



CONTROL DE DISTANCIA DE SEGURIDAD EN ADELANTAMIENTOS A BICICLETAS

Enrique Tortajada González

Grado de Tecnologías de Telecomunicación

05.663 TFG Arduino

Consultor: Antoni Morell Pérez

Profesor responsable de Área: Pere Tuset Peiró

9 de junio de 2018



Aquesta obra està subjecta a una llicència de
[Reconeixement-NoComercial-SenseObraDerivada 3.0](https://creativecommons.org/licenses/by-nc-nd/3.0/)
[Esanya de Creative Commons](https://creativecommons.org/licenses/by-nc-nd/3.0/)

“A Amparo, Sofía y Sergio por su apoyo incondicional en todas mis locuras”

FITXA DEL TREBALL FINAL

Títol del treball:	<i>Control de distancia de seguridad en adelantamientos a bicicletas</i>
Nom de l'autor:	<i>Enrique Tortajada González</i>
Nom del consultor/a:	<i>Antoni Morell Pérez</i>
Nom del PRA:	<i>Peret Tuset Peiró</i>
Data de lliurament (mm/aaaa):	<i>09/06/2018</i>
Titulació o programa:	<i>Grado de Tecnologías de Telecomunicación</i>
Àrea del Treball Final:	<i>05.663 TFG Arduino</i>
Idioma del treball:	<i>Castellà</i>
Paraules clau	<i>Arduino, seguridad en bicicleta, distancia de seguridad</i>

RESUMEN

El sistema propuesto consiste en la medición de la distancia lateral cuando un vehículo rebasa a un ciclo o ciclomotor en un adelantamiento. Si este se realiza respetando una distancia de seguridad establecida (más de un metro y medio) el dispositivo solo indica la medida en un display LCD. En caso contrario, el dispositivo podrá activar una o varias salidas para un gestionar un dispositivo externo como, por ejemplo, una cámara de fotografía y/o video para registrar al infractor.

Además, el dispositivo registrará la posición a través del localizador GPS integrado, de forma que guardan la posición, fecha, hora y distancia indebida de adelantamiento del suceso para poder realizar a posteriori estudios referentes a seguridad de las vías y forma de evitar que se produzcan situaciones estas situaciones de peligro.

El medidor podría ser utilizado por el cuerpo legal competente para denunciar este tipo de infracción o por el usuario del vehículo de dos ruedas. Sin embargo, también podría ser utilizado por los vehículos que realizan el adelantamiento con la finalidad de controlar que este se hace a la distancia adecuada.

ABSTRACT

The proposed system consists of the measurement of the lateral distance when an automotive vehicle passes bicycle or moped in an overtaking. If this is done an established safety distance (more than one and a half meters) the device only indicates the measurement on an LCD screen. However, if an infraction occurs, the device could activate one or more outputs to manage an external device, like a photography/video camera to register the traffic infraction.

In addition, the device will register the user's position through the integrated GPS tracker, which then archives the position, date, time and violation of lateral distance during overtaking in order to be able to do studies later related to road safety and prevention of these dangerous situations.

The meter could be used by the law enforcement or vehicle operators to report a lateral distance infraction. This device could also be used by vehicles that monitor overtaking processes to ensure this is done at the proper distance.

CONTENIDO

TABLA DE ILUSTRACIONES.....	III
CAPITULO 1: INTRODUCCIÓN.....	1
1.1 INTRODUCCIÓN.....	1
1.2 MOTIVACIÓN Y OBJETIVOS.....	2
1.2.1 REQUERIMIENTOS TECNICOS DEL PRODUCTO.....	2
1.3 VIABILIDAD DEL PROYECTO.....	3
1.3.1 VIABILIDAD ECONÓMICO FINANCIERA.....	3
1.3.2 VIABILIDAD OPERACIONAL.....	4
1.3.3 VIABILIDAD DE MERCADO.....	4
1.4 ORGANIZACIÓN DE LA MEMORIA.....	5
CAPITULO 2: ESTADO DEL ARTE.....	6
2.1 INTRODUCCIÓN A LAS TECNOLOGÍAS USADAS.....	6
2.2 SISTEMAS DE MEDICIÓN DE DISTANCIAS POR MEDIO DE ONDAS.....	6
2.3 SISTEMAS DE POSICIONAMIENTO VÍA SATELITE.....	8
2.4 ARDUINO.....	9
2.4.1 ARQUITECTURA AVR.....	10
2.5 MEMORIA FLASH.....	12
2.6 APLICACIONES SIMILARES AL PROYECTO.....	13
CAPITULO 3: PROPUESTA DE PROYECTO.....	15
3.1 INTRODUCCIÓN.....	15
3.2 DESCRIPCION DE LOS BLOQUES HARDWARE.....	16
3.2.1 PLACA DE CONTROL.....	16
3.2.2 MÓDULO GPS.....	18
3.2.3 MÓDULO DE ULTRASONIDOS.....	20
3.2.4 DISPLAY 16 x 2 VÍA BUS I2C.....	22

3.2.5 MÓDULO ALMACENAMIENTO SD.....	24
3.2.7 CONSIDERACIONES ADICIONALES SOBRE EL HARDWARE	25
3.4 SOFTWARE NECESARIO	26
CAPITULO 4: DISEÑO DEL DISPOSITIVO: MONTAJE Y PRUEBAS.....	27
4.1 ENTORNO DE DESARROLLO Y SOFTWARE DE SOPORTE	27
4.2 PRUEBA DE LOS DIFERENTES MÓDULOS.....	29
4.2.1 MEDIDOR DE DISTANCIA POR ULTRASONIDOS.....	29
4.2.2 LOCALIZADOR GPS Y ESCRITURA DE DATOS EN LA MICRO SD.....	36
4.3 FUNCIONAMIENTO DEL DISPOSITIVO	42
4.4 MONTAJE FINAL Y PRUEBAS DISPOSITIVO	46
4.4.1 MONTAJE EN PLACA DE PRUEBAS	47
4.4.2 MONTAJE DEFINITIVO.....	49
4.4.3 MONTAJE EN LA BICICLETA Y ALIMENTACIÓN.....	52
4.5 PRUEBAS DEL DISPOSITIVO	54
4.5.1 MEDIDAS EN ESTÁTICO.....	54
4.5.2 MEDIDAS EN DINÁMICO.....	57
CAPITULO 5: CONCLUSIONES Y LINEAS FUTURAS DE DESARROLLO	59
CAPITULO 6: PLANIFICACIÓN TEMPORAL.....	60
BIBLIOGRAFÍA.....	63
ANEXOS.....	68
1. CÓDIGO PRUEBA MÓDULOS LCD I2C Y MEDIDOR ULTRASONIDOS (APARTADO 4.2.1)	68
2. CÓDIGO PRUEBA MÓDULOS GPS Y MICRO SD (APARTADO 4.2.2).....	71
3. CÓDIGO DISPOSITIVO DE CONTROL DISTANCIA DE ADELANTAMIENTO (APARTADO 4.3).....	75
COSTE DE MATERIAL DEL PROTOTIPO	81

TABLA DE ILUSTRACIONES

Figura 1: Arquitectura AVR de Arduino	12
Figura 2: Detalle del dispositivo C3FT v3 (imagen procedente de codexus.com)	14
Figura 3: Detalle de montaje C3FT v3 (imagen procedente de codexus.com)	14
Figura 4: Diagrama de bloques del dispositivo	15
Figura 5: Arduino nano (https://store.arduino.cc/Arduino-nano)	16
Figura 6: Placa compatible Elegoo Mega 2560	17
Figura 7: Módulo GPS neo 6	18
Figura 8: Sensor ultrasonidos HC SR-04	20
Figura 9: Funcionamiento temporal del HC SR-04	21
Figura 10: Bus I2C	23
Figura 11: Display LCD 16 x 2 y controlador I2C para Arduino	23
Figura 12: Módulo micro SD	24
Figura 13: IDE Arduino	28
Figura 14: Montaje en la placa de pruebas medidor distancia con LCD	30
Figura 15: Esquema eléctrico del medidor distancia con LCD	30
Figura 16: Librería Arduino I2C LCD en GitHub	31
Figura 17: Pruebas medición distancia	33
Figura 18: Pruebas de GPS y tarjeta SD	39
Figura 19: Monitor serial Arduino con datos GPS	39
Figura 20: Archivo TXT generado	40
Figura 21: Diagrama de flujo GPS-sd	41
Figura 22: Dispositivo sin señal de GPS	42
Figura 23: Dispositivo con señal de GPS guardando datos en SD	43
Figura 24: Datos registrados en SD con acuse de error	43
Figura 25: Funcionamiento del dispositivo	44

Figura 26: Diagrama de funcionamiento del medidor.....	44
Figura 27: Prueba montaje final en placa protoboard.....	48
Figura 28: Placa prototipo PCB para Arduino Mega.....	49
Figura 29: Proceso de montaje y pruebas del montaje final.....	49
Figura 30 Conector micro o aviación 5 pines	50
Figura 31: Conexiones USB tipo A.....	51
Figura 32: Resistencia de pull-up y filtrado de rebotes de señal del pulsador.....	52
Figura 33: Batería tipo Power Bank de alimentación del medidor	53
Figura 34: Anclaje del medidor a la bicicleta	53
Figura 35: Detalle del montaje en la bicicleta del dispositivo final.....	54
Figura 36: deriva de sensibilidad.....	56
Figura 37:Detalle del montaje para medición en estático.....	56
Figura 38: Dispersión del dispositivo en estático.....	57
Figura 39: Dispersión del dispositivo en dinámico.....	58

CAPITULO 1: INTRODUCCIÓN

1.1 INTRODUCCIÓN

El número de personas que utilizan la bicicleta por las vías que se comparten con los vehículos a motor no deja de crecer. Tanto si su utilización es lúdica o como medio de transporte, la convivencia de la bicicleta con los vehículos de motor de cuatro ruedas genera mucha controversia y acaba siendo peligrosa para el usuario de el de dos, que en definitiva es la parte más débil. Mas de cuatrocientos muertos en más de cinco mil accidentes de tráfico en las que había bicicletas implicadas en la última década en las carreteras españolas, según la Dirección General de Tráfico, nos dan una medida de la importancia del problema a abordar.

Además, en un estudio realizado por A3Media-AXA para la conocida campaña “Ponle freno” se determinó que cada uno de cinco adelantamientos a bicicletas se hace sin respetar la distancia de seguridad. Para ello, dos ciclistas aficionados recorrieron más de 7000 km con dispositivos laser con contadores, que determinaron las conclusiones del estudio.

El conocimiento de las normas de circulación y el respeto resultan una piedra angular para evitar los accidentes en casos de adelantamientos a ciclistas. Por ejemplo, muchos conductores desconocen que pueden rebasar la línea continua al adelantar a una bicicleta si las condiciones de visibilidad lo permiten. Por parte de los ciclistas, el desconocimiento y el incumplimiento de las normas también es notable. Por lo tanto, resulta de vital importancia que esta maniobra se haga a una velocidad y separación que no pongan en peligro al ciclista.

1.2 MOTIVACIÓN Y OBJETIVOS

El objetivo principal sería desarrollar un dispositivo capaz de medir la distancia a la que un ciclo, ciclomotor, vehículo de tracción animal o peatón es rebasado, registrar el evento cuando este se realice de forma insegura y enviar la localización de este para que quede registrada en una base de datos. El dispositivo también podría ser utilizado por los agentes de tráfico o cualquier usuario para denunciar este tipo de infracciones registrando por medio de una cámara externa activada por el dispositivo.

Aunque la finalidad principal podría ser el estudio a gran escala de puntos negros en las carreteras donde se producen estos adelantamientos, dada la capacidad por parte del sistema de almacenar los puntos donde se producen estos a menos de un metro y medio. Se puede imaginar pues, que a gran escala el dispositivo formaría parte de La Internet de las Cosas y tras la gestión adecuada de los datos almacenados la administración pública pueda actuar en consecuencia, por ejemplo, adecuando las vías con arcenes más amplios o construyendo carriles bici o reduciendo la velocidad en esas vías.

El objetivo final del uso del dispositivo debe ser el de reducir la siniestralidad en carretera por este tipo de infracciones por encima del fin sancionador en sí mismo, en base al conocimiento de las zonas donde se producen este tipo de incidentes. Por tanto, los datos recogidos deberían estar al alcance de todos los usuarios de la vía, de forma que se evite parte de tráfico de cualquiera de los vehículos implicados o se tenga conciencia del peligro tomando las precauciones pertinentes.

1.2.1 REQUERIMIENTOS TECNICOS DEL PRODUCTO

El dispositivo tendrá que ser capaz de efectuar la medida de separación lateral a la que el vehículo (bicicleta o ciclomotor) es sobrepasado y en caso de ser menor que el límite legal activar uno o varios contactos externos (mediante un pulso) que podrán activar dispositivos externos: cámara fotográfica o de video. Además, deberá registrar en una memoria SD, la posición, la distancia medida, la fecha y hora a la que se efectuó el

adelantamiento indebido, con el fin de recabar información que posteriormente pueda ser tratada para evitar accidentes.

Debe permitir adicionalmente el poder añadir marcas sobre medidas que no sean reales, pues puede haber errores de medida o rebasamientos hechos por otras bicicletas a menos de esa distancia. Este proceso se hará en el mismo momento de la medición con un pulsador en el dispositivo que registrará un valor junto a la medición indicando que estos datos deben ser ignorados cuando se procesen en un futuro.

La medición debe realizarse con cierta precisión, por lo que se deberán hacer las pruebas necesarias para que se verifique el correcto funcionamiento o al menos una primera aproximación para que se preparen futuras mejoras que puedan mejorar la precisión del dispositivo.

1.3 VIABILIDAD DEL PROYECTO

1.3.1 VIABILIDAD ECONÓMICO FINANCIERA

La inversión inicial en el producto es mínima, ya que el coste de producto es muy inferior al de la competencia. El coste del producto de la competencia es de 1230,24 euros con portes incluidos, a lo que habría que añadir los impuestos, ya que sólo se puede adquirir desde Estados Unidos, de forma que el precio del producto se vería sensiblemente incrementado. La idea en este proyecto es crear un producto que mejore en prestaciones al existente y reduzca sensiblemente el precio para que sea accesible al mayor número de usuarios.

El precio de mercado podría ser de la cuarta parte (400 euros frente a 1600) y se tendría mucho margen de beneficio ya que el coste de los materiales del primer prototipo no llega a 130 euros. Aunque se prevé mejorar el prototipo y externalizar su fabricación lo que podría suponer costes adicionales.

Se partirá de un capital social de 3000 euros, mínimo necesario para crear una sociedad de responsabilidad limitada unipersonal proveniente de recursos propios. En principio se decidirá fabricar una cantidad limitada de dispositivos una vez mejoradas sus funciones; unas 20 unidades. No se pueden hacer estimaciones de ventas iniciales,

dada la novedad del producto. Cabría realizar un estudio de mercado con un dispositivo mejorado con el fin de poder hacer estimaciones iniciales de venta.

1.3.2 VIABILIDAD OPERACIONAL

Se dispone de los conocimientos necesarios para el desarrollo y mejora del producto. Dado que se prevé la externalización del producto en función del crecimiento de ventas del producto y en principio se trabajará casi a demanda. Se utilizará un local propio y se pedirán los permisos requeridos para el desarrollo de la actividad. Los recursos de personal pueden ser ampliados en función de la demanda, pero siempre se ajustarán a un mínimo de costes.

1.3.3 VIABILIDAD DE MERCADO

Dado que el producto resulta novedoso y todavía no existe la necesidad de su uso, más bien se basa en la concienciación de los usuarios de la vía para respetar las medidas de seguridad en adelantamientos puede no existir un mercado objetivo inicialmente. El desarrollo de una aplicación gratuita que de soporte a la información que puede aportar al dispositivo funcionalidad y mercado objetivo. Si, además, el dispositivo se integra con los dispositivos móviles inteligentes a través de una aplicación gratuita se puede reducir el coste y tamaño del dispositivo (el dispositivo solo realiza la medición de distancia).

1.4 ORGANIZACIÓN DE LA MEMORIA

En el capítulo uno se expone la motivación y objetivos de proyecto que pasan por mejorar la seguridad y bajar la siniestralidad de los ciclistas en carretera. En el segundo capítulo se da un marco teórico de la idea presentada, de manera que se explican parte de las tecnologías necesarias para el desarrollo final.

En el tercer capítulo se entra en la parte técnica del producto a desarrollar, dando ejemplos de componentes que, si bien pueden ser los que se utilizarán finalmente, pueden sufrir cambios en la siguiente etapa de pruebas. El capítulo está dividido en dos partes: una parte del hardware necesario para implementar el dispositivo y otra parte de software, la cual está poco desarrollada todavía, pero en la cual se dan ideas genéricas sobre el funcionamiento final.

El cuarto capítulo corresponde al diseño montaje, y pruebas del dispositivo. En el caso de ser necesario, se puede cambiar alguno de los módulos mencionados en apartados anteriores. El quinto capítulo recoge las conclusiones y posibles mejoras del dispositivo tanto en relación al propio dispositivo, como al entorno y uso de los datos recogidos por el dispositivo. Finalmente, los anexos recogen, el código utilizado para las pruebas de los diferentes módulos y el código del diseño definitivo, además de la inversión en material realizada para implementar el medidor

CAPITULO 2: ESTADO DEL ARTE

2.1 INTRODUCCIÓN A LAS TECNOLOGÍAS USADAS

El desarrollo de este producto hace uso de tecnologías ya extendidas en el uso cotidiano que por ejemplo puedan estar implementadas en cualquier teléfono móvil, como pueden ser las tecnologías GSM, GPRS y las de geolocalización GPS. Para el caso de las mediciones de distancia se pueden utilizar tecnologías laser o ultrasonidos. En el primer caso existen muchos medidores que utilizan la tecnología láser para medir distancias mayores que las que se pueden medir con ultrasonidos. Para el caso de la medición con ultrasonidos encontramos esta tecnología en objetos que ya son tan comunes como los robots aspiradores o en cualquier sensor de aparcamiento de los vehículos.

Por otro lado, para poder integrar estas tecnologías para el desarrollo de un prototipo de una forma fácil se hará uso de un sistema embebido creado a partir de una placa de desarrollo *Arduino* (o placa compatible) con los módulos de extensión (*shields*) necesarios para dotar de las capacidades necesarias de medición de distancias, geolocalización y guardado de datos para su posterior procesado.

Por último, se hará una revisión de los productos similares existentes en el mercado. Sólo se ha encontrado un producto que reúna parte de las características del dispositivo creado en este proyecto: el dispositivo C3FT (*See-Three-Feeds*) de *Codaxus LLC*, compañía situada en Austin (Texas) que comercializa el producto a través de su página web: <http://www.codaxus.com/>. El producto comenzó su comercialización en 2015 y en el año 2017 comenzó la venta tercera versión del producto (C3FT v3).

2.2 SISTEMAS DE MEDICIÓN DE DISTANCIAS POR MEDIO DE ONDAS

Una forma para medir distancia a partir de ondas, ya sean ondas mecánicas (sonoras o ultrasónicas) o electromagnéticas (luz visible, infrarroja u ondas de radio) implica el mismo principio básico para la medición de distancias; se debe conocer la velocidad a la que se transmiten esas ondas y el tiempo que tardan en recorrerla.

$$\text{Distancia} = \text{Velocidad de las ondas} \times \text{tiempo en recorrer la distancia}$$

Dependiendo de la distancia y del medio nos convendrá usar un tipo de ondas u otras y un rango de frecuencias u otro. Por ejemplo, en el SONAR (*Sound Navigation And Ranging*, navegación por sonido) utilizado por los submarinos donde se usan ondas infrasónicas, es decir por debajo de 20 Hz por transmitirse bien en el medio acuoso este tipo de ondas.

Para medir grandes distancias se usan ondas electromagnéticas como pueden ser la luz visible, la infrarroja o las ondas de radio debido a que su velocidad de propagación es mucho mayor que la del sonido: 299.792.458 m/s de la luz frente a 343,5 m/s de las ondas sónicas. Además, también depende del medio y se ha de tener en cuenta el caso de la refracción para el cambio de medio.

