del proyecto. Fuente: extraído del trabajo anteriormente citado.

Conclusiones para este proyecto.

Si se compara con el sistema que se quiere obtener, a priori es bastante similar, pero tiene algunas pegas que hacen que se desmarque del objetivo del proyecto. Por un lado, está diseñado para funcionar durante mucho tiempo (8 horas) y haciendo una actividad normal, no se especifica si es válido el proyecto para deportistas, cosa que sí se quiere llevar a cabo en este proyecto. Además, fija un tiempo de uso muy grande frente a lo que se puede tardar en hacer el ensayo sobre un deportista. Otra pega es el número de sensores, que a priori puede antojarse escaso y hay partes del pie que no están siendo recogidas por el sistema de medición. Tampoco se dan detalles de cómo es el software del PC, ni el tipo de visualización.

3.2.2. Plantillas para mediciones versátiles (alta y baja frecuencia).

El segundo proyecto que se encontró en la recopilación de información para crear el proyecto propio fue otro artículo, escrito en inglés, de la Universidad de Ciencias Aplicadas de Oulu, en Finlandia.[4] La traducción más ajustada del título sería: Desarrollo de una plantilla sensible a presiones basada en sensores de película electromecánica para el análisis de la marcha.

Se puede traducir el resumen del artículo de forma similar a lo que se hizo con el anterior artículo para afinar más en lo que el artículo pretende (extracto literal):

La distribución de presiones en la suela incluye información importante de la marcha mientras se anda o se corre. Por ejemplo, monitorizando los

[4] Hannula, Säkkinen, and Kylmänen, "Development of EMFI-Sensor Based Pressure Sensitive Insole for Gait Analysis."

detalles de la distribución de presiones en la suela caminando normalmente puede ayudar a detectar muchas posiciones deficientes en el pie. En este estudio, un simple y versátil sistema de mapeado basado en sensores EMFI (sensores de película electromecánica) fue diseñado para el análisis de la marcha.

El sistema consiste en una plantilla delgada y flexible, incluyendo 16 elementos EMFI, conectada mediante amplificadores analógicos personalizados y una tarjeta de adquisición de datos a un ordenador.

El software del ordenador del sistema de medida muestra la presión en cada sensor en la suela en tiempo real, y graba los datos en un archivo.

La precisión y la fiabilidad del sistema fue preliminarmente evaluada mediante medidas experimentales. Los resultados mostraron que el sistema tenía capacidad de medir la distribución de presiones en la suela del pie de forma fiable. El coeficiente de correlación entre la presión aplicada y la salida del sensor era de 0.99. La siguiente fase de este proyecto de investigación es evaluar la aplicación del sistema para la evaluación de la marcha en un gran número de sujetos.

Al contrario que el anterior, este sistema no especifica si es para pisada estática, andar (baja velocidad) o correr (alta velocidad). No obstante, en la descripción del artículo sí que se habla de esa posibilidad, por lo que a priori, se observa un sistema más versátil que el anterior. Algunas de sus características básicas son:

- Este sistema mide la pisada en tiempo real y, además, almacena en un archivo los resultados.
- La plantilla dispone de 16 sensores EMFI.
- Requiere de una tarjeta de adquisición de datos para ser conectada con el PC.
- En el ordenador, la salida es una gráfica tal y como se aprecia en la figura. En la figura estaría representado un solo ciclo de marcha. La primera parte de la gráfica sería correspondiente al inicio de la pisada, donde los sensores situados en la parte posterior de pie reciben toda la presión, mientras que se aprecia un descenso paulatino de éstas y un ascenso en los otros sensores, situados en la parte anterior. Otro aspecto a destacar es que este usuario, emplea más tiempo pisando con la parte delantera que con la trasera del pie, pues la segunda mitad de la gráfica es más ancha que la primera.

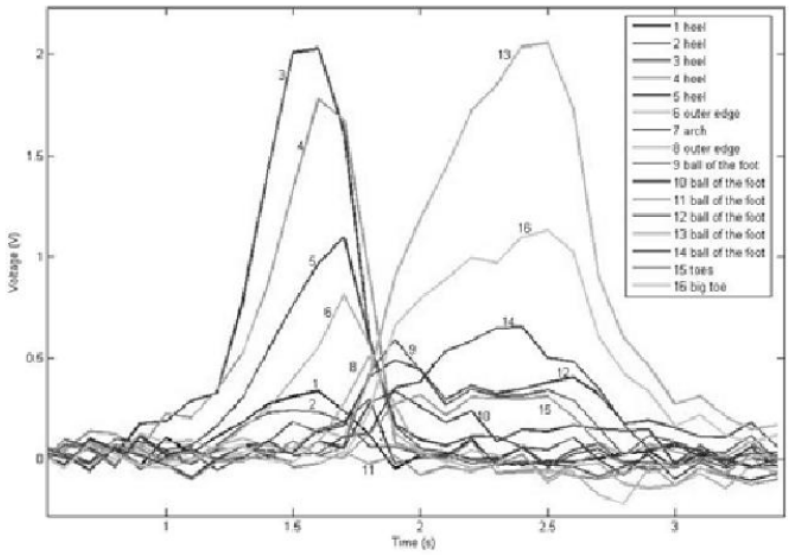


Figura 4: variación de presiones en un ciclo de marcha. Fuente: extraída del artículo anteriormente citado

En cuanto a los resultados del proyecto, en palabras de sus creadores, la mayor ventaja de éste es su simplicidad y versatilidad. Usando esta misma estructura, es posible construir plantillas dedicadas para propósitos más concretos, por ejemplo, medir tan solo las presiones en el talón o en la punta de los dedos. El software fue desarrollado con LabView, que también es fácilmente programable y adaptable.

Conclusiones para este proyecto.

A diferencia del anterior proyecto, este sí ofrece un sistema de visualización de datos, y unos resultados más concretos, como la gráfica mostrada. También, cambia su filosofía, mientras que el otro buscaba un estudio a largo plazo de la pisada en actividades cotidianas, éste es algo más directo, mostrando la salida en tiempo real, y siendo válido también para carrera. Quizá una gran pega es el medio de adquisición de datos, que es una tarjeta que puede ampliar el precio del equipo a utilizar. Aunque no es apartado para esto, cabe hacer un breve recordatorio que uno de los objetivos del proyecto es un sistema de bajo coste.

3.2.3. Plantillas comerciales.

Por último, y abarcando ya casi la mayoría de posibilidades existentes hoy en día en esta área, se encontraron las plantillas Biofoot/IBV, desarrolladas por el Instituto de Biomecánica de Valencia, centro concertado entre el Instituto Valenciano de Competitividad Empresarial (IVACE) y la Universitat Politècnica de València (UPV). La descripción del producto, extraída de su página web, es la siguiente:

> *Biofoot/IBV es un avanzado sistema de plantillas instrumentadas diseñadas para medir y analizar las presiones en la planta del pie en las condiciones en que éste se desenvuelve habitualmente, es decir, con calzado y en movimiento.*

Biofoot/IBV supone un significativo avance, frente a los tradicionales sistemas ópticos de exploración del apoyo plantar, utilizado en las siguientes áreas de aplicación:

- *Diseño y evaluación de calzado y sus complementos.*
- *Biomecánica: análisis de la marcha, caracterización de la marcha.*
- *Deporte: estudio, selección y adaptación del calzado deportivo. Análisis de los gestos deportivos (salto, carrera, marcha, ciclismo).*

Con el fin de mantener un historial organizado de los sujetos, Biofoot IBV gestiona los datos de cada sujeto implementando una estructura básica conformada por estudios, sesiones y medidas.

Biofoot/IBV permite realizar sesiones de medida de ambos pies simultáneamente, facilitando el análisis y la comparación de las presiones ejercidas por cada pie durante la marcha.

Éste ya es un sistema comercial, desarrollado con un propósito de venta al público, al contrario que los anteriores proyectos. En la siguiente imagen, se puede ver el pack que se suministraría al comprar este equipo:

Figura 5: presentación de las plantillas Biofoot/IBV. Fuente: página oficial del producto.

Dentro del maletín se incluyen los dos amplificadores de cada una de las plantillas, así como otro dispositivo adicional para ponerlo en la cintura que recibe la señal de éstos y la

transmite en tiempo real al computador, mediante una tarjeta de recepción. También otros elementos, como baterías de repuesto, o cargador de baterías.

Las características de este sistema son:

- Dos plantillas, con 32 sensores cada una.
- Representación de datos en tiempo real.
- Conexión con PC mediante transmisión inalámbrica.
- Dos circuitos de amplificación, uno por plantilla.
- El software del computador es capaz de mostrar la visualización de diferentes y múltiples maneras muy visuales, como se puede apreciar en la imagen 6:

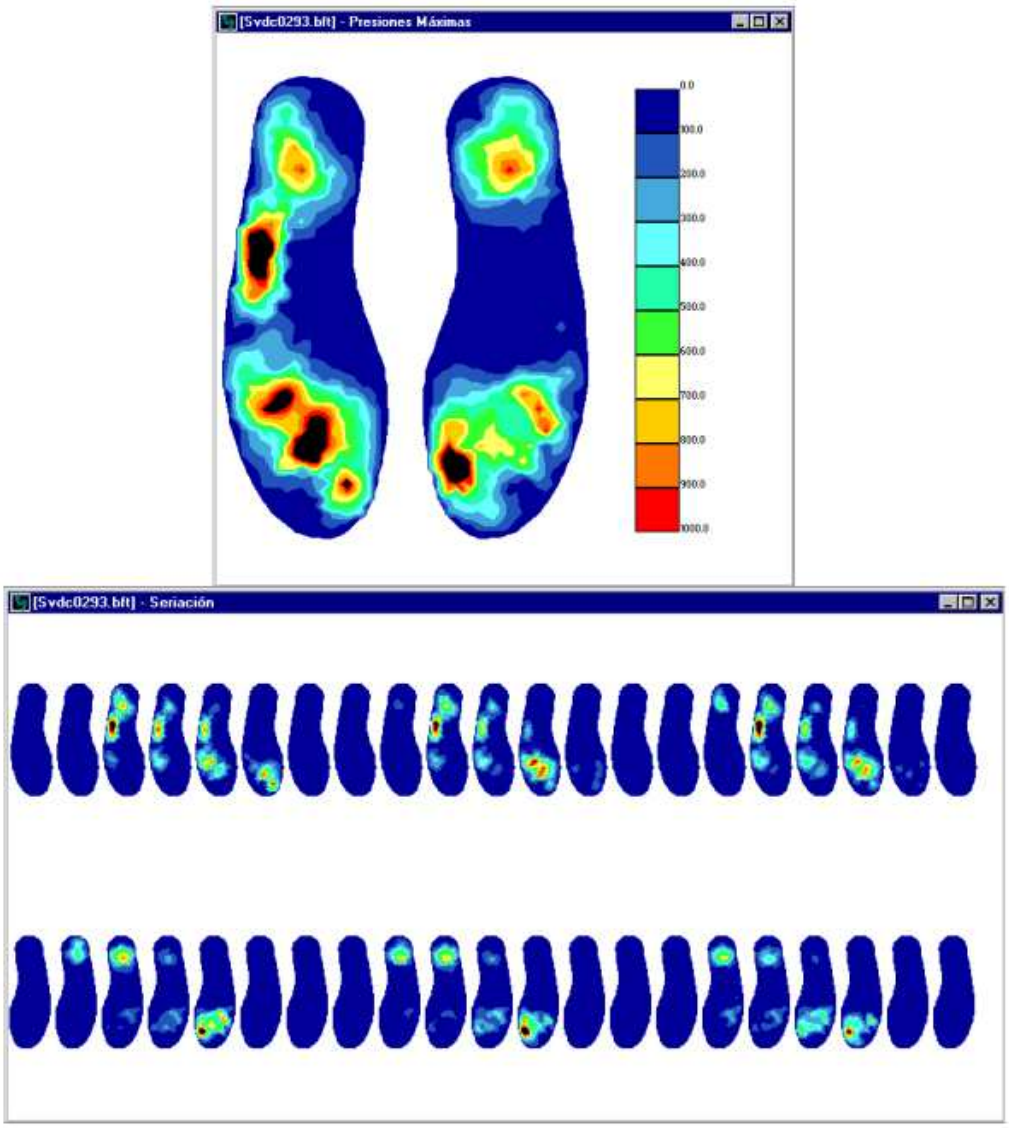


Figura 6: presentación de los resultados de un ciclo de marcha de las plantillas Biofoot/IBV. Fuente: página oficial del producto.

En el primer caso, es un mapa de presiones muy visual; mientras que en segundo, sería el desglose a través del tiempo del primer mapa para observar la evolución de la pisada del usuario.

A diferencia de los otros proyectos, este proyecto sí ha sido probado por personal especializado, puesto que es un producto que en la actualidad está a la venta; y no es muy difícil en una búsqueda sencilla encontrar ejemplos de clínicas y ortopedias que usan este producto.

Conclusiones para este proyecto.

Este proyecto reúne algunas características muy atractivas. En general, es el proyecto más completo de los estudiados, ofrece un producto final y existen multitud de pruebas de su funcionamiento. El sistema de visualización es muy eficaz, con múltiples opciones de visionado, para todo tipo de público, permite almacenar los datos de los pacientes… En general, un sistema de visualización con muchas posibilidades, y por supuesto que sería más que deseable lograr uno similar.

Claramente, el punto negativo, es el coste económico; es más elevado que el de cualquiera de los otros proyectos que se han visto, y también que el que se está desarrollando en este trabajo. Es normal puesto que se está hablando de un producto comercial, que no ha sido desarrollado por un único individuo y con un objetivo económico y no académico.

4. Requisitos de diseño.

En virtud de lo que se ha visto en apartados anteriores, se pueden establecer unos requisitos que serán necesarios para crear un sistema capaz de dar respuesta a las incógnitas planteadas. En el anterior punto se han desarrollado varias ideas que podrían ser adquiridas para este proyecto en concreto. Unas serán descartadas por el tiempo o recursos humanos que necesiten; otras por su coste; y otras incorporadas para mejorar el diseño.

La filosofía de este proyecto se puede resolver en una cadena de causalidad como la que aparece en la imagen:

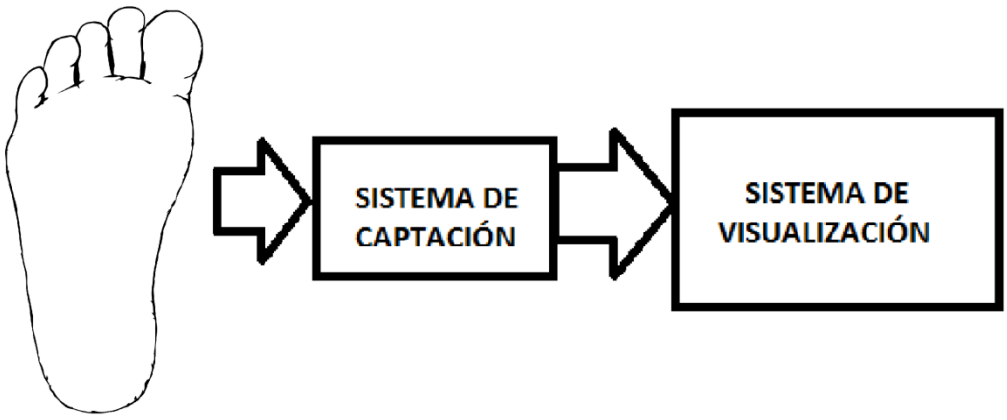


Figura 7: diagrama de bloques del supuesto sistema.

Lo que hay que hacer es decidir cuál será cada uno de estos sistemas, y decidirlo en base a unas decisiones que serán explicadas en este apartado.

Primero de todo, se necesita un sistema capaz de captar datos de un pie humano. O lo que es lo mismo, un sensor que nos transmita los valores de presión que produce una pisada. Es un apartado donde hay libertad, puesto que al ser de las primeras decisiones a tomar, no está condicionada por ningún otro parámetro. Se sabe de antemano que tendrá que ser un sistema analógico, como la gran mayoría de los sensores, puesto que en la naturaleza las magnitudes son analógicas. Actualmente, existen distintos tipos de sensores que son capaces de medir este valor; serán detallados en el siguiente apartado, así como un estudio de cuál se ha considerado más acorde con nuestro proyecto y cuáles se han descartado por sus características.

Es necesario un sistema de interpretación de los datos obtenidos por los sensores. Estos datos, como ya se ha comentado, serán analógicos. Luego, en función del sistema de visualización, este sistema será condicionado. Si es un sistema digital, será necesario, además de la captación y adecuación del dato, el paso a digital; si no lo es (por ejemplo, un osciloscopio), este paso será omisible. Por tanto, este sistema de interpretación, dependerá en gran medida tanto del paso previo, los sensores; como del paso siguiente, el sistema de visualización. Esto fijará las condiciones como si es necesaria la transformación de datos, la amplificación o no de los valores obtenidos por los sensores, etc. Pero lo que se puede deducir es que este sistema será el núcleo del proyecto, que se encargará de hacer de intermediario

entre lo que el usuario verá y el pie. Por tanto, habrá que poner especial énfasis en la elección de este elemento (o elementos).

Por último, es necesario un sistema que sea capaz de mostrar los datos de una forma sencilla para el usuario. Hay que considerar que este sistema no tiene como objetivo un público especializado, como pueden ser otros sistemas que sí serán usados por otros técnicos, o ingenieros. Las características de este trabajo hacen que su destinatario sea bien diferente, del sector de la sanidad, como pueden ser médicos, enfermeros, fisioterapeutas, etc. Esto restringe las posibilidades, pues no es deseable un sistema de visualización que solo unos pocos conozcan, y sea necesario adquirir conocimientos nuevos para poder usarlo, si no que interesa un sistema lo más general posible, que sea fácil de interpretar por todos, independientemente del sector que trabaje.

Todo esto es desde el punto de vista conceptual. Considerando otros requisitos, también se deben de tener en cuenta aspectos como la velocidad de captación del sistema, la velocidad de transmisión, etc. Como ya se discutió en el apartado 3, el estudio de la pisada puede tener distintos objetivos y distintas formas de enfocarlo. Esto condiciona también el sistema final, pues habrá que estudiar si la velocidad de adquisición de datos es suficiente para un corredor de competición, o por el contrario, si la captación del dato por parte del sensor es suficientemente lenta para que el sistema la recoja.

Concretando aspectos, la velocidad de captación del sistema deberá ser suficiente para adquirir la variación de presión en el pie de forma confiable. Esto es, que no sólo deberá ser mayor que la frecuencia de pisado, si no que deberá ser mucho mayor para captar la variación durante el tiempo que el pie está en contacto con el suelo. Cabe recordar que según el ciclo de marcha, estábamos en un tipo de carrera o en otro, y que una forma de diferenciarlo es si el pie estaba más del 50 % del ciclo en el suelo o menos. Esto influirá en la frecuencia que deberá ser escogida para el sistema.

También será necesario establecer las comunicaciones apropiadas entre los sistemas, pues según uno u otro, será necesario establecer distintos tipos de transmisión de datos, y todos ellos deberán ser suficientes para garantizar la fiabilidad del sistema y la no pérdida de datos relevantes de por medio. Así, si se establece una frecuencia mínima para el corredor, también tendrá gran influencia en el sistema global.

Otros requisitos pueden ser meramente estéticos o desde el punto de vista de la comodidad, puesto que es un elemento que, previsiblemente, una persona deberá llevar en la planta del pie, lo que será necesario que sea lo menos “intrusivo” posible, es decir, que no provoque molestias al andar o correr.

También, otro aspecto a tener en cuenta es el tamaño del sistema final. Al margen de lo hablado en el anterior párrafo de la comodidad, pues se debe considerar que el sistema final, formado por cualesquiera que sean los elementos escogidos en el análisis de soluciones, no deberá de ser de tamaño excesivo, para no llegar a ser molesto para el usuario.

Una vez se tienen todos estos requisitos claros, queda el análisis de soluciones, donde se estudiarán las diferentes soluciones que por uno u otro método se han encontrado para

estos problemas, se discutirá ampliamente sobre la posibilidad de incluirlas o no en el sistema final, comparándolas entre sí y alcanzando la solución final que se decidió incluir para la consecución satisfactoria de este trabajo.

5. Análisis de soluciones.

Por lo visto en el apartado anterior, ya se tiene una idea básica del sistema que se pretende obtener en la finalización de este proyecto. No obstante, también se han hablado de múltiples posibilidades, y mucha libertad en muchas de las decisiones; lo que obliga a hacer un extenso estudio de las diferentes soluciones. También se verá más adelante, que en la elaboración del proyecto, se tomó una mala decisión y se trabajó con unos sensores que finalmente no fueron convenientes al propósito y hubo que dar un cambio de dirección en un momento dado.

5.1. Elección del sistema de captación.

Entre las diferentes opciones que se barajaron estaban:

- Galgas extensiométricas.
- Sensores piezoeléctricos.
- Sensores de resistencia variable.

Cada uno de ellos posee unas características diferentes y necesitarán requisitos diferentes si se incluyen finalmente en el sistema. Aunque todos ellos tienen como objetivo el medir fuerzas, se observará que entre ellos hay profundas diferencias, tanto a nivel conceptual como a nivel funcional; por lo que es muy relevante la decisión que se tome.

No sólo en esta parte del sistema, sino también en el resto de partes, su influencia es notoria. La mayoría de los sensores necesitan de circuitos de acondicionamiento, más o menos complejos. La misión de un circuito de acondicionamiento es una ligera “traducción” del dato recibido a una magnitud que sea fácilmente asimilable por el sistema de interpretación. Dependerá de muchos aspectos, como la amplitud de salida del sensor (puede ser de algunos voltios, de milivoltios…), el tipo de salida (puede ser en tensión, en corriente, resistividad variable…), la frecuencia de salida, la respuesta a la entrada…

Es decir, que según el tipo de sensor que sea escogido, será necesario uno u otro tipo de circuito de acondicionamiento. Este circuito de acondicionamiento será necesario configurarlo para que la salida sea adecuada para el sistema de interpretación, por tanto, el circuito de acondicionamiento se englobará dentro de la segunda parte del sistema.

5.1.1. Galgas extensiométricas.

Una galga extensiométrica o extensómetro es un sensor, que sirve para medir la deformación, presión, carga,… entre otras cosas, que está basado en el efecto piezorresistivo. El efecto piezorresistivo es la propiedad que tienen ciertos materiales de cambiar el valor nominal de su resistencia cuando se le someten a ciertos esfuerzos y se deforman en dirección de los ejes mecánicos. Un esfuerzo que deforma a la galga producirá una variación en su resistencia eléctrica, que puede ser fácilmente medida por un circuito de acondicionamiento. En la imagen 8 se puede apreciar una galga extensiométrica real.

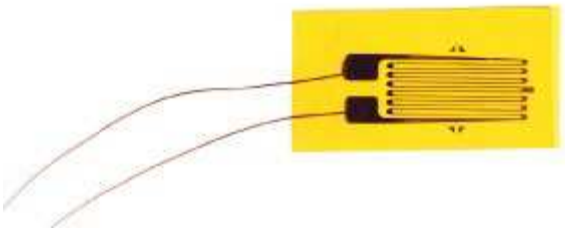


Figura 8: galga extensiométrica real. Fuente: apuntes Instrumentación Electrónica, EDII de Albacete.

En esencia, es una lámina metálica o semiconductora que reposa sobre un material aislante. El caso más general, es que la galga se adhiere al sistema donde se quiere realizar la medida y las deformaciones producidas en el sistema pueden ser cuantificadas. Poseen una zona de funcionamiento lineal, que se rige por la siguiente ecuación:

$$R=R_0(1+x)$$

Siendo R_0 la resistencia nominal de la galga y x el margen de variación. Así pues, la resistencia de salida R, será igual a la resistencia nominal en caso de no haber esfuerzo (en principio); y aumentará según aumente el mismo. Esto es relativamente cuestionable, pues uno de los problemas de las galgas es su sensibilidad a cambios de temperatura; no obstante, las desventajas se comentarán más adelante, después de hacer un análisis adicional del circuito de acondicionamiento.

El principio de funcionamiento básico ya está explicado, pero hay que añadir que la galga mide esfuerzos según lo que se ve en la siguiente imagen:

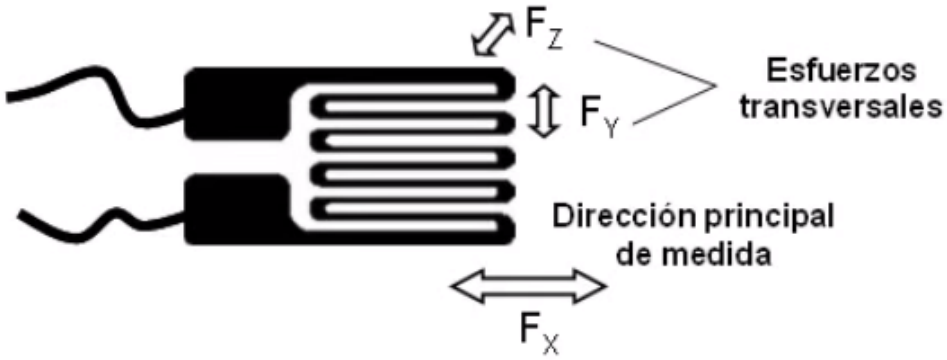


Figura 9: diagrama de una galga extensiométrica y explicación. Fuente: apuntes Instrumentación Electrónica, EDII de Albacete.

Que ya deja intuir uno de las desventajas para el propósito principal del proyecto.

