industriales
etsii

Escuela Técnica Superior de Ingeniería Industrial

UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Escuela Técnica Superior de Ingeniería Industrial

Sistema de ayuda a la conducción y toma de datos para motocicletas clásicas

TRABAJO FIN DE GRADO

GRADO EN ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

Autor: **Alberto Muñoz Ortega**
Director: José Alfonso Vera Repullo
Codirector: Pedro Díaz Hernández

Cartagena, ……

Universidad Politécnica de Cartagena

Índice

1.0 Introducción 6
2.0 Estado del arte 9
3.0 Memoria 19
3.1 Hardware: 19
3.1.1 GPS 19
3.1.2 GSM 22
3.1.3 Microcontrolador 24
3.1.4 Inclinómetro 26
3.1.5 Caudalímetro 27
3.1.6 Tarjeta SD 29
3.1.7 Pantalla 31
3.1.8 Temperatura Exterior 34
3.1.9 Temperatura Motor 35
3.2 Software 36
3.2.1 Arduino 36
4.0 Anexo 41
4.1 Descripción del funcionamiento del sistema 41
4.1.1 Inclinómetro: 43
4.1.2 Pantalla: 48
4.1.3 GPS y GSM: 56
4.1.4 Velocidad: 60
4.1.5 Revoluciones: 62
4.1.6 Caudalímetro: 63
4.1.7 Temperatura Exterior 63
4.1.8 Temperatura Motor 65
4.2 Planos 68
4.3 Presupuestos 75
4.4 DataSheets 76
4.4.1 Datasheet AD623 76
4.4.2 DataSheet MPU 6050 77
4.4.3 DataSheet GPS NEO 6 78
4.4.4 DataSheet SIM808 80
4.4.5 DataSheet BCX56 82
4.4.6 DataSheet Arduino Mega 84
4.4.7 DataSheet Arduino Micro 87

4.5 Código.....**¡Error! Marcador no definido.**
5.0 Futuras mejoras.....157
6.0 Conclusiones.....159
7.0 Bibliografía 160

Imágenes

Ilustración 1 : Evolución de los espacios del salpicadero....9
Ilustración 2 : Mercedes clase S de 1991. 9
Ilustración 3 : Byton M Byte....10
Ilustración 4 : Pantalla de Mazda....11
Ilustración 5 : Moto Honda con CarPLay....12
Ilustración 6 : Motocicleta Indian Ride Command....13
Ilustración 7 : App Indian Ride Command....14
Ilustración 8 : Display universal motocicleta 14
Ilustración 9 : e-Call....17
Ilustración 10 : Icono satelite....20
Ilustración 11 : Antena GPS....20
Ilustración 12 : Pigtail....21
Ilustración 13 : Módulo GPS....21
Ilustración 14 : Pigtail y antena GSM 22
Ilustración 15 : Módulo GSM 22
Ilustración 16 : Microcontrolador 24
Ilustración 17 : Caudalímetro....27
Ilustración 18 : Tarjeta microSd con adaptador....29
Ilustración 19 : Termometro Temperatura Exterior 34
Ilustración 20 : Termometro Motor....35
Ilustración 21 : Inclinómetro....43
Ilustración 22 : Inclinómetro Antes de calibración 44
Ilustración 23 : Valores de calibración del inclinómetro....45
Ilustración 24 : Offset calibración en programa....46
Ilustración 25 : Datos inclinómetro después de calibración. 47
Ilustración 26 : Prototipo de pantalla inicial 48
Ilustración 27 : Pantalla de carga 49
Ilustración 28 : Pestaña de conduccion 49
Ilustración 29 : Pestaña carrera /etapa....50
Ilustración 30 : Pestaña de configuracion....50
Ilustración 31 : Diferencias entre barras (verde) 51
Ilustración 32 : Diferencias entre barras (naranja) 51
Ilustración 33 : Diferencias entre barras (roja) 52
Ilustración 34 : Función posicionamiento cuadrado 52
Ilustración 35 : Bloque borrar cuadrados....53
Ilustración 36 : Bloque insertar cuadrados 54
Ilustración 37 : Colores cuadrados....54
Ilustración 38 : GPS NEO6 56
Ilustración 39 : Significado información AT....57
Ilustración 40 : Circuito transistor NPN....60
Ilustración 41 : Entrada (amarilla) y salida (azul) del transistor....61
Ilustración 42 : Circuito de reparación del transistor dañado 61
Ilustración 43 : Relación de magnitudes....62
Ilustración 44 : Sensor de temperatura DHT11 63
Ilustración 45 : Circuito sensor DHT11....64
Ilustración 46 : Pines DHT11 64
Ilustración 47 : PT100....65
Ilustración 48 : Estudio de resistencias para ajustar PT100....66
Ilustración 49 : Conexionado y pines de la pantalla....69
Ilustración 50 : Pantalla conectada al Mega 69

Agradecimientos

Primero que todo, quiero agradecer a los profesores
José Alfonso Vera Repullo y Pedro Díaz Hernández
haber depositado su confianza en mí para este proyecto.

Su ayuda y consejos han sido de gran importancia
para poder sacarlo hacia delante.

Agradecer también a los técnicos de laboratorio
y UPCT Racing Team por prestarnos su ayuda.

A los profesores y profesoras que me han impartido clase
estos últimos cuatro años.

A mis compañeros, que me han aguantado día sí,
y día también.

A mí a mi familia, que sin ellos,
hoy no podría estar escribiendo estas letras.

Y por último a mi Cristi, a la que le dedico el triunfo
de la carrera por intentar darme lo mejor y preocuparse
por mí cada día pase lo que pase.

1.0 Introducción

El proyecto a presentar, consiste en un sistema de visualización de datos para vehículos. Para poder llevar acabo dicho proyecto, han sido necesarias aplicar las diferentes disciplinas que nos ofrece la Universidad Politécnica de Cartagena en el grado de Electrónica Industrial y Automática. Los conocimientos que han aportado para que el proyecto haya sido posible vienen dados por la asignatura de Diseño y Simulación Electrónica, Programación de Sistemas en Tiempo Real e Instrumentación Electrónica.

Este proyecto, busca satisfacer las necesidades, en concreto de las motocicletas, para adquirir datos de diferente índole y representarlas al usuario mediante una pantalla. Por supuesto, existen aparatos en el mercado que pueden suplir esta necesidad en mayor o menor medida.

El proyecto invita al concepto “hágalo usted mismo” o Do it yourself, para pasar de tener que juntar varios aparatos para hacer lo que tú deseas, a construirte un aparato que hace lo que le pides, y además lo haces tú mismo, con menos coste (salvo tu valioso tiempo), aprendiendo y diseñándolo al gusto.

El mundo automovilístico, sufre el auge de las nuevas tecnologías y pocos son los coches de nueva hornada que no poseen una pantalla para gestionar la música, ver información detallada del kilometraje o consumos e incluso conectar nuestro móvil al coche para ver el correo o realizar llamadas entre otras tareas.

Sin embargo, el mundo de las motocicletas, este tipo de servicios, por temas de seguridad (que un piloto no puede o no debe, manipular con una mano la pantalla y con la otra sosteniendo el manillar de la moto, siendo éste, un acto que provoca un serio peligro

para el piloto así como para el resto de conductores o viandantes que lo rodeen), por espacio (no todas las motos tienen espacio para su colocación) y por su precio, hacen que este servicio sea prohibitivo para la mayoría de usuarios, por ello, el proyecto en cuestión, pretende ofrecer un producto barato, de tamaño relativamente pequeño (más pequeño que la pantalla de un móvil de hoy en día), ofreciendo unas opciones y características limitadas (tanto de software como de hardware), y no olvidando la seguridad del piloto, pudiendo usar el dispositivo sin levantar las manos del manillar.

Para comenzar a dar unas pinceladas al proyecto, toca decir que para que fuera posible, se necesitaron sensores, dispositivos y por supuesto, un microcontrolador que fuese capaz de gestionar la carga de trabajo que se le imponga. Y cómo no, es imposible concebir un microcontrolador sin un programa que ordene que debe hacer en cada momento, cuando parar una ejecución, y atender de manera urgente a algunas funciones si se encuentra en una situación límite o crítica.

Tanto software como hardware tienen un peso importante, cuanto mejor sea el microcontrolador, más rápido podrá ejecutar sus tareas y podrá realizar tareas más complejas en poco tiempo. Sin embargo, el software, tiene de especial, que es capaz de evitar que el microcontrolador trabaje en vano, o más de la cuenta, en pocas palabras, el software realizado a conciencia de los puntos débiles que posee el microcontrolador, pueden permitirte bajar su carga de trabajo, y destinar esos recursos a otras áreas, claro está, para poder diseñar de esa manera, antes exige un esfuerzo por conocer, qué límites posee.

Sin más preámbulo, pasaremos a comentar, qué clase de dispositivos necesitaremos y elementos software para poder llevar acabo el trabajo:

Elementos hardware:

1. GPS
2. GSM
3. Microcontrolador/es

4. Inclinómetro
5. Caudalímetro
6. Tarjeta SD

Elementos Software:

1. Programa estilo Processing
2. Programa para diseñar PCBs.

2.0 Estado del arte

Para ponernos en situación, en los últimos 60 años, la estética en los automóviles ha cambiado considerablemente. En los primeros 600[1], que no había ni retrovisores ni cinturones de seguridad, la radio era un extra apagar aparte que poseía el coche.

Ilustración 1 : Evolución de los espacios del salpicadero

Cada vez se fueron sofisticando más los salpicaderos, llenándose de botones y diales que ofrecían diversas funciones. Cada vez se parecían más y más a la cabina de una avión, haciendo que la "calidad o exclusividad del coche" fuese en función de la cantidad de botones que llevaba.

Ilustración 2 : Mercedes clase S de 1991.

A final de la década de los 90 [2], ya aparecieron los primeros paneles digitales, que bien podían ser fijos o escamoteables a modo de ordenador de viaje que mostraba información.

Los diseñadores de aquella época, aseguraban que era complicado integrarlas en el salpicadero, porque había que reubicar los mandos para hacerles sitio. Si antes se diseñaba el interior y luego se pensaba como se colocaba, ahora se piensa dónde va a ir la pantalla, y se diseña el interior en base a lo anterior.

Los relojes analógicos han dejado paso a la instrumentación digital, y las pantallas centrales, que sirven como centro de control de las funciones de a bordo, han permitido eliminar la mayoría de teclas físicas, depurar los salpicaderos y ofrecer presentaciones más limpias. Ahora parece que el estatus del coche se mide por el tamaño de su *display*.

Así pues, en los próximos años no deberíamos asustarnos si observamos que hay fabricantes que pretenden lanzar al mercado pantallas de hasta 49 pulgadas[3], pantalla en el volante del conductor, en el túnel central y dos en los asientos.

Ilustración 3 : Byton M Byte

Efectivamente, estamos ante un derroche de medios sin lugar a dudas, por no hablar, de las voces que apuntan a que este tipo de elementos en un automóvil, puede causar distracciones innecesarias al conductor, provocando situaciones de peligro a los ocupantes del vehículo a elementos cercanos que le rodeen mientras conduce.

En concreto, un fabricante[4], Mazda, a dado un paso atrás y dejará de usar pantallas en sus próximos vehículos. Un ingeniero de Mazda, ha declarado que mirar una pantalla distrae, si a eso le sumamos ha que hay que tocarla, el problema de agrava.

Cuando la pantalla posee funciones que solo pueden activarse mediante un sistema táctil, obligas al conductor a prestar más atención a la pantalla que a la propia conducción. Es por ello, que Mazda ha decido quitar las pantalla táctiles y sustituir estas por unas que no lo sean para evitar dicho problema de distracción. Además, también buscará que la posición donde estará situada la pantalla, evite al conductor tener que mover en exceso la vista para poder ver que se muestra en la pantalla en ese momento. Otro punto a tener en cuenta es que no será como la pantalla que se ha comentado anteriormente(Ilustración 3), si no que tendrá un tamaño más comedido para no restar visibilidad al conductor. Por último, destacar que al no ser táctil, volverá a los clásicos botones y diales para poder desplazarse por los menús y opciones que ofrezca.

Ilustración 4 : Pantalla de Mazda

Así pues, volvemos a los clásicos botones en el volante, dando a reflejar, que lo que tan bien a funcionado hasta ahora, no debe ser cambiado a la ligera si la seguridad depende de ello.

Lo anterior expuesto es una visión de cómo ha evolucionado la tecnología en los automóviles. En cuanto a las motos, su proceso es más delicado. Sin embargo, en 2018, un fabricante, Honda, dio un golpe en la mesa mostrando la primera moto, con características muy similares a las ofrecidas en los vehículos utilizando la característica CarPlay de Apple.

Ilustración 5 : Moto Honda con CarPLay

CarPlay es una forma de usar tu iPhone mientras vas conduciendo, enviar y recibir mensajes o escuchar la música de tu teléfono. Mediante Siri puedes dictar mensajes para otras personas o bien que te lea lo que te han enviado.

La moto en cuestión monta una pantalla de 7 pulgadas, y de nuevo, siguiendo el ejemplo de Mazda, el panel no es táctil y para navegar por él habrá que manejarse con los pulsadores de la empuñadura o bien vía bluetooth con un headset para que los menús nos respondan.

Existe otra motocicleta que posee un sistema propio, y no necesariamente debe tener un iPhone para funcionar. Se trata de la Indian Motorcycle Ride Command[5], en la cual, no necesita de un teléfono para funcionar aunque tiene la posibilidad de conectar el teléfono por bluetooth a la motocicleta.

Ilustración 6 : Motocicleta Indian Ride Command

Estamos hablando de una pantalla táctil a diferencia del caso anterior, pero el tamaño de pantalla no ha variado, es de 7 pulgadas como la Honda. Además, viene con una app para poder conectar tu móvil a la moto, y poder ver y transferir datos de la moto desde tu dispositivo.

Ilustración 7 : App Indian Ride Command

Como vemos, son soluciones específicas, caras, para motos de dimensiones muy grandes y aparatosas, por lo que el usuario de a pie puede no convencerle estas soluciones y prefiera algo más modesto.

Actualmente en el mercado, venden dispositivos, que permiten transformar las señales analógicas a otras digitales y mostrarlas en un display.

Ilustración 8 : Display universal motocicleta

La hora, la marcha, la velocidad, las revoluciones, los intermitentes, las luces etc. Estos y otros datos son los que podemos ver en esta pantalla LCD[6], por menos de 30 euros ya

podemos hacernos con una pantalla universal para nuestra moto, con un contenido más que amplio y con el cual cubrir nuestras necesidades más básicas a un precio realmente competitivo.

Entonces, el lector se estará preguntando, hemos visto soluciones muy caras, particulares y sofisticadas, y también baratas y con buen contenido, por lo tanto, ¿qué puede ofrecer este proyecto por el cual el consumidor pueda apostar?

La respuesta se halla en que es una mezcla entre las soluciones sofisticadas y las soluciones sencillas. Intentar coger lo mejor de cada clase parece un buen punto de partida, la vistosidad de una pantalla TFT-LCD frente a una LCD, no llegará a tener la misma calidad que una de alta gama pero la diferencia con respecto a una LCD ya en notable y su precio no se dispara.

Las funciones serán muy parecidas a las pantallas de baja gama, pero para darle un plus de servicio, incorporará un GPS, aunque éste solo tendrá la función de ver la cobertura que haya en el momento, comprobar que la motocicleta no ha sido sustraída, y añadir la hora en pantalla, sin embargo, será imposible ver caminos o carreteras por pantalla, algo que sí cuentan las de gama alta. En la gama baja directamente no hay gps, por lo que el proyecto ya cuenta con un punto a favor.

En términos de seguridad, los usuarios son capaces de pagar más por un producto que garantice que en caso de accidente, les pueda salvar la vida hasta que se vuelve en algo obligatorio. La historia del airbag refleja claramente este hecho.

Para poner en situación al lector[7], en 1952 apareció la patente de un airbag, a manos del ingeniero norteamericano John Hetrick. Los airbags[8] son bolsas de tejido, fibra sintética de poliamida o nailon, que se llenan de nitrógeno en el momento en que los sensores

dispuestos en el vehículo (pequeñas bolas de acero que, al moverse, cierran un circuito eléctrico) los disparan a partir de registrar impactos de 3 g.

El objetivo de los airbags es amortiguar a los ocupantes del vehículo el impacto contra distintos elementos rígidos de su interior, como el volante, el salpicadero o el parabrisas, los cristales y pilares laterales, etc. Sólo es realmente efectivo si se usa acompañado del cinturón.

Fue en 1981 cuando Mercedes, en su Clase S, incorporó los airbag que tenemos a día de hoy, prácticamente ha variado su funcionamiento. Aparte de los clásicos airbag en los asientos frontales, podemos ver modelos que los tienen instalados en los laterales o en los cinturones.

En 2006, este artículo ya no era una opción, se impuso como obligación que los vehículos dispusieran como mínimo los dos airbags frontales. Estos protegen en caso de impacto frontal a la cabeza y el tórax del conductor y el pasajero. El del primero se sitúa siempre en el núcleo central del volante; el del segundo, por lo general en el salpicadero, por encima de la guantera. En algunos coches este último también puede desplegarse desde la parte superior del techo, por ejemplo; o incluso de la zona inferior del salpicadero.

Recientemente, la Unión Europea, puso en marcha una normativa en el 2018[9], que los vehículos de nueva homologación fabricados dentro de la Unión, deben poseer un sistema llamado e-Call.

Ilustración 9 : e-Call

Se trata de un sistema de asistencia a la conducción que permite realizar una llamada automática en el caso de que los sensores del vehículo detecten que se ha podido producir una emergencia … y transmitir la localización GPS del vehículo, hora del accidente, tipo de vehículo y número de ocupantes a las centrales del 112 con el fin de agilizar el servicio, y reducir el tiempo de respuesta, vital en muchos casos de accidentes en los que el cronómetro no perdona.

El problema es que este sistema no está pensado para motocicletas a día de hoy, aunque en un futuro se espera que implante por completo en todo tipo de vehículos.

Es por ello que el proyecto quiere emular esta seguridad de alguna manera mandando un SMS a algún teléfono que el usuario indique. Este mensaje incluirá la posición GPS anunciando que se ha producido un accidente. Evidentemente, no estamos descubriendo América con este sistema, pero alcanzamos parte de esa tecnología a los usuarios que aún no pueden disfrutar de estas ventajas que propone la Unión Europea. Si bien es cierto, cuenta con los medios para poder avisar mediante un SMS al móvil que se indique, esta función no estará operativa, pues los accidentes de tráfico, son múltiples, son muchas las pruebas que hay que hacer para que sea un sistema homologado, y por tanto no se puede asegurar que

sea fiable por lo que en el apartado de mejoras del proyectos, abordaremos éstas y otras cuestiones.

Ofrecemos también, un sistema que permite al usuario más experimentado a comprobar qué tan bien sabe pilotar una moto en un circuito de carreras para probar sus habilidades y medir el tiempo que tarda en recorrer una serie de vueltas (las que el piloto desee, a través del menú de configuración), midiendo el tiempo que tarda en dar cada vuelta y el tiempo que tarda en recorrerlas todas. En caso de tratarse de alguna travesía, el contador llega a hasta las 10 horas.

El usuario, mediante un menú de configuración, podrá ajustar ciertos apartados para hacer el producto más personalizado. Con este y anteriores puntos, es por lo que nuestro proyecto se diferencia de lo que hay en el mercado, de tal modo que, compitiendo con la gama baja en mayor o menor medida, podamos ofrecer algo novedoso a un precio asequible.

3.0 Memoria

Como se ha comentado en la introducción, elementos hardware y software son imprescindibles. Para llevar un orden, empezaremos con los elementos hardware y seguiremos, por los elementos software.

3.1 Hardware:

3.1.1 GPS

El GPS, hoy en día estamos acostumbrados a relacionarlo con meternos en internet, teclear Google maps, y pedir llegar a algún sitio, ver cuánto se va a tardar, la distancia que hay que recorrer, ver los restaurantes, gasolineras, talleres mecánicos ... que hay por los alrededores. Nos ayuda a orientarnos, son más dinámicos, visuales, actualiza los cambios que puedan sufrir las carreteras, aparte de ser un servicio gratuito de Google, haciendo que los mapas “de la antigua usanza” queden en un segundo plano, destinados a desaparecer.

Esto es una realidad para numerosos vehículos, sin embargo, nuestro proyecto no cuenta con estas características. Esto es debido que es necesario mover una ingente cantidad de información gráfica, la cual no es posible con el microcontrolador que se va a utilizar, tampoco es posible por su manejo, pues para que sea práctico de usar, la pantalla debe ser táctil y en nuestro caso no va a ser así. También, por temas de conocimiento, no era viable introducirse en ese campo.

Así pues, la funcionalidad que tendrá nuestro GPS, será de seguridad y de adquisición de la hora. También, aprovechando que el GPS puede indicarnos el número de satélites, podemos de manera visual, alertar al conductor con el icono estandarizado del satélite, si está en posesión o no, de cobertura para poder fiarse de la información.

Ilustración 10 : Icono satelite

Cuantas menos barras curvas sobre el satélite tenga, menos cobertura tendrá, pudiendo a no llegar a tener ninguna, mostrando entonces que el GPS no se ha conectado, por lo que no será posible mostrar información, o la información que se muestre sea de dudosa reputación.

El GPS constará de 3 partes, el módulo GPS propiamente dicho, dado su reducido tamaño, necesitará un pigtail para conectarse a la antena, y efectivamente, la antena que será la encargada de enviar y recibir señales.

Ilustración 11 : Antena GPS

Ilustración 12 : Pigtail

Ilustración 13 : Módulo GPS

Para que el dispositivo pueda ofrecer los datos de manera que puedan ser leídos y manejados por el microcontrolador de manera sencilla, es necesario utilizar la librería #include <TinyGPS.h>.

3.1.2 GSM

El siguiente aparato a usar será uno que sea capaz de poder enviar SMS a dispositivos móviles, lo cual necesitará, como en el caso anterior, una antena GSM que le permita dicha función y el pigtail para poder conectar el módulo a la antena.

Ilustración 14 : Pigtail y antena GSM

Ilustración 15 : Módulo GSM

La información que vaya a mandar dicho módulo, se trata de la latitud y longitud en la cual se encuentre la motocicleta motocicleta. Este mensaje, tiene la finalidad de ayudarnos a encontrar nuestra moto si por algún motivo nos la sustrajeran de manera ilegal. Cada X tiempo, el microcontrolador guarda la latitud y la longitud en la EEPROM (memoria no volátil del microcontrolador) ofrecidas por el GPS, de tal modo, que cuando uno apaga la moto, queda almacenado dicha posición. El microcontrolador hace uso de su memoria EEPROM, almacenando dicha información, la próxima vez que el microcontrolador arranque, éste comprobará si las coordenadas ofrecidas por el GPS son más o menos iguales a las que guardó cuando se apagó (evidentemente, nunca podremos obtener los mismos datos aunque el aparato que los mida sea el mismo y la moto no haya cambiado su posición, por lo que se deja un margen de error), entonces, significará que la moto sigue en su sitio y que todo está en orden y la moto sigue en su sitio. Sin embargo, si las posiciones difieren, significará que la moto ha sido movida de manera deliberada y significaría que se ha producido un robo, por lo que debe avisar al dueño inmediatamente con un mensaje, señalizando con las nuevas coordenadas donde se encuentra la moto. Así, el dueño de la moto, podrá dirigirse a las autoridades correspondientes para poder iniciar la búsqueda de su motocicleta.

El tiempo el cual se guarda la posición, es algo problemático, pues si el usuario apaga la moto y aun no ha guardado las coordenadas porque aun no ha pasado el tiempo establecido para guardarlas, puede darse el caso, que en ese tiempo que no ha guardado nada, la moto se encuentre en una posición bien distinta a la anterior vez que se guardó, por lo que mandaría un mensaje diciendo que la moto ha sido sustraída la próxima vez que la encendamos aunque no nos la hayan robado. Claramente, cuando nos envíe ese mensaje nosotros estaremos frente a la moto y sabremos que esa información recibida a través de SMS será falsa pues estaremos junto a la moto. En el apartado del microcontrolador, explicaremos porqué podría sucedernos este suceso.

3.1.3 Microcontrolador

El microcontrolador, es uno de los actores principales en este proyecto, pues será el que tenga el papel de gestionar todas las operaciones que se le exijan. Tendrá que hacerse cargo de la pantalla y los sensores, así como de las interrupciones provenientes del caudalímetro, los pulsos de la velocidad y de las revoluciones.

El lector debe saber, que para que el precio sea ajustado, el microcontrolador debe ser barato, lo que implica que su memoria RAM, la memoria flash y la cantidad de ciclos de reloj que tenga serán reducidos en comparación con otros microcontroladores de gama alta. Por ello, optimizar los recursos en una tarea imprescindible.

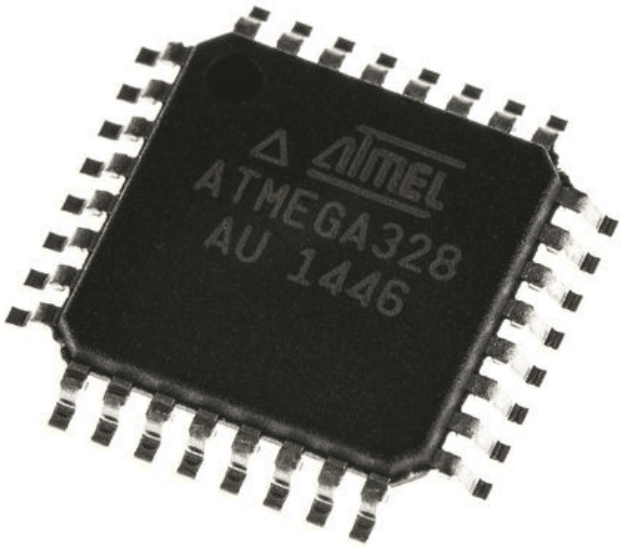


Ilustración 16 : Microcontrolador

Un detalle a tener en cuenta a la hora de optimizar recursos, es la memoria EEPROM, esta memoria, tiene un límite de escrituras, por lo que, si el programa está continuamente ordenando al microcontrolador que escriba, se dará el caso, más tarde o más temprano (dependiendo de las frecuencia a la que digas que escriba), la memoria llegará a su límite dejará de ser posible volver a escribir, por lo que necesitaríamos otro microcontrolador

nuevo. Esto a simple vista suena a un grave problema a tener en cuenta, pues un mal uso de esa memoria, puede acarrearnos tener que cambiar de microcontrolador cada poco tiempo, estaríamos hablando de una duración de días o unas pocas semanas. Es por ello, que hay que intentar limitar esas escrituras cada X tiempo para evitar estos problemas. Perderemos rápida respuesta a tener siempre los datos actualizados, pero a cambio obtendremos una mayor durabilidad.

Así mismo, una actividad que consume muchos recursos, es la pantalla. Por tanto, si estamos continuamente refrescando la totalidad de la pantalla a cada segundo, la carga de trabajo sería enorme y de cara al usuario, esto solo le causaría malestar, pues un producto que no se ve con la suficiente rapidez, provoca parpadeos y la sensación es la de un producto sin pulir, aparte de perder la confianza en el usuario.

Por eso mismo, el microcontrolador, solo debe dibujar en pantalla lo estrictamente necesario, y lo que verdad cambie (números, colores, barras de estado…). Es decir, si el piloto pasa de 120 kilómetros hora a 125 kilómetros hora, los dígitos 1 y 2 no cambiarán, solo cambiaría el 0 que pasaría a ser 5, de esa forma, no tenemos que estar continuamente borrando todos los números, y escribiendo de nuevo sobre la pantalla, ahorrando varios ciclos de reloj para que el microcontrolador los destine a otras áreas.

En resumidas cuentas, solamente cuando una variable cambia su valor, es entonces cuando se procede a su escritura, sea para un número, una barra de estado o la actualización de un icono.

3.1.4 Inclinómetro

El inclinómetro será un elemento importante a tener en cuenta para el dispositivo GSM y para la seguridad del piloto, pues cuando la aceleración caiga de un valor preestablecido a cero en una brevedad de tiempo, lo suficiente como para considerar que se ha producido una anomalía. También, si la motocicleta va a una cierta velocidad, y la moto sufre una inclinación brusca, podemos intuir que el piloto ha volcado de la moto y por tanto, también sería considerado como accidente. Dichas acciones, necesitarán un sistema rápido que las lea, por lo que es recomendable, que un segundo microcontrolador se haga cargo de manera exclusiva del sensor, así podrá atender de manera rápida todos los cambios, y, si es el caso, notificar el accidente mediante una interrupción al microcontrolador principal.

Para que el inclinómetro pueda funcionar de manera correcta, necesita unas librerías específicas, las cuales son:

```
#include "I2Cdev.h"

#include "MPU6050.h"

#include "Wire.h"
```

I2Cdev y Wire se utilizan para poder comunicarse mediante I2C, y MPU6050 es la librería propia del sensor.

I2Cdev y Wire se utilizan para poder comunicarse mediante I2C, y MPU6050 es la librería propia del sensor.

3.1.5 Caudalímetro

Para el piloto, medir la cantidad de combustible, es vital para su conducción, eso le puede ayudar a prever si debe parar a una gasolinera la próxima vez que divise una o por el contrario continuar su viaje.

Es por tanto que el caudalímetro es un compañero de viaje indispensable para planificar tus paradas y no quedarte tirado en la carretera. Es por eso, que contar el consumo, requiere de la máxima precisión. Un caudalímetro cuyo caudal mínimo no llegue a medir lo que pase a través de la bomba de la moto, nos dará medidas inexactas y de dudosa fiabilidad. Es por ello que no vale instalar cualquier caudalímetro, debe ser uno que asegure medir aun cuando por la bomba no circule gasolina en exceso.

Cada X pulsación que nos dé el caudalímetro, significará que han pasado tantos mililitros de gasolina, y son esos mililitros los que deberemos descontar del depósito total hasta llegar a cero.

Ilustración 17 : Caudalímetro

Para comodidad del usuario, tiene la opción de “recargar” virtualmente el depósito con el fin de indicar al microcontrolador que el usuario ha ido a una gasolinera y ha llenado el depósito al límite, así el microcontrolador podrá seguir contando el nivel de depósito sin errar.

En el apartado de configuración, que es donde se realiza esa “recarga virtual” del depósito, si por un casual, el usuario selecciona que desea recargarlo, pero físicamente no ha llenado el depósito de gasolina, el usuario, mientras no se haya ido de la pantalla de configuración, podrá cancelar dicha acción tantas veces como desee. En cuanto se vaya de la página de configuración, esos cambios serán permanentes y no habrá marcha atrás.

3.1.6 Tarjeta SD

Las tarjetas SD son un medio para llevar información almacenada y que no se nos borre aun cuando le falte energía eléctrica, así pues, es una memoria no volátil, como la EEPROM que se ha mencionado anteriormente.

Ilustración 18 : Tarjeta microSd con adaptador

Para que la SD sea detectada por el microcontrolador, depende de varios factores. Entre ellos es el tamaño de ésta, pues hay librerías que solamente llegan a poder gestionar hasta 4 Gb, serían librerías #include <Sd.h>. Sin embargo, los tiempos han cambiado, las memorias Sd cada vez poseen más memoria en el mismo espacio físico, por lo que esas librerías ya se quedan cortas y es necesario usar otras como #include <SdFat.h>, las cuales soportan tarjetas por encima de 4Gb. En nuestro caso, se ha escogido una tarjeta Sd de 16Gb por el simple hecho de estar en posesión de una.

Además, dado que la librería de la pantalla #include <TFT_HX8357.h> en las pruebas que se le realizaron no admitía una tarjeta sd de manera directa, hubo entonces que ingeniárselas para utilizar otras librerías que pudieran dar con la clave para que la Sd fuese reconocida, dichas librerías fueron la #include <UTFT.h> y la #include <UTFT_SdRaw.h>. Dicha librería UTFT nos venía con un programa que transformaba cualquier imagen en archivos .raw, facilitando la tarea. Hay que destacar que, en las pruebas, tan solo deja inicializar una librería de pantalla, bien TFT_HX8357 o bien UTFT, pero las dos al mismo

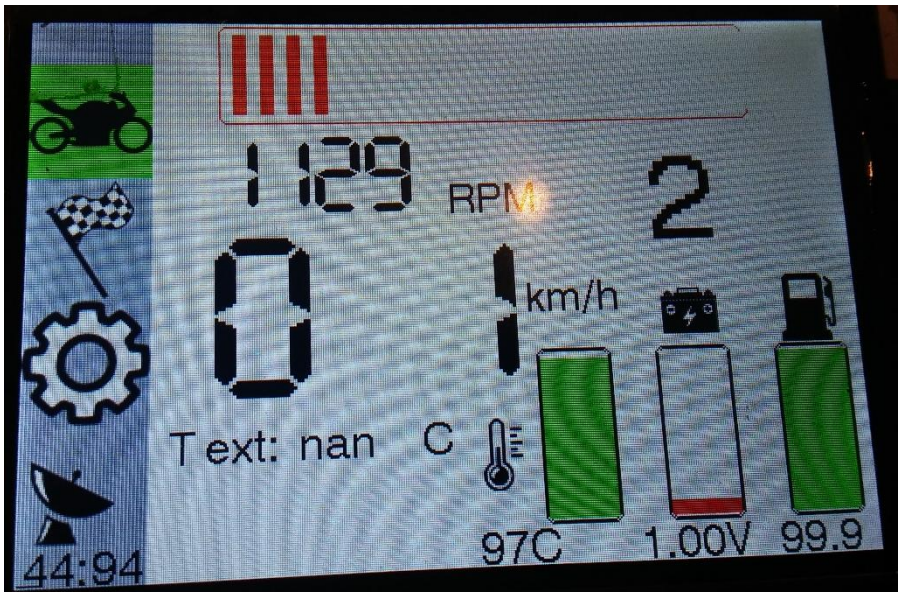
tiempo era imposible pues el microcontrolador se veía incapaz. Sin embargo, si inicializabas TFT_HX8357 y la librería UTFT no la inicializabas, podías llevarte ambas características al mismo tiempo, es decir, con la librería TFT_HX8357 podías imprimir números barras círculos etc… y con la UTFT podías usar la Sd sin el menor de los problemas, de tal forma que te podías beneficiar de ambas ventajas.

3.1.7 Pantalla

Otro elemento que no podía faltar a la cita, es lo que el usuario realmente puede ver, la pantalla. La pantalla será lo que el usuario vea a primera vista, por lo que su apariencia debe estar cuidada al detalle dentro de lo posible que nos permita el microcontrolador. En la pantalla se debe albergar la información de importancia al usuario. Se ha apostado por información numérica, pero también con señalizaciones gráficas para que, de un vistazo, sin tener que alejar la vista de la conducción, pueda ver si hay algún elemento que requiera atención, ya sea la batería, la temperatura del motor o el depósito de gasolina. Cuando los elementos gocen de un buen estado, sus barras representativas aparecerán en verde. Si empeora la situación, entonces su color pasará a ser naranja, y si sigue empeorando, acabará con un color rojo. En el caso de las revoluciones, antes de que llegue a las máximas revoluciones aparecerá una señal de alarma antes de sobrepasar ese límite.

Los números de la velocidad son de un tamaño considerablemente grande con respecto al resto de elementos en pantalla, siguiendo la misma filosofía de antes, que en un vistazo rápido veas lo más destacable.

