PROYECTO FIN DE CARRERA:

SISTEMA DE GEOLOCALIZACIÓN SGPS EN ARDUINO

Autor: Teresa Fernández Marina
Tutor: Luis Enrique Moreno Lorente
Director de proyecto: Javier Victorio Gómez González

INDICE

1. INTRODUCCIÓN

El presente trabajo describe las tareas realizadas como parte del Proyecto Fin de Carrera de Ingeniería Técnica Industrial especialidad en Electrónica Industrial de la Universidad Carlos III de Madrid.

Actualmente los sistemas de posicionamiento y localización geográfica deben su funcionamiento a sistemas meramente antropogénicos. La tecnología GPS que actualmente se utiliza [1], fácilmente accesible para todo el mundo gracias a los smartphones, determinan la posición a partir de las coordenadas que reciben de los satélites enviados por el ser humano a la órbita terrestre.

Estos sistemas sin dejar de ser extremadamente útiles y precisos, podrían dejar de ser funcionales en determinadas situaciones como tormentas solares[1], en cuyo caso se necesitaría un método alternativo para la obtención de coordenadas. Esto lleva a pensar que la referencia astronómica de la que se ha hecho uso durante siglos, el sol, podría ser una buena solución. En efecto, utilizando el sol como principio básico de funcionamiento y añadiendo algoritmos y sistemas electrónicos propios de nuestra era tecnológica, se obtiene una alternativa viable a la tecnología actualmente usada, GPS.

El posicionamiento global basado en intensidad de luz solar ha sido estudiado durante años [2], obteniéndose exitosos resultados en determinadas aplicaciones [3]. Así pues, debido a las posibles limitaciones del sistema GPS y a los buenos resultados obtenidos en estudios anteriores, surge el proyecto de plataforma abierta que se basa en el posicionamiento global basado en la intensidad de luz solar, *"Solar Global Positioning System Project Community"* o *"SGPS Project Community*[2] *"*, creado por Javier V. Gómez.

En esta plataforma están incluidos todos los trabajos previos relativos a esta tecnología, capaces de proveer las coordenadas terrestres de un objeto estático con solo la entrada de la fecha[4][5]. En dichos trabajos se proporciona una formulación simple para la obtención de coordenadas terrestres y se incluye un análisis profundo de ventajas y desventajas contra el sistema de posicionamiento global basado en satélites, GPS.

Este Proyecto Fin de Carrera es una aportación más a esta plataforma. Combina los principios de la ya mencionada tecnología SGPS con la tecnología mote y el entorno open source que ofrece la placa Arduino. Éste requiere un modesto hardware y un software simple. El SGPS basado en Arduino es un sistema de bajo coste con un prototipo alrededor de 50 euros.

En las siguientes secciones se describen las motivaciones y objetivos del presente estudio, así como la estructura general del texto.

[1] *http://www.gps.gov/news/2013/06/2013-06-NRE-public-summary.pdf (Sep,2013)*
[2] *http://sgpsproject.sourceforge.net/wiki*

1.1 Motivación

Como ya se ha comentado anteriormente, la tecnología GPS es un método de localización geográfica preciso y altamente fiable, sin embargo, tiene algunas limitaciones las cuales hacen pensar que sería recomendable investigar y trabajar en alternativas a esta tecnología, una de ellas es la que en este trabajo se presenta.

Entre estas limitaciones, las dos más importantes son, que toda la infraestructura de la que esta tecnología hace uso, los satélites, depende de los gobiernos, de modo que no existe ninguna garantía de que éstos sean siempre accesibles públicamente, y en segundo lugar, es que se ha demostrado que los satélites son susceptibles a las tormentas solares, como ya comunicó el Gobierno de Estados Unidos en Marzo del 2012[3].

Algunos otros inconvenientes cuando se habla de tecnología GPS, son que los satélites dependen de un consumo de energía, lo que hace necesario el cambio de baterías cada cierto tiempo, con todo lo que ello supone, dinero, recursos, riesgo, etc. Además, la infraestructura de satélite tienen una vida útil que está llegando a su fin, lo que conllevaría a renovar toda la infraestructura.