Pero en este caso el proyecto se centrará en el caso de los ultrasonidos (son adecuados para la distancia a medir), ondas mecánicas a partir de 20 KHz, frecuencias por encima del margen audible desplazándose por un medio homogéneo que es el aire. Existen sensores que son usados comúnmente en robótica donde emisor-receptor pueden estar en el mismo transductor (implica emitir y conmutar el uso para recibir) o de forma separada un transductor que emite y otro que recibe. Hay dos formas de medir la distancia:

- Tiempo de vuelo: Emitir un tren de pulsos, poner un temporizador en marcha y esperar al eco de los pulsos para determinar la distancia como se ha explicado al principio.
- Cambio de fase: Se envía una señal periódica continua y se mide el desfase de onda de la señal recibida para determinar la distancia. Entonces, la distancia se puede conseguir a partir la convolución entre la señal emitida y la señal de recepción, pero la medida estará limitada a un desfase de 360° entre las dos señales, por lo tanto, será útil para medición de distancias atendiendo a la longitud de onda usada.

Propiedades de los sensores ultrasónicos:

- Se suelen utilizar cuatro frecuencias: 49 KHz, 52,5 KHz, 56 KHz y 60KHz.
- Existe un ángulo de detección sensible determinado. Se indica en la hoja de características la sensibilidad y la atenuación en *dBs* según el ángulo de incidencia del lóbulo principal.
- Tienen una distancia máxima de detección que depende de la frecuencia, sensibilidad del dispositivo y del medio de transmisión que van de los 3 a 4 metros, aunque se pueden encontrar sensores de hasta 10 metros.
- Los sensores pueden verse afectados por un fenómeno llamado *cross-talking*: las ondas de un sensor pueden verse interferidas por otros sensores si se dispone de varios sensores o del mismo sensor si los tiempos de disparo están muy próximos, pudiendo recibir ecos de dos disparos consecutivos.

2.3 SISTEMAS DE POSICIONAMIENTO VÍA SATELITE

Los sistemas de posicionamiento engloban el conjunto de tecnologías para determinar la ubicación de objetos o personas. Para ello se tiene que determinar la posición de estos en un lugar en el espacio, entendiendo esto como unas determinadas coordenadas geográficas sin ningún tipo de dato más asociado ubicación. Los sistemas de posicionamiento vía satélite, GNSS (*Global Navigation Satellite System*) usan una constelación de satélites en el espacio que se comunica con el dispositivo de posicionamiento para determinar la posición de este.

Para este fin, el sistema utiliza la visión de al menos cuatro satélites para determinar su posición. Cada uno de ellos envía información al receptor, que calcula la posición en base a la diferencia de la distancia calculada en base a los retardos de las señales que envían los satélites. Los sistemas de posicionamiento americano GPS (*Global Positioning System*) y el sistema ruso GLONASS (*Global'naya Navigatsionnaya Sputnikovaya Sistema*) son los más establecidos y los que usan nuestros dispositivos actualmente, aunque hay otros sistemas en desarrollo como el europeo GALILEO o chino COMPASS entre otros.

Por último, se tiene que asociar posición con algún lugar conocido para poder hacer uso de esta información. El término geotelemática engloba la tecnología de posicionamiento con la localización en la Tierra. Pero para geolocalizar hace falta la información que asocie esos números (coordenadas cartográficas) con unos datos geográficos que ubiquen esa posición en un lugar conocido. Por tanto, hace falta que el dispositivo, o bien disponga de esos datos que relacionen a esa posición o bien pueda acceder de forma remota a ellos a través de alguna red de comunicaciones pública o privada a esos datos; el uso de *Google Maps* en nuestro terminal móvil es un ejemplo claro de esto último.

2.4 ARDUINO

Arduino es una plataforma electrónica de programación de arquitectura abierta realizada sobre un microcontrolador que permite el desarrollo de aplicaciones en ciencias exactas y en el área de ingeniería. A través de un entorno de programación IDE permite desarrollar aplicaciones en lenguajes C y C++ para automatizar procesos físicos. Es un sistema empotrado que gira en torno a la familia de microcontroladores ATMEL de 8 bits, puede ser empleado para automatizar procesos a través de sensores y módulos de ampliación adicionales que van desde aplicaciones de robots, industriales, comunicaciones, etc.

El microcontrolador contiene memoria RAM para almacenar variables, parámetros y el programa en ejecución y memoria flash para almacenar el código máquina del *sketch*. Además, incluye memoria EEPROM para almacenar un pequeño sistema de arranque que inicializa la tarjeta, controla la comunicación con el pc, contiene el programa cargador que recibe del entorno IDE, entre otras funciones básicas de la placa.

La filosofía de trabajo de Arduino es de arquitectura abierta, lo que significa que la programación se puede adaptar al proceso físico en función de sus características. El entorno de programación (IDE) se puede obtener de manera gratuita. La placa Arduino se comunica a través de USB con el pc para su programación. Cada programa se denomina *sketch* y es el IDE el encargado de realizar la compilación, depuración de

código y la traducción a código máquina para el controlador. La placa Arduino es la encargada de ejecutar el *sketch*.

2.4.1 ARQUITECTURA AVR

Los microcontroladores Atmel de las placas Arduino y sus compatibles de 8 y 32 bits obtienen su nombre de sus desarrolladores: AVR (*Alf-Egil Bogen Vegard Wollan*). Su arquitectura es RISC (*Reduced Instruction Set Computing*).

La arquitectura AVR dispone de dos tipos de buses: un bus de instrucciones y otro para leer y escribir datos. EL primero lee y ejecuta las instrucciones de programa en la CPU (unidad central del proceso) y el otro bus lee y escribe datos. Esto permite ejecutar una instrucción cada ciclo de reloj y elimina tiempos de espera cuando no hay instrucciones a procesar.

Además, la arquitectura AVR dispone de registros a nivel de CPU que permiten almacenar datos de forma dinámica sin tener que recurrir a la memoria SRAM (*Static Random Acces Memory*), por lo que se ahorran ciclos de reloj también por esta parte. La velocidad de estos registros es lo suficiente rápida para que en un ciclo de reloj se lea, almacene y ejecute la información en un solo ciclo de reloj del microcontrolador y permiten realizar operaciones a nivel interno de la CPU: la unidad aritmético lógica (ALU) se encarga de hacer operaciones lógicas y matemáticas utilizando este tipo de registros.

El cerebro electrónico es la CPU que es la encargada de ejecutar instrucciones de programa y que esta se realice de forma adecuada. El contador de programa (*IP instruction pointer o PC program counter*) se encarga de apuntar la instrucción a ejecutar de forma secuencial.

Por otro lado, se dispone de los diferentes tipos de memoria: memoria flash (no volátil) para almacenar el programa o *sckech*, memoria dinámica SRAM para la ejecución del programa y almacenamiento de variables y memoria EEPROM (*electrically erasable programable read-only memory*) donde se almacena un pequeño programa para determinar el funcionamiento de circuitos periféricos.

La comunicación con el exterior se realiza a través de los puertos de entrada-salida de propósito general, donde además algunos de ellos tienen características que los hacen

específicos para diferentes protocolos de comunicación: comunicación serie, modulación por pulsos, determinado tipo de buses, etc. Entre estos están:

- USART (*universal synchronous and asynchronous receiver-transmitter*): Comunicación serie síncrona o asíncrona de la que podemos disponer de al menos un puerto serie con el que comunicamos con la computadora o tener adicionales para comunicación con otros dispositivos.
- I²C (*inter integrated circuit*) bus serie también conocido con TWI (two wire interface) para la versión de Atmel. Se explica más adelante en el apartado 3.2.4 (uso LCD a través de módulo I2C)
- SPI (*serial peripheral interface*) Es un bus síncrono. Se detalla su funcionamiento en el apartado 3.2.5 (comunicación con el módulo SD).

En la arquitectura AV existen tres temporizadores/contadores de propósito general con diferentes modos de comparación y manejo de interrupciones (una interrupción es una solicitud de servicio que detiene la ejecución de un programa para realizar otra actividad y cuando acaba esta vuelve al programa principal para continuar en la instrucción donde se había detenido). Dispone además de un convertidor analógico/digital ADC (*analogic digital converter*) que emplea la técnica de aproximaciones sucesivas, con lo que es necesario una señal de referencia. El ADC tiene su propia fuente de alimentación (ACVCC). Por último, existe un dispositivo programable llamado guardián de eventos o perro guardián (*watchdog timer*) que se encarga de monitorizar mediante una temporización el correcto funcionamiento del programa. En caso de que no se ejecute en el tiempo previsto es que existe algún problema, por lo que puede utilizarse para reiniciar el programa, por ejemplo.

Todos los dispositivos conectados al bus de datos (*data bus*): memoria flash, EEPROM, convertidor A/D, direcciones entrada/salida, TWI, USART, SPI tienen asignadas direcciones de memoria. Por último, la señal de reset es considerada como una interrupción que tiene la mayor prioridad sobre el microcontrolador, deteniendo el programa en y llevando el contador a la posición de memoria inicial para ejecutar de nuevo el programa.

Una interrupción es una solicitud de servicio con la que el microcontrolador detiene temporalmente la ejecución del programa para realizar una actividad demandada por un suceso externo (puede ser el flanco ascendente o descendente de una entrada física).

Esta actividad es la rutina de servicio de interrupción, que se ejecutará para después volver al programa principal y continuar su ejecución normal. Se establecen diferentes prioridades de interrupción para atender ordenadamente su demanda.

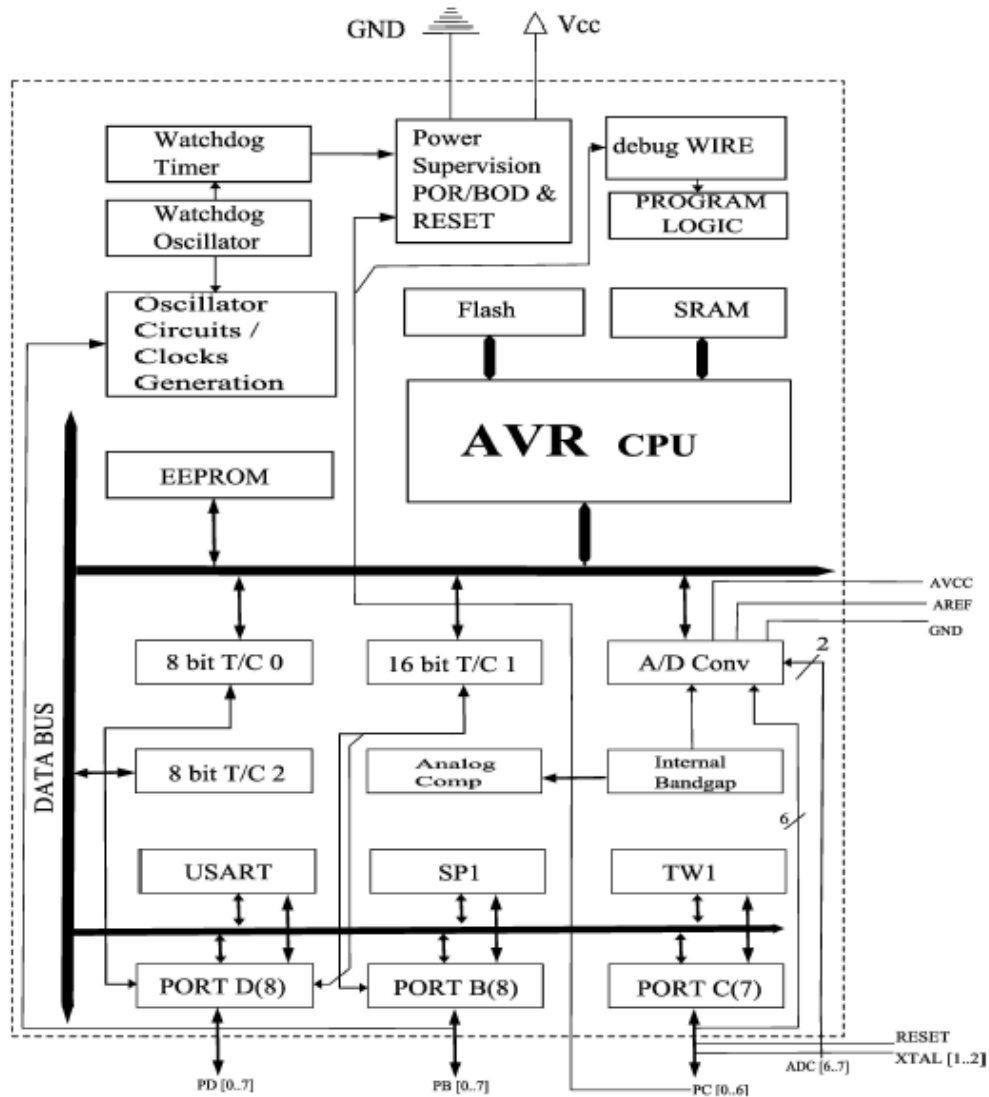


FIGURA 1: ARQUITECTURA AVR DE ARDUINO

2.5 MEMORIA FLASH

La memoria flash es un tipo de memoria no volátil y re escribible. Posee muchas de las capacidades de la memoria RAM, pero sus datos se almacenan en bloques o celdas en vez de bytes. Debido a su velocidad, bajo consumo y durabilidad su uso en la actualidad

está presente en la mayoría de dispositivos electrónicos actuales: cámaras, teléfonos móviles, ordenadores de sobremesa y portátiles, etc.

La ventaja del uso de este tipo de memoria frente a la memoria EEPROM disponible en la placa Arduino reside en la mayor capacidad y número de ciclos útiles de escritura además de su portabilidad.

Existen varios formatos de los cuales el más extendido actualmente es el formato SD (*Secure Digital*) y micro SD (también existe la mini, pero su uso no se ha extendido tanto como las anteriores). El tamaño micro SD es muy utilizado en teléfonos móviles y tabletas. Dentro de estos formatos existen cuatro familias atendiendo a su capacidad:

- Capacidad estándar (*SDSH Secure Digital Estándar Capacity*) De 1MB a 4 GB. Primeras en salir en el año 1999 con especificación 1.0. con rendimiento de bus de 12,5 a 25 MB/s
- Alta capacidad (*SDHC Secure Digital High Capacity*). De 2 a 32 GB. Año 2006. Especificación 2.0 y velocidad de bus de 12,5 a 104 MB/s
- Capacidad extendida (*SDXC Secure Digital Extended Capacity*). De 32 GB a 2TB. Año 2009. Especificación 3,01 y 4 y velocidad de bus de 156 a 312 MB/s

2.6 APLICACIONES SIMILARES AL PROYECTO

En los diferentes estudios realizados sobre el tema de adelantamientos a ciclos se han utilizado dispositivos diseñados para el trabajo en cuestión, pero luego solamente se ha encontrado comercializado un producto para realizar estas mediciones; el dispositivo C3FT (*See-Three-Feeds*) de *Codaxus LLC* situada en Austin (Texas), que comercializa el producto a través de su página web: codexus.com.

La última versión del dispositivo (C3FT v3) consiste básicamente en una unidad medidora por ultrasonidos que se monta en el manillar de la bicicleta, que dispone de un display donde registra la medida del vehículo que adelanta y avisa mediante sonido o luz si ciertos umbrales configurables de (20 a 250 cm) de medida son rebasados. Adicionalmente se puede conectar una cámara para registrar video o fotografía con función de disparo automático. El dispositivo está alimentado por una batería de ion litio.

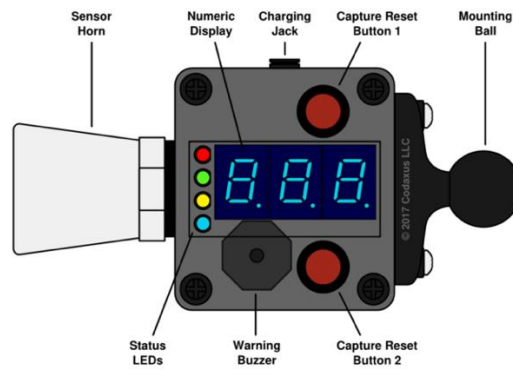


FIGURA 2: DETALLE DEL DISPOSITIVO C3FT V3 (IMAGEN PROCEDENTE DE CODEXUS.COM)

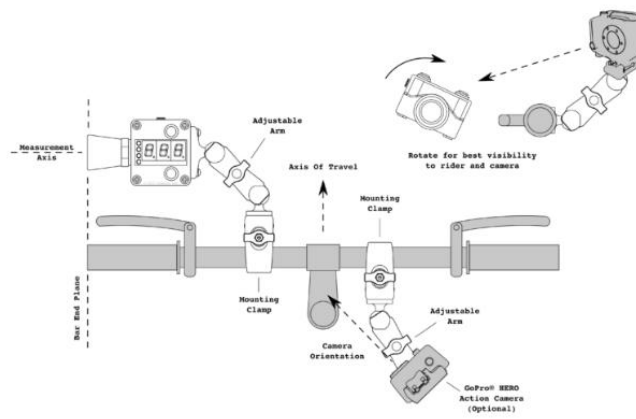


FIGURA 3: DETALLE DE MONTAJE C3FT V3 (IMAGEN PROCEDENTE DE CODEXUS.COM)

CAPITULO 3: PROPUESTA DE PROYECTO

3.1 INTRODUCCIÓN

El dispositivo a diseñar gira en torno a la plataforma Arduino con los módulos de extensión necesarios para realizar las mediciones de distancia por ultrasonidos, localización del adelantamiento indebido a través de un módulo GPS y posterior archivo de datos en una tarjeta SD mediante un *shield* SD. Se utilizará un *shield* LCD para visualizar la medida lateral en el momento que la distancia de adelantamiento sea inferior a 1'5 metros. Además, esta medición inferior a un metro y medio será el nivel de disparo para activar una salida que a través de un módulo de relés podrán activar dispositivos externos como pueda ser una cámara.

El hecho de almacenar los datos en una tarjeta SD permite dotar de independencia completa al dispositivo. De este modo, no es necesario depender de ningún dispositivo externo como pueda ser un teléfono móvil.

El diagrama de bloques del dispositivo a nivel físico:

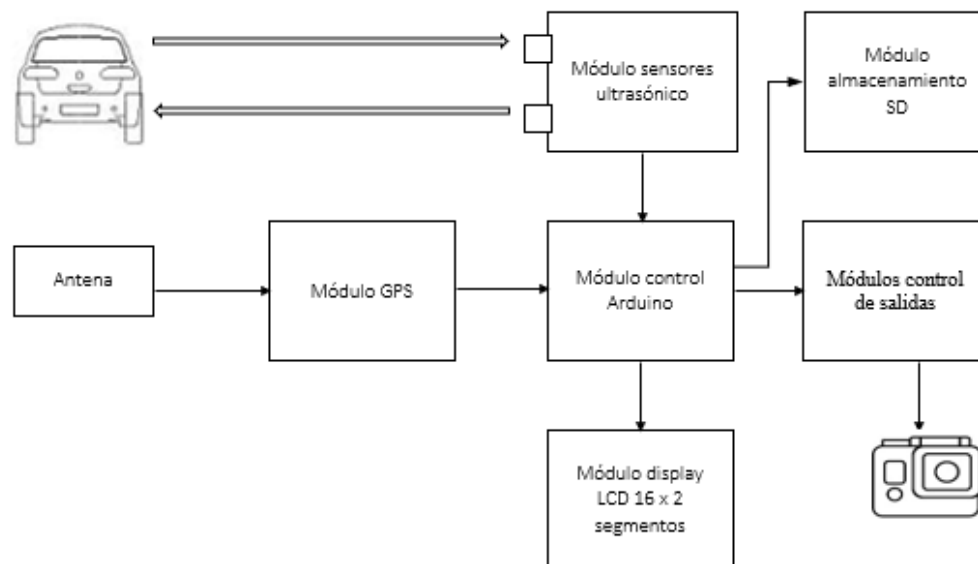


FIGURA 4: DIAGRAMA DE BLOQUES DEL DISPOSITIVO

3.2 DESCRIPCION DE LOS BLOQUES HARDWARE

Se utilizarán componentes reales incluso en esta fase del proyecto, de forma que sea posible la explicación de la tecnología necesaria para la interconexión y funcionamiento de los bloques de hardware.

3.2.1 PLACA DE CONTROL

La placa de control será la que con el soporte del software necesario gestionará los demás módulos. Para ello se buscará una placa de la marca Arduino o compatible que reúna las características necesarias para poder gestionar los demás dispositivos con el menor tamaño y coste posible, para poder integrarse en un dispositivo que ocupe poco espacio. Por ejemplo, la placa Arduino nano y compatibles tiene unas características en cuanto a consumo, tamaño con la capacidad suficiente para gestionar los dispositivos.