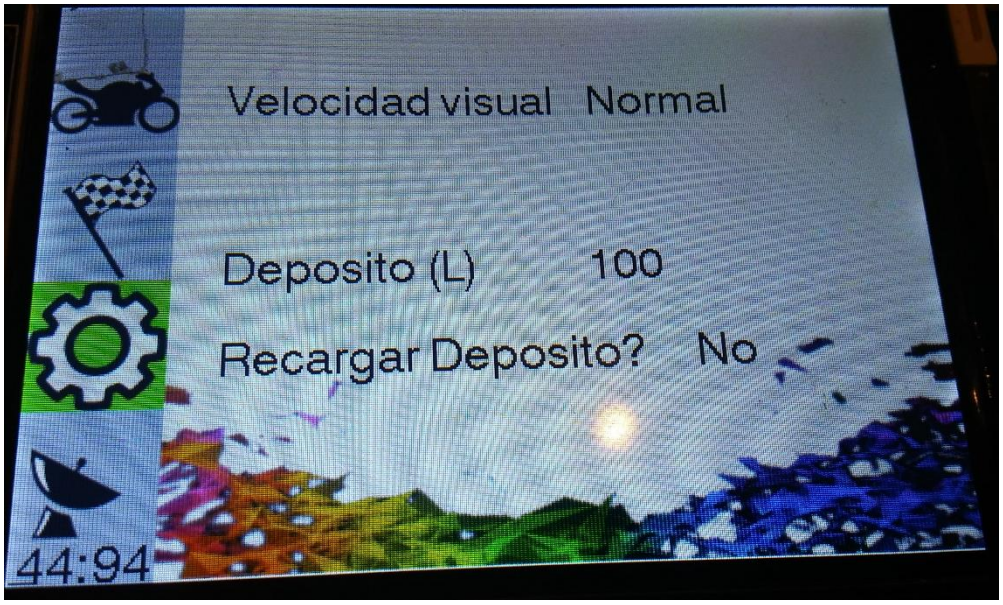
Un panel a la izquierda, indica sobre qué pantalla estamos, bien la de conducción (icono de la moto), bien la de etapas (icono de la bandera), o bien la de configuración (icono del engranaje). Dichos iconos, serán recreados mediante imágenes tipo RAW almacenados en la tarjeta Sd, facilitando así el diseño y evitar largas horas de trabajo intentando replicar esos iconos mediante figuras geométricas simples (líneas, cuadrados, círculos, triángulos y puntos). Por ello, la pantalla saca un buen potencial visual gracias a la Sd que le permite lograr tener un aspecto más profesional pero conservando la línea de ahorrar ciclos de reloj al microcontrolador.



Sabiendo las revoluciones y la velocidad, podemos ofrecer al usuario, qué marcha es la que lleva en cada momento.

En la pantalla de etapas, el piloto podrá ver en cada momento, el tiempo total que lleva recorrido, y el tiempo que lleva en cada vuelta. Cuando acabe las diferentes etapas, al final podrá revisar todos los tiempos realizados en cada vuelta. Como el tiempo es lo que prima en una carrera, ha tomado mayor peso en la visualización y por eso posee dígitos más grandes, pero no hay que dejar de lado que aun así, el piloto podrá consultar en esa misma pantalla la velocidad y las revoluciones que lleva.

En la pantalla de configuración, el piloto podrá elegir si desea ver la velocidad ofrecida por el GPS o por el tacómetro. Podrá decir cuántas marchas tiene su moto, cuantas etapas desea realizar, cuantos litros tiene su depósito o si desea recargar el depósito.



3.1.8 Temperatura Exterior

Medir la temperatura ambiente es una de las tareas que se lleva haciendo a lo largo de los siglos, con peores o mejores métodos, basándose primero en la dilatación de líquidos, luego por mercurio y ya finalmente por lo que hoy en día conocemos por termómetros digitales.

En este proyecto se utilizará un termómetro que mida la temperatura ambiente de manera que la información que nos ofrezca venga dada en señales temporizadas de manera que el microcontrolador pueda leer los datos en por un único cable.

Leer la temperatura exterior, no es un apartado crítico, pero por poco presupuesto se puede añadir, pues su coste en el mercado no es muy exigente y ofrecen precios populares siempre y cuando no se trate de un sensor de altísima precisión.

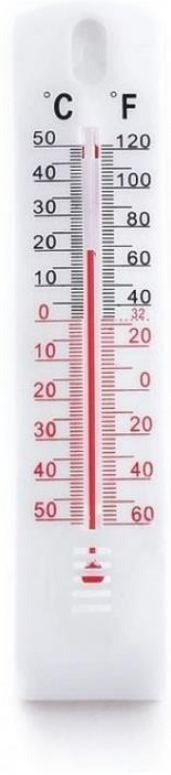


Ilustración 19 : Termometro Temperatura Exterior

3.1.9 Temperatura Motor

En el caso del motor, nos interesa medir la temperatura del agua, y mantenerla vigilada para que no alcance los 100 grados Celsius. Para ello, haremos uso de un termómetro que pueda alcanzar dichas temperaturas.

Sin embargo, no basta con que pueda medir esa temperatura. Se tendrá que situar muy próximo al bloque del motor, por lo que estará en constante vibración, (que a la larga puede ocasionar roturas en el termómetro) por lo que el material que cubra el termómetro no podrá ser de plástico, si no de un elemento metálico que soporte dichas inclemencias.

Nuevamente, como en el caso de la temperatura ambiente, éste termómetro deberá traducir la temperatura a señales digitales para que el microcontrolador pueda interpretarlas y pueda operar con ellas y reflejar la temperatura medida.

Ilustración 20 : Termometro Motor

3.2 Software

3.2.1 Arduino

El software, como ya hemos comentado con anterioridad, está fuertemente ligado al microcontrolador, pues un buen software que sepa bien como optimizar los recursos del microcontrolador, podrá realizar más tareas con las mismas condiciones.

El software utilizado por excelencia del movimiento conocido como “Do it yourself” es el Arduino, basado en Processing para facilitar la programación a los diseñadores. Así pues, a sus espaldas hay una gran comunidad que permite a que mucha gente ajena a la electrónica pueda llegar a realizar sus propios proyectos.

Es cierto, que una vez que se van añadiendo más y más cosas al proyecto, el programa va creciendo de manera que empieza a convertirse en un problema para el diseñador, pues es fácil perder la noción de qué se está haciendo o dónde estaba tal o cual función. El impedir que el usuario pueda ocultar o hacer visibles las funciones a modo de despegable, hace que el exceso de información acabe por confundir al diseñador y no ver las cosas claras. En la mayoría de casos, puede llegar perder el interés por lo que está haciendo y echar más horas al proyecto para poder asimilar lo que está escrito.

En cambio, ocultar funciones agiliza al vuelo una visión general del estado del programa.

Muchas veces, lo que creemos que está bien cuando realizamos un programa, más tarde es probable que no hayamos tenido en cuenta un detalle insignificante pero que sea clave para que nuestro programa haga lo que le pidamos, por lo que se requiere paciencia si las cosas no salen a la primera, que es algo muy habitual. Más tarde, después de pasar horas y horas programando, los fallos que nos vayan saliendo, podrán ser vistos de manera más

rápida y se podrá abordarlos de manera eficaz. La satisfacción de ver que las cosas empiezan a funcionar serán el mayor de los consuelos.

3.2.2 DipTrace

En cambio, ocultar funciones agiliza al vuelo una visión general del estado del programa.

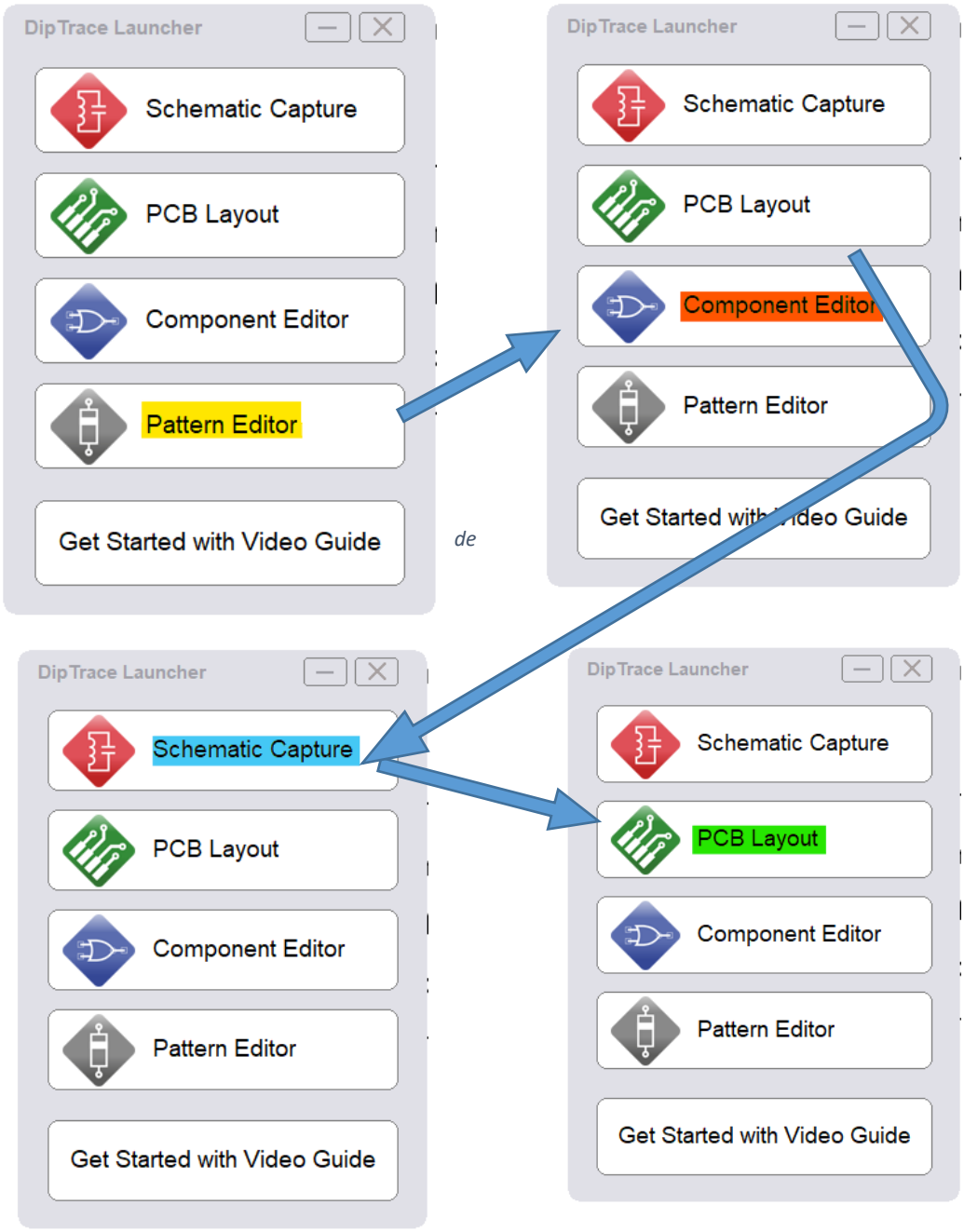
Uno de los apartados del proyecto consiste en hacer una placa PCB para poder agrupar los diferentes componentes y dispositivos que engloban el proyecto a modo de conseguir que las conexiones y firmeza de todas las piezas soldadas garanticen un buen funcionamiento y eviten que los componentes se desconecten o salten de sus zócalos (cosa que sucedería si el montaje de los componentes estuviera montado en una protoboard y no en una placa soldada).

Para poder realizar una PCB, lo primero que hay que tener claro, es qué componentes se utilizarán, para ello si el programa no posee los componentes que buscamos, entonces tendremos que crearlos nosotros mismos.

Así pues, el programa nos permite hacer dicha función. Primeo que todo deberemos realizar un pattern del componente que deseamos con sus medidas reales en la opción *Pattern Edittor*. Una vez que esté hecho, entonces podremos hacer el componente en el apartado *Component Edittor*. Como tercer paso, teniendo los componentes listos, deberemos agruparlos en un esquemático y realizar las conexiones pertinentes, dicha acción se realizará en el *Schematic Capture*. Finalmente, como cuarto paso, teniendo ya todas las conexiones realizadas, esteremos en condiciones de poder realizar nuestra PCB en el apartado *PCB Layout.* Allí, deberemos abordar qué medidas tendrá la placa y ajustar los componentes para que quepan en ese espacio.

Es bien sabido que el programa, una vez colocas los componentes dentro de lo que vendría a ser el tamaño de la placa, te permite *Routear* las pistas de manera automática. Sin embargo, no es conveniente dejarle que realice ese trabajo de primeras, pues en la mayoría de los casos deja pistas sin *routear,* por lo que se recomienda que se agrupen los componentes que se encuentran conectados unos a otros, en una posición cercan de manera

que el programa, así por ejemplo, que no tenga que *routear* dos resistencias cruzando la placa de lado a lado pudiendo haberlo evitado si hubieran estado la una pegada a la otra, permitiendo además, que otros componentes se pudieran beneficiar por el espacio dejado al tratarse de una pista de grandes longitudes.



de

Ilustración 21 : Menú Diptrace (pasos 1-4)

4.0 Anexo

4.1 Descripción del funcionamiento del sistema

El proyecto a presentar, se trata de representar la información habitualmente recogida en un cuadro de instrumentos, (tacómetro, cuenta revoluciones, nivel de depósito, temperatura del motor…), así como incluir ciertas características como la lectura del nivel de batería, poder elegir entre si deseamos ver la velocidad que nos ofrece el tacómetro propio de la moto o por el contrario usar la velocidad que nos ofrece el GPS. Utilizar las posiciones latitud y longitud del GPS para poder detectar, si se da el caso, que hayamos sufrido un robo del vehículo, (cuando la moto vuelve a ser encendida en un lugar que no fue el mismo al apagarse), en ese caso mandaría un SMS añadiendo la posición en la que se encuentra la moto. Posee una pantalla que nos permite ver, en una competición de etapas o vueltas, el tiempo total y el tiempo entre vueltas que vamos obteniendo, además de una última pantalla que nos permite configurar ciertas características, como los litros del depósito, o como se ha comentado antes, el poder elegir la velocidad a representar.

Este proyecto comprende mayoritariamente un trabajo de programación, pero también cabe destacar la importancia de los sensores y su acondicionamiento para que sus lecturas puedan ser leídas por el microcontrolador, que en este caso será un Arduino Mega. La elección de un Arduino Mega es debida a su forma de programar, a la gran comunidad y apoyo que posee actualmente y sobre todo, su precio.

Muchos de los sensores que se van a utilizar están preparados para Arduino, luego también es un aliciente su uso.

Los sensores que se van utilizar son: un inclinómetro MPU-6065 para medir aceleraciones y rotaciones en los 3 ejes, un GPS NEO 6M, para poder obtener la hora, la latitud, la longitud y los satélites, un Sim808 para poder enviar mensajes SMS, un

caudalímetro para llevar la cuenta del gasto de combustible, un sensor de temperatura para tener vigilante el calentamiento del motor, tres divisores de tensión que nos permitan acondicionar las señales a los voltajes del Arduino, tanto de la velocidad, de la tensión como de las revoluciones, una pantalla TFT-LCD de 480x320 píxeles par Arduino Mega 2560 y una tarjeta Sd (la cual se instala en un slot de la pantalla y se conecta por vía SPi al Arduino) para facilitar el almacenaje de imágenes.

El esquemático se realizará mediante Diptrace, en el cual podremos hacernos una idea de manera más amplia cómo irá todo conectado. Para ello deberemos crear los patterns de los diferentes elementos, crear de forma aproximada el dibujo de cada componente y asociarle el pattern asociado, seguidamente podremos crear el conjunto del circuito en el schematic, y si así se desea, pasar el circuito a PCB, para poder mandarlo a realizar, de manera que se obtenga un producto con buenos acabados.

4.1.1 Inclinómetro:

El MPU-6050 incorpora un procesador interno (DMP Digital Motion Processor) que ejecuta complejos algortimos de MotionFusion para combinar las mediciones de los sensores internos, evitando tener que realizar los filtros de forma exterior.

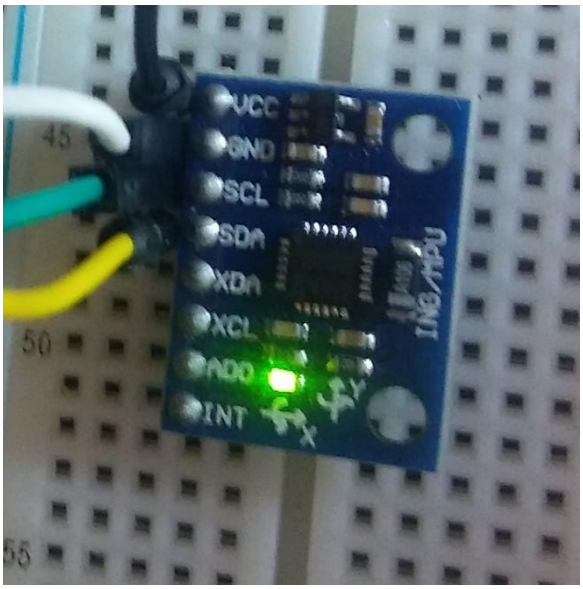


Ilustración 22 : Inclinómetro

Primero de todo, debemos calibrar el aparato, debido a que, aunque poseas varios inclinómetros de la misma marca, los datos serian dispares entre ellos, pues internamente no son iguales de fábrica, por lo que cada uno tendrá su propio offset. Para corregir los datos que nos ofrezca, debemos probar un programa[10] que se encarga de sacar dichos valores, los cuales luego pondremos en el programa final.

Para que veamos una comparación, vamos a mostrar los datos que nos salen del inclinómetro, antes y después de calibrar, además de mostrar los resultados de la calibración.

Datos del inclinómetro antes de la calibración:

Ilustración 23 : Inclinómetro Antes de calibración

Como podemos apreciar, de vez en cuando la aceleración en el eje Z nos da valores superiores a 9,8 llegando a los 10 incluso, al estar en una posición horizontal y no estar en movimiento, su valor debiera aproximarse al valor de la gravedad lo máximo posible y para el resto de valores, su valor debería ser próximo a cero.

Ahora calibramos:

Y nos salen los siguientes resultados:

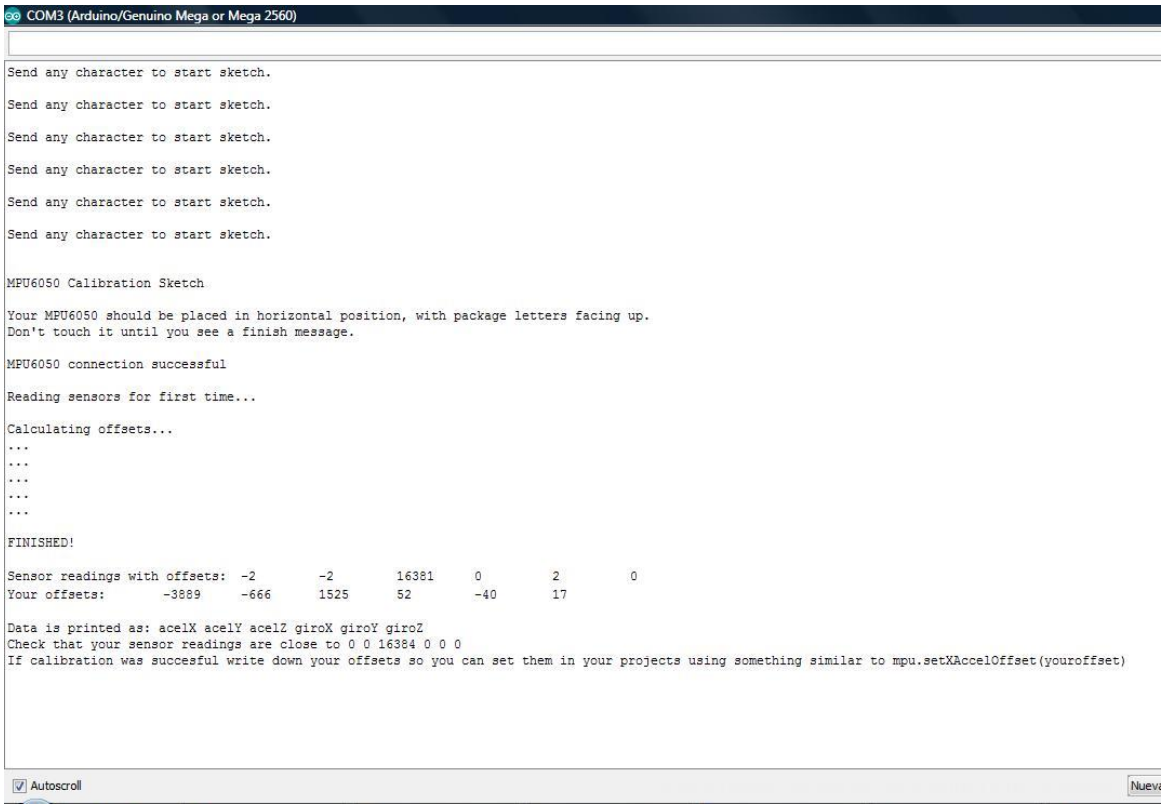


Ilustración 24 : Valores de calibración del inclinómetro

Tal y como sale en la imagen, para que la calibración sea válida, nos deben salir unos valores próximos a:

0 0 16384 0 0 0

Como nos ha salido: -2 -2 16381 0 2 0, podemos decir que la consideramos válida. Así pues, nos han salido unos valores de offset de:

Aceleración en eje X= -3889

Aceleración en eje Y= -666

Aceleración en eje Z= 1525

Rotación eje X= 52

Rotación eje Y= -40

Rotación eje Z=17

Dichos valores los metemos dentro del programa tal que así dentro del setup del programa

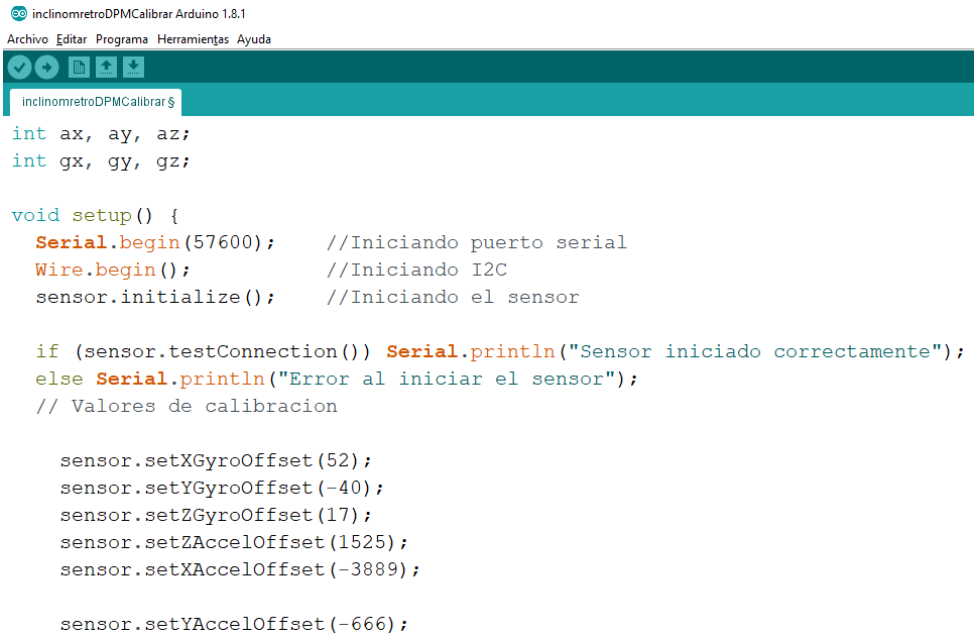


Ilustración 25 : Offset calibración en programa

Comprobemos si ahora esos valores nos dan mejor resultado en los datos obtenidos.

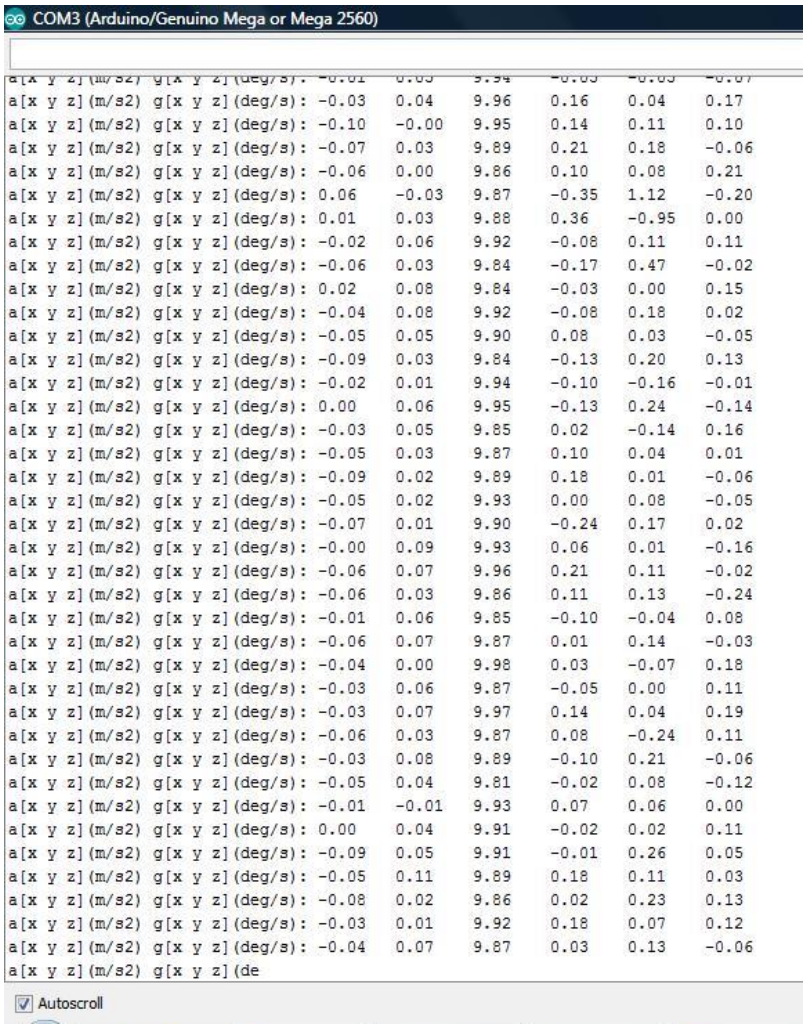


Ilustración 26 : Datos inclinómetro después de calibración.

Como vemos, ya no nos aparecen aceleraciones en el eje Z con valores superiores o iguales a 10, predominando los valores cercanos a 9,8. Cabe destacar que ben el resto de casos el ofsett no ha conseguido aproximar más a cero, en cuyo caso, anteriormente a la calibración, no obteníamos del todo malos resultados, por lo que cabe esperar a que estos offset puedan hacerse más notorios cuando se encuentre el dispositivo en movimiento.

4.1.2 Pantalla:

A la hora del diseño se acordó de hacer 3 pantallas diferenciadas, una con todos los datos necesarios para la conducción del piloto, otra para poder llevar la cuenta de las vueltas/etapas y el tiempo de realización, y finalmente una de configuración.

El primer prototipo de pantalla fue concebida como una pantalla principal, en la cual debías seleccionar a que pestaña ir de las 3 opciones que sugería (como acabamos de mencionar, sería algo así como pestaña conducción, pestaña carrera/etapas y pestaña configuración).

Ilustración 27 : Prototipo de pantalla inicial

Tal como se observa los nombres difieren de los dichos anteriormente, pero se trataba de una prueba para ver cómo se desenvolvía en ese modo de funcionamiento, más tarde, se cambió de opinión, y se prefirió que cada vez que se pulsara un botón, la pantalla cambiase

de una pestaña a otra, por tanto el concepto de menú desapareció, convirtiendo el funcionamiento de la pantalla mucho más directo y dinámico.

Cuando la pantalla se enciende, nos genera una imagen de inicio, la cual nos muestra el logo de la UPCT, sin embargo, este logo ha sido modificado a propósito para darle más colorido y haciendo que gane un toque de originalidad.

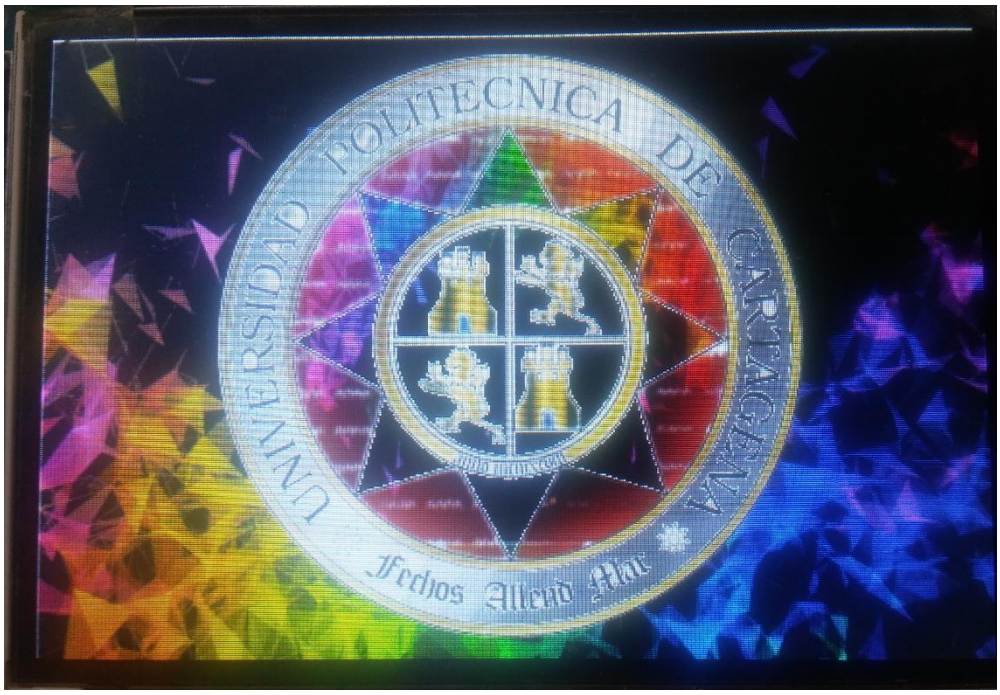


Ilustración 28 : Pantalla de carga

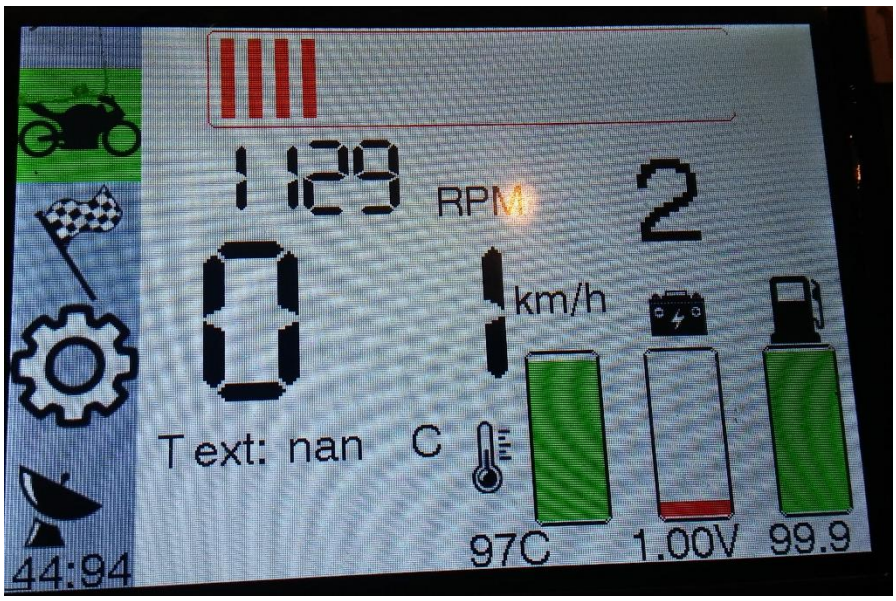


Ilustración 29 : Pestaña de conduccion

Ilustración 30 : Pestaña carrera /etapa

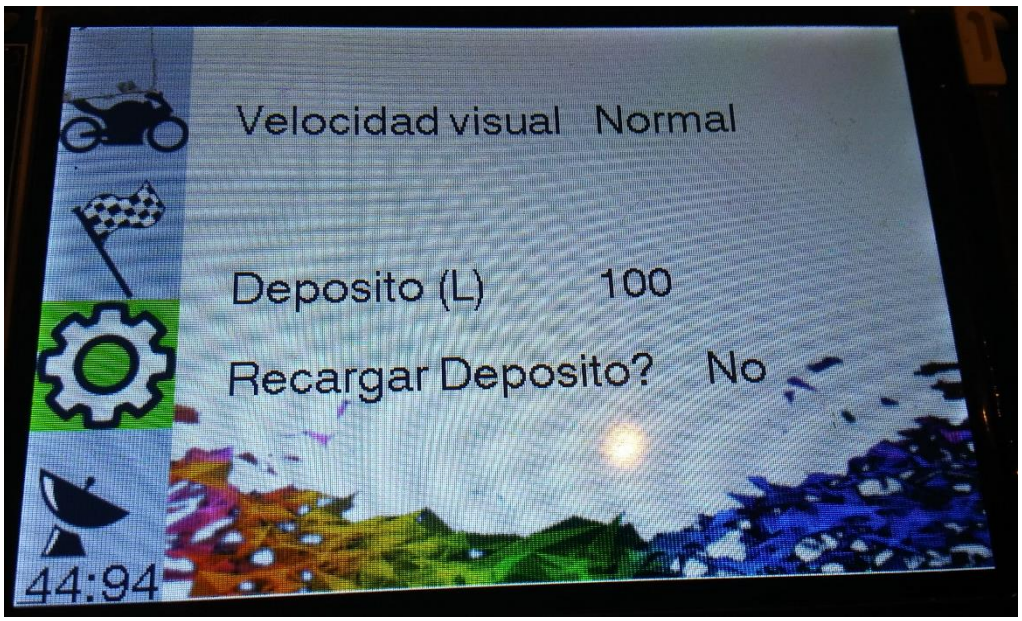


Ilustración 31 : Pestaña de configuración

Una modificación considerable a destacar fue la de cambiar las barras de degradado, por unas de color sólido como las que vemos en la Ilustración 29. Su diferencia y funcionamiento es el siguiente:

Ha pasado de 20 cuadrados para formar una barra a ser un cuadrado que su dimensión y color cambiar con respecto el valor de la variable, es decir, a mayor es el valor de la variable mayor es el cuadrado, y a menor sea menor será el tamaño de este, cambiando a su vez los

distintos colores según convenga. En el caso anterior, los diferentes cuadrados, ofrecían un gradiente de colores y no un color sólido, pero el funcionamiento era el mismo que el anterior, mayor valor, mayor era el cuadrado, menos era el cuadrado cuanto menor fuese la variable.

Veamos una imagen para verlo.

Hemos colocado que cada tercera parte del valor aproximadamente cambie su valor, por debajo de 70 pasa de verde a naranja, por debajo de 30, pasa de naranja a rojo.