Si se comenta sobre el circuito de acondicionamiento, se aprecia una característica básica de los sensores que cambian su resistencia: siempre será necesario alimentarlos con tensión, es decir, si no hay alimentación será imposible su funcionamiento. El funcionamiento básico es alimentarlo, y medir la caída de tensión producida en él, lo que se puede entender en el esquema mostrado en la siguiente figura:

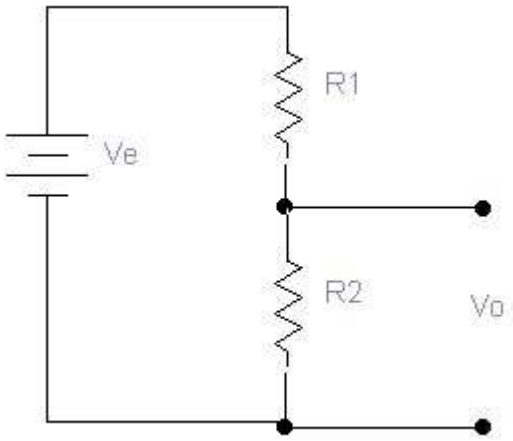


Figura 10: circuito de acondicionamiento para sensores resistivos.

Esta figura configuración es conocida en el ámbito de circuitos como un divisor de tensión. Según la relación entre las dos resistencias, la salida Vo será mayor o menor, nunca superior a la entrada Ve. La ecuación que rige este tipo de circuito es:

$$V_o = R_2 \cdot \frac{V_e}{R_2 + R_1}$$

Por lo que si uno de ellos tiene un valor de resistencia variable, la salida, Vo, dependerá totalmente de la variación. Este método es muy interesante a la par que simple, pues tan solo es necesario una resistencia y una fuente de tensión para acondicionar al circuito. La amplitud de salida variará en función de la resistencia estática que se sitúe para acompañar a la resistencia variable. Así por ejemplo, se puede configurar fácilmente un rango que sea entre 0 y 5 V o cualquiera de las necesidades que sean necesarias para el sistema de interpretación.

No obstante, las galgas extensiométricas no acaban de funcionar del todo correctamente con esta configuración, debido a que o los valores de deformación son muy grandes, o la variación de resistencia será muy pequeña, en torno al 1%. Esto induce fácilmente a error, puesto que una variación no se sabrá con firmeza si es debida a una variación de carga, o una variación de temperatura, un pequeño pico de tensión… Por tanto, habrá que buscar un método más complejo de acondicionamiento de este sensor.

En la propia documentación que proporcionan los fabricantes, ya indican un nombre repetidas veces que sugiere que es el método óptimo: el puente de Wheatstone. En la imagen 11 se puede apreciar un puente de este tipo.

Fundamentalmente, coge la idea del divisor resistivo y la amplia, poniendo dos divisores resistivos, uno de ellos con resistencias fijas y el otro, con una resistencia fija y la otra variable.

En la imagen, Rx se corresponde con la resistencia variable, siendo el resto de resistencias fijas. Se puede apreciar que, una vez más, el puente necesita alimentación externa, V, para trabajar.

Si se consideran todos los valores estáticos excepto el de Rx, la salida, Vm, será proporcional a las variaciones de resistencias.

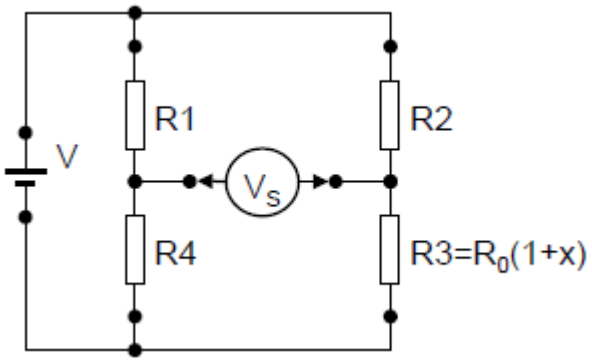


Figura 11: puente de Wheatstone. Fuente: apuntes Instrumentación Electrónica, EDII de Albacete.

El puente debe estar equilibrado. Esto es que para una variación de x = 0; la tensión de salida debe ser 0. Para que esté equilibrado, se debe cumplir la siguiente ecuación:

$$\frac{R1}{R4} = \frac{R2}{R0} = k$$

Siendo R0 el valor de la galga nominal. La tensión de salida se puede calcular mediante la siguiente ecuación:

$$V_s = V\left(\frac{R3}{R2+R3} - \frac{R4}{R1+R4}\right) = V\frac{kx}{(k+1)(k+1+x)}$$

Como se puede ver, totalmente dependiente de x y de las relaciones entre las resistencias estáticas. Además, con unos sencillos cálculos, se puede observar que la variación será lineal si k + 1 es muy superior a x. Este puente también ofrece otras ventajas, como poder trabajar alimentado en corriente. Comercialmente hablando, estos puentes pueden ser vendidos junto al sensor, con lo la señal de salida está ya adaptada para ser recogida por un sistema.

En un inicio parece un sensor que cumple las características, pero se han visto un par de detalles que hacen que sea descartado de manera casi inmediata. Por un lado, si se observa la figura 9, el funcionamiento de este sensor es transversal, es decir, que acoplado a una plantilla que funcionase bajo el pie de un humano, mediría esfuerzos aplicados sobre la plantilla en el plano horizontal, pero sería independiente del esfuerzo vertical, que es lo que interesa en este proyecto. Una opción idílica sería el que existiesen galgas de este tipo, pero orientadas en vertical, y que no fuesen diseñadas sobre una lámina. Esto permitiría medir la fuerza vertical, pero supondría unos sensores más grandes, y por consiguiente, supondremos un diseño que no cumple con el requisito de comodidad para el usuario.

Otra desventaja es que hasta ahora no se han considerado magnitudes, pero es conocido que las galgas son diseñadas para deformaciones, y aunque se producen, las deformaciones en el cuerpo humano no son suficientes para producir una apreciación

significativa de estos cambios, por lo que quizá fuese una tarea casi imposible el encontrar unas galgas que fuesen adecuadas para este proyecto.

5.1.2. Sensores piezoeléctricos.

Los sensores piezoeléctricos están basados en el efecto piezoeléctrico, propiedad de algunos materiales que consiste en la variación de su tamaño en función de una corriente eléctrica que atraviese el material, o al contrario, que la modificación del tamaño del material sea capaz de generar una corriente eléctrica; que es el caso más interesante para este proyecto. La idea aquí explicada se puede ver de una forma visual en la imagen 12, que como se aprecia, muestra ambos efectos, en la izquierda la tensión de salida ante una deformación producida por una fuerza F; en la derecha, un incremento de longitud provocado por el generador.

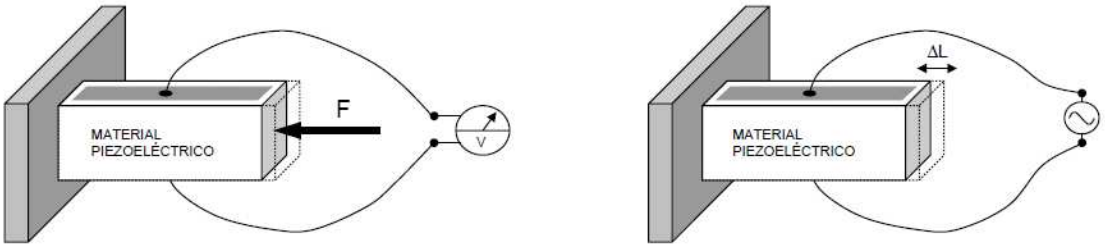


Figura 12: dibujo y fundamento de funcionamiento de un material piezoeléctrico. Fuente: apuntes Instrumentación Electrónica, EDII de Albacete.

Un sensor piezoeléctrico de los que puede tener cabida en este contexto se aprecia en la siguiente imagen:

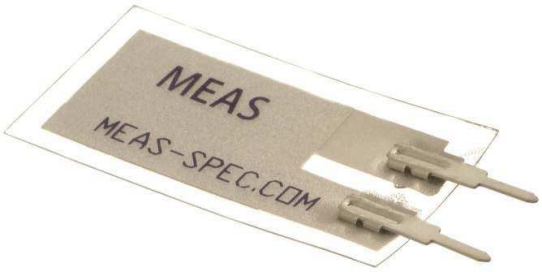


Figura 13: sensor piezoeléctrico real. Fuente: documentación del fabricante, MEAS-SPEC.

Es una lámina de plástico que aísla en su interior una pequeña porción de material piezoeléctrico y que comunica con el exterior mediante unas patillas dispuestas para ser conectadas con el circuito de acondicionamiento necesario. A primera vista, todo parece conveniente para el proyecto, puesto que elimina el problema anterior de la galga del tamaño. Esta lámina sí mide esfuerzos verticales, que es lo que se necesita a fin de cuentas. Antes de estudiar sus ventajas o inconvenientes, se estudiará su acondicionamiento.

Desde la propia documentación del sensor, el fabricante indica las formas más comunes de acondicionar la señal de salida del sensor. Se tratarán las más importantes:

- Amplificador de carga.
- Amplificador no inversor de carga.
- Amplificador de voltaje.
- Amplificador no inversor de voltaje.

Las configuraciones más importantes son amplificador de carga y amplificador no inversor de voltaje. Las otras son simples consecuencias de utilizar la entrada inversora o la no inversora de un amplificador operacional. Se hará un estudio por tanto de estas otras dos:

- Amplificador (inversor o no inversor) de carga.

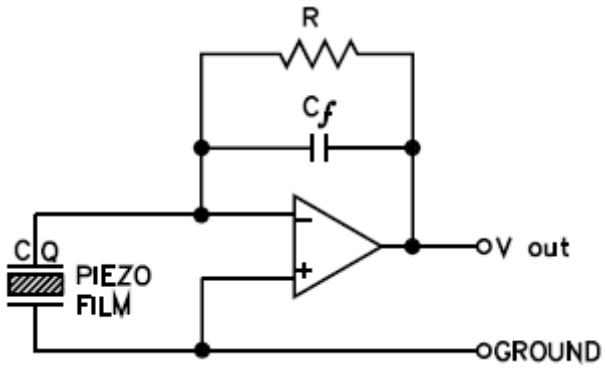


Figura 14: circuito de acondicionamiento de sensor piezoeléctrico. Fuente: hojas de características del sensor Piezo Sensor- DT series; de MEAS – SPEC.

Esta configuración es muy simple, pues toma la salida del sensor y la lleva a la entrada inversora del amplificador, mientras que la otra entrada se sitúa a tierra. Para ajustar la ganancia del amplificador se debe de variar el valor del condensador C_f que servirá para adecuar el nivel de la salida a lo que el siguiente elemento del sistema sea capaz de recoger. Este amplificador tiene la ventaja de que, al trabajar con la carga, no se ve afectada su salida por la posible caída de tensión producida en el cable. No obstante, según el sistema que se encuentre en el paso posterior, se tendrá que elegir una u otra configuración. La diferencia entre la configuración inversora y no inversora es que a la salida, el voltaje conserva el signo previo a la amplificación o lo invierte, respectivamente.

- Amplificador (inversor o no inversor) de voltaje.

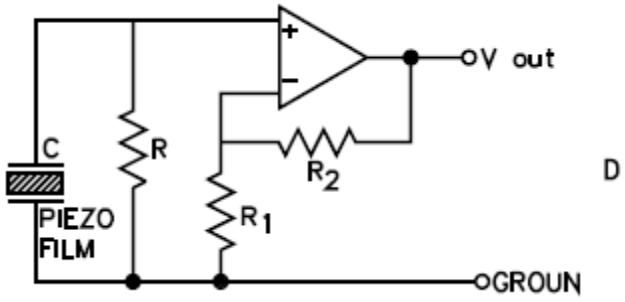


Figura 15: circuito de acondicionamiento de sensor piezoeléctrico. Fuente: hojas de características del sensor Piezo Sensor- DT series; de MEAS – SPEC.

En esta configuración se añaden algunos elementos más, pero son todos resistivos. En este caso, la ganancia se configura mediante la relación de las resistencias mediante la ecuación:

$$Ganancia = 1 + \frac{R_2}{R_1}$$

Aunque más compleja a nivel de montaje (más elementos), con toda probabilidad esta configuración será más conveniente, puesto que amplifica el voltaje y no la carga; que probablemente, el hipotético sistema posterior trabajará más fácil con voltaje que con carga. Esta configuración conserva el signo positivo o negativo de la señal a la salida del amplificador.

Una vez estudiados las configuraciones básicas y el fundamento del sensor, se puede discutir ya sobre su inclusión o no en el proyecto. Si se piensa en los requisitos de diseño vistos en el anterior apartado, se puede apreciar que cumple con los requisitos de confort y de tamaño, mide la magnitud que se pretende tomar de referencia y demás. Habría que elegir pues las características del siguiente sistema para poder decidir si este sensor es conveniente o no. Se queda, por tanto, la elección del mismo en el aire, hasta que se decida el sistema de captación y sus características.

5.1.3. Sensores de resistencia variable.

Por último dentro de la elección de los sensores, consideramos un tipo, menos extendido que los otros dos, pero que quizá tenga mejor aplicación en el proyecto. Este tipo de sensores, como ya se ha dicho, no es muy común en este contexto. En otros sí, como puede ser LDR (*light dependent resistence*, medir intensidad lumínica), termistores (para medir temperatura), etc.

Pero, en el desarrollo del proyecto, con muchas incógnitas abiertas, el tutor sugirió mirar estos sensores, que habían sido financiados por medio de la fórmula *crowfunding* o *micromecenazgo*, (concretamente, del sitio *kickstarter)*, que es una plataforma donde gente de muchos sectores propone proyectos de innovación y son financiados de forma anónima por personas de todo el mundo que han encontrado el proyecto más o menos interesante. Estos sensores fueron financiados con éxito el 3 de Noviembre de 2.013. Tanto en su página web oficial como en su página de *kickstarter* se pueden encontrar múltiples vídeos y ejemplos de aplicaciones. Esto fue lo que más llamó la atención pues no sólo se trataba de un sensor poco extendido, sino que también era una iniciativa concebida desde el punto de vista académico, en el departamento de ingeniería biomecánica de la universidad de Clemson; que al final es lo que se busca también con este proyecto: formas diferentes de hacer lo que ya existe, dotándolo además del necesario rigor científico.

Este sensor viene para ser totalmente adaptable a las necesidades de cualquier proyecto. Son 5 capas las que hay que ensamblar de la forma que se aprecia en el siguiente esquema, extraído de la página web del sensor:

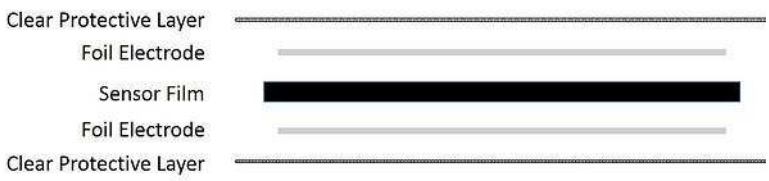


Figura 16: esquema de las capas del sensor. Fuente: página web del proyecto Sensor Film Kit.

Se puede apreciar que en realidad, sólo hay 3 materiales diferentes. Se trata de:

- Una capa autoadhesiva de material transparente y aislante eléctrico. Se sitúa en ambos extremos del sensor, para proporcionar asilamiento al sensor; además de adherirse para compactar las capas.
- Una capa de aluminio conductora. Será necesario dejar accesible dos salientes que harán las veces de conectores para poder acceder a ellos desde el exterior.
- La capa central, la más importante, del material que posee la propiedad esencial para el funcionamiento del sensor. Es un material que ante una presión, cambia su resistencia eléctrica. La idea recuerda a la galga extensiométrica, pero capaz de medir directamente fuerzas de compresión, por tanto, difiere en su utilidad para este proyecto.

El fabricante, por medio de la página web, muestra su funcionamiento de una forma gráfica para que se vea todo más claro en cuanto a este sensor. En la imagen 17 se observa su comportamiento frente a carga mecánica y la salida que aporta el sensor.

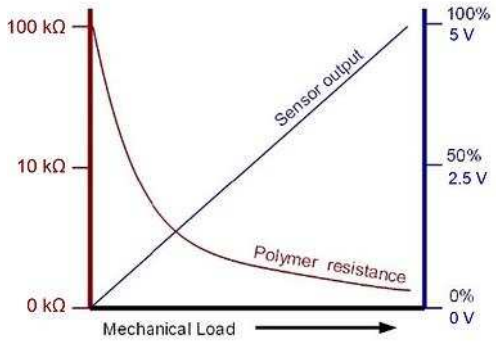


Figura 17: salida del sensor y resistencia del material frente a aumentos de carga mecánica. Fuente: página web del proyecto Sensor Film Kit.

Para conseguir esta linealidad, lo que es necesario hacer es acondicionar el sensor como se muestra en la siguiente imagen:

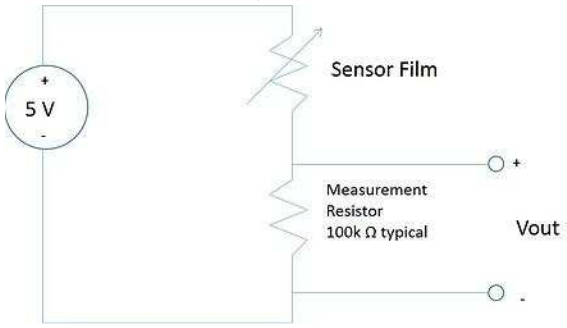


Figura 18: circuito de acondicionamiento de sensor de resistencia variable. Fuente: página web del proyecto Sensor Film Kit.

Es el mismo método que se vio en las galgas extensiométricas, el básico. Solo que, como se ha comentado, al ser diferente el objetivo, también lo es la fiabilidad de este acondicionamiento. En aquella ocasión, se vio que este método no era suficiente, y que era necesario cambiarlo por otro con más precisión, mientras que aquí, en este método, para ajustar la precisión sólo es necesario cambiar el resistor por uno de mayor o menor resistencia. Además, se puede observar en la imagen 17, que la salida es lineal entre 0 y 5 V con este circuito de acondicionamiento.

No obstante, si aún quedasen dudas, se puede seguir indagando en la documentación hasta llegar a la siguiente imagen que nos indica su funcionamiento frente a diferentes magnitudes:

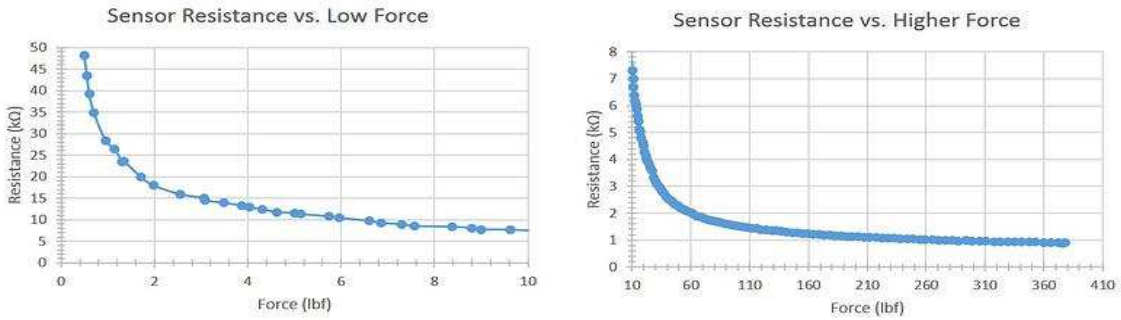


Figura 19: variación de la resistencia del sensor frente a la fuerza aplicada. Fuente: página web del proyecto Sensor Film Kit.

En la gráfica de la izquierda se aprecia la respuesta frente a fuerzas de entre 0 y 44 Newton, aproximadamente; que equivale a unos pocos kilogramos (menos de 5); mientras que en la gráfica de la derecha se muestra frente a fuerzas de hasta 1600 Newton, que suponen más de 160 kg.

De los tres métodos estudiados, se aprecia claramente que este último, posee una simplicidad en el sistema y una respuesta más que deseable. No obstante, hay que considerar el factor novedad bien como un aspecto positivo desde el punto de vista académico (trabajar con algo pionero e innovador) pero quizá no tanto desde el punto de vista práctico. No obstante, si en el desarrollo del proyecto se apreciase la inviabilidad de estos sensores, también sería un paso adelante en el sentido que se observa un error para no repetirlo.

5.1.4. Elección final.

En este punto, sólo existen dos opciones posibles, puesto que las galgas extensiométricas fueron totalmente desechadas. Finalmente, se escogerán los sensores de resistencia variable en base a dos criterios fundamentales:

- Simplicidad en el circuito de acondicionamiento. Si se recuerdan los requisitos de diseño, se comentó que el sistema final no debería ser muy voluptuoso, por lo que es deseable la simplicidad en el circuito de acondicionamiento.
- Funcionamiento claro. Este sensor, aparentemente, presenta un funcionamiento, a priori, más sencillo que el de efecto piezoeléctrico.
- El sensor es novedoso, poco extendido, lo que produce un deseo de buscar nuevas e interesantes aplicaciones, y ésta podría ser una que otros usuarios no hayan considerado.
- De forma práctica, se demostró la invalidez de los sensores piezoeléctricos durante el desarrollo del proyecto. Esta anécdota se narrará en el siguiente apartado, donde se mostrarán los fallos de planteamiento que hubo y por qué la solución debió ser escoger los sensores de resistencia variable.

5.1.5. La anécdota: Problemas con los sensores piezoeléctricos.

Como ya se ha comentado; al iniciar la búsqueda de alternativas para hallar la mejor solución, se dio con los sensores piezoeléctricos. En un primer vistazo, estos sensores parecían perfectos para el propósito; sensores que su deformación la transmitían por medio de los cables en forma de tensión. Se encontraron unos sensores comerciales, relativamente famosos en la red; por lo que se presumió que eran de calidad y cumplían con lo que el fabricante prometía.

Lo que el fabricante afirma de ellos es que se usan para medir deformaciones, contactos, vibraciones e impactos. Su salida ofrecía una tensión de entre 90 y -90 V y con un pequeño rizado de corriente alterna. Con tan solo un resistor, estos 90 V se podrían convertir en una tensión que pueda trabajar un convertidor analógico digital (5 V).

En este caso, interesarían las aplicaciones de contactos y de impactos. No hay nada que indique que no sean válidos para el proyecto; como mucho, el hecho del rizado de tensión, pero que se puede corregir mediante un puente rectificador de diodos o soluciones similares. Otra opción, sería adquirir el valor eficaz de esta tensión, y luego, mediante técnicas de software, ser capaces de saber cuál era el valor inicial del sensor.

Con esta idea en mente, se adquirieron algunas unidades del mismo. Se comenzaron a realizas pruebas, conectándolo directamente a un voltímetro. Ante un incremento de masa, la salida era un pequeño pico de tensión pero que enseguida se volvía a estabilizar en 0. A mayor masa, mayor era el incremento, pero el resultado era el mismo; en unos instantes se estabilizaba en 0. Se comenzaron a hacer, por tanto, pruebas con amplificadores, para estudiar si la señal no se estabilizaría en un valor pero que era tan pequeño que no se apreciaba en el voltímetro.

Con un amplificador operacional (ua741), se probaron diferentes ganancias, todas ellas elevadas, del orden de varios miles. Igualmente, el resultado era el mismo; un aumento inicial, mucho más acusado, pero se estabilizaba en valores próximos a 0; muy probablemente por ruido inducido por la gran ganancia. Estos resultados ya eran bastante raros, pero se pensó en errores o bien en los aparatos de medida, o errores en la forma de adquirir la medida.

Por tanto, se diseñó una primera versión del programa del PIC que permitiese hacer unas pruebas preliminares. Esta versión adquiría los datos de un sensor y los enviaba al PC. Lo obtenido seguía siendo igual; incluso se probaron las configuraciones de amplificador que proponía el fabricante (no sólo ser amplificador de ganancia) vistas en el apartado 5.1.2 y era igual de poco esperanzador.

Se intentaron otras opciones, como poner un condensador en paralelo con el sensor para que el pico de tensión fuese más prolongado y hacer más fácil su medida, cosa que tampoco funcionó. En este punto, tras haber intentado distintas medidas, se concluyó que estos sensores no eran útiles para el propósito. La justificación es que, aunque se sabía de antemano que estos sensores están diseñados para aplicaciones de alta frecuencia (donde se supondrá que sí que producen salidas fácilmente mesurables), se supuso que con algún tipo de técnica sería posible lograr que la frecuencia de la pisada fuese suficiente para detectar el impacto del pie y muestrearlo.

No siendo así, la elección final del sensor se quedó en los sensores de resistencia variable, concretamente el modelo que se habló que había sido recientemente financiado.

5.2. Elección del sistema de visualización.

Como ya se discutió en la parte de requisitos de diseño, es necesario escoger esta parte antes de escoger el sistema de interpretación, fundamentalmente porque al ser el sistema de interpretación el intermediario; es necesario saber de dónde se parte y a dónde se quiere llegar para escoger el mejor intermediario. Por tanto, y en base a los requisitos ya vistos en el apartado 4, se procede a un estudio de las diferentes opciones. Así, las más generalistas serán:

- Osciloscopio
- Pantalla LCD
- Visionado posterior
- Ordenador

Por supuesto, existen otros métodos, pero que bien por sus características o poca extensión, se considera no necesaria su inclusión aquí.

5.2.1. Osciloscopio.

Éste sería un sistema de representación analógica. Se habla de él más que nada para entender que existen otros sistemas que no supongan un paso de analógico a digital, por tediosos que sean. Cabe recordar que un osciloscopio es un sistema usado para la medición de señales electrónicas que varían en el tiempo.