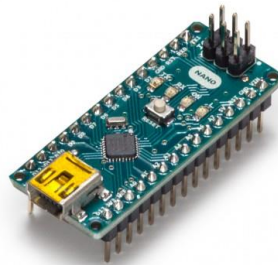


FIGURA 5: ARDUINO NANO ([HTTPS://STORE.ARDUINO.CC/ARDUINO-NANO](https://store.arduino.cc/arduino-nano))

Características técnicas:

- Microcontrolador: ATmega328
- Arquitectura: AVR
- Voltaje de operación: 5 V
- Memoria Flash: 32 KB (2 KB usados por el gestor de arranque)
- SRAM: 2 K
- Velocidad de reloj: 16 MHz

- Entradas/ Salidas analógicas (I/O): 8
- EEPROM: 1KB
- I/O corriente DC: 40 mA (pines I/O)
- Voltaje de entrada: 7-12V
- Entradas/ Salidas digitales: 22
- Salidas PWM (*Pulse Width Modulation*: modulación de ancho de pulso): 6
- Consumo: 19 mA
- Tamaño placa: 18 x 45 mm
- Peso: 7 g

Otra posibilidad a valorar es la utilización de una placa de mayor capacidad dado los requerimientos del uso del módulo de tarjeta SD, que como se verá más adelante sólo este hace uso de al menos un sesenta por cien de la memoria dinámica. En concreto se planteará el uso de una tarjeta compatible con Arduino Mega, pero del fabricante *Elegoo*. Al igual que Arduino Mega está basada en el microcontrolador *ATMega 2560* y dispone de mayor capacidad tanto de entradas y salidas como de memoria RAM y almacenamiento para el sketch.

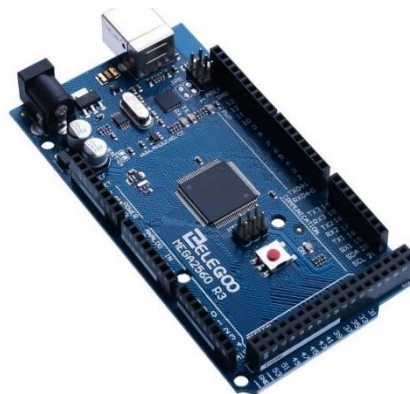


FIGURA 6: PLACA COMPATIBLE ELEGOO MEGA 2560

Características técnicas:

- Microcontrolador: ATmega2560
- Voltaje Operativo: 5V

- Voltaje de Entrada: 7-12V
- Pines digitales de Entrada/Salida: 54 (de los cuales 15 proveen salida PWM)
- Pines analógicos de entrada: 16
- Corriente DC por cada Pin Entrada/Salida: 40 mA
- Corriente DC entregada en el Pin 3.3V: 50 mA
- Memoria Flash: 256 KB (8KB usados por el *bootloader*)
- SRAM: 8KB
- EEPROM: 4KB
- Velocidad de reloj: 16 MHz
- Dimensiones: 100 x 50 mm

3.2.2 MÓDULO GPS

Para la elección de este módulo se escoge otro fabricante de *shields* UBLOX del cual existe bastante documentación. Se comunica mediante puerto serial UART ofreciendo datos de latitud, longitud, velocidad, altitud, fecha y hora en formato NMEA (*National Marine Electronics Association*).

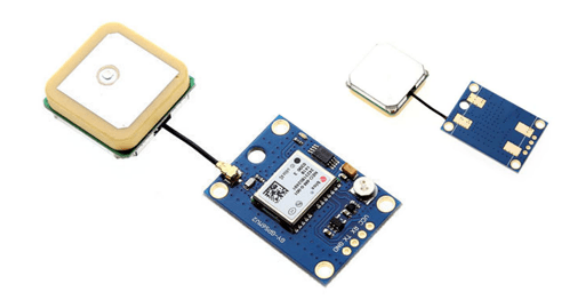


FIGURA 7: MÓDULO GPS NEO 6

El chip GPS viene montado en un PCB con una EEPROM que tiene la configuración de fábrica de los parámetros y una antena cerámica. Además, dispone de un regulador de tensión que le permite la alimentación de 3,3 a 5V. La tasa de símbolos por segundo por defecto es de 9600 baudios

Características técnicas:

Tipo de receptor	L1 frequency band, C/A code 22 Tracking / 66 Canales de lectura	
Sensibilidad	Tracking Adquisición	-165dBm -148dBm
Precisión	Posición Velocidad Timing	3mts. 3D RMS sin SA 0.1m/s sin SA 60ns RMS
Tiempo de lectura	Cold Start Warm Start Hot Start Re-Acquisition	36s 33s 1s <1s
Consumo de energía	Tracking Adquisición Sleep/Standby	<30mA @ 3V Vcc 40mA TBD
Frecuencia de actualización de datos de navegación	1Hz	
Límites de operación	Altitud Velocidad Aceleración	Max 18,000m Max 515m/s Menor a 4g
Especificaciones de antena	Medidas Frecuencia Ancho de banda Impedancia Ratio Axial Polarización	18.2 x 18.2 x 4.0 mm 1575 ± 3 MHz 10 MHz min 50 Ω 3 dB max RHCP
Dimensiones y peso	Dimensiones Peso	30mm x20mm x 11.4mm 9g
Fuente de alimentación	VCC Corriente	5V ±5% 55mA(typical)
Condiciones ambientales	Temperatura operación Humedad	40 ~ +85 (sin batería de respaldo) 0 ~ +125

3.2.3 MÓDULO DE ULTRASONIDOS

El módulo a usar en las pruebas iniciales se hará con un sencillo módulo multi-fabricante HC-SR 04 que es un módulo emisor-receptor de ultrasonidos que trabaja a 40 KHz y puede operar en un rango de 2 cm a 400 cm, que nos puede ser válido para la detección del umbral 150 cm necesario además de poder ofrecer una medida bastante exacta en el display LCD tras los cálculos necesarios como se comentó en el apartado anterior (sistemas de medición por ultrasonidos).

Las características de este sensor no se ven afectadas por la luz, pero si a materiales finos como la tela ya que pueden ser atravesados de forma sencilla por los ultrasonidos, que no es el caso ya que tenemos que detectar un vehículo. La velocidad del sonido en el aire a unos 20°C es de 343 m/s, aunque por cada grado centígrado la velocidad del sonido aumenta en 0,6 m/s por lo que puede afectar a la distancia medida y habría que tenerlo en cuenta dependiendo de la precisión que se desee. La superficie de detección debe ser de 0,5m² para una detección fiable.

La variación de la velocidad del sonido en el aire determinada por su temperatura, donde la variación de esta resulta de 0,6 m/s por cada grado centígrado. La expresión para conseguir fácilmente el cálculo es la siguiente:

$$C = 331 + (T * 0,6)$$

Siendo C la velocidad del sonido y T la temperatura en grados centígrados.

La forma de compensar este error sería medir la temperatura con un sensor de temperatura y efectuar la compensación vía software o encontrar un sensor que ya incorpore incluida una sonda y corrija la diferencia de tiempos, de forma que no se tendría que controlar esto vía software; el tiempo de vuelo entre la señal que envía (disparo) y que la que recibe (eco) permanece constante, aunque varíe la temperatura.

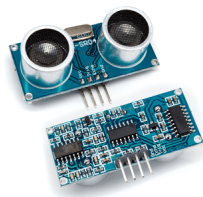


FIGURA 8: SENSOR ULTRASONIDOS HC SR-04

El sensor dispone de 4 pines dos de los cuales son de alimentación (5V) y los otros dos corresponden uno a la entrada (*Trigger*) y otro a su salida (*Echo*). El funcionamiento es el siguiente:

1. Enviar un Pulso "1" de al menos de 10uS por el Pin *Trigger* (Disparador).
2. El sensor enviará 8 Pulsos de 40KHz) al transductor (altavoz emisor) y coloca su salida *Echo* a nivel alto (puesta a 1). Se debe detectar este evento e iniciar un conteo de tiempo por parte del microcontrolador.
3. La salida *Echo* se mantendrá en alto hasta recibir el eco reflejado por el obstáculo por parte del receptor (micrófono ultrasónico) a lo cual el sensor pondrá su pin Echo a nivel bajo (0 V), es decir, terminar de contar el tiempo.
4. Se recomienda dar un tiempo de aproximadamente 50 ms de espera después de terminar la cuenta.
5. La distancia es proporcional a la duración del pulso y puedes calcularla con las siguiente formula (Utilizando la velocidad del sonido = 340m/s): Si se tiene en cuenta que la distancia que recorre la onda es dos veces la distancia del objeto:

$$Distancia (m) = Tiempo (s) \cdot \frac{343 m/s}{2} = Tiempo(s) \cdot 171,5$$

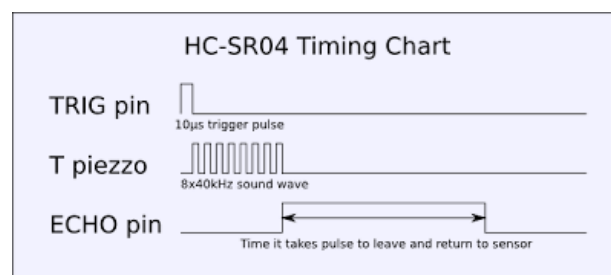


FIGURA 9: FUNCIONAMIENTO TEMPORAL DEL HC SR-04

Características técnicas:

- 4 pines:
 - VCC: Alimentación +5V (4,5V – 5,5 V)
 - TRIGGER: Disparador, entrada del sensor (TTL)
 - ECHO: Eco, salida del sensor

- GND: Alimentación 0 V
- Corriente de reposo: < 2mA
- Corriente de trabajo: 15mA
- Ángulo de medición: 30°
- Ángulo de medición efectivo: < 15°
- Detección de 2cm a 400cm o 1" a 13 pies (Sirve a más de 4m, pero el fabricante no garantiza una buena medición).
- “Resolución” La precisión puede variar entre los 3mm o 0.3cm.
- Dimensiones: 45mm x 20mm x 15mm
- Frecuencia de trabajo: 40KHz

3.2.4 DISPLAY 16 x 2 VÍA BUS I2C

El display es el elemento que nos permitirá la visualización de la medida en el momento del adelantamiento. Ya que solo nos interesa la medición dentro del campo de medición del sensor ultrasónico y además solo queremos registrar las medidas que no superen la distancia de seguridad. La elección de este dispositivo ha de tener en cuenta que las condiciones de visibilidad pueden verse alteradas por la luz del sol. Por lo que en principio se usará un display LCD (*Liquid Cristal Display*) con retroiluminación de 16 caracteres x 2 líneas.

Por otro lado, para reducir el número de cables para gestionar el display se usará un bus I2C (*Inter-Integrated Circuit*) de la placa de control a través para reducir a 2 el número de cables necesarios para comunicar el LCD a través de un controlador I2C para LCD. I2C es un estándar de mercado y permite multiplexar datos y enviarlos mediante comunicación serie síncrona a través de dos cables: señal de reloj (CLK) y datos (SDA). Es posible direccionar hasta 128 dispositivos a través de un bus I2C (2^7), ya que disponemos de 7 bits para denominar cada uno de ellos. Uno de ellos debe ser configurado como maestro y los demás como esclavos.

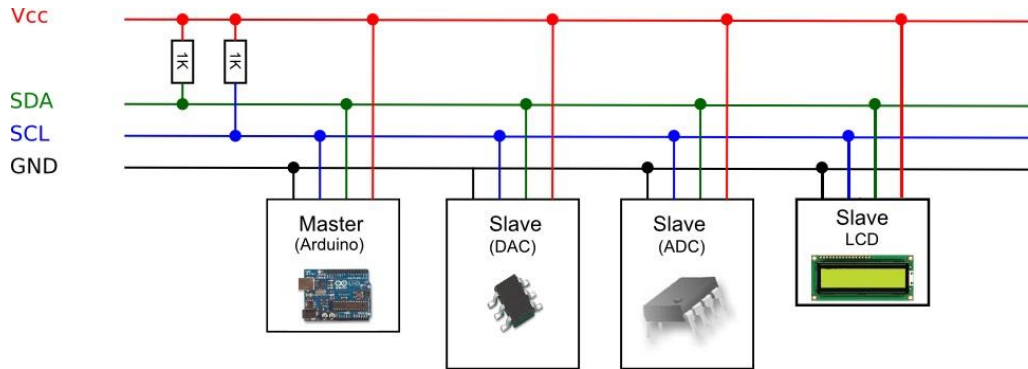


FIGURA 10: BUS I2C

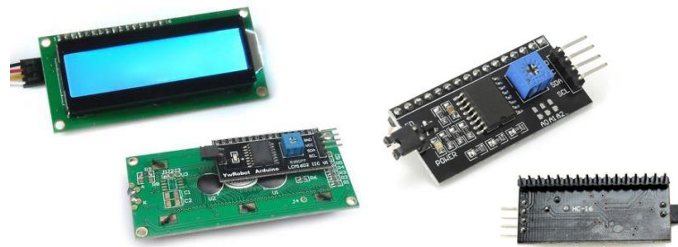


FIGURA 11: DISPLAY LCD 16 X 2 Y CONTROLADOR I2C PARA ARDUINO

Principales características del display 16 x 2 Hitachi HD44780 o compatibles:

- Pantalla de caracteres ASCII, además de los caracteres Kanji y griegos
- Desplazamiento de los caracteres hacia la izquierda o la derecha
- Proporciona la dirección de la posición absoluta o relativa del carácter
- Memoria de 40 caracteres por línea de pantalla
- Movimiento del cursor y cambio de su aspecto
- Permite que el usuario pueda programar 8 caracteres
- Conexión a un procesador usando un interfaz de 4 u 8 caracteres
- Tensión de alimentación: +5V

Características controlador I2C serial PCF8574 para pantalla:

- Basado en el expansor I/O PCF8574
- Solamente 4 líneas en total para conectar una pantalla al microcontrolador

- Compatible con pantalla LCD 16x2 o 20x4
- Utiliza el protocolo I2C, por lo que puede compartir el bus con otros dispositivos

3.2.5 MÓDULO ALMACENAMIENTO SD

EL módulo micro SD es genérico, funciona con tensiones de alimentación de 3,3V o 5V, por lo cual es compatible para ser alimentado como los demás módulos que componen el sistema (5V). Admite tarjetas micro SD de hasta 2GB y micro SDHC (de alta velocidad de hasta 32GB). Deben ser formateadas en FAT16 o FAT 32.

La ventaja de este tipo de memoria es que aparte de no ser volátil puede ser extraída y conectada a un ordenador con facilidad, pero por el contrario el nivel de exigencia para Arduino es muy alto, por lo que se deberá verificar el correcto funcionamiento y en caso necesario emplear una placa Arduino con más capacidad de procesamiento, memoria estática y memoria dinámica.

EL bus SPI es un bus serie síncrono que es un estándar de facto. Tiene una arquitectura maestro-esclavo donde la comunicación es full-dúplex por lo que requiere de las siguientes líneas de comunicación:

- **SCK:** Señal de reloj enviada por el maestro
- **MOSI:** (*Master-Out, Slave-In*) Comunicación del maestro al esclavo
- **MISO** (*Master-In, Slave-Out*) para comunicación del esclavo al maestro
- **SS** (*Slave Select*) Línea para seleccionar el dispositivo con el que se va a realizar la comunicación.



FIGURA 12: MÓDULO MICRO SD

Características técnicas:

- Tipos de tarjeta soportado: micro SD (hasta 2GB), micro SDHC (hasta 32GB)
- Tensión de alimentación: De 4,5V a 5,5V mediante regulador integrado o 3,3V.
- Consumo de corriente: 80 mA típico (0,2 mA mínimo, 200 mA máximo)
- Pines:(GND, VCC, MISO, MOSI, SCK, CS). Vcc y GND son la alimentación y CS es equivalente al SS explicado en el párrafo anterior.
- Dimensiones: 42 x24 x12 mm
- Peso: 5g

3.2.7 CONSIDERACIONES ADICIONALES SOBRE EL HARDWARE

Se han presentado los módulos más importantes del hardware ofreciendo ejemplos que pueden ser finalmente usados para el desarrollo del proyecto, aunque estos pueden verse modificados con la finalidad de mejorar las propiedades del producto final como pueden ser el consumo, peso o mejora de alguna de las características del dispositivo.

La alimentación del dispositivo se hará mediante una batería externa recargable para facilitar su portabilidad que se determinará en el apartado de diseño cuando se dispongan de todos los cálculos y componentes finales. Puede ser necesario adecuar la tensión a alguno de los módulos o de la propia batería en función del hardware.

El control de dispositivos externos puede realizarse mediante una placa de relés o bien mediante dispositivos semiconductores (transistores, tiristores u optoacopladores) dependiendo de los dispositivos que se quiera controlar. Esto se determinará también en la fase de diseño tras los cálculos y necesarios.

Si finalmente no es posible el montaje interior del módulo de control de las salidas se habilitarán conexiones en el dispositivo que posteriormente podrán ser utilizadas por el dispositivo a activar. Luego atendiendo a las características propias de activación /desactivación del dispositivo se tendrá que habilitar-deshabilitar esas salidas por software y controlar externamente de modo que exista un aislamiento eléctrico adecuado mediante los métodos expuestos en el párrafo anterior.

Será necesario el uso de pulsadores, interruptores y hardware adicional en función al control externo que se quiera hacer del dispositivo. Finalmente, todos los módulos irán encastrados en una caja plástica con la protección necesaria para su uso en exterior y su correspondiente anclaje al tipo de vehículos al que va destinado.

3.4 SOFTWARE NECESARIO

EL entorno de desarrollo IDE necesario para programar la lógica de todo el sistema es de código abierto y gratuita. Se descarga a través del sitio www.arduino.cc . El lenguaje necesario para la programación de Arduino se basa en lenguaje C y C++. Los programas se denominan *sketchs*. El IDE se encarga de generar el código máquina que se descargará a través del puerto de comunicación serie USB a la placa del microcontrolador, que se quedará residente en la placa, aunque esta pierda la alimentación ya que está dispone de memoria Flash donde se encuentra el gestor de arranque y se almacena el *scketch*, además de EEPROM (*Electrically Erasable Programmable Read-Only Memory*) disponible para otros usos.

El puerto USB además del dispositivo puede comunicarse con el IDE para mandar información al PC de cualquier tipo de tarea que realice el sistema empotrado diseñado. Se puede requerir el uso de librerías específicas adicionales para el control de los *shields*: módulo ultrasonidos, módulo GPS módulo I2C para el LCD y módulo micro SD.

En el caso de delegar por hardware la localización GPS (el dispositivo diseñado se comunicaría por *Bluetooth* con un dispositivo con sistema operativo *Android* sería necesario implementar una aplicación móvil, que en cierta forma simplificaría tanto la parte de hardware como la de software, teniendo en cuenta que los dispositivos móviles engloban ya parte de las características tecnológicas que queremos hacer uso con el dispositivo a diseñar.

CAPITULO 4: DISEÑO DEL DISPOSITIVO: MONTAJE Y PRUEBAS

En primer lugar, se harán pruebas de cada uno de los módulos por separado para verificar su correcto funcionamiento y para facilitar el entendimiento del funcionamiento del dispositivo final. Se utilizan adicionalmente librerías para dar soporte a los *shields* utilizados en el diseño.

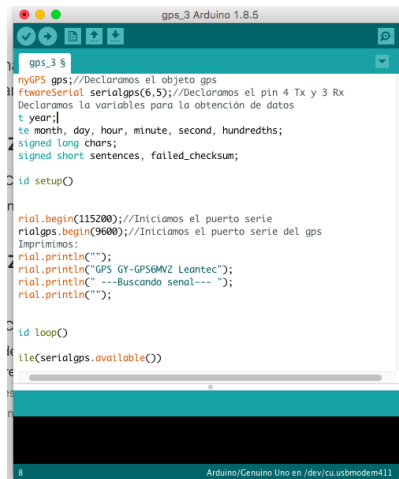
Una vez probado cada uno de los módulos por separado se interconectarán los módulos, y se desarrollará el código necesario para el correcto funcionamiento del dispositivo. Aún en este estadio cabe la posibilidad de sustituir algún módulo o la misma placa de desarrollo en el caso de que hagan falta más recursos o se pueda mejorar el funcionamiento del conjunto.