Ilustración 32 : Diferencias entre barras (verde)

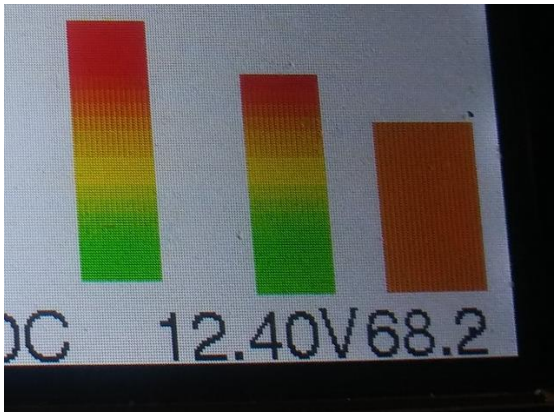


Ilustración 33 : Diferencias entre barras (naranja)

Ilustración 34 : Diferencias entre barras (roja)

El código cuando tenemos una barra formada por 20 cuadrados diferentes queda de la siguiente manera.

```
void posicionamientodeposito()
{

  for (rellenodeposicionesXD = 0; rellenodeposicionesXD < 20; rellenodeposicionesXD++)
  {
    if (horizontalD)
    {
      misposicionesXD[rellenodeposicionesXD] = posx1D;
      posx1D = posx1D + (anchuraD);
    }
    else {
      misposicionesXD[rellenodeposicionesXD] = posicionhorizontalfijaD;
    }
  }
  for (rellenodeposicionesYD = 0; rellenodeposicionesYD < 20; rellenodeposicionesYD++)
  {
    if (verticalD)
    {
      misposicionesYD[rellenodeposicionesYD] = posy1D;
      posy1D = posy1D - (anchuraD);
    }
    else {
      misposicionesYD[rellenodeposicionesYD] = posicionverticalfijaD;

    }
```

Ilustración 35 : Función posicionamiento cuadrado

Para el ejemplo del depósito, necesitamos una función que se encargue de dar a cada cuadrado su posición X e Y, debiendo elegir, si los cuadrados se dibujarán a lo largo (horizontal) o alto (vertical) de la pantalla. Una vez que se elige una de las dos disposiciones a representar, automáticamente, X o Y adquirirá un valor contante (una barra vertical tendrá su componente X contante, mientras que una barra horizontal, su componente constante será la Y). Esto se hace al principio del programa (setup).

```
void  indica_temperatura(int numero)
{

  intervalo = map(numero, 0, numeromax, 0, 21);



  if (intervalo + 1 <= intervalodedondevengo)
  {
    for (int p = intervalodedondevengo; p > intervalo + 1; p--)
    {
      if (p == 1)
      {
        tft.fillRect(misposicionesX[0], misposicionesY[0], precision, altura, TFT_BLACK);
        intervalodedondevengo = p;

      }

      if (p == 2)
      {
        tft.fillRect(misposicionesX[1], misposicionesY[1], precision, altura, TFT_BLACK);
        intervalodedondevengo = p;
      }
```

Ilustración 36 : Bloque borrar cuadrados

La función que pinta cuadrados, necesita mapear el valor que le llega con el máximo valor que esa variable puede alcanzar, de esta manera, el mapeo controlará los cuadrados que deben añadirse o borrarse para que se cumpla la relación, y sea coherente, el valor con su representación gráfica. Primero, arranca con un bucle for, comprobando si por ejemplo, estando en el cuadrado 19, nuestro intervalo ahora pasara a ser el de 15, entonces tendría que borrar 4 cuadrados para llegar al 15 (se borra siempre con el color de fondo de la pantalla).

Si por el contrario estamos en el cuadrado 7, y el intervalo nos marca el 12, entonces debemos rellenar 5 cuadrados, y `puesto que son independientes, pueden tener el color hexadecimal de 16 bits que deseemos[11].

```
if (intervalo + 1 >= intervalodedondevengo)
{
  for (int p = intervalodedondevengo; p < intervalo + 1; p++)
  {
    if (p == 1)
    {
      tft.fillRect(misposicionesX[0], misposicionesY[0], precision, altura, verde1);
      intervalodedondevengo = p;

    }

    if (p == 2)
    {
      tft.fillRect(misposicionesX[1], misposicionesY[1], precision, altura, verde2);
      intervalodedondevengo = p;
    }
    if (p == 3)
    {
      tft.fillRect(misposicionesX[2], misposicionesY[2], precision, altura, verde3);
      intervalodedondevengo = p;
    }

    if (p == 4)
    {
      tft.fillRect(misposicionesX[3], misposicionesY[3], precision, altura, verde4);
      intervalodedondevengo = p;
```

Ilustración 37 : Bloque insertar cuadrados

```
///                                                                                                    COLORES
////////////////////////////////////////////////////////////////////////////////////////////////////////////

#define verde1  0x0FE0
#define verde2  0x27E0
#define verde3  0x37E0
#define verde4  0x67E0
#define verde5  0x97E0
#define verde6  0xAFE0
#define verde7  0xCFE0
#define amarillo1  0xE7E0
#define amarillo2  0xFFE0
#define amarillo3  0xFF80//10
#define amarillo4  0xFEC0
#define amarillo5  0xFE60
#define naranja1  0xFDC0
#define naranja2  0xFD00
#define naranja3  0xFC60
#define naranja4  0xFBA0
#define naranja5  0xFAA0
#define naranja6  0xF9E0
#define naranja7  0xF920
#define rojo  0xF800
uint16_t colores;
```

Ilustración 38 : Colores cuadrados

En el nuevo caso, la barra, posee una función muy corta, cosa que ha permitido acortar de manera notoria la cantidad de líneas que poseía el programa inicialmente.

```
void  indica_depositoR(int numero)////recibes el numero a representar en la barra
{

  intervaloD = map(numero, 0, deposito_total_maximo, 0, -90); //usamos el intervalo negativo porque si no el movimiento seria de abajo a
  //usamos unos 90 pixeles de altura,
  if (deposito_total > 70.0) {
    colores = verde1;
  }
  if ((30.0 < deposito_total) && (deposito_total < 70.0)) {
    colores = naranja1;
  }
  if (deposito_total < 30.0) {
    colores = rojo;
  }

  tft.fillRect(430, 204, 40/*ancho*/, intervaloD + 92/*alto*/, TFT_WHITE); //usamos un cuadrado pero esta vez con altura positiva
  //para que vaya borrando el negro, el 92 es para borra dos pixeles que nos quedan  por ahi perdidos
  tft.fillRect(430, 295, 40/*ancho*/, intervaloD/*alto*/, colores); //usamos el intervalo del momento


}
```

4.1.3 GPS y GSM:

El GPS[12] a utilizar en este caso, es un NEO-6M-0-001[13], posee las 4 conexiones que nos hacen falta, **Vcc** que irá a **5V, GND** que irá a **Masa**, **Tx** que irá al **RX2** del Arduino Mega, y **Rx** que irá al **TX2** del respectivamente.

Ilustración 39 : GPS NEO6

Para poder obtener los datos del GPS, se pueden utilizar los comandos AT, sin embargo, de ellos obtendríamos una expresión algo ardua de entender, un ejemplo de ellos sería el siguiente:

```
$GPRMC,225446,A,4916.45,N,12311.12,W,000.5,054.7,191194,020.3,E*68
```

Donde su significado vendría dado por la siguiente Ilustración 40 : Significado información AT:

225446	Hora 22:54:46 UTC
A	Estado receptor A = OK
4916.45,N	Latitud 49° 16.45 min Norte
12311.12,W	Longitud 123°11.12 min Oeste
000.5	Velocidad 0.5 nudos
054.7	Curso 54.7°
191194	Fecha 19 Noviembre 1994
020.3,E	Variación magnética 20.3° East
*68	Checksum

Ilustración 40 : Significado información AT

Para interpretar la secuencia $GPRMC de forma sencilla disponemos de la librería TinyGPS, que nos permite disgregar esos datos en otros más entendibles y manejables para su posterior utilización.

El programa sería de la siguiente manera:

```
 #include <TinyGPS.h>

TinyGPS gps;

void setup() {
  // Open serial communications and wait for port to open:
  Serial.begin(9600);
  // set the data rate for the SoftwareSerial port
  Serial2.begin(9600);
  Serial.println();
}
void loop() {
  unsigned long start = millis();
  // Every 5 seconds we print an update
  while (millis() - start < 500) {
    if (Serial2.available()) {
      char c = Serial2.read();
      if (gps.encode(c)) {
      }
    }

  }
float flat,flon;
unsigned long age;
  gps.f_get_position(&flat, &flon, &age);
      Serial.print("LAT=");
      Serial.println(flat);

      Serial.print(" LON=");
      Serial.println(flon );
      Serial.print(" SAT=");
      Serial.println(gps.satellites());
  float numero=gps.f_speed_mph();
  Serial.print("KMH");
  Serial.println(numero);
}
```

Nos creamos un objeto de la clase TinyGPS llamado gps. Nos vamos a comunicar a través del puerto seria 2, puesto que el 1 ya está ocupado con las interrupciones del caudalímetro.

En el set up solo inicializamos los Serial(). Dentro del loop, para adquirir lo datos del gps, debe permanecer medio segundo o cuarto de segundo leyendo para adquirir el dato. Una vez leído, podemos adjudicar a la variable que queramos el valor de satélites, velocidad, longitud, etc, con solo igualarlo a gps.xxxx. En nuestro caso nos interesaba saber la longitud,

la latitud, los satélites y la velocidad en kilómetros/hora, aunque existen más variables que podemos incluir.

En el caso de enviar SMS, haremos uso de un dispositivo GSM, el Sim808. A diferencia que el GPS, éste dispositivo utilizará los comandos AT para poder operar sus funciones. Posee de nuevo al igual que su homólogo el GPS, un pin de 5V, uno de masa, un pin RX de lectura y un pin Tx de escritura, que deberemos conectar de manera cruzada con los pines Tx y Rx propios del Arduino al que vayan a ser conectados, es decir, El pin Rx del Sim808 con el Tx del Arduino y el Tx del Sim808 con el pin Rx del Arduino. En la sección de planos podremos ver como van conectados por lo que no supondrá un problema verlo.

4.1.4 Velocidad:

Para medir la velocidad ofrecida por la moto, existe un conexionado, en el cual tenemos acceso a los pulsos de onda cuadrada que ofrece, de salida 12V, los cuales debemos convertir a 4,5V (el Arduino no soporta más de 5V de entrada en los pines).

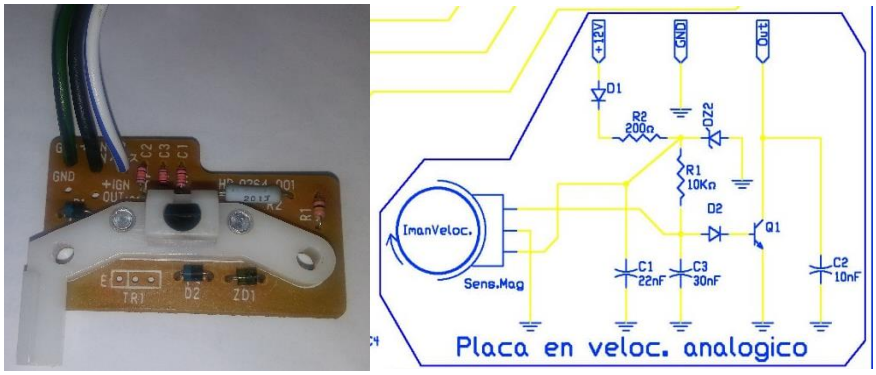


Ilustración 41 : Circuito Velocidad Moto

El circuito, a través de un sensor de efecto Hall, genera los pulsos provocando cambios de nivel en los 12V. Sin embargo, cuando se probó, resultó que el transistor que debía disparar dichos flancos de subida y bajada, no funcionaba correctamente. Se trataba de un transistor NPN con resistencias internas, por lo que se sustituyó por un NPN BHP57 y dos resistencias de 4,7K (así la ganancia es 1), ofreciendo un resultado considerablemente bueno.

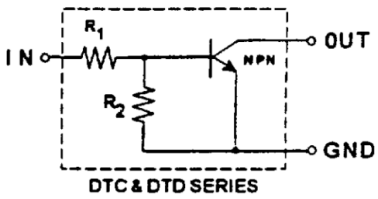


Ilustración 42 : Circuito transistor NPN

En el Ilustración 43 observamos que la salida (azul) reacciona a la entrada (amarilla) de manera inversa a la entrada debido al amplificador de ganancia -1.

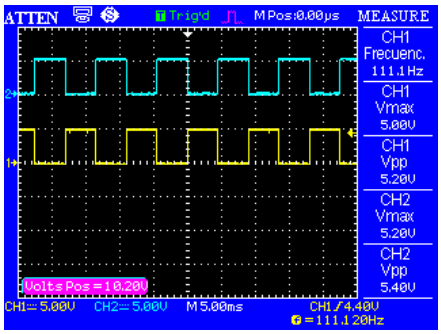


Ilustración 43 : Entrada (amarilla) y salida (azul) del transistor.

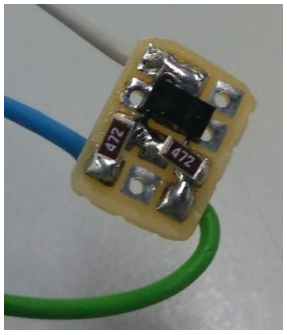


Ilustración 44 : Circuito de reparación del transistor dañado

En el Ilustración 43 observamos que la salida (azul) reacciona a la entrada (amarilla) de manera inversa a la entrada debido al amplificador de ganancia -1.

Para poder identificar los pulsos y transformarlos a velocidad, se realizó un experimento muy sencillo: con la motocicleta elevada, se le dio una vuelta completa a la rueda de delante, y se vieron los pulsos que daba en el osciloscopio. Así se pudo ver que una vuelta completa daba 12 pulsos. Después la motocicleta se puso e el suelo, y con marcas, se midió la distancia que recorre cuando la rueda gira una vuelta completa. Dicha distancia tuvo un valor de 2,15 metros de longitud, por lo que aproximadamente, cada pulso que nos da, significa que ha recorrido 18 centímetros. Por tanto, 18 centímetros por pulso nos darña la distancia recorrida. Como muestreamos cada medio segundo, para saber cuantos pulsos nos da por segundo, vemos que hay que multiplicar por dos, así tendríamos *cm/s*, por lo que para poder mostrar los km/h simplemente tendremos que hacer un cambio de unidades.

Finalmente nos queda que 1.296*numero de pulsos = Velocida (km/h)

4.1.5 Revoluciones:

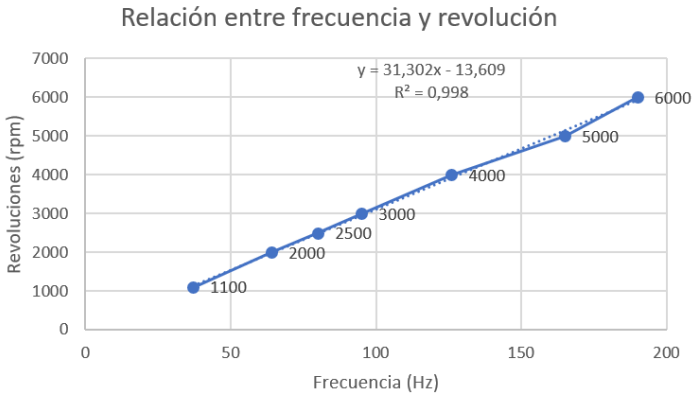
Para el caso de las revoluciones, dependiendo de la frecuencia de los pulsos que nos dé, adquirirá una revolución u otra. En la siguiente tabla tenemos datos obtenidos de manera experimental, los cuales relacionan, que revolución lleva la moto con la frecuencia dada.

Revoluciones (rpm)	Frecuencia (Hz)
1100	37
2000	64
2500	80
3000	95
4000	126
5000	175
6000	190

Tabla 1 : Revoluciones y frecuencias

Realizando el gráfico nos sale la ecuación que relacionará la frecuencia y la revolución:

Ilustración 45 : Relación de magnitudes



La el valor de R^2 es muy cercano a uno luego podemos dar por válida que existe una linealidad entre la frecuencia y las revoluciones.

Como aclaración, no se comentará todo el programa, se han expuesto algunos puntos como detalles, para ver el código, diríjase al apartado de Axeso, código.

4.1.6 Caudalímetro:

El caudalímetro es un elemento muy sencillo, tan solo cuenta con tres cables, uno de 5 V, otro de masa y otro que da pulsos conforme pasa el líquido por su interior. Cada pulso que se produzca, significará que han pasado tantos mililitros. Luego quiere decir, que cuando detecte un pulso, deberá restar esa cantidad de mililitros al depósito total y así poder llevar la cuenta de lo que nos queda en el depósito.

4.1.7 Temperatura Exterior

Existen muchas maneras de medir la temperatura, pero la que más cómoda que nos ha parecido, se trata de un sensor dht11[14], este sensor dispone de un procesador interno que realiza el proceso de medición, proporcionando la medición mediante una señal digital, por lo que resulta muy sencillo obtener la medición desde un microprocesador como Arduino.

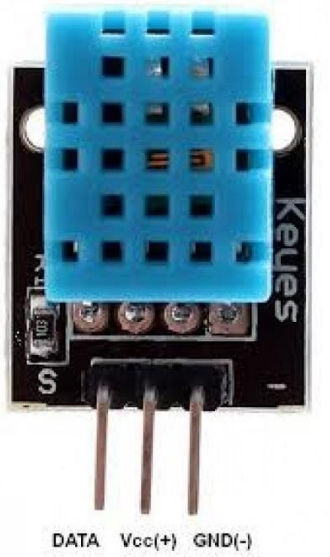


Ilustración 46 : Sensor de temperatura DHT11

Como aclaración, no se comentará todo el programa, se han expuesto algunos puntos como detalles, para ver el código, diríjase al apartado de Axeso, código.

Primero que todo, hay que aclarar que existen muchos modelos de DHT11 en el mercado, algunos vienen con 4 pines, otros con 3 como el de la Ilustración 46, que además incorporan una placa que añade la resistencia de 10K que necesita para su circuito.

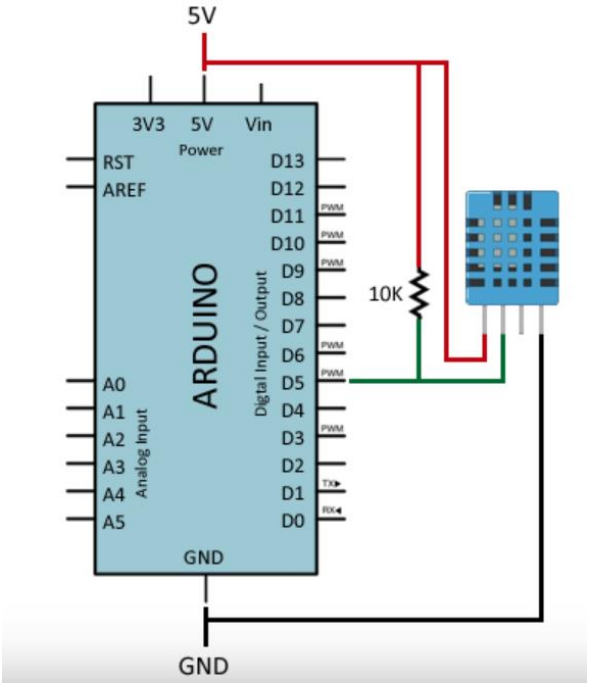


Ilustración 47 : Circuito sensor DHT11

Hacemos hincapié en esto, porque normalmente los pines siguen esta secuencia que vemos en la Ilustración 48, pero como vemos, en la Ilustración 46, la distribución es completamente otra. Por lo que un motivo de fallo puede ser que hayamos uno o varios pines de forma incorrecta si no nos hemos fijado en el datasheet que ofrezca el fabricante que haya realizado el sensor.

Las características del sensor se mostrará en el apartado de Anexos.

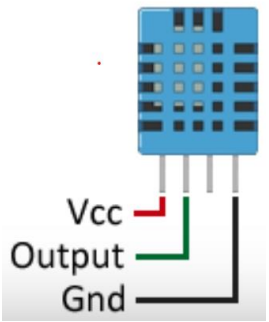


Ilustración 48 : Pines DHT11

4.1.8 Temperatura Motor

Si para la temperatura exterior hemos contado con un sensor cuya precisión no era su punto fuerte, pues la temperatura exterior no es algo que en principio nos sea de gran importancia, para medir la temperatura del motor hemos recurrido a un sensor ampliamente utilizado en la industria hoy en día, una PT100.

La PT100 es una RDT (Detector de Temperatura Resistivo), posee una resistencia de platino que a 0 grados centígrados se encuentra a 100Ω. Para fortalecer la consistencia de la PT100 se mete en una capsula de metal.

Ilustración 49 : PT100

Este sensor, a diferencia del sensor de temperatura ambiente, necesita de un acondicionamiento de señal, pues es un sensor pasivo que necesita de alimentación, además la variación de resistencia es muy poca, por lo que es necesario amplificar la señal para que el Arduino sea capaz de leer los voltajes.

El circuito para abordar este problema se encuentra en los planos del anexo.

Para ajustar la PT100 a nuestros intereses y como las temperaturas de la moto, nos interesa que la medición se realice de cara a altas temperaturas, los 25 ºC se considerarán como 0 voltios, y los 105 como 4 voltios. Para ello, hemos ajustado los valores de las resistencias en el laboratorio,

mediante el horno de calibración.

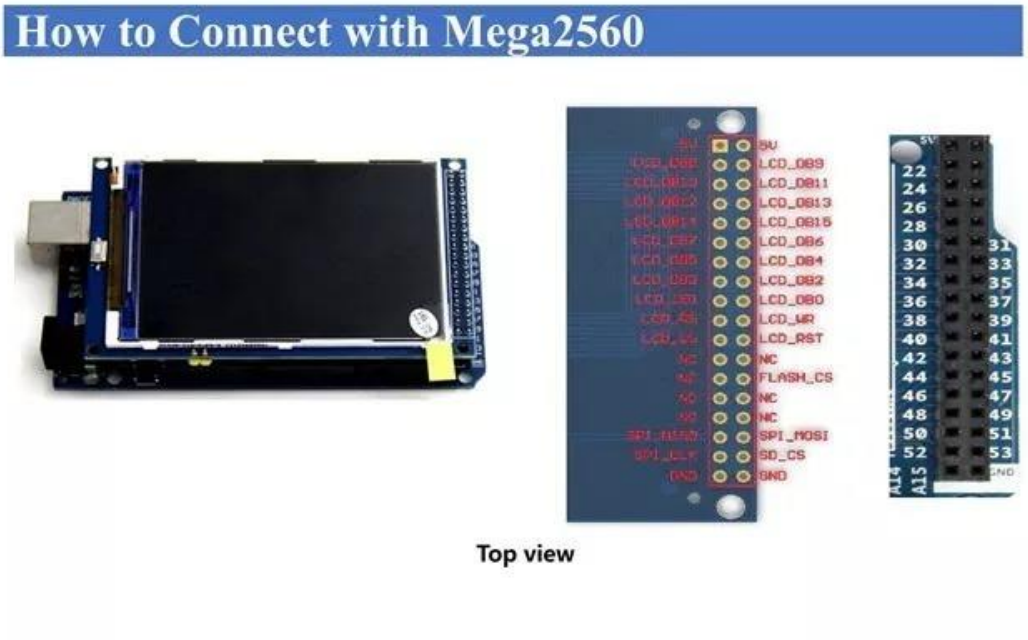
Ilustración 50 : Estudio de resistencias para ajustar PT100

El circuito para abordar este problema se encuentra en los planos del anexo.

Para ello colocamos el horno de calibración a 25 grados y ahí ajustamos para que la salida de nuestro circuito nos ofrezca 0 voltios. Luego nos fuimos a 105 grados centígrados y ajustamos la ganancia para que nos diera 4 voltios a la salida. Una vez alcanzamos los 105ºC, nos fuimos a 25 para comprobar que efectivamente nos daba aproximadamente 0 V. Dimos por buena la aproximación, y medimos las resistencias que nos permitían dicho ajuste. Como estábamos tratando con potenciómetros, e nuestra placa final serían resistencias fijas, por lo que para ser finos a la hora de ponerlas, decidimos colocar varias en serie para poseer una aproximación lo más exacta posible.

4.2 Planos

Como aclaración, no se comentará todo el programa, se han expuesto algunos puntos como detalles, para ver el código, diríjase al apartado de Axeso, código.



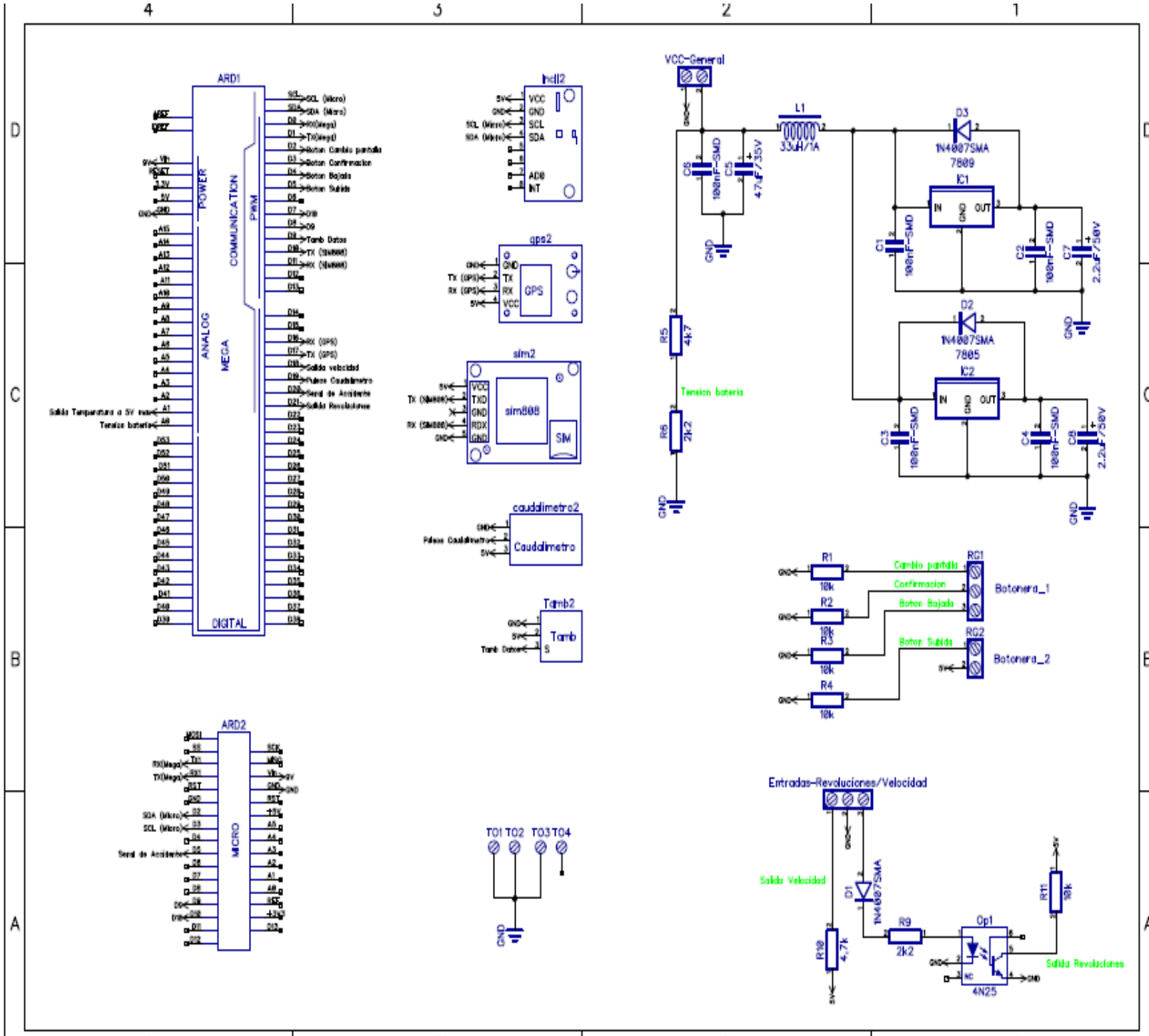
En este apartado, vamos a ver el conexionado del microcontrolador Arduino Mega y Micro, con el resto de componentes que van a formar parte del proyecto. En cuanto a la pantalla, no aparecerá en el esquema, debido a la inmensa cantidad de pines que posee. La pantalla ya viene diseñada de fábrica para encajar a la perfección con la placa Arduino Mega, por lo que, a nivel de plano,

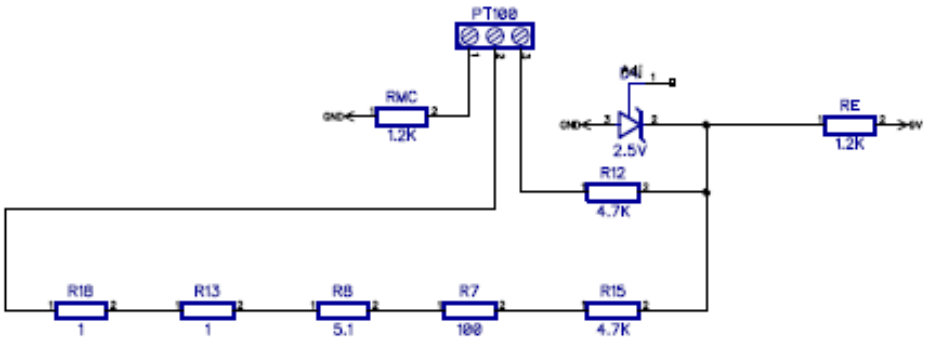
mostraremos unas imágenes para que su conexión pueda ser mejor entendida y no dé lugar a confusión.

Ilustración 52 : Pantalla conectada al Mega

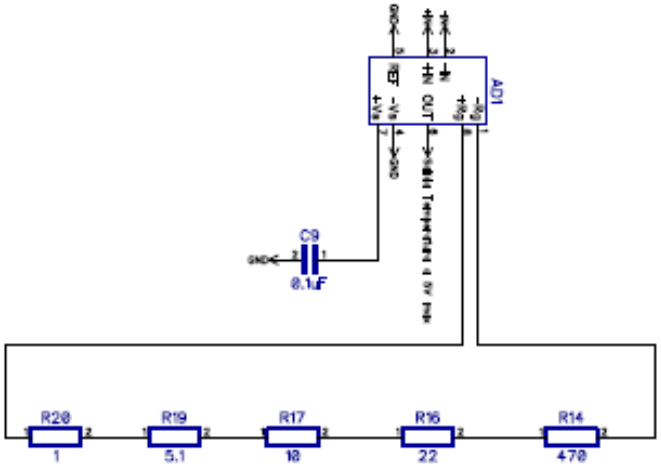
Ilustración 51 : Conexionado y pines de la pantalla

Sin embargo, el resto de componentes se han implementado en un esquemático, realizado en el programa gratuito Dip Trace, el cual nos permite realizar el pad de los componentes, y cohesionar en un mismo documento todos los componentes para realizar un esquemático y posteriormente convertirlo a PCB.


ARD1
POWER
COMMUNICATION
PWM
ANALOG
MEGA
DIGITAL
SCL (Micro)
SDA (Micro)
TX(Mega)
Boton Cambio pantalla
Boton Confirmacion
Boton Bajada
Boton Subida
Tamb Datos
TX (SIM808)
RX (SIM808)
RX (GPS)
TX (GPS)
Salida velocidad
Pulsos Cauddalimetro
Senal de Accidente
Salida Revoluciones
Salida Temperatura a 5V max
Tension bateria
hall2
VCC
GND
SCL
SDA
ADR
INT
gps2
GPS
TX
RX
sim2
sim808
SIM
VCC
TXD
GND
RDX
cauddalimetro2
Cauddalimetro
Tamb2
Tamb
VCC-General
L1
33uH/1A
D3
1N4007SMA
7809
IC1
IN
GND
OUT
C1
100nF-SMD
C2
100nF-SMD
C7
2.2uF/50V
D2
1N4007SMA
7805
IC2
C3
100nF-SMD
C4
100nF-SMD
C8
2.2uF/50V
C6
100nF-SMD
C5
47uF/35V
R5
4k7
R6
2k2
Tension bateria
R1
18k
R2
18k
R3
18k
R4
18k
Cambio pantalla
Confirmacion
Boton Bajada
Boton Subida
RG1
RG2
Botonera_1
Botonera_2
ARD2
MICRO
TO1 TO2 TO3 TO4
Entradas-Revoluciones/Velocidad
Salida Velocidad
D1
1N4007SMA
R9
2k2
R10
4,7k
Op1
4N25
R11
10k
Salida Revoluciones



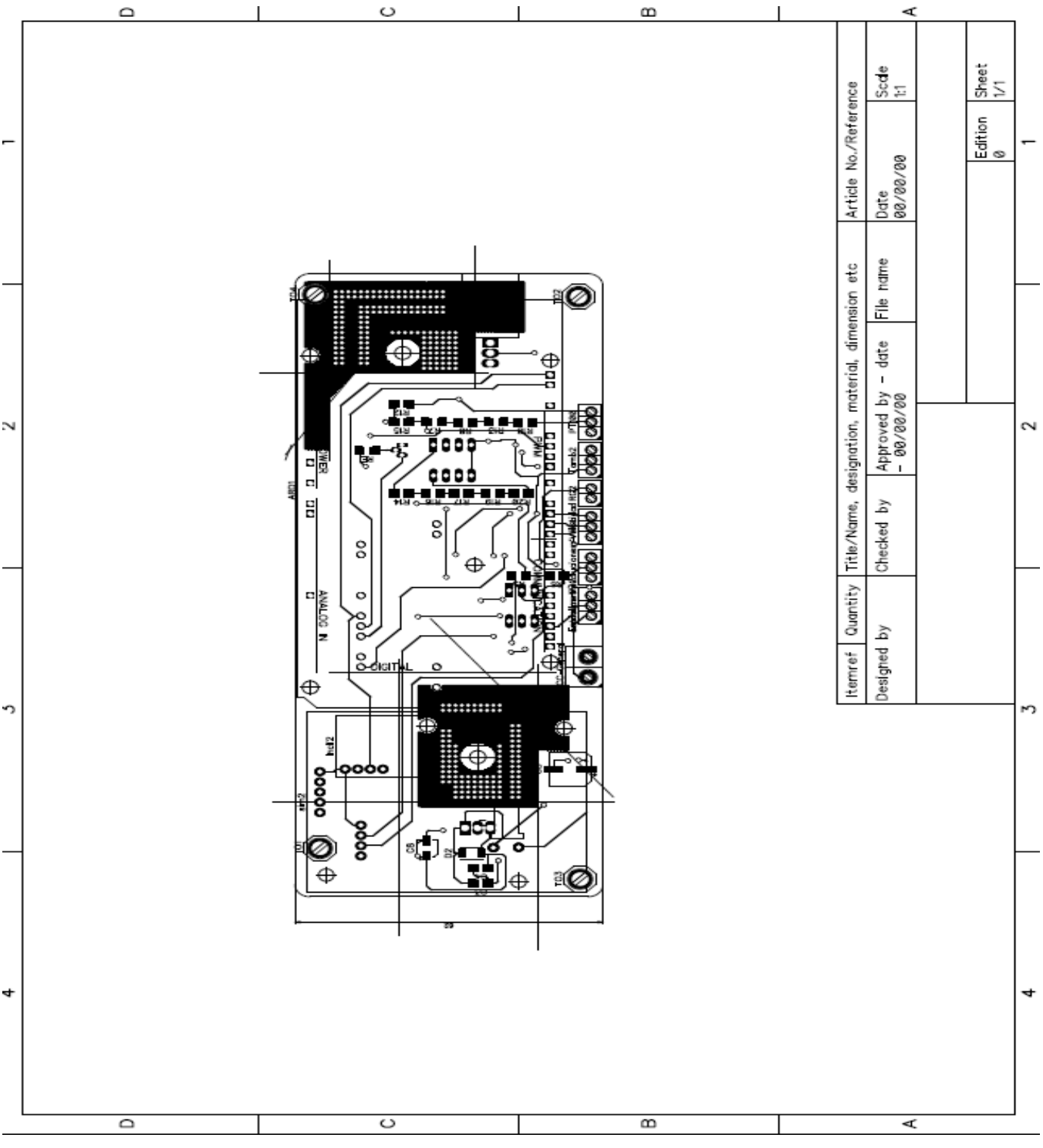
Resistencia de 170.1



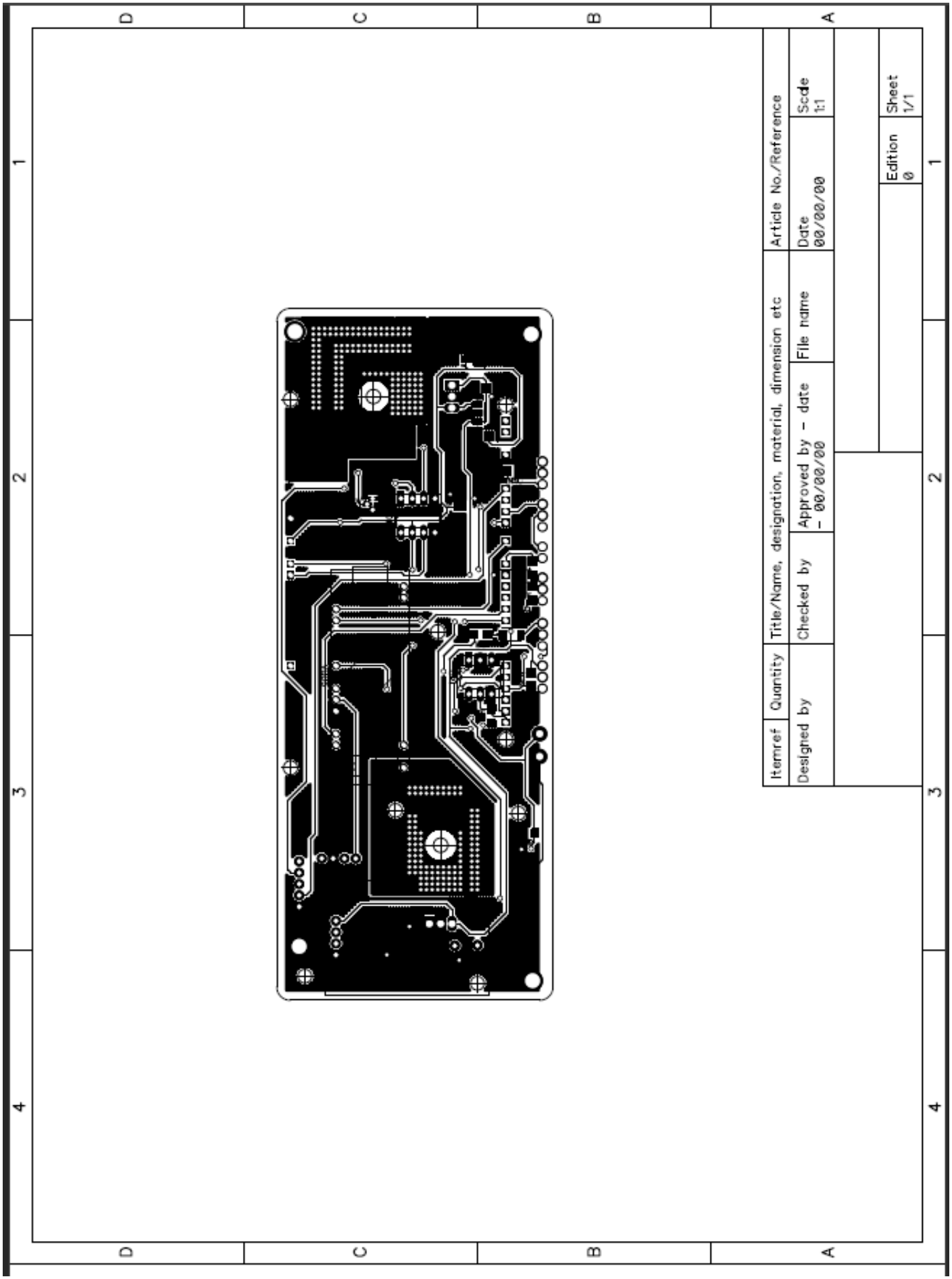
Resistencia de 508.1

Mediante este esquema, podremos saber en qué pin va conectado cada elemento. Sensores, microcontroladores, botones, elementos pasivos, todo ello quedará reflejado en el esquemático para que nada quede al azar o al criterio de quien lo monte.