Así, el hipotético uso de un osciloscopio, sería mostrar las variaciones de presión en el tiempo de cada uno de los sistemas de captación. La visualización consistiría en una gráfica que se modificaría su valor pico en función de la presión recibida.

Esta solución plantea muchísimos problemas. Por un lado, un osciloscopio es un elemento de laboratorio, relativamente caro y grande; y serían necesarios tantos osciloscopios como sensores tenga el sistema; que ya se comentó que para conseguir que el proyecto haga la labor deseada, deberá tener varios sensores. Una posible solución (si este fuese el único problema) sería el "crear" un osciloscopio que muestrease múltiples señales para poder compararlas entre sí. Claro que este osciloscopio plantea varios problemas además de su construcción.

En el apartado de requisitos de diseño, se habló de sistemas que fuesen entendibles por todo usuario, no sólo técnicos y personal del área de ingeniería. Por tanto, un osciloscopio, siempre presentará un modo de visualización un tanto tedioso para alguien no acostumbrado a su uso.

Por estas características, se puede concluir que el osciloscopio no es conveniente al proyecto.

5.2.2. Sistema con pantalla LCD.

Este sistema consistiría en una pantalla que mostrase los datos recogidos por los sensores. En función del sistema intermedio (sistema de interpretación), estos datos podrían ser mostrados de una u otra forma.

Cabe destacar en el mundo actual que al hablar de pantallas LCD, también se pueden entender otros sistemas como Smartphones o Tablets, que permite una programación de alto nivel donde la visualización puede ser mucho más amigable para el usuario.

Continuando con la problemática del usuario objetivo de este sistema, se omiten los valores numéricos sin tratar. Sí podrían ser valores numéricos con post procesado, por ejemplo, valores que se convirtiesen en relativos en lugar de absolutos. Se plantean entonces dos posibilidades:

- Que la pantalla LCD sea sólo una pantalla y, por tanto, el sistema de interpretación no sólo tenga que captar los datos y traducirlos, sino que además, deberá indicar cómo mostrarlos a la pantalla LCD.
- Que la pantalla LCD sí tenga sistema operativo, y por tanto, el sistema de interpretación sólo se encargaría de enviar los datos al sistema, donde un software propio los interpretaría y daría una visualización.

De estas dos opciones, la segunda se obviará porque se solapa con la idea de un ordenador.

En cuanto a la primera, los problemas fundamentales son dos:

- Por un lado, cabe recordar que se está buscando un sistema de bajo coste, o al menos, más bajo que las soluciones comerciales que existen de las que se hablaron

en el apartado 3.2. Si se añade un sistema de visualización propio, el producto se encarecerá.

- Por otro lado, un sistema de este tipo, es sólo práctico si el usuario que usa la plantilla es el que observará y analizará los datos; cuando lo más lógico es que el usuario con la plantilla la use de forma despreocupada, mientras que hay otra persona controlando lo que se ve.

Por tanto, se considera que un sistema de estas características tampoco es válido para la consecución del proyecto.

5.2.3. Visionado posterior.

Otra opción, que además es una de las que los proyectos existentes proponen, es el que se guarde lo captado por los sensores en una memoria, y después esto se vuelque en un PC. De esta forma, el usuario podría ir andando algunas horas y después, mediante un software en el PC, mostrar sus hábitos al andar.

Este método es útil sobre todo si se busca la naturaleza a la hora de caminar, pues al ser posible usarlo durante horas, el usuario no se verá sugestionado por estar en un contexto diferente (laboratorio).

Claro, que si se considera esto, también cabe destacar algunos aspectos esenciales adicionales:

- Dotar al sistema de interpretación de memoria, pues los datos deberán ser almacenados en algún sistema. Se deberá decidir si este sistema es extraíble, o se conecta mediante un cable al PC…
- Si se quiere hacer un sistema de estas características, se obligan añadir al sistema una batería o sistema de almacenamiento de energía, suficiente para dar autonomía al sistema y evitar la pérdida de datos en caso de agotarse la batería (si la memoria es volátil).
- Será necesario programar un software que reconozca estos datos y sea capaz de leer una memoria externa o de sincronizarse con el dispositivo.

Al margen de todo esto, un aspecto que no se ha tenido en cuenta es que esta forma de visualización es óptima para cuando se estudia la pisada andando; mientras que, por lo mencionado en el apartado 3, lo que este sistema pretende es un sistema que sea capaz de medir bien el comportamiento de la pisada a baja frecuencia (caminar) y en alta frecuencia (correr).

Entonces, si se considera la acción de correr, se observan más deseables opciones de visualización en tiempo real puesto que, según el tipo de carrera, no harán falta más que unos minutos para estudiar la forma de correr del usuario. También se permiten así comentarios del personal sanitario sobre el usuario, de que modifique uno u otro aspecto; o que acelere la marcha o la ralentice, etc.

5.2.4. Ordenador.

Un ordenador o computador es un dispositivo capaz de recibir y procesar datos para convertirlos en información útil. Se caracteriza por un conjunto de circuitos integrados que funcionan como dispositivos independientes con funciones exclusivas gestionados por un microprocesador.

Esta característica hace que en función de los diferentes circuitos integrados que acompañan a este microprocesador, la función del ordenador puede verse ampliada con diversos tipos de periféricos. Así, tenemos un dispositivo con grandes posibilidades de comunicación con dispositivos de entrada, de salida o mixtos, y ampliable.

Un ordenador estándar suele estar compuesto por una unidad central, la CPU; y algunos dispositivos de comunicación con el usuario, los más básicos suelen ser teclado, pantalla y ratón. A parte de estos, se le puede conectar un sinfín de dispositivos más en función del uso que queramos para la máquina, como altavoces si está enfocado al ocio, impresoras si su aplicación es ofimática, etc.…

Es esta característica la lo hace, probablemente, la mejor opción para el objeto del trabajo:

- Por un lado, el hecho de que la comunicación entre usuario y proceso sea mediante teclado, pantalla y ratón, nos proporcionan la posibilidad de crear una interfaz tremendamente simple e intuitiva, además de el ordenador es una máquina de uso cada vez más común entre usuarios.
- Por otro lado, la posibilidad de conectar distintos tipos de dispositivos, hacen que un ordenador estándar pueda ser ampliado mediante un dispositivo para conseguir que este ordenador interactúe con el medio, es decir, conectar a este ordenador un dispositivo capaz de medir entradas y actuar en función de ellas aprovechando su capacidad de procesamiento.

El ordenador en un proyecto en el que estamos mezclando disciplinas es, claramente, una opción más que destacable. Es un elemento que con el paso de los años, se ha convertido esencial en el día a día de cualquier persona, especialmente en el ámbito laboral. Proporciona versatilidad, sencillez de usar…

La idea será la de cablear el dispositivo de interpretación desde el pie del usuario hasta el PC donde esté instalado el hipotético software. El usuario, por tanto, deberá correr en una cinta transportadora o similar para la recogida de datos.

Otra desventaja que posee es derivada también de su mayor ventaja: la libertad que posee hace que se necesiten muchos pasos para lograr vincular los tres sistemas. No obstante, el proceso de configuración se dará en cualquiera de los sistemas que elijamos, pues ninguno está destinado a la aplicación concreta de este proyecto.

5.2.5. Elección final.

Se ha visto que se abren dos opciones muy diferentes y ambas muy válidas. Por un lado, está la idea de elaborar un sistema portable, que el usuario pueda llevar mientras se recogen los datos, y después volcar los datos en un ordenador; mientras que también está la opción de un sistema en tiempo real, que pueda ser controlado y comentado por el personal mientras el corredor trabaja sobre una cinta transportadora o un sistema similar.

Habiendo visto ya lo anteriormente descrito, queda tomar la decisión entre las dos opciones. En ambas, un PC será el sistema de visualización, por lo que el sistema queda fijado. No obstante, es necesario elegir también el medio en el que se tomarán las medidas para poder proseguir con la elección del sistema de interpretación. Se escoge la solución de sistema en laboratorio por las siguientes consideraciones:

- Se ahorra el gasto de batería + memoria, además de las complicaciones derivadas (una programación más complicada, elección de batería y voltajes, autonomía de la misma…).
- Muchas pruebas médicas de este tipo son efectuadas en laboratorios, que también suelen estar equipados con cintas para correr (por ejemplo, pruebas de esfuerzo cardiovascular; reconocimiento médico de atletas…) por lo que no supone un gran cambio en la filosofía de esta ciencia.
- La ventaja del tiempo real, donde mientras el corredor está sobre la cinta, el personal puede estar observando simultáneamente la pisada en el monitor del PC y en lo que sus ojos ven, por lo que se pueden ahorrar problemas como falsos positivos o comportamientos anómalos en el sistema.

Con esta decisión tomada, se puede proseguir con el sistema intermedio entre el sistema de visualización y el de captación.

5.3. Elección del sistema de interpretación.

Previamente al análisis, conviene recordar la misión de este sistema. Debe encargarse de recoger la salida de los circuitos de acondicionamiento estudiados en el apartado anterior e interpretar estas señales. En este punto, ya son conocidos los otros sistemas que formarán el proyecto, con lo que lo que queda es “rellenar huecos”.

Del primer sistema, se obtendrá una tensión, que variará entre los valores que se establezcan. Pero dependerá de una fuente de alimentación externa. El sistema de interpretación probablemente también necesite alguna fuente de alimentación; por lo que una posible solución es que ambos sistemas compartan la fuente de alimentación.

Del segundo sistema se sabe que será un PC, por lo que el sistema de interpretación necesitará tener comunicación con el mismo. También se sabe que el PC es un sistema tarde o temprano, necesitará transformar la información de analógica a digital. Según el sistema, este paso se hará antes o después.

Hay múltiples sistemas que no se verán en este análisis, como pueden ser los autómatas programables, que también podrían cumplir los requisitos; pero que ofrecen Los sistemas que fueron estudiados son:

- Tarjetas de Adquisición de Datos.
- Circuito con componentes discretos.
- Microcontroladores.

5.3.1. Tarjetas de Adquisición de Datos.

Una vez que se sabe el computador como sistema de visualización, se presenta interesante este tipo de tarjetas.

Una tarjeta de adquisición de datos es una tarjeta que, conectada a una unidad de procesamiento, permite la adquisición de datos, que consiste en la toma de muestras del mundo real (sistema analógico) para generar datos que puedan ser manipulados por un ordenador u otros dispositivos electrónicos (sistema digital). En esencia, toma un conjunto de señales físicas, las convierte en tensiones eléctricas y las digitaliza de manera que se puedan procesar en una computadora. Se requiere una etapa de acondicionamiento, que adecúa la señal a niveles compatibles con el elemento que hace la transformación a señal digital. En ocasiones, las señales que toma del exterior no son solo señales analógicas, sino que también toma otras señales digitales, cuyo procesamiento es mucho más sencillo que el de las señales analógicas, aunque es un caso que en este proyecto no se dará.

Son tarjetas que se conectan a puertos estándar el PC, como puede ser el puerto PCI (*Peripheral Component Interconnect*), por lo que en el PC será necesario crear un software que interprete los datos a la hora de la visualización, aunque ya se discutió que este paso será necesario, independientemente del sistema de interpretación.

Donde sí habrá diferencia es en la forma de programar, pues en el caso de este tipo de tarjetas, el propio fabricante ofrece librerías con funciones y demás para poder acceder directamente a la tarjeta mediante el software, así como software auxiliar que ayuda en la configuración de la misma y a entender su funcionamiento.

El problema más grande que se deriva de este tipo de sistemas, es que los sensores se cablearían individualmente desde la planta del pie hasta la tarjeta, lo que supone el doble de cables que de sensores que habrá, lo cual es muy tedioso, Otro gran problema es el precio, pues cabe recordar que se está buscando un sistema lo más barato posible y estas tarjetas, suelen ser caras, variará su precio en función de las características, pero una tarjeta de gama media pueden ser en torno a 400 €, lo que prácticamente descarta este sistema de este trabajo.

5.3.2. Circuito con componentes discretos (basado en convertidor AD).

Uno de los pasos esenciales será la conversión analógico digital. Se presentan por tanto deseables los sistemas modulares, que además se pueden confeccionar a medida de cada necesidad. Un diagrama de bloques de un posible circuito en este proyecto sería:

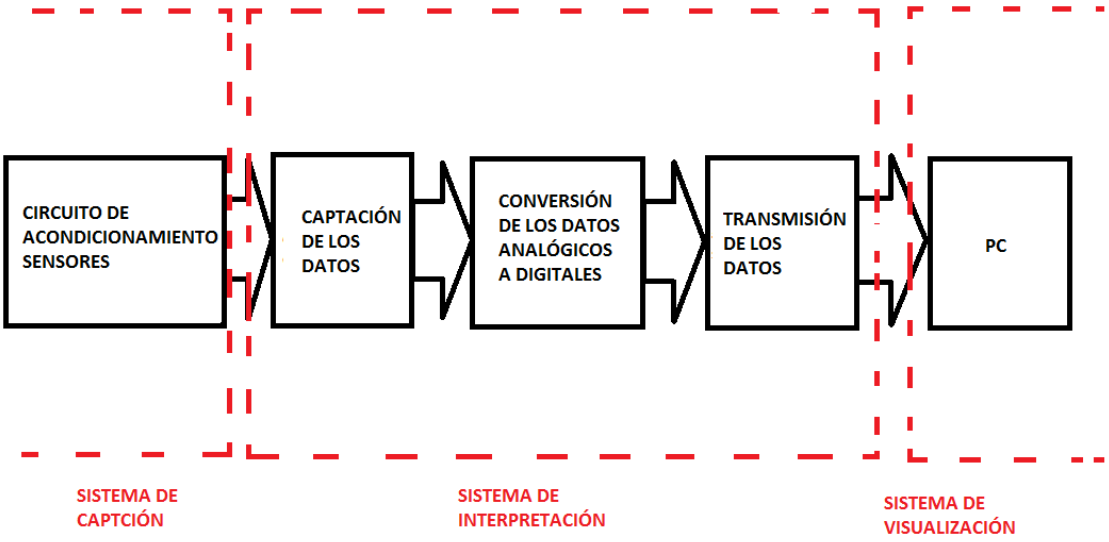


Figura 20: diagrama de bloques del sistema.

Entonces, en caso de usar esta solución, sería necesario escoger y configurar cada uno de los pasos. El más importante de todos es la conversión de los datos analógicos a digitales, sin descuidar los otros pasos. Es el más importante puesto que condicionará los otros dos, por ejemplo, si el convertidor tiene 8 entradas, y la plantilla tiene 12 sensores, será necesario emplear un multiplexor para poder alcanzar el número de entrada necesario.

Algo similar ocurrirá en la salida, pues el número de salidas condicionará y posibilitará la transmisión en uno u otro método (transmisión en serie, en paralelo) así como el tipo de puerto. La transmisión en paralelo es bastante antigua y poco usada, y muchos ordenadores ya no traen disponibilidad para esta conexión, por lo que será obviada.

La salida del convertidor analógico digital deberá ser recogida pues por un circuito que sea capaz de enviar los datos a un PC y dotar de sincronía a los mismos. Se abren también muchas posibilidades en el sentido que pueden ser métodos cableados o inalámbricos, como Bluetooth, USB, etc…

En realidad, este sistema es muy deseable, sobre todo en el sentido de que se puede adaptar 100% a lo que se necesita en este proyecto, evitando hacer gastos innecesarios en funciones que no serán utilizadas. No obstante, es tedioso hoy en día crear un circuito de estas características, puesto que, como se verá en la siguiente opción, existen soluciones que pueden cubrir todas las necesidades de forma sencilla. Pero se considera necesario tomarlo como opción.

Es más, pensando en posibles ampliaciones, en un caso de que el producto fuese elaborado a escala industrial, lo más conveniente quizá fuese hacerlo mediante circuitos propietarios de este tipo.

5.3.3. Microcontroladores.

Tras lo visto anteriormente, los microcontroladores se presentan como la mejor opción, probablemente. Un microcontrolador es un circuito integrado programable, capaz de ejecutar las órdenes grabadas en su memoria. Está compuesto de varios bloques funcionales,

los cuales cumplen una tarea específica. Un microcontrolador incluye en su interior las tres principales unidades funcionales de una computadora: unidad central de procesamiento, memoria y periféricos de entrada/salida.

Un microcontrolador típico tendrá un generador de reloj integrado y una pequeña cantidad de memoria RAM y otra pequeña cantidad de memoria ROM, con lo que para hacerlo funcionar todo lo que se necesita son unos pocos programas de control y un cristal de sincronización. Los microcontroladores disponen generalmente también de una gran variedad de dispositivos de entrada/salida, como convertidor analógico digital, temporizadores, UARTs y buses de interfaz serie especializados, como I2C y CAN. Frecuentemente, estos dispositivos integrados pueden ser controlados por instrucciones de procesadores especializados. Los modernos microcontroladores frecuentemente incluyen un lenguaje de programación integrado, como el lenguaje de programación BASIC que se utiliza bastante con este propósito. Si no, será necesario introducir el programa de control mediante código hexadecimal; o hacerlo mediante software proporcionado por el fabricante.

Existen multitud de fabricantes, y cada uno difiere en características, métodos de programación, etc… Esta será una de las decisiones que será necesario tomar y que será condicionada por las otras partes del sistema.

El microcontrolador integra en su sistema todos los bloquees vistos en la imagen 20, característica que los hace tan deseables, puesto que, recordando el requisito del circuito pequeño, cumplirá mejor un único circuito integrado que englobe a todas las partes antes que un circuito difuso, que además será necesario establecer las comunicaciones entre ellos mediante cableados.

5.3.4. Elección final.

Vistos estos tres sistemas, sólo queda tomar la decisión. Se tienen dos opciones casi empatadas en utilidad, que son la del microncontrolador y la del circuito difuso basado en ADC. Para encontrar la mejor solución, lo más indicado es volver a los requisitos de diseño y estudiar cuál de ellos cumple mejor:

- Desde el punto de vista del tamaño, gana claramente el microncotrolador puesto que es un único circuito frente a varios circuitos de la otra solución, aunque no se sabe bien cuántos, se sabe que, al menos, constará de dos o más. También, supondría cablear más conexiones con los posibles fallos que pueda dar esto (malas soldaduras, captación de ruido,…)
- Desde el punto de vista económico, se puede hacer un estudio sencillo, como el mostrado en la siguiente tabla:

Circuito basado en ADC		Microcontrolador	
	Precio (€)		Precio (€)
ADC 0801L	14,58	PIC 18F2458	5,70
PL-2303	14,18		
Total:	28,76 €	Total:	5,70 €

Tabla 1: comparación de precios entre circuito basado en ADC y un circuito con microcontrolador.

Siendo el circuito ADC 0801L el convertidor analógico digital; el circuito PL-2303 el que se encargaría de la comunicación con el PC y faltando por tanto, la interfaz de entrada y otros elementos que no se han incluido en el estudio debido a que no es necesario ahondar más en la cuestión. Es un estudio muy simple, pero hecho en base a tiendas de electrónica reales (http://www.dieltron.com/) y, aunque hay muchos aspectos que no se han tenido en cuenta, se puede observar claramente cuál va a ser a la larga más rentable económicamente.

Por tanto, la solución final escogida, será basar el sistema de interpretación en un microcontrolador, por las ventajas económicas y de espacio que parecen mejores para este trabajo. Será necesario, no obstante, programar el microcontrolador, paso que no sería necesario en el otro caso; pero aún así, es un esfuerzo que parece asumible, frente a los otros esfuerzos que nos requerían las otras soluciones.

5.4. Elección de la frecuencia del sistema.

El tema de frecuencia es bastante delicado, puesto que cada elemento tiene unas frecuencias de trabajo según la cual podrá trabajar de una u otra forma. Estos elementos serán:

- Los sensores. Tendrán una capacidad de respuesta que variará según el tipo escogido. Habiendo escogido los resistivos, la forma más rápida de adquirir esta información es consultando directamente al fabricante.
- El circuito de acondicionamiento, aunque en este caso sea sólo un resistor, también se deberá de llevar precaución puesto que a altas frecuencias pueden aparecer efectos no deseados (capacidades parásitas por ejemplo).
- El microcontrolador, debe tener tiempo suficiente de captar la señal y convertirla a un valor digital. Además, debe ser capaz de comunicarse con el PC a una velocidad suficiente que permita la transmisión de todos los datos, sin pérdidas de por medio, lo que también puede afectar al sistema de transmisión.
- El software diseñado para el PC, deberá ser también lo suficientemente rápido para analizar los datos que sean enviados por el microcontrolador.

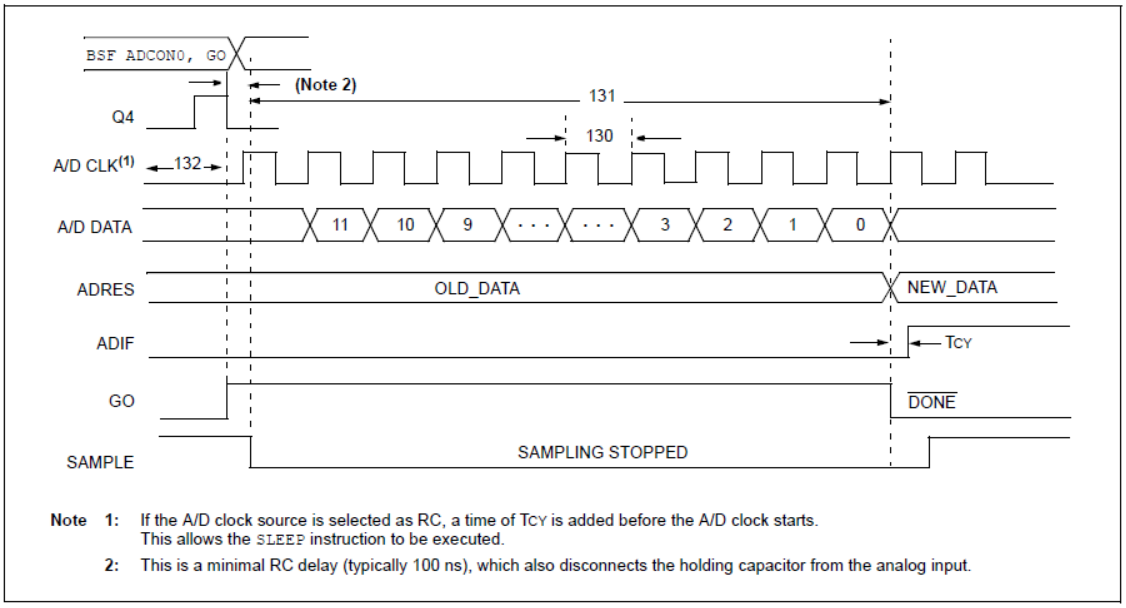
Un error en que se puede incurrir al principio es que se debe muestrear sólo cuando pisa el corredor o paciente; pero esto no es así; si no que se deben muestrear los cambios que se producen durante la pisada. Lógicamente, ésta será entonces una frecuencia mucho más alta que de la otra forma, que apenas serían unos 5 Hz (que serían 5 pasos por segundo y pie).

Lo que se puede hacer para establecer una frecuencia aproximada, para después ver si el sistema es capaz de cumplir con ella, es observar los sistemas tratados en el apartado 3.2,

donde se habló de sistemas ya existentes diseñados para problemáticas similares. En el primero de los que se trataron, para medir presiones ambulatorias, su frecuencia era de 25 Hz; y aseguraban que era posible con esta frecuencia muestrear los cambios de presión durante la marcha normal. En el segundo sistema, hablan de que posee frecuencias de muestreo variable; por lo que es de suponer que según el tipo de carrera que se quiera medir ésta podrá ser mayor o menor. Se cita un ejemplo, en el que la frecuencia es de 10 Hz, aunque no especifica el tipo de carrera. El tercer caso, y de forma casi evidente por ser la solución comercial, es el más exigente, puesto que su frecuencia de muestreo llega hasta los 100 Hz.

Es evidente que alcanzar la frecuencia de las plantillas Biofoot sería lo óptimo, puesto que así se aseguraría igualar a un sistema comercial y probado en pacientes, pero hay que ser realista antes y estudiar cada componente de forma individual para determinar si es posible alcanzar este objetivo en este sistema. Estudiamos pues de forma independiente los elementos:

- Los **sensores**, como ya se dijo, se consultará la documentación del fabricante. En ella la información es escasa y no aparece, lo que obligará a realizar pruebas sobre ellos.
- El **microcontrolador**. En su programación y configuración, se puso que trabajase a 4 MHz, pero ésta no es la frecuencia de captación del sistema. Para averiguarla, se ha de estudiar la velocidad del convertidor analógico digital, que es lo que marcará la frecuencia máxima de trabajo. También se debe acudir a la documentación del fabricante, en este caso, las hojas de características del PIC. De la hoja de características se obtiene la información reflejada en la imagen 21. En la parte inferior de la imagen, se aprecia que los microcontroladores de la familia 18FXXXX (como los que serán usados) el periodo máximo es de 12,5 µs. Se aprecia en la parte superior que cada ciclo se corresponde aproximadamente con la conversión de un bit, por lo que se darán dos casos:
    - Si el convertidor es de 10 bits:
      $$12{,}5\frac{\mu s}{ciclo} * 1\frac{bit}{ciclo} * 10\frac{bits}{conversión} = 125\frac{\mu s}{conversión}$$
      Como habrá 13 canales, la conversión llevará 125*13 = 1625 µs, que se corresponde con una frecuencia de **615,38 Hz.**
    - Si el convertidor es de 12 bits:
      $$12{,}5\frac{\mu s}{ciclo} * 1\frac{bit}{ciclo} * 12\frac{bits}{conversión} = 150\frac{\mu s}{conversión}$$
      Como habrá 13 canales, la conversión llevará 150*13 = 1950 µs, que se corresponde con una frecuencia de **512,82 Hz.**



Note 1: If the A/D clock source is selected as RC, a time of TCY is added before the A/D clock starts. This allows the SLEEP instruction to be executed.