4.1 ENTORNO DE DESARROLLO Y SOFTWARE DE SOPORTE

Antes de probar cada uno de los módulos se ha de preparar el entorno de desarrollo integrado IDE (*Integrated Development Environment*) de Arduino. Aunque existen otras opciones gratuitas como Eclipse, Visual Studio, Atmel Studio, que incluso pueden añadir facilidades y funcionalidades, en este proyecto se usará el IDE propio de Arduino disponible:

<https://www.arduino.cc/en/Main/Software>

No se entrará en detalles de la instalación del software, ya que dependiendo del sistema operativo puede variar algún paso y dada la facilidad del proceso no se ha visto conveniente en entrar en detalles (en este caso nuestro sistema operativo es OS X El Capitán).



```

gps_3 $
nyGPS gps; //Declaramos el objeto gps
#include <SoftwareSerial.h>
SoftwareSerial serialgps(6,5); //Declaramos el pin 4 Tx y 3 Rx
Declaramos la variables para la obtención de datos
int year;
int month, day, hour, minute, second, hundredths;
signed long chars;
signed short sentences, failed_checksum;

void setup()

{
  serial.begin(115200); //Iniciamos el puerto serie
  serialgps.begin(9600); //Iniciamos el puerto serie del gps
  //Printimos:
  serial.println("");
  serial.println("GPS GY-GPS04V2 Leantec");
  serial.println(" ---Buscando señal--- ");
  serial.println("");
}

void loop()

{
  if(serialgps.available())

```

FIGURA 13: IDE ARDUINO

La instalación de drivers si las placas de desarrollo están firmadas es directa desde el IDE, aunque en nuestro caso hemos usado placas de desarrollo no compatibles que utilizan un integrado diferente no oficial (chip 340G frente a FTDI de Arduino) para la comunicación serie para controlar el USB y por tanto la comunicación con el pc. Los controladores para el chip 340G está disponibles en: <http://www.5v.ru/ch340g.htm>. El uso de este tipo de placas abarata mucho el coste de las placas de desarrollo y no presenta ningún problema en el funcionamiento.

Aunque no se entrará en detalles del funcionamiento del software, cabe detallar que se precisará el uso de librerías adicionales para el funcionamiento de algunos de los módulos y que nos son descargables de manera directa desde el IDE de Arduino, por lo que se hará mención de un importante repositorio público y gratuito (de pago en su versión privada) de código de donde podemos obtener algunas librerías: *GitHub*. Es una plataforma de desarrollo colaborativo de código abierto donde podemos subir código u obtener código de terceros de forma gratuita para usarlo o mejorarlo. Se puede acceder a través de: <https://github.com/>

Por último, hacer referencia una herramienta especializada para la realización de esquemas eléctricos para interconexión entre placas de Arduino, que permite realizar montajes en la placa de pruebas (*protoboard*) si no se tiene mucha experiencia, además de diseñar las placas de circuito impreso de forma automática: *Fritzing*. Está disponible en: <http://fritzing.org/download/>

4.2 PRUEBA DE LOS DIFERENTES MÓDULOS

Con el fin de verificar el funcionamiento los diferentes módulos, se procede a separar y probar de manera independiente las partes que componen el sistema, aunque se decide montar el medidor con el LCD vía I2C en vez de utilizar el monitor serie. Se pueden desglosar las siguientes partes más destacadas:

- Medidor de distancia por ultrasonidos
- Localizador GPS
- Almacenamiento de datos
- Salida de datos por LCD y activación de salidas externas

4.2.1 MEDIDOR DE DISTANCIA POR ULTRASONIDOS

El funcionamiento teórico del medidor de distancia por ultrasonidos se explica en el apartado 3.2.3. Se decide ya que de esa forma tendremos parte del código necesario implementado y se verifica el funcionamiento de los módulos de ultrasonidos HC SR-04, I2C y LCD. Las conexiones con Arduino nano o Uno quedan de la siguiente forma:

- Módulo sensor ultrasonidos:

HC-SR04	Arduino nano o Uno	Color
Vcc	5v	Rojo
Trigger	D6	Verde
Echo	D5	Marrón
GND	GND	Negro

- Módulo I2C con LCD: no presenta problemas ya que es pin a pin.
- Módulo I2C con Arduino:

Módulo I2C	Arduino nano o Uno	Color
GND	GND	Negro
Vcc	5v	Rojo
SDA	SDA (A4)	Naranja
SCL	SCL (A5)	Amarillo

El montaje en la placa de pruebas queda de la siguiente forma:

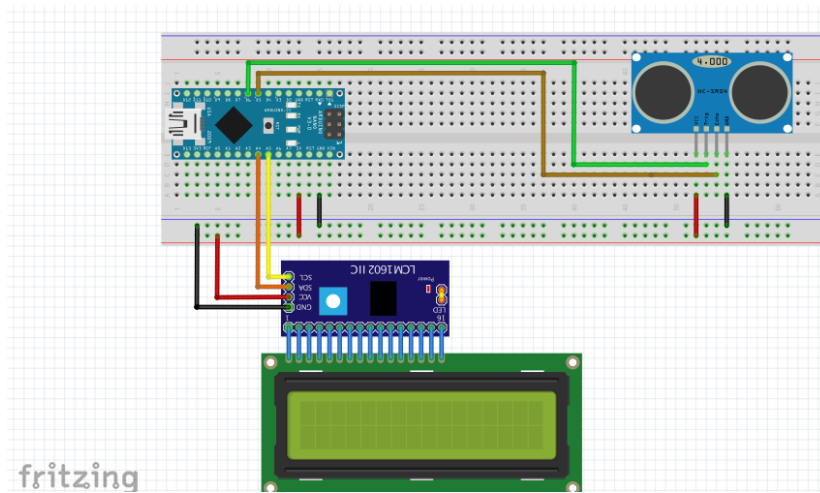


FIGURA 14: MONTAJE EN LA PLACA DE PRUEBAS MEDIDOR DISTANCIA CON LCD

El esquema de montaje realizado también de *Fritzing*:

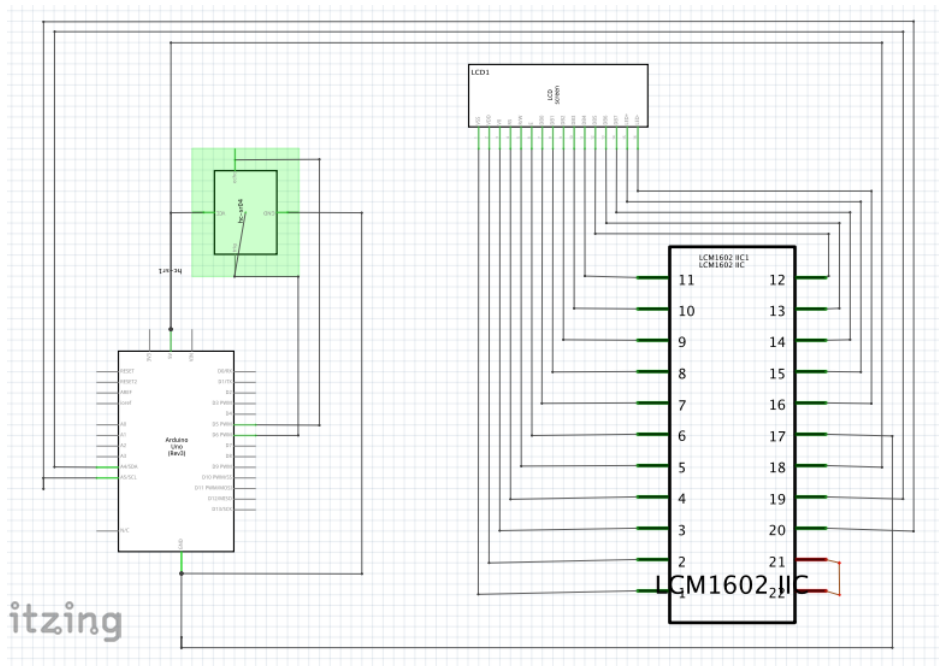


FIGURA 15: ESQUEMA ELÉCTRICO DEL MEDIDOR DISTANCIA CON LCD

Para el funcionamiento del *skech* son necesarias 2 librerías para el control del LCD vía I2C: *Wire.h* y *LiquidCrystal_I2C.h*. *Wire* está disponible en las librerías que trae por defecto el IDE, pero *LiquidCrystal_I2C* está disponible en *GitHub* con el nombre de *Arduino-LiquidCrystal-I2C-library* disponible en:

<https://github.com/fdebrabander/Arduino-LiquidCrystal-I2C-library>

Una vez allí conviene descargar el paquete *zip* que incluye todos los archivos requeridos por la librería más ejemplos para poder realizar pruebas de la siguiente forma: Hacer clic en “*Clone or download*” y después en “*Download ZIP*”.

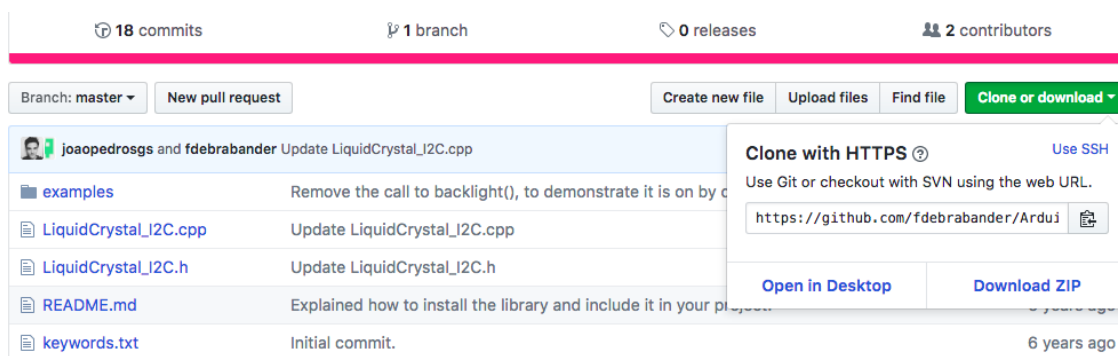


FIGURA 16: LIBRERÍA ARDUINO I2C LCD EN GITHUB

Una vez descargada se procede a instalar el paquete en el IDE de Arduino desde: Programas /Incluir librería/ Añadir librería ZIP.

La librería *Wire.h* contiene las funciones necesarias para controlar el hardware del bus I2C integrado en la placa. Algunas de las funciones son las siguientes:

- *Wire.begin()* // Inicializa el hardware del bus
- *Wire.beginTransmission(address);* //Comienza la transmisión
- *Wire.endTransmission();* // Finaliza la transmisión
- *Wire.requestFrom(address,nBytes);* //solicita un numero de bytes al esclavo en la dirección *address*
- *Wire.available();* // Detecta si hay datos pendientes por ser leídos
- *Wire.write();* // Envía un byte

- *Wire.read(); // Recibe un byte*
- *Wire.onReceive(handler); // Registra una función de callback al recibir un dato*
- *Wire.onRequest(handler); // Registra una función de callback al solicitar un dato*

La librería *LiquidCrystal_I2C.h* dispone las siguientes funciones para el control de hardware de un lcd a través del protocolo I2C. Las principales funciones de esta librería:

- *LiquidCrystal_I2C (lcd_Addr, lcd_cols, lcd_rows); //Función constructor; crea un objeto de la clase LiquidCrystal_I2C, con dirección, columnas y filas indicadas.*
- *init(); //Inicializa el módulo adaptador LCD a I2C, esta función internamente configura e inicializa el I2C y el LCD.*
- *clear(); //Borra la pantalla LCD y posiciona el cursor en la esquina superior izquierda (posición (0,0)).*
- *setCursor(col, row); //Posiciona el cursor del LCD en la posición indicada por col y row(x,y) de forma que establece su posición en pantalla.*
- *print(); //Escribe un texto o mensaje en el LCD.*
- *scrollDisplayLeft(); //Se desplaza el contenido de la pantalla un espacio hacia la izquierda.*
- *scrollDisplayRight(); //Se desplaza el contenido de la pantalla (texto y el cursor) un espacio a la derecha.*
- *backlight(); //Enciende la luz del fondo del LCD*
- *noBacklight(); //Apaga la luz del fondo del LCD*
- *createChar (num, datos); //Crea un carácter personalizado en la pantalla LCD. Se admiten hasta ocho caracteres de 5x8 píxeles (numeradas del 0 al 7). Dónde: num es el número de carácter y datos es una matriz que contienen los píxeles del carácter.*

El código del medidor de distancia con LCD I2C se detalla en los anexos de código. Por un lado, tenemos la función que hace uso del medidor de ultrasonidos. Primeramente, se necesita generar un pulso en el pin de disparo (*Trigger*) de al menos 10 μ S, aunque previamente se pondrá el pin de disparo a cero durante 4 μ S para asegurar una señal limpia.

Después, se utiliza la función “*pulseIn*” para obtener el tiempo de vuelta al sensor. Una vez obtenido este tiempo se utiliza la ecuación vista en el apartado 3.2.3 para la obtención de la distancia (se usan números enteros en vez de en coma flotante para no sobrecargar y ralentizar el procesador):

$$\text{Distancia (m)} = \text{Tiempo (s)} \cdot \frac{343 \text{ m/s}}{2} = \text{Tiempo(s)} \cdot 171,$$

Por otro lado, se presenta la medida en la primera el display LCD activando la luz de fondo. Después se añade la condición de que si la medida es inferior a 1,5 metros en la segunda línea del display se indica que se ha rebasado la distancia de seguridad. En caso de no cumplirse esta condición muestra otra lectura de medida correcta.

Como conclusión a las primeras pruebas, debido a las características del sensor de ultrasonidos, donde puede haber rebotes de la señal el mayor ancho del haz acústico puede dar lugar a medidas erróneas, el sensor HC SR-04 nos sirve para una primera aproximación, de forma que se deberá buscar un sensor de otro tipo (laser) o de más precisión.

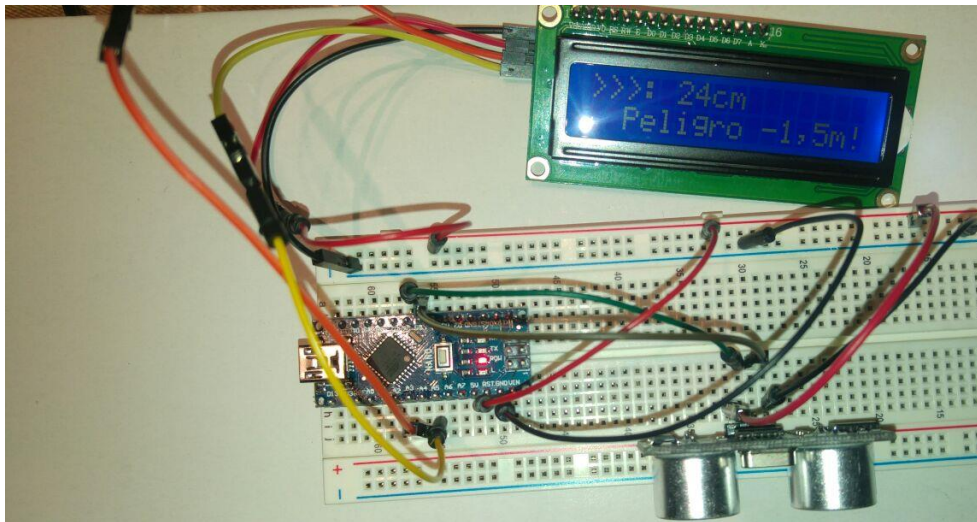


FIGURA 17: PRUEBAS MEDICIÓN DISTANCIA

En un sketch de Arduino podemos encontrar tres partes:

1. Cabecera: Donde se importan bibliotecas necesarias, se declaran variables globales, constantes y objetos.
2. Función *setup* (): Se ejecuta solo una vez tras arrancar el dispositivo o hacer uso de *reset*. Es donde se da valor a las variables, se declaran los pines como entrada o salida, por ejemplos se inicializan los objetos, se invoca a las librerías que se importan. Etc. Es pues, una zona de configuración
3. Función *loop* (): Es una función que se repite indefinidamente y donde reside el código que queremos ejecutar y se ejecuta indefinidamente tras ejecutar una vez la función *setup* ().

Para las pruebas de los módulos LCD y *shield* de ultrasonido el programa quedaría del siguiente modo:

1. Cabecera:
 - a. Importación de librerías necesarias: *Wire.h* y *LiquidCrystal_I2C* para el control del LCD vía I2C. Declaración de constantes de pines de *echo* y *trigger* del módulo ultrasonico
 - b. Declaración del objeto LCD con la dirección de bus I2C preasignada por el fabricante (0x3F) y 2 líneas con 16 caracteres. Esto puede cambiar según el LCD y el fabricante.
2. Función *setup*:
 - a. Se inicializa LCD
 - b. Se enciende retroiluminación del LCD
 - c. Se definen los pines *trigger* como salida y *echo* como entrada
3. Función *loop*:
 - a. Se le pasan los parámetros a la función *ping* , que a su vez retornará el valor de la medida a la variable *cm* desde la función *ping*.
 - b. Se presentan los datos de medida en pantalla en la línea 1.
 - c. Se condiciona la escritura de la segunda línea la escritura de "Distancia OK" o "No OK" en función de si la medida es superior a 1,5m.
 - d. Se da un retardo de 1s para que pueda mostrarse el valor durante 1s.
 - e. La función *ping* realiza el cálculo de la distancia:

- i. Se declaran dos variables locales (uso exclusivo de la función ping) tipo *long* (entero largo) *duration* y *DistanceCm*
- ii. Se pone a nivel bajo el pin de *trigger* durante 4 μ s para posteriormente ponerlo a 1 durante 10 μ s.
- iii. La función *pulseIn* de Arduino recoge el tiempo entre pulsos en microsegundos.
- iv. A partir de la duración se calcula la distancia en centímetros y que es devuelta a la variable *cm* a través de la función ping.

Diagrama de flujo:

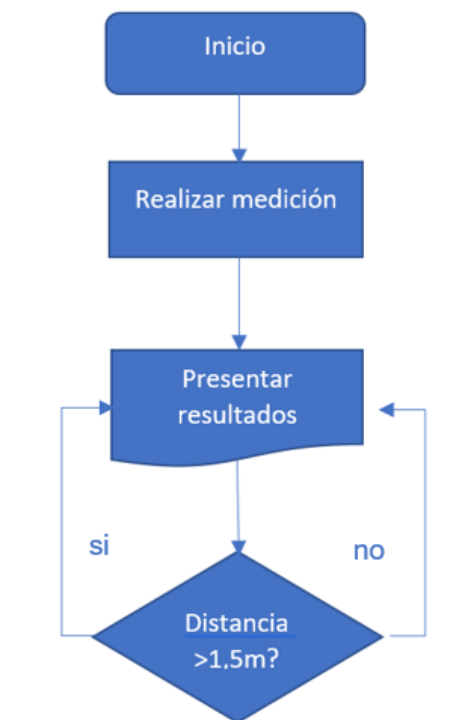


FIGURA 18: DIAGRAMA FLUJO MEDIDOR DE DISTANCIA

4.2.2 LOCALIZADOR GPS Y ESCRITURA DE DATOS EN LA MICRO SD

Para obtener los datos del GPS en una secuencia de datos atendiendo el protocolo NMEA (*National Marine Electronics Association*) solamente es necesario una librería que ya está disponible en el IDE: *SoftSerial.h* con lo que creamos un puerto serie en las entradas digitales que se seleccionen. Lo que se obtiene es una trama con los siguientes valores:

```
$GPRMC, hhmss.ss,A,III,II,a;yyyyy.yy,a,x.x,x.x,ddmmyy,x.x,a]hh
```

Donde cada uno de los valores obtenidos:

hhmss.ss	Hora UTC
A	Estado receptor (A = OK, V = warning)
III.II,a	Latitud (a = N o S)
yyyy.yy,a	Longitud (a = E o W)
x.x	Velocidad en nudos in knots
x.x	Curso en grados
ddmmyy	Fecha UT
x.x,a	Variacion magnética en grados (a = E o W)

Para extraer los datos de la trama utilizaremos la librería *TinyGPS* donde ya se ha facilitado este trabajo disponible en: <https://github.com/mikalhart/TinyGPS>

Además, se escribirán los datos en una tarjeta micro SD, por lo que se requiere el uso de una librería adicional *SD.h* para el manejo de este *shield* incluida en el IDE de Arduino.