Debido a todas estas limitaciones e inconvenientes, resulta razonable que exista una potente motivación para profundizar en el desarrollo de una nueva tecnología que lleva años siendo estudiada, la tecnología de posicionamiento global basada en intensidad de luz solar, ya que éste es un sistema autónomo que no depende de ninguna infraestructura. Además, este proyecto de SGPS se ha desarrollado en una plataforma open source, lo que significa que cualquiera está autorizado a aportar ideas así como a consultar todos los trabajos y avances realizados, esto fomenta el desarrollo y el rápido crecimiento de la iniciativa.

Otra motivación para seguir profundizando en la tecnología SGPS, son los buenos resultados que se están obteniendo hasta ahora en anteriores trabajos [5], el cual proponía una implementación hardware y software sencilla y de bajo coste para el problema existente en la monitorización de toma de datos en los océanos, con resultados muy precisos, demostrando que la tecnología SGPS es un campo de investigación abierto y muy prometedor.

1.2 Objetivos

A continuación se muestran una lista con los principales objetivos de este trabajo.

- Construir un dispositivo basándose en la tecnología SGPS que sea capaz de proporcionar las coordenadas, latitud y longitud, de un objeto exterior con tan solo la introducción de la fecha.
- Construir el dispositivo SGPS con el menor presupuesto posible sin que afecte a su precisión y capacidad.
- Hacer la implementación hardware y software en entorno Arduino y su posterior puesta en marcha.
- Hacer que el sistema sea autónomo el cuál no consuma casi batería, de modo que no sea necesario un mantenimiento por cambio de batería en años.

[3] *http://www.gps.gov/news/2012/03/solarstorm/ (Sep, 2013)*

- Demostrar mediante los resultados obtenidos que esta alternativa de investigación es prometedora y tiene un gran potencial debido a su gran precisión en proporción con su coste.

1.3 Resumen

Este Proyecto Fin de Carrera describe el fundamento teórico y esquema de funcionamiento del Sistema de Posicionamiento Global de tecnología Solar, SGPS, adaptado a entorno Arduino. La característica principal y que diferencia este dispositivo de un GPS común es la utilización del Sol como elemento de referencia para el cálculo de la posición.

La localización es un problema cambiante por muchas razones. La comúnmente usada tecnología GPS depende de energía que consume del hardware, largos tiempos de bloqueo de los satélites, y de una infraestructura de satélite que está llegando al final de su vida útil. Este trabajo propone una técnica de localización que no depende de tal infraestructura, pero que también usa el sol como punto de referencia.

Mediante un sensor de luz, el dispositivo solar determinará la hora de salida y puesta del Sol. Estos datos tras ser introducidos en un Modelo Celestial, conceden la localización geográfica del Sistema de Posicionamiento a través de la longitud y latitud terrestre.

1.4 Estructura

Este trabajo está organizado en 5 secciones, estructuradas de la siguiente manera:

- Sección 1: La presente introducción. Se centra en las motivaciones por las que se ha llevado a cabo este trabajo y en general todo el proyecto SGPS, así como los objetivos fijados al inicio del trabajo.

- Sección 2: Se describe el fundamento teórico de la tecnología SGPS, esto es, el modelo celestial que se ha aplicado, algunas mejoras que se podrían desarrollar y resultados experimentales de estudios anteriores.

- Sección 3: Se detallan los componentes y dispositivos necesarios para la posterior implementación hardware y software.

- Sección 4: Se describe la implementación tanto software como hardware del sistema completo del SGPS.

- Sección 5: Resultados experimentales del presente trabajo y análisis de los mismos.

- Sección 6: Conclusiones y futuras vías de trabajo e investigación.

2. FUNDAMENTOS TEÓRICOS DEL SGPS

2.1 Introducción

El Sistema de Posicionamiento Global basado en Intensidad de Luz Solar (SGPS) es capaz de localizar objetos al aire libre por sus coordenadas (longitud y latitud) usando tan solo información de la intensidad de luz.

El sistema descrito tiene en cuenta objetos estacionarios. Esto quiere decir que el objeto tiene que estar quieto en la misma posición durante el día completo (o al menos durante el tiempo de luz del día). Debido a la precisión del sistema y al hecho de que la intensidad de luz solar se puede considerar constante en pequeñas áreas, son posibles pequeños movimientos. En este caso el término "pequeño" depende de la precisión del sistema, en pocos kilómetros.