El aspecto del PCB en la vista top quedaría de la siguiente manera:



El aspecto del PCB en la vista bottom quedaría de la siguiente manera:



4.3 Presupuestos

Todo desarrollo de proyecto conlleva inevitablemente un coste que es necesario cubrir, materiales, equipamiento, salarios etc. Por ello se ha elaborado una tabla con todos los costes para realizar el proyecto.

Concepto	Cantidad	Precio*
MPU-6065 (inclinómetro)	1	7.99€
GPS NEO 6M	1	4.61€
Sim808	1	13.96€
Caudalímetro	1	
Pantalla 480x320 TFT	1	~7€
Arduino Mega	1	13.99€
Arduino Micro	1	2.20€

Tabla 2 : Presupuesto

Todos los artículos incluyen IVA.

4.4 DataSheets

4.4.1 Datasheet AD623

ANALOG DEVICES

Single Supply, Rail-to-Rail, Low Cost Instrumentation Amplifier

AD623

FEATURES
Easy to Use
Higher Performance than Discrete Design
Single and Dual Supply Operation
Rail-to-Rail Output Swing
Input Voltage Range Extends 150 mV Below Ground (Single Supply)
Low Power, 575 µA Max Supply Current
Gain Set with One External Resistor
Gain Range 1 (No Resistor) to 1,000

HIGH ACCURACY DC PERFORMANCE
0.1% Gain Accuracy (G = 1)
0.35% Gain Accuracy (G > 1)
25 ppm Gain Drift (G = 1)
200 µV Max Input Offset Voltage (AD623A)
2 µV/°C Max Input Offset Drift (AD623A)
100 µV Max Input Offset Voltage (AD623B)
1 µV/°C Max Input Offset Drift (AD623B)
25 nA Max Input Bias Current

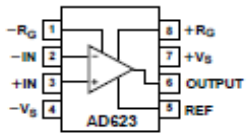
NOISE
35 nV/√Hz RTI Noise @ 1 kHz (G = 1)

EXCELLENT AC SPECIFICATIONS
90 dB Min CMRR (G = 10); 84 dB Min CMRR (G = 5) (@ 60 Hz, 1K Source Imbalance)
800 kHz Bandwidth (G = 1)
20 µs Settling Time to 0.01% (G = 10)

APPLICATIONS
Low Power Medical Instrumentation
Transducer Interface
Thermocouple Amplifier
Industrial Process Controls
Difference Amplifier
Low Power Data Acquisition

CONNECTION DIAGRAM

8-Lead Plastic DIP (N), SOIC (R) and µSOIC (RM) Packages



PRODUCT DESCRIPTION

The AD623 is an integrated single supply instrumentation amplifier that delivers rail-to-rail output swing on a single supply (+3 V to +12 V supplies). The AD623 offers superior user flexibility by allowing single gain set resistor programming, and conforming to the 8-lead industry standard pinout configuration. With no external resistor, the AD623 is configured for unity gain (G = 1) and with an external resistor, the AD623 can be programmed for gains up to 1,000.

The AD623 holds errors to a minimum by providing superior AC CMRR that increases with increasing gain. Line noise, as well as line harmonics, will be rejected since the CMRR remains constant up to 200 Hz. The AD623 has a wide input common-mode range and can amplify signals that have a common-mode voltage 150 mV below ground. Although the design of the AD623 has been optimized to operate from a single supply, the AD623 still provides superior performance when operated from a dual voltage supply (±2.5 V to ±6.0 V).

Low power consumption (1.5 mW at 3 V), wide supply voltage range, and rail-to-rail output swing make the AD623 ideal for battery powered applications. The rail-to-rail output stage maximizes the dynamic range when operating from low supply voltages. The AD623 replaces discrete instrumentation amplifier designs and offers superior linearity, temperature stability and reliability in a minimum of space. Until the AD623, this level of instrumentation amplifier performance has not been achieved.

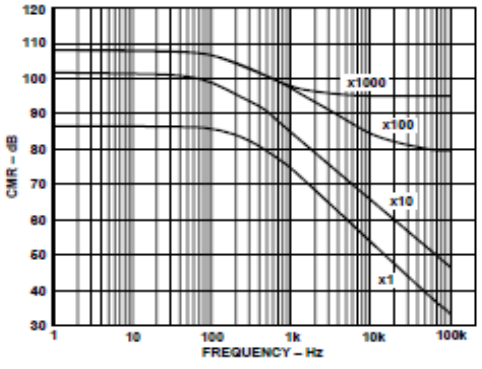


Figure 1. CMR vs. Frequency, +5 V_S, 0 V_S

REV. C

One Technology Way, P.O. Box 9106, Norwood, MA 02062-9106, U.S.A.
Tel: 781/329-4700 World Wide Web Site: http://www.analog.com
Fax: 781/326-8703

4.4.2 DataSheet MPU 6050

5 Features

5.1 Gyroscope Features

The triple-axis MEMS gyroscope in the MPU-60X0 includes a wide range of features:

- Digital-output X-, Y-, and Z-Axis angular rate sensors (gyroscopes) with a user-programmable full-scale range of ±250, ±500, ±1000, and ±2000°/sec
- External sync signal connected to the FSYNC pin supports image, video and GPS synchronization
- Integrated 16-bit ADCs enable simultaneous sampling of gyros
- Enhanced bias and sensitivity temperature stability reduces the need for user calibration
- Improved low-frequency noise performance
- Digitally-programmable low-pass filter
- Gyroscope operating current: 3.6mA
- Standby current: 5µA
- Factory calibrated sensitivity scale factor
- User self-test

5.2 Accelerometer Features

The triple-axis MEMS accelerometer in MPU-60X0 includes a wide range of features:

- Digital-output triple-axis accelerometer with a programmable full scale range of ±2*g*, ±4*g*, ±8*g* and ±16*g*
- Integrated 16-bit ADCs enable simultaneous sampling of accelerometers while requiring no external multiplexer
- Accelerometer normal operating current: 500µA
- Low power accelerometer mode current: 10µA at 1.25Hz, 20µA at 5Hz, 60µA at 20Hz, 110µA at 40Hz
- Orientation detection and signaling
- Tap detection
- User-programmable interrupts
- High-G interrupt
- User self-test

5.3 Additional Features

The MPU-60X0 includes the following additional features:

- 9-Axis MotionFusion by the on-chip Digital Motion Processor (DMP)
- Auxiliary master I^2C bus for reading data from external sensors (e.g., magnetometer)
- 3.9mA operating current when all 6 motion sensing axes and the DMP are enabled
- VDD supply voltage range of 2.375V-3.46V
- Flexible VLOGIC reference voltage supports multiple I^2C interface voltages (MPU-6050 only)
- Smallest and thinnest QFN package for portable devices: 4x4x0.9mm
- Minimal cross-axis sensitivity between the accelerometer and gyroscope axes
- 1024 byte FIFO buffer reduces power consumption by allowing host processor to read the data in bursts and then go into a low-power mode as the MPU collects more data
- Digital-output temperature sensor
- User-programmable digital filters for gyroscope, accelerometer, and temp sensor
- 10,000 *g* shock tolerant
- 400kHz Fast Mode I^2C for communicating with all registers
- 1MHz SPI serial interface for communicating with all registers (MPU-6000 only)
- 20MHz SPI serial interface for reading sensor and interrupt registers (MPU-6000 only)

NEO-6 series

Versatile u-blox 6 GPS modules

Highlights

- UART, USB, DDC (I²C compliant) and SPI interfaces
- Available in Crystal and TCXO versions
- Onboard RTC crystal for faster warm and hot starts
- 1.8 V and 3.0 V variants

NEO-6:
12.2 x 16.0 x 2.4 mm

Features

- u-blox 6 position engine:
  - Navigate down to –162 dBm and –148 dBm coldstart
  - Faster acquisition with AssistNow Autonomous
  - Configurable power management
  - Hybrid GPS/SBAS engine (WAAS, EGNOS, MSAS)
  - Anti-jamming technology
- Simple integration with u-blox wireless modules
- A-GPS: AssistNow Online and AssistNow Offline services, OMA SUPL compliant
- Backward compatible (hardware and firmware); easy migration from NEO-5 family or NEO-4S
- LCC package for reliable and cost effective manufacturing
- Compatible with u-blox GPS Solution for Android
- Based on GPS chips qualified according to AEC-Q100
- Manufactured in ISO/TS 16949 certified production sites
- Qualified according to ISO 16750

Product description

The NEO-6 module series brings the high performance of the u-blox 6 position engine to the miniature NEO form factor. u-blox 6 has been designed with low power consumption and low costs in mind. Intelligent power management is a breakthrough for low-power applications. These receivers combine a high level of integration capability with flexible connectivity options in a small package. This makes them perfectly suited for mass-market end products with strict size and cost requirements. The DDC interface provides connectivity and enables synergies with u-blox LEON and LISA wireless modules.

All NEO-6 modules are manufactured in ISO/TS 16949 certified sites. Each module is tested and inspected during production. The modules are qualified according to ISO 16750 - Environmental conditions and electrical testing for electrical and electronic equipment for road vehicles.

Product selector

Model	Type				Supply		Interfaces				Features						
	Standalone GPS	Standalone GLONASS	Timing & Raw Data	Dead Reckoning	1.75 V – 2.0 V	2.7 V – 3.6 V	UART	USB	SPI	DDC (I²C compliant)	Programmable (Flash) FW update	Oscillator	RTC crystal	Antenna supply and supervisor	Configuration pins	Timepulse	External interrupt / Wakeup
NEO-6G	•				•		•	•	•	•		T	•	o	3	1	•
NEO-6Q	•					•	•	•	•	•		T	•	o	3	1	•
NEO-6M	•					•	•	•	•	•		C	•	o	3	1	•

o = requires external components and integration on application processor

C = Crystal / T = TCXO

Receiver performance data

Receiver type	50-channel u-blox 6 engine GPS L1 C/A code SBAS: WAAS, EGNOS, MSAS		
Navigation update rate	up to 5 Hz		
Accuracy[1]	Position SBAS	2.5 m CEP 2.0 m CEP	
Acquisition[1]		NEO-6G/Q	NEO-6M
	Cold starts: Aided starts[2]: Hot starts:	26 s 1 s 1 s	27 s < 3 s 1 s
Sensitivity[3]		NEO-6G/Q	NEO-6M
	Tracking: Cold starts: Hot starts:	–162 dBm –148 dBm –157 dBm	–161 dBm –147 dBm –156 dBm

[1] All SV @ –130 dBm
[2] Dependent on aiding data connection speed and latency
[3] Demonstrated with a good active antenna

Electrical data

Power supply	2.7 V – 3.6 V (NEO-6Q/6M) 1.75 V – 2.0 V (NEO-6G)
Power consumption	111 mW @ 3.0 V (continuous) 33 mW @ 3.0 V Power Save Mode (1 Hz) 68 mW @ 1.8 V (continuous) 22 mW @ 1.8 V Power Save Mode (1 Hz)
Backup power	1.4 V – 3.6 V, 22 µA
Supported antennas	Active and passive

Interfaces

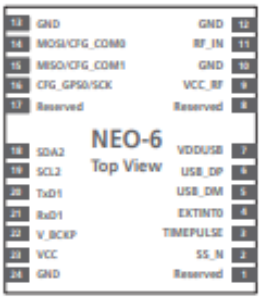
Serial interfaces	1 UART 1 USB V2.0 full speed 12 Mbit/s 1 DDC (I²C compliant) 1 SPI	
Digital I/O	Configurable timepulse 1 EXTINT input for Wakeup	
Serial and I/O	Voltages	2.7 – 3.6 V (NEO-6Q/6M) 1.75 – 2.0 V (NEO-6G)
Timepulse	Configurable	0.25 Hz to 1 kHz
Protocols	NMEA, UBX binary, RTCM	

Specification applies to FW 7

Package

24 pin LCC (Leadless Chip Carrier): 12.2 x 16.0 x 2.4 mm, 1.6 g

Pinout



Environmental data, quality & reliability

Operating temp.	–40° C to 85° C
Storage temp.	–40° C to 85° C

RoHS compliant (lead-free)

Qualification according to ISO 16750

Manufactured in ISO/TS 16949 certified production sites

Support products

u-blox 6 Evaluation Kits:

Easy-to-use kits to get familiar with u-blox 6 positioning technology, evaluate functionality, and visualize GPS performance.

EVK-6H:	u-blox 6 Evaluation Kit with TCXO, suitable for NEO-6G, NEO-6Q
EVK-6P:	u-blox 6 Evaluation Kit with crystal, suitable for NEO-6M

Ordering information

NEO-6G-0	u-blox 6 GPS Module, 1.8V, TCXO, 12x16mm, 250 pcs/reel
NEO-6M-0	u-blox 6 GPS Module, 12x16mm, 250 pcs/reel
NEO-6Q-0	u-blox 6 GPS Module, TCXO, 12x16mm, 250 pcs/reel

Available as samples and tape on reel (250 pieces)

Contact us

HQ Switzerland
+41 44 722 7444
info@u-blox.com

EMEA
+41 44 722 7444
info@u-blox.com

Americas
+1 703 483 3180
info_us@u-blox.com

APAC – Singapore
+65 6734 3811
info_ap@u-blox.com

China
+86 10 68 133 545
info_cn@u-blox.com

Japan
+81 3 5775 3850
info_jp@u-blox.com

Korea
+82 2 542 0861
info_kr@u-blox.com

Taiwan
+886 2 2657 1090
info_tw@u-blox.com

4.4.4 DataSheet SIM808

1 Introduction

This document describes SIM808 hardware interface in great detail. This document can help user to quickly understand SIM808 interface specifications, electrical and mechanical details. With the help of this document and other SIM808 application notes, user guide, users can use SIM808 to design various applications quickly.

2 SIM808 Overview

Designed for global market, SIM808 is integrated with a high performance GSM/GPRS engine, a GPS engine and a BT engine. The GSM/GPRS engine is a quad-band GSM/GPRS module that works on frequencies GSM 850MHz, EGSM 900MHz, DCS 1800MHz and PCS 1900MHz. SIM808 features GPRS multi-slot class 12/ class 10 (optional) and supports the GPRS coding schemes CS-1, CS-2, CS-3 and CS-4. The GPS solution offers best-in-class acquisition and tracing sensitivity, Time-To-First-Fix (TTFF) and accuracy.

With a tiny configuration of 24*24*2.6mm, SIM808 can meet almost all the space requirements in user applications, such as M2M, smart phone, PDA, tracker and other mobile devices.

SIM808 has 68 SMT pads, and provides all hardware interfaces between the module and customers' boards.

- Support 4*4*2 keypads.
- One full modem serial port.
- One USB, the USB interfaces can debug, download software.
- Audio channels which include a microphone input and a receiver output.
- One SIM card interface.
- Charging interface.
- Programmable general purpose input and output.
- Support Bluetooth function.
- Support PWM and ADC.
- PCM/SPI/SD card interface, only one function can be accessed synchronously. (Default function is PCM).

SIM808 is designed with power saving technique so that the current consumption is as low as 1mA in sleep mode (GPS engine is powered down).
SIM808 integrates TCP/IP protocol and extended TCP/IP AT commands which are very useful for data transfer applications. For details about TCP/IP applications, please refer to *document [2]*.

2.1 SIM808 Key Features

Table 1: SIM808 GSM/GPRS engine key features

Feature	Implementation
Power supply	3.4V ~ 4.4V
Power saving	Typical power consumption in sleep mode is 1mA (BS-PA-MFRMS=9, GPS engine is powered down)
Charging	Supports charging control for Li-Ion battery
Frequency bands	• SIM808 Quad-band: GSM 850, EGSM 900, DCS 1800, PCS 1900. SIM808

	can search the 4 frequency bands automatically. The frequency bands also can be set by AT command "AT+CBAND". For details, please refer to *document [1]*. ● Compliant to GSM Phase 2/2+
Transmitting power	● Class 4 (2W) at GSM 850 and EGSM 900 ● Class 1 (1W) at DCS 1800 and PCS 1900
GPRS connectivity	● GPRS multi-slot class 12（default） ● GPRS multi-slot class 1~12 (optional)
Temperature range	● Normal operation: -40℃ ~ +85℃ ● Storage temperature -45℃~ +90℃
Data GPRS	● GPRS data downlink transfer: max. 85.6 kbps ● GPRS data uplink transfer: max. 85.6 kbps ● Coding scheme: CS-1, CS-2, CS-3 and CS-4 ● PAP protocol for PPP connect ● Integrate the TCP/IP protocol. ● Support Packet Broadcast Control Channel (PBCCH) ● CSD transmission rates：2.4，4.8，9.6，14.4 kbps
CSD	● Support CSD transmission
USSD	● Unstructured Supplementary Services Data (USSD) support
SMS	● MT, MO, CB, Text and PDU mode ● SMS storage: SIM card
SIM interface	Support SIM card: 1.8V, 3V
External antenna	Antenna pad
Audio features	Speech codec modes: ● Half Rate (ETS 06.20) ● Full Rate (ETS 06.10) ● Enhanced Full Rate (ETS 06.50 / 06.60 / 06.80) ● Adaptive multi rate (AMR) ● Echo Cancellation ● Noise Suppression
Serial port and USB interface	**Serial port:** ● Full modem interface with status and control lines, unbalanced, asynchronous. ● 1200bps to 115200bps. ● Can be used for AT commands or data stream. ● Support RTS/CTS hardware handshake and software ON/OFF flow control. ● Multiplex ability according to GSM 07.10 Multiplexer Protocol. ● Autobauding supports baud rate from 1200 bps to 115200bps. **USB interface:** ● Can be used as debugging and firmware upgrading.
Phonebook management	Support phonebook types: SM, FD, LD, RC, ON, MC.
SIM application toolkit	GSM 11.14 Release 99
Real time clock	Support RTC
Alarm function	Can be set by AT command
Physical characteristics	Size: 24*24*2.6mm Weight: 3.5g

4.4.5 DataSheet BCX56

BCP56; BCX56; BC56PA

80 V, 1 A NPN medium power transistors

Rev. 9 — 25 October 2011 **Product data sheet**

1. Product profile

1.1 General description

NPN medium power transistor series in Surface-Mounted Device (SMD) plastic packages.

Table 1. Product overview

Type number[1]	Package			PNP complement
	Nexperia	JEITA	JEDEC	
BCP56	SOT223	SC-73	-	BCP53
BCX56	SOT89	SC-62	TO-243	BCX53
BC56PA	SOT1061	-	-	BC53PA

[1] Valid for all available selection groups.

1.2 Features and benefits

- High current
- Three current gain selections
- High power dissipation capability
- Exposed heatsink for excellent thermal and electrical conductivity (SOT89, SOT1061)
- Leadless very small SMD plastic package with medium power capability (SOT1061)
- AEC-Q101 qualified

1.3 Applications

- Linear voltage regulators
- Low-side switches
- Battery-driven devices
- Power management
- MOSFET drivers
- Amplifiers

1.4 Quick reference data

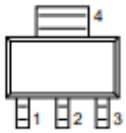
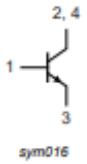
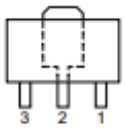
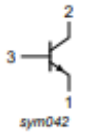
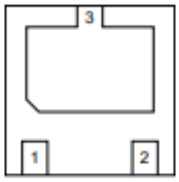
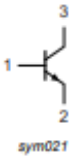
Table 2. Quick reference data

Symbol	Parameter	Conditions		Min	Typ	Max	Unit
V_{CEO}	collector-emitter voltage	open base		-	-	80	V
I_C	collector current			-	-	1	A
I_{CM}	peak collector current	single pulse; $t_p \leq 1$ ms		-	-	2	A
h_{FE}	DC current gain	$V_{CE} = 2$ V; $I_C = 150$ mA	[1]	63	-	250	
	h_{FE} selection -10	$V_{CE} = 2$ V; $I_C = 150$ mA	[1]	63	-	160	
	h_{FE} selection -16	$V_{CE} = 2$ V; $I_C = 150$ mA	[1]	100	-	250	

[1] Pulse test: $t_p \leq 300$ µs; $\delta = 0.02$.

2. Pinning information

Table 3. Pinning

Pin	Description	Simplified outline	Graphic symbol
SOT223			
1	base		 sym016
2	collector		
3	emitter		
4	collector		
SOT89			
1	emitter		 sym042
2	collector		
3	base		
SOT1061			
1	base	 Transparent top view	 sym021
2	emitter		
3	collector		

3. Ordering information

Table 4. Ordering information

Type number[1]	Package		
	Name	Description	Version
BCP56	SC-73	plastic surface-mounted package with increased heatsink; 4 leads	SOT223
BCX56	SC-62	plastic surface-mounted package; exposed die pad for good heat transfer; 3 leads	SOT89
BC56PA	HUSON3	plastic thermal enhanced ultra thin small outline package; no leads; 3 terminals; body 2 × 2 × 0.65 mm	SOT1061

[1] Valid for all available selection groups.

4.4.6 DataSheet Arduino Mega

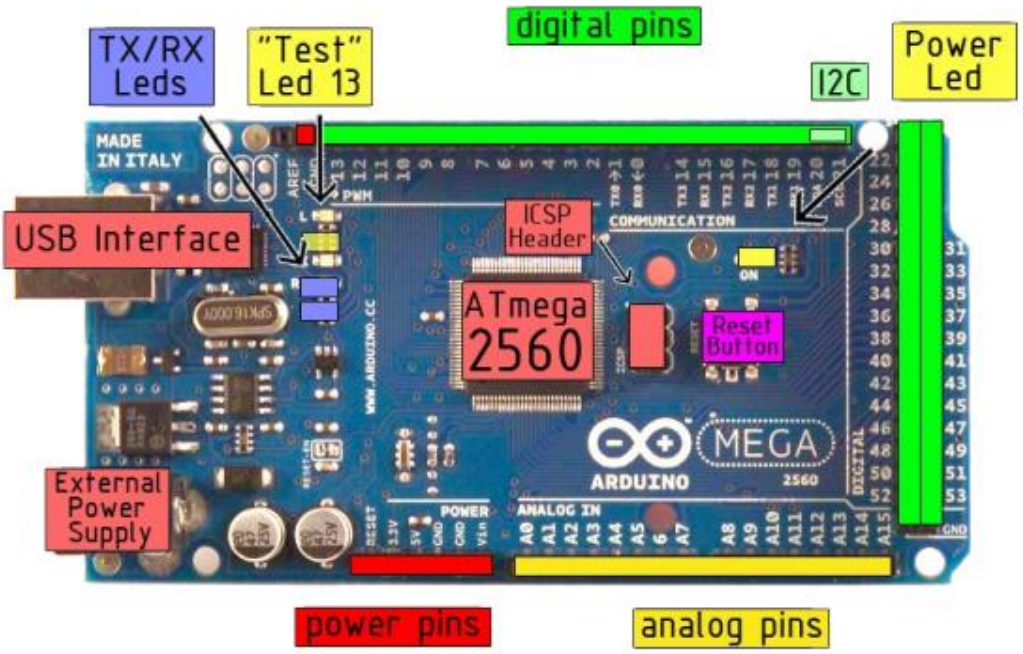
Technical Specification

EAGLE files: arduino-mega2560-reference-design.zip Schematic: arduino-mega2560-schematic.pdf

Summary

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	54 (of which 14 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz

the board



Power

The Arduino Mega2560 can be powered via the USB connection or with an external power supply. The power source is selected automatically. External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.

The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The Mega2560 differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega8U2 programmed as a USB-to-serial converter.

The power pins are as follows:

- **VIN.** The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- **5V.** The regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5V supply.
- **3V3.** A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- **GND.** Ground pins.

Memory

The ATmega2560 has 256 KB of flash memory for storing code (of which 8 KB is used for the bootloader), 8 KB of SRAM and 4 KB of EEPROM (which can be read and written with the EEPROM library).

Input and Output

Each of the 54 digital pins on the Mega can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

- **Serial: 0 (RX) and 1 (TX); Serial 1: 19 (RX) and 18 (TX); Serial 2: 17 (RX) and 16 (TX); Serial 3: 15 (RX) and 14 (TX).** Used to receive (RX) and transmit (TX) TTL serial data. Pins 0 and 1 are also connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip .
- **External Interrupts: 2 (interrupt 0), 3 (interrupt 1), 18 (interrupt 5), 19 (interrupt 4), 20 (interrupt 3), and 21 (interrupt 2).** These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.
- **PWM: 0 to 13.** Provide 8-bit PWM output with the analogWrite() function.
- **SPI: 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS).** These pins support SPI communication, which, although provided by the underlying hardware, is not currently included in the Arduino language. The SPI pins are also broken out on the ICSP header, which is physically compatible with the Duemilanove and Diecimila.
- **LED: 13.** There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.
- **I²C: 20 (SDA) and 21 (SCL).** Support I²C (TWI) communication using the Wire library (documentation on the Wiring website). Note that these pins are not in the same location as the I²C pins on the Duemilanove.

The Mega2560 has 16 analog inputs, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and analogReference() function.

There are a couple of other pins on the board:

- **AREF.** Reference voltage for the analog inputs. Used with analogReference().
- **Reset.** Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

Communication

The Arduino Mega2560 has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega2560 provides four hardware UARTs for TTL (5V) serial communication. An ATmega8U2 on the board channels one of these over USB and provides a virtual com port to software on the computer (Windows machines will need a .inf file, but OSX and Linux machines will recognize the board as a COM port automatically. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the board. The RX and TX LEDs on the board will flash when data is being transmitted via the ATmega8U2 chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

A SoftwareSerial library allows for serial communication on any of the Mega's digital pins.

The ATmega2560 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the documentation on the Wiring website for details. To use the SPI communication, please see the ATmega2560 datasheet.

Programming

The Arduino Mega2560 can be programmed with the Arduino software (download). For details, see the reference and tutorials.

The Atmega2560 on the Arduino Mega comes preburned with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol (reference, C header files).

You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header; see these instructions for details.

4.4.7 DataSheet Arduino Micro

Arduino Micro

A000053

Arduino Micro Front *Arduino Micro Rear*

Overview

The Arduino Micro is a microcontroller board based on the ATmega32u4 (datasheet), developed in conjunction with Adafruit. It has 20 digital input/output pins (of which 7 can be used as PWM outputs and 12 as analog inputs), a 16 MHz crystal oscillator, a micro USB connection, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a micro USB cable to get started. It has a form factor that enables it to be easily placed on a breadboard. The Micro is similar to the Arduino Leonardo in that the ATmega32u4 has built-in USB communication, eliminating the need for a secondary processor. This allows the Micro to appear to a connected computer as a mouse and keyboard, in addition to a virtual (CDC) serial / COM port. It also has other implications for the behavior of the board; these are detailed on the getting started page.

Summary

Microcontroller	ATmega32u4
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	20
PWM Channels	7
Analog Input Channels	12
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega32u4) of which 4 KB used by bootloader
SRAM	2.5 KB (ATmega32u4)
EEPROM	1 KB (ATmega32u4)
Clock Speed	16 MHz

Schematic & Reference Design

EAGLE files: arduino-micro-reference-design.zip
Schematic: arduino-micro-schematic-rev3b.pdf

Power

The Arduino Micro can be powered via the micro USB connection or with an external power supply. The power source is selected automatically.
External (non-USB) power can come either from a DC power supply or battery. Leads from a battery or DC power supply can be connected to the Gnd and Vin pins.