2: This is a minimal RC delay (typically 100 ns), which also disconnects the holding capacitor from the analog input.

Param No.	Symbol	Characteristic		Min	Max	Units	Conditions
130	TAD	A/D Clock Period	PIC18**F**XXXX	0.8	12.5(1)	µs	TOSC based, VREF ≥ 3.0V
			PIC18**LF**XXXX	1.4	25.0(1)	µs	VDD = 3.0V; TOSC based, VREF full range
			PIC18**F**XXXX	—	1	µs	A/D RC mode
			PIC18**LF**XXXX	—	3	µs	VDD = 3.0V; A/D RC mode
131	TCNV	Conversion Time (not including acquisition time)(2)		13	14	TAD	
132	TACQ	Acquisition Time(3)		1.4	—	µs	
135	TSWC	Switching Time from Convert → Sample		—	**(Note 4)**		
137	TDIS	Discharge Time		0.2	—	µs	

Figura 21: cronograma y tabla de características del convertidor analógico digital del microcontrolador. Fuente: hojas de características del microcontrolador 18F4550/18F2458.

- La **velocidad de comunicación** fue establecida en 9600 baudios, puesto que el microcontrolador está simulando un puerto serie tradicional. Por la programación del microcontrolador, éste necesita transmitir en cada ciclo de programa 76 caracteres correspondientes a todos los valores de los sensores y otros caracteres adicionales necesarios. Por lo tanto, con una pequeña ecuación, se puede averiguar la frecuencia máxima de envío de datos que soportará el sistema:

$$\frac{9600\ baudios * 1\frac{bit}{baudios * s}}{76\frac{bits}{ciclo}} = 126{,}31\frac{ciclos}{segundo}$$

  Es decir, que el microcontrolador podría enviar hasta 126 veces durante un segundo la información de todos los sensores.

- El **software del PC**, que es de suponer que no habrá problema en su tiempo de ejecución por la velocidad de los sistemas actuales; sí que debe ser configurado para recibir todos los datos que envíe el microcontrolador. Para crear los bucles necesarios durante la ejecución del programa sí que será necesario establecer los tiempos de

espera (*delays*) de forma que posibiliten la lectura secuenciada del microcontrolador sin pérdida de datos y con la frecuencia necesaria de lectura.

5.5. Resumen de la solución escogida.

Antes de hablar de aspectos más técnicos, se hará un resumen de lo hablado en estos apartados anteriores para exponer de una forma clara y concisa la solución final acordada. Dicho de otra forma, en este apartado se rellenarán las incógnitas que se habían dejado en el apartado de requisitos de diseño.

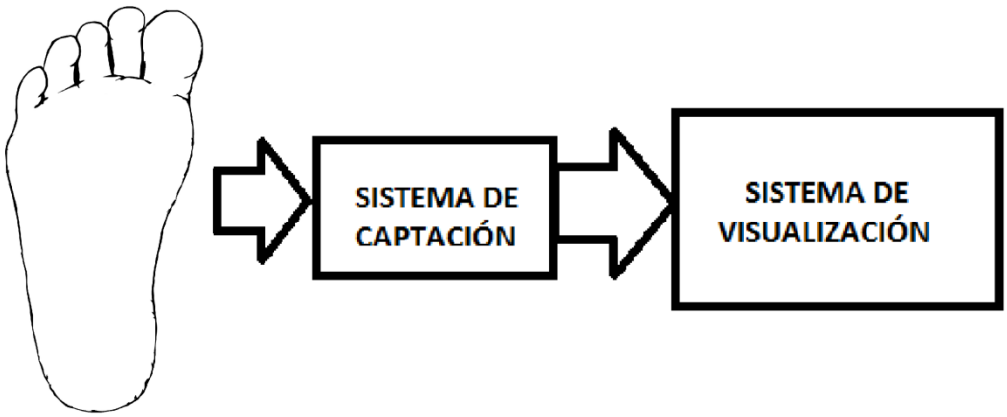


Figura 7: diagrama de bloques del supuesto sistema.

Si se retoma la imagen 7 vista en el apartado 4, ya se pueden poner nombres a esos elementos que aparecen genéricos, formando entonces la imagen 22:

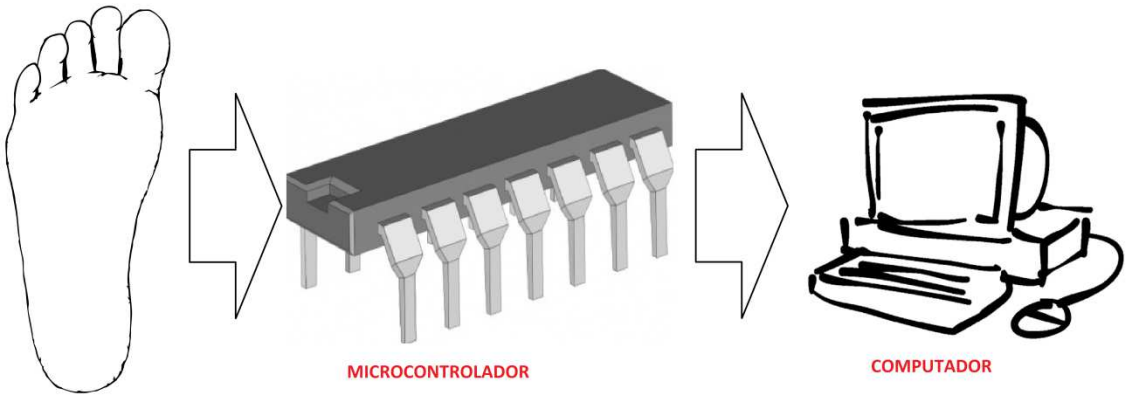


Figura 22: diagrama ilustrativo de la solución final.

- Como sistema de captación:

  Se usarán sensores que varían su resistencia en función de la carga que soportan. Al ser sensores de resistencia variable, para poder ser medidos, se dispondrán en un circuito de acondicionamiento con una fuente de alimentación de corriente continua y una resistencia de valor estático. Se usarán 13 sensores para garantizar cubrir lo mejor posible la superficie del pie.

- Como sistema de interpretación:

  Se usará un microncontrolador que necesitará de las siguientes características:
  - Módulo de entrada/salida en tensión analógicos.
  - Convertidor analógico digital de, al menos, 10 bits.
  - Comunicación con PC por puerto USB.

  De esta forma, se cubrirán todas las necesidades vistas en los apartados anteriores.

- Como sistema de visualización:

  Se usará un computador convencional, ya que el único requisito es tener puerto USB que, en la actualidad, todos poseen varios. Otro requisito es la pantalla, pero es imposible el funcionamiento de un PC sin pantalla. Los requisitos del sistema operativo y otras características se verán más adelante cuando se haya realizado la programación del software.

6. Desarrollo de la solución final.

Bien definido el problema y los objetivos, queda enlazarlo todo mediante lo que será el grueso del trabajo realizado y diseñar el producto. Se puede establecer ahora un calendario según el que guiar el desarrollo del proyecto:

Primer paso: conseguir los sensores de resistencia variable. Sobre ellos, realizar pruebas para entender mejor el funcionamiento, poder ser capaces de cuantificar los datos que sean necesarios y poder configurar el circuito de acondicionamiento de forma correcta.

Segundo paso: elegir un microcontrolador adecuado a las necesidades planteadas entre las diferentes marcas y modelos. Sobre él, programar una rutina que sea capaz de adquirir los datos de los sensores que sean necesarios. Ajustar la frecuencia de trabajo para que sea suficiente.

Tercer paso: establecer la comunicación entre el computador y el microcontrolador y comprobar su fiabilidad. Comprobar que los datos sean correspondidos con los recogidos por los sensores.

Cuarto paso: elaborar sobre el PC un software capaz de recoger los datos enviados por el puerto USB y que muestre al usuario estos datos de forma visual.

Se puede apreciar que son pasos sucesivos con respecto al sistema, es decir, se sigue aquella idea anteriormente citada “del pie a la máquina”. Por tanto, este proceso se detallará dividiendo este apartado en cada uno de los sistemas.

6.1. Sensores.

Retomando las ideas vistas en el apartado de análisis de soluciones, se aprecia que el primer problema es el “montar el sensor”. Se puede recuperar la imagen que ya se vio en dicho apartado:

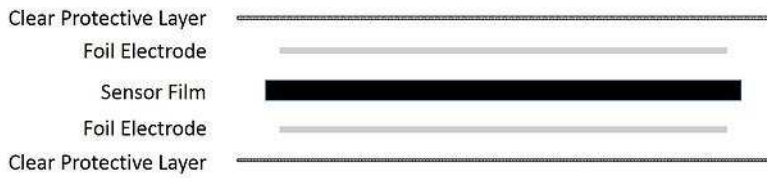


Figura 23: esquema de las capas del sensor. Fuente: página web del proyecto Sensor Film Kit.

El sensor es suministrado en esta forma, cinco capas de tres materiales distintos, que ofrecen la posibilidad de ser cortado a gusto del consumidor; no obstante, las limitaciones de espacio fueron las que impusieron un tamaño aproximado.

Son dos láminas transparentes autoadhesivas que dentro de sí encierran dos láminas de aluminio, que harán la labor de electrodos y se deberán de cortar de forma que queden los contactos fuera de la superficie de material transparente; y por último el material sensible, de color negro, que es el que soportará los esfuerzos. En la imagen 24 se pueden ver estas 5 capas tal y como las proporciona el fabricante; y en la imagen 25 se aprecian algunos de los sensores montados como aparecerán finalmente sobre la plantilla.

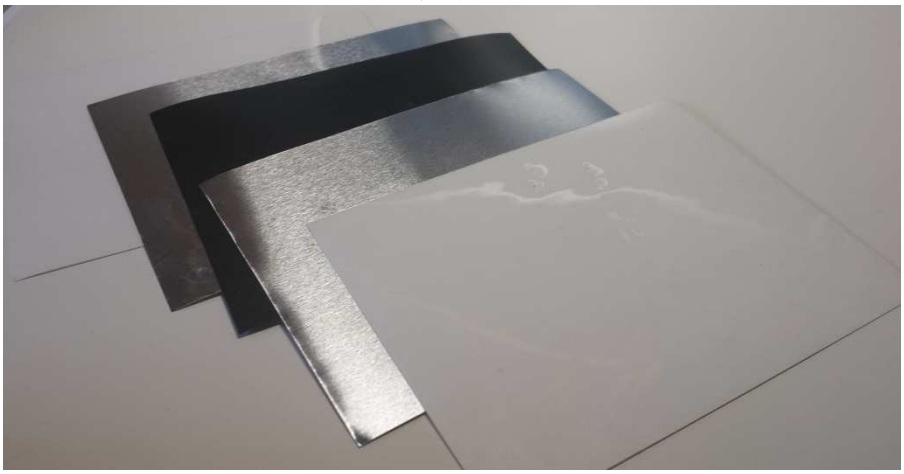


Figura 24: fotografía de las capas del sensor tal y como las proporciona el fabricante.

Figura 25: fotografía de algunos sensores ya montados antes de ser cableados e insertados en la plantilla.

Una vez que están montados, se realizan pruebas sobre ellos. Con un polímetro simple, se puede apreciar que al ejercer una presión sobre ellos varía su resistencia. Si se monta en conjunto con un circuito de acondicionamiento como el que se ve en la imagen:

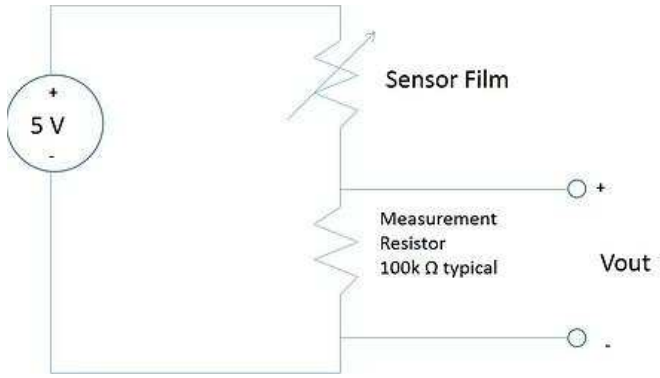


Figura 18: circuito de acondicionamiento de sensor de resistencia variable. Fuente: página web del proyecto Sensor Film Kit.

Pero modificando el valor de la resistencia, a la salida lo que se obtiene en lugar de una variación de resistencia se tiene una variación de tensión entre 0 y 5 V, que es lo deseable puesto que este valor sí puede ser leído por las entradas analógicas de un microcontrolador.

Cuando la presión es muy grande, la resistencia del sensor disminuye mucho, de forma que la tensión de la fuente recae únicamente en el resistor de medida, dando a su salida prácticamente 5 V. Si no hay presión sobre el sensor, al contrario, lo que se produce es una resistencia muy grande que provoca que no haya prácticamente caída de tensión en el resistor de medida, y ofrece 0 V a la salida del circuito.

Este circuito deberá ser replicado tantas veces como sensores se tengan en el proyecto (en virtud de lo hablado en el apartado 4), pero tanto la fuente como la referencia a tierra, pueden ser comunes. La fuente de tensión se obtendrá del voltaje que es capaz de proporcionar el ordenador, es decir, unos 5 V, de corriente continua, estables. De esta forma, el circuito se ahorra el poseer alimentación externa, que sí sería necesaria en otro tipo de comunicaciones y/o métodos.

Por tanto, confeccionando este circuito n veces, se tendrán cubiertas las necesidades de captación.

Para poder vincular el circuito con el pie, la solución es sobre unas plantillas de las tradicionales para calzado, situar sobre ella los sensores en una disposición que permita medir adecuadamente las presiones. De lo visto en el apartado 3, se sabe que las partes del pie que soportarán más presiones son:

- El talón, sobre todo al inicio del ciclo de marcha al andar o al correr.
- La zona de los metatarsianos, justo antes del comienzo de las falanges.
- La zona de las falanges.
- A lo largo de puente mayor, en los laterales, aparecen también en ciertos instantes acumulación de presiones.

Con las limitaciones de tamaño y de las herramientas de las que se disponen, el tamaño de los sensores serán aproximadamente de unos 3 cm de largo y sobre 1,5 de ancho. Adicionalmente, llevan todos los sensores dos cables. Esto provoca que, aunque sea deseable alcanzar tantos sensores como las Biofoot/IBV (32 sensores por plantilla, como se vio en el apartado 3.2.3), no es fácil al tratarse de una solución discreta (es decir, no siendo un proceso largo y cuidado de producción). Esto viene a significar que no se podrán disponer tantos sensores en la plantilla como se quieran, si no que ese número está restringido por distintos factores.

Así mismo, también hay que considerar que el convertidor del PIC posee tan solo 13 canales; es decir, que poner sensores adicionales es meter circuitería adicional. Esta circuitería tiene 3 grandes desventajas:

- Aumenta el coste del sistema.
- Reduce la frecuencia máxima de trabajo.
- Aumenta el tamaño del circuito.

De donde se puede deducir que solo por ganar algunos sensores más, no merecen la pena estas pérdidas; puesto que estos sensores adicionales no es probable que ofrezcan información crucial con respecto a una plantilla de “solo” 13 sensores. Por tanto, queda fijado el número en 13 sensores, con sus 13 resistencias del circuito de acondicionamiento.

La salida del sensor se hace lineal al trabajar con una resistencia de medida, independientemente del valor que sea. Lo que determina la resistencia es el fondo de escala. Por las comprobaciones hechas sobre uno de los sensores ya montados, se corrobora lo que dice el fabricante: cuanto mayor sea el valor de la resistencia, mayores serán las variaciones de tensión producidas por un incremento de fuerza. La justificación está en que si el material reduce su resistencia con la fuerza aplicada, la variación de la salida será inversamente proporcional, y aumentará, pues la caída de tensión en la resistencia fija se hace cada vez más grande. Si esta resistencia es mucho mayor, una variación pequeña de fuerza sobre el sensor provoca una caída mucho mayor.

Pero esto son aspectos generales; si se centra la atención en este proyecto, lo que se busca es un fondo de escala amplio, pero no en exceso; es decir, un humano nunca provocará fuerzas muy elevadas y no tendría sentido poner un resistor muy pequeño que posibilite la medida de estas fuerzas, perdiendo resolución. Tampoco sería lógico poner uno muy grande, que permita medir fuerzas pequeñas pero con grandes incrementos, donde fuera inapreciable la diferencia entre dos puntos que soporten presiones similares pero no iguales.

Para determinar el valor de esta resistencia se montó un sensor y se dispusieron varias resistencias para comparar la salida obtenida del sensor. Sobre él se situaron pesos conocidos para observar la evolución de la salida. En la siguiente tabla se pueden apreciar los resultados obtenidos:

Valor de resistencia (Ω)	Masa (kg)	Fuerza (N)	Salida (V)
138	1	9.8	0
	2	19.6	0
	5	49	0
1.4 k	1	9.8	0
	2	19.6	0
	5	49	0
120 k	1	9.8	0.322
	2	19.6	0.674
	5	49	1.76
14 M	1	9.8	0.72
	2	19.6	1.62
	5	49	3.08

Tabla 2: resultados experimentales sobre sensor.

Con un sencillo cálculo, se puede apreciar que las del orden de cientos de ohmios y las de 1.4 k no ofrecen resultados, así como las de 14 M son demasiado, puesto que alcanzaría los 5 V (máximo) a unos 10 – 15 kg, por lo que no permitiría medir masas de humanos de talla media.

Se puede, por tanto, deducir que el orden de magnitud para las resistencias será de unos cientos de kilo ohmios. Se podría coger otro valor (250k, 300k) pero no tendrán gran influencia sobre el sistema, más que acotar por arriba la fuerza máxima que admite el sensor.

6.2. Microcontrolador.

Antes que nada, se buscará en la base de datos de una empresa como Microchip entre los microcontroladores que puedan convenir al propósito del proyecto, en base a lo visto en el último apartado de la sección análisis de soluciones. Allí se vio que, además de las características que suelen llevar, serían necesarias:

- Módulo de entrada/salida en tensión.
- Convertidor analógico digital de, al menos, 10 bits.
- Comunicación con PC por puerto USB.

Esto restringe las posibilidades, pues no todos llevan un convertidor de tantos bits, o no todos tienen posibilidad de comunicación USB. Hay dos modelos muy usados que son los que más parecen convenientes para el proyecto; el PIC 18F2458 y el PIC 18F4550. La mayor diferencia entre ellos será el número de bits del convertidor analógico digital, pero otras como el tamaño del encapsulado y demás características harán tomar una u otra decisión.

Nota: de ahora en adelante, se usarán indistintamente la palabra PIC y microcontrolador, debido a que los microcontroladores de la marca Microchip son denominados también PIC.

Ya se ha decidido que la pieza fundamental del proyecto será un microncontrolador, quedará por tanto programar éste para que envíe los datos que necesitamos al ordenador y que sea capaz de recibir los datos de los sensores, adecuarlos e interpretarlos.

Aunque no se ha hablado anteriormente de ello, también serán necesarios ciertos componentes de hardware para posibilitar el funcionamiento de un microcontrolador; que se describirá en el siguiente subapartado, y una parte de programación, de la que también se hablará extensamente.

Finalmente, el microcontrolador elegido fue el 18F4550, ya que éste posee 40 patillas en contraposición a las 28 del otro; lo que nos permite acceder a todos los canales del convertidor analógico digital.

6.2.1. Parte de hardware:

En la imagen 26 podemos ver las partes esenciales que necesitará el circuito para funcionar. En su mayoría, son condensadores que dotan de estabilidad al sistema, pero también se aprecia un componente de grandísima importancia, un cristal.

Los cristales son elementos que al atravesarlos una corriente, vibran a una frecuencia determinada, convirtiéndose así en la señal de reloj del sistema. Éste puede ser de cualquier valor comercial, pues luego es mediante la programación del PIC donde se indicará la frecuencia de trabajo del sistema y si hay que usar algún tipo de divisor para conseguir esta frecuencia.

Los condensadores también pueden tener valores diferentes y, aún así, garantizar el funcionamiento del sistema. No obstante, lo más cómodo, es consultar las instrucciones que da el fabricante y observar los valores que recomienda.

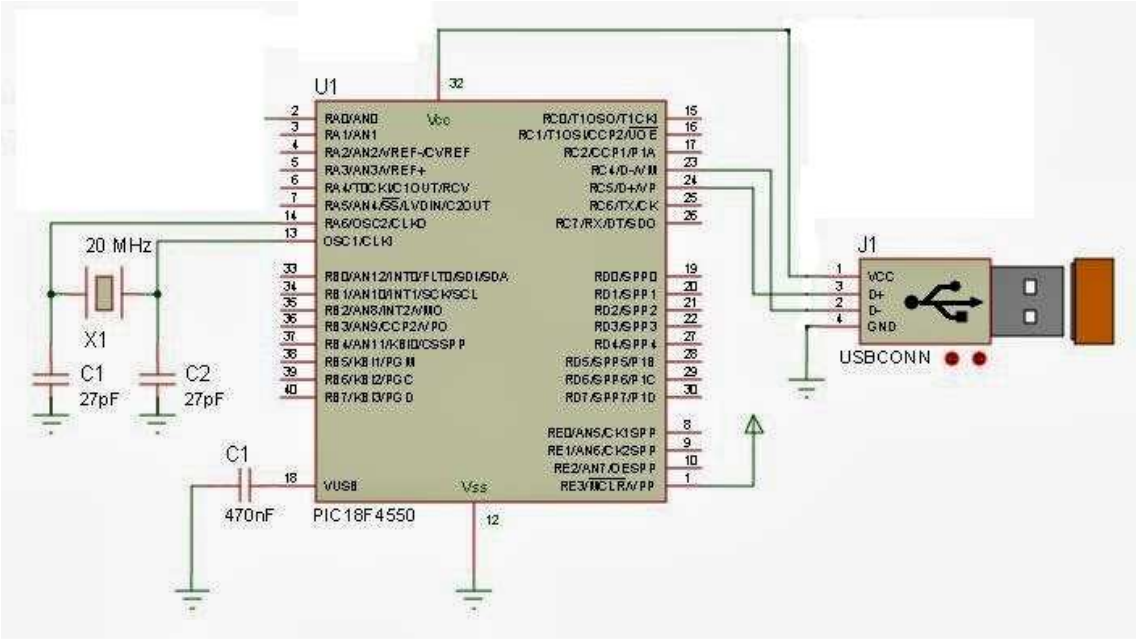


Figura 26: circuito mínimo para conexión USB del microcontrolador.

En la imagen 26 se aprecia el controlador 18F4550, pero la nomenclatura del PIC no cambiará independientemente de la decisión final entre este modelo o el otro.

También es importante cablear de forma correcta la conexión USB; que posee cuatro hilos:

- Dos de alimentación, correspondiente a los 5 V que se obtienen del PC y uno que se conecta a tierra.
- Dos de datos, denominados D+ y D-, que por el protocolo USB envían los datos de forma que se garantice su fiabilidad.

Al ofrecer el puerto USB 5 V, se conectará también a las patillas de alimentación del PIC, la denominada Vcc y Vpp, mientras que las patillas denominadas Vss y VUSB se conectarán al terminal GND (*ground*) del puerto. De esta forma, se ahorra el añadir fuentes de alimentación adicionales (baterías, etc…) y se tendrá el circuito totalmente alimentado.

Los sensores deberán ser conectados a las entradas marcadas como ANX, siendo X un número que variará entre el 0 y la 12. Si son necesarios más de 13 sensores, se deberán multiplexar las entradas, pero a efectos de patillaje, permanecerá igual.

En la imagen 27, se puede observar la disposición final del circuito, siendo la parte recuadrada en rojo la que habrá que replicar tantas veces como sensores tengamos. Como se puede apreciar, este circuito se conecta a la entrada AN0, el resto de circuitos se deberán ir colocando sucesivamente en AN1, AN2… Tanto el cable de Vcc (5 V) como el de tierra serán comunes a todos los sensores.