2.2 Requisitos del Sistema

El método propuesto está diseñado de la forma más sencilla posible, sus componentes son un sensor de luz, un microprocesador y un reloj. Integrando el reloj, se mantiene una cuenta relativamente precisa de la hora y la fecha.

Para elegir el reloj, se debe tener en cuenta que el dispositivo puede preguntar en cualquier momento la hora *t* y la fecha *d,* por lo tanto es necesario un dispositivo que proporcione ambas. Una opción sencilla es un reloj digital, que puede funcionar durante muchos años con una sola batería. La hora se fija a la hora universal (UTC) lo cual hace los cálculos más simples, y está representada en forma decimal de 0 a 24 eliminando así el problema del horario de verano o distintas franjas horarias entre países. La fecha *d* es representada como el día del año, donde 1 de Enero es día 1 y así sucesivamente.

En cuanto al sensor de luz, la estrategia requiere que el dispositivo sea capaz de medir el estado de iluminación *e*. El dispositivo posible más simple es un sensor de intensidad de luz, el cual proporcione un voltaje directamente proporcional a la intensidad de luz incidente. Este tipo de sensores consume energía no eléctrica ya que en realidad transforman la energía de luz en energía eléctrica sin ninguna fuente de potencia. Podrían usarse algunos otros dispositivos, por ejemplo un simple y de bajo coste medidor de valor de exposición como los incorporados en la mayoría de las cámaras digitales de bajo costo [6 - 7].

Por último, el microprocesador que debe analizar los datos dados por el sensor y que consumirá la mayor parte de la energía del sistema.

La operación SGPS esta resumida en diagrama de flujo mostrado en la Figura 1.

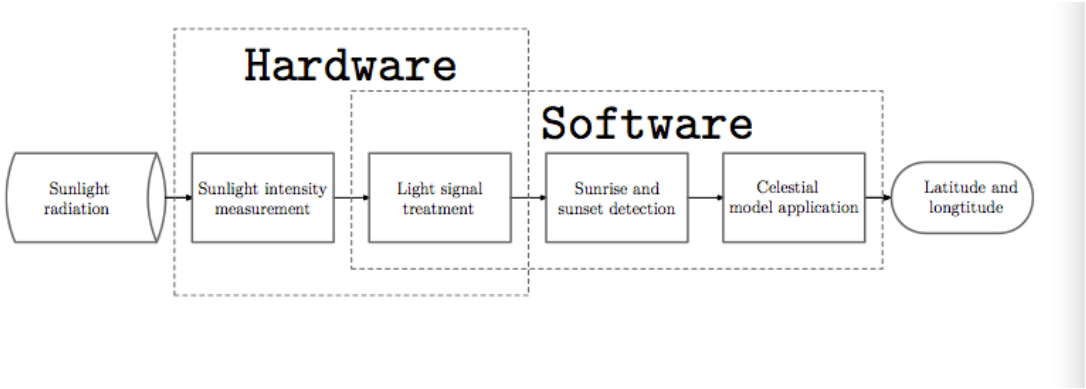

Figura 1. *Diagrama de flujo del SGPS*

Este sistema se puede diseñar en muchas formas complejas, incluidos otros dispositivos de medición de datos, o usando un microprocesador más potente, pero el sistema propuesto aquí en este documento está centrado en una simple estrategia de medición, con bajo coste, y componentes de eficiencia energética. Se han utilizado: fuente de potencia, dispositivo de almacenaje SD, reloj RTC (Real Time Clock) y sensor LDR (light dependant resistor). Esta implementación de SGPS es la más sencilla y tiene un coste alrededor de 50 euros en componentes. En la Figura 2 se pueden ver los componentes necesarios para implementación en Arduino del SGPS.