Las librerías usadas por tanto en este apartado son: *SoftwareSerial.h*, *SD.h*, *TinyGPS*. Las dos primeras están incluidas en el IDE de Arduino. La librería *TinyGPS.h* facilita la

obtención de datos para su uso en cualquier aplicación del módulo GPS neo6. Las funciones principales:

- *Gpsneo (rx, tx); //Constructor. Si no se pone nada en baudrate o establece el baudrate de comunicación por defecto*
- *Google (char *link); //Ofrece un link para Google Maps.*
- *getDataGPRMC (); //Obtiene los datos del GPS*
- *convertLongitude (char *longitude, char *destination); //Convierte una longitud dada en formato NMEA a grados*
- *convertLatitude (char *longitude, char *destination); //Convierte una longitud dada en formato NMEA a grados*

La ventaja al usar esta librería es obvia, ya que facilita la obtención de datos en un formato listo para su uso, sin tener que extraer los datos de la trama NMEA mediante código, lo que supondría mucha más complejidad en nuestro código.

La librería *SoftwareSerial.h* replica por software el funcionamiento de los puertos serie físicos. En los modelos básicos como nano o Uno de Arduino sólo existe un puerto serie y este suele usarse para la comunicación con el PC, por lo que esta librería nos permite habilitar puertos serie adicionales en otros pines para como en este caso la comunicación con el módulo GPS.

La librería *SD.h* permite leer y escribir desde tarjetas SD. Soporta formatos de fichero FAT16 y FAT32 y tarjetas de hasta 32 GB (SDHC). La comunicación usada es SPI. Las funciones principales de esta librería:

- Objeto SD:
 - *SD.begin(cspin); // Iniciar la SD*
 - *SD.exists(filename); // Comprobar si existe un fichero (devuelve true si existe, falso en caso contrario)*
 - *SD.remove(filename); // Borrar un fichero*
 - *SD.open(filepath, mode); // Abrir un fichero: modo FILE_READ para sólo lectura y modo FILE_WRITE para lectura y escritura*
 - *SD.mkdir(directory); // Crear un directorio*
 - *SD.rmdir(dirname); // Eliminar un directorio*

- Objeto File:
 - *file.size(); // Obtener el tamaño de un fichero*
 - *file.available(); // Comprobar si quedan bytes por leer*
 - *file.read(); // Leer un byte del fichero*
 - *file.write(data); // Escribir un byte en el fichero*
 - *file.print(data); // Escribir una variable en un fichero*
 - *file.position(); // Obtener el punto de lectura/escritura actual*
 - *file.seek(pos) // Mover el punto de lectura/escritura actual. Debe estar entre 0 y file.size()*
 - *file.close(); // Cerrar el fichero*

La idea en esta prueba es obtener los datos del GPS cada cierto periodo de tiempo, abrir un archivo en una tarjeta micro SD (previamente formateada en FAT16 o FAT32) y escribir los datos. A través de las funciones del GPS ESPERA_DATOS donde obtenemos monitorizamos el puerto serie hasta obtener la señal. Mediante la función DATOS_GPS se obtienen los datos requeridos haciendo uso de las funciones de la librería *Tiny_GPS.h* necesarias.

Los datos recibidos se van escribiendo en un archivo "prueba.txt" que se crea haciendo uso de las funciones de la librería *SD.h*. EL archivo se abre y cierra cada vez que obtenemos un bloque de datos. En este caso se hace cada segundo. Además, se comprueba que no haya errores y que esté presente la tarjeta SD. El *scketch* de esta prueba está disponible en el anexo de código.

Las conexiones físicas del GPS con Arduino y del módulo quedarían de la siguiente forma en Arduino Uno o nano (en el caso del bus SPI los pines están preestablecidos):

Módulo GPS	Arduino nano o Uno	Color
GND	GND	Negro
TX	RX(4)	Amarillo
RX	TX(3)	Naranja
Vcc	5v	Rojo

Módulo SD	Arduino nano o Uno	Color
GND	GND	Negro
Vcc	5v	Rojo
MISO	MISO (12)	Naranja
MOSI	MOSI (12)	Blanco
SCK	SCK (D13)	Azul
CS	CS(D9)	Amarillo

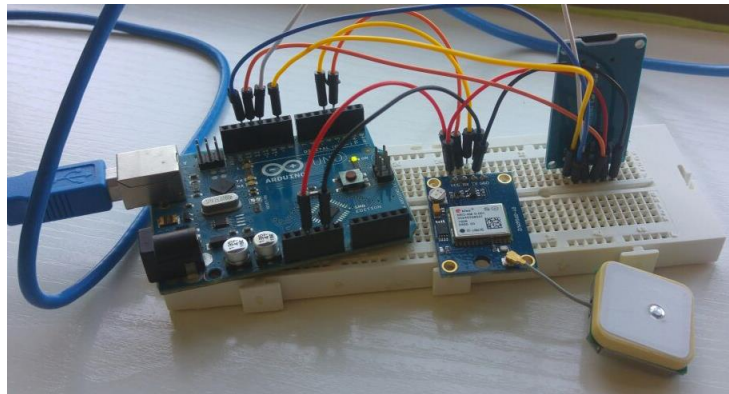


FIGURA 18: PRUEBAS DE GPS Y TARJETA SD

Por el monitor serie podemos observar los resultados del GPS y una vez apagado y extraída la tarjeta se puede ver que se ha creado un archivo "prueba.txt" con los registros de posición, hora y fecha, que serán los que precisemos para el registro del sistema a diseñar:

```

/*LIBRERIAS*/
#include <SoftwareSerial.h>
#include <TinyGPS.h>

Float LAT;
unsigned int YEAR;
byte MONTH;
byte HOUR;
byte MINUTE;
byte SECOND;

// TinyGPS
SoftwareSerial mySerial(2, 3); // RX, TX
TinyGPS gps;

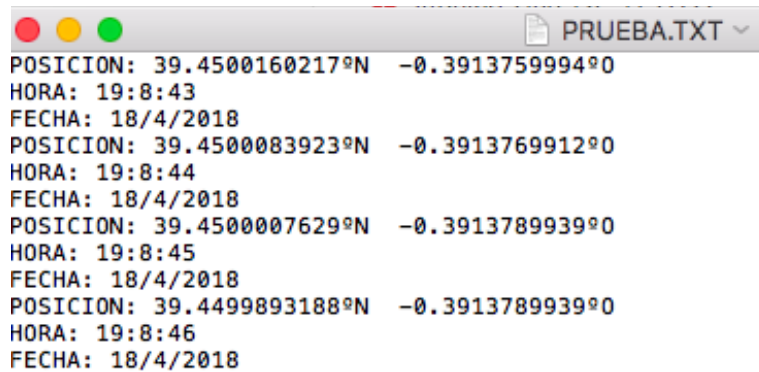
static const int RX_PIN = 2;
static const int TX_PIN = 3;

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

void loop() {
  while (gps.encode(mySerial.read())) {
    if (gps.location_is_valid()) {
      LAT = gps.location.lat();
      YEAR = gps.location.year();
      MONTH = gps.location.month();
      HOUR = gps.location.hour();
      MINUTE = gps.location.minute();
      SECOND = gps.location.second();
      Serial.print("POSICION: ");
      Serial.print(LAT);
      Serial.print(" ");
      Serial.print(gps.location.lng());
      Serial.println(" ");
      Serial.print("HORA: ");
      Serial.print(HOUR);
      Serial.print(":");
      Serial.print(MINUTE);
      Serial.print(":");
      Serial.print(SECOND);
      Serial.println(" ");
      Serial.print("FECHA: ");
      Serial.print(YEAR);
      Serial.print("/");
      Serial.print(MONTH);
      Serial.print("/");
      Serial.print(YEAR);
      Serial.println(" ");
    }
  }
}

```

FIGURA 19: MONITOR SERIAL ARDUINO CON DATOS GPS



```

PRUEBA.TXT
POSICION: 39.4500160217°N -0.3913759994°O
HORA: 19:8:43
FECHA: 18/4/2018
POSICION: 39.4500083923°N -0.3913769912°O
HORA: 19:8:44
FECHA: 18/4/2018
POSICION: 39.4500007629°N -0.3913789939°O
HORA: 19:8:45
FECHA: 18/4/2018
POSICION: 39.4499893188°N -0.3913789939°O
HORA: 19:8:46
FECHA: 18/4/2018

```

FIGURA 20: ARCHIVO TXT GENERADO

El funcionamiento del *sketch* es el siguiente:

1. Cabecera:
 - a. Importación de librerías necesarias: *SoftwareSerial* para crear un puerto serie por software, *TinyGPS* para obtener los datos en un formato adecuado para tratarlos de forma directa y *SD* para la tarjeta de memoria.
 - b. Declaración del a constante del pin *chip selected* CS de la comunicación SPI.
 - c. Declaración de variables para los datos que vamos a obtener a través de *TinyGPS*.
 - d. Creación de los objetos puerto_GPS (puerto serie definido por software gracias a la librería *SoftwareSerial*. creación del objeto *GPS_NEO* para el uso de las funciones de la librería *TinyGPS* y del objeto para la creación del fichero *DATALOGGER*.
2. Función *setup*:
 - a. Se inicializan los puertos serie, tanto el físico como el de hardware, este último para el uso de monitor serial a través del PC.
 - b. Se establece como salida el pin 9 de CS (*chip selected*).
 - c. Se realiza el control de errores de la SD a través de una sentencia condicional.
 - d. Se llama a las funciones *GPS* y *ESPERA_DATOS* para una primera lectura de arranque.
3. Función *loop*:

- a. Se invocan las funciones de GPS y ESPERA_DATOS
- .
- b. Se crea el fichero "Prueba.txt" a través de uso de las funciones de la librería SD que queda abierto para la escritura.
- c. Se escriben los datos obtenidos en la función DATOS_GPS (se escribe y se cierra el archivo); previamente se hace una comprobación de la escritura de estos datos como control de error de escritura a través de una sentencia condicional..
- d. Se muestra la información escrita también en el pc a través del monitor serie.
- e. La función ESPERA_DATOS lee continuamente el puerto serie que hemos asignado al módulo GPS mientras haya datos disponibles (siempre los va a haber si se ha establecido la velocidad en baudios del puerto correctamente a la que tiene configurada el módulo GPS, aunque no tenga señal de GPS).
- f. La función de DATOS_GPS obtiene los datos con la función *data_crack*.

El diagrama de flujo de la escritura de los datos en la SD del GPS:

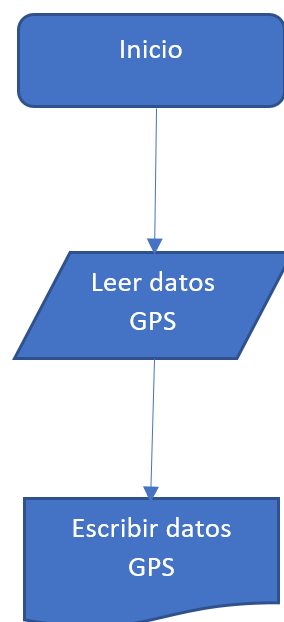


FIGURA 21: DIAGRAMA DE FLUJO GPS-SD

Como conclusiones de este apartado podemos extraer que la ocupación de memoria es de un 68% de almacenamiento del programa y un 62% de la memoria dinámica, lo que supone una carga elevada, si además tenemos que añadir las funciones del LCD-I2C y el medidor de ultrasonidos. Por tanto, cabe la posibilidad de que sea necesario buscar otra placa de desarrollo con mayor capacidad o usar dos de estas, de forma que el programa se divida entre las dos. Un ejemplo de esto último serían las dos pruebas por separado: el medidor, que por un lado active las señales de salida externa y otra adicional de disparo para la segunda placa que se encargaría de gestionar el GPS y la escritura en la micro SD.

4.3 FUNCIONAMIENTO DEL DISPOSITIVO

Una vez propuesto cada uno de los elementos principales de hardware necesarios para el dispositivo se establecerá como debe funcionar el medidor de distancia y establecer el diagrama de flujo que nos ayude a desarrollar el código del sistema.

Para empezar, una vez arrancado el sistema (se alimenta el sistema a través de un interruptor de encendido) se inicia directamente la medición. A su vez, el dispositivo registra continuamente la posición a través del módulo GPS. Cuando se tiene la señal GPS se indica la hora en el display. En caso contrario no aparece la hora y el LCD muestra "Sin GPS" y en el extremo derecho inferior aparece la señal parpadeando BS (buscando satélite).



FIGURA 22: DISPOSITIVO SIN SEÑAL DE GPS

Si la medición está dentro del rango de donde los sensores son fiables (se ha establecido 2,8 metros como rango límite) entonces el display marca la medida.

Si se produce una medición por debajo de la distancia de disparo (1,5m), en cuyo caso se supone un adelantamiento indebido, la distancia y la posición deben quedar registradas en la tarjeta SD ese momento.

Además, se activarán las salidas por el tiempo necesario para controlar el disparo de una cámara de video o fotográfica, por ejemplo. Cuando los datos están siendo grabados en la SD, el LCD muestra DR (datos registrados) en el extremo inferior derecho. En caso de no estar disponible la tarjeta se muestra el mensaje "Error de SD".



FIGURA 23: DISPOSITIVO CON SEÑAL DE GPS GUARDANDO DATOS EN SD

Hay que tener en cuenta que el dispositivo tiene que tener control de falsas mediciones, pues cabe la posibilidad de ser rebasado a menos de 1,5 metros de distancia por otro ciclista o que el dispositivo efectúe una falsa medición, por lo que en ese caso deberíamos tener opción a marcar esa medida como errónea para que quede constancia del hecho y no tenerla en cuenta. A través del pulsador azul podemos marcar la medida como errónea. Está se mostrará en la lectura registrada como "Error = 1". El botón rojo corresponde al *reset* de Arduino (no hubiera sido necesario su montaje, pero debido al carácter experimental del prototipo se ha incluido)

```
23 POSICION: 39.4497375488°N -0.3914370059°O
HORA: 13:22:56
FECHA: 17/5/2018
DISTANCIA ADELANTAMIENTO: 86 cm
Error = 1
```

FIGURA 24: DATOS REGISTRADOS EN SD CON ACUSE DE ERROR

Diagrama de bloques del funcionamiento del dispositivo:

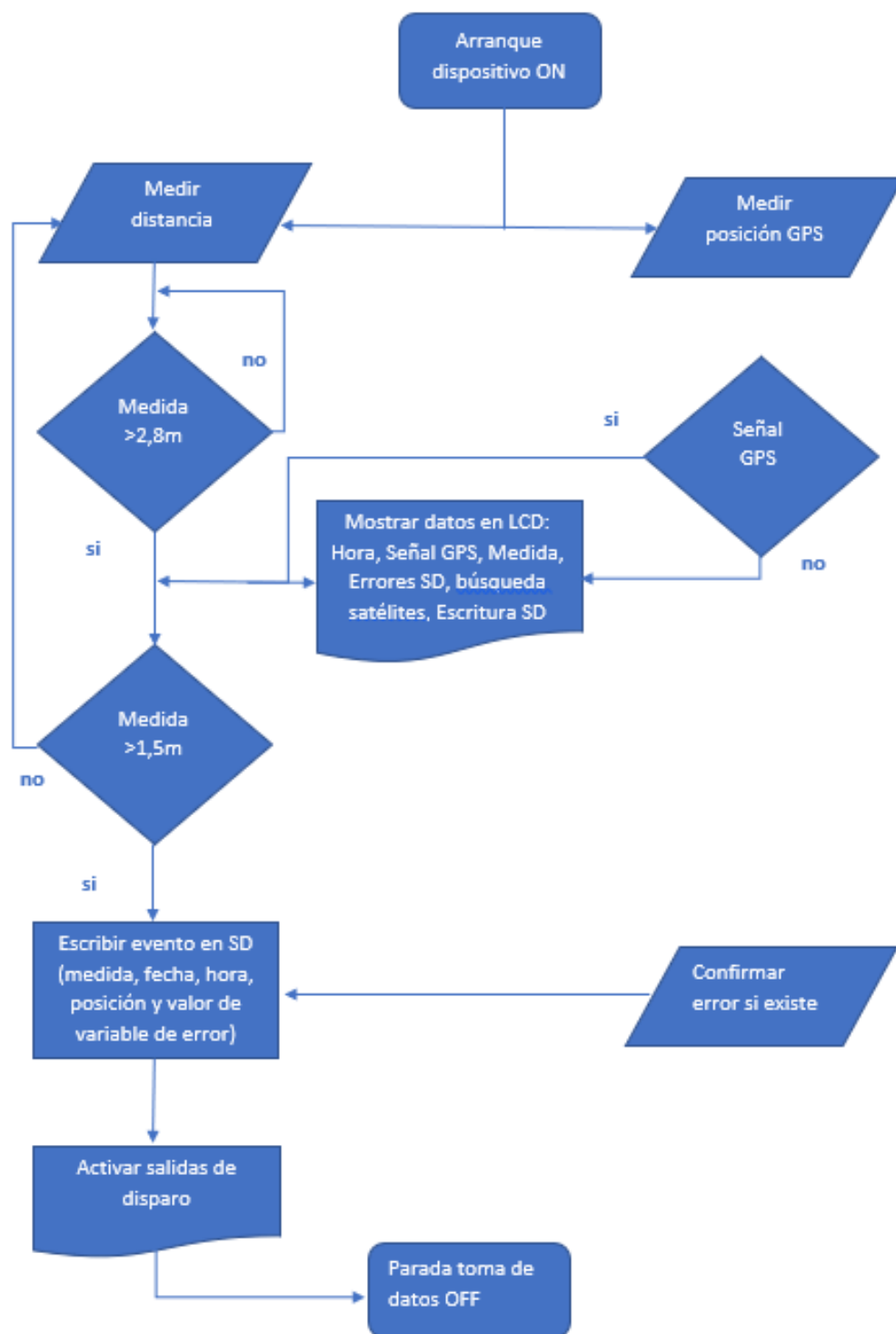


FIGURA 26: DIAGRAMA DE FUNCIONAMIENTO DEL MEDIDOR

Descripción del *sketch*:

1. Cabecera:

- a. Importación de librerías necesarias: *Wire.h* y *LiquidCrystal_I2C* para el control del LCD vía I2C, *TinyGPS* y *SD.h*
- b. Declaración de variables de datos del GPS
- c. Creación del objeto GPS_NEO y del objeto para la creación de fichero *DATALOGGER*
- d. Declaración de constantes para determinar los pines: sensor de ultrasonidos, salidas de activación, ófset de medida y pin CS de comunicación SPI de la tarjeta SD
- e. Creación de variable tipo *volatile* para gestionar la interrupción y variable contador de medidas.
- f. Creación del objeto *lcd* que hará uso de las funciones de la librería *LiquidCrystal_I2C*
- g. Se declara como estático la función *ESPERA_DATOS* y se declara la variable local *ms* tipo *long* de esta función

2. Función *setup*:

- a. Definen pines de entrada salida del módulo ultrasónico y se ponen a 1 y a 0 los pines de alimentación respectivamente.
- b. Se definen los pines de salida de activación como pines de salida.
- c. Se define la interrupción
- d. Se inicializa LCD
- e. Se enciende retroiluminación del LCD
- f. Se realiza la comprobación de la tarjeta SD
- g. Se hace una lectura de datos inicial del GPS llamando a la función *DATOS_GPS*

3. Función *loop*:

- a. Se llama a la función *ping*, que a su vez retornará el valor de la medida a la variable local *cm* creada en la misma línea.
- b. Se limpia la pantalla LCD con la función *clear* perteneciente a la librería *LiquidCrystal_I2C*.
- c. Se condiciona la escritura de la tarjeta SD a que estén disponibles los satélites : en caso de que no estén disponibles el LCD muestra "Sin GPS" y "BS" buscando satélite" en la segunda línea. En caso de estar disponibles se muestra la hora y se permite la escritura en la SD.