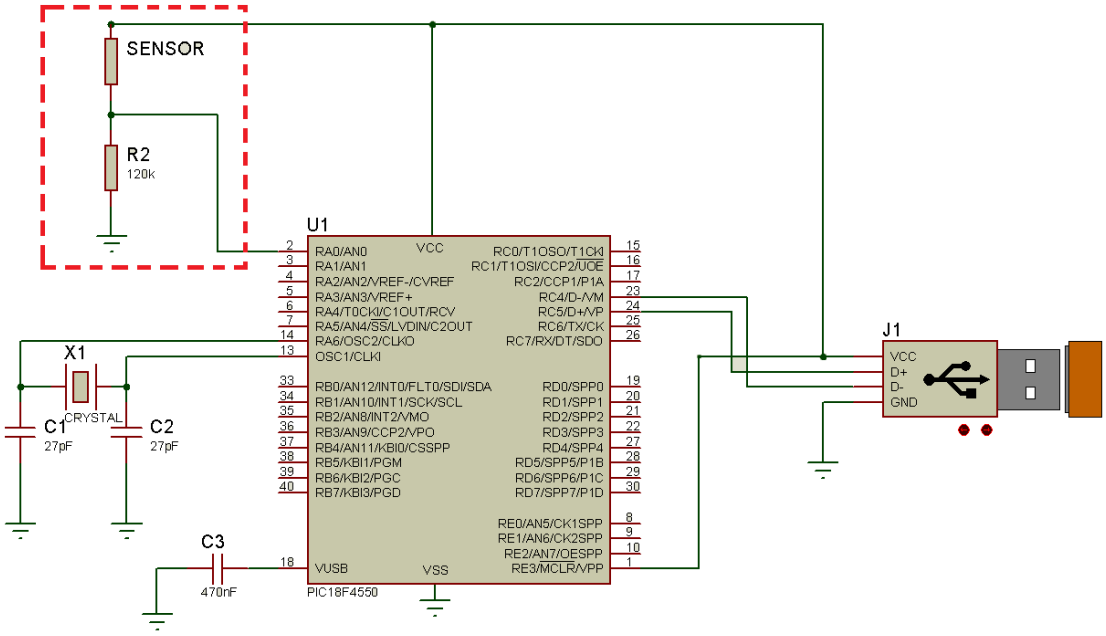


Figura 27: circuito final con un solo sensor montado.

Conectando los componentes como se ha indicado, el circuito ya estaría listo para funcionar. Será necesario por tanto hacer una rutina de control que tome los datos analógicos de los sensores.

6.2.2. Parte de software:

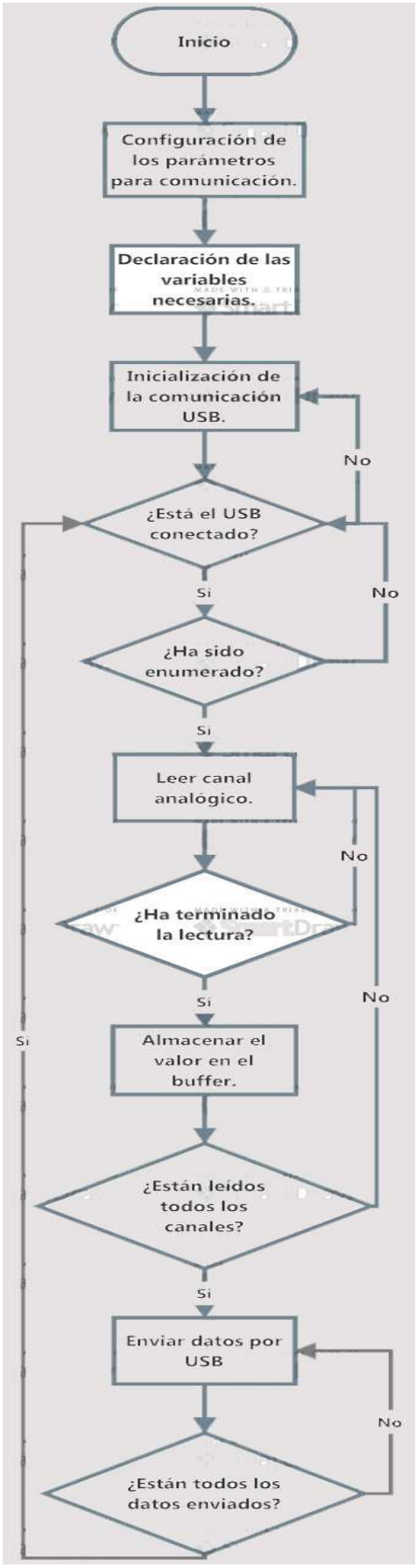
Antes de iniciar la programación, se elabora un diagrama de flujo como el que se ve en la imagen 28. En él, se observan los pasos a seguir para que el sistema cumpla con la función requerida. Como se puede apreciar, existen operaciones de realizar tareas (rectángulos) y operaciones de decisión (rombos). Mediante éstas, es posible controlar todo el programa que llevará el microcontrolador en su memoria y que se encargará de obtener los datos de los sensores, transformarlos en digitales, y enviarlos al computador.

Ahora sí, se debe prestar atención a la programación en sí del dispositivo. La programación consistirá en transformar lo visto en el diagrama de flujo en sentencias y variables que entienda el PIC. Está la posibilidad de usar el lenguaje ensamblador, aunque se elige usar un lenguaje de alto nivel que facilite en gran medida la tarea. Para ello se usa el compilador "PIC C compiler". Este compilador, de la empresa CCS (*Custom Computer Services, Inc.*) permite programar en lenguaje C y luego extraer un archivo *.hex con el que se podrá programar la memoria del microcontrolador. En definitiva, es el archivo hex lo que el microcontrolador entenderá y trabajará con él.

Es necesario ampliar el uso del lenguaje C con algunas librerías que sean capaces de realizar las tareas convenientes a este propósito. Para ello, se deberá incluir en el proyecto, al menos, las siguientes librerías:

- 18F4550.h: esta librería es la que indica las funciones particulares de cada modelo de microcontrolador, así como la distribución del patillaje y demás. Es necesaria

para que se vincule el hardware creado por el fabricante con el software desarrollado por nosotros.



- USB_cdc.h: esta librería es fundamental en el proyecto. Es la que posibilita la comunicación que establece el microcontrolador con el puerto USB. CDC hace referencia al método de envío de datos, puesto que el desarrollador del compilador nos da varias opciones de transmisión de datos. CDC hace referencia a *Communicactions Device Class,* que es una clase para comunicaciones USB que simula un puerto serie tradicional.

Una vez que tenemos las librerías, los pasos que hay que seguir son:

Los *fuses* (fusible): cada uno de estos hace referencia a un valor configurable del microcontrolador. La configuración de éstos será:

- HSPLL: uso de un cristal de más de 4MHz (el cristal seleccionado es de 20 MHz).
- NOWDT: deshabilita el reset por *watchdog.*
- NOPROTECT: No protege el código de ser leído. En un proyecto comercial, puede interesar variar esto, pero en uno académico, es mejor permitir el acceso para cambios futuros.
- NOLVP: deshabilita la programación a bajo voltaje.
- NODEBUG:
- USBDIV: para el módulo USB se tomará el valor del PLL dividido por dos.
- PLL5: la frecuencia a la entrada del PLL se dividirá por 5.
- CPUDIV1: la frecuencia para el reloj del procesador será de 48 MHz (se divide una vez por 2 los 96 MHz iniciales).
- VREGEN: Habilita un regulador necesario para el módulo USB.

El PLL (*phase – locked loop*) es un dispositivo electrónico incorporado en el PIC utilizado para obtener una frecuencia muy estable para el funcionamiento del mismo. En la hoja de características del circuito, se especifica que a la entrada del PLL se necesita una frecuencia de 4MHz, por lo que el cristal de 20 MHz puesto en el montaje hardware deberá ser dividido por un factor de 5, que es lo que se consigue con el fusible PLL5. A la salida del PLL existirán siempre 96 MHz, que deberán ser divididos por los fusibles USBDIV para la frecuencia del USB y CPUDIV1 para el procesador.

También es parte fundamental del proyecto el nombrar los descriptores USB. Los descriptores son los denominados PID y VID del periférico (Vendor ID y Product ID). Éstos son números que garantizan que nuestro microcontrolador, que se comporta como un periférico, no entra en conflicto con ningún otro elemento USB del ordenador. Además, le indica al ordenador dónde debe buscar el driver para el correcto funcionamiento del dispositivo.

Si se tratase de un producto comercial, deberíamos contactar con usb.org, creadores de la especificación USB, y adquirir un VID de por vida, de forma que garantice que nuestro producto puede venderse y que funcionará en todo ordenador.

En este proyecto, simplemente necesitamos un VID y un PID que no esté ya en uso en nuestro PC, puesto que no hay intención de venderlo. Usaremos los siguientes:

```
0x04D8,     //vendor id (0x04D8 is Microchip)

0x000A,     //product id
```

Estos valores deben ser modificados en el encabezado de nuestro programa, en un apartado reservado para ello.

Después se escribiría el grueso del programa, que tendría dos partes bien diferenciadas:

- Por una parte, definimos las variables, y algunas opciones más del microcontrolador, así como iniciamos la comunicación USB.
- Por otra parte, creamos un bucle infinito que, de forma continua, está recibiendo los datos del convertidor y los está enviando al ordenador, los datos de salida del convertidor analógico digital que serán los tomados de los sensores.

En cuanto a los datos; la rutina hace un barrido por cada uno de los canales del convertidor analógico digital, toma el dato y lo almacena en una matriz. Este dato será un valor de 4 dígitos comprendido entre 0 y 1024. Es necesario obligar al PIC enviar el dato con los ceros adicionales hasta completar los cuatro dígitos para posibilitar la mejor comprensión del programa del PC. Esto se hace dando este formato al tipo de dato a recoger:

```
"%04.0f "
```

Donde % significa que es una variable; 04 significa rellenar de 0 hasta 4 cifras; .0 significa que no habrá cifras después de la coma y f es el tipo de dato (*float*). De esta forma, siempre se rellenarán los datos hasta tener 4 cifras (por ejemplo, el 4 pasaría a ser 0004).

6.3. Instalación del driver.

Antes que programar la interfaz en el PC, se debe comprobar que es posible lograr una comunicación entre él y el PIC. Para ello, una vez montado el circuito mínimo visto en la imagen 26, se conecta a un puerto USB libre y se observa lo que pasa.

El PIC se comportará como un elemento hardware nuevo, por lo que el proceso será el habitual, será necesario instalar un driver. Este driver lo proporciona Microchip, mediante la utilidad *Microchip Libraries for Applications*, descargable en la página web de Microchip[5].

Esta utilidad instala en el PC varios controladores útiles para productos de Microchip, interesando solo al proyecto los correspondientes a la comunicación USB, concretamente, mediante el tipo CDC. Al finalizar la instalación de este producto, se debe comprobar qué VID y PID aparece en el driver correspondiente a la comunicación USB CDC. Como ya se ha comentado en el apartado anterior, es crucial que el VID y el PID que se ha programado en el circuito coincida con los que proporciona Microchip mediante su utilidad.

Si no aparecen los valores indicados en el apartado anterior, el documento se puede editar simplemente con el Bloc de notas de Windows. Una vez hecho esto, al conectar el sistema al PC, aparecería un error, que el hardware no es reconocido por el PC. El proceso a seguir sería ir al Administrador de dispositivos, hacer *click* sobre el componente que aparecerá

[5] http://www.microchip.com/

con un símbolo de exclamación, indicar que se quiere instalar un controlador nuevo y acudir a la ruta donde se haya instalado la utilidad de Microchip. Esta ruta, por lo general, será "*C:\Microchip Solutions\USB Tools\USB CDC Serial Demo\inf".*

Completando estos pasos, ya estaría por tanto preparado el PC para trabajar con el PIC y sería necesario continuar con la creación del software para analizar los datos enviados por el microcontrolador.

6.4. Ordenador.

Como ya se sabe, el ordenador o PC será el sistema que finalmente muestre lo adquirido por los sensores al usuario. Se acordó en el apartado de requisitos de diseño que la visualización debería ser de una forma amigable y accesible para todo usuario independientemente de su nivel de cualificación.

De forma conceptual, al pensar en este problema, lo que se pensó fue una interfaz simple, donde aparezca reflejada en la pantalla la disposición de los sensores sobre el dibujo de una planta del pie; y que el valor de éstos fuese variando en función de la presión que soportasen; todo esto en tiempo real. La variación será simbolizada mediante el cambio de color, siendo el rojo el más cargado y el verde el menos cargado.

El lenguaje utilizado será Java, sobre la plataforma NetBeans. Se eligió así puesto que es un lenguaje que, al ser desarrollado por Sun Microsystems, ofrece muchísimas posibilidades gratuitas a todo nivel, por ejemplo, a nivel de entorno de desarrollos, u otras múltiples funcionalidades. Además, gracias a su potente plataforma, es capaz de ser ejecutado en muchísimos dispositivos independientemente del sistema operativo que éstos usen ni la arquitectura de la máquina que lo ejecuta. Esto brinda la posibilidad de programar una única vez el software y funcionar en distintos sistemas operativos.

El programa posee dos módulos fundamentales, y algunos más, pero de menor importancia. En el módulo principal, se crean las rutinas para leer el microcontrolador, se establecen los tiempos de espera del programa, etc.

6.4.1. Módulo principal.

Archivos importados.

Previo a toda la rutina de la programación, es necesario "importar" ciertos archivos. La idea es idéntica a los *include* del lenguaje C; lo que se hace es servirse de las herramientas de programación que proporcionan las grandes empresas para ayudar al desarrollador en su tarea. Se importarán de cuatro sitios fundamentales:

- **Java.awt:** contiene clases para crear interfaces gráficas de usuario y para pintar gráficos e imágenes (como los puntos que equivaldrán a los sensores).
- **Javax.swing:** proporciona un conjunto de componentes "ligeros" que, en la medida de lo posible, funcionan del mismo modo en todas las plataformas.
- **Java.io:** provee herramientas para la entrada y salida del sistema a través de cadenas de datos, serialización y archivos.

- **Gyovynet Driver:** es un elemento de gran importancia en el proyecto, al que se le dedicará el siguiente subapartado.

Con estos archivos que se importen, en algunos casos de forma total (escribiendo import java.io.*; que indica importar todos los elementos) o de forma parcial (concretando lo que se quiere importar); se tendrán los elementos adicionales con los que desarrollar íntegramente el software.

Gyovynet Driver.

Una de las cosas fundamentales era el poder vincular el software que se estaba diseñando con el PIC. Es decir, no sólo conectar el PIC al PC, que es relativamente fácil, si no que ahora la necesidad era conectarlo al software propio. Para lograrlo es necesario añadir componentes extra al programa. Es en esta búsqueda cuando se dio con el denominado GiovynetDriver_2.0. Haciendo una traducción casi literal de lo que aparece en la página web del desarrollador, dice lo siguiente:

> *Giovynet driver es un espacio de trabajo que le permite usar Java para crear aplicaciones que permite comunicar circuitos externos con un PC. Giovynet driver presenta un completo set de instrucciones para manejar los puertos GIV8DAQ y GIV4R. También soporta enviar y recibir caracteres ASCII a través del puerto RS-232.*

Lo que interesa es esta última parte, pues los citados puertos GIV8DAQ y GIV4R son puertos específicos de la empresa que desarrolla el driver; mientras que el RS-232 es el más conocido como puerto serie. Aunque se usa comunicación USB, al ser ésta muy complicada, el dispositivo emulará ser un puerto serie como se comentó en el apartado de programación del PIC.

El uso de este driver es gratuito para uso personal; no así para uso comercial. No obstante, para el objeto del proyecto, es 100 % funcional. Se suministra en versiones de 32 y 64 bits, por lo que según el hardware del ordenador donde se instale el software será necesario escoger uno u otro.

Las funciones necesarias para usar este driver en el proyecto son:

- Un objeto tipo SerialPort. Este objeto hace referencia al puerto de serie de forma global. Se usará para enumerar los dispositivos serie conectados al ordenador en este caso concreto, aunque tiene más utilidades como pueden ser leer caracteres de los puertos o establecer el tiempo de espera para la lectura de un puerto.

- Un objeto tipo Parameters. Este objeto se encarga de almacenar los diferentes parámetros para configurar el objeto tipo Com.
    - parameters.setPort(). Retoma la lista hecha por el objeto SerialPort y elige el primero de los puertos libres.
    - parameters.setBaudRate(). Establece la tasa de baudios por segundo que usará el puerto.

- parameters.setByteSize(). Establece el tamaño del byte que será enviado a través del PIC.
- parameters.setParity(). Establece la paridad del diseño. La paridad es un método para asegurar la fiabilidad de la comunicación.

- Un objeto tipo Com. Éste es el encargado de pasar los parámetros al puerto escogido por el método setPort visto anteriormente y de realizar el grueso de la recepción mediante el siguiente método.
  - com.receiveToString(). Este método recoge lo que está en el buffer de salida del microcontrolador y lo transforma en un string. Es la función que se quería obtener desde el primer momento, pero los pasos anteriores fueron necesarios para configurarla de forma adecuada.

Bucle principal.

Previo a esta orden fundamental, com.receiveToString(), existe una sentencia tipo *switch-case.* Esta sentencia sirve para diferenciar entre varios casos:

- Primero, el caso en que el programa se ejecute para leer los datos directamente de la plantilla (visionado en tiempo real y guardado de los mismos en un archivo).
- Segundo, el caso en que el usuario decida detener la lectura de datos.
- Tercero, el caso en que el usuario decida leer los datos almacenados en un fichero.

El caso fundamental y que más interesa es el primero, que es donde se desarrolla la idea principal del proyecto; pues los otros dos casos, fueron ideas que expanden y complementan al proyecto, pero que en realidad no eran uno de los objetivos deseados al inicio.

Uno de los problemas que se tuvieron a la hora de usar el comando receiveToString() es que la sincronización no siempre era la adecuada. Como se vio en el apartado 6.2.2, para facilitar en un principio la lectura de estos datos, se enviaban en forma similar a una matriz. Pero al ser leída por el comando receiveToString(), esta matriz no siempre empezaba en lo que sería el elemento 1, 1; y tampoco era siempre el mismo defecto, por lo que no se podía corregir añadiendo un *offset* o índice fijo. Se desconoce a qué podría ser debido, si a datos falsos, a fallos en la sincronía de ambos dispositivos… Pero estaba claro que no garantizaba la fiabilidad de los datos y que era necesario buscar una solución.

Ésta fue simple, pues se eligió iniciar la transmisión con una cadena de caracteres que no pueda repetirse en ningún otro espacio de la sincronización. Después, desde el programa, simplemente crear una rutina que busque dentro de los caracteres recibidos esta cadena de inicio, y mover el inicio de la transmisión hasta ese punto.

Para asegurar que se recogerá siempre al menos una transmisión completa, la cadena de caracteres recibida será el doble de lo que sería una cadena simple; en caso contrario, se corre el riesgo de que se pierdan valores, por ejemplo, si la transmisión empieza inmediatamente después de la cadena de inicio de valores, se tendrían prácticamente todos los caracteres de la primera cadena, y se necesitaría otra cadena igual de larga para volver a

almacenar todos estos valores. Por tanto, se elige que la recepción sea una cadena el doble de larga para evitar este problema.

En cuanto a la rutina creada, se puede observar el código en la siguiente ventana:

```
        do{             //Bucle para buscar dentro de la cadena los 4 nueves.
                        j++;
                        if(data.datoRecibido.charAt(j)=='9')
                            if(data.datoRecibido.charAt(j+1)=='9')
                                if(data.datoRecibido.charAt(j+2)=='9')
                                    if(data.datoRecibido.charAt(j+3)=='9'){
                                        nueves=1;
                                        indice=j;
                                        j=170;
                                    }
        } while (j<170);
```

Se busca una cadena de cuatro nueves, debido a que es un valor que nunca podrá alcanzar ninguno de los sensores. Si recibe 4 valores consecutivos, entonces sale del bucle igualando la variable j a 170, que es el número de iteraciones que deberá hacer.

Una vez encontrada la cadena de caracteres, se desplaza el índice hasta ese valor; o lo que es lo mismo, se relativiza todo en función del índice.

En el siguiente paso, lo que es necesario hacer es transportar el fragmento de la cadena que se corresponde con el dato del sensor a la clase data. Para ello, son necesarias dos operaciones:

- Por un lado, trabajar sólo con los caracteres que ocuparán el número, excluyendo espacios, caracteres de salto de línea, etc. Esto se consigue mediante el comando subString(), que permite tomar un fragmento de la cadena entre dos valores que se le otorguen.
- Por otro lado, transformar este fragmento de cadena a un valor numérico (integer). Se puede conseguir con el método Integer.parseInt(), que transforma un número escrito en caracteres en una variable de tipo *integer* (entero).

Así, una línea de código para transformar esto quedará con el siguiente formato:

```
data.valorNum[0]=Integer.parseInt(data.datoRecibido.substring(indice+6,indice+10));
```

Siendo indice el valor inicial que se obtuvo en la rutina creada para dotar al sistema de sincronía. El valor añadido a indice es lo que indicará si estamos en uno u otro sensor; siempre habrá un valor de 4 entre ambos "`(indice+6,indice+10)`", puesto que el microcontrolador ofrece un dato numérico de 4 caracteres correspondiente al valor del sensor, como se estableció en el apartado 6.2.2.

También se deberá modificar el color de cada una de las formas que representan los sensores. Este color, como se habló en el apartado 6.4, será en relación a la presión detectada por el sensor.

Para ello, hay que anticipar lo que se hará en el apartado 6.4.2, que se creó un vector de colores adecuados al proyecto. Este vector está almacenado en la variable Colores[]; vector que almacena 32 valores y que son todos consecutivos.

A la hora de cambiar el color de la forma en función de la presión, se establece una rutina de comparación como se ve a continuación. En ésta, se crea una variable para comenzar la iteración, ite; y se incrementa su valor. Se establece una relación entre el valor incremental y esta variable de iteración (ite*ganancia) de forma que se crean 31 intervalos entre los que la forma, definida como jPanelX, donde X es un número que variará según el sensor, adquirirá un color del vector Colores[] en función del valor numérico adquirido, data.valorNum[k].

```
        //***************** Sensor 4**********************
             k=3;
             for(int ite=0;ite<31;ite++)
            {
                if(data.valorNum[k]>ite*ganancia && data.valorNum[k]<(ite+1)*ganancia)
                    ventana.jPanel2.setBackground(ventana.Colores[ite]);
                if(data.valorNum[k]>800)
                    ventana.jPanel2.setBackground(ventana.Colores[31]);
            }
```

La variable ganancia sirve para modificar la ganancia en función de si se representan bien los cambios o, por el contrario, son casi inapreciables.

Esta rutina será necesario replicarla tantas veces como sensores hay, cambiando en cada caso tan solo el jPanel. En el siguiente apartado se hablarán de los distintos componentes java, incluyendo el jPanel.

6.4.2. Módulo ventana principal.

Archivos importados.

En este módulo también habrá *imports*, que aunque tienen diferente propósito, son comunes a los anteriores. Así, se vuelven a importar partes pertenecientes a java.awt, java.io, y javax.swing.

Creación de la ventana.

Para la creación de la ventana se usó el modo diseño, donde simplemente se arrastraron los diferentes elementos sobre la ventana principal del programa. Esta ventana es un JFrame grande, componente java importado de la clase swing. Es simplemente un marco con un borde superior que contiene las funciones básicas de un programa (cerrar, minimizar…) y el título del programa.

Dentro del código, se crea la clase *main* que extiende a JFrame, es decir, es una clase que hereda todas las propiedades del elemento, pero además se les puede añadir propiedades específicas para el proyecto. Esta será la clase donde se encuentre todo el código que desarrolle el software.

Se creará una matriz con los diferentes colores que se corresponderán con diferentes valores de los sensores. La idea inicial era hacer un barrido por los colores, con incrementos

fijos que fuesen desde el color verde (poca o ninguna presión) hasta el rojo (grandes sobrepresiones); pasando por el amarillo que sería una presión media. Esta idea no se pudo llevar a cabo puesto que el sistema de representación en color sigue la idea RGB (rojo – verde - azul); representando cada valor con dos dígitos en hexadecimal. Para que se entienda bien, se diseñaron las siguientes imágenes:

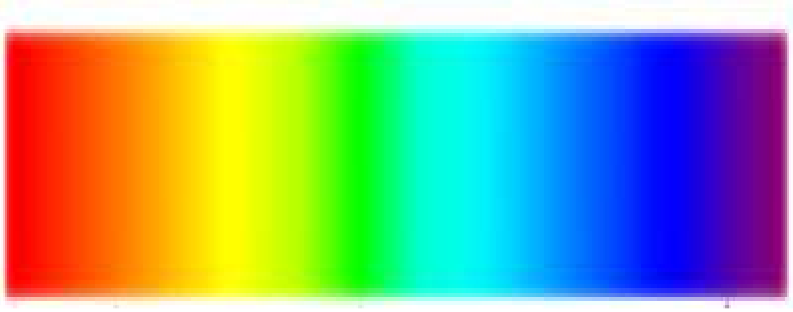


Figura 29: representación de los colores según la naturaleza (como en un arcoíris).

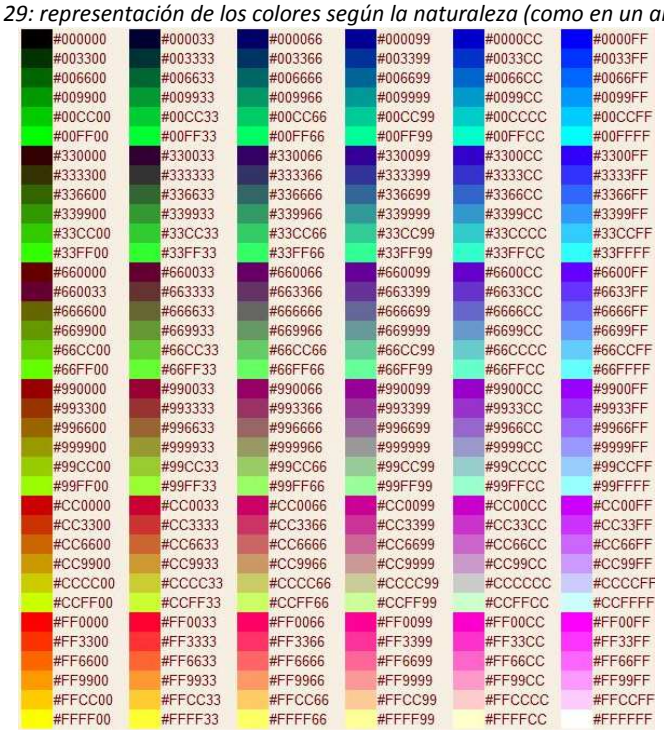


Figura 30: representación de los colores según sistema RGB.