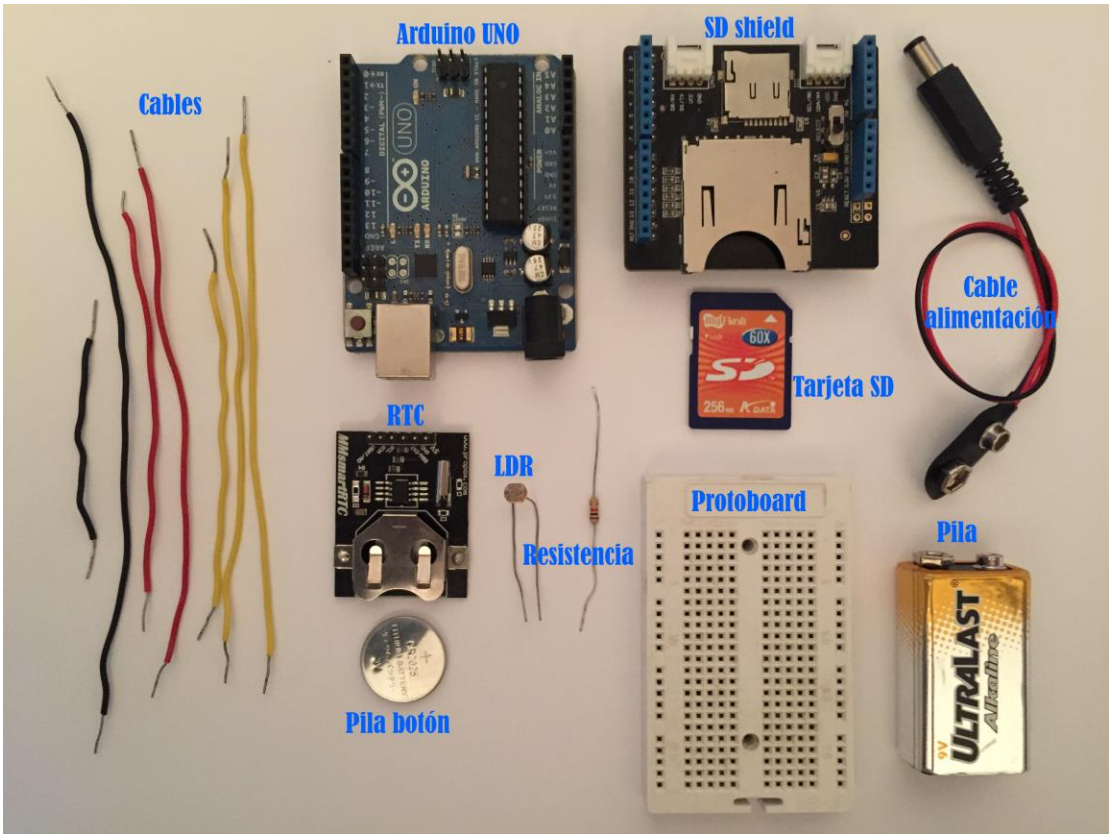

Figura 2. *Hardware necesario para el desarrollo de SGPS*

2.3 Modelo Celestial

El SGPS está basado en un algoritmo o modelo llamado Celestial Model, en español Modelo Celestial, que localiza objetos al aire libre usando solo datos de la intensidad de luz solar de una localización dada. Con esta información, le es posible obtener la hora de amanecer y anochecer para esa localización. Sólo con estos valores, el modelo celestial determina las coordenadas (tanto longitud como latitud). Así, las coordenadas se pueden expresar como función de la hora del amanecer y anochecer como se muestra en la Figura 3.