4.5 Código

```
////////////////////////////////////////////////////////////////
//////////////GPS
////////////////////////////////////////////////////////////////
//#include <SoftwareSerial.h>
#include <TinyGPS.h>
#include <EEPROM.h>

//SoftwareSerial mySerial(4, 3); // RX, TX
TinyGPS gps;

void gpsdump(TinyGPS &gps);
float flat, flon;
unsigned long age;
void printFloat(double f, int digits = 2);
////////////////////////////////////////////////////////////////
//////////////////INCLINOMETRO
////////////////////////////////////////////////////////////////
#include "I2Cdev.h"
#include "MPU6050.h"
#include "Wire.h"

// La dirección del MPU6050 puede ser 0x68 o 0x69, dependiendo
// del estado de AD0. Si no se especifica, 0x68 estará implicito
MPU6050 sensor;

// Valores RAW (sin procesar) del acelerometro y giroscopio en los ejes x,y,z
int ax, ay, az;
int gx, gy, gz;
////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
//////////////////TEMPERATURA EXTERIOR
////////////////////////////////////////////////////////////////

#include "DHT.h"
#define DHTTYPE DHT11
const int DHTPin = 9;
DHT dht(DHTPin, DHTTYPE);
float h;
float hprevias = 0;
////////////////////////////////////////////////////////////////////////
#include <TFT_HX8357.h>
#include <UTFT.h>
TFT_HX8357 tft = TFT_HX8357();
int i  = 100;
#include <SdFat.h>
#include <SPI.h>
#include <UTFT_SdRaw.h>
#define SD_CHIP_SELECT 53
```

```
int caudalimetro = 19;
int velocimetro = 18;
int cuentarevoluciones = 21;
SdFat sd;
UTFT myGLCD(ILI9341_16, 38, 39, 40, 41);
UTFT_SdRaw myFiles(&myGLCD);
//bool gps=true;
int numeromax = 120;
int numeromaxR = 100;
int numeromaxV = 15;
int numeromaxD = 100; //variables del random que no valdran para nada
int numeromin = 0; //variables del random que no valdran para nada

int page = 3;//pagina primera a la que entra por defecto

////////////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////
////                                                  FUNCION MOVIMIENTO
////////////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////
bool entre = true; //permite el paso a la funcion movimiento que, de tal manera que la barra no se
repite cada vez y siempre está activa en todas las pantallas
bool altohastaquesuelteslasubidaMov = true; //controla el estado de la pulsacion y evita revotes
bool altohastaquesuelteslasubidaCon = true; //controla el estado de la pulsacion y evita revotes
bool paginacargada2 = true; //indica que la pagina ya se cargó
bool paginacargada3 = true;
bool paginacargada4 = true;
int satelitecomprovacionprevias = 0;
int satelitecomprovacion = 0;
int Mancho = 69;
int Malto = 320;
bool entre1 = true;
bool entre2 = true;
bool entre3 = true;
bool entre4 = true;
bool entre5 = true;
int nuevahora = 0;
int minutosi = 0;
int nuevahoraprevias = 0;
int minutosiprevias = 0;
////////////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////
////                                                  FUNCION VECTORDETIEMPO
////////////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////
// bool entre=true;//permite el paso a la funcion movimiento que, de tal manera que la barra no se
repite cada vez y siempre está activa en todas las pantallas
bool altohastaquesuelteslabajada = true; //controla el estado de la pulsacion y evita revotes
bool filtro = true;
bool entreporprimeravezalafuncionvectordetiempo = false;
bool yapuedesreogereltiempo = false;
bool entradassucesivas = false;
```

```
bool prohibidoelpaso1 = false;
bool altohastaquesuelteslaconfirmacion = true;
bool altohastaquesuelteselretroceso = true;
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////
///                                                FUNCION INDICA_TEMPERATURA
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////


int intervalo;//vasriable que muestra el mapeado que se produce en cualquier variable
int intervalodedondevengo = 1; //variable de la funcion de la barra de la medicion, guarda eln numero del cuadrado ultimo que se ha dibujado
int precision = 30; //regula el tamaño de los cuadrados de la barra de medición //12 ya es demasiado si no se cambia la anchura //Un 8 15 3 no queda mal en horizontal// Un 15 6 2 no esta mal en vertical
int altura = 6; //altura del cxuadrado de la barra de medición
int anchura = 5; //anchura del cuadrado de la barra de medición
int posx = 50; //posicion x del cuadrado de la barra de medición
int posy = 240; //posicion y del cuadrado de la barra de medición
int depx = 430;
int depy = 204;
int batx = 360;
int baty = 204;
int tempx = 290;
int tempy = 204;
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////
///                                                FUNCION INDICA_TENSION
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////

int intervaloV;//vasriable que muestra el mapeado que se produce en cualquier variable
int intervalodedondevengoV = 1; //variable de la funcion de la barra de la medicion, guarda eln numero del cuadrado ultimo que se ha dibujado
int precisionV = 30; //regula el tamaño de los cuadrados de la barra de medición //12 ya es demasiado si no se cambia la anchura //Un 8 15 3 no queda mal en horizontal// Un 15 6 2 no esta mal en vertical
int alturaV = 6; //altura del cxuadrado de la barra de medición
int anchuraV = 5; //anchura del cuadrado de la barra de medición
int posxV = 50; //posicion x del cuadrado de la barra de medición
int posyV = 240; //posicion y del cuadrado de la barra de medición
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////
///                                                FUNCION INDICA_REVOLUCION
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////

int intervaloR;//vasriable que muestra el mapeado que se produce en cualquier variable
int intervalodedondevengoR = 1; //variable de la funcion de la barra de la medicion, guarda eln numero del cuadrado ultimo que se ha dibujado
int precisionR = 8; //regula el tamaño de los cuadrados de la barra de medición //12 ya es demasiado si no se cambia la anchura //Un 8 15 3 no queda mal en horizontal// Un 15 6 2 no esta mal en vertical
int alturaR = 50; //altura del cxuadrado de la barra de medición
```

```
int anchuraR = 8; //anchura del cuadrado de la barra de medición
int posxR = 50; //posicion x del cuadrado de la barra de medición
int posyR = 240; //posicion y del cuadrado de la barra de medición
////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//
////////////////////////////////////////////////////////////////////////////////Revolucion
volatile int contadorRevolucion = 0;
volatile boolean Interrupcion_producida = false;
////////////////////////////////////////////////////////////////////////////////Velocidad
volatile float contadorVelocidad = 0;
volatile long distanciacontador = 0;
long distanciacontadorprevio;
volatile float distanciacontadorlargo = 0;
float distanciacontadorlargoprevio;
volatile float distanciacontadorinstantaneo = 0;
int kilometraje = 0;
float kilometrajelargo = 0;
float kilometrajeinstantaneo = 0;
float consumomedio = 0;
float consumomedioprevias = 0;
float autonomiainstantaneoprevias = 0;
float autonomiainstantaneo = 0;
float autonomiamediaprevias = 0;
float autonomiamedia = 0;
//volatile boolean Interrupcion_producida = false;
////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////
///                                                 FUNCION INDICA_DEPOSITO
////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////

int intervaloD;//vasriable que muestra el mapeado que se produce en cualquier variable
int intervalodedondevengoD = 1; //variable de la funcion de la barra de la medicion, guarda eln numero del cuadrado ultimo que se ha dibujado
int    intervalodedondevengoDR = 14; //para el color rojo
int    intervalodedondevengoDN = 7; //para el color naranja
int precisionD = 30; //regula el tamaño de los cuadrados de la barra de medición //12 ya es demasiado si no se cambia la anchura //Un 8 15 3 no queda mal en horizontal// Un 15 6 2 no esta mal en vertical
int alturaD = 6; //altura del cxuadrado de la barra de medición
int anchuraD = 5; //anchura del cuadrado de la barra de medición
int posxD = 50; //posicion x del cuadrado de la barra de medición
int posyD = 240; //posicion y del cuadrado de la barra de medición
////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////
///                                                 ESTAS VARIABKLES YA NI SE USAN
////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////
bool semaforo1 = true; //Controla la  ejecución del movimiento de  subida y bajada

bool semaforo = true; //Controla la ejecución de las pantallas para evitar que se recarguen sin necesidad alguna.
bool retrocesoactivo = false;
```

```
bool retrocesocedeelpaso = true;
bool retrocesoesperacambio = true;
bool eleccioncedeelpaso = true;
bool eleccionesperacambio = true;
bool yasecargo3 = false; //indica que la pagina ya se ha cargado, así que cuando se suelte el boton de confirmar , el no pulsar yamará constantemente a la pagina de la cual entró, en este caso la 3
//bool altohastaquesuelteslabajada=true;
bool paginacargada1 = true;

////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
///                                                FUNCION POSICIONAMIENTOTEMPERATURA
////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////

int misposicionesX[20];//vector donde se almacenan las posiciones de los cuadrados
int rellenodeposicionesX;//variable que rellena el vector anterior
int posx1 = 80; //variable que indica que posicion x empiezan a imprimirse los cuadrados
int misposicionesY[20];
int posy1 = 290;
int rellenodeposicionesY;
bool horizontal = false;
bool vertical = true;
int posicionhorizontalfija = 315; //posicion eje x de la impresion de cuadrados
int posicionverticalfija = 240;
////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
///                                                FUNCION POSICIONAMIENTOTENSION
////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
int misposicionesXV[20];//vector donde se almacenan las posiciones de los cuadrados
int rellenodeposicionesXV;//variable que rellena el vector anterior
int posx1V = 80; //variable que indica que posicion x empiezan a imprimirse los cuadrados
int misposicionesYV[20];
int posy1V = 290;
int rellenodeposicionesYV;
bool horizontalV = false;
bool verticalV = true;
int posicionhorizontalfijaV = 380;
int posicionverticalfijaV = 240;
////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
///                                                FUNCION CONFIGURACION
////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
bool cambioconfiguracionagps = false;
int iconfiguracion = 50;
int numerototalmarchaprevia = 0;
float numerototaldepositoprevia = 0.0;
bool pagina1enactivo = true;
int mientrasnopulsesporsegundavezSnosaltaras = 2;
bool pagina2enactivo = false;
```

```
bool configurarpaginaenblanco = true;
bool altohastaquesuelteslabajadaC = true;
bool altohastaquesuelteslabajadaC2 = true;
bool altohastaquesuelteslabajadaC3 = true;
bool altohastaquesuelteslabajadaC4 = true;
bool altohastaquesuelteslabajadaC5 = true;
bool altohastaquesuelteslabajadaC6 = true;
bool altohastaquesuelteslabajadaC7 = true;
bool esperaobligatoria = false;
bool estuveennormal = false;
bool entoncesgpsactivated;
bool estuveengps = true;
bool entoncesnormalactivated;

bool estuveenNo = false;
bool estuveenNoResetConsumo = false;
bool estuveenNoKilometraje = false;

bool entoncesSiactivated = false;
bool entoncesSiactivatedResetConsumo = false;
bool entoncesSiactivatedKilometraje = false;

bool estuveenSi = true;
bool estuveenSiResetConsumo = true;
bool estuveenSiKilometraje = true;


bool entoncesNoactivated = false;
bool entoncesNoactivatedResetConsumo = false;
bool entoncesNoactivatedKilometraje = false;

String tipovelocidad = " Normal ";
String recarga_gasolina = " No ";
String resetear_consumo_medio = " No ";
String reset_kilometraje = " No ";

////////////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////
///                                                FUNCION POSICIONAMIENTOREVOLUCION
////////////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////
int misposicionesXR[20];//vector donde se almacenan las posiciones X de los cuadrados
int rellenodeposicionesXR;//variable que rellena el vector anterior
int posx1R = 115; //variable que indica que posicion x empiezan a imprimirse los cuadrados
int misposicionesYR[20];//vector donde se almacenan las posiciones Y de los cuadrados
int posy1R = 120; //variable que indica que posicion y empiezan a imprimirse los cuadrados
int rellenodeposicionesYR;
bool horizontalR = true; //indica se la barra va horizontal
bool verticalR = false; //indica si la barra va vertical
int posicionhorizontalfijaR = 167; //es la parte de las Y de los cuadrados que se mantiene constante al ser una barra vertical
```

```
int posicionverticalfijaR = 7; //es la parte de las X de los cuadrados que se mantiene constante al ser una barra horizontal
bool alert;//indica si se produjo una alarma al llegar a revoluciones un tanto peligrosas
int XTRI = 456; //posicion X del triangulo de alerta
int YTRI = 0; //posicion Y del trangulo de alerta
int LTRI = 9; // longitud del lado del triangulo de alerta
////////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
///                                                    FUNCION POSICIONAMIENTODEPOSITO
////////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
int misposicionesXD[20];//vector donde se almacenan las posiciones de los cuadrados
int rellenodeposicionesXD;//variable que rellena el vector anterior
int posx1D = 80; //variable que indica que posicion x empiezan a imprimirse los cuadrados
int misposicionesYD[20];
int posy1D = 290;
int rellenodeposicionesYD;
bool horizontalD = false;
bool verticalD = true;
int posicionhorizontalfijaD = 450;
int posicionverticalfijaD = 240;
////////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
///                                                    VARIABLES CONTROLADORAS DEL TIEMPO
////////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////



int tiempo = 500;
int tiempo_actual = 0;
int tiempo_previo1 = 0;
int tiempo_actual1 = 0;
int tiempo_previo2 = 0;
int tiempo_actual3 = 0;
int tiempo_previo3 = 0;
int tiempo3 = 60000;
int tiempo_actual4 = 0;
int tiempo_previo4 = 0;
int tiempo4 = 6000;
////////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
///                                                    FUNCION PAGINA3
////////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////

int Pos_Xnum = 100;//posicion de nuestros numeros de velocidad
int Pos_Ynum = 130;// esta sera nuestra i
int TAMnum = 2;
int Pos_XnumR = 100;//posicion de nuestros numeros de velocidad
int Pos_YnumR = 70;// esta sera nuestra i
int TAMnumR = 1;
```

```
int XCnum = 140;
int YCnum = 60;
int TFontnum = 5;
int Pos_X5 = 240;
int Pos_Y5 = 50;// esta sera nuestra i
int TAM5 = 1;
int XC5 = 104;
int YC5 = 40;
int TFont5 = 4;
int bordex = 107;
int bordey = 2;
int anchuraP3 = 320;
int alturaP3 = 61;
////////////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////
///                                                  FUNCION PAGINA4
////////////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////
int Pos_XnumP4 = 75;//posicion de nuestros numeros de velocidad
int Pos_YnumP4 = 270;// esta sera nuestra i
int TAMnumP4 = 2;
int Pos_XnumRP4 = 230;//posicion de nuestros numeros de velocidad
int Pos_YnumRP4 = 270;// esta sera nuestra i
int TAMnumRP4 = 1;
////////////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////
///                                                  FUNCION INCLINOMETRO
////////////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////
float ACX = 0;
float ACY = 0;
float ACZ = 0;
float GiX = 0;
float GiY = 0;
float GiZ = 0;
float ACXprevias = 0;
float ACYprevias = 0;
float ACZprevias = 0;
float GXprevias = 0;
float GYprevias = 0;
float GZprevias = 0;
int TAMINCLINOMETRO = 1;
int PosXnumINCLI = 75;
int PosYnumINCLI = 250;

////////////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////
///                                                  FUNCION TIEMPOTRIAL
////////////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////
int numerodeetapas = 1;
int numerodeetapasprevia = 0;
```

```
int Vhd[100] = {0}; //cambiar 20 por una variable X
int Vmd[100] = {0};
int Vsd[100] = {0};
int Vhu[100] = {0}; //cambiar 20 por una variable X
int Vmu[100] = {0};
int Vsu[100] = {0};
int Vhdprevias[100] = {0}; //cambiar 20 por una variable X
int Vmdprevias[100] = {0};
int Vsdprevias[100] = {0};
int Vhuprevias[100] = {0}; //cambiar 20 por una variable X
int Vmuprevias[100] = {0};
int Vsuprevias[100] = {0};
int tamhoras = 7;
int tamtextohoras = 2;

int itrial = -1;
int trial = 0;
int trialprevio = -1;
int u_horaprevias = 0;          //Cargamos un 0 en la variable "u_hora"
int u_minutoprevias = 0;        //Cargamos un 0 en la variable "u_minuto"
int u_segundoprevias = 0;       //Cargamos un 0 en la variable "u_segundo"
int d_horaprevias = 0;          //Cargamos un 0 en la variable "d_hora"
int d_minutoprevias = 0;        //Cargamos un 0 en la variable "d_minuto"
int d_segundoprevias = 0;       //Cargamos un 0 en la variable "d_segundo"
int u_hora = 0;          //Cargamos un 0 en la variable "u_hora"
int u_minuto = 0;        //Cargamos un 0 en la variable "u_minuto"
int u_segundo = 0;       //Cargamos un 0 en la variable "u_segundo"
int d_hora = 0;          //Cargamos un 0 en la variable "d_hora"
int d_minuto = 0;        //Cargamos un 0 en la variable "d_minuto"
int d_segundo = 0;       //Cargamos un 0 en la variable "d_segundo"
int u_horapreviasT = 0;          //Cargamos un 0 en la variable "u_hora"
int u_minutopreviasT = 0;        //Cargamos un 0 en la variable "u_minuto"
int u_segundopreviasT = -1;      //Cargamos un 0 en la variable "u_segundo"
int d_horapreviasT = 0;          //Cargamos un 0 en la variable "d_hora"
int d_minutopreviasT = 0;        //Cargamos un 0 en la variable "d_minuto"
int d_segundopreviasT = 0;       //Cargamos un 0 en la variable "d_segundo"
int u_horaT = 0;          //Cargamos un 0 en la variable "u_hora"
int u_minutoT = 0;        //Cargamos un 0 en la variable "u_minuto"
int u_segundoT = -1;      //Cargamos un 0 en la variable "u_segundo"
int u_segundoTC = 0;
int d_horaT = 0;          //Cargamos un 0 en la variable "d_hora"
int d_minutoT = 0;        //Cargamos un 0 en la variable "d_minuto"
int d_segundoT = 0;       //Cargamos un 0 en la variable "d_segundo"
int contadorv = 0;
int contadorvprevio = 2;
unsigned long timer1 = 0;
unsigned long timer2 = 0;
unsigned long timer1T = 0;
unsigned long timer2T = 0;
bool primeraveztime = true;
bool primeraveztimeT = true;
```

```
bool segundaveztime = true;
bool solounavezcomootrastantasya = true;
////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
///                                                  FUNCION PRINCIPAL
////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
//POSICION DEL VALOR DE LA TENSION
int PXTN = 340; //350;
int PYTN = 300;
//POSICION DEL VALOR DE LA TEMPERATURA
int PXTM = 250; //290;
int PYTM = 300;
//POSICION DEL VALOR DEL DEPOSITO
int PXDP = 420; // 430;
int PYDP = 300;
//////////////VARIABLES QUE COMPRUEBAN QUE UNA UNIDAD DE TIEMPO, ESPACIO, TENSION,
TEMPERATURA, DEPOSITO,ETC HAN SUFRIDO ALGUNA MODIFICACION
int millaresprevios = 0;
//int millares;
//int centenas;
//int decenas;
int centenasprevias = 0;
int decenasprevias = 0;
int unidadesprevias = 0;
int millarespreviosR = 0;
int centenaspreviasR = 0;
int decenaspreviasR = 0;
int unidadespreviasR = 0;
int numeroprevio;
int numero;
int temperaturaprevias = 0;
int tensionprevias = 0;
float depositoprevias = 0;
bool escogemosGPS = false;
bool escogemosNORMAL = true;
bool escogemosSi = false;
bool escogemosNo = true;
bool escogemosSiResetConsumo = false;
bool escogemosNoResetConsumo = true;
bool escogemosSiKilometraje = false;
bool escogemosNoKilometraje = true;
int pulsos_caudal = 0;
float deposito_total = 30.0;
float deposito_total2 = 30.0;
float deposito_total_maximo = 30.0;
float contadorconsumoinstantaneo = 0.5;
float consumoinstantaneo = 0;
float consumoinstantaneoprevias = 0;
float litrosconsumo = 0;
float litrosconsumoprevias = 0;
float contadorconsumolargo = 0;
```

```
float contadorconsumolargoprevio;
float litrosconsumolargo = 0;
float litrosconsumopreviaslargo = 0;

float mililitro_pulso = 0.01;
float deposito_previo;
float bateria_total_maxima = 14.0;
float bateria_total = 14.0;
int temperatura_total_maxima = 100;
int temperatura_total = 100;
////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
///                                                FUNCION MARCHA
////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
bool neutro = false; //indica si estamos en modo neutro
int marcha_en_la_que_estoy = 4; //indica la marcha en la que te encuentras ahora mismo esta se
actualiza en la tabla que te de jose alfonso
int marcha_recomendada = 5; //indica la marcha que recomienda el modo es la que te recomienda la
tabla de jose alfonso, se compara con la que estas
int marcha_actual = 0; //indica la marcha actual
int relacion_marcha;
////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
///                                                FUNCION TRIANGULOARRIBA
////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
bool subir;
int PXTA = 420;
int PYTA = 80;
////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
///                                                FUNCION TRIANGULOABAJO
////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
bool bajar;
int PXTB = 420;
int PYTB = 155;
////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
///                                                COLORES
////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////

#define verde1  0x0FE0
#define verde2  0x27E0
#define verde3  0x37E0
#define verde4  0x67E0
#define verde5  0x97E0
#define verde6  0xAFE0
#define verde7  0xCFE0
#define amarillo1  0xE7E0
```

```
#define amarillo2  0xFFE0
#define amarillo3  0xFF80//10
#define amarillo4  0xFEC0
#define amarillo5  0xFE60
#define naranja1  0xFDC0
#define naranja2  0xFD00
#define naranja3  0xFC60
#define naranja4  0xFBA0
#define naranja5  0xFAA0
#define naranja6  0xF9E0
#define naranja7  0xF920
#define rojo  0xF800
uint16_t colores;
////////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////
///                                                    EEPROM
////////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////

union Float_Byte
{
  float datoF;
  byte  datoB[4];
} unionFB;

float EPSILON = 10;
bool AreSame(float a, float b)
{
  return fabs(a - b) < EPSILON;
}

union Float_Byte4
{
  float datoF;
  byte  datoB[4];
} guardarconsumomediokm;

union Float_Byte5
{
  float datoF;
  byte  datoB[4];
} guardarconsumomediolitros;

union Integer_Byte4
{
  int  datoI;//lo cambie por float, antes era un int,
  byte datoB[2];
} guardadodistanciatotal;


union Float_Byte2
{
```

```
  float datoF;
  byte  datoB[4];
} latitud;
union Float_Byte3
{
  float datoF;
  byte  datoB[4];
} longitud;

union Integer_Byte
{
  int  datoI;
  byte datoB[4];
} numerototaldeposito;

union Integer_Byte2
{
  int  datoI2;
  byte datoB[2];
} numerototalmarcha;
union Integer_Byte3
{
  float datoI3;
  byte datoB[4];
} numerodeposito;
#define PConfirmar  4
#define PRetroceso  5
#define PSubir  2
#define PBajar  3
#define TFT_GREY 0x8410
//0x5AEB
UTFT lcd(ILI9481, 38, 39, 40, 41);
void setup() {
  ////////////////////////////////////////////////// Inclinometro
  Wire.begin();          //Iniciando I2C
  sensor.initialize();    //Iniciando el sensor


  // Valores de calibracion
  sensor.setXGyroOffset(56);
  sensor.setYGyroOffset(-44);
  sensor.setZGyroOffset(15);
  sensor.setZAccelOffset(1526);
  ////////////////////////////////////////////////////

  ////////////////////////////////////////////CAUDALIMETRO
  Serial1.begin(9600);
  pinMode(caudalimetro, INPUT);
  attachInterrupt(digitalPinToInterrupt(caudalimetro), debounceCount, RISING);
  ////////////////////////////////////////////velocimetro
  Serial1.begin(9600);
  pinMode(velocimetro, INPUT);
```

```
attachInterrupt(digitalPinToInterrupt(velocimetro), debounceCount3, RISING);

//////////////////////////////////////////cuenta revoluciones
Serial1.begin(9600);
pinMode(cuentarevoluciones, INPUT);
attachInterrupt(digitalPinToInterrupt(cuentarevoluciones), debounceCount2, RISING);
//////////////////////////////////////
//////////////////////////// Temperatura externa

dht.begin();
//   Serial.begin(9600);
Serial2.begin(9600);
Serial2.begin(9600);
tft.init();
tft.invertDisplay(1);
tft.setRotation(1);
tft.fillScreen(TFT_BLACK);
pinMode(PConfirmar, INPUT);
pinMode(PRetroceso, INPUT);
pinMode(PSubir, INPUT);
pinMode(PBajar, INPUT);
//attachInterrupt(digitalPinToInterrupt(PSubir), objetivo_moverse, RISING);

//lcd.InitLCD();

//  numerodeposito.datoI3=100.0;
//     EEPROM.write(10, numerodeposito.datoB[0]);
//    EEPROM.write(11, numerodeposito.datoB[1]);
//    EEPROM.write(12, numerodeposito.datoB[2]);
//    EEPROM.write(13, numerodeposito.datoB[3]);
//////////////////Lectura valores EEPROM
numerototalmarcha.datoB[0] =  EEPROM.read(4);
numerototalmarcha.datoB[1] =  EEPROM.read(5);
numerototaldeposito.datoB[0] =  EEPROM.read(6);
numerototaldeposito.datoB[1] =  EEPROM.read(7);
numerototaldeposito.datoB[2] =  EEPROM.read(8);
numerototaldeposito.datoB[3] =  EEPROM.read(9);
numerodeposito.datoB[0] =  EEPROM.read(10);
numerodeposito.datoB[1] =  EEPROM.read(11);
numerodeposito.datoB[2] =  EEPROM.read(12);
numerodeposito.datoB[3] =  EEPROM.read(13);
longitud.datoB[0] = EEPROM.read(0);
longitud.datoB[1] = EEPROM.read(1);
longitud.datoB[2] = EEPROM.read(2);
longitud.datoB[3] = EEPROM.read(3);
latitud.datoB[0] = EEPROM.read(14);
latitud.datoB[1] = EEPROM.read(15);
latitud.datoB[2] = EEPROM.read(16);
latitud.datoB[3] = EEPROM.read(17);
guardadodistanciatotal.datoB[0] = EEPROM.read(18);
guardadodistanciatotal.datoB[1] = EEPROM.read(19);
guardarconsumomediokm.datoB[0] = EEPROM.read(20);
```

```
  guardarconsumomediokm.datoB[1] = EEPROM.read(21);
  guardarconsumomediokm.datoB[2] = EEPROM.read(22);
  guardarconsumomediokm.datoB[3] = EEPROM.read(23);
  guardarconsumomediolitros.datoB[0] = EEPROM.read(24);
  guardarconsumomediolitros.datoB[1] = EEPROM.read(25);
  guardarconsumomediolitros.datoB[2] = EEPROM.read(26);
  guardarconsumomediolitros.datoB[3] = EEPROM.read(27);
  guardarconsumomediolitros.datoF = 2000.0;
  guardarconsumomediokm.datoF = 55556.0;
  guardadodistanciatotal.datoI = 0;
  EEPROM.write(18, guardadodistanciatotal.datoB[0]);
  EEPROM.write(19, guardadodistanciatotal.datoB[1]);
  EEPROM.write(20, guardarconsumomediokm.datoB[0]);
  EEPROM.write(21, guardarconsumomediokm.datoB[1]);
  EEPROM.write(22, guardarconsumomediokm.datoB[2]);
  EEPROM.write(23, guardarconsumomediokm.datoB[3]);
  EEPROM.write(24, guardarconsumomediolitros.datoB[0]);
  EEPROM.write(25, guardarconsumomediolitros.datoB[1]);
  EEPROM.write(26, guardarconsumomediolitros.datoB[2]);
  EEPROM.write(27, guardarconsumomediolitros.datoB[3]);

  contadorconsumolargo = guardarconsumomediolitros.datoF;
  distanciacontadorlargo = guardarconsumomediokm.datoF;
  distanciacontador = guardadodistanciatotal.datoI;
  deposito_total_maximo = numerototaldeposito.datoI;
  deposito_total = numerodeposito.datoI3;
  deposito_total2 = deposito_total;
  posicionamiento_revolucion();
  delay(1000);
  bool mysd = 0;
  while (!mysd) {
    if (!sd.begin(SD_CHIP_SELECT, SPI_FULL_SPEED)) {
      Serial.println(F("fallo"));
      Serial.println(F("reintento"));

    }
    else {
      mysd = 1;
      Serial.println(F("inicializo"));

    }
  }
  //myFiles.load(0, 0, 480, 320, "cerberus.RAW", 1, 0);
  myFiles.load(0, 0, 480, 320, "upct3.RAW", 1, 0);

  delay(3000);
}

void loop() {
  /// Si page es 2 abremos seleccionado la pagina 2
  if (page == 2)
```

```
  {
    Page2(); ///Funcion que permite inicializar la pantalla, solo se realiza una vez.
    configuracion();// Funcion que se encarga de la confifuración,es la parte dinamica de la pantalla
    movimiento(); //Funcion que se encarga de detectar que se requiere o no cambiar a la siguiente pantalla, en este caso por estar en la pantalla 2, pasariamos a la pantalla 3 de tipo GENERAL
  }
  /// Si page es 2 abremos seleccionado la pagina 3
  if (page == 3)
  {

    Page3(); ///Funcion que permite inicializar la pantalla, solo se realiza una vez.
    //int numero=random(0,numeromax);
    movimiento();//Funcion que se encarga de detectar que se requiere o no cambiar a la siguiente pantalla, en este caso por estar en la pantalla 2, pasariamos a la pantalla 3 de tipo GENERAL
    ////////////////////////////////////////////////////////Parte de funcion del GPS debe esperar 500 milisegunos para poder traer un dato del Satelite, eso nos lleva el problema de que el resto debe esperar 1 segundo hasta ser representado
    adquisiciondevelocidadyrevolucionconosingps();


  }
  if (page == 4)
    ////////////////////////REVISAR PAGINA 4 QUE SE VE DE PENA, CONTROLAR EL ROLLO DE REVOLUCIONES
  {
    Page4();///Inicializacion de la pagina 4
    movimiento(); ///La funcion permite cambiar entre paginas
    velocidadrevolucionrepresentacioncontadordevueltas();
  }

  movimiento();




}////////aqui se acaba el LOOP
//////////////////////////FUNCION ADQUISICION DE VELOCIDAD Y REVOLUCIONES GPS O TACOMETRRO
void velocidadrevolucionrepresentacioncontadordevueltas()
{
  distanciacontador = 25600000;
  kilometraje = (distanciacontador * 18) / 100000;

  int millares = kilometraje / 1000;
  int centenas = (kilometraje - (millares * 1000)) / 100;
  int decenas = (kilometraje - (millares * 1000 + centenas * 100)) / 10;
  int unidades = kilometraje - (millares * 1000 + centenas * 100 + decenas * 10 );
  //int numeroR = random(0, 3500);
  // int millaresR = numeroR / 1000;
  //  int centenasR = (numeroR - (millaresR * 1000)) / 100;
  //  int decenasR = (numeroR - (millaresR * 1000 + centenasR * 100)) / 10;
  //  int unidadesR = numeroR - (millaresR * 1000 + centenasR * 100 + decenasR * 10 );
```

```
 int temperatura = 100;
 float tension = 12.4;
 int deposito = 60;
 ////////////para ver el kilometraje por primera vez y que los ceros no se queden en blanco
 if (entre5)
 {
  tft.setCursor(Pos_XnumP4, Pos_YnumP4 , 4);
  tft.setTextColor(TFT_BLACK);  tft.setTextSize(TAMnumRP4);
  tft.print(millares);
  millaresprevios = millares;

  tft.setCursor(Pos_XnumP4 + 20, Pos_YnumP4 , 4);
  tft.setTextColor(TFT_BLACK);  tft.setTextSize(TAMnumRP4);
  tft.print(centenas);
  centenasprevias = centenas;

  tft.setCursor(Pos_XnumP4 + 40, Pos_YnumP4 , 4);
  tft.setTextColor(TFT_BLACK);  tft.setTextSize(TAMnumRP4);
  tft.print(decenas);
  decenasprevias = decenas;

  tft.setCursor(Pos_XnumP4 + 60, Pos_YnumP4 , 4);
  tft.setTextColor(TFT_BLACK);  tft.setTextSize(TAMnumRP4);
  tft.print(unidades);
  unidadesprevias = unidades;
  entre5 = false;
 }
 tiempo_actual = millis();
 if (tiempo_actual >= tiempo_previo1 + tiempo)
 {
  //tft.fillRoundRect(Pos_Xnum-(XCnum*TAMnum)+26*TAMnum, Pos_Ynum-(5*TAMnum),
XCnum*TAMnum, YCnum*TAMnum, 7, TFT_BLACK); // Base
  if (millaresprevios != millares)
  {
   tft.setCursor(Pos_XnumP4, Pos_YnumP4 , 4);
   tft.setTextColor(TFT_WHITE);  tft.setTextSize(TAMnumRP4);
   tft.print(millaresprevios);
   tft.setCursor(Pos_XnumP4, Pos_YnumP4 , 4);
   tft.setTextColor(TFT_BLACK);  tft.setTextSize(TAMnumRP4);
   //Indica_PotenciaConsumida(numero);

   tft.print(millares);
   millaresprevios = millares;

  }
  if (centenasprevias != centenas)
  {
   tft.setCursor(Pos_XnumP4 + 20, Pos_YnumP4 , 4);
   tft.setTextColor(TFT_WHITE);  tft.setTextSize(TAMnumRP4);
   tft.print(centenasprevias);
   tft.setCursor(Pos_XnumP4 + 20, Pos_YnumP4 , 4);
   tft.setTextColor(TFT_BLACK);  tft.setTextSize(TAMnumRP4);
```

```
    tft.print(centenas);
    centenasprevias = centenas;
   }
   if (decenasprevias != decenas)
   {
    tft.setCursor(Pos_XnumP4 + 40, Pos_YnumP4 , 4);
    tft.setTextColor(TFT_WHITE);  tft.setTextSize(TAMnumRP4);
    tft.print(decenasprevias);
    tft.setCursor(Pos_XnumP4 + 40, Pos_YnumP4 , 4);
    tft.setTextColor(TFT_BLACK);  tft.setTextSize(TAMnumRP4);
    tft.print(decenas);
    decenasprevias = decenas;
   }

   if (unidadesprevias != unidades)
   {
    tft.setCursor(Pos_XnumP4 + 60, Pos_YnumP4 , 4);
    tft.setTextColor(TFT_WHITE);  tft.setTextSize(TAMnumRP4);
    tft.print(unidadesprevias);
    tft.setCursor(Pos_XnumP4 + 60, Pos_YnumP4 , 4);
    tft.setTextColor(TFT_BLACK);  tft.setTextSize(TAMnumRP4);
    tft.print(unidades);
    unidadesprevias = unidades;
   }
   movimiento();
   litrosconsumo = (contadorconsumoinstantaneo) / 1000.0;
   kilometrajeinstantaneo = (distanciacontadorinstantaneo * 18.0) / 100000.0;
   consumoinstantaneo = (litrosconsumo * 100.0) / kilometrajeinstantaneo;

   if (consumoinstantaneoprevias != consumoinstantaneo)
   {

    tft.setCursor(Pos_XnumRP4 + 130, Pos_YnumRP4 , 4);
    tft.setTextColor(TFT_WHITE);  tft.setTextSize(TAMnumRP4);
    tft.print(consumoinstantaneoprevias, 1);
    tft.setCursor(Pos_XnumRP4 + 130, Pos_YnumRP4 , 4);
    tft.setTextColor(TFT_BLACK);  tft.setTextSize(TAMnumRP4);
    tft.print(consumoinstantaneo, 1);
    consumoinstantaneoprevias = consumoinstantaneo;
    contadorconsumoinstantaneo = 0;
    distanciacontadorinstantaneo = 0;

   }
   litrosconsumolargo = (contadorconsumolargo) / 1000.0; //consumo medio en litros
   kilometrajelargo = (distanciacontadorlargo * 18.0) / 100000.0; //kilometraje en kilometros
   consumomedio = (litrosconsumolargo * 100.0) / kilometrajelargo; //consumo medio en litros /
kilommetros
   ////////////////////////          REPRESENTACIÓN DEL CONSUMO MEDIO CADA 100 KILOMETROS
   if (entre4)
   {
    tft.setCursor(Pos_XnumRP4 + 130, 300 , 4);
    tft.setTextColor(TFT_BLACK);  tft.setTextSize(TAMnumRP4);
```

```
 tft.print(consumomedio, 1);
 entre4 = false;
}
if (consumomedioprevias != consumomedio)
{

 tft.setCursor(Pos_XnumRP4 + 130, 300 , 4);
 tft.setTextColor(TFT_WHITE);  tft.setTextSize(TAMnumRP4);
 tft.print(consumomedioprevias, 1);
 tft.setCursor(Pos_XnumRP4 + 130, 300 , 4);
 tft.setTextColor(TFT_BLACK);  tft.setTextSize(TAMnumRP4);
 tft.print(consumomedio, 1);
 consumomedioprevias = consumomedio;

}
//////////////////////////////////////////////////////
/*if (centenaspreviasR != centenasR)
 {
 tft.setCursor(Pos_XnumRP4 + 20, Pos_YnumRP4 , 4);
 tft.setTextColor(TFT_WHITE);  tft.setTextSize(TAMnumRP4);
 tft.print(centenaspreviasR);
 tft.setCursor(Pos_XnumRP4 + 20, Pos_YnumRP4 , 4);
 tft.setTextColor(TFT_BLACK);  tft.setTextSize(TAMnumRP4);
 tft.print(centenasR);
 centenaspreviasR = centenasR;
 }
 if (decenaspreviasR != decenasR)
 {
 tft.setCursor(Pos_XnumRP4 + 40, Pos_YnumRP4 , 4);
 tft.setTextColor(TFT_WHITE);  tft.setTextSize(TAMnumRP4);
 tft.print(decenaspreviasR);
 tft.setCursor(Pos_XnumRP4 + 40, Pos_YnumRP4 , 4);
 tft.setTextColor(TFT_BLACK);  tft.setTextSize(TAMnumRP4);
 tft.print(decenasR);
 decenaspreviasR = decenasR;
 }

 if (unidadespreviasR != unidadesR)
 {
 tft.setCursor(Pos_XnumRP4 + 60, Pos_YnumRP4 , 4);
 tft.setTextColor(TFT_WHITE);  tft.setTextSize(TAMnumRP4);
 tft.print(unidadespreviasR);
 tft.setCursor(Pos_XnumRP4 + 60, Pos_YnumRP4 , 4);
 tft.setTextColor(TFT_BLACK);  tft.setTextSize(TAMnumRP4);
 tft.print(unidadesR);
 unidadespreviasR = unidadesR;
 }*/
movimiento();

tiempo_previo1 = millis();

tiempo_actual = millis();
```

```
  }

  if (solounavezcomootrastantasya) {
    escribe_hora();//ovio para que dibuje el 00:00:00
    escribe_horaT();
    solounavezcomootrastantasya = false;
  }
  vectordetiempo();
}
void adquisiciondevelocidadyrevolucionconosingps()
{
  unsigned long start = millis();
  // Every 5 seconds we print an update
  while (millis() - start < 500) {
    if (Serial2.available() > 0) { ///comprobamos que puerto serie numero 1 tiene datos para recoger
      char c = Serial2.read();
      if (gps.encode(c)) {
      }
    }
  }
  gps.f_get_position(&flat, &flon, &age);
  if (flon < 1000.0 )
  {
    flat = flat * 100000.0;
    flon = flon * 100000.0;

    if (AreSame(flat, latitud.datoF * 100000.0))
    {
      Serial.println("No robado");
      Serial.println(latitud.datoF * 100000.0);

    }
    else
    {
      Serial.println("Si robado");
      Serial.println(latitud.datoF * 100000.0);

    }
    if (AreSame(flon, longitud.datoF * 100000.0))
    {
      Serial.println("No robado");
      Serial.println(longitud.datoF * 100000.0);

    }
    else
    {
      Serial.println("Si robado");
      Serial.println(longitud.datoF * 100000.0);

    }
  }
  // if (newdata) {
```

```
  if (escogemosGPS) { ///aqui decidimos que la velocidad mostrada será la del gps
    numero = gps.f_speed_mph();
    //}
  }
  if (escogemosNORMAL) { //aqui decidimos que la velocidad mostrada será la de la propia moto
    //numero=contadorVelocidad*1.296;
    numero = random(0, 2/*numeromax*/);

  }
  contadorVelocidad = 0;
  //separacion de numeros en unidades mas sencillas provenientes de la variable numero Este caso es
para la velocidad
  int millares = numero / 1000;
  int centenas = (numero - (millares * 1000)) / 100;
  int decenas = (numero - (millares * 1000 + centenas * 100)) / 10;
  int unidades = numero - (millares * 1000 + centenas * 100 + decenas * 10 );

  ////////////////////AQUI PILLAMOS LA REVOLUCION
  //   int numeroR = (contadorRevolucion*2)*31.302-13.609; ///Revisar porque si 6000 revoluciones es
alta, habrá que modificar los limites de la representacion del map
  contadorRevolucion = 0;
  //int numeroR=2000;
  int numeroR = random(0, 6000);
  //  int numeroR=int(gps.f_speed_kmph());
  //separacion de numeros en unidades mas sencillas provenientes de la variable numero Este caso es
para la revolucion

  int millaresR = numeroR / 1000;
  int centenasR = (numeroR - (millaresR * 1000)) / 100;
  int decenasR = (numeroR - (millaresR * 1000 + centenasR * 100)) / 10;
  int unidadesR = numeroR - (millaresR * 1000 + centenasR * 100 + decenasR * 10 );
  ////////////////////////////aqui van los datos de las temperaturas que luego veremos