En ellas se aprecian las dos formas de representar los colores mencionadas: en la primera, sí se podría realizar un barrido desde un número mínimo a un máximo desde el rojo al verde; pero la forma de representar esto en java es como se ve en la segunda imagen; el incremento no pasa del rojo al verde; si no que por ejemplo, va de un tono de verde a un tono de azul, y si se sigue incrementando; volvemos a un tono, más claro, de verde, y se vuelve a ir hasta un tono más claro de azul.

Por tanto, fue necesario crear alguna estrategia para evitar este problema. Se creó una matriz de colores de 32, y a cada uno de forma “manual” se le fueron introduciendo los cambios con respecto al anterior. Esto se puede ver en el siguiente fragmento de código:

```
        //Declaración de los colores
                Colores[0]=new java.awt.Color(0x00,0xFF,0x00); //verde
                Colores[1]=new java.awt.Color(0x1A,0xFF,0x00);
                Colores[2]=new java.awt.Color(0x2B,0xFF,0x00);
                Colores[3]=new java.awt.Color(0x3C,0xFF,0x00);
```

```
            Colores[4]=new java.awt.Color(0x4D,0xFF,0x00);
            Colores[5]=new java.awt.Color(0x5E,0xFF,0x00);
            Colores[6]=new java.awt.Color(0x6F,0xFF,0x00);
            Colores[7]=new java.awt.Color(0x7C,0xFF,0x00);
            Colores[8]=new java.awt.Color(0x8D,0xFF,0x00);
            Colores[9]=new java.awt.Color(0x9E,0xFF,0x00);
            Colores[10]=new java.awt.Color(0xAE,0xFF,0x00);
            Colores[11]=new java.awt.Color(0xBF,0xFF,0x00);
            Colores[12]=new java.awt.Color(0xCF,0xFF,0x00);
            Colores[13]=new java.awt.Color(0xDF,0xFF,0x00);
            Colores[14]=new java.awt.Color(0xEF,0xFF,0x00);
            Colores[15]=new java.awt.Color(0xFF,0xFF,0x00); //amarillo
            Colores[16]=new java.awt.Color(0xFF,0xFE,0x00);
            Colores[17]=new java.awt.Color(0xFF,0xED,0x00);
            Colores[18]=new java.awt.Color(0xFF,0xDB,0x00);
            Colores[19]=new java.awt.Color(0xFF,0xC8,0x00);
            Colores[20]=new java.awt.Color(0xFF,0xB6,0x00);
            Colores[21]=new java.awt.Color(0xFF,0xA4,0x00);
            Colores[22]=new java.awt.Color(0xFF,0x92,0x00);
            Colores[23]=new java.awt.Color(0xFF,0x80,0x00);
            Colores[24]=new java.awt.Color(0xFF,0x7F,0x00);
            Colores[25]=new java.awt.Color(0xFF,0x6E,0x00);
            Colores[26]=new java.awt.Color(0xFF,0x5D,0x00);
            Colores[27]=new java.awt.Color(0xFF,0x4B,0x00);
            Colores[28]=new java.awt.Color(0xFF,0x38,0x00);
            Colores[29]=new java.awt.Color(0xFF,0x26,0x00);
            Colores[30]=new java.awt.Color(0xFF,0x14,0x00);
            Colores[31]=new java.awt.Color(0xFF,0x00,0x00); //rojo
```

Una vez hecho esto, cuando en el módulo principal se escriba el código correspondiente al cambio de colores, se llamarán a cada uno de los elementos de esta matriz; en función del valor del sensor.

Tras esto, se añaden desde la vista de diseño los diferentes elementos al JFrame principal. Existen elementos de los siguientes tipos:

- jPanel: es un componente plano que puede ser rellenado con una imagen o con un color. Se usará para contener la imagen de la planta de pie y para mostrar el valor de los sensores mediante un color de los definidos en la matriz Colores[].
- jButton: es un componente sobre el que se puede hacer click con el ratón y añadir código de comportamiento. Estos botones pueden abrir otras ventanas, cambiar el valor de una variable u otras diversas funciones.
- jLabel: es un componente de texto estático, es decir, que el usuario no puede cambiar el contenido; aunque sí se puede cambiar presionando algún botón y otras acciones similares.
- jTextField: es un componente de entrada de texto, donde el usuario puede introducir un texto y el programa puede reconocerlo y actuar en consecuencia.
- jComboBox: es un componente desplegable que muestra distintas opciones entre una lista prefijada.

En la siguiente imagen se pueden ver estos elementos señalados mediante líneas discontinuas de puntos para facilitar un poco la explicación arriba dada.

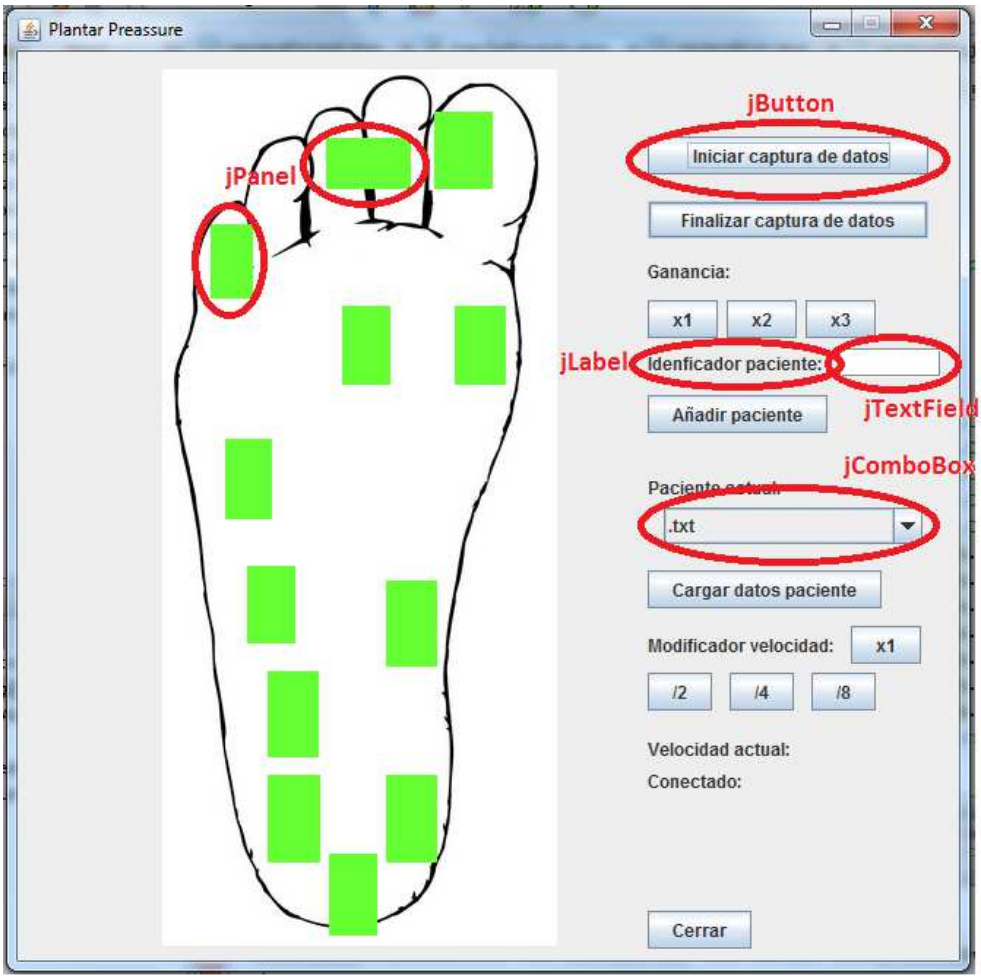


Figura 31: captura para explicación de los componentes java que se utilizaron.

Cada uno de estos botones tiene un código asociado que se ejecutará cuando el usuario pulse sobre ellos. En el apartado 8 se explicará el funcionamiento del programa; y por tanto, la función de cada botón. Este apartado explica cómo se efectuó la programación, por lo que solo se destacan los aspectos más importantes de la misma.

6.4.3. Módulos adicionales.

Además de los dos módulos principales vistos, se crearon dos módulos más, de menor complejidad e importancia. No obstante, se considera necesario hablar de ambos para tenerlos presentes para que el lector lo tenga presente y para futuras ampliaciones del proyecto.

Módulo ventana error/advertencia.

Este módulo consiste en sólo un jFrame con un jButton en el centro y un jLabel sobre él. Su objetivo es mostrar mensajes de advertencia o de error que puedan ser lanzados por el programa por incidencias o mensajes que se quieran indicar al usuario. Todos los elementos se declararon como *public* con el objetivo de que desde cualquier parte del programa se pueda modificar el texto a mostrar, así como el título o el texto que aparezca sobre el jButton.

Tiene la siguiente apariencia:

Figura 32: ejemplo mensaje de error con el componente VentanaError.

Para provocar su aparición, es tan sencillo como introducir el siguiente código cuando se produzca el evento; por ejemplo dentro de una condición *if*:

```
VentanaError error = new VentanaError();
           error.setTitle("ERROR");
           error.setVisible(true);
```

De esta forma, se hace visible este componente nuevo. De forma adicional, se marcó la propiedad "*alwaysOnTop",* que hará que el aviso se quede siempre en primer plano del programa para evitar que el usuario lo pase por alto.

Módulo ventana error/advertencia.

Este módulo es muy similar que el anterior, igualmente consiste en sólo un jFrame, un jLabel centrado y dos jButton en lugar de uno. Esto es debido a que su objetivo es mostrar preguntas que se deban hacer al usuario, dando la oportunidad de aceptar o denegar. Del mismo modo, los elementos se declararon como *public* con el objetivo de que desde cualquier parte del programa se pueda modificar el texto a mostrar, así como el título o el texto de la pregunta.

Tiene la siguiente apariencia:

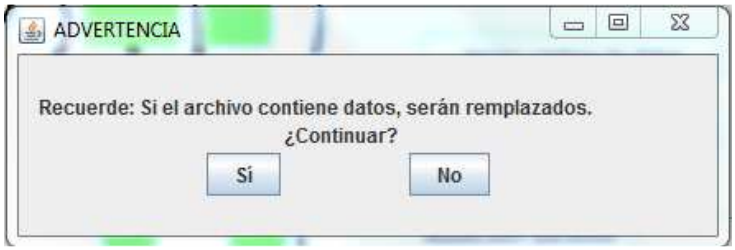


Figura 33: ejemplo mensaje con el componente VentanaSiNo.

Para provocar su aparición, la idea es idéntica al caso anterior, por lo que no se añadirá aquí el fragmento de código. Igualmente, se marcó la propiedad "*alwaysOnTop",* para que aparezca sobre el resto de elementos del programa.

7. Descripción del hardware.

En cuanto al hardware del sistema, constará de dos partes fundamentales:

- La plantilla con los sensores repartidos en su superficie, incluyendo el cableado.
- El circuito encargado de la captación, conversión y transmisión de los datos, como se aprecia en la imagen 34.

La plantilla poca explicación necesita, no siendo así con el circuito. En él, se puede apreciar el conector, a la izquierda, USB de tipo A; para el cual será necesario un cable USB macho – macho. Este cable se conectará directamente al PC y al circuito, sin nada adicional. Al conectarlo, en el PC será necesario instalar el driver (apartado 6.3) y ya estará listo para funcionar.

Posee un indicador LED de testigo que comprueba que el circuito está correctamente alimentado. De esta forma, si surge algún problema es una forma rápida de ver si es problema es del PC, del circuito integrado, del cable… (Por ejemplo, si es problema del PC, que no da corriente suficiente, el LED se iluminaría más tenue o no se iluminaría, mientras si el problema es un condensador que falle, el LED se iluminaría, pero el circuito no daría conexión al PC).

Tanto el cristal y los condensadores que se ven en la imagen ya fueron explicados en el apartado 6.2.1.

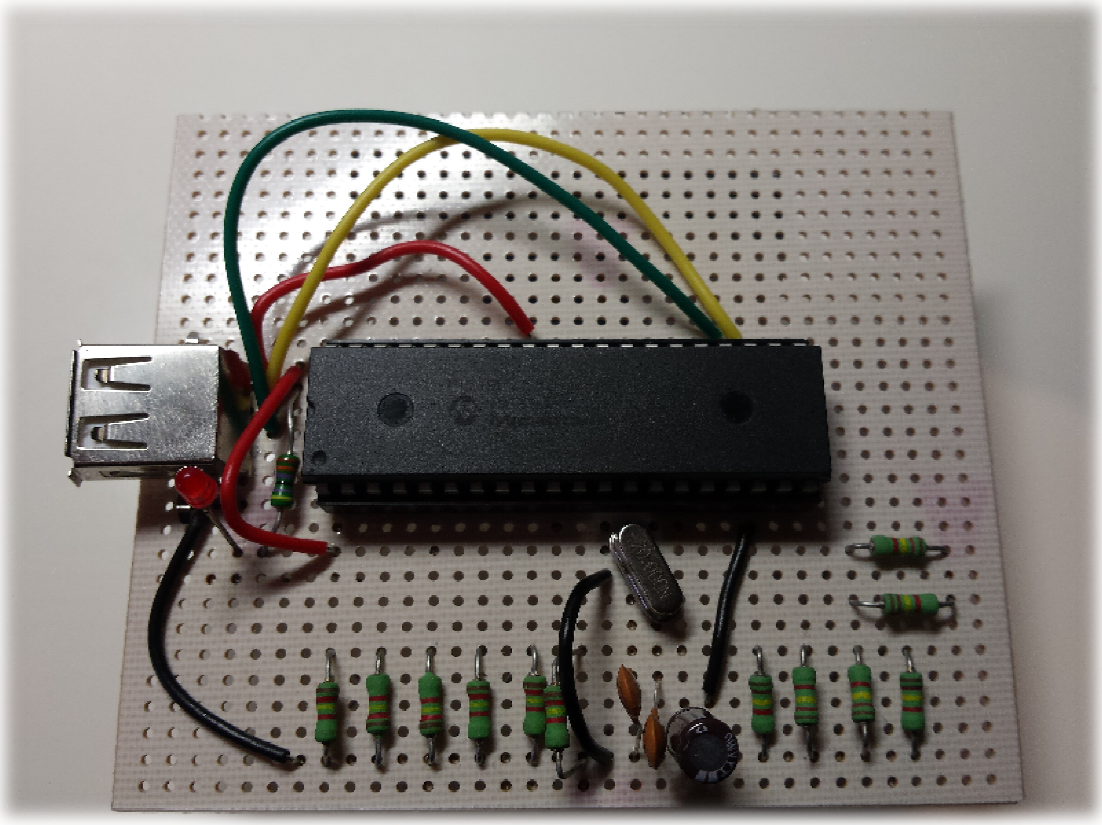


Figura 34: imagen del circuito final soldado sin los sensores y la comunicación con las entradas.

Para facilitar la comprensión del circuito, se respetó un código de colores en el cableado. Así, los cables amarillo y verde se corresponden con los cables de datos del USB por los que se transmite la información; los rojos están relacionados con el valor de alimentación positiva (5 V); y los negros con las conexiones a masa. Aunque no se aprecia en la imagen, los cables de información de los sensores serán morados.

Posteriormente, se conectaron los cables de los sensores al circuito, y éste fue introducido en una caja para protegerlo. Tanto a la plantilla como al circuito, fueron añadidas tiras de *velcro* para posibilitar su adhesión tanto al pie como a la pierna del usuario.

8. Descripción del software.

Este apartado será dedicado a una introducción al usuario del programa desarrollado junto con la plantilla para su fácil interpretación. Una vez instalado el driver, como lo visto en el apartado 6.3, ya estaría todo disponible para iniciar el funcionamiento de este programa. En la siguiente imagen, se puede ver lo primero que el usuario se encuentra al ejecutarlo:

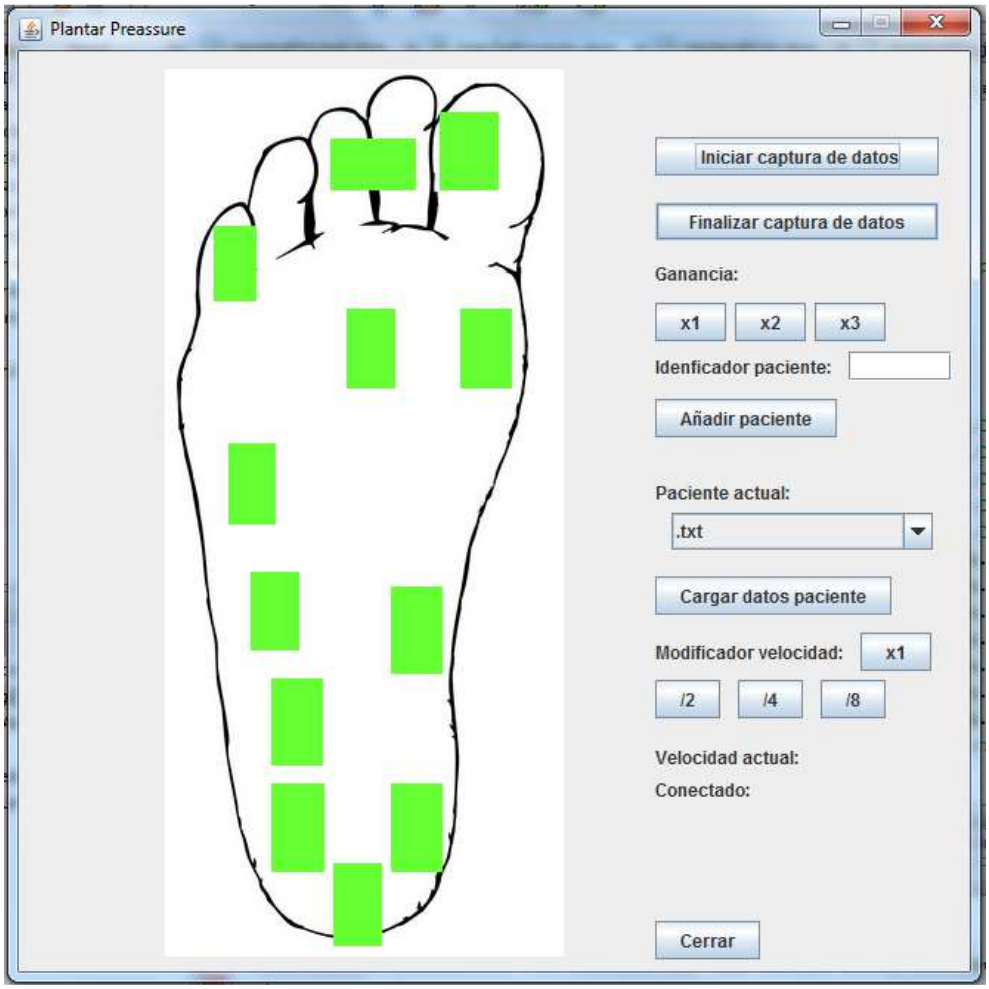


Figura 35: vista general del software.

Es una aplicación de una única ventana donde todas las funciones se desarrollan haciendo *click* en los diferentes botones que aparecen en la parte derecha. En el programa existen tres funciones principales, como se dijo en el apartado 6.4.1. Los tres botones que llevan a estas funciones son los más grandes, que aparecen remarcados en la imagen 37:

- Iniciar captura de datos: haciendo *click* en este botón, se inicia la rutina que toma los datos del microcontrolador y los muestra sobre el dibujo del pie que aparece en la parte izquierda del programa mediante el cambio de color de los recuadros (verdes en la imagen al considerarse que no tienen ninguna presión). Además, existen 3 botones para modificar la ganancia del sistema; es decir, que según el peso del usuario y si los cambios son apreciables o no, se podrá facilitar el visionado mediante el uso de estos tres botones.

- Finalizar captura de datos: este botón detiene la rutina iniciada por el anterior botón, y también la rutina que se inicia cuando el usuario haga *click* en el tercer botón. Además, restaura los recuadros al valor original.
- Cargar datos pacientes: este botón sirve para cargar en el software los datos del paciente que aparezca seleccionado bajo el título "Paciente actual". Además, mediante los botones x1, /2, /4 y /8 se puede modificar la velocidad de reproducción de datos, para poder observar más lentamente el progreso de la forma de andar del paciente.

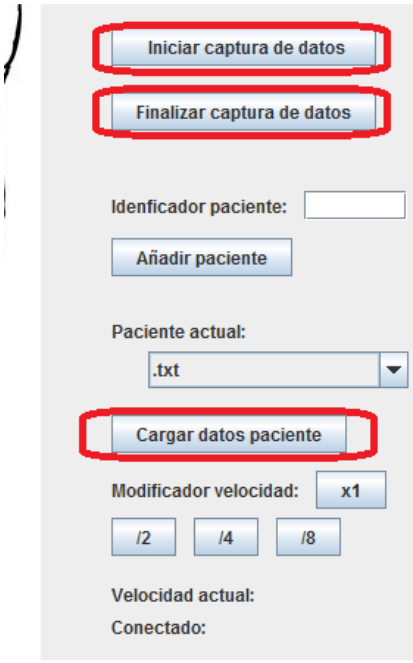


Figura 36: los tres botones esenciales.

Además, existen otros botones y elementos que no controlan el flujo principal del programa, pero que aún así, son necesarios para el correcto funcionamiento del mismo. Son los marcados de color azul en la siguiente imagen:

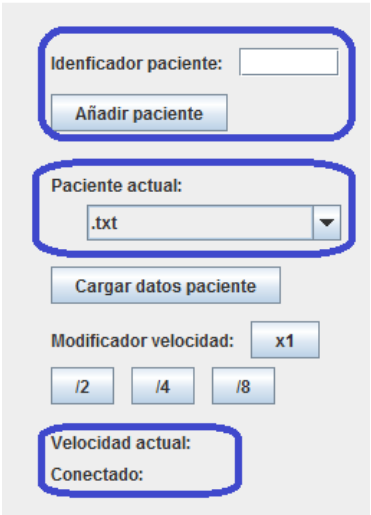


Figura 37: otras funciones adicionales.

- El primer recuadro azul muestra los elementos que permiten añadir pacientes. Junto al texto "Identificador paciente:", existe un componente de entrada de texto, en el que se puede escribir un número identificador, un nombre… Tras introducir este identificador, el usuario debe pulsar el botón "Añadir paciente" para que se cree un archivo con extensión .txt donde se almacenarán los datos recogidos por el programa. Estos datos se almacenarán en la ruta "*C:\Pacientes*" por defecto.
- El siguiente es el selector de pacientes. El programa al iniciarse busca, de forma automática, en la carpeta "*C:\Pacientes*" e introduce en el índice todos los archivos .txt que encuentra.
- El último recuadro marcado en azul son dos etiquetas que, junto a ellas, aparecerá información útil sobre el estado y ejecución del programa. En la primera de ellas, "Velocidad actual", aparece la velocidad a la que el programa cargará los datos de un paciente (x1, /2, /4 o /8); mientras que en la segunda, aparecerá si está conectada o no la plantilla.

Algunas recomendaciones que se dan al usuario son:

- Procurar que el usuario esté andando/corriendo antes de pulsar el botón "Iniciar captura de datos" para evitar recoger en los archivos datos "vacíos".
- El identificador de paciente siembre deberá tener al menos 5 caracteres, de lo contrario, el programa devolverá un error y no creará el usuario deseado. Admite espacios, sólo letras, sólo números, pero no admite caracteres especiales (asteriscos, barras oblicuas,…). La letra ñ, por ejemplo, sí la admite.
- En el menú desplegable de paciente actual los pacientes aparecerán siempre ordenados de forma alfabética, excepto los que hayan sido añadidos mientras se ejecuta el programa, que aparecerán al final independientemente del orden alfabético. Si el programa se cierra y se vuelve a abrir, se ordenarán todos, incluso los añadidos recientemente, de forma alfabética.
- El modificador de velocidad por defecto es 1, aunque no aparezca en el correspondiente lugar al iniciarse el programa.
- Entre el botón "Cerrar" y la tradicional X sobre fondo rojo situada en la parte superior derecha del programa no hay ninguna distinción, se puede usar cualquiera de ellas en cualquier momento.

En el siguiente apartado se podrán ver ejemplos de uso e imágenes del funcionamiento real del software.

9. Resultados, imágenes de muestra y capturas de pantalla.

9.1. Sistema final.

En la siguiente imagen se aprecia tanto la plantilla como la disposición final de los sensores, como la caja que contiene el circuito montada sobre un usuario. El cable azul que se visualiza en la parte derecha es el que está conectado al equipo para la toma de medidas. Se puede apreciar también la cinta para correr sobre la cual serán realizadas las siguientes pruebas.

Figura 38: sistema final.