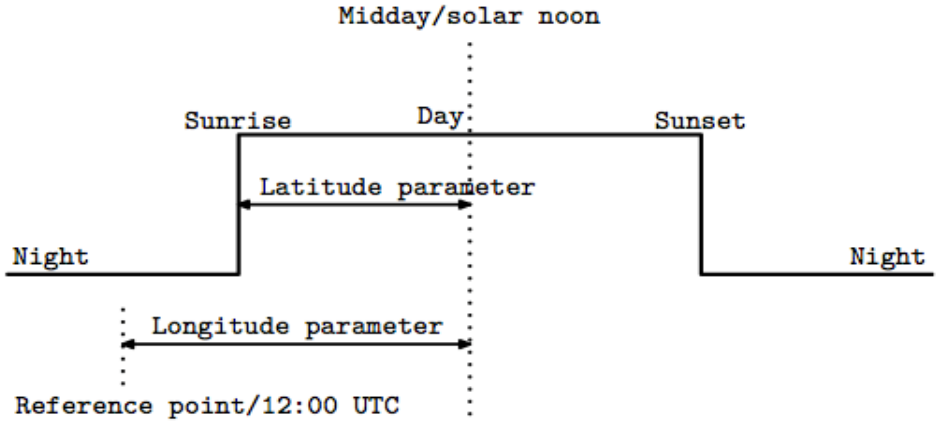

Figura 3. *Parámetros de luz solar influenciados por la longitud y latitud.*

Luego, basado en esto, el método aquí descrito se puede reducir al problema de la identificación del amanecer y anochecer para un día dado. Dada una medida precisa de la hora del amanecer $t_{sunrise}$ y de la hora del ocaso t_{sunset} , el mediodía t_{midday} es simplemente como se observa en la Ecuación 1.

$$t_{midday} = \frac{t_{sunrise} + t_{sunset}}{2}$$

Ecuación 1.

Se debe tener en cuenta que la representación UTC del tiempo asume que el amanecer ocurre antes del ocaso. Si el ocaso ocurre antes del amanecer dentro de la ventana de las 24h de hora UTC entonces significa que es un día fraccionado, porque el amanecer ocurre el día de antes (en formato UTC) que el ocaso. Esto se tiene que tener en cuenta cuando se aplique esta ecuación, porque si la ecuación se aplica directamente con un ocaso anterior al amanecer, el mediodía no será correcto (de hecho, el resultado podría ser la medianoche). Este problema se soluciona sumándole 12 horas al medio día calculado.

A continuación se muestran las fórmulas empleadas en el modelo celestial. Como se mostró en la Sección 2.2 la hora ha de ser expresada en UTC y en formato decimal (de 0 a 24), luego todos las horas serán expresadas en este formato y los ángulos en radianes.

La puesta de sol angular se puede calcular según la Ecuación 2.

$$a_{sunset} = \frac{\pi}{12}[t_{sunset} - t_{midday}]$$

Ecuación 2.

y la declinación del sol, δ, se puede aproximar por la siguiente serie de Fourier mostrada en la Ecuación 3.

$$\delta = 0.006918 - 0.399912\cos\beta + 0.070257\sin\beta - 0.006758\cos(2\beta) + 0.000907\sin(2\beta) - 0.002697\cos(3\beta) + 0.00148\sin(3\beta)$$

Ecuación 3.

donde β es la fracción de año expresada en radianes dada por la Ecuación 4.

$$\beta = \frac{2\pi}{365}d$$

Ecuación 4.

y d es el día de 1 a 365. En caso de año bisiesto, donde d puede ser 366 el valor β es cercano a 2π, que da el mismo resultado en la ecuación para δ como cuando d es 1 y β es igual a $2\pi/365$.

Finalmente, se pueden obtener las coordenadas. Para la longitud λ (en radianes), donde valores positivas representan este y negativos oeste, se emplea la siguiente Ecuación 5.

$$\lambda = 2\pi\frac{12 - t_{midday}}{24}$$

Ecuación 5.

y la latitud φ (en radianes) de la ubicación de los objetos se puede encontrar resolviendo numéricamente para la latitud usando la Ecuación 6, donde los valores positivos representan norte y los negativos sur:

$$\cos(a_{sunset}) = \frac{\sin(-0.0145) - \sin\delta - \sin\varphi}{\cos\delta\cos\varphi}$$

Ecuación 6.

Las ecuaciones dadas en esta sección permiten encontrar las coordenadas de objetos al aire libre por medio de la hora de puesta de sol y del amanecer para un día dado.

2.4 Estrategia de Medición

El sistema está diseñado para trabajar de forma autónoma, siendo capaz de localizarse a sí mismo dentro de las primeras 24 h de haber sido encendido. Esto implica que el sistema puede determinar sus coordenadas sin ninguna información previa a parte de la hora.

Dando un suministro suficiente de electricidad, por ejemplo si el objeto tiene un suministro constante de potencia, la inicialización puede ser simplemente desempeñada usando la "fuerza bruta" mediante el muestreo continuo de las condiciones de iluminación. Si la luz del sensor es muestreada a r veces por segundo, entonces esto es la mismo que M veces cada ciclo de 24h, dado por la Ecuación 7. La precisión de las mediciones estarán por lo tanto en el rango que comprende la Ecuación 8.

$$M = 24 * 60 * 60 * r$$

Ecuación 7.

$$a = \frac{360}{M}$$

Ecuación 8.