  int temperatura = 100;
  float tension = 12.4;
  int deposito = 60;
  //Momento de representar las variables cada 0.5 segundos teniendo en cuentaque ya perdemos 0.5
por el satelite
  tiempo_actual = millis();
  if (tiempo_actual >= tiempo_previo1 + tiempo)
  {
    //tft.fillRoundRect(Pos_Xnum-(XCnum*TAMnum)+26*TAMnum, Pos_Ynum-(5*TAMnum),
XCnum*TAMnum, YCnum*TAMnum, 7, TFT_BLACK); // Base
    /*   if (millaresprevios!=millares)
      {
      tft.setCursor(Pos_Xnum-100*TAMnum,Pos_Ynum ,7);
      tft.setTextColor(TFT_BLUE);  tft.setTextSize(TAMnum);
      tft.print(millaresprevios);
      tft.setCursor(Pos_Xnum-100*TAMnum,Pos_Ynum ,7);
      tft.setTextColor(TFT_YELLOW);  tft.setTextSize(TAMnum);
      //Indica_PotenciaConsumida(numero);
```

```
  tft.print(millares);
  millaresprevios=millares;

  }*/
/////////////////Representacion velocidad en pantalla
if (centenasprevias != centenas)
{
  tft.setCursor(Pos_Xnum, Pos_Ynum , 7);
  tft.setTextColor(TFT_WHITE);  tft.setTextSize(TAMnum);
  tft.print(centenasprevias);
  tft.setCursor(Pos_Xnum, Pos_Ynum , 7);
  tft.setTextColor(TFT_BLACK);  tft.setTextSize(TAMnum);
  tft.print(centenas);
  centenasprevias = centenas;
}
if (decenasprevias != decenas)
{
  tft.setCursor(Pos_Xnum + 60, Pos_Ynum , 7);
  tft.setTextColor(TFT_WHITE);  tft.setTextSize(TAMnum);
  tft.print(decenasprevias);
  tft.setCursor(Pos_Xnum + 60, Pos_Ynum , 7);
  tft.setTextColor(TFT_BLACK);  tft.setTextSize(TAMnum);
  tft.print(decenas);
  decenasprevias = decenas;
}

if (unidadesprevias != unidades)
{
  tft.setCursor(Pos_Xnum + 120, Pos_Ynum , 7);
  tft.setTextColor(TFT_WHITE);  tft.setTextSize(TAMnum);
  tft.print(unidadesprevias);
  tft.setCursor(Pos_Xnum + 120, Pos_Ynum , 7);
  tft.setTextColor(TFT_BLACK);  tft.setTextSize(TAMnum);

  tft.print(unidades);
  unidadesprevias = unidades;
}
movimiento();////Controlamos que nos llamen a cambiar de pagina

////Represenatacion de la revolucion en pantalla
if (millarespreviosR != millaresR)
{

  tft.setCursor(Pos_XnumR, Pos_YnumR , 7);
  tft.setTextColor(TFT_WHITE);  tft.setTextSize(TAMnumR);
  tft.print(millarespreviosR);
  tft.setCursor(Pos_XnumR, Pos_YnumR , 7);
  tft.setTextColor(TFT_BLACK);  tft.setTextSize(TAMnumR);
  tft.print(millaresR);
  millarespreviosR = millaresR;

}
```

```
  if (centenaspreviasR != centenasR)
  {
   tft.setCursor(Pos_XnumR + 30, Pos_YnumR , 7);
   tft.setTextColor(TFT_WHITE);  tft.setTextSize(TAMnumR);
   tft.print(centenaspreviasR);
   tft.setCursor(Pos_XnumR + 30, Pos_YnumR , 7);
   tft.setTextColor(TFT_BLACK);  tft.setTextSize(TAMnumR);
   tft.print(centenasR);
   centenaspreviasR = centenasR;
  }
  if (decenaspreviasR != decenasR)
  {
   tft.setCursor(Pos_XnumR + 60, Pos_YnumR , 7);
   tft.setTextColor(TFT_WHITE);  tft.setTextSize(TAMnumR);
   tft.print(decenaspreviasR);
   tft.setCursor(Pos_XnumR + 60, Pos_YnumR , 7);
   tft.setTextColor(TFT_BLACK);  tft.setTextSize(TAMnumR);
   tft.print(decenasR);
   decenaspreviasR = decenasR;
  }

  if (unidadespreviasR != unidadesR)
  {
   tft.setCursor(Pos_XnumR + 90, Pos_YnumR , 7);
   tft.setTextColor(TFT_WHITE);  tft.setTextSize(TAMnumR);
   tft.print(unidadespreviasR);
   tft.setCursor(Pos_XnumR + 90, Pos_YnumR , 7);
   tft.setTextColor(TFT_BLACK);  tft.setTextSize(TAMnumR);
   tft.print(unidadesR);
   unidadespreviasR = unidadesR;
  }
  movimiento(); ///Controlamos que tengamos que cambiar de pantalla
  ///
  indica_revolucion(numeroR);//caso de revolucion
  //
  //



  // myFiles.load(380,170,50,50,"upct.RAW",1,0);
  //////////Representacion de temperatura
  if (temperaturaprevias != temperatura_total)
  {
    indica_temperatura(); //imprime los cuadrados corrspodientes al numero que le salga en caso de
temperatura
    tft.setCursor(PXTM, PYTM, 4);
    tft.setTextColor(TFT_WHITE);  tft.setTextSize(1);
    tft.print(temperaturaprevias); tft.print("C");
    tft.setCursor(PXTM, PYTM , 4);
    tft.setTextColor(TFT_BLACK);  tft.setTextSize(1);
    tft.print(temperatura_total); tft.print("C"); //no sabe dibujar ese tipo de caracter º de grados
centigrados
```

```
    temperaturaprevias = temperatura_total;
   }
   ///representacion de tension
   if (tensionprevias != bateria_total)
   {
    indica_tension();//caso de tension
    tft.setCursor(PXTN, PYTN, 4);
    tft.setTextColor(TFT_WHITE);  tft.setTextSize(1);
    tft.print(tensionprevias); tft.print("V");
    tft.setCursor(PXTN, PYTN , 4);
    tft.setTextColor(TFT_BLACK);  tft.setTextSize(1);
    tft.print(bateria_total); tft.print("V");
    tensionprevias = bateria_total;
   }
   ///representacin de deposito
   if (entre3)
   {
    indica_depositoR(deposito_total); //caso de deposito
    entre3 = false;
   }

   if (deposito_total < deposito_total2 - 1.0)
   {
    indica_depositoR(deposito_total); //caso de deposito
    deposito_total2 = deposito_total;
   }
   if (depositoprevias != deposito_total)
   {

    tft.setCursor(PXDP, PYDP , 4);
    tft.setTextColor(TFT_WHITE);  tft.setTextSize(1);
    tft.print(depositoprevias); //tft.print("%");
    tft.setCursor(PXDP, PYDP, 4);
    tft.setTextColor(TFT_BLACK);  tft.setTextSize(1);
    tft.print(deposito_total);/// tft.print("%");
    depositoprevias = deposito_total;
   }
   ////Representaciond de marcha y posibilidad de recomendacion
   marcha();
   litrosconsumo = (contadorconsumoinstantaneo) / 1000.0;
   kilometrajeinstantaneo = (distanciacontadorinstantaneo * 18.0) / 100000.0;
   consumoinstantaneo = (litrosconsumo * 100.0) / kilometrajeinstantaneo;
   litrosconsumolargo = (contadorconsumolargo) / 1000.0; //consumo medio en litros
   kilometrajelargo = (distanciacontadorlargo * 18.0) / 100000.0; //kilometraje en kilometros
   consumomedio = (litrosconsumolargo * 100.0) / kilometrajelargo; //consumo medio en litros /
kilommetros
   movimiento(); ///La funcion permite cambiar entre paginas
   tiempo_previo1 = millis();

   tiempo_actual = millis();
  }
  //  tiempo_actual1=millis();
```

```
// if (tiempo_actual1>=tiempo_previo2+10000)
//   {
inclinometro();
//   tiempo_previo2=millis();

//    tiempo_actual1=millis();
//}
movimiento();///La funcion permite cambiar entre paginas
tiempo_actual3 = millis();
if (tiempo_actual3 >= tiempo_previo3 + tiempo3)
{
 guardadodistanciatotal.datoI = distanciacontador;
 guardarconsumomediolitros.datoF = contadorconsumolargo;
 guardarconsumomediokm.datoF = distanciacontadorlargo;
 numerodeposito.datoI3 = deposito_total;
 latitud.datoF = flat;
 longitud.datoF = flon;
 EEPROM.write(10, numerodeposito.datoB[0]);
 EEPROM.write(11, numerodeposito.datoB[1]);
 EEPROM.write(12, numerodeposito.datoB[2]);
 EEPROM.write(13, numerodeposito.datoB[3]);
 EEPROM.write(14, latitud.datoB[0]);
 EEPROM.write(15, latitud.datoB[1]);
 EEPROM.write(16, latitud.datoB[2]);
 EEPROM.write(17, latitud.datoB[3]);
 EEPROM.write(0, longitud.datoB[0]);
 EEPROM.write(1, longitud.datoB[1]);
 EEPROM.write(2, longitud.datoB[2]);
 EEPROM.write(3, longitud.datoB[3]);

 h = dht.readTemperature();
 if (hprevias != h)
 {
  tft.setCursor(150, 240, 4);
  tft.setTextColor(TFT_WHITE);  tft.setTextSize(1);
  tft.print(hprevias);
  tft.setCursor(150, 240, 4);
  tft.setTextColor(TFT_BLACK);  tft.setTextSize(1);
  tft.print(h);

  hprevias = h;
 }
 autonomiainstantaneo = deposito_total / consumoinstantaneo;
 autonomiamedia = deposito_total / consumomedio;
 if (autonomiainstantaneoprevias != autonomiainstantaneo)
 {
  tft.setCursor(145, 260, 4);
  tft.setTextColor(TFT_WHITE);  tft.setTextSize(1);
  tft.print(autonomiainstantaneoprevias);
  tft.setCursor(145, 260, 4);
  tft.setTextColor(TFT_BLACK);  tft.setTextSize(1);
  tft.print(autonomiainstantaneo);
```

```
    autonomiainstantaneoprevias = autonomiainstantaneo;
   }
   if (autonomiamediaprevias != autonomiamedia)
   {
    tft.setCursor(145, 280, 4);
    tft.setTextColor(TFT_WHITE);  tft.setTextSize(1);
    tft.print(autonomiamediaprevias);
    tft.setCursor(145, 280, 4);
    tft.setTextColor(TFT_BLACK);  tft.setTextSize(1);
    tft.print(autonomiamedia);

    autonomiamediaprevias = autonomiamedia;
   }

   tiempo_previo3 = millis();

   tiempo_actual3 = millis();
  }


  //    if (tiempo_actual4 >= tiempo_previo4 + tiempo4)
  //  {
  //
  //
  //      tiempo_previo4 = millis();
  //
  //    tiempo_actual4 = millis();
  //  }
}

//////////////////////FUNCION CAUDALIMETRO
void debounceCount()
{

  contadorconsumolargo = contadorconsumolargo + mililitro_pulso; //Este se resetea cuando el usuario
le da la gana, todo en double el consumo esta en mililitros

  contadorconsumoinstantaneo = contadorconsumoinstantaneo + mililitro_pulso; //Este se resetea cada
medio segundo, sirve para el consumo instantáneo
  //pulsos_caudal++;
  deposito_total = deposito_total - mililitro_pulso;
  if (deposito_total < 0) {
    deposito_total = 0;
  }
}
//////////////////////////////    Interrucion para moverse
void objetivo_moverse()
{
  movimiento();
}
//////////////////////FUNCION Revolución interrupcion
```

```
void debounceCount2()
{

  contadorRevolucion = contadorRevolucion + 1;

}
void debounceCount3()
{

  contadorVelocidad = contadorVelocidad + 1.0;//para medir la velocidad
  distanciacontador = distanciacontador + 1;//el cuenta kilometris de toda la vida, no es necesario que
sea double porque no necesita un dato rapido.
  distanciacontadorlargo = distanciacontadorlargo + 1.0;//para ver el consume medio. solo se resetea si
lo hace el consumolargo por parte del usuario
  distanciacontadorinstantaneo = distanciacontadorinstantaneo + 1.0;
}


/////////////// fUNCION INCLINOMETRO
void inclinometro() {
  sensor.getAcceleration(&ax, &ay, &az); //Adquisicion de datos
  sensor.getRotation(&gx, &gy, &gz);
  float ax_m_s2 = ax * (9.81 / 16384.0);
  float ay_m_s2 = ay * (9.81 / 16384.0);
  float az_m_s2 = az * (9.81 / 16384.0);
  float gx_deg_s = gx * (250.0 / 32768.0);
  float gy_deg_s = gy * (250.0 / 32768.0);
  float gz_deg_s = gz * (250.0 / 32768.0);

  //Mostrar las lecturas separadas por un [tab]

  ACX = ax_m_s2;
  ACY = ay_m_s2;
  ACZ = az_m_s2;
  GiX = gx_deg_s;
  GiY = gy_deg_s;
  GiZ = gz_deg_s;
  /*float ACX=0;
    float ACY=0;
    float ACZ=0;
    float GX=0;
    float GY=0;
    float GZ=0;
    float ACXprevias=0;
    float ACYprevias=0;
    float ACZprevias=0;
    float GXprevias=0;
    float GYprevias=0;
    float GZprevias=0;
    int TAMINCLINOMETRO=1;
    int PosXnumINCLI= 75;
    int PosYnumINCLI=220;*/
```

```
 if (ACXprevias != ACX)
 {
  tft.setCursor(PosXnumINCLI, PosYnumINCLI , 7);
  tft.setTextColor(TFT_WHITE);  tft.setTextSize(TAMINCLINOMETRO);
  tft.print(ACXprevias);
  tft.setCursor(PosXnumINCLI, PosYnumINCLI , 7);
  tft.setTextColor(TFT_BLACK);  tft.setTextSize(TAMINCLINOMETRO);
  tft.print(ACX);
  ACXprevias = ACX;
 }
}
////////////////////////////////////////////////
/////////FUNCION DE CONFIGURACION, NOS PERMITE MOSTRAR LAS PANTALLAS DINAMICAS Y
GUARDAR NUESTRAR OPPCIONES PARA LA CONDUCCION
void configuracion() {

 movimientoconfiguracion();///eS EL QUE SE ENCARGA DE QUE MOVERSE POR EL MENU DE LA
APLICACCION  MUY IMPORTANTE
 if (pagina1enactivo) {
  if (configurarpaginaenblanco) {
   tft.fillRect(Mancho, 0, 480 - Mancho, 320, TFT_WHITE);
   configurarpaginaenblanco = false;
   myFiles.load(69, 205, 411, 115, "kuja.RAW", 1, 0);

  }

  tft.setCursor(Mancho + 20, 50 , 4);
  tft.setTextColor(TFT_BLACK);  tft.setTextSize(1);
  tft.println(" Velocidad visual ");
  tft.setCursor(Mancho + 20, 100 , 4);
  tft.setTextColor(TFT_BLACK);  tft.setTextSize(1);
  tft.println(" Resetear Consumos Medios? ");
  tft.setCursor(Mancho + 20, 150 , 4);
  tft.setTextColor(TFT_BLACK);  tft.setTextSize(1);
  tft.println(" Deposito (L) ");
  tft.setCursor(Mancho + 20, 200 , 4);
  tft.setTextColor(TFT_BLACK);  tft.setTextSize(1);
  tft.println(" Recargar Deposito? ");
  tft.setCursor(Mancho + 220, 50 , 4);
  tft.setTextColor(TFT_BLACK);  tft.setTextSize(1);
  tft.println(tipovelocidad);
  tft.setCursor(Mancho + 350, 100 , 4);
  tft.setTextColor(TFT_BLACK);  tft.setTextSize(1);
  tft.print(resetear_consumo_medio);
  tft.setCursor(Mancho + 220, 150 , 4);
  tft.setTextColor(TFT_BLACK);  tft.setTextSize(1);
  tft.print(numerototaldeposito.datoI);
  tft.setCursor(Mancho + 270, 200 , 4);
  tft.setTextColor(TFT_BLACK);  tft.setTextSize(1);
  tft.print(recarga_gasolina);
  if (iconfiguracion == 50)
  {
```

```
    if (digitalRead(PBajar) == HIGH)
    {
      if (altohastaquesuelteslabajadaC) //impide que se vuelva loco y en una pulsacion la cuente como
20(exagerando).////////POR DIOS, CADA UNO CON SU PROPIO BLOQUEO, NO ME LA JODAS MAS
      {
        if (estuveennormal)
        { entoncesgpsactivated = true;
        }
        if (estuveengps)
        { entoncesnormalactivated = true;
        }
        if ( entoncesgpsactivated) {
          tft.setCursor(Mancho + 220, 50 , 4);
          tft.setTextColor(TFT_WHITE);  tft.setTextSize(1);
          tipovelocidad = " Normal ";
          tft.println(tipovelocidad);
          tft.setCursor(Mancho + 220, 50 , 4);
          tft.setTextColor(TFT_BLACK);  tft.setTextSize(1);
          tipovelocidad = " GPS ";
          tft.println(tipovelocidad);
          estuveengps = true;
          estuveennormal = false;
          entoncesgpsactivated = false;
          escogemosGPS = true;
          escogemosNORMAL = false;

        }
        if (entoncesnormalactivated ) {
          tft.setCursor(Mancho + 220, 50 , 4);
          tft.setTextColor(TFT_WHITE);  tft.setTextSize(1);
          tipovelocidad = " GPS ";
          tft.println(tipovelocidad);
          tft.setCursor(Mancho + 220, 50 , 4);
          tft.setTextColor(TFT_BLACK);  tft.setTextSize(1);
          tipovelocidad = " Normal ";
          tft.println(tipovelocidad);
          estuveengps = false;
          estuveennormal = true;
          entoncesnormalactivated = false;

          escogemosGPS = false;
          escogemosNORMAL = true;
        }
        altohastaquesuelteslabajadaC = false;

      }
    }
    if (digitalRead(PBajar) == LOW)
    {


      altohastaquesuelteslabajadaC = true;
```

```
    }


  }

  if (iconfiguracion == 100)
  {
   if (digitalRead(PBajar) == HIGH)
    {
     if (altohastaquesuelteslabajadaC) //impide que se vuelva loco y en una pulsacion la cuente como
20(exagerando).////////POR DIOS, CADA UNO CON SU PROPIO BLOQUEO, NO ME LA JODAS MAS
      {
       if (estuveenNoResetConsumo)
       {
        entoncesSiactivatedResetConsumo = true;
       }
       if (estuveenSiResetConsumo)
       { entoncesNoactivatedResetConsumo = true;
       }
       if ( entoncesSiactivatedResetConsumo) {
        tft.setCursor(Mancho + 350, 100 , 4);
        tft.setTextColor(TFT_WHITE);  tft.setTextSize(1);
        resetear_consumo_medio = " No ";
        tft.println(resetear_consumo_medio);
        tft.setCursor(Mancho + 350, 100 , 4);
        tft.setTextColor(TFT_BLACK);  tft.setTextSize(1);
        resetear_consumo_medio = " Si ";
        tft.println(resetear_consumo_medio);
        estuveenSiResetConsumo = true;
        estuveenNoResetConsumo = false;
        entoncesSiactivatedResetConsumo = false;
        escogemosSiResetConsumo = true;
        escogemosNoResetConsumo = false;
        if (escogemosSiResetConsumo)
        {
         contadorconsumolargo = 0;
         distanciacontadorlargo = 0;

        }
       }
       if (entoncesNoactivatedResetConsumo ) {
        tft.setCursor(Mancho + 350, 100 , 4);
        tft.setTextColor(TFT_WHITE);  tft.setTextSize(1);
        resetear_consumo_medio = " Si ";
        tft.println(resetear_consumo_medio);
        tft.setCursor(Mancho + 350, 100 , 4);
        tft.setTextColor(TFT_BLACK);  tft.setTextSize(1);
        resetear_consumo_medio = " No ";
        tft.println(resetear_consumo_medio);
        estuveenSiResetConsumo = false;
        estuveenNoResetConsumo = true;
```

```
        entoncesNoactivatedResetConsumo = false;

        escogemosSiResetConsumo = false;
        escogemosNoResetConsumo = true;
        if (escogemosNoResetConsumo)
        {
         contadorconsumolargo = contadorconsumolargoprevio;
         distanciacontadorlargo = distanciacontadorlargoprevio;
        }
       }
       altohastaquesuelteslabajadaC = false;

     }
    }
    if (digitalRead(PBajar) == LOW)
    {


     altohastaquesuelteslabajadaC = true;

    }

   }
   if (iconfiguracion == 150)
   {
    if (digitalRead(PBajar) == HIGH)
    {
     if (altohastaquesuelteslabajadaC2) //impide que se vuelva loco y en una pulsacion la cuente como
20(exagerando).
     {
      mientrasnopulsesporsegundavezSnosaltaras = 0;
     }
     altohastaquesuelteslabajadaC2 = false;

    }
    esperaobligatoria = false;
    while (mientrasnopulsesporsegundavezSnosaltaras < 1) {
     //   altohastaquesuelteslabajadaC2=true;

     if (digitalRead(PRetroceso) == HIGH)
     {
      if (altohastaquesuelteslabajadaC3) //parte en la que sumas con la R
      {
       numerototaldeposito.datoI = numerototaldeposito.datoI + 1.0;
       if (numerototaldepositoprevia != numerototaldeposito.datoI)
       {
        tft.setCursor(Mancho + 220, 150 , 4);
        tft.setTextColor(TFT_WHITE);  tft.setTextSize(1);
        tft.print(numerototaldepositoprevia);
        tft.setCursor(Mancho + 220, 150 , 4);
        tft.setTextColor(TFT_BLACK);  tft.setTextSize(1);
        tft.print(numerototaldeposito.datoI);
```

```
        numerototaldepositoprevia = numerototaldeposito.datoI;
       }
       altohastaquesuelteslabajadaC3 = false;
      }
     }
     if (digitalRead(PRetroceso) == LOW)
     {


      altohastaquesuelteslabajadaC3 = true;

     }
     if (digitalRead(PConfirmar) == HIGH)
     {
      if (altohastaquesuelteslabajadaC4) //parte en la que sumas con la R
      {
       numerototaldeposito.datoI = numerototaldeposito.datoI - 1.0;
       if (numerototaldepositoprevia != numerototaldeposito.datoI)
       {
        tft.setCursor(Mancho + 220, 150 , 4);
        tft.setTextColor(TFT_WHITE);  tft.setTextSize(1);
        tft.print(numerototaldepositoprevia);
        tft.setCursor(Mancho + 220, 150 , 4);
        tft.setTextColor(TFT_BLACK);  tft.setTextSize(1);
        tft.print(numerototaldeposito.datoI);
        numerototaldepositoprevia = numerototaldeposito.datoI;
       }
       altohastaquesuelteslabajadaC4 = false;
      }
     }
     if (digitalRead(PConfirmar) == LOW)
     {


      altohastaquesuelteslabajadaC4 = true;

     }
     if (digitalRead(PBajar) == LOW)
     {
      //if(altohastaquesuelteslabajadaC2)//impide que se vuelva loco y en una pulsacion la cuente como
20(exagerando).
      //  {
      esperaobligatoria = true;
      //    }
      //   altohastaquesuelteslabajadaC2=false;

      //mientrasnopulsesporsegundavezSnosaltaras=2;
     }
     if (esperaobligatoria) {
      if (digitalRead(PBajar) == HIGH)
      {
```

```
        //if(altohastaquesuelteslabajadaC2)//impide que se vuelva loco y en una pulsacion la cuente como 20(exagerando).
        //  {
        mientrasnopulsesporsegundavezSnosaltaras = 1;
        //    }
        //   altohastaquesuelteslabajadaC2=false;

        //mientrasnopulsesporsegundavezSnosaltaras=2;
       }
      }
    }////////////////////////////aqui ACABA EL WHILE DONDE SE SUMA O SE RESTA HASTA QUE SE PULSE LA BAJADA OTRA VEZ

    if (digitalRead(PBajar) == LOW)
    {


      altohastaquesuelteslabajadaC2 = true;

    }
  }
  if (iconfiguracion == 200)
  {
    if (digitalRead(PBajar) == HIGH)
    {
      if (altohastaquesuelteslabajadaC) //impide que se vuelva loco y en una pulsacion la cuente como 20(exagerando).////////POR DIOS, CADA UNO CON SU PROPIO BLOQUEO, NO ME LA JODAS MAS
      {
        if (estuveenNo)
        { entoncesSiactivated = true;
        }
        if (estuveenSi)
        { entoncesNoactivated = true;
        }
        if ( entoncesSiactivated) {
          tft.setCursor(Mancho + 270, 200 , 4);
          tft.setTextColor(TFT_WHITE);  tft.setTextSize(1);
          recarga_gasolina = " No ";
          tft.println(recarga_gasolina);
          tft.setCursor(Mancho + 270, 200 , 4);
          tft.setTextColor(TFT_BLACK);  tft.setTextSize(1);
          recarga_gasolina = " Si ";
          tft.println(recarga_gasolina);
          estuveenSi = true;
          estuveenNo = false;
          entoncesSiactivated = false;
          escogemosSi = true;
          escogemosNo = false;
          if (escogemosSi)
          {
            deposito_total = deposito_total_maximo;
```

```
        }
       }
       if (entoncesNoactivated ) {
        tft.setCursor(Mancho + 270, 200 , 4);
        tft.setTextColor(TFT_WHITE);  tft.setTextSize(1);
        recarga_gasolina = " Si ";
        tft.println(recarga_gasolina);
        tft.setCursor(Mancho + 270, 200 , 4);
        tft.setTextColor(TFT_BLACK);  tft.setTextSize(1);
        recarga_gasolina = " No ";
        tft.println(recarga_gasolina);
        estuveenSi = false;
        estuveenNo = true;
        entoncesNoactivated = false;

        escogemosSi = false;
        escogemosNo = true;
        if (escogemosNo)
        {
         deposito_total = deposito_previo;
        }
       }
       altohastaquesuelteslabajadaC = false;

     }
    }
    if (digitalRead(PBajar) == LOW)
    {


     altohastaquesuelteslabajadaC = true;

    }


   }
  }

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////
  /////////TANTAS PAGINAS COMO NECESITES
  if (pagina2enactivo) {
   if (configurarpaginaenblanco) {
    tft.fillRect(Mancho, 0, 480 - Mancho, 320, TFT_WHITE);
    configurarpaginaenblanco = false;
    myFiles.load(69, 205, 411, 115, "kuja.RAW", 1, 0);

   }
   tft.setCursor(Mancho + 20, 50 , 4);
   tft.setTextColor(TFT_BLACK);  tft.setTextSize(1);
   tft.println(" Numero de etapas ");
   tft.setCursor(Mancho + 20, 100 , 4);
```

```
  tft.setTextColor(TFT_BLACK);  tft.setTextSize(1);
  tft.println(" Resetear Kilometraje?");
  tft.setCursor(Mancho, 150 , 4);
  tft.setTextColor(TFT_BLACK);  tft.setTextSize(1);
  //tft.println(" Elemento 3 ");
  tft.setCursor(Mancho, 200 , 4);
  tft.setTextColor(TFT_BLACK);  tft.setTextSize(1);
  // tft.println(" Elemento 4 ");
  tft.setCursor(Mancho + 270, 50 , 4);
  tft.setTextColor(TFT_BLACK);  tft.setTextSize(1);
  tft.println(numerodeetapas);
  tft.setCursor(Mancho + 350, 100 , 4);
  tft.setTextColor(TFT_BLACK);  tft.setTextSize(1);
  tft.print(reset_kilometraje);
  if (iconfiguracion == 50)
  {

    if (digitalRead(PBajar) == HIGH)
    {
      if (altohastaquesuelteslabajadaC2) //impide que se vuelva loco y en una pulsacion la cuente como
20(exagerando).
      {
        mientrasnopulsesporsegundavezSnosaltaras = 0;
      }
      altohastaquesuelteslabajadaC2 = false;

    }
    esperaobligatoria = false;
    while (mientrasnopulsesporsegundavezSnosaltaras < 1) {
      //    altohastaquesuelteslabajadaC2=true;

      if (digitalRead(PRetroceso) == HIGH)
      {
        if (altohastaquesuelteslabajadaC3) //parte en la que sumas con la R
        {
          numerodeetapas = numerodeetapas + 1;
          if (numerodeetapasprevia != numerodeetapas)
          {
            tft.setCursor(Mancho + 270, 50 , 4);
            tft.setTextColor(TFT_WHITE);  tft.setTextSize(1);
            tft.print(numerodeetapasprevia);
            tft.setCursor(Mancho + 270, 50 , 4);
            tft.setTextColor(TFT_BLACK);  tft.setTextSize(1);
            tft.print(numerodeetapas);
            numerodeetapasprevia = numerodeetapas;
          }
          altohastaquesuelteslabajadaC3 = false;
        }
      }
      if (digitalRead(PRetroceso) == LOW)
      {
```

```
  altohastaquesuelteslabajadaC3 = true;

 }
 if (digitalRead(PConfirmar) == HIGH)
 {
  if (altohastaquesuelteslabajadaC4) //parte en la que sumas con la R
  {
   numerodeetapas = numerodeetapas - 1;
   if (numerodeetapasprevia != numerodeetapas)
   {
    tft.setCursor(Mancho + 270, 50 , 4);
    tft.setTextColor(TFT_WHITE);  tft.setTextSize(1);
    tft.print(numerodeetapasprevia);
    tft.setCursor(Mancho + 270, 50 , 4);
    tft.setTextColor(TFT_BLACK);  tft.setTextSize(1);
    tft.print(numerodeetapas);
    numerodeetapasprevia = numerodeetapas;
   }
   altohastaquesuelteslabajadaC4 = false;
  }
 }
 if (digitalRead(PConfirmar) == LOW)
 {


  altohastaquesuelteslabajadaC4 = true;

 }
 if (digitalRead(PBajar) == LOW)
 {
  //if(altohastaquesuelteslabajadaC2)//impide que se vuelva loco y en una pulsacion la cuente como
20(exagerando).
  //  {
  esperaobligatoria = true;
  //    }
  //   altohastaquesuelteslabajadaC2=false;

  //mientrasnopulsesporsegundavezSnosaltaras=2;
 }
 if (esperaobligatoria) {
  if (digitalRead(PBajar) == HIGH)
  {
   //if(altohastaquesuelteslabajadaC2)//impide que se vuelva loco y en una pulsacion la cuente
como 20(exagerando).
   //  {
   mientrasnopulsesporsegundavezSnosaltaras = 1;
   //    }
   //   altohastaquesuelteslabajadaC2=false;

   //mientrasnopulsesporsegundavezSnosaltaras=2;
  }
```

```
      }
    }//////////////////////////aqui ACABA EL WHILE DONDE SE SUMA O SE RESTA HASTA QUE SE PULSE LA BAJADA OTRA VEZ

    if (digitalRead(PBajar) == LOW)
    {


      altohastaquesuelteslabajadaC2 = true;

    }

  }

  if (iconfiguracion == 100)
  {
    if (digitalRead(PBajar) == HIGH)
    {
      if (altohastaquesuelteslabajadaC) //impide que se vuelva loco y en una pulsacion la cuente como 20(exagerando).////////POR DIOS, CADA UNO CON SU PROPIO BLOQUEO, NO ME LA JODAS MAS
      {
        if (estuveenNoKilometraje)
        {
          entoncesSiactivatedKilometraje = true;
        }
        if (estuveenSiKilometraje)
        { entoncesNoactivatedKilometraje = true;
        }
        if ( entoncesSiactivatedKilometraje) {
          tft.setCursor(Mancho + 350, 100 , 4);
          tft.setTextColor(TFT_WHITE);  tft.setTextSize(1);
          reset_kilometraje = " No ";
          tft.println(reset_kilometraje);
          tft.setCursor(Mancho + 350, 100 , 4);
          tft.setTextColor(TFT_BLACK);  tft.setTextSize(1);
          reset_kilometraje = " Si ";
          tft.println(reset_kilometraje);
          estuveenSiKilometraje = true;
          estuveenNoKilometraje = false;
          entoncesSiactivatedKilometraje = false;
          escogemosSiKilometraje = true;
          escogemosNoResetConsumo = false;
          if (escogemosSiKilometraje)
          {

            distanciacontador = 0;
          }
        }
        if (entoncesNoactivatedKilometraje ) {
          tft.setCursor(Mancho + 350, 100 , 4);
          tft.setTextColor(TFT_WHITE);  tft.setTextSize(1);
          reset_kilometraje = " Si ";
```

```
          tft.println(reset_kilometraje);
          tft.setCursor(Mancho + 350, 100 , 4);
          tft.setTextColor(TFT_BLACK);  tft.setTextSize(1);
          reset_kilometraje = " No ";
          tft.println(reset_kilometraje);
          estuveenSiKilometraje = false;
          estuveenNoKilometraje = true;
          entoncesNoactivatedKilometraje = false;

          escogemosSiKilometraje = false;
          escogemosNoKilometraje = true;
          if (escogemosNoKilometraje)
          {

            distanciacontador = distanciacontadorprevio;
          }
        }
        altohastaquesuelteslabajadaC = false;

      }
    }
    if (digitalRead(PBajar) == LOW)
    {


      altohastaquesuelteslabajadaC = true;

    }


  }

 }
}
void movimientoconfiguracion() //tiene pinta de ser el que mueve arriba y abajo
{
 if (pagina1enactivo) {
   if (digitalRead(PConfirmar) == HIGH)
   {
     if (altohastaquesuelteslasubidaCon)
     {
       if (semaforo1)
       {
         //tft.setTextColor(TFT_WHITE);
         // tft.drawRightString(">", 90, iconfiguracion, 4); //VA A INDICAR EN QUE PUNTO NOS
ENCONTRAMOS
         tft.fillCircle(85, iconfiguracion + 10, 5, TFT_WHITE);
         iconfiguracion = iconfiguracion + 50;
         if (iconfiguracion > 200)
         {
           configurarpaginaenblanco = true;
           pagina1enactivo = false;
```

```
          pagina2enactivo = true;
          iconfiguracion = 50;
        }
        // tft.setTextColor(TFT_BLACK);
        // tft.drawRightString(">", 90, iconfiguracion, 4); //VA A INDICAR EN QUE PUNTO NOS
ENCONTRAMOS
        tft.fillCircle(85, iconfiguracion + 10, 5, TFT_BLACK);

        altohastaquesuelteslasubidaCon = false;
       }

     }
   }
   if (digitalRead(PConfirmar) == LOW)
   {
    altohastaquesuelteslasubidaCon = true;

   }
   if (digitalRead(PRetroceso) == HIGH)
   {
    if (altohastaquesuelteslabajada)
    {
      if (semaforo1)
      {
        //if (HIGH ==digitalRead(PBajar))
        //{ //i = 100;
        //tft.setTextColor(TFT_WHITE);
        //tft.drawRightString(">", 90, iconfiguracion, 4); //VA A INDICAR EN QUE PUNTO NOS
ENCONTRAMOS
        tft.fillCircle(85, iconfiguracion + 10, 5, TFT_WHITE);

        iconfiguracion = iconfiguracion - 50;


        if (iconfiguracion < 50)
        {
          iconfiguracion = 50;
        }
        //tft.setTextColor(TFT_BLACK);
        //tft.drawRightString(">", 90, iconfiguracion, 4); //VA A INDICAR EN QUE PUNTO NOS
ENCONTRAMOS
        tft.fillCircle(85, iconfiguracion + 10, 5, TFT_BLACK);

        altohastaquesuelteslabajada = false;
       }
     }
   }
   if (digitalRead(PRetroceso) == LOW)
   {
    altohastaquesuelteslabajada = true;
   }
 }
```

```
 ////////////////////////      PAGINA2 CONFIGURACION
 if (pagina2enactivo) {
  if (digitalRead(PConfirmar) == HIGH)
  {
   if (altohastaquesuelteslasubidaCon)
   {
    if (semaforo1)
    {
     //tft.setTextColor(TFT_WHITE);
     //tft.drawRightString(">", 90, iconfiguracion, 4); //VA A INDICAR EN QUE PUNTO NOS
ENCONTRAMOS
     tft.fillCircle(85, iconfiguracion + 10, 5, TFT_WHITE);

     iconfiguracion = iconfiguracion + 50;
     if (iconfiguracion > 200)
     {

      iconfiguracion = 200;
     }
     //tft.setTextColor(TFT_BLACK);
     //tft.drawRightString(">", 90, iconfiguracion, 4); //VA A INDICAR EN QUE PUNTO NOS
ENCONTRAMOS
     tft.fillCircle(85, iconfiguracion + 10, 5, TFT_BLACK);

     altohastaquesuelteslasubidaCon = false;
    }

   }
  }
  if (digitalRead(PConfirmar) == LOW)
  {
   altohastaquesuelteslasubidaCon = true;

  }
  if (digitalRead(PRetroceso) == HIGH)
  {
   if (altohastaquesuelteslabajada)
   {
    if (semaforo1)
    {
     //if (HIGH ==digitalRead(PBajar))
     //{ //i = 100;
     //tft.setTextColor(TFT_WHITE);
     //tft.drawRightString(">", 90, iconfiguracion, 4); //VA A INDICAR EN QUE PUNTO NOS
ENCONTRAMOS
     tft.fillCircle(85, iconfiguracion + 10, 5, TFT_WHITE);

     iconfiguracion = iconfiguracion - 50;


     if (iconfiguracion < 50)
     {
```

```
          pagina2enactivo = false;
          pagina1enactivo = true;
          iconfiguracion = 200;
          configurarpaginaenblanco = true;

        }
        //         tft.setTextColor(TFT_BLACK);
        //       tft.drawRightString(">", 90, iconfiguracion, 4); //VA A INDICAR EN QUE PUNTO NOS
ENCONTRAMOS
        tft.fillCircle(85, iconfiguracion + 10, 5, TFT_BLACK);

        altohastaquesuelteslabajada = false;
       }
     }
   }
   if (digitalRead(PRetroceso) == LOW)
   {
    altohastaquesuelteslabajada = true;
   }
 }

}
void movimiento() //tiene pinta de ser el que mueve arriba y abajo
{
 if (entre)
 {
  tft.fillRect(0, 0, Mancho, Malto, TFT_GREY);
  entre = false;
 }
 if (gps.satellites() == 255)
 {
  satelitecomprovacion = 1;
 }
 if (gps.satellites() < 4)
 {
  satelitecomprovacion = 2;
 }
 if ((gps.satellites() >= 4) && (gps.satellites() <= 8))
 {
  satelitecomprovacion = 3;

 }
 if ((gps.satellites() >= 9) && (gps.satellites() < 20))
 {
  satelitecomprovacion = 4;

 }
 if (satelitecomprovacionprevias != satelitecomprovacion)
 {
  if (gps.satellites() == 255)
  {
   myFiles.load(4, 236, 50, 64, "satelite1.RAW", 1, 0);
```

```
   satelitecomprovacionprevias = 1;
  }
  if (gps.satellites() < 4)
  {
   myFiles.load(4, 236, 50, 64, "satelite2.RAW", 1, 0);
   satelitecomprovacionprevias = 2;
  }
  if ((gps.satellites() >= 4) && (gps.satellites() <= 8))
  {
   myFiles.load(4, 236, 50, 64, "satelite3.RAW", 1, 0);
   satelitecomprovacionprevias = 3;

  }
  if ((gps.satellites() >= 9) && (gps.satellites() < 20))
  {
   myFiles.load(4, 236, 50, 64, "satelite4.RAW", 1, 0);
   satelitecomprovacionprevias = 4;

  }
}
if (entre1) {
  myFiles.load(0, 166, 69, 69, "engranajegris.RAW", 1, 0);
  myFiles.load(0, 96, 69, 69, "banderagris.RAW", 1, 0);
  myFiles.load(0, 26, 69, 69, "motoverde.RAW", 1, 0);



  entre1 = false;
}
unsigned long age, date, time;

gps.get_datetime(&date, &time, &age);

String entero = String(time);
String dhora = String(entero.charAt(0));
String uhora = String(entero.charAt(1));
String dmin = String(entero.charAt(2));
String umin = String(entero.charAt(3));
String horas = dhora + uhora;
String minutos = dmin + umin;
int horasi = horas.toInt();
minutosi = minutos.toInt();
if (horasi == 22)
{
  nuevahora = 0;
}
if (horasi == 23)
{
  nuevahora = 1;
}
if (horasi != 22 )
{
```

```
  if (horasi != 23)
  {
    nuevahora = horasi + 2;
  }

 }
 if (nuevahoraprevias != nuevahora)
 {
  tft.setCursor(4, 300 , 4);
  tft.setTextColor(TFT_GREY);  tft.setTextSize(1);
  tft.print(nuevahoraprevias);
  tft.setCursor(4, 300 , 4);
  tft.setTextColor(TFT_BLACK);  tft.setTextSize(1);
  tft.print(nuevahora); tft.print(":");
  nuevahoraprevias = nuevahora;
 }


 if (minutosiprevias != minutosi)
 {


  tft.setCursor(40, 300 , 4);
  tft.setTextColor(TFT_GREY);  tft.setTextSize(1);
  tft.print(minutosiprevias);
  tft.setCursor(40, 300 , 4);
  tft.setTextColor(TFT_BLACK);  tft.setTextSize(1);
  tft.print(minutosi);

  minutosiprevias = minutosi;


 }

 /*tft.setCursor(4, 300 , 4);
  tft.setTextColor(TFT_BLACK);  tft.setTextSize(1);
  tft.print(nuevahora); tft.print(":"); tft.print(minutosi); //imprime la hora*/
 if (digitalRead(PSubir) == HIGH)
 {
  if (altohastaquesuelteslasubidaMov) //impide que se vuelva loco y en una pulsacion la cuente como
20(exagerando).
  {

    i = i + 50;
    if (i > 200)
    {
      i = 100;

    }
    if (i == 100) ///posicion para la pagina de la configuracion
    {
```

```
      page = 2; //DEBES SIEMPRE HABILITAR LA PAGINA SIUGIENTE A LA TUYA PARA QUE LUEGO PUEDA
FUNCIONAR, PUESTO QUE LAS VARIABLES SE DESHABILITAN AL ENTRAR POR PRIMERA VEZ
      paginacargada3 = true;
    }
    if (i == 150) //posicion para la pagina general
    {
      page = 3;
      paginacargada4 = true;

    }
    if (i == 200) //posicion para la pagina del trial
    {
      page = 4;
      paginacargada2 = true;
    }

    /*     tft.setTextColor(TFT_YELLOW);
      tft.setCursor(100,i ,4);
      tft.println("a");
    */
    altohastaquesuelteslasubidaMov = false;

   }
  }
  if (digitalRead(PSubir) == LOW)
  {

   altohastaquesuelteslasubidaMov = true;

  }
}
//////////////////////////////////////////////////////
void vectordetiempo() //tiene pinta controla el tiempo del trial
{

  if (digitalRead(PBajar) == HIGH) //pulsamos el boton de comenzar el tiempo a contar
  {
   if (filtro) {
    entreporprimeravezalafuncionvectordetiempo = true;
    filtro = false;
   }
   if (altohastaquesuelteslabajada) //impide que se vuelva loco y en una pulsacion la cuente como
20(exagerando).
   {

    yapuedesreogereltiempo = true;//permite empezar a recoger los tiempos por los que pasas

    altohastaquesuelteslabajada = false;

   }
  }
  if (digitalRead(PBajar) == LOW)
```

```
 {
  if (!prohibidoelpaso1)
  {
   if (entreporprimeravezalafuncionvectordetiempo) {

    un_seg ();    //Va a la rutina un_seg

    contador();   //Va a la rutina contador

    escribe_hora();  //Va a la rutina escribe_hora
    un_segT ();    //Va a la rutina un_seg TOTALES

    //  contadorT();   //Va a la rutina contador TOTALES

    // escribe_horaT();  //Va a la rutina escribe_hora TOTALES
    entreporprimeravezalafuncionvectordetiempo = false;
    entradassucesivas = true;
   }
   if (entradassucesivas) { //constantemente entra para actualizar las horas


    un_seg ();    //Va a la rutina un_seg

    contador();   //Va a la rutina contador

    escribe_hora();  //Va a la rutina escribe_hora
    ///////////////LA VERGA SI ESTO VA
    un_segT ();    //Va a la rutina un_seg TOTALES

    contadorT();   //Va a la rutina contador TOTALES

    escribe_horaT();  //Va a la rutina escribe_hora TOTALES
    ////////////////////OJO AL COJO
    if (yapuedesreogereltiempo) {
     if (-1 < itrial < numerodeetapas ) { //aqui te metes los tiempos al pulsar //esta puesto para tres vueltas, el -1 es para cepillarse la priemra pulsacion, pues en el tiempo 0 no queremos contarlo como vuelta
       Vhd[itrial] = d_hora; //cambiar 20 por una variable X
       Vmd[itrial] = d_minuto;
       Vsd[itrial] = d_segundo;
       Vhu[itrial] = u_hora; //cambiar 20 por una variable X
       Vmu[itrial] = u_minuto;
       Vsu[itrial] = u_segundo;
       tft.setCursor(390 , 120, 4);
       tft.setTextColor(TFT_WHITE);  tft.setTextSize(4);
       tft.print(trialprevio);
       tft.setCursor(390 , 120, 4);
       tft.setTextColor(rojo);  tft.setTextSize(4);
       tft.print(trial);
       trialprevio = trial;
       trial = trial + 1;
```

```
        }

        itrial = itrial + 1;
        yapuedesreogereltiempo = false;
        //la novedad
        //timer1=0;
        //timer2=0;
        //reseteamos las variables para el siguiente conteo
        u_segundo = 0;

        u_minuto = 0;
        u_hora = 0;
        d_segundo = 0;
        d_minuto = 0;
        d_hora = 0;

        ////////
      }

    }
   }
   if (itrial == numerodeetapas)
   { //si llegas al final de las vueltas, debes dejar de seguir contando, ademas, habras sumado un trial de mas, por lo que hay que restarlo y prohibir el paso a mas conteos
     if (!prohibidoelpaso1) {
       contadorv = itrial - 1;///es decir, lo maximo a los que puedes contar, es el trial menos 1, pues si no te pasarias con una vuelta de mas,
       prohibidoelpaso1 = true;

     }
     if (digitalRead(PRetroceso) == HIGH)
     {
       if (altohastaquesuelteselretroceso) {
         contadorv = contadorv + 1;
         if (contadorv == itrial) {
           contadorv = itrial - 1; ///cuando vayas por los vectores, por mas que le pulses hacia arriba para ver mas valores, no podras,
         }
         escribe_marcador();