- d. Se condiciona la escritura de la medida en el LCD para que se muestre si es inferior a 280 cm
 - i. Se condiciona la escritura de datos a que la medida sea inferior a la distancia de seguridad de 150 cm
 - ii. En caso de ser la medida inferior en la segunda línea se muestra Dist. NOK y si es superior Dist. OK”
 - iii. En este caso se llama a la función DATOS_GPS y ESPERA DATOS.
 - iv. Se condiciona también la escritura de datos a que estén los satélites disponibles.
 - v. Se escriben los datos en el archivo que se crea : “DatosGPS.txt. y se cierra el archivo.
 - vi. Se escribe en la segunda línea “DR (datos recogidos)
- e. Se da un retardo para todo el sketch de 200ms para separar los conteos de medición y escritura que se sumará al ciclo de *loop* del programa y a los posibles retardos de cada función llamada.
- f. Se resetea la variable *Error* por si esta ha sido activada anteriormente por la interrupción.
- g. Se definen las funciones *ping*, ESPERA_DATOS, DATOS_GPS , (explicadas ya en los dos apartados anteriores).
- h. Se define la función de interrupción ESCRIBE_ERROR que lo único que hace es poner la variable tipo *volatile Error* a 1 (*True*)

4.4 MONTAJE FINAL Y PRUEBAS DISPOSITIVO

Finalmente se escoge la placa de desarrollo compatible con *Arduino Mega (Elegoo Mega 2560 R3)*, lo que permitirá cualquier ampliación a nivel de software y hardware en caso de ser necesario y dispone de la capacidad que de la que no se disponía para el manejo de todos los *shields* con Arduino nano (al cargar todo el *sketch* aparecía un error de memoria insuficiente). Por un lado, esta placa dispone de tres puertos serie adicionales, con lo que ya no será necesario el uso de la librería *SoftwareSerial.h*. Por otro lado, la mayor capacidad de la placa permitirá añadir algún sensor laser o iterar las mediciones con tal de disminuir el error en las mediciones.

4.4.1 MONTAJE EN PLACA DE PRUEBAS

Para las primeras pruebas antes de proceder a su montaje final se ha de tener en cuenta que los pines distribución de los pines cambia respecto a los montajes realizados con Arduino Nano o Uno. Además, para comodidad en el montaje y debido a que el consumo de del sensor de ultrasonidos es de unos 15 mA aprovecharemos pines digitales para alimentar el sensor, de modo que se ahorra cableado final y las salidas son capaces de ofrecer hasta 40 mA. La salida 41 se tendrá que poner a 1 (+5v) y la 35 a 0 (0v). El conexionado queda ahora:

- Módulo sensor ultrasonidos:

HC-SR04	Arduino Mega	Color
Vcc	D41	Sin cables (conexión directa)
Trigger	D39	
Echo	D37	
GND	D35	

- Módulo I2C con LCD: no presenta problemas ya que es pin a pin.
- Módulo I2C con Arduino:

Módulo I2C	Arduino Mega	Color
GND	GND	Negro
Vcc	5v	Rojo
SDA	SDA (20)	Amarillo
SCL	SCL (21)	Naranja

- Módulo GPS Neo 6:

Módulo GPS	Arduino Mega	Color
GND	GND	Negro
TX	RX1(19)	Blanco
RX	TX1(18)	Marrón
Vcc	5v	Rojo

- Módulo SD:

Módulo SD	Arduino Mega	Color
GND	GND	Negro
Vcc	5v	Rojo
MISO	MISO (D50)	Verde
MOSI	MOSI (D51)	Azul
SCK	SCK (D52)	Morado
CS	CS(D53)	Gris

Se combina el código de las dos pruebas anteriores de forma que los datos queden registrados y se cumplan las condiciones de la figura 14. Se considerarán mejoras tales como que no se muestre la medida a partir de una distancia donde esta puede no resultar precisa (por ejemplo 2,8 metros). El montaje en la placa de pruebas (*protoboard*) verifica el funcionamiento de todos los componentes y nos permite conocer el consumo final de corriente del dispositivo (a través del amperímetro de la fuente de alimentación de laboratorio).

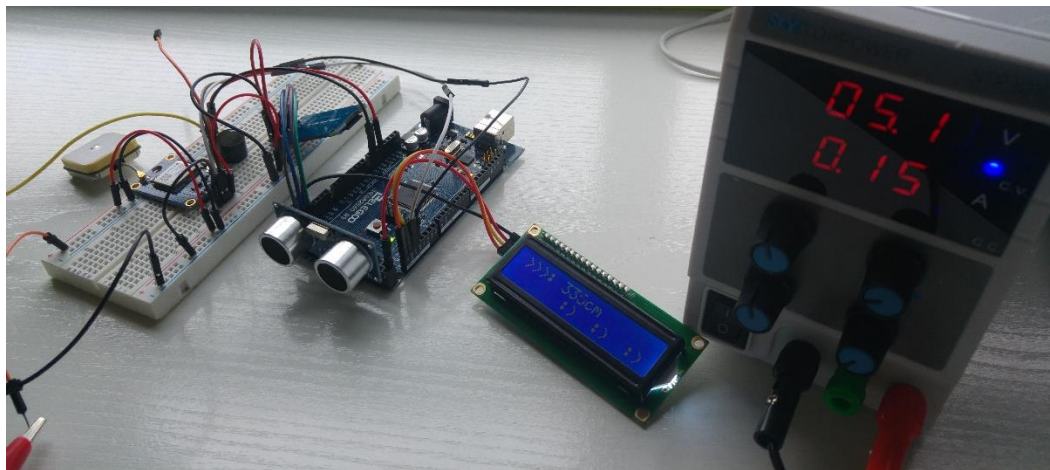


FIGURA 27: PRUEBA MONTAJE FINAL EN PLACA PROTOBOARD

4.4.2 MONTAJE DEFINITIVO

El paso siguiente es alojar el dispositivo en una caja y disponer de los componentes adicionales para el montaje. En este caso el uso haremos uso de una placa prototipo PCB (*Printed Circuit Board*) que nos permitirá alojar y conectar los diferentes módulos a la placa de control. También será necesario el uso de dos pulsadores y un interruptor de encendido. El módulo LCD queda sujeto a la PCB. Uno de los pulsadores hará la función de generar una interrupción para registrar que ha existido un error en la medida. El otro, es el *reset* de la placa.

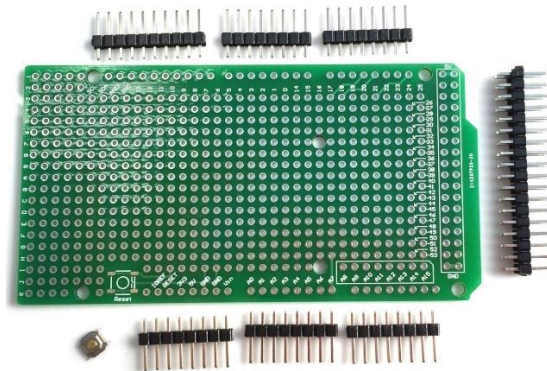


FIGURA 28: PLACA PROTOTIPO PCB PARA ARDUINO MEGA

El conexionado entre los *shields* se realizará para el prototipo con cables (mismo código de colores que el usado anteriormente). Algunas de las conexiones sobre todas las de alimentación se realizarán uniéndose en la misma placa PCB prototipo.

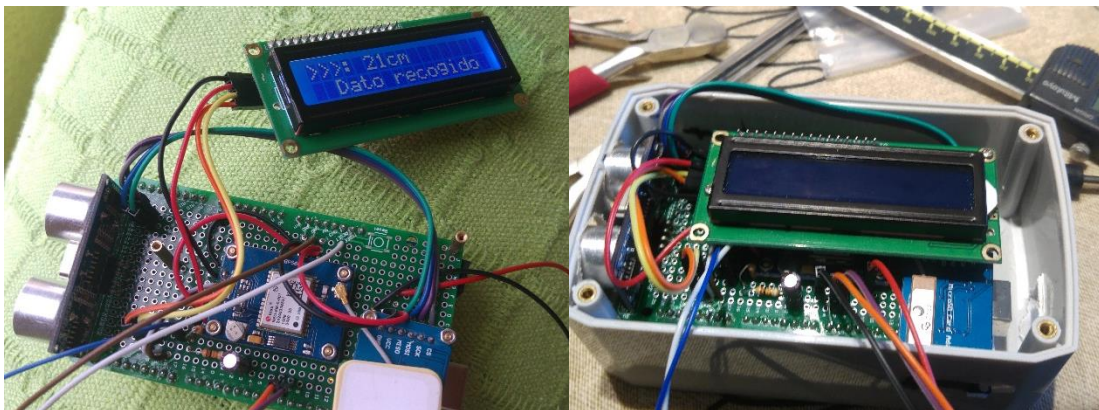


FIGURA 29: PROCESO DE MONTAJE Y PRUEBAS DEL MONTAJE FINAL

Finalmente, no se ubica la placa de relés dentro de la caja, per se habilita una conexión tipo micro o aviación de 5 pines (GX 16-5) para proveer las salidas de activación necesarias en caso hacer uso de ellas, además se aprovecha este conector para alimentar el dispositivo También se practica un orificio para poder acceder al puerto 0 (USB) para las posibles mejoras del dispositivo, así como añadir nuevas funcionalidades por software.

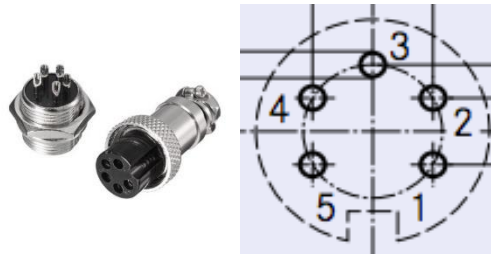


FIGURA 30 CONECTOR MICRO O AVIACIÓN 5 PINES

Conexiones en la conexión macho (caja):

Pin	Función	Color
1	Salida D7	Naranja
2	Salida D8	Morado
3	GND	Negro
4	No conectado	No conectado
5	5v	Rojo

En el conector hembra aérea sólo se cablean los terminales de alimentación, de forma que no se hace uso de las salidas para las pruebas. El cable USB con conector tipo A por tanto, solo utiliza los cables negro (GND) y rojo (+5V):

Pin GX-16	Color y función	Pin USB tipo A
3	Negro (GND)	1
5	Rojo(+5V)	4



FIGURA 31: CONEXIONES USB TIPO A

Para registrar los errores en los adelantamientos se hará uso de una interrupción, que detendrá el *sketch* y ejecutará una rutina para escribir una marca en la tarjeta SD para posteriormente poder anular la medida. Para ello hacemos uso del pin 2 de Arduino Mega (Interrupción 0 de las 5 disponibles), ya que este pin está preparado para hacer uso de estas. Se utilizará el paso de alto a bajo de la entrada (flanco descendente) para activar la interrupción (evento FALLING). La variable que utilizaremos para marcar el error se declara como tipo *volatile* y consiste en pasar a 1 una variable booleana que se encuentra a cero mientras no exista ningún error.

Se usarán tres componentes pasivos dos resistencias de $10\text{K}\Omega$ $1/4\text{W}$ y un condensador electrolítico de $1\ \mu\text{F}$ 16V . La función de R1 es la de hacer de resistencia de pull-up: mantener a nivel alto la salida mientras no activamos el pulsador, ya que de otro modo la salida puede estar en un modo indeterminado (la entrada presenta alta impedancia sin conectar). R2 y C1 se usan para filtrar los rebotes de la señal que se pueden ocasionar tras presionar el pulsador; estos pueden disparar varias veces la interrupción.

Los valores de la resistencia de *pull-up* se tiene que escoger un valor de compromiso que asegure ofrecer el valor lógico de entrada sin ninguna indeterminación, para que el consumo sea el menor posible para evitar sobreconsumos. Se suele utilizar la regla de utilizar un valor 10 veces menor a la impedancia de las entradas. En este caso en valor escogido es de $10\ \text{K}\Omega$ para R1 ya que las entradas pueden llegar a estar en valores de impedancia de $1\text{M}\Omega$ para el caso de usar tecnología CMOS. El valor máximo de potencia que disipará la resistencia será (cuando se accione el pulsador):

$$I = \frac{V}{R} : P = V \cdot I \rightarrow P = \frac{V^2}{R} = \frac{5^2}{10 \cdot 10^3} = 2,5 \cdot 10^{-3}\text{W} = 2,5\ \text{mW}$$

Con los valores escogidos del circuito RC (R2, C1) se tiene un filtro pasa-bajos que eliminará los espurios producidos al cambio de estado de la entrada. La frecuencia de corte de este filtro:

$$f = \frac{1}{2\pi RC} = \frac{1}{2\pi 10^{-2}} = 15,91 \text{ Hz}$$

En el caso de R2 la potencia máxima disipada es la mitad que R1, por lo que los valores de ¼ W son más que suficientes para el montaje.

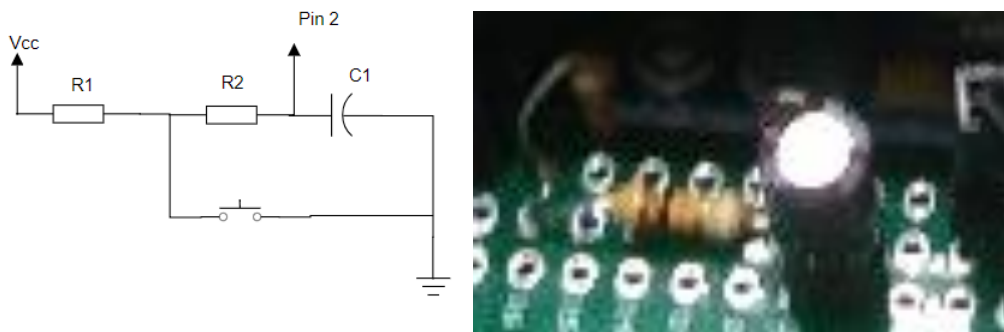


FIGURA 32: RESISTENCIA DE PULL-UP Y FILTRADO DE REBOTES DE SEÑAL DEL PULSADOR

De todas formas, es posible activar una resistencia interna de *pull-up* que tienen asociados los pines E/S las placas de Arduino, al igual que el rebote se puede tratar por software, pero se decide optar por añadir estos componentes discretos para no añadir más complejidad al software, a no ser que se requiera tras las pruebas del dispositivo.

4.4.3 MONTAJE EN LA BICICLETA Y ALIMENTACIÓN

El dispositivo se alimentará de forma externa a través de una batería de ion-litio comercial de tipo *Power Bank*. Este tipo de baterías es muy común para dar soporte de carga a dispositivos móviles y se puede usar cualquier tipo de ellas. La capacidad de la batería usada es de 10000 mA/h con dos salidas USB (5V) capaces de proporcionar hasta 2,1 A. Esto nos dará una vida útil teórica para el dispositivo suficientemente larga. Si suponemos un factor de corrección de 0,7 por ejemplo para prever situaciones que pueden hacer que la batería se descargue más rápido (variaciones de temperatura o humedad, por ejemplo) tenemos:

$$\text{Horas duración estimadas} = \frac{10A/h}{0,15A} \cdot 0,7 = 46,6 \text{ horas}$$



FIGURA 33: BATERÍA TIPO POWER BANK DE ALIMENTACIÓN DEL MEDIDOR

La batería se ubicará en el botellero de la bicicleta en una bolsa portaherramientas. Para fijar el medidor a la bicicleta se utilizan dos extensores de manillar y un soporte de teléfono móvil para bicicletas que se ha adaptado de forma que queda fijo con la caja del dispositivo.



FIGURA 34: ANCLAJE DEL MEDIDOR A LA BICICLETA

Como se ha comentado anteriormente se la distancia que mide el dispositivo es partir del extremo más saliente de la bicicleta. Se ha tomado esta referencia, pese a que la normativa existente no dice nada más que la separación lateral ha de ser de 1,5 metros. Por lo tanto, se ha modificado la fórmula para el cálculo de la distancia para tener en cuenta 4 cm de ófset.



FIGURA 35: DETALLE DEL MONTAJE EN LA BICICLETA DEL DISPOSITIVO FINAL

4.5 PRUEBAS DEL DISPOSITIVO

4.5.1 MEDIDAS EN ESTÁTICO

Se toman medidas en estático mediante un trípode a un metro de altura y frente a una superficie lisa para determinar la exactitud y la precisión del dispositivo. Se ajusta el ófset a 0 por lo que la escala irá de 3 cm (de donde mide el dispositivo sin problemas) hasta 280 cm (aunque se ha incrementado el rango a 300 cm para poder ver mejor la deriva de sensibilidad en el gráfico), que es donde se extraerán los cálculos.

EXACTITUD DEL DISPOSITIVO

La siguiente tabla muestra la diferencia de medida por el dispositivo comparada con la medida tomada por una cinta métrica que usamos como patrón. En todo caso la

apreciación es de un centímetro, que es la máxima resolución del dispositivo medidor y se realizan tres medidas para cada distancia ajustando el dispositivo para que la medida de exacta en saltos de 20 cm:

distancia nº	VALOR MEDIDO (cm)	VALOR REAL (cm)
1	20	20
2	40	40
3	60	61
4	80	81
5	100	102
6	120	122
7	140	143
8	160	163
9	180	183
10	200	204
11	220	224
12	240	245
13	260	266
14	280	287
15	300	308

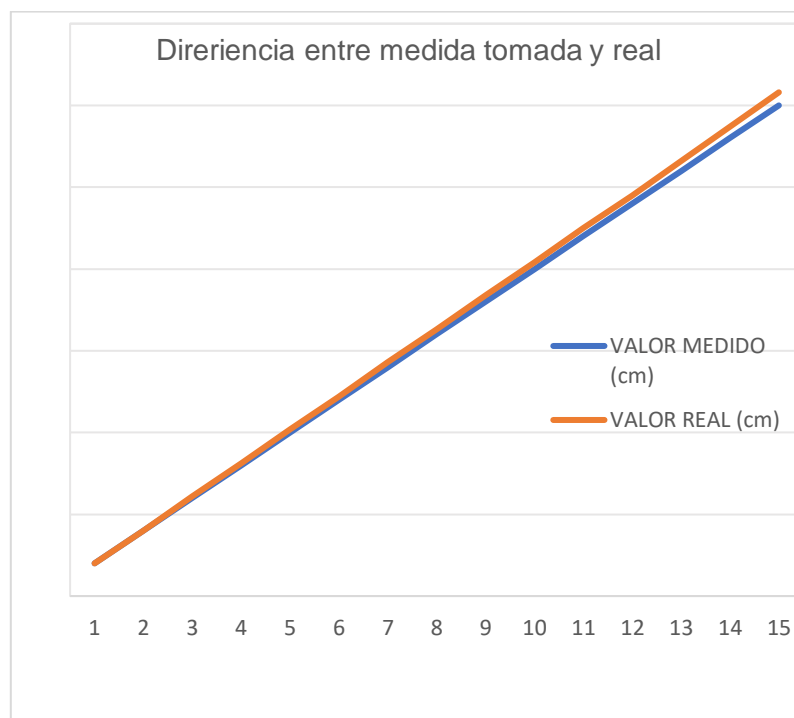


FIGURA 36: DERIVA DE SENSIBILIDAD

Se puede apreciar que el dispositivo presenta una deriva de sensibilidad ya el error se va incrementando a medida que nos vamos a medida que aumentamos la distancia. El máximo error absoluto es de ± 5 cm.

El error para la medida de disparo de seguridad 150 cm es de ± 3 por lo que el error relativo a esta medida que es la que nos interesa será:

$$\text{Error relativo} = \left(\frac{3}{150} \right) \cdot 100 = 2\%$$

El error de linealidad que presenta el dispositivo vendría dado por el porcentaje sobre el fondo de escala y la máxima desviación. Consideramos 280 como fondo de escala, ya que es la 8:

$$\text{Error de linealidad} = \left(\frac{285 - 280}{280} \right) \cdot 100 = 1,79\%$$



FIGURA 37: DETALLE DEL MONTAJE PARA MEDICIÓN EN ESTÁTICO

MEDIDAS DE PRECISIÓN DEL DISPOSITIVO

Para determinar la repetibilidad del dispositivo se toman muestras en varios valores del dispositivo 50, 100, 150, 200, 250 y 300 centímetros sin apreciarse gran dispersión del dispositivo. Para el cálculo de la reproductibilidad se deberían modificar las condiciones

de temperatura y superficie a medir, por ejemplo, pero finalmente no se realizan estas pruebas debido a que no se considera la suficiente precisión para dispositivo por lo que no se realiza un estudio de %R y R (porcentaje de repetibilidad y reproductibilidad) para verificar el proceso de medición.