Tomará como máximo 24h identificar la localización y el esfuerzo que supone está definido por $E = M\ x\ p$, donde p es la energía consumida para llevar a cabo cada muestra y E es la energía total. Esto es la solución óptima en términos de precisión y rapidez. Sin embargo, para un dispositivo de potencia limitada, la estrategia es poco realista, un dispositivo muy simple podría quedarse sin energía después de sólo unas pocas horas o antes, a pesar de que las últimas tecnologías lanzadas están mejorando el consumo de energía y algunos dispositivos puede funcionar durante meses con el estándar de la plataforma de pilas AA.

2.5 Mejoras

Son posibles muchas mejoras teóricas. Los amaneceres se pueden predecir si se toman mediciones tempranas. Esto es debido a que la intensidad de la luz aumenta gradualmente durante algún intervalo de tiempo antes de que se pase el umbral. Si se toma una medida de la intensidad de luz que está por encima del valor de referencia de noche, pero todavía por debajo del umbral, entonces es una señal de que el amanecer se acerca pronto y la frecuencia de muestreo se puede aumentar de forma dinámica. Esto está ilustrado en la Figura 4(a) la cual muestra un auténtico gráfico de intensidad obtenido usando una webcam. Es evidente que la intensidad se eleva durante aproximadamente 20 minutos antes de los saltos del amanecer y anochecer.

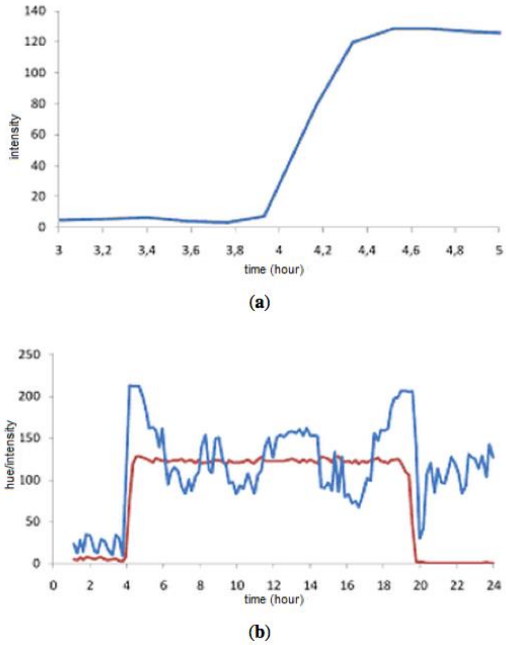

Figura 4.(a) *Incremento gradual de intensidad hacia el anochecer;* ***(b)*** *cambios bruscos de tono que predicen amaneceres y puestas de sol.*

Sin embargo, será más difícil obtener una pre-advertencia de una puesta de sol de esta manera ya que la intensidad de luz puede llegar a caerse por debajo del valor umbral por otras circunstancias como nubes. Una manera de superar esto es observar síntomas adicionales. Si el sensor de luz es capaz de capturar información de espectros de color, entonces la información adicional se puede explotar para predecir mejor los amaneceres y las puestas de sol. Por ejemplo , los sensores CCD en las cámaras digitales son capaces de detectar color además de intensidad. Esto es porque los amaneceres y las puestas de sol a menudo están caracterizados por grandes cambios en tonalidad los cuales afectan a todas las zonas. Tales cambios en tonalidad ocurren prioritariamente en las puestas de sol y así una detección de cambio de tonalidad puede usarse para predecir una inminente puesta de sol. Esto se ilustra en la Figura 4(b) la cual presenta un gráfico de 24h obtenido usando webcam. La tonalidad *h(r, g, b)* de cada pixel es calculada de las componentes *r, g* y *b* (rojo, verde y azul) como sigue:

$$h(r,g,b) = atan2\,(2r - g - b, \sqrt{3}(g - b))$$

Ecuación 9.

La línea continua muestra la imagen global de intensidad y la otra línea muestra una imagen global de la tonalidad. Claramente, la tonalidad está cambiando drásticamente justo antes del amanecer y la puesta de sol. De hecho, estos se pueden ver como los dos picos en la gráfica de tonalidad. Sin embargo, la explotación de tales síntomas es un tópico de futuras investigaciones.