         altohastaquesuelteselretroceso = false;
       }

     }
     if (digitalRead(PRetroceso) == LOW)
     {
       altohastaquesuelteselretroceso = true;

     }
     if (digitalRead(PConfirmar) == HIGH)
     {
       if (altohastaquesuelteslaconfirmacion) {
```

```
        contadorv = contadorv - 1;
        if (contadorv < 0) { //por mas que le pulses hacia abajo, para ver mas valores no podras,
          contadorv = 0;
        }
        escribe_marcador();
        altohastaquesuelteslaconfirmacion = false;
       }
     }
     if (digitalRead(PConfirmar) == LOW)
     {
       altohastaquesuelteslaconfirmacion = true;

     }

   }
   altohastaquesuelteslabajada = true;
  }
}
//////////////////////////////////////////////////////
void escribe_marcador() { //Rutina para escribir la hora en el LCD
  int xt = 80;
  if (segundaveztime) {
    for (int as = 0; as < numerodeetapas; as++) { //repasaras el vector del 0 al 2 luego 0 1 y 2 seran los
puntos de la carrera
      Vhdprevias[as] = Vhd[itrial - 1]; //cambiar 20 por una variable X//DEBEN SER COMENTARIOS
CHORRA//como trial se quedaba con uno de mas por eso se le resta uno para ser su maximo
      Vmdprevias[as] = Vmd[itrial - 1];
      Vsdprevias[as] = Vsd[itrial - 1];
      Vhuprevias[as] = Vhu[itrial - 1]; //cambiar 20 por una variable X
      Vmuprevias[as] = Vmu[itrial - 1];
      Vsuprevias[as] = Vsu[itrial - 1];
    }

    segundaveztime = false;
  }/////esto copiaba todas las previas al valor de la que tuviera el verdadero, de tal modo que pude
borrar luego las variables en blanco con facilidad, es decir, todos los minutos horas y segundos,
adquieren el mismo valor en cualquiera de sus posiciones, asi, podemos borrar el numero ultimo cuando
empieces a desplazarte por el marcador


  /*  if (Vhdprevias[contadorv] != Vhd[contadorv]) {
      tft.setCursor(xt, 100, tamhoras);
      tft.setTextColor(TFT_WHITE);  tft.setTextSize(tamtextohoras);
      tft.print(Vhdprevias[contadorv]);
      tft.setCursor(xt, 100, tamhoras);
      tft.setTextColor(TFT_BLACK);  tft.setTextSize(tamtextohoras);
      tft.print(Vhd[contadorv]);
      for (int as = 0; as < 3; as++) {//como digimos, rellenamos todas las posiciones de las previas con el
valor del la posicion que se representa, para que cuando deba cambiar, este en la posicion que este,
siempre tengael numero accesible
        Vhdprevias[as] = Vhd[contadorv];
      }
```

```
  }*/
 if (contadorv != contadorvprevio) {
  tft.setCursor(390 , 120, 4);
  tft.setTextColor(TFT_WHITE);  tft.setTextSize(4);
  tft.print(contadorvprevio + 1);
  tft.setCursor(390 , 120, 4);
  tft.setTextColor(rojo);  tft.setTextSize(4);
  tft.print(contadorv + 1);
  contadorvprevio = contadorv;
 }
 if (Vhuprevias[contadorv] != Vhu[contadorv]) {
  tft.setCursor(xt , 40, tamhoras);
  tft.setTextColor(TFT_WHITE);  tft.setTextSize(tamtextohoras);
  tft.print(Vhuprevias[contadorv]);
  tft.setCursor(xt , 40, tamhoras);
  tft.setTextColor(TFT_BLACK);  tft.setTextSize(tamtextohoras);
  tft.print(Vhu[contadorv]);  //tft.print(":");
  for (int as = 0; as < numerodeetapas; as++) {
    Vhuprevias[as] = Vhu[contadorv];
  }

 }
 if (Vmdprevias[contadorv] != Vmd[contadorv]) {
  tft.setCursor(xt + 60, 40, tamhoras);
  tft.setTextColor(TFT_WHITE);  tft.setTextSize(tamtextohoras);
  tft.print(Vmdprevias[contadorv]);
  tft.setCursor(xt + 60, 40, tamhoras);
  tft.setTextColor(TFT_BLACK);  tft.setTextSize(tamtextohoras);
  tft.print(Vmd[contadorv]);
  for (int as = 0; as < numerodeetapas; as++) {
    Vmdprevias[as] = Vmd[contadorv];
  }

 }
 if (Vmuprevias[contadorv] != Vmu[contadorv]) {
  tft.setCursor(xt + 120, 40, tamhoras);
  tft.setTextColor(TFT_WHITE);  tft.setTextSize(tamtextohoras);
  tft.print(Vmuprevias[contadorv]);
  tft.setCursor(xt + 120, 40, tamhoras);
  tft.setTextColor(TFT_BLACK);  tft.setTextSize(tamtextohoras);
  tft.print(Vmu[contadorv]); tft.print(":");
  for (int as = 0; as < numerodeetapas; as++) {
    Vmuprevias[as] = Vmu[contadorv];
  }

 }
 if ( Vsdprevias[contadorv] != Vsd[contadorv]) {
  tft.setCursor(xt + 180, 40, tamhoras);
  tft.setTextColor(TFT_WHITE);  tft.setTextSize(tamtextohoras);
  tft.print( Vsdprevias[contadorv]);
  tft.setCursor(xt + 180, 40, tamhoras);
  tft.setTextColor(TFT_BLACK);  tft.setTextSize(tamtextohoras);
```

```
  tft.print(Vsd[contadorv]);
  for (int as = 0; as < numerodeetapas; as++) {
    Vsdprevias[as] = Vsd[contadorv];
  }

 }
 if ( Vsuprevias[contadorv] != Vsu[contadorv]) {
  tft.setCursor(xt + 240, 40, tamhoras);
  tft.setTextColor(TFT_WHITE);  tft.setTextSize(tamtextohoras);
  tft.print( Vsuprevias[contadorv]);
  tft.setCursor(xt + 240, 40, tamhoras);
  tft.setTextColor(TFT_BLACK);  tft.setTextSize(tamtextohoras);
  tft.print(Vsu[contadorv]);
  for (int as = 0; as < numerodeetapas; as++) {
    Vsuprevias[as] = Vsu[contadorv];
  }

 }

}

void Page2()
{

 if (paginacargada2)
 {
  contadorconsumolargoprevio = contadorconsumolargo;
  distanciacontadorlargoprevio = distanciacontadorlargo;
  distanciacontadorprevio = distanciacontador;
  entre3 = true;
  entre4 = true;
  entre5 = true;
  deposito_total2 = deposito_total;
  Vhd[100] = {0}; //cambiar 20 por una variable X
  Vmd[100] = {0};
  Vsd[100] = {0};
  Vhu[100] = {0}; //cambiar 20 por una variable X
  Vmu[100] = {0};
  Vsu[100] = {0};
  Vhdprevias[100] = {0}; //cambiar 20 por una variable X
  Vmdprevias[100] = {0};
  Vsdprevias[100] = {0};
  Vhuprevias[100] = {0}; //cambiar 20 por una variable X
  Vmuprevias[100] = {0};
  Vsuprevias[100] = {0};
  itrial = -1;
  u_horaprevias = 0;          //Cargamos un 0 en la variable "u_hora"
  u_minutoprevias = 0;        //Cargamos un 0 en la variable "u_minuto"
  u_segundoprevias = 0;       //Cargamos un 0 en la variable "u_segundo"
  d_horaprevias = 0;          //Cargamos un 0 en la variable "d_hora"
  d_minutoprevias = 0;        //Cargamos un 0 en la variable "d_minuto"
  d_segundoprevias = 0;       //Cargamos un 0 en la variable "d_segundo"
```

```
  u_hora = 0;          //Cargamos un 0 en la variable "u_hora"
  u_minuto = 0;        //Cargamos un 0 en la variable "u_minuto"
  u_segundo = 0;       //Cargamos un 0 en la variable "u_segundo"
  d_hora = 0;          //Cargamos un 0 en la variable "d_hora"
  d_minuto = 0;        //Cargamos un 0 en la variable "d_minuto"
  d_segundo = 0;       //Cargamos un 0 en la variable "d_segundo"
  u_horapreviasT = 0;          //Cargamos un 0 en la variable "u_hora"
  u_minutopreviasT = 0;        //Cargamos un 0 en la variable "u_minuto"
  u_segundopreviasT = -1;       //Cargamos un 0 en la variable "u_segundo"
  d_horapreviasT = 0;          //Cargamos un 0 en la variable "d_hora"
  d_minutopreviasT = 0;        //Cargamos un 0 en la variable "d_minuto"
  d_segundopreviasT = 0;       //Cargamos un 0 en la variable "d_segundo"
  u_horaT = 0;          //Cargamos un 0 en la variable "u_hora"
  u_minutoT = 0;        //Cargamos un 0 en la variable "u_minuto"
  u_segundoT = -1;       //Cargamos un 0 en la variable "u_segundo"
  u_segundoTC = 0;
  d_horaT = 0;          //Cargamos un 0 en la variable "d_hora"
  d_minutoT = 0;        //Cargamos un 0 en la variable "d_minuto"
  d_segundoT = 0;       //Cargamos un 0 en la variable "d_segundo"
  contadorv = 0;
  contadorvprevio = 2;
  trial = 0;
  trialprevio = -1;
  timer1 = 0;
  timer2 = 0;
  timer1T = 0;
  timer2T = 0;
  primeraveztime = true;
  primeraveztimeT = true;

  segundaveztime = true;
  solounavezcomootrastantasya = true;
  entradassucesivas = false;
  prohibidoelpaso1 = false; ///variables para recargar el trial
  primeraveztimeT = true; ///variables para recargar el trial
  segundaveztime = true; ///variables para recargar el trial
  primeraveztime = true; ///variables para recargar el trial
  filtro = true; ///variables para recargar el trial
  entreporprimeravezalafuncionvectordetiempo = false;
  tft.fillRect(Mancho, 0, 480 - Mancho, 320, TFT_WHITE);


  int Pos_X5 = 240;
  int Pos_Y5 = 0;// esta sera nuestra i
  int TAM5 = 1;
  int XC5 = 104;
  int YC5 = 40;
  int TFont5 = 4;
  //tft.fillRoundRect(Pos_X5-(XC5*TAM5)+53*TAM5, Pos_Y5-(10*TAM5), XC5*TAM5, YC5*TAM5, 7,
TFT_BLUE); // Base
  tft.setCursor(Pos_X5 - 50 * TAM5, Pos_Y5 , TFont5);
  tft.setTextColor(rojo, verde1);  tft.setTextSize(TAM5);
```

```
   tft.println(" Configuracion ");
   myFiles.load(0, 166, 69, 69, "engranajeverde.RAW", 1, 0);
   myFiles.load(0, 96, 69, 69, "banderagris.RAW", 1, 0);
   //myFiles.load(0, 26, 69, 69, "motogris.RAW", 1, 0);
   paginacargada2 = false;

  }
}
void Page3()
{
  if (paginacargada3)
  {
   deposito_total2 = deposito_total;

   Vhd[100] = {0}; //cambiar 20 por una variable X
   Vmd[100] = {0};
   Vsd[100] = {0};
   Vhu[100] = {0}; //cambiar 20 por una variable X
   Vmu[100] = {0};
   Vsu[100] = {0};
   Vhdprevias[100] = {0}; //cambiar 20 por una variable X
   Vmdprevias[100] = {0};
   Vsdprevias[100] = {0};
   Vhuprevias[100] = {0}; //cambiar 20 por una variable X
   Vmuprevias[100] = {0};
   Vsuprevias[100] = {0};
   itrial = -1;
   u_horaprevias = 0;          //Cargamos un 0 en la variable "u_hora"
   u_minutoprevias = 0;        //Cargamos un 0 en la variable "u_minuto"
   u_segundoprevias = 0;       //Cargamos un 0 en la variable "u_segundo"
   d_horaprevias = 0;          //Cargamos un 0 en la variable "d_hora"
   d_minutoprevias = 0;        //Cargamos un 0 en la variable "d_minuto"
   d_segundoprevias = 0;       //Cargamos un 0 en la variable "d_segundo"
   u_hora = 0;          //Cargamos un 0 en la variable "u_hora"
   u_minuto = 0;        //Cargamos un 0 en la variable "u_minuto"
   u_segundo = 0;       //Cargamos un 0 en la variable "u_segundo"
   d_hora = 0;          //Cargamos un 0 en la variable "d_hora"
   d_minuto = 0;        //Cargamos un 0 en la variable "d_minuto"
   d_segundo = 0;       //Cargamos un 0 en la variable "d_segundo"
   u_horapreviasT = 0;          //Cargamos un 0 en la variable "u_hora"
   u_minutopreviasT = 0;        //Cargamos un 0 en la variable "u_minuto"
   u_segundopreviasT = -1;       //Cargamos un 0 en la variable "u_segundo"
   d_horapreviasT = 0;          //Cargamos un 0 en la variable "d_hora"
   d_minutopreviasT = 0;        //Cargamos un 0 en la variable "d_minuto"
   d_segundopreviasT = 0;       //Cargamos un 0 en la variable "d_segundo"
   u_horaT = 0;          //Cargamos un 0 en la variable "u_hora"
   u_minutoT = 0;        //Cargamos un 0 en la variable "u_minuto"
   u_segundoT = -1;       //Cargamos un 0 en la variable "u_segundo"
   u_segundoTC = 0;
   d_horaT = 0;          //Cargamos un 0 en la variable "d_hora"
   d_minutoT = 0;        //Cargamos un 0 en la variable "d_minuto"
   d_segundoT = 0;       //Cargamos un 0 en la variable "d_segundo"
```

```
  contadorv = 0;
  trial = 0;
  trialprevio = -1;
  contadorvprevio = 2;
  timer1 = 0;
  timer2 = 0;
  timer1T = 0;
  timer2T = 0;
  primeraveztime = true;
  primeraveztimeT = true;
  pagina1enactivo = true;
  pagina2enactivo = false;
  iconfiguracion = 50;
  configurarpaginaenblanco = true;
  segundaveztime = true;
  solounavezcomootrastantasya = true;
  entradassucesivas = false;
  prohibidoelpaso1 = false; ///variables para recargar el trial
  segundaveztime = true; ///variables para recargar el trial
  primeraveztimeT = true; ///variables para recargar el trial
  primeraveztime = true; ///variables para recargar el trial
  entreporprimeravezalafuncionvectordetiempo = false;
  filtro = true; //para que asi se recargue el tiempo otra vez que si no pifostio en la funcion vector de
tiempo  y escribir temperatura del trial
  tft.fillRect(Mancho, 0, 480 - Mancho, 320, TFT_WHITE);
  intervalodedondevengo = 1;
  intervalodedondevengoR = 1;
  intervalodedondevengoV = 1;
  intervalodedondevengoD = 1; //para el color verde
  intervalodedondevengoDR = 14; //para el color rojo
  intervalodedondevengoDN = 7; //para el color naranja
  //tft.fillRoundRect(Pos_X5-(XC5*TAM5)+53*TAM5, Pos_Y5-(10*TAM5), XC5*TAM5, YC5*TAM5, 7,
TFT_BLUE); // Base
  //tft.setCursor(Pos_X5-50*TAM5,Pos_Y5 ,TFont5);
  // tft.setTextColor(rojo,verde1);  tft.setTextSize(TAM5);
  // tft.println(" Ciudad ");
  // tft.fillRoundRect(Pos_Xnum-(XCnum*TAMnum)+26*TAMnum, Pos_Ynum-(5*TAMnum),
XCnum*TAMnum, YCnum*TAMnum, 7, TFT_BLUE); // Base numerica
  tft.drawRoundRect(bordex, bordey, anchuraP3, alturaP3, 5, rojo);
  temperaturaprevias = 0; //redeclaro un cero PARA QUE CUANDO CARGE LA PAGINA OTRA VEZ VEA
QUE SON DISTINTOS LOS NUMEROS Y LOS VUELVA A IMPRIMIR, DE OTRO MODO SERIA EN BLANCO
SIEMPRE SI EL CAMBIO NO ES SUSTANCIAL
  tensionprevias = 0;
  depositoprevias = 0;
  tft.setCursor(240, 100, 4);
  tft.setTextColor(TFT_BLACK);  tft.setTextSize(1);
  tft.print("RPM");
  tft.setCursor(283, 160, 4);
  tft.setTextColor(TFT_BLACK);  tft.setTextSize(1);
  tft.print("km/h");
  myFiles.load(0, 166, 69, 69, "engranajegris.RAW", 1, 0);
  //  seguir dejandola comentada // myFiles.load(0, 96, 69, 69, "banderagris.RAW", 1, 0);
```

```
   myFiles.load(0, 26, 69, 69, "motoverde.RAW", 1, 0);
   paginacargada3 = false; //evita recargamientos inutiles
   tft.drawRoundRect(depx - 1, depy - 2, 42/*ancho*/, 96/*alto*/, 5, TFT_BLACK); //cuadrado para el
borde de la<s barras
   tft.drawRoundRect(depx - 2, depy - 3, 44/*ancho*/, 98/*alto*/, 5, TFT_BLACK); //cuadrado para el
borde de la<s barras
   tft.drawRoundRect(batx - 1, baty - 2, 42/*ancho*/, 96/*alto*/, 5, TFT_BLACK); //cuadrado para el
borde de la<s barras
   tft.drawRoundRect(batx - 2, baty - 3, 44/*ancho*/, 98/*alto*/, 5, TFT_BLACK); //cuadrado para el
borde de la<s barras
   tft.drawRoundRect(tempx - 1, tempy - 2, 42/*ancho*/, 96/*alto*/, 5, TFT_BLACK); //cuadrado para el
borde de la<s barras
   tft.drawRoundRect(tempx - 2, tempy - 3, 44/*ancho*/, 98/*alto*/, 5, TFT_BLACK); //cuadrado para el
borde de la<s barras
   myFiles.load(245, 240 , 40, 40, "termometro2.RAW", 1, 0);
   myFiles.load(360, 160, 40, 40, "bateria.RAW", 1, 0);///par la bateria
   myFiles.load(430, 160, 40, 40, "gasolinera.RAW", 1, 0);///para la gasolina
   EEPROM.write(4, numerototalmarcha.datoB[0]);
   EEPROM.write(5, numerototalmarcha.datoB[1]);
   EEPROM.write(6, numerototaldeposito.datoB[0]);
   EEPROM.write(7, numerototaldeposito.datoB[1]);
   EEPROM.write(8, numerototaldeposito.datoB[2]);
   EEPROM.write(9, numerototaldeposito.datoB[3]);
   tft.setCursor(80 , 240, 4);
   tft.setTextColor(TFT_BLACK);  tft.setTextSize(1);
   tft.print("T ext:");
   tft.setCursor(220, 240, 4);
   tft.setTextColor(TFT_BLACK);  tft.setTextSize(1);
   tft.print("C");
   tft.setCursor(72 , 260, 4);
   tft.setTextColor(TFT_BLACK);  tft.setTextSize(1);
   tft.print("Aut. I");
   tft.setCursor(220, 260, 4);
   tft.setTextColor(TFT_BLACK);  tft.setTextSize(1);
   tft.print("km");
   tft.setCursor(72 , 280, 4);
   tft.setTextColor(TFT_BLACK);  tft.setTextSize(1);
   tft.print("Aut. M");
   tft.setCursor(220, 280, 4);
   tft.setTextColor(TFT_BLACK);  tft.setTextSize(1);
   tft.print("km");




  }


}
void Page4()
{
```

```
 if (paginacargada4)
 {
  entre3 = true;
  entre4 = true;
  entre5 = true;
  deposito_total2 = deposito_total;
  estuveenNo = false;
  entoncesSiactivated = false;
  estuveenSi = true;
  entoncesNoactivated = false;
  recarga_gasolina = " No ";
  deposito_previo = deposito_total;

  pagina1enactivo = true;
  pagina2enactivo = false;
  configurarpaginaenblanco = true;
  iconfiguracion = 50;
  tft.fillRect(Mancho, 0, 480 - Mancho, 320, TFT_WHITE);
  int Pos_X5 = 240;
  int Pos_Y5 = 50;// esta sera nuestra i
  int TAM5 = 1;
  int XC5 = 104;
  int YC5 = 40;
  int TFont5 = 4;
  //tft.fillRoundRect(Pos_X5-(XC5*TAM5)+53*TAM5, Pos_Y5-(10*TAM5), XC5*TAM5, YC5*TAM5, 7,
TFT_BLUE); // Base  tft.setCursor(Pos_X5-50*TAM5,Pos_Y5 ,TFont5);
  /*tft.setCursor(75, 50, 4);
    tft.setTextColor(TFT_BLACK);  tft.setTextSize(1);
    tft.print("Distancia");
    tft.setCursor(180, 50, 4);
    tft.setTextColor(TFT_BLACK);  tft.setTextSize(1);
    tft.print("Tiempo");*/
  tft.setCursor(370, 15, 4);
  tft.setTextColor(TFT_BLACK);  tft.setTextSize(1);
  tft.print("VUELTA");
  tft.setCursor(75, 10, 4);
  tft.setTextColor(TFT_BLACK);  tft.setTextSize(1);
  tft.print("Tiempo  ");    tft.print("Vuelta");
  // tft.setCursor(75, 30, 4);
  // tft.setTextColor(TFT_BLACK);  tft.setTextSize(1);

  tft.setCursor(75, 150, 4);
  tft.setTextColor(TFT_BLACK);  tft.setTextSize(1);
  tft.print("Tiempo  ");
  // tft.setCursor(75, 140, 4);
  //tft.setTextColor(TFT_BLACK);  tft.setTextSize(1);
  tft.print("Total");
  //tft.drawRightString(">",100,i,1); //VA A INDICAR EN QUE PUNTO NOS ENCONTRAMO
  //  semaforo=false;
  //   myFiles.load(0, 166, 69, 69, "engranajegris.RAW", 1, 0);
  myFiles.load(0, 96, 69, 69, "banderaverde.RAW", 1, 0);
```

```
   myFiles.load(0, 26, 69, 69, "motogris.RAW", 1, 0);
   tft.setCursor(Pos_XnumP4 + 90, Pos_YnumP4 , 4);
   tft.setTextColor(TFT_BLACK);  tft.setTextSize(TAMnumRP4);
   tft.print("km");
   tft.setCursor(Pos_XnumRP4 , Pos_YnumRP4 , 4);
   tft.setTextColor(TFT_BLACK);  tft.setTextSize(TAMnumRP4);
   tft.print("Cons. Inst.");
   tft.setCursor(Pos_XnumRP4, 300 , 4);
   tft.setTextColor(TFT_BLACK);  tft.setTextSize(TAMnumRP4);
   tft.print("Cons. Med.");
   paginacargada4 = false;
   //   if (!eleccioncedeelpaso)
   // {
   // eleccionesperacambio=false;
   // }
  }
  //}
  // }

}
////////////////////////////////////////////////////////////////////////EXPERIMENTO CON TIEMPOS
TOTALES
void un_segT() {   //Rutina para cada segundo

  timer2T = (millis() / 1000);
  if ( timer1T != timer2T ) {
    timer1T = timer2;

    u_segundoT = u_segundoT + 1; // unidades de segundo se incrementa cada segundo
  }
}

void contadorT() { // Rutina para el reloj

  if ( u_segundoT == 10 ) {
    u_segundoT = 0;
    d_segundoT++;
  }

  if ( ( d_segundoT == 6 ) && ( u_segundoT == 0 ) ) {
    d_segundoT = 0;
    u_minutoT++;
  }


  // Rutina de minutos

  if ( u_minutoT == 10 ) {
    u_minutoT = 0;
    d_minutoT++;
  }
```

```
 if ( ( d_minutoT == 6 ) && ( u_minutoT == 0 ) ) {
  d_minutoT = 0;
  u_horaT++;
 }
 // Rutina de horas

 if ( u_horaT == 10 ) {
  u_horaT = 0;
  d_horaT++;
 }

 if ( (d_horaT == 2) &&  (u_horaT == 4) ) {
  u_horaT = 0;
  d_horaT = 0;
 }


}

void escribe_horaT() { //Rutina para escribir la hora en la pantalla //recuerdas qyue las rutinas
anteriores las sacaste de internet, no se te olvide de incluirlas en el zotero
 int xt = 80;
 if (primeraveztimeT) {
  /* tft.setCursor(xt, 150, tamhoras);
    tft.setTextColor(TFT_BLACK);  tft.setTextSize(tamtextohoras);
    tft.print(d_horaT);*/
  tft.setCursor(xt , 170, tamhoras);
  tft.setTextColor(TFT_BLACK);  tft.setTextSize(tamtextohoras);
  tft.print(u_horaT);
  tft.setCursor(xt + 55, 200, 4 );

  tft.setTextColor(rojo);  tft.setTextSize(3 );
  tft.print(",");
  tft.setCursor(xt + 60, 170, tamhoras);
  tft.setTextColor(TFT_BLACK);  tft.setTextSize(tamtextohoras);
  tft.print(d_minutoT);
  tft.setCursor(xt + 120, 170, tamhoras);
  tft.setTextColor(TFT_BLACK);  tft.setTextSize(tamtextohoras);
  tft.print(u_minutoT);
  tft.setCursor(xt + 170, 200, 4 );
  tft.setTextColor(rojo);  tft.setTextSize(3 );
  tft.print(",");
  tft.setCursor(xt + 180, 170, tamhoras);
  tft.setTextColor(TFT_BLACK);  tft.setTextSize(tamtextohoras);
  tft.print(d_segundoT);
  tft.setCursor(xt + 240, 170, tamhoras);
  tft.setTextColor(TFT_BLACK);  tft.setTextSize(tamtextohoras);
  tft.print(u_segundoTC);
  primeraveztimeT = false;
 }
 /*if (d_horapreviasT != d_horaT) {
  tft.setCursor(xt, 200, tamhoras);
```

```
  tft.setTextColor(TFT_WHITE);  tft.setTextSize(tamtextohoras);
  tft.print(d_horapreviasT);
  tft.setCursor(xt, 200, tamhoras);
  tft.setTextColor(TFT_BLACK);  tft.setTextSize(tamtextohoras);
  tft.print(d_horaT);
  d_horapreviasT = d_horaT;
  }*/
 if (u_horapreviasT != u_horaT) {
  tft.setCursor(xt , 170, tamhoras);
  tft.setTextColor(TFT_WHITE);  tft.setTextSize(tamtextohoras);
  tft.print(u_horapreviasT);
  tft.setCursor(xt , 170, tamhoras);
  tft.setTextColor(TFT_BLACK);  tft.setTextSize(tamtextohoras);
  tft.print(u_horaT); // tft.print(":");
  u_horapreviasT = u_horaT;
 }
 if (d_minutopreviasT != d_minutoT) {
  tft.setCursor(xt + 60, 170, tamhoras);
  tft.setTextColor(TFT_WHITE);  tft.setTextSize(tamtextohoras);
  tft.print(d_minutopreviasT);
  tft.setCursor(xt + 60, 170, tamhoras);
  tft.setTextColor(TFT_BLACK);  tft.setTextSize(tamtextohoras);
  tft.print(d_minutoT);
  d_minutopreviasT = d_minutoT;
 }
 if (u_minutopreviasT != u_minutoT) {
  tft.setCursor(xt + 120, 170, tamhoras);
  tft.setTextColor(TFT_WHITE);  tft.setTextSize(tamtextohoras);
  tft.print(u_minutopreviasT);
  tft.setCursor(xt + 120, 170, tamhoras);
  tft.setTextColor(TFT_BLACK);  tft.setTextSize(tamtextohoras);
  tft.print(u_minutoT);// tft.print(":");
  u_minutopreviasT = u_minutoT;
 }
 if (d_segundopreviasT != d_segundoT) {
  tft.setCursor(xt + 180, 170, tamhoras);
  tft.setTextColor(TFT_WHITE);  tft.setTextSize(tamtextohoras);
  tft.print(d_segundopreviasT);
  tft.setCursor(xt + 180, 170, tamhoras);
  tft.setTextColor(TFT_BLACK);  tft.setTextSize(tamtextohoras);
  tft.print(d_segundoT);
  d_segundopreviasT = d_segundoT;
 }
 if (u_segundopreviasT != u_segundoT) {
  tft.setCursor(xt + 240, 170, tamhoras);
  tft.setTextColor(TFT_WHITE);  tft.setTextSize(tamtextohoras);
  tft.print(u_segundopreviasT);
  tft.setCursor(xt + 240, 170, tamhoras);
  tft.setTextColor(TFT_BLACK);  tft.setTextSize(tamtextohoras);
  tft.print(u_segundoT);
  u_segundopreviasT = u_segundoT;
 }
```

```
}

//////////////////////////////////////////////////////////////////////ARRIBA ESTA EL EPERIMENTO
void un_seg() {   //Rutina para cada segundo

  timer2 = (millis() / 1000);
  if ( timer1 != timer2 ) {
    timer1 = timer2;

    u_segundo = u_segundo + 1; // unidades de segundo se incrementa cada segundo
  }
}

void contador() { // Rutina para el reloj

  if ( u_segundo == 10 ) {
    u_segundo = 0;
    d_segundo++;
  }

  if ( ( d_segundo == 6 ) && ( u_segundo == 0 ) ) {
    d_segundo = 0;
    u_minuto++;
  }


  // Rutina de minutos

  if ( u_minuto == 10 ) {
    u_minuto = 0;
    d_minuto++;
  }

  if ( ( d_minuto == 6 ) && ( u_minuto == 0 ) ) {
    d_minuto = 0;
    u_hora++;
  }
  // Rutina de horas

  if ( u_hora == 10 ) {
    u_hora = 0;
    d_hora++;
  }

  if ( (d_hora == 2) &&  (u_hora == 4) ) {
    u_hora = 0;
    d_hora = 0;
  }


}
```

```
void escribe_hora() { //Rutina para escribir la hora en el LCD
  int xt = 80;
  if (primeraveztime) {
    //tft.setCursor(xt, 100, tamhoras);
    //tft.setTextColor(TFT_BLACK);  tft.setTextSize(tamtextohoras);
    //tft.print(d_hora);
    tft.setCursor(xt , 40, tamhoras);
    tft.setTextColor(TFT_BLACK);  tft.setTextSize(tamtextohoras);
    tft.print(u_hora);
    tft.setCursor(xt + 55, 79, 4 );

    tft.setTextColor(rojo);  tft.setTextSize(3 );
    tft.print(",");
    tft.setCursor(xt + 60, 40, tamhoras);
    tft.setTextColor(TFT_BLACK);  tft.setTextSize(tamtextohoras);
    tft.print(d_minuto);
    tft.setCursor(xt + 120, 40, tamhoras);
    tft.setTextColor(TFT_BLACK);  tft.setTextSize(tamtextohoras);
    tft.print(u_minuto);
    tft.setCursor(xt + 170, 79, 4 );
    tft.setTextColor(rojo);  tft.setTextSize(3 );
    tft.print(",");
    tft.setCursor(xt + 180, 40, tamhoras);
    tft.setTextColor(TFT_BLACK);  tft.setTextSize(tamtextohoras);
    tft.print(d_segundo);
    tft.setCursor(xt + 240, 40, tamhoras);
    tft.setTextColor(TFT_BLACK);  tft.setTextSize(tamtextohoras);
    tft.print(u_segundo);
    primeraveztime = false;
  }
  /* if (d_horaprevias != d_hora) {
     tft.setCursor(xt, 100, tamhoras);
     tft.setTextColor(TFT_WHITE);  tft.setTextSize(tamtextohoras);
     tft.print(d_horaprevias);
     tft.setCursor(xt, 100, tamhoras);
     tft.setTextColor(TFT_BLACK);  tft.setTextSize(tamtextohoras);
     tft.print(d_hora);
     d_horaprevias = d_hora;
    }*/
  if (u_horaprevias != u_hora) {
    tft.setCursor(xt , 40, tamhoras);
    tft.setTextColor(TFT_WHITE);  tft.setTextSize(tamtextohoras);
    tft.print(u_horaprevias);
    tft.setCursor(xt , 40, tamhoras);
    tft.setTextColor(TFT_BLACK);  tft.setTextSize(tamtextohoras);
    tft.print(u_hora);  //tft.print(":");
    u_horaprevias = u_hora;
  }
  if (d_minutoprevias != d_minuto) {
    tft.setCursor(xt + 60, 40, tamhoras);
    tft.setTextColor(TFT_WHITE);  tft.setTextSize(tamtextohoras);
```

```
   tft.print(d_minutoprevias);
   tft.setCursor(xt + 60, 40, tamhoras);
   tft.setTextColor(TFT_BLACK);  tft.setTextSize(tamtextohoras);
   tft.print(d_minuto);
   d_minutoprevias = d_minuto;
  }
  if (u_minutoprevias != u_minuto) {
   tft.setCursor(xt + 120, 40, tamhoras);
   tft.setTextColor(TFT_WHITE);  tft.setTextSize(tamtextohoras);
   tft.print(u_minutoprevias);
   tft.setCursor(xt + 120, 40, tamhoras);
   tft.setTextColor(TFT_BLACK);  tft.setTextSize(tamtextohoras);
   tft.print(u_minuto); //tft.print(":");
   u_minutoprevias = u_minuto;
  }
  if (d_segundoprevias != d_segundo) {
   tft.setCursor(xt + 180, 40, tamhoras);
   tft.setTextColor(TFT_WHITE);  tft.setTextSize(tamtextohoras);
   tft.print(d_segundoprevias);
   tft.setCursor(xt + 180, 40, tamhoras);
   tft.setTextColor(TFT_BLACK);  tft.setTextSize(tamtextohoras);
   tft.print(d_segundo);
   d_segundoprevias = d_segundo;
  }
  if (u_segundoprevias != u_segundo) {
   tft.setCursor(xt + 240, 40, tamhoras);
   tft.setTextColor(TFT_WHITE);  tft.setTextSize(tamtextohoras);
   tft.print(u_segundoprevias);
   tft.setCursor(xt + 240, 40, tamhoras);
   tft.setTextColor(TFT_BLACK);  tft.setTextSize(tamtextohoras);
   tft.print(u_segundo);
   u_segundoprevias = u_segundo;
  }

}
void marcha()
{
  relacion_marcha = 4300 / 50;
  if (relacion_marcha <= 138 && relacion_marcha >= 130)
  {
    marcha_en_la_que_estoy = 1;
  }
  if (relacion_marcha <= 86 && relacion_marcha >= 85)
  {
    marcha_en_la_que_estoy = 2;
  }
  if (relacion_marcha <= 67 && relacion_marcha >= 64)
  {
    marcha_en_la_que_estoy = 3;
  }
  if (relacion_marcha <= 55 && relacion_marcha >= 52)
  {
```

```
   marcha_en_la_que_estoy = 4;
  }
  if (relacion_marcha <= 46 && relacion_marcha >= 47)
  {
   marcha_en_la_que_estoy = 5;
  }
  if (numero == 0) {
   tft.setCursor(360, 80, 4);
   tft.setTextColor(TFT_WHITE);  tft.setTextSize(3);
   tft.print(marcha_actual);
   tft.setCursor(360, 80, 4);
   tft.setTextColor(TFT_BLACK);  tft.setTextSize(3);
   tft.print("N");
  }


  else {
   tft.setCursor(360, 80, 4);
   tft.setTextColor(TFT_WHITE);  tft.setTextSize(3);
   tft.print("N");


   if (marcha_en_la_que_estoy != marcha_actual) {
    tft.setCursor(360, 80, 4);
    tft.setTextColor(TFT_WHITE);  tft.setTextSize(3);
    tft.print(marcha_actual);
    tft.setCursor(360, 80, 4);
    tft.setTextColor(TFT_BLACK);  tft.setTextSize(3);
    tft.print(marcha_en_la_que_estoy);
    marcha_actual = marcha_en_la_que_estoy;

   }
   tft.setCursor(360, 80, 4);
   tft.setTextColor(TFT_BLACK);  tft.setTextSize(3);
   tft.print(marcha_actual);
  }

}
void drawAlert(int x, int y , int side)
{
  if (alert) {
   tft.fillTriangle(x, y, x + 30, y + 47, x - 30, y + 47, rojo);
   tft.setTextColor(TFT_BLACK);
   tft.drawCentreString("!", x, y + 6, 4);

  }
  else {
   tft.fillTriangle(x, y, x + 30, y + 47, x - 30, y + 47, TFT_WHITE);

  }
}
void trianguloarriba(int x, int y , int side)
```

```
{
  if (subir) {
    tft.fillTriangle(x, y, x + 15, y + 24, x - 15, y + 24, rojo);
    tft.setTextColor(TFT_BLACK);

  }
  else {
    tft.fillTriangle(x, y, x + 15, y + 24, x - 15, y + 24, TFT_WHITE);

  }
}
void trianguloabajo(int x, int y , int side)
{
  if (bajar) {
    tft.fillTriangle(x, y, x - 15, y - 24, x + 15, y - 24, rojo);
    tft.setTextColor(TFT_BLACK);

  }
  else {
    tft.fillTriangle(x, y, x - 15, y - 24, x + 15, y - 24, TFT_WHITE);

  }
}
void  indica_temperatura()
{
  //temperatura_total = analogRead(A1);
  temperatura_total = 1000;
  temperatura_total = map(temperatura_total, 0, 1023, 0, temperatura_total_maxima);
  intervalo = map(temperatura_total, 0, temperatura_total_maxima, 0, -90); //usamos el intervalo
negativo porque si no el movimiento seria de abajo arriba, usamos unos 90 pixeles de altura,



  if (temperatura_total > 70.0) {
    colores = verde1;
  }
  if ((30.0 < temperatura_total) && (temperatura_total < 70.0)) {
    colores = naranja1;
  }
  if (temperatura_total < 30.0) {
    colores = rojo;
  }

  tft.fillRect(tempx, tempy, 40/*ancho*/, intervalo + 92/*alto*/, TFT_WHITE); //usamos un cuadrado
pero esta vez con altura positiva para que vaya borrando el negro, el 92 es para borra dos pixeles que
nos quedan  por ahi perdidos
  tft.fillRect(tempx, tempy + 91, 40/*ancho*/, intervalo/*alto*/, colores); //usamos el intervalo ne

  intervalo = map(numero, 0, numeromax, 0, 21);


}
```

```
void  indica_revolucion(int numero)
{

  intervaloR = map(numero, 0, 6000, 0, 21);




  if (intervaloR + 1 <= intervalodedondevengoR)
  {
    for (int p = intervalodedondevengoR; p > intervaloR + 1; p--)
    {
      if (p == 1)
      {
        tft.fillRect(misposicionesXR[0], misposicionesYR[0], precisionR, alturaR, TFT_WHITE);
        intervalodedondevengoR = p;

      }

      if (p == 2)
      {
        tft.fillRect(misposicionesXR[1], misposicionesYR[1], precisionR, alturaR, TFT_WHITE);
        intervalodedondevengoR = p;
      }
      if (p == 3)
      {
        tft.fillRect(misposicionesXR[2], misposicionesYR[2], precisionR, alturaR, TFT_WHITE);
        intervalodedondevengoR = p;
      }

      if (p == 4)
      {
        tft.fillRect(misposicionesXR[3], misposicionesYR[3], precisionR, alturaR, TFT_WHITE);
        intervalodedondevengoR = p;
      }
      if (p == 5)
      {
        tft.fillRect(misposicionesXR[4], misposicionesYR[4], precisionR, alturaR, TFT_WHITE);
        intervalodedondevengoR = p;

      }

      if (p == 6)
      {
        tft.fillRect(misposicionesXR[5], misposicionesYR[5], precisionR, alturaR, TFT_WHITE);
        intervalodedondevengoR = p;
      }
      if (p == 7)
      {
        tft.fillRect(misposicionesXR[6], misposicionesYR[6], precisionR, alturaR, TFT_WHITE);
        intervalodedondevengoR = p;
```

```
}

if (p == 8)
{
 tft.fillRect(misposicionesXR[7], misposicionesYR[7], precisionR, alturaR, TFT_WHITE);
 intervalodedondevengoR = p;
}
if (p == 9)
{
 tft.fillRect(misposicionesXR[8], misposicionesYR[8], precisionR, alturaR, TFT_WHITE);
 intervalodedondevengoR = p;
}

if (p == 10)
{
 tft.fillRect(misposicionesXR[9], misposicionesYR[9], precisionR, alturaR, TFT_WHITE);
 intervalodedondevengoR = p;
}
if (p == 11)
{
 tft.fillRect(misposicionesXR[10], misposicionesYR[10], precisionR, alturaR, TFT_WHITE);
 intervalodedondevengoR = p;
}

if (p == 12)
{
 tft.fillRect(misposicionesXR[11], misposicionesYR[11], precisionR, alturaR, TFT_WHITE);
 intervalodedondevengoR = p;
}
if (p == 13)
{
 tft.fillRect(misposicionesXR[12], misposicionesYR[12], precisionR, alturaR, TFT_WHITE);
 intervalodedondevengoR = p;
}

if (p == 14)
{
 tft.fillRect(misposicionesXR[13], misposicionesYR[13], precisionR, alturaR, TFT_WHITE);
 intervalodedondevengoR = p;
}
if (p == 15)
{
 tft.fillRect(misposicionesXR[14], misposicionesYR[14], precisionR, alturaR, TFT_WHITE);
 intervalodedondevengoR = p;
}

if (p == 16)
{
 tft.fillRect(misposicionesXR[15], misposicionesYR[15], precisionR, alturaR, TFT_WHITE);
 intervalodedondevengoR = p;
}
if (p == 17)
```

```
   {
    tft.fillRect(misposicionesXR[16], misposicionesYR[16], precisionR, alturaR, TFT_WHITE);
    intervalodedondevengoR = p;
   }

   if (p == 18)
   {
    tft.fillRect(misposicionesXR[17], misposicionesYR[17], precisionR, alturaR, TFT_WHITE);
    intervalodedondevengoR = p;
   }
   if (p == 19)
   {
    tft.fillRect(misposicionesXR[18], misposicionesYR[18], precisionR, alturaR, TFT_WHITE);
    alert = false;
    drawAlert(XTRI, YTRI , LTRI);
    intervalodedondevengoR = p;
   }

   if (p == 20)
   {
    tft.fillRect(misposicionesXR[19], misposicionesYR[19], precisionR, alturaR, TFT_WHITE);

    intervalodedondevengoR = p;
   }//fin del if

  }//fin del for

}//fin del if menor

if (intervaloR + 1 >= intervalodedondevengoR)
{
  for (int p = intervalodedondevengoR; p < intervaloR + 1; p++)
  {
   if (p == 1)
   {
    tft.fillRect(misposicionesXR[0], misposicionesYR[0], precisionR, alturaR, naranja7);
    //posx=posx+(anchura*(precision/4));
    intervalodedondevengoR = p;

   }

   if (p == 2)
   {
    tft.fillRect(misposicionesXR[1], misposicionesYR[1], precisionR, alturaR, naranja7);
    //posx=posx+(anchura*(precision/4));
    intervalodedondevengoR = p;
   }
   if (p == 3)
   {
    tft.fillRect(misposicionesXR[2], misposicionesYR[2], precisionR, alturaR, naranja7);
    // posx=posx+(anchura*(precision/4));
    intervalodedondevengoR = p;
```

```
}

if (p == 4)
{
 tft.fillRect(misposicionesXR[3], misposicionesYR[3], precisionR, alturaR, naranja7);
 // posx=posx+(anchura*(precision/4));
 intervalodedondevengoR = p;
}
if (p == 5)
{
 tft.fillRect(misposicionesXR[4], misposicionesYR[4], precisionR, alturaR, naranja7);
 // posx=posx+(anchura*(precision/4));
 intervalodedondevengoR = p;
}

if (p == 6)
{
 tft.fillRect(misposicionesXR[5], misposicionesYR[5], precisionR, alturaR, naranja7);
 //posx=posx+(anchura*(precision/4));
 intervalodedondevengoR = p;
}
if (p == 7)
{
 tft.fillRect(misposicionesXR[6], misposicionesYR[6], precisionR, alturaR, naranja7);
 // posx=posx+(anchura*(precision/4));
 intervalodedondevengoR = p;
}

if (p == 8)
{
 tft.fillRect(misposicionesXR[7], misposicionesYR[7], precisionR, alturaR, naranja7);
 //posx=posx+(anchura*(precision/4));
 intervalodedondevengoR = p;
}
if (p == 9)
{
 tft.fillRect(misposicionesXR[8], misposicionesYR[8], precisionR, alturaR, naranja7);
 // posx=posx+(anchura*(precision/4));
 intervalodedondevengoR = p;
}

if (p == 10)
{
 tft.fillRect(misposicionesXR[9], misposicionesYR[9], precisionR, alturaR, naranja7);
 //  posx=posx+(anchura*(precision/4));
 intervalodedondevengoR = p;
}
if (p == 11)
{
 tft.fillRect(misposicionesXR[10], misposicionesYR[10], precisionR, alturaR, naranja7);
 //  posx=posx+(anchura*(precision/4));
 intervalodedondevengoR = p;
```

```
}

if (p == 12)
{
 tft.fillRect(misposicionesXR[11], misposicionesYR[11], precisionR, alturaR, naranja7);
 //   posx=posx+(anchura*(precision/4));
 intervalodedondevengoR = p;
}
if (p == 13)
{
 tft.fillRect(misposicionesXR[12], misposicionesYR[12], precisionR, alturaR, naranja7);
 //   posx=posx+(anchura*(precision/4));
 intervalodedondevengoR = p;
}

if (p == 14)
{
 tft.fillRect(misposicionesXR[13], misposicionesYR[13], precisionR, alturaR, naranja7);
 //   posx=posx+(anchura*(precision/4));
 intervalodedondevengoR = p;
}
if (p == 15)
{
 tft.fillRect(misposicionesXR[14], misposicionesYR[14], precisionR, alturaR, naranja7);
 //   posx=posx+(anchura*(precision/4));
 intervalodedondevengoR = p;
}

if (p == 16)
{
 tft.fillRect(misposicionesXR[15], misposicionesYR[15], precisionR, alturaR, naranja7);
 //    posx=posx+(anchura*(precision/4));
 intervalodedondevengoR = p;
}
if (p == 17)
{
 tft.fillRect(misposicionesXR[16], misposicionesYR[16], precisionR, alturaR, naranja7);
 //    posx=posx+(anchura*(precision/4));
 intervalodedondevengoR = p;
}

if (p == 18)
{
 tft.fillRect(misposicionesXR[17], misposicionesYR[17], precisionR, alturaR, naranja7);
 //     posx=posx+(anchura*(precision/4));
 intervalodedondevengoR = p;
}
if (p == 19)
{
 tft.fillRect(misposicionesXR[18], misposicionesYR[18], precisionR, alturaR, naranja7);
 //      posx=posx+(anchura*(precision/4));
 alert = true;
```

```
      drawAlert(XTRI, YTRI , LTRI);
      intervalodedondevengoR = p;
     }

     if (p == 20)
     {
      tft.fillRect(misposicionesXR[19], misposicionesYR[19], precisionR, alturaR, naranja7);
      //    posx=posx+(anchura*(precision/4));
      intervalodedondevengoR = p;
     }//fin del if

   }//fin del for

  }//fin del if mayor
}
void posicionamiento_revolucion()
{

  for (rellenodeposicionesXR = 0; rellenodeposicionesXR < 20; rellenodeposicionesXR++)
  {
   if (horizontalR)
   {
    misposicionesXR[rellenodeposicionesXR] = posx1R;
    posx1R = posx1R + (anchuraR * (precisionR / 4));
   }
   else {
    misposicionesXR[rellenodeposicionesXR] = posicionhorizontalfijaR;
   }
  }
  for (rellenodeposicionesYR = 0; rellenodeposicionesYR < 20; rellenodeposicionesYR++)
  {
   if (verticalR)
   {
    misposicionesYR[rellenodeposicionesYR] = posy1R;
    posy1R = posy1R - (anchuraR * (precisionR / 4));
   }
   else {
    misposicionesYR[rellenodeposicionesYR] = posicionverticalfijaR;

   }

  }
}

void  indica_tension()
{
  //bateria_total = analogRead(A0);
  bateria_total = 100.0;
  bateria_total = map(bateria_total, 0, 1023, 0, bateria_total_maxima);

  intervaloV = map(bateria_total, 0, bateria_total_maxima, 0, -90); //usamos el intervalo negativo
porque si no el movimiento seria de abajo arriba, usamos unos 90 pixeles de altura,
```

```
  if (bateria_total > 10.0) {
    colores = verde1;
  }
  if ((5.0 < bateria_total) && (bateria_total < 10.0)) {
    colores = naranja1;
  }
  if (bateria_total < 5.0) {
    colores = rojo;
  }

  tft.fillRect(batx, baty, 40/*ancho*/, intervaloV + 92/*alto*/, TFT_WHITE); //usamos un cuadrado pero esta vez con altura positiva para que vaya borrando el negro, el 92 es para borra dos pixeles que nos quedan  por ahi perdidos
  tft.fillRect(batx, baty + 91, 40/*ancho*/, intervaloV/*alto*/, colores); //usamos el intervalo ne

}

void  indica_depositoR(int numero)////recibes el numero a representar en la barra
{

  intervaloD = map(numero, 0, deposito_total_maximo, 0, -90); //usamos el intervalo negativo porque si no el movimiento seria de abajo arriba, usamos unos 90 pixeles de altura,
  if (deposito_total > 20.0) {
    colores = verde1;
  }
  if ((10.0 < deposito_total) && (deposito_total < 20.0)) {
    colores = naranja1;
  }
  if (deposito_total < 10.0) {
    colores = rojo;
  }

  tft.fillRect(depx, depy, 40/*ancho*/, intervaloD + 92/*alto*/, TFT_WHITE); //usamos un cuadrado pero esta vez con altura positiva para que vaya borrando el negro, el 92 es para borra dos pixeles que nos quedan  por ahi perdidos
  tft.fillRect(depx, depy + 91, 40/*ancho*/, intervaloD/*alto*/, colores); //usamos el intervalo ne


}
```