9.2. Pisada estática.

En la imagen, se aprecia una pisada estática con una ganancia de 3x (cuatro veces ampliada):

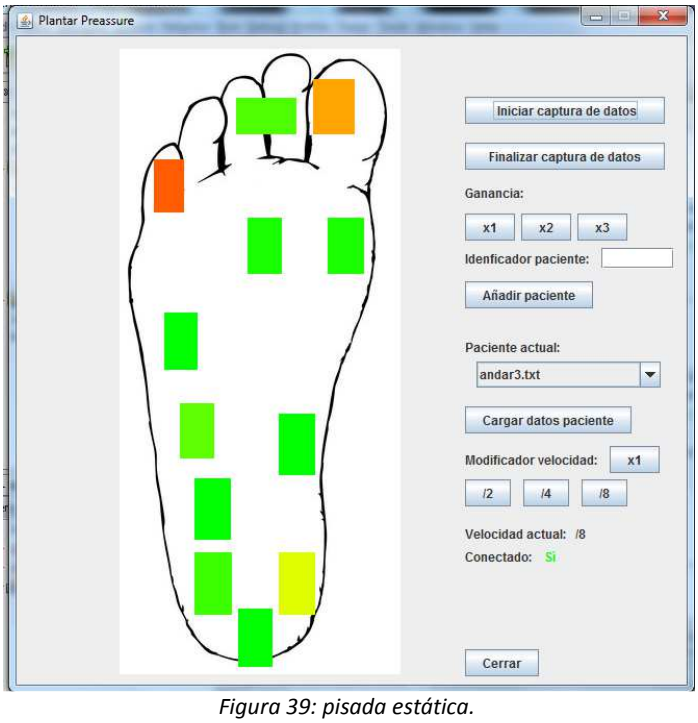


Figura 39: pisada estática.

Este paciente, podría deducirse mucho apoyo en tres puntos fundamentales; en el talón, en el pulgar, y en el extremo izquierdo de las falanges.

9.3. Andando.

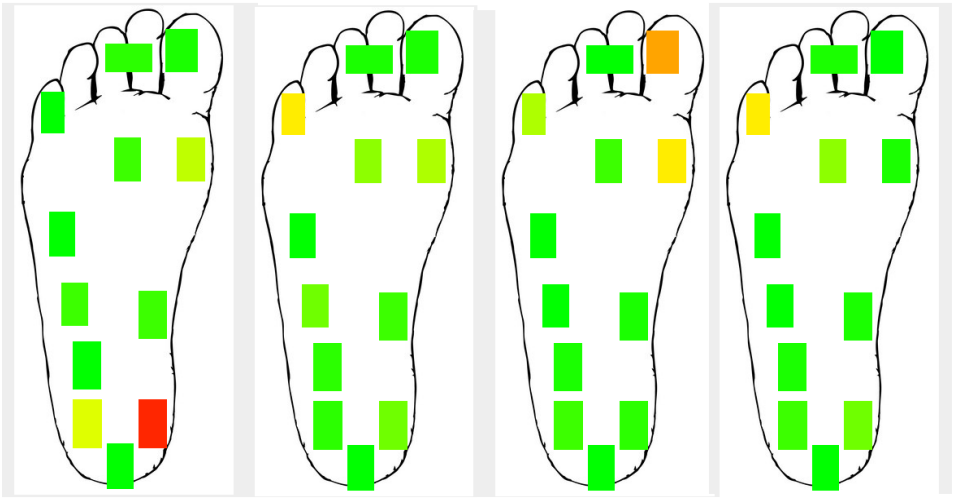


Figura 40: evolución de la pisada andando, con ganancia x3.

En la imagen se puede apreciar el ciclo de marcha de un usuario andando; usuario que se aprecia un incremento en la parte derecha del pie tanto en la fase de aterrizaje (primera por la izquierda) tanto como en la fase de despegue.

9.4. Corriendo.

Figura 41: evolución de la pisada corriendo, con ganancia x3.

Esta sucesión de imágenes contempla un ciclo de marcha de un usuario en carrera. Se aprecia la sucesión de fuerzas de la parte posterior de pie hacia el pulgar.

10. Discusión y conclusiones finales.

Para elaborar unas conclusiones finales adecuadas, lo mejor es recordar los objetivos que se persiguieron inicialmente y evaluar si éstos se cumplieron. Así, se retoma lo visto en el apartado 3 y se tiene:

Primer objetivo: analizar la pisada de una persona mediante un sistema electrónico.

Como tal, el sistema proporciona un sistema que permite tomar medidas de lo que sucede en el pie humano mientras que anda. Este sistema, además, está diseñado para exportar estas medidas en un archivo .txt, al que se puede acceder mediante cualquier otro software que permita la lectura de este archivo.

Se puede concluir que este objetivo está cumplido; puesto que se ha desarrollado desde cero un sistema completo, es decir, que al sistema no es necesario añadir más elementos para su funcionamiento; ya que se han configurado todos y cada uno de los elementos que lo componen.

Lo que sí se echa de menos es haber contrastado los resultados con algún otro de los sistemas existentes (los tratados en el apartado 3.2) al no tener ninguno disponible. No obstante, en el desarrollo de un producto, esta sería la fase inmediatamente posterior, el demostrar su validez comparándolo con otros sistemas, probándolo en diferentes pacientes, etc.

Segundo objetivo: el coste económico del proyecto, y el esfuerzo por reducirlo al máximo.

Para mostrar si este objetivo se ha conseguido, lo ideal es hacer el presupuesto que ha costado el sistema íntegramente.

Elemento	Coste por unidad	Cantidad	Coste total
PIC 18F4550 - Microcontrolador con comunicación USB	10,35 €	1 ud.	10,35 €
Sensor Film Kit - Suministrado para hacer la forma deseada. - Tamaño 3” x 4”.	18,91 €	3” x 4” (para hacer unas 16 – 18 uds.)	18,91 € (gastos de envío incluidos)
Resistencias 120 kΩ	0,10 €	13 ud.	1,30 €
Plantillas talla estándar	1,80 €	1 ud.	1,80 €
Zócalo DIP 40-P	0,31 €	1 ud.	0,31 €
Conector USB hembra tipo A	0,28 €	1 ud.	0,28 €
Cable USB macho-macho	2 €	1 ud.	2 €
Gastos varios (cableado, estaño de soldaduras,…)	5 €	1 ud.	5 €
		Total	39,92 €

Tabla 3: presupuesto estimativo del sistema.

Este presupuesto es tan solo para hacer una idea del coste final; puesto que según el sitio donde se adquieran los elementos, éstos podrán variar su precio. También influye mucho el hecho de los gastos de envío, pues en el caso más general, pedir varias unidades abaratará los precios; así como tampoco es fácil tener en cuenta otros gastos como las soldaduras empleadas; puesto que es difícil determinar cuánto estaño se ha empleado y el coste del mismo. Por ello, se ha añadido un sobre coste de 5 € de este concepto y otros que se puedan englobar.

De todas formas, lo que se aprecia es que es un sistema que efectivamente, no se hace caro, puesto que es un precio que, como ya se comentó, laboratorios caseros, pequeñas ortopedias y similares, sí pueden asumir.

Futuras mejoras.

Lo normal en un proyecto es que mientras se investigan las diferentes soluciones, mientras que se avanza en su ejecución, se encuentran muchas alternativas a cómo se está realizando. Muchas de ellas, en la toma de decisiones, tienen que ser rechazadas por muchísimos motivos; lo que no hace que sean malas opciones; si no que a criterio del diseñador, no tienen cabida en el proyecto.

La auténtica lástima es que lo que obliga a desechar muchas de estas buenas opciones es la falta de tiempo, o de presupuesto, o ambos unidos. Y es por eso por lo que el producto no puede ser mejor. No obstante, para el futuro, es necesario dejar constancia de estas posibles mejoras que harían al producto aún más redondo y más completo.

Tallas de las plantillas.

Uno de las preguntas que cabe hacerse al respecto del proyecto es, ¿para qué talla de pie es este producto? Está claro que cada usuario tendrá una talla de pie diferente, por lo que la posición de los sensores no será la misma para un hombre con un pie de talla 40 que 43; y mucho menos si se miran los extremos.

Las plantillas Bioofoot/IBV solucionan este problema suministrando varias tallas de plantillas y permitiendo conectar una u otra plantilla al circuito en función de la necesidad. Al margen de esta solución, que desde luego es válida, existen otras posibilidades.

Una de ellas sería diseñar un sistema de “sensores móviles” de forma que fuese posible situar los sensores más arriba o más abajo. Algo así como una superficie microperforada y un método de conexión de los sensores para que pudiesen ser insertados en la plantilla a presión.

Otra opción es una que se estudió, sugerida por el tutor, en la que se realizase una matriz de sensores. La idea en este caso sería asegurarse que la matriz fuese del tamaño del pie más grande que se pudiese medir, asegurando así todos los tamaños de pie. Esta idea se aprecia en la entrada de blog de donde se extrajo la siguiente imagen[6]:

[6] “Low-Cost Fabric Tactile Sensing Skin.”

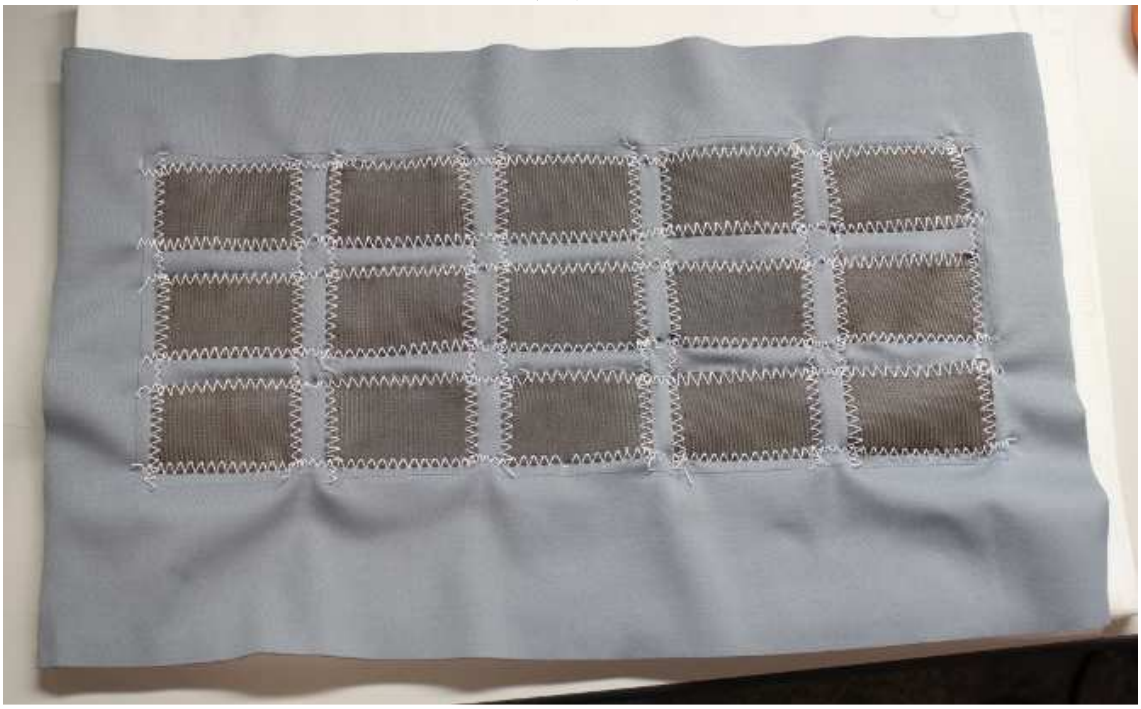


Figura 42: matriz dispuesta para la colocación de sensores sobre ella.

En el blog hablaban de fabricar una "tela táctil" para un brazo robótico con el fin de estudiar si hay contacto con el brazo robótico. Esta idea es perfectamente extrapolable al proyecto, solo que lo que sí está claro es que el número de sensores crecería; por lo que sería necesario o buscar otro PIC, o multiplexar estas entradas o cualquier otra estrategia para aumentar el número de sensores.

Métodos de visualización.

Al hablar de las soluciones existentes, se vieron varias formas de representar los datos de las plantillas. Como se está hablando de opciones, y los datos ya se transmiten al PC, en realidad sería tan solo mejorar el producto. Sería añadir al programa nuevos métodos de representar los datos obtenidos. Algunos de estos métodos serían:

- Representación en eje de coordenadas: sería tan simple como lo que se vio en el apartado 3.2.2; representando en un eje de coordenadas el valor de la presión medida frente al tiempo transcurrido; de forma que se pueda observar la evolución de un ciclo de marcha (o varios).
- Mapa de presiones: similar a los mapas meteorológicos, se trataría de unir los puntos que estén a la misma presión y darles el mismo color, como se aprecia en la imagen 42. Esto es similar a lo que se ha hecho en este proyecto, pero no limitándose a los puntos representativos, si no haciendo una interpolación y dando valor a toda la planta del pie.
- Representación 3D: sería una mejora visual respecto al anterior, pues no sólo tendríamos colores, sino también "alturas". Sería contemplar la planta del pie desde una perspectiva que permita observar tanto el color como la altura correspondiente al valor que se obtenga del sensor. Un ejemplo se puede ver en la figura 43. Esto es posible apoyándose en el software de hojas de cálculo Excel, de Microsoft. La imagen

ha sido generada mediante los datos que se obtuvieron en una demostración de la plantilla; pero claro, de momento copiando y pegando manualmente los datos en una hoja de cálculo. La mejora está en que en un futuro, esto se podría hacer o bien sin apoyarse en el software de Microsoft; o de una forma automática.

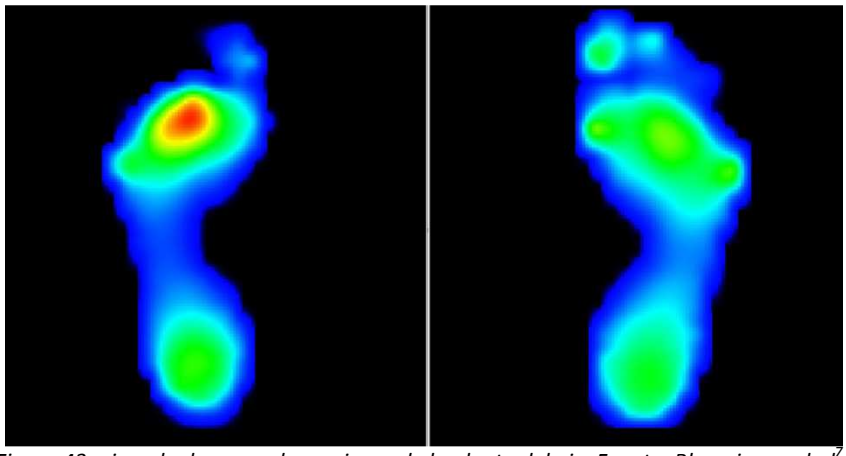


Figura 43: ejemplo de mapa de presiones de la planta del pie. Fuente: Blog pies y salud[7].

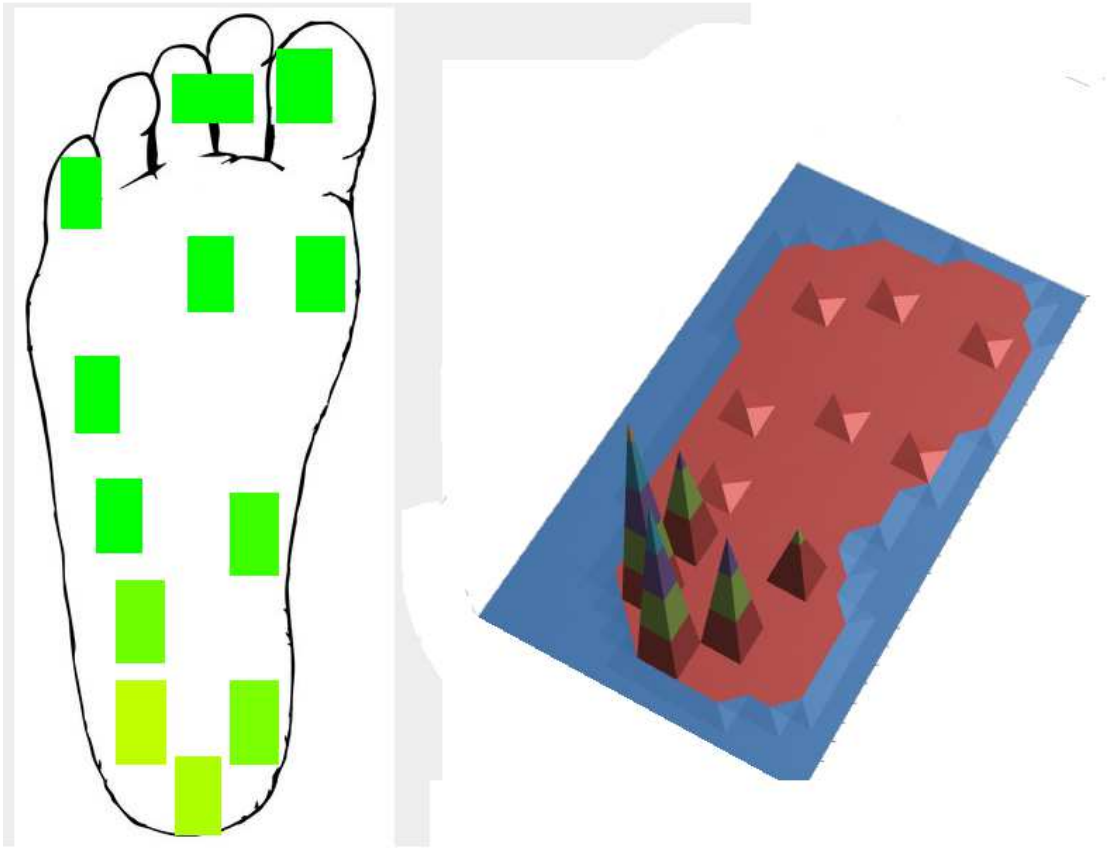


Figura 44: ejemplo de comparación entre los resultados del programa y los de un Excel para visionado 3D.

Portabilidad del sistema.

Otro aspecto que se apreció en las soluciones eran los métodos a distancia. Esto sería muy fácil implementarlo, añadiendo un módulo Bluetooth, o WiFi o cualquier tecnología

[7] "LA INFLUENCIA DEL PIE INESTABLE EN LA RODILLA."

disponible a la salida del PIC, para que en lugar de comunicarse por USB, lo hiciera por una de estas tecnologías. Esto haría no tener que depender de cables, pero tendría dos desventajas: por un lado, encarecería el sistema; pues habría que adquirir emisor y receptor de la tecnología usada; y por otro lado, sería necesario añadir sistema de alimentación portátil (baterías) que en un principio fue desechada esta idea.

Otra opción, sobre todo en estos días, es modificar el método de visualización. Sería posible comunicar las plantillas mediante un sistema basado en iOS o en android, lo que potenciaría la portabilidad del sistema. Esto se debe a que en la actualidad, al menos un portátil sería necesario para usar el producto. Si se añadiese la posibilidad de verlo en un *Smartphone* o *tablet*, está claro que sería tremendamente interesante y más portable que un ordenador portátil.

Otras mejoras.

Por último, se comentarán algunas mejoras más pequeñas que las anteriormente citadas, pero igual de interesantes.

- Añadir opciones de configuración. Esto es, por ejemplo, elegir la frecuencia de muestreo, elegir si se va a utilizar en uno u otro ambiente, cambiar la carpeta donde se guardan los datos de los pacientes...
- En la actualidad, existe una plantilla y un circuito. Sería quizá deseable modificar el software para que se mostrasen los dos pies al tiempo; claro que para esto, sería necesario o duplicar el circuito del PIC o multiplexarlo (pero asumiendo la bajada de la frecuencia máxima).

11. Bibliografía.

http://vivircondiabetes.net/analiza-tu-pisada-y-evita-complicaciones-del-pie-diabetico/

http://www.gym19.com.ar/biomecanica_pie.html

http://www.usb.org/home

http://www.ibv.org/index.php/es/productos/aplicaciones-biomecanicas/show_product/99/8

http://www.microchip.com/

Bronzino, Joseph D. *The Biomedical Engineering Handbook, Second Edition.* Boca Raton: CRC Press LLC, 2000.

Hanks, G. A., and A. Kalenak. "Running Injuries." *Clinical Sports Medicine*, 1982.

Hannula, M., A. Säkkinen, and A. Kylmänen. "Development of EMFI-Sensor Based Pressure Sensitive Insole for Gait Analysis," n.d.

Kerwin, David George. *Force Plate Analyses of Human Jumping*. Loughborough Universuty, 1997.

"LA INFLUENCIA DEL PIE INESTABLE EN LA RODILLA." *Pies Y Salud*, n.d. http://www.piesysalud.com/2012/03/la-influencia-del-pie-inestable-en-la-rodilla/.

"Low-Cost Fabric Tactile Sensing Skin." *Healthcare Robotics*, n.d. http://www.imsolidstate.com/archives/758#more-758.

Pallás Areny, R. *Sensores Y Acondicionadores de Señal*. Marcombo, 1994.

Prat, J., J. Sánchez - Lacuesta, and E. Alcántara Alcover. *Biomecánica de La Marcha Humana Normal Y Patológica*. Instituto de Biomecánica de Valencia, 1999.

Sierra, K., and B. Bates. *Head First Java*. O'Reilly, 2005.

Torres, Jorge, Antonio García, Carlos Villarraga, Alí Egel, and Rafael Polanía. "DESARROLLO DE UN SISTEMA ELECTRÓNICO PARA MEDIR AMBULATORIAMENTE PRESIONES EN LA PLANTA DEL PIE INSENSIBLE," n.d.

Valdés, F., and R. Pallas. *Microcontroladores: Fundamentos Y Aplicaciones Con PIC.* Marcombo, 2007.

II. ANEXOS

Índice

1. Módulo Interfaz.proyecto.

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

package interfaz.proyecto;

import javax.swing.ImageIcon;
import javax.swing.JLabel;
import java.awt.event.*;
import java.awt.Shape;
import javax.swing.*;

//librerías necesarias para la lectura y escritura de ficheros
import java.io.*;

//Librerías necesarias para la utilización del puerto de comunicaciones

import giovynet.nativelink.SerialPort;
import giovynet.permissions.Info;
import giovynet.serial.Baud;
import giovynet.serial.Com;
import giovynet.serial.Parameters;
import java.util.List;
import javafx.scene.paint.Color;

/**
 *
 * @author Toño
 */


public class InterfazProyecto {

    public class Data
    {
        public String datoRecibido;
        int valorNum[]= new int[16]; //creamos una matriz que contendrá el
valor numérico de los sensores.

    }


    public static void main (String[] arg) throws Exception{

        //codigo para mostrar la ventana principal

        VentanaPrincipal ventana = new VentanaPrincipal();
        ventana.setTitle("Plantar Preassure");
        ventana.setVisible(true);
        Data data = new InterfazProyecto().new Data();
```


```
//código para la comunicación serial

        showsInfoAboutGiovynetDriver();// Shows Information about Giovynet
Driver
        SerialPort serialPort = new SerialPort();
        List<String> lstFreeSerialPort =
serialPort.getFreeSerialPort();//Gets a list of serial ports free

        if(lstFreeSerialPort.size()>0){//if there are free ports
            System.out.println(""+lstFreeSerialPort);
            System.out.println(""+lstFreeSerialPort.get(1));
            ventana.jLabel5.setForeground(java.awt.Color.green);
            ventana.jLabel5.setText("Sí");
            Parameters parameters = new Parameters();//Create a parameter
object
            parameters.setPort(lstFreeSerialPort.get(1));//assigns the first
port found
            parameters.setBaudRate(Baud._460800);//assigns baud rate
            parameters.setByteSize("8");// assigns byte size
            parameters.setParity("N");// assigns parity
            Com com  = new  Com(parameters);// With the "parameters" creates
a "Com"

            do  {

            switch (interfaz.proyecto.VentanaPrincipal.presionadoBoton)
            {
                case 1:
            //Comieenza la adquisición de datos

//*****************************************************************

                //Para dotar sincronía, recibimos una cadena que siempre
marcará el inicio de la transmisión.
                //Esta cadena serán 4 nueves, creamos una variable para
facilitarnos la tarea de hallar el inicio de la cadena de 4 nueves.
                for (int  i = 0; i < 5000; i++){
                int j=0;
                int indice=0;
                int nueves=0;
                Thread.sleep(13);

                data.datoRecibido=com.receiveToString(190);

                System.out.println(data.datoRecibido);
                System.out.println(i);

                //Buscamos en la cadena los 4 nueves de inicio de cadena.

                do{ //Bucle para buscar dentro de la cadena los 4 nueves.
                    j++;
                    if(data.datoRecibido.charAt(j)=='9')
                        if(data.datoRecibido.charAt(j+1)=='9')
                            if(data.datoRecibido.charAt(j+2)=='9')
                                if(data.datoRecibido.charAt(j+3)=='9'){
                                    nueves=1;
                                    indice=j;
                                    j=170;
```

```
                                }
                } while (j<170);

                if (nueves>0)
                {
                    System.out.println("indice es "+indice);
                }
                else System.out.println("No se encontraron los caracteres de
inicio de cadena.");



                System.out.println("Valor sensor 1 =" +
data.datoRecibido.substring(indice+6,indice+10));

data.valorNum[0]=Integer.parseInt(data.datoRecibido.substring(indice+6,indice
+10));
                System.out.println("Valor sensor 2 =" +
data.datoRecibido.substring(indice+11, indice+15));

data.valorNum[1]=Integer.parseInt(data.datoRecibido.substring(indice+11,
indice+15));
```

(líneas similares)

```
            //************************************************************
            //Instrucciones para la muestra de los datos obtenidos
            //************************************************************
             int ganancia =20 ;
                if (interfaz.proyecto.VentanaPrincipal.presionadoBotonG==1)
                    ganancia = 20;
                else if
(interfaz.proyecto.VentanaPrincipal.presionadoBotonG==2)
                    ganancia = 10;
                else if
(interfaz.proyecto.VentanaPrincipal.presionadoBotonG==3)
                    ganancia = 5;

             // ********************Sensor 1************************
                int k=0;

                for(int ite=0;ite<31;ite++)
           {
               if(data.valorNum[k]>ite*ganancia &&
data.valorNum[k]<(ite+1)*ganancia)
                   ventana.jPanel9.setBackground(ventana.Colores[ite]);
               if(data.valorNum[k]>800)
                   ventana.jPanel9.setBackground(ventana.Colores[31]);
           }

             // ********************Sensor 2************************
             k=1;

             for(int ite=0;ite<31;ite++)
           {
               if(data.valorNum[k]>ite*ganancia &&
data.valorNum[k]<(ite+1)*ganancia)
                   ventana.jPanel5.setBackground(ventana.Colores[ite]);
               if(data.valorNum[k]>800)
                   ventana.jPanel5.setBackground(ventana.Colores[31]);
```

```
            }

             // ********************Sensor 3************************
             k=2;

             for(int ite=0;ite<31;ite++)
            {
                if(data.valorNum[k]>ite*ganancia &&
data.valorNum[k]<(ite+1)*ganancia)
                    ventana.jPanel3.setBackground(ventana.Colores[ite]);
                if(data.valorNum[k]>800)
                    ventana.jPanel3.setBackground(ventana.Colores[31]);
            }
```

(Líneas similares)

```
            //Abrimos el fichero seleccionado y escribimos sobre él los
valores de los sensores, si se ha presionado
                 //el  botón de iniciar toma de datos.