3. COMPONENTES Y DISPOSITIVOS

En esta sección se dará una descripción detallada de cada uno de los elementos usados para el montaje y puesta en marcha del dispositivo SGPS se justificará la razón por la que se han escogido éstos en lugar de otros disponibles en el mercado

3.1 Arduino

Arduino es una plataforma de electrónica abierta para la creación de prototipos basada en software y hardware flexibles y fáciles de usar. Se creó para artistas, diseñadores, aficionados y cualquiera interesado en crear entornos u objetos interactivos.

Arduino puede tomar información del entorno a través de sus pines de entrada de toda una gama de sensores y puede afectar aquello que le rodea controlando luces, motores y otros actuadores. El microcontrolador en la placa Arduino se programa mediante el lenguaje de programación Arduino (basado en Wiring) y el entorno de desarrollo Arduino (basado en Processing). Los proyectos hechos con Arduino pueden ejecutarse sin necesidad de conectar a un ordenador, si bien tienen la posibilidad de hacerlo y comunicar con diferentes tipos de software (p.ej. Flash, Processing, MaxMSP).

Las placas se pueden hacer a mano o comprarlas montadas de fábrica; el software puede ser descargado de forma gratuita. Los ficheros de diseño de referencia (CAD) están disponibles bajo una licencia abierta, así pues cada uno es libre de adaptarlos a sus necesidades.

Arduino dispone de página web[4] donde se puede encontrar toda la información relativa a éste, desde la descarga del software, actualizaciones, instrucciones de uso, instalación y tutoriales hasta ejemplos de programación. En esta web además, se pueden encontrar librerías para descargar y un foro en el que gente de todo el mundo aporta ideas, proyectos, dudas y material nuevo.

Desde dicha página web es posible comprar las placas Arduino o instrucciones para hacerlas uno mismo. Existen diversos tipos de placas en función del microcontrolador integrado, tamaño o número de pines. Diversos tipos de placas se muestran en la figura 7 a continuación.

[4] http://www.arduino.cc

Figura 7. *Diferentes modelos de Arduino.*

Además de las placas de Arduino, es posible encontrar en el mercado y en la página de Arduino, ampliaciones para mejorar sus funcionalidades. Estas pueden ser por ejemplo conexión a internet, displays, o zócalos para la adaptación de tarjetas de memoria SD, como el dispositivo que se usa en este proyecto y que veremos más adelante.

Se ha escogido este microcontrolador por varias razones, el entorno que presta Arduino es un entorno simple y directo, fácil de usar y de familiarizarse con él incluso para principiantes en programación pero a la vez es suficientemente flexible para usuarios más avanzados. Otra de las razones es que al tratarse de software open source, licencia abierta, tiene una gran posibilidad de ampliación ya que gente con todo tipo de experiencia puede compartir y aportar ideas de software. No sólo en cuestión de software es posible ampliarlo, ya que, como ya se ha comentado, cualquiera puede montar las placas de Arduino gracias a que toda la información está en su página web. Esto provoca que sea un hardware ampliable.

Debido a todas estas razones, y también a que se el precio se ajusta a uno de los objetivos de este proyecto, se decidió escoger una placa Arduino como microcontrolador para el proyecto.

3.1.1 Arduino UNO

Dentro de toda la variedad que ofrece Arduino en cuestión de microcontroladores así como en variedad de montajes, en este proyecto va a hacer uso de la placa Arduino Uno basado en el microcontrolador ATmega328. Fue elegida ésta debido a que, aunque es una de las más pequeñas dentro del mercado, proporciona un número suficiente de entradas y salidas para lo que este proyecto demanda y posibles extensiones si fueran necesarias.

Además Arduino UNO incluye los pines para el protocolo de comunicación SPI, que permite comunicarse con la tarjeta de memoria SD, parte fundamental de este proyecto.