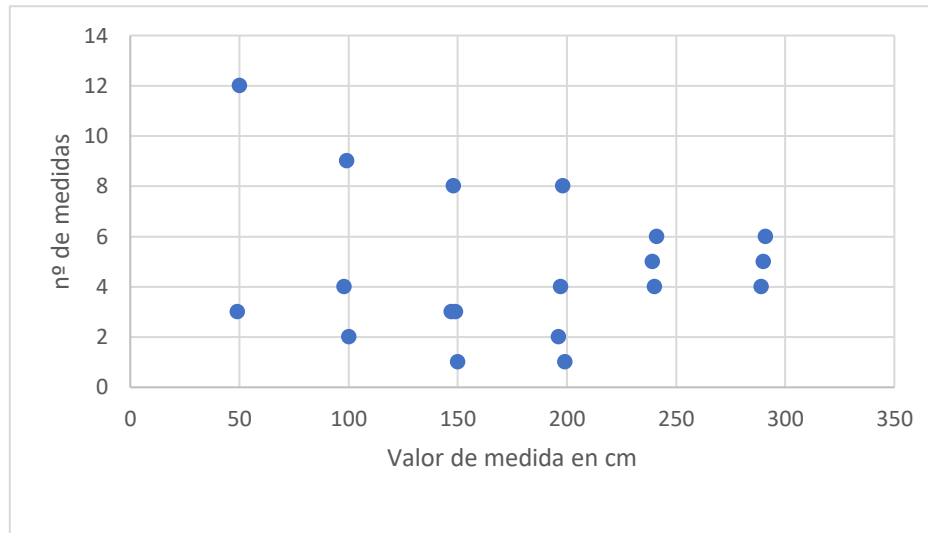


FIGURA 38: DISPERSIÓN DEL DISPOSITIVO EN ESTÁTICO

4.5 1 MEDIDAS EN DINÁMICO

Se corrige el ófset a -18 cm con el fin de tomar la medición desde el centro de la rueda de la bicicleta en dinámico y se establece el máximo rango para que escriba en la SD medidas de hasta 2,8 metros. Se marca una línea en el suelo de medida a 50 cm, 1 metro, 1,5 metros y 2 metros de distancia sobre una señal de tráfico (poste metálico de unos 6 cm de espesor (simulación de adelantamiento de un vehículo)). La detección la realiza sin problemas. No se realizan más medidas por encima de 2 metros, porque a partir de esta medida la dispersión en dinámico es muy alta.

Ahora hay que tener en cuenta que el rango irá de 0 a 280 cm y que el error de deriva se habrá incrementado al desplazar el ófset, aunque se ha tenido en cuenta en los 18 cm para que la medida de disparo sean 150 cm reales, pero habrá que tener en cuenta ese error en las medidas registradas conforme la medida decrezca. Otra solución habría sido encontrar la relación exacta para corregir la expresión en el sketch con lo que las medidas serían mas exactas. Hay que tener en cuenta, además, que vamos a estar afectados por más factores en esta prueba como son la verticalidad durante la marcha

o la exactitud a la hora de circular sobre las rayas marcadas. Los resultados obtenidos para quince muestras en cada medida. Se puede observar que la dispersión para la medida de 2 metros es muy alta, para los demás valores es aceptable:

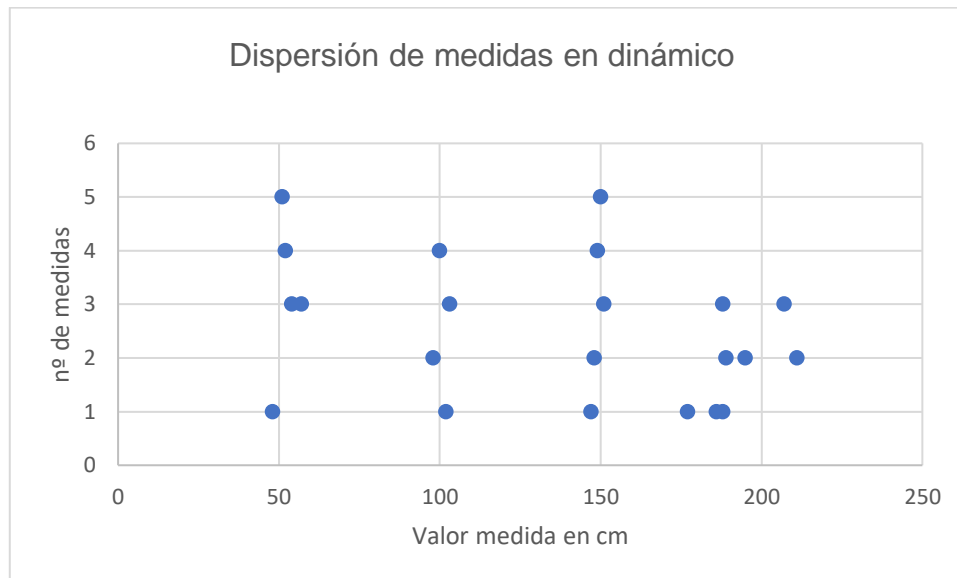


FIGURA 39: DISPERSIÓN DEL DISPOSITIVO EN DINÁMICO

Además, el objeto donde se refleja la señal supera con creces las peores condiciones en las que se pueda encontrar el dispositivo, debido a que estamos hablando de un poste de 6 centímetros de grosor, cuando el vehículo más corto puede tener casi dos metros de longitud para el caso de una motocicleta. La dispersión producida a partir de 2 metros puede deberse a rebotes de la señal en la superficie u otros objetos que producen la modificación del recorrido de la señal al realizarse en diferentes ángulos de inclinación que si afectan a partir de esta medida.

Por último, se vuelve a ajustar el ófset a 4 cm + 2 cm del error acumulado a esa medida y se realiza una salida de 20 Km en carretera para verificar el funcionamiento del dispositivo. Los resultados refuerzan los resultados. Los adelantamientos a menos de un metro y medio son recogidos en su totalidad. Aparecen algunas falsas medidas debido al adelantamiento de ciclistas, pero se marcan correctamente a través del pulsador.

CAPITULO 5: CONCLUSIONES Y LINEAS FUTURAS DE DESARROLLO

La medición por ultrasonidos en este caso no da una fiabilidad tan alta como cabría esperar, aunque la exactitud, precisión y fiabilidad del dispositivo es suficiente como uso didáctico. Sería necesario perfeccionar el dispositivo sustituyendo el sensor por uno también ultrasónico, pero más direccional o que se realizara la medición por láser, Además, se deberían haber hecho un mayor número de pruebas, para poder corregir los errores generados por las características inherentes a la tecnología usada para la medición como por ejemplo la variación de temperatura.

Por otra parte, el dispositivo si podría ser utilizado para dar una estadística de donde se producen más adelantamientos a menos de un metro y medio. Aunque hubiese una cierta tolerancia en la medición sí que se podría tener una información a nivel global si esta información se vuelca en la red. El dispositivo podría ser integrado en los GPS deportivos de las bicicletas como elemento accesorio y finalmente esa información ser compartida por los usuarios de redes sociales tipo *Strava* o como información adicional ofrecida por *Google Maps*.

A nivel del propio dispositivo, una serie de mejoras hubieran sido poder parametrizar los datos de ófset y distancias umbral de disparo e inversión de la pantalla permitiendo que el dispositivo pueda ser configurado para su utilización en diferentes países. También se podría crear un software para tratar los datos recogidos Otra de las mejoras sería que el dispositivo solo realizase la medición y el resto de funciones las hiciese un smartphone, de este modo la información podría ser volcada a la red y obtenerse información en tiempo real por los usuarios. En este caso el dispositivo estaría conectado mediante *Bluetooth*.

CAPITULO 6: PLANIFICACIÓN TEMPORAL

La idea del proyecto surge en el mes de octubre al cursar la asignatura de Aplicaciones y Servicios Multimedia, donde se empieza a gestar el proyecto. La primera idea de la se parte octubre del 2017 es la de poder enviar datos en tiempo real de adelantamientos indebidos a bicicletas, de modo que el diseño principal haría uso de un módulo GPS-GSM-GPRS para enviar los datos de posición. Otra posibilidad que se barajó fue la de dar las funciones de geoposición y almacenamiento de medidas al un dispositivo móvil y que el medidor se comunicará por tecnología *durante el mes de mar*.

Finalmente se opta por que el dispositivo sea totalmente autónomo, de forma que no tenga dependencia de otro dispositivo para almacenar los datos de adelantamiento indebido. Co ello se facilita el desarrollo totalmente en entorno Arduino. El diseño del prototipo final de desarrolla en el mes de abril, una vez se tienen en cuenta las capacidades la placa de desarrollo, que ha de ser sustituida por una con mayores recursos que la inicialmente propuesta; tras las pruebas de los elementos que componen el proyecto por separado se ve la necesidad de mayor capacidad para esta placa.

Durante la tercera semana de mayo se realizan pruebas del dispositivo, con lo que se verifican los resultados durante las dos últimas semanas de mayo. Es necesario realizar algunas modificaciones en el código para que se tenga en cuenta el ófset de montaje en la bicicleta. Los plazos de lo inicialmente previsto variaron a finales de marzo, al cambiarse la idea sobre el prototipo final. Se ha podido terminar y probar el dispositivo a partir de la segunda planificación (principios de abril) en el tiempo estipulado desde entonces.

Según el diagrama de Gantt los días dedicados al proyecto han sido 96, sin tener en cuenta que el proyecto empezó realmente en octubre de 2017 con lo que se tenía algo adelantado el proyecto. En días dedicados hasta el hito entrega 1 (desde el 7 de marzo hasta el 19 de abril) se dedican un número de 93 horas totales repartidas a lo largo del periodo (3 horas diarias de lunes a viernes en 31 días de trabajo). A partir de aquí hasta entrega 2 el número de horas dedicadas casi se duplica. Desde el 19 de abril hasta el día 23 de mayo la cantidad de horas dedicadas es de 160 horas repartidas en 35 días

(donde se dedican 4 horas de lunes a viernes y 8 horas cada uno de los 5 sábados del periodo destinados al diseño, pruebas y montaje del dispositivo).

La revisión de la memoria del proyecto y la revisión de los datos obtenidos en esta prueba se realizan en esta fase con una duración de 16 días hasta la entrega 3 de los cuales se dedican 12 días con una duración de 36 horas; 3 horas diarias de lunes a viernes. En esta fase se empieza preparar la presentación del proyecto. La entrega 4 tiene una duración de 7 días con una dedicación de 12 horas con una dedicación de 2 horas durante 6 días.

	Modo de	Nombre de tarea	Duración	Comienzo	Fin	Predeci
1		Inicio Proyecto	0 días	mié 07/03/18	mié 07/03/18	
2		Entrega 1	82,67 días?	mié 07/03/18	mié 18/04/18	
3		Elección de hardware y tecnologías para el desarrollo	21,33 días?	mié 07/03/18	vie 16/03/18	1
4		Desarrollo del prototipo	24 días?	mar 20/03/18	vie 30/03/18	3
5		Pruebas iniciales	34,67 días?	lun 02/04/18	mié 18/04/18	4
6		Entrega 2	68,33 días	jue 19/04/18	mié 23/05/18	
7		Elección de software de apoyo y plataformas	10,67 días?	lun 23/04/18	jue 26/04/18	5
8		Desarrollo de software y comunicaciones con el dispositivo	42,67 días?	vie 27/04/18	vie 18/05/18	7
9		Montaje definitivo del prototipo	10,67 días?	vie 11/05/18	mié 16/05/18	5
10		Pruebas finales	13,33 días?	jue 17/05/18	mié 23/05/18	9
11		Entrega 3	36,33 días	jue 24/05/18	dom 10/06/18	
12		Revisión y entrega de la memoria	32 días?	jue 24/05/18	vie 08/06/18	10
13		Entrega 4	17,67 días	lun 11/06/18	dom 17/06/18	
14		Revisión código	10,67 días?	lun 11/06/18	jue 14/06/18	8
15		Realización del la presentación	10,67 días?	mar 12/06/18	vie 15/06/18	
16		Fin de proyecto	0 días	dom 17/06/18	dom 17/06/18	

FIGURA 1 1: PLANIFICACIÓN TEMPORAL

Diagrama de GANTT:

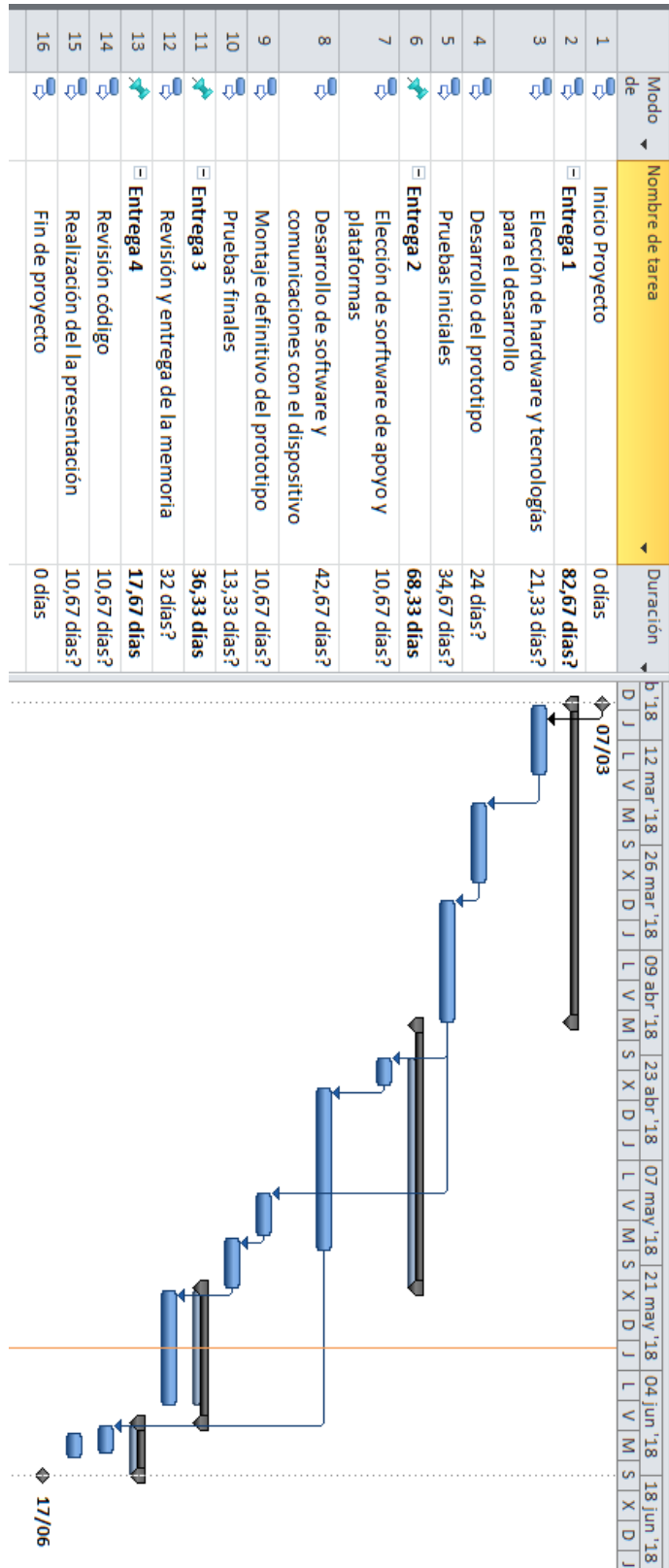


FIGURA 1 2: DIAGRAMA DE GANTT

BIBLIOGRAFÍA

Arduino.cc. Arduino Mega 2560 rev3 [sede Web].Arduino cc 2017 [acceso 24 de diciembre de 2017]. Disponible en: <https://store.arduino.cc/arduino-mega-2560-rev3>

Arduino.cc. Arduino nano [sede Web].Arduino cc 2017 [acceso 24 de diciembre de 2017]. Disponible en: <https://store.arduino.cc/Arduino-nano>

Arduino.cc. Digital pins [sede Web].Arduino cc 2017 [acceso de mayo de 2018]. Disponible en: <https://www.arduino.cc/en/Tutorial/DigitalPins>

Askix.com [sede Web]. Mejorar la precisión del sensor ultrasónico gama.[acceso 27 de abril de 2018]. Disponible en: https://www.askix.com/mejorar-la-precision-del-sensor-ultrasonico-gama_4.html#title

Botscience.net [sede Web]. Módulo GPS para Arduino UBlox NEO 6M [acceso 18 de marzo de 2018]. Disponible en:

http://botscience.net/store/index.php?route=product/product&product_id=73

Codaxus.com [sede Web]. Austin, Texas: Codaxus LLC. [acceso 15 de diciembre de 2017]. Disponible en: <http://codaxus.com/>

Codaxus.com [sede Web]. Austin, Texas. Codaxus LLC. [acceso 15 de diciembre de 2017]. Disponible en: <http://codaxus.com/c3ft/c3ft-v3/>

Compromiso.atresmedia.com. Ponle freno y AXA presentan el primer estudio sobre adelantamientos a ciclistas. Madrid: Atres Media Corporación de medios de comunicación S.A. [sede Web]. Actualizado el 16 de agosto de 2017 [acceso el 15 de diciembre de 2017]. Disponible en:

http://compromiso.atresmedia.com/ponlefreno/centro-estudios/estudios/ciclistas/ponle-freno-presenta-primer-estudio-adelantamientos-ciclistas-espana_201310255835d7510cf244336f124183.html

Conociendogithub.readthedocs.io [sede Web] [sede Wev]. Conociendo Github [acceso 4 de abril de 2018]. Disponible en:

<http://conociendogithub.readthedocs.io/en/latest/data/introduccion/>

Datalogger.pbworks.com [sede Web]. Micro SD Card Micro SDHC Mini TF Card Adapter Reader Module for Arduino [acceso 6 de abril de 2018]. Disponible en:

<http://datalogger.pbworks.com/w/file/attach/89507207/Datalogger%20-%20SD%20Memory%20Reader%20Datasheet.pdf>

Del Pozo Domínguez, Jesús Ángel. Radar para comprobar la distancia de seguridad en adelantamientos a los ciclistas. Ciclismo master [revista en Internet]. 13 de julio de 2015 [acceso 12 de diciembre 2017]. Disponible en:

<http://masters.abloque.com/2015/07/31/radar-para-comprobar-la-distancia-de-seguridad-en-los-adelantamientos-a-ciclistas/>

Elegoo.com [sede Web]. Elegoo MEGA 2560 R3 Board Black ATmega2560 ATMEGA16U2 + USB Cable. Henzhen (China). [acceso 27 de abril de 2018]. Disponible en: <https://www.elegoo.com/product/elegoo-mega-2560-r3-board-atmega2560-atmega16u2-usb-cable/>

Gpsinformation.org [sede Web]. NMEA Data. [acceso 14 de abril de 2018] Disponible en: <http://www.gpsinformation.org/dale/nmea.htm>

Jordi Rovira Jofre. Geotelemática, posicionamiento y navegación. 1ª edición 2012, Material docente de la Universitat Oberta de Catalunya (realización editorial Eureca Media S.L.).

Jiménez Gálvez, José María. Los 400 ciclistas que dejaron de pedalear. Elpais.com [diario en Internet]. 9 de mayo de 2017 [acceso 7 de diciembre de 2017]. Disponible en: https://politica.elpais.com/politica/2017/05/08/actualidad/1494241879_927343.html

Leonsonidovirtual.com.ar [sede Web] El sonido, factores que influyen; temperatura y humedad. Buenos Aires (Argentina). [acceso 29 de abril de 2018]. Disponible en: <http://www.leonsonidovirtual.com.ar/el-sonido-factores-que-influyen-temperatura-y-humedad/>

Llamas Binaburo, Luis. Instalar IDE de Arduino y otro software útil. Luisllamas.com [sede Web]. 13 de octubre de 2013 [acceso 25 de marzo de 2018]. Disponible en: <https://www.luisllamas.es/instalar-ide-arduino/>

Llamas Binaburo, Luis. Arduinos baratos gracias al chip CH340G. Luisllamas.com [sede Web]. 24 de abril de 2015 [acceso 26 de abril de 2018]. Disponible en: <https://www.luisllamas.es/arduinios-baratos-gracias-al-chip-ch340g/>

Llamas Binaburo, Luis. Medir distancia con Arduino y sensor de ultrasonidos HC SR-04. Luisllamas.com [sede Web]. 16 de junio de 2015 [acceso 10 de marzo de 2018]. Disponible en: <https://www.luisllamas.es/medir-distancia-con-arduino-y-sensor-de-ultrasonidos-hc-sr04/>

Llamas Binaburo, Luis. Conectar un display LCD Hitachi a Arduino por bus I2C. Luisllamas.com [sede Web]. 22 de mayo de 2016 [acceso 21 de diciembre de 2017]. Disponible en: <https://www.luisllamas.es/arduino-lcd-i2c/>

Llamas Binaburo, Luis. Leer y escribir en una tarjeta SD con Arduino. Luisllamas.com [sede Web]. 16 de octubre de 2016 [acceso 13 de abril de 2018]. Disponible en: <https://www.luisllamas.es/tarjeta-micro-sd-arduino/>

Llamas Binaburo, Luis. Localización GPS con Arduino y los módulos neo 6. Luisllamas.com [sede Web]. 27 de septiembre de 2017 [acceso 28 de febrero de 2018]. Disponible en: <https://www.luisllamas.es/localizacion-gps-con-arduino-y-los-modulos-gps-neo-6/>

Mueller, Scott. Manual de Actualización y reparación de PCs. Almacenamiento de alta capacidad. "Tarjeta flash y película digital". 12º edición. Prentice Hall. Pearson educación. México 2001. p 722-724.