                     FileWriter fichero = null;
                     PrintWriter pw = null;
                     Object paciente= ventana.jComboBox1.getSelectedItem();

                     System.out.println(paciente);
                     try
                     {
                         fichero = new FileWriter("c:\\Pacientes\\"+paciente,
true);

fichero.write(data.datoRecibido.substring(indice,indice+77)+"\r\n");
                        // pw = new PrintWriter(fichero);
                          //pw.write(impFichero,70*i,70);


                     } catch (Exception e) {
                         e.printStackTrace();
                     } finally {
                     try {
                      // Nuevamente aprovechamos el finally para
                     // asegurarnos que se cierra el fichero.
                     if (null != fichero)
                          fichero.close();
                     } catch (Exception e2) {
                         e2.printStackTrace();
                     }
                     }
                     if(interfaz.proyecto.VentanaPrincipal.presionadoBoton!=1)
i=5000;
                 }
                     break;

                 case 2:

                     ventana.jPanel9.setBackground(ventana.Colores[0]);
                     ventana.jPanel8.setBackground(ventana.Colores[0]);
                     ventana.jPanel5.setBackground(ventana.Colores[0]);
                     ventana.jPanel2.setBackground(ventana.Colores[0]);
```

```
                    ventana.jPanel3.setBackground(ventana.Colores[0]);
                    ventana.jPanel13.setBackground(ventana.Colores[0]);
                    ventana.jPanel14.setBackground(ventana.Colores[0]);
                    ventana.jPanel12.setBackground(ventana.Colores[0]);
                    ventana.jPanel15.setBackground(ventana.Colores[0]);
                    ventana.jPanel22.setBackground(ventana.Colores[0]);
                    ventana.jPanel21.setBackground(ventana.Colores[0]);
                    ventana.jPanel10.setBackground(ventana.Colores[0]);
                    //ventana.jPanel15.setBackground(ventana.Colores[0]);
                    break;

                case 3:

                    Object paciente= ventana.jComboBox1.getSelectedItem();
                    String texto;
                    try
                    {
                        //Creamos un archivo FileReader que obtiene lo que
tenga el archivo
                        FileReader lector=new
FileReader("c:\\Pacientes\\"+paciente);

                        //El contenido de lector se guarda en un
BufferedReader
                        BufferedReader contenido=new BufferedReader(lector);

                        //Con el siguiente ciclo extraemos todo el contenido
del objeto "contenido" y lo mostramos

while(interfaz.proyecto.VentanaPrincipal.presionadoBoton==3)
                    {

                        texto=contenido.readLine();
                        texto=contenido.readLine();
                data.valorNum[0]=Integer.parseInt(texto.substring(1,4));
                data.valorNum[1]=Integer.parseInt(texto.substring(6,9));
                data.valorNum[2]=Integer.parseInt(texto.substring(11,14));
                data.valorNum[3]=Integer.parseInt(texto.substring(16,19));
                texto=contenido.readLine();
                            (líneas similares)


            //*********************************************************
            //Instrucciones para la muestra de los datos obtenidos
            //*********************************************************
            int ganancia =20 ;
                if (interfaz.proyecto.VentanaPrincipal.presionadoBotonG==1)
                    ganancia = 20;
                else if
(interfaz.proyecto.VentanaPrincipal.presionadoBotonG==2)
                    ganancia = 10;
                else if
(interfaz.proyecto.VentanaPrincipal.presionadoBotonG==3)
                    ganancia = 5;
            // ********************Sensor 1************************
                int k=0;
                for(int ite=0;ite<31;ite++)
          {
```

```
                if(data.valorNum[k]>ite*ganancia &&
data.valorNum[k]<(ite+1)*ganancia)
                    ventana.jPanel9.setBackground(ventana.Colores[ite]);
                if(data.valorNum[k]>800)
                    ventana.jPanel9.setBackground(ventana.Colores[31]);
            }

             // ********************Sensor 2************************
             k=1;

             for(int ite=0;ite<31;ite++)
            {
                if(data.valorNum[k]>ite*ganancia &&
data.valorNum[k]<(ite+1)*ganancia)
                    ventana.jPanel5.setBackground(ventana.Colores[ite]);
                if(data.valorNum[k]>800)
                    ventana.jPanel5.setBackground(ventana.Colores[31]);
            }

             // ********************Sensor 3************************
             k=2;

             for(int ite=0;ite<31;ite++)
            {
                if(data.valorNum[k]>ite*ganancia &&
data.valorNum[k]<(ite+1)*ganancia)
                    ventana.jPanel3.setBackground(ventana.Colores[ite]);
                if(data.valorNum[k]>800)
                    ventana.jPanel3.setBackground(ventana.Colores[31]);
            }
```

(líneas similares)

```
Thread.sleep(13*interfaz.proyecto.VentanaPrincipal.velocidad);
                    }

                    }


                         //Si se causa un error al leer cae aqui
                    catch(Exception e)
                    {
                        System.out.println("Error al leer");
                        VentanaError error = new VentanaError();
                        error.setTitle("FIN");
                        error.jLabel1.setText("Se finalizó la lectura del
fichero                                 ");
                        error.jLabel2.setText("");
                        error.setVisible(true);
                        interfaz.proyecto.VentanaPrincipal.presionadoBoton=2;
                    }


                    break;


             }
    }while (true);
```


```
           // System.out.println("-- End of transmission--");

        }else{
            System.out.println("There is no free serial ports.");
            ventana.jLabel5.setForeground(java.awt.Color.red);
            ventana.jLabel5.setText("No");
        }



   }


    /**
     * @param args the command line arguments
     */

       public static void showsInfoAboutGiovynetDriver(){
       System.out.println("-------------------------------------------------
----------------------------->");
       System.out.println("Giovynet Driver version "+Info.getVersion());
       System.out.println("Type of distribution:
"+Info.getTypeOfDistribution());
       System.out.println("Number of devices allowed:
"+Info.getNumDevicesAllowed());
       System.out.println("Distribution permissions:
"+Info.getDistributionLicense());
       System.out.println("<-------------------------------------------------
-----------------------------");
    }


}
```

2. Módulo ventana principal.

```
package interfaz.proyecto;


import java.awt.Color;
import java.awt.Dialog;
import javax.swing.ImageIcon;
import java.awt.Dimension;
import java.awt.Graphics;
import java.awt.Image;

//librerías necesarias para la lectura y escritura de ficheros
import java.io.*;

import javax.swing.ImageIcon;
import javax.swing.JFrame;
import javax.swing.JPanel;
import java.util.List;
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

/**
 *
 * @author Toño
 */
public class VentanaPrincipal extends javax.swing.JFrame {


    /**
     * Creates new form VentanaPrincipal
     */


    /**
     * Declaramos las variables correspondientes a los colores
     */
    Color Colores[]=new Color[32];
    int i=0;
    public static int presionadoBoton = 2; //Se crea una variable que
determinará qué función quiere el usuario usar.
    public static int velocidad = 1;
    public static int presionadoBotonG = 1; //Se crea una variable que
determinará la ganancia a aplicar.



    public VentanaPrincipal() {

        initComponents();
```

```
        //Código para la lectura del directorio
        File dir = new File ("c:\\Pacientes");
        File[] ficheros = dir.listFiles();

        for (int x=0;x<ficheros.length;x++){
        System.out.println(ficheros[x].getName());
        jComboBox1.addItem(ficheros[x].getName());
        }

        //Declaración de los colores
        Colores[0]=new java.awt.Color(0x00,0xFF,0x00); //verde
        Colores[1]=new java.awt.Color(0x1A,0xFF,0x00);
        Colores[2]=new java.awt.Color(0x2B,0xFF,0x00);
        Colores[3]=new java.awt.Color(0x3C,0xFF,0x00);
        Colores[4]=new java.awt.Color(0x4D,0xFF,0x00);
        Colores[5]=new java.awt.Color(0x5E,0xFF,0x00);
        Colores[6]=new java.awt.Color(0x6F,0xFF,0x00);
        Colores[7]=new java.awt.Color(0x7C,0xFF,0x00);
        Colores[8]=new java.awt.Color(0x8D,0xFF,0x00);
        Colores[9]=new java.awt.Color(0x9E,0xFF,0x00);
        Colores[10]=new java.awt.Color(0xAE,0xFF,0x00);
        Colores[11]=new java.awt.Color(0xBF,0xFF,0x00);
        Colores[12]=new java.awt.Color(0xCF,0xFF,0x00);
        Colores[13]=new java.awt.Color(0xDF,0xFF,0x00);
        Colores[14]=new java.awt.Color(0xEF,0xFF,0x00);
        Colores[15]=new java.awt.Color(0xFF,0xFF,0x00); //amarillo
        Colores[16]=new java.awt.Color(0xFF,0xFE,0x00);
        Colores[17]=new java.awt.Color(0xFF,0xED,0x00);
        Colores[18]=new java.awt.Color(0xFF,0xDB,0x00);
        Colores[19]=new java.awt.Color(0xFF,0xC8,0x00);
        Colores[20]=new java.awt.Color(0xFF,0xB6,0x00);
        Colores[21]=new java.awt.Color(0xFF,0xA4,0x00);
        Colores[22]=new java.awt.Color(0xFF,0x92,0x00);
        Colores[23]=new java.awt.Color(0xFF,0x80,0x00);
        Colores[24]=new java.awt.Color(0xFF,0x7F,0x00);
        Colores[25]=new java.awt.Color(0xFF,0x6E,0x00);
        Colores[26]=new java.awt.Color(0xFF,0x5D,0x00);
        Colores[27]=new java.awt.Color(0xFF,0x4B,0x00);
        Colores[28]=new java.awt.Color(0xFF,0x38,0x00);
        Colores[29]=new java.awt.Color(0xFF,0x26,0x00);
        Colores[30]=new java.awt.Color(0xFF,0x14,0x00);
        Colores[31]=new java.awt.Color(0xFF,0x00,0x00); //rojo


        interfaz.proyecto.InterfazProyecto.Data valor = new
interfaz.proyecto.InterfazProyecto().new Data();


        int valorNum[]= new int[16]; //creamos una matriz que contendrá el
valor numérico de los sensores.


    }
```

(Código generado por el entorno de programación)

```
    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
```


```
        VentanaSiNo error = new VentanaSiNo();
           error.setTitle("ADVERTENCIA");
           error.jLabel1.setText("Recuerde: Si el archivo contiene datos,
serán remplazados.            ");
           error.jLabel2.setText("¿Continuar?");
           error.setVisible(true);

           interfaz.proyecto.VentanaPrincipal.presionadoBoton=1;
           System.out.println("Iniciando!");

    }

    private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        System.exit(0);
    }

    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        presionadoBoton=2;
        System.out.println("Pausado..."+i);

    }

    private void jComboBox1ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        presionadoBoton =3;
    }

    private void jToggleButton1ActionPerformed(java.awt.event.ActionEvent
evt) {
        // TODO add your handling code here:
        String paciente;
        paciente = jTextField1.getText();

        if(paciente.length()<5)
        {
           VentanaError error = new VentanaError();
           error.setTitle("ERROR");
           error.setVisible(true);
        }
        else {
        jComboBox1.addItem("" + paciente+".txt");
        jTextField1.setText("");

        FileWriter fichero = null;
        PrintWriter pw = null;
        try
        {
            fichero = new FileWriter("c:\\Pacientes\\"+paciente+".txt");
            pw = new PrintWriter(fichero);
            //pw.println("Fichero " + paciente);

        } catch (Exception e) {
```


```
            e.printStackTrace();
        } finally {
           try {
           // Nuevamente aprovechamos el finally para
           // asegurarnos que se cierra el fichero.
           if (null != fichero)
              fichero.close();
           } catch (Exception e2) {
              e2.printStackTrace();
           }}
        }
    }

    private void jButton8ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        velocidad = 1;
        jLabel8.setText("1");
    }

    private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        velocidad = 20;
        jLabel8.setText("/2");
    }

    private void jButton6ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        velocidad = 40;
        jLabel8.setText("/4");
    }

    private void jButton7ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        velocidad = 80;
        jLabel8.setText("/8");
    }

    private void jButton9ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        presionadoBotonG = 1;
    }

    private void jButton10ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        presionadoBotonG = 2;
    }

    private void jButton11ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        presionadoBotonG = 3;
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting
code (optional) ">
```

```
        /* If Nimbus (introduced in Java SE 6) is not available, stay with
the default look and feel.
         * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {

javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(VentanaPrincipal.class.getName()).log(java
.util.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(VentanaPrincipal.class.getName()).log(java
.util.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(VentanaPrincipal.class.getName()).log(java
.util.logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(VentanaPrincipal.class.getName()).log(java
.util.logging.Level.SEVERE, null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new VentanaPrincipal().setVisible(true);

            }
        });

    }

    class ImagePanel extends JPanel {

  private Image img;

  public ImagePanel(String img) {
    this(new ImageIcon(img).getImage());
  }

  public ImagePanel(Image img) {
    this.img = img;
    Dimension size = new Dimension(img.getWidth(null), img.getHeight(null));
    setPreferredSize(size);
    setMinimumSize(size);
    setMaximumSize(size);
    setSize(size);
    setLayout(null);
```


```
  }

  public void paintComponent(Graphics g) {
    g.drawImage(img, 0, 0, null);
  }

}


    // Variables declaration - do not modify
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton10;
    private javax.swing.JButton jButton11;
    private javax.swing.JButton jButton2;
    private javax.swing.JButton jButton3;
    private javax.swing.JButton jButton4;
    private javax.swing.JButton jButton5;
    private javax.swing.JButton jButton6;
    private javax.swing.JButton jButton7;
    private javax.swing.JButton jButton8;
    private javax.swing.JButton jButton9;
    public javax.swing.JComboBox jComboBox1;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    public javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JLabel jLabel7;
    private javax.swing.JLabel jLabel8;
    private javax.swing.JLabel jLabel9;
    private javax.swing.JPanel jPanel1;
    public javax.swing.JPanel jPanel10;
    public javax.swing.JPanel jPanel12;
    public javax.swing.JPanel jPanel13;
    public javax.swing.JPanel jPanel14;
    public javax.swing.JPanel jPanel15;
    public javax.swing.JPanel jPanel2;
    public javax.swing.JPanel jPanel21;
    public javax.swing.JPanel jPanel22;
    public javax.swing.JPanel jPanel3;
    public javax.swing.JPanel jPanel5;
    public javax.swing.JPanel jPanel8;
    public javax.swing.JPanel jPanel9;
    private javax.swing.JTextField jTextField1;
    private javax.swing.JToggleButton jToggleButton1;
    // End of variables declaration
}
```

3. Archivo de salida del programa.

Es solo un fragmento, pues cada segundo se toman aproximadamente 80 muestras.

```
9999
0102  0021  0153  0054
0099  0012  0006  0000
0000  0004  0010  0011
0009
9999
0082  0023  0191  0057
0115  0044  0017  0007
0000  0003  0009  0008
0014
9999
0062  0008  0147  0063
0080  0024  0011  0000
0006  0006  0000  0012
0009
9999
0005  0000  0032  0010
0011  0005  0010  0006
0006  0000  0006  0012
0009
9999
0002  0000  0026  0008
0007  0003  0009  0001
0001  0003  0008  0002
0004
9999
0004  0004  0024  0001
0000  0004  0000  0004
0000  0006  0000  0000
0010
9999
0007  0005  0052  0004
0004  0009  0008  0001
0000  0004  0006  0009
0002
9999
0036  0000  0148  0037
0023  0019  0002  0000
0005  0001  0005  0014
0014
```

4. Algunas páginas de las hojas de características del PIC.

MICROCHIP **PIC18F2455/2550/4455/4550**

28/40/44-Pin, High-Performance, Enhanced Flash, USB Microcontrollers with nanoWatt Technology

Universal Serial Bus Features:

- USB V2.0 Compliant
- Low Speed (1.5 Mb/s) and Full Speed (12 Mb/s)
- Supports Control, Interrupt, Isochronous and Bulk Transfers
- Supports up to 32 Endpoints (16 bidirectional)
- 1 Kbyte Dual Access RAM for USB
- On-Chip USB Transceiver with On-Chip Voltage Regulator
- Interface for Off-Chip USB Transceiver
- Streaming Parallel Port (SPP) for USB streaming transfers (40/44-pin devices only)

Power-Managed Modes:

- Run: CPU on, Peripherals on
- Idle: CPU off, Peripherals on
- Sleep: CPU off, Peripherals off
- Idle mode Currents Down to 5.8 µA Typical
- Sleep mode Currents Down to 0.1 µA Typical
- Timer1 Oscillator: 1.1 µA Typical, 32 kHz, 2V
- Watchdog Timer: 2.1 µA Typical
- Two-Speed Oscillator Start-up

Flexible Oscillator Structure:

- Four Crystal modes, including High-Precision PLL for USB
- Two External Clock modes, Up to 48 MHz
- Internal Oscillator Block:
  - 8 user-selectable frequencies, from 31 kHz to 8 MHz
  - User-tunable to compensate for frequency drift
- Secondary Oscillator using Timer1 @ 32 kHz
- Dual Oscillator Options allow Microcontroller and USB module to Run at Different Clock Speeds
- Fail-Safe Clock Monitor:
  - Allows for safe shutdown if any clock stops

Peripheral Highlights:

- High-Current Sink/Source: 25 mA/25 mA
- Three External Interrupts
- Four Timer modules (Timer0 to Timer3)
- Up to 2 Capture/Compare/PWM (CCP) modules:
  - Capture is 16-bit, max. resolution 5.2 ns (TCY/16)
  - Compare is 16-bit, max. resolution 83.3 ns (TCY)
  - PWM output: PWM resolution is 1 to 10-bit
- Enhanced Capture/Compare/PWM (ECCP) module:
  - Multiple output modes
  - Selectable polarity
  - Programmable dead time
  - Auto-shutdown and auto-restart
- Enhanced USART module:
  - LIN bus support
- Master Synchronous Serial Port (MSSP) module Supporting 3-Wire SPI (all 4 modes) and I^2C™ Master and Slave modes
- 10-Bit, Up to 13-Channel Analog-to-Digital Converter (A/D) module with Programmable Acquisition Time
- Dual Analog Comparators with Input Multiplexing

Special Microcontroller Features:

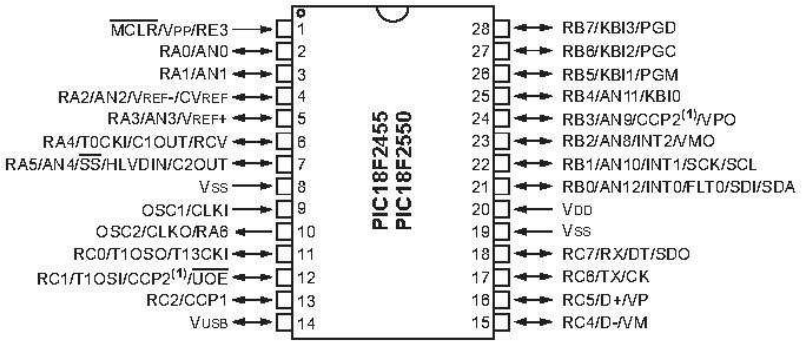
- C Compiler Optimized Architecture with Optional Extended Instruction Set
- 100,000 Erase/Write Cycle Enhanced Flash Program Memory Typical
- 1,000,000 Erase/Write Cycle Data EEPROM Memory Typical
- Flash/Data EEPROM Retention: > 40 Years
- Self-Programmable under Software Control
- Priority Levels for Interrupts
- 8 x 8 Single-Cycle Hardware Multiplier
- Extended Watchdog Timer (WDT):
  - Programmable period from 41 ms to 131s
- Programmable Code Protection
- Single-Supply 5V In-Circuit Serial Programming™ (ICSP™) via Two Pins
- In-Circuit Debug (ICD) via Two Pins
- Optional Dedicated ICD/ICSP Port (44-pin, TQFP package only)
- Wide Operating Voltage Range (2.0V to 5.5V)

Device	Program Memory		Data Memory		I/O	10-Bit A/D (ch)	CCP/ECCP (PWM)	SPP	MSSP		EUSART	Comparators	Timers 8/16-Bit
	Flash (bytes)	# Single-Word Instructions	SRAM (bytes)	EEPROM (bytes)					SPI	Master I^2C™			
PIC18F2455	24K	12288	2048	256	24	10	2/0	No	Y	Y	1	2	1/3
PIC18F2550	32K	16384	2048	256	24	10	2/0	No	Y	Y	1	2	1/3
PIC18F4455	24K	12288	2048	256	35	13	1/1	Yes	Y	Y	1	2	1/3
PIC18F4550	32K	16384	2048	256	35	13	1/1	Yes	Y	Y	1	2	1/3

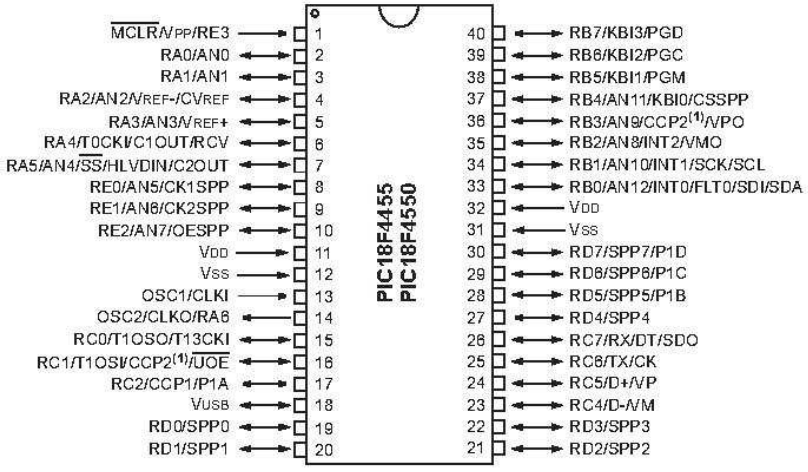
PIC18F2455/2550/4455/4550

Pin Diagrams

28-Pin PDIP, SOIC



40-Pin PDIP



Note 1: RB3 is the alternate pin for CCP2 multiplexing.

TABLE 1-1: DEVICE FEATURES

Features	PIC18F2455	PIC18F2550	PIC18F4455	PIC18F4550
Operating Frequency	DC – 48 MHz	DC – 48 MHz	DC – 48 MHz	DC – 48 MHz
Program Memory (Bytes)	24576	32768	24576	32768
Program Memory (Instructions)	12288	16384	12288	16384
Data Memory (Bytes)	2048	2048	2048	2048
Data EEPROM Memory (Bytes)	256	256	256	256
Interrupt Sources	19	19	20	20
I/O Ports	Ports A, B, C, (E)	Ports A, B, C, (E)	Ports A, B, C, D, E	Ports A, B, C, D, E
Timers	4	4	4	4
Capture/Compare/PWM Modules	2	2	1	1
Enhanced Capture/ Compare/PWM Modules	0	0	1	1
Serial Communications	MSSP, Enhanced USART	MSSP, Enhanced USART	MSSP, Enhanced USART	MSSP, Enhanced USART
Universal Serial Bus (USB) Module	1	1	1	1
Streaming Parallel Port (SPP)	No	No	Yes	Yes
10-Bit Analog-to-Digital Module	10 Input Channels	10 Input Channels	13 Input Channels	13 Input Channels
Comparators	2	2	2	2
Resets (and Delays)	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), MCLR (optional), WDT	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), MCLR (optional), WDT	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), MCLR (optional), WDT	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), MCLR (optional), WDT
Programmable Low-Voltage Detect	Yes	Yes	Yes	Yes
Programmable Brown-out Reset	Yes	Yes	Yes	Yes
Instruction Set	75 Instructions; 83 with Extended Instruction Set enabled	75 Instructions; 83 with Extended Instruction Set enabled	75 Instructions; 83 with Extended Instruction Set enabled	75 Instructions; 83 with Extended Instruction Set enabled
Packages	28-Pin PDIP 28-Pin SOIC	28-Pin PDIP 28-Pin SOIC	40-Pin PDIP 44-Pin QFN 44-Pin TQFP	40-Pin PDIP 44-Pin QFN 44-Pin TQFP