5.0 Futuras mejoras

Ningún proyecto es perfecto, y por este motivo, se han pensado unas mejoras que pueden ayudar al proyecto a ser más ambicioso, es por eso por lo que se han dejado una serie de aspectos preparados para que, en futuras versiones, el proyecto pueda aumentar sus características o servicios.

Los aspectos que compondrían una mejoría serían:

1. Un estudio que refleje los diferentes tipos de accidentes, condiciones en las que se producen y valores exactos que testifiquen qué sí y qué no es un accidente.
2. Realizar una tabla en la cual se muestren los kilómetros parciales de varias etapas, los consumos instantáneos y los consumos medios, así como el usuario pueda resetear el parcial que quiera.

Para refleja los diferentes tipos de accidentes, como ya se dijo anteriormente, el inclinómetro nos servirá para monitorizar las aceleraciones que sufre el vehículo. Además se ha establecido una comunicación por I2C, y por el puerto serie entre las tarjetas Arduino Mega y Micro. Así damos diferentes opciones a la hora de comunicar Arduinos que puede llegar a ser interesante. El Arduino Mega que recibe la velocidad, podría hacérsela pasar al Arduino Micro a través de estos puertos.

El Arduino Micro, a parte de ser el encargado de poder de decidir si hubo accidente o no, debe poder comunicarlo, pero debido a que él no tiene la información de GPS ni la capacidad de enviar mensajes, debe entonces mandar una señal al Arduino Mega. La única manera de que el Arduino Mega pueda enterarse de lo sucedido, dicha señal debe comportarse como una interrupción, de tal modo que nos aseguramos que haga caso de ella. Una vez reciba la interrupción del accidente, se preparará para lanzar el mensaje[15].

```
void mensaje_sms()
   {
      Serial.println("Enviando SMS...");
      SIM900.print("AT+CMGF=1\r");  //Configura el modo texto para enviar o recibir mensajes
      delay(1000);
      SIM900.println("AT+CMGS=\"XXXXXXXXX\"");  //Numero al que vamos a enviar el mensaje
      delay(1000);
      SIM900.println("SMS enviado desde un Arduino. Saludos de Prometec.");  // Texto del SMS
      delay(100);
      SIM900.println((char)26); //Comando de finalización ^Z
      delay(100);
      SIM900.println();
      delay(5000);  // Esperamos un tiempo para que envíe el SMS
      Serial.println("SMS enviado");
   }
```

La función *"mensaje_sms"*, es a la que acudirá para enviar el mensaje una vez se produzca la interrupción. Tal y como nos muestra el programa, tenemos una parte de mensaje en la cual podemos añadir aquello que nos sea relevante. Por su puesto, en este mensaje no podría faltar la posición GPS. Sin ella este mensaje carecería de cuerpo, pues es un paso importante notificar que ha habido un accidente, pero saber su posición es muy importante para evitar dar palos de ciego en el rescate.

Ilustración 53 : Parte de programación de Enviar Mensajes

Es posible que quepa la posibilidad que el sistema e algún momento, pueda fallar. Y mande un SMS de alerta de accidente cuando en realidad no haya ocurrido nada. Por lo que recomendamos que ese mensaje se envíe, a parte de una o varias personas en concreto, que envíe una copia al usuario. Así, en caso de un falso accidente, el usuario recibirá un mensaje de alerta, por lo que podrá pulsar un botón que envíe otro mensaje diciendo que ha sido todo una falsa alarma o que está todo correcto, facilitando la tranquilidad de los usuarios que recibieron el mensaje de alerta.

6.0 Conclusiones

El trabajo de fin de grado, ha supuesto para mí, un verdadero reto. He aprendido y afianzado conocimientos adquiridos a lo largo de la carrera. En cierto modo, me he enfrentado a problemas que anteriormente no había visto. Los puntos que más problemática tuve a la hora de hacerlo, fue la programación. Personalmente no me considero muy habilidoso en ese tema, pero distribuyendo los problemas poco a poco, he podido abordarlo con mejor resultado que enfrentándose de golpe. Comenzando por las estructuras generales y acabando por cosas más concretas.

En cuando a la parte de PCB, cuando los circuitos se hacen más grandes, no vale con colocar las piezas de una manera aleatoria, hay que ser conscientes y cuidadosos, o de lo contrario, la opción de *"AutoRoute"* no servirá de mucho si resulta que los componentes que pertenecen a un mismo circuito (el circuito que estabiliza la tensión por ejemplo) están muy alejados unos de otros obligando a crear pistas que cruzan la placa de lado a lado.

Por este motivo, hay que agrupar los componentes que posean relación para que el *routeo* se realice de manera que ayudas al programa a crear mejores opciones.

El planteamiento de los circuitos me ha ayudado a recordar lo visto en asignaturas y ha permitido interiorizarlos y ver desde otra perspectiva sus utilidades.

Mis conocimientos sobre motocicletas o en general automóviles, era, antes de introducirme al trabajo de finde grado, muy básico. Cuando me introduje más de lleno en el trabajo, me fui familiarizando más con los conceptos y las nomenclaturas del sector.

Gracias a que la Universidad nos permitió utilizar el taller de los equipos de motos y coches, así como sus herramientas respectivamente, pudimos trabajar con la moto de pruebas para poder sacar las señales que nos interesaban. Fue entonces cuando probé el trabajo de campo del proyecto.

7.0 Bibliografía

[1] «De extra de lujo a la pantalla táctil: así ha cambiado la radio de los automóviles en 60 años», *abc*, 13-feb-2018. [En línea]. Disponible en: https://www.abc.es/motor/reportajes/abci-extra-lujo-pantalla-tactil-cambiado-radio-automoviles-60-anos-201802130159_noticia.html. [Accedido: 14-ago-2019].

[2] M. Baeza, «Por qué el salpicadero del coche ya no va a tener botones», *El Motor*, 13-feb-2017. [En línea]. Disponible en: https://motor.elpais.com/actualidad/salpicadero-y-puesto-de-mando-del-futuro/. [Accedido: 14-ago-2019].

[3] elEconomista.es, «Los coches con las pantallas táctiles más grandes, ¡hasta 49 pulgadas! - Ecomotor.es». [En línea]. Disponible en: https://www.eleconomista.es/ecomotor/motor/noticias/9828720/04/19/Los-coches-con-las-pantallas-tactiles-mas-grandes-.html. [Accedido: 11-ago-2019].

[4] «¿Es peligroso tener pantallas táctiles en los coches?», *El Español*, 30-jun-2019. [En línea]. Disponible en: https://www.elespanol.com/omicrono/20190630/peligroso-tener-pantallas-tactiles-coches/410209646_0.html. [Accedido: 14-ago-2019].

[5] «Ride Command Touchscreen Display | Indian Motorcycle». [En línea]. Disponible en: https://www.indianmotorcycle.com/en-us/touch-screen/. [Accedido: 18-ago-2019].

[6] «Odometro LCD - TOOGOO(R)Oodometro velocimetro velocimetro calibrador universal de retroiluminacion LCD digital de motocicleta: Amazon.es: Coche y moto». [En línea]. Disponible en: https://www.amazon.es/Odometro-LCD-velocimetro-retroiluminacion-motocicleta/dp/B01EYDQILI?SubscriptionId=AKIAI3FHFDM5WSV7KXSA&tag=cosasdemoto.com-21&linkCode=xm2&camp=2025&creative=165953&creativeASIN=B01EYDQILI. [Accedido: 18-ago-2019].

[7] «La historia del airbag, la bolsa que salva vidas -- Autobild.es». [En línea]. Disponible en: https://www.autobild.es/reportajes/historia-airbag-bolsa-que-salva-vidas-296491#targetText=La%20historia%20del%20airbag%2C%20como,el%20Oldsmobile%20Toronado%20de%201973. [Accedido: 05-sep-2019].

[8] «21 cosas que no sabías sobre los airbags de los coches», *Autopista.es*. [En línea]. Disponible en: https://www.autopista.es/tecnologia/articulo/airbags-coches-caracteristicas-como-funcionan. [Accedido: 05-sep-2019].

[9] RACE, «La llamada de emergencia eCall, obligatoria en coches nuevos», *RACE*, 13-abr-2018. .

[10] «Determinar la orientación con Arduino y el IMU MPU-6050», *Luis Llamas*. .

[11] «16 bit color generator (RGB565 color picker) | Electrical engineering and programming notepad», *16 bit color generator (RGB565 color picker) | Electrical engineering and programming notepad*. .

[12] «Localización GPS con Arduino y los módulos GPS NEO-6», *Luis Llamas*. .

[13] «arduino uno - NEO 6M GPS speed», *Arduino Stack Exchange*. [En línea]. Disponible en: https://arduino.stackexchange.com/questions/22873/neo-6m-gps-speed. [Accedido: 04-ago-2019].

[14] «Medir temperatura y humedad con Arduino y DHT11 o DHT22». [En línea]. Disponible en: https://www.luisllamas.es/arduino-dht11-dht22/. [Accedido: 12-sep-2019].

[15] designthemes, «MÓDULO GSM/GPRS: llamar y enviar SMS | Tienda y Tutoriales Arduino». .