R. Blanco Patricia. Uno de cada cinco conductores adelanta mal a los ciclistas. Elpais.com [diario en Internet]. 25 de octubre de 2013 [acceso 10 de febrero de 2018]. Disponible en: https://politica.elpais.com/politica/2013/10/25/actualidad/1382711846_123161.html

Reyes Cortés, Fernando; Cid Monjarez, Jaime. Arduino: Aplicaciones en Robótica, Mecatrónica e Ingenierías. 1ª edición Alfaomega Group Editor, S. A.. México. Marcombo S.A. Enero de 2015

Libroweb.alfaomega.com.mx [sede Web parte online del libro] Reyes Cortés, Fernando; Cid Monjarez, Jaime. Arduino: Aplicaciones en Robótica, Mecatrónica e Ingenierías. 1ª edición Alfaomega Group Editor, S. A.. México. Marcombo S.A. Capítulo 3: Plataforma electrónica. Enero de 2015. Disponible en: <http://libroweb.alfaomega.com.mx/book/890/free>

Libroweb.alfaomega.com.mx [sede Web parte online del libro] Reyes Cortés, Fernando; Cid Monjarez, Jaime. Arduino: Aplicaciones en Robótica, Mecatrónica e Ingenierías. 1ª edición Alfaomega Group Editor, S. A.. México. Marcombo S.A. Capítulo 4 : Language C. Enero de 2015. Disponible en: <http://libroweb.alfaomega.com.mx/book/890/free>

Libroweb.alfaomega.com.mx [sede Web parte online del libro] Reyes Cortés, Fernando; Cid Monjarez, Jaime. Arduino: Aplicaciones en Robótica, Mecatrónica e Ingenierías. 1ª edición Alfaomega Group Editor, S. A.. México. Marcombo S.A. Capítulo 6: Librerías y funciones Arduino. Enero de 2015. Disponible en: <http://libroweb.alfaomega.com.mx/book/890/free>.

Libroweb.alfaomega.com.mx [sede Web parte online del libro] Reyes Cortés, Fernando; Cid Monjarez, Jaime. Arduino: Aplicaciones en Robótica, Mecatrónica e Ingenierías. 1ª edición Alfaomega Group Editor, S. A.. México. Marcombo S.A. Capítulo 12: Manejo de interrupciones. Enero de 2015. Disponible en: <http://libroweb.alfaomega.com.mx/book/890/free>

Prometec.net [sede Web] La familia de los chips CH340G. Bilbao (España). [acceso 18 de marzo de 2018]- Disponible en: <https://www.prometec.net/ch340g/>

Seguridad-vial.net [sede Web]. El Sistema ADAS ayuda a prevenir accidentes de tráfico. [acceso 15 de mayo de 2018]. Disponible en: <https://www.seguridad-vial.net/vehiculo/seguridad-pasiva/156-el-sistema-adas-ayuda-a-prevenir-accidentes-de-trafico-a-los-conductores>

U-box.com. NEO-6_DataSheet_(GPS.G6-HW-09005).pdf. U-blox AG Zuercherstrasse 68 CH-8800 Thalwil Switzerland [sede Web]. Acceso 25 de marzo de 2018. Disponible en:[https://www.u-blox.com/sites/default/files/products/documents/NEO-6_DataSheet_\(GPS.G6-HW-09005\).pdf](https://www.u-blox.com/sites/default/files/products/documents/NEO-6_DataSheet_(GPS.G6-HW-09005).pdf)

Wiki.robotica.webs.upv.es. Sensor de ultrasonidos. Universidad Politécnica de Valencia [sede Web]. Acceso el 18 de febrer de 2018. Disponible en:<http://wiki.robotica.webs.upv.es/wiki-de-robotica/sensores/sensores-proximidad/sensor-de-ultrasonidos/>

ANEXOS

1.CÓDIGO PRUEBA MÓDULOS LCD I2C Y MEDIDOR ULTRASONIDOS (apartado 4.2.1)

Código para la prueba del medidor de distancia (Apartado 4.3.1)

```
#include <Wire.h>

#include <LiquidCrystal_I2C.h>

/*se definen pines de eco y trigger para el sensor ultrasónico*/
const int ECHO = 5;
const int TRIGGER = 6;

/*Se crea el objeto lcd dirección 0x3F y 16 columnas x 2 filas */
LiquidCrystal_I2C lcd(0x3F,16,2);

void setup() {

    lcd.begin();// se inicializa lcd

    lcd.backlight();//se enciende luz de fondo

    // se definen las entradas y salidas del módulo de ultrasonidos
    pinMode(TRIGGER, OUTPUT);

    pinMode(ECHO, INPUT);
}
```

```
void loop() {  
  
    int cm = ping(TRIGGER, ECHO);  
  
    lcd.clear();//limpiamos pantalla para no sobrescribir caracteres  
  
    lcd.print(">>>: ");  
  
    // lcd.setCursor (7,1);  
  
    lcd.print (cm);  
  
    lcd.print ("cm");  
  
    lcd.setCursor(2,1); //nos situamos en la segunda linea del lcd  
  
    //condiciones de sobrepasada medida de seguridad  
  
    if ( cm <150)  
    {  
        lcd.print ("Peligro -1,5m!!!");  
    }  
  
    else  
    {  
        lcd.print ("Distancia respetada ");  
    }  
  
    delay(1000);  
}  
  
int ping(int TRIGGER, int ECHO) {  
  
    long duration, distanceCm;  
  
    digitalWrite(TRIGGER, LOW); //ponemos a LOW 4us para crear pulso  
limpio  
  
    delayMicroseconds(4);  
  
    digitalWrite(TRIGGER, HIGH); //generamos Trigger (disparo) de 10us
```

```
    delayMicroseconds(10);  
    digitalWrite(TRIGGER, LOW);  
  
    duration = pulseIn(ECHO, HIGH); //medimos el tiempo entre pulsos,  
en microsegundos  
  
    distanceCm = duration * 10 / 292 / 2; //convertimos a distancia, en  
cm  
    return distanceCm;  
}
```

2.CÓDIGO PRUEBA MÓDULOS GPS Y MICRO SD (apartado 4.2.2)

```
/*LIBRERIAS*/  
  
#include <SoftwareSerial.h>  
  
#include <TinyGPS.h>  
  
#include <SD.h>  
  
  
int CS = 9; /*Este pin es el que corresponde a SS(Slave Selected) del  
micro del Arduino UNO y que se encarga  
  
de decirle al Arduino que la SD esta conectada al BUS SPI que es el  
encargado de establecer la comunicacion.*/  
  
float LATITUD, LONGITUD;  
  
unsigned long EDAD;  
  
int YEAR;  
  
byte MES, DIA, HORA, MINUTO, SEGUNDO, SIGLO;  
  
  
TinyGPS GPS_NEO;  
  
SoftwareSerial Puerto_GPS(4, 3);  
  
File DATALOGGER; //Nombre con el que bautizamos el archivo a crear en la  
SD.  
  
  
static void ESPERA_DATOS(unsigned long ms);  
  
  
void setup()  
{  
  
  Serial.begin(115200); //Abir puerto serie USB  
  
  Puerto_GPS.begin(9600); //Abrir Puerto GPS  
  
  pinMode(9, OUTPUT); //Chip Selected
```



```
/*COMPROBACIÓN SD*/  
Serial.println("Iniciando chequeo estado SD...");  
  
if (!SD.begin(9)) {  
  
    Serial.println("Fallo SD");  
  
    return;//De haber problemas con la SD, con "return" impedimos que  
continúe leyendo código.  
  
}  
  
Serial.println("SD OK!");//Si no hay problema con la SD.  
  
/*LECURA DATOS GPS DE ARRANQUE*/  
DATOS_GPS();  
ESPERA_DATOS(1000);  
}  
  
void loop()  
{  
  
    DATOS_GPS();  
  
    ESPERA_DATOS(1000);  
  
    DATALOGGER = SD.open("Prueba.txt", FILE_WRITE); //Creamos el Archivo  
"Prueba.txt"  
  
    if (DATALOGGER) { //Sentencia en la que escribimos en "Prueba.txt"  
  
        DATALOGGER.print("POSICION: "); DATALOGGER.print(LATITUD, DEC);  
DATALOGGER.print("°N "); DATALOGGER.print(LONGITUD, DEC);  
DATALOGGER.println("°O");  
  
    }  
  
}
```

```

    DATALOGGER.print("HORA:  "); DATALOGGER.print(HORA + 2, DEC);
DATALOGGER.print(":");          DATALOGGER.print(MINUTO,      DEC);
DATALOGGER.print(":"); DATALOGGER.println(SEGUNDO, DEC);

    DATALOGGER.print("FECHA:  "); DATALOGGER.print(DIA,      DEC);
DATALOGGER.print("/");          DATALOGGER.print(MES,        DEC);
DATALOGGER.print("/"); DATALOGGER.println(YEAR, DEC);

    DATALOGGER.close();//Cerramos el archivo "Prueba.txt"

} else {

    Serial.println("Escritura Erronea en SD!!");//Si fallar la creacion
del archivo y su escritura.

}

/*MOSTRAR INFORMACIÓN*/

Serial.print("POSICION:  "); Serial.print(LATITUD,      DEC);
Serial.print("°N "); Serial.print(LONGITUD, DEC); Serial.println("°O");

Serial.print("HORA:  "); Serial.print(HORA + 2,      DEC);
Serial.print(":"); Serial.print(MINUTO,      DEC); Serial.print(":");
Serial.println(SEGUNDO, DEC);

Serial.print("FECHA: "); Serial.print(DIA, DEC); Serial.print("/");
Serial.print(MES, DEC); Serial.print("/"); Serial.println(YEAR, DEC);
}

////////////////////////////////////
// FUNCION ESPERA DATOS GPS //
////////////////////////////////////

static void ESPERA_DATOS(unsigned long MS)
{
    unsigned long INSTANTE_INICIAL = millis();

do

```

```
{  
    while (Puerto_GPS.available())  
        GPS_NEO.encode(Puerto_GPS.read());  
} while (millis() - INSTANTE_INICIAL < MS);  
}  
  
////////////////////////////////////  
// FUNCION LECTURA DATOS GPS //  
////////////////////////////////////  
void DATOS_GPS( void ) {  
    /*OBTENER POSICION GPS*/  
    GPS_NEO.f_get_position(&LATITUD, &LONGITUD, &EDAD);  
    /*OBTENER FECHA/HORA*/  
    GPS_NEO.crack_datetime(&YEAR, &MES, &DIA, &HORA, &MINUTO, &SEGUNDO,  
&SIGLO, &EDAD);  
}
```

3. CÓDIGO DISPOSITIVO DE CONTROL DISTANCIA DE ADELANTAMIENTO (apartado 4.3)

```

include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <TinyGPS.h>
#include <SD.h>

int CS = 53; /*Este pin es el que corresponde a SS(Slave Selected) del
micro del Arduino Mega.*/

/*Declaracion variables GPS*/
float LATITUD, LONGITUD;
unsigned long EDAD;
int YEAR;
byte MES, DIA, HORA, MINUTO, SEGUNDO, SIGLO;

TinyGPS GPS_NEO;
File DATALOGGER; //Nombre con el que bautizamos el archivo a crear en
la SD.
/*Pines del sensor de ultrasonidos*/
const int Vcc = 41; // pin de Vcc para el sensor
const int GND = 35; // pin de GND para el sensor
const int ECHO = 37; //pin eco para medidor ultrasonidos
const int TRIGGER = 39; //pin trigger para medidor de ultrasonidos
/*Diferencia de medida en el montaje con la parte más saliente de la
bicicleta en cm*/
const int offset = 4;
/*Pin interrupción Y Variable error*/
const int intPin = 2;
volatile bool Error = false;

/*Salidas activación dispositivos externos*/
const int outPin1 = 7;

```

```

const int outPin2 = 8;
/*Contador de medidas por debajo de 1'5 metros*/
int Contador_Medidas_Infraccion;

/*Crear el objeto lcd dirección 0x3F y 16 columnas x 2 filas */
LiquidCrystal_I2C lcd(0x3F,16,2);

static void ESPERA_DATOS(unsigned long ms);

void setup() {
  pinMode (Vcc,OUTPUT); //Pone a Vcc del SR4 como salida
  digitalWrite(Vcc,HIGH); //Pone a Vcc a +5v
  pinMode (GND,OUTPUT); // Set the gnd pin as an output
  digitalWrite(GND, LOW); //Pone GND a 0V
  lcd.begin(); // se inicializa lcd
  lcd.backlight(); //se enciende luz de fondo
  // se definen las entradas y salidas del modulo de ultrasonidos
  pinMode(TRIGGER, OUTPUT); //se define como salida el disparo de la
  señal
  pinMode(ECHO, INPUT); // se define como entrada la recepcion del eco
  Serial1.begin(9600); //Abrir Puerto GPS
  pinMode(CS, OUTPUT); //Chip Selected en tarjeta SD
  pinMode(intPin, INPUT); // define entrada interrupción
  pinMode(outPin1, OUTPUT); //define salida física 1
  pinMode(outPin2, OUTPUT); // define salida física 2
  attachInterrupt(digitalPinToInterrupt(intPin), ESCRIBE_ERROR,
  FALLING);
  /*COMPROBACIÓN SD*/
  //Serial.println("Iniciando chequeo estado SD...");

  if (!SD.begin(CS) ){
    Serial.println("Fallo SD");
    return; //De haber problemas con la SD, con "return" impedimos que
    continue leyendo codigo.
  }
}

```

```
}  
Serial.println("SD OK!"); //Si no hay problema con la SD.  
  
/*LECURA DATOS GPS DE ARRANQUE*/  
DATOS_GPS();  
}  
void loop()  
{  
  int cm = ping(TRIGGER, ECHO);  
  lcd.clear(); //limpiamos pantalla para no sobrescribir caracteres  
  /*Indica la hora en la parte superior derecha si ha señal de GPS*/  
  if (EDAD == TinyGPS::GPS_INVALID_AGE) {  
    lcd.print ("Sin GPS");  
  }  
  else { lcd.print(HORA + 2, DEC); lcd.print(":"); lcd.print(MINUTO,  
DEC);  
  }  
  /*Muestra la distancia si es menor del rango de medición  
funcional*/  
  lcd.print( " DT:");  
  if (cm < 280)  
  {  
    lcd.print (cm);  
    lcd.print("cm");  
  }  
  lcd.setCursor(0,1); //nos situamos en la segunda fila del lcd  
(columna 0)  
  //condiciones de sobrepasada medida de seguridad y hay datos del GPS  
  if (cm < 150)  
  {  
    lcd.print("Dist. NOK");  
  
    DATOS_GPS();  
    ESPERA_DATOS(500);  
  }  
}
```

```

/*Es condición tener el señal de GPS para poder grabar la SD*/
if (EDAD == TinyGPS::GPS_INVALID_AGE)
{
    lcd.setCursor(14,1); lcd.print ("BS");
}
else
{
    Contador_Medidas_Infraccion++;
    DATALOGGER = SD.open("DatosGPS.txt", FILE_WRITE); //Creamos
el Archivo "DatosGPS.txt"
    if (DATALOGGER) { //Sentencia en la que escribimos en
"DatosGPS.txt"
        DATALOGGER.print(Contador_Medidas_Infraccion);
        DATALOGGER.print(" POSICION: "); DATALOGGER.print(LATITUD,
DEC); DATALOGGER.print("°N "); DATALOGGER.print(LONGITUD, DEC);
DATALOGGER.println("°O");
        DATALOGGER.print(" HORA: "); DATALOGGER.print(HORA + 2,
DEC); DATALOGGER.print(":"); DATALOGGER.print(MINUTO, DEC);
DATALOGGER.print(":"); DATALOGGER.println(SEGUNDO, DEC);
        DATALOGGER.print(" FECHA: "); DATALOGGER.print(DIA, DEC);
DATALOGGER.print("/"); DATALOGGER.print(MES, DEC);
DATALOGGER.print("/"); DATALOGGER.println(YEAR, DEC);
        DATALOGGER.print(" DISTANCIA ADELANTAMIENTO: ");
DATALOGGER.print(cm, DEC); DATALOGGER.println("
cm");DATALOGGER.print(" Error =
");DATALOGGER.println(Error);DATALOGGER.println(" ");
        DATALOGGER.close();//Cerramos el archivo "DatosGPS.txt"
        lcd.setCursor(14,1);
        lcd.print("DR");//Indica que el dato se ha recogido por la
SD

    } else {

        lcd.setCursor(0,1);
        lcd.print("Error en SD");
        delay(1000);
    }
}

```

```
}
  }
else
  {
    lcd.print (" Dist. OK");
  }
  delay(200);
  Error = false; //se resetea la variable de error en caso de haberse
hecho uso de la interrupción
}

////////////////////////////////////
// FUNCION ping MEDIDA DISTANCIA //
////////////////////////////////////

int ping(int TRIGGER, int ECHO) {
  long duration, distanceCm;

  digitalWrite(TRIGGER, LOW); //para generar un pulso limpio ponemos
a LOW 4us
  delayMicroseconds(4);
  digitalWrite(TRIGGER, HIGH); //generamos Trigger (disparo) de 10us
  delayMicroseconds(10);
  digitalWrite(TRIGGER, LOW);
  duration = pulseIn(ECHO, HIGH); //medimos el tiempo entre pulsos,
en microsegundos

  distanceCm = (duration * 10 / 292 / 2) - offset; //convertimos a
distancia, en cm y restamos el offset de montaje
  return distanceCm;
}
```



```

////////////////////////////////////
// FUNCION ESPERA DATOS GPS //
////////////////////////////////////
static void ESPERA_DATOS(unsigned long MS)
{
  unsigned long INSTANTE_INICIAL = millis();
  do
  {
    while (Serial1.available())
      GPS_NEO.encode(Serial1.read());
  } while (millis() - INSTANTE_INICIAL < MS);
}
////////////////////////////////////
// FUNCION LECTURA DATOS GPS //
////////////////////////////////////
void DATOS_GPS( void ) {
  /*OBTENER POSICION GPS*/
  GPS_NEO.f_get_position(&LATITUD, &LONGITUD, &EDAD);
  /*OBTENER FECHA/HORA*/
  GPS_NEO.crack_datetime(&YEAR, &MES, &DIA, &HORA, &MINUTO, &SEGUNDO,
&SIGLO, &EDAD);
}
////////////////////////////////////
// FUNCION ISR DETECCION MEDIDA ERRONEA //
////////////////////////////////////
void ESCRIBE_ERROR( void ) {

  Error = true;

}

```

COSTE DE MATERIAL DEL PROTOTIPO

Descripción	Cantidad	Precio
Placa Elegoo Mega R3	1	11,99
Placa PCB d expansión Arduino Mega	1	6,99
Módulo GPS Neo 6V	1	14,99
Módulo tarjeta micro SD	1	3,60
Módulo ultrasonidos HC-SR04	1	3,07
Módulo LCD + I2C 16 x 2 caracteres	1	7,19
Caja 124 x 75 x 70 con tapa transparente	1	8,85
Amarre para la caja a los soportes (soporte para móvil)	1	8,55
Barras de extensión del manillar	2	25,98
1 interruptor y 2 pulsadores	3	3,50
Batería Power Bank 10.000 mA	1	12,99
Cable, estaño, termoretráctil, tornillos, tuercas y separadores	1	15,45
Cable USB con conector tipo A 1,5 metros	1	2,25
Conectores aviación GX16 5 pines (macho de superficie)	1	1,75
Conector aviación GX16 5 pines (hembra aéreo)	1	1,80
COSTE TOTAL DE MATERIALES (en euros)		128,95