Figura 8. *Arduino UNO. Imagen cortesía de http://arduino.cc/en/Main/ArduinoBoardUno*

Otra de las razones para elegir esta placa es la memoria, Arduino UNO con ATmega328 tiene 32KB (con 0.5 KB usados para el sistema de arranque). También tiene 2 KB de SRAM y 1KB de EEPROM (la cual puede ser leída y escrita con la librería EEPROM library). Esta memoria es más que suficiente para el presente trabajo ya que se está usando una tarjeta SD adicional. Debido a esto, no se necesitan versiones de Arduino más grandes, que a la vez son más caras, y se puede seguir cumpliendo uno de los principales objetivos que es implementar este proyecto con el menor coste posible.

3.1.1.1 Características físicas

La longitud y anchura máxima del PCB Uno son 68.58 y 53.34 mm respectivamente, con el conector USB y el conector Jack extendiéndolo más allá de la dimensión formada. Cuatro agujeros de tornillo permiten fijar la placa a una superficie o carcasa. La distancia entre los pines digitales 7 y 8 es 4.064 mm.

3.1.1.2 Características de Arduino UNO

Esta placa consta de 14 pins entrada/salida digitales de las cuales pueden usarse 6 como salidas PWM, 6 entradas analógicas, un amplificador cerámico de 16 MHz, un conector USB, una conexión de energía Jack, conectores ICSP, y un botón de Reset. Contiene todo lo necesario para soportar el microcontrolador, basta con conectarlo al ordenador con un cable USB o darle potencia con un adaptador AC-DC o batería para empezar.

El UNO difiere de las placas predecesoras en que no se usa el chip controlador FTDI USB-to-serial (driver chip). Sin embargo , desarrolla ATmega16U2 programado como un convertidor USB-to-serial.

La placa está dotada de un LED integrado conectado digitalmente al pin 13. Cuando el pin está a nivel ALTO, el LED está en ON, cuando el pin está a nivel BAJO, el LED esta en OFF. Esto es muy útil a la hora de iniciarse con Arduino para familiarizarse con el entorno y con algunas funciones sencillas.

En la tabla 1 se muestra un resumen de las características de la placa Arduino UNO.

Tabla 1. *Características Arduino UNO*

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz

3.1.1.3 Alimentación

Arduino Uno se puede alimentar vía USB o con una fuente de alimentación externa. La fuente de alimentación se selecciona automáticamente.

La alimentación externa puede venir desde un adaptador AC/DC o con una batería, para ello la placa viene equipada con distintos conectores y pines para alimentarla.

Para una información más detallada de cómo alimentar la placa y de sus pines de alimentación, se puede consultar la página web[5] de Arduino en la sección de Arduino UNO. Los pines de alimentación de los que hace uso este proyecto son los siguientes:

- **5V.** Este pin saca 5V regulados por el regulador de la placa. Se puede suministrar la placa con alimentación o bien desde un DC jack (7-12v), el conector USB (5V), o el pin VIN de la placa (7-12V). Suministrándole voltaje a través de los pines de 5 o 3.3V evitando el regulador, puede dañar tu placa. No se recomienda.

- **GND.** Pines de tierra.

En la figura 9 se muestra enmarcada en rojo la sección de alimentación, POWER, donde se encuentran los pines de alimentación utilizados en este trabajo y todos los que ésta posee.

[5] *http://arduino.cc/en/Main/ArduinoBoardUno*

Figura 9. *Bloque de pines de alimentación.*

3.1.1.4 Pines de la placa

En la placa Arduino UNO se pueden encontrar pines de diversos tipos; digitales, para comunicaciones puerto serie, de interrupciones, analógicos, de señal PWM, para protocolo SPI y algunos otros como tensión de referencia y reset. Todos ellos al igual que los de alimentación se pueden hallar perfectamente descritos en la página web de Arduino en la sección de Arduino UNO anteriormente mencionada en el apartado 3.1.2.3. A continuación se detallan aquellos que han sido utilizados en este proyecto:

a. Pines Digitales.

Cada uno de los 14 pines digitales del UNO se pueden usar como entradas y salidas. Usando las funciones *pinMode(), digitalWrite(),* y *digitalRead()*. Estos funcionan a 5V. Cada pin provee o recibe un máximo de 40mA y tiene una Resistencia interna pull-up (desconectada por defecto) de 20-50 KOhms.
En el presente trabajo sólo se hace uso del pin digital número 10 para la entrada de datos del sensor LDR. El bloque de pines digitales se muestra en la figura 10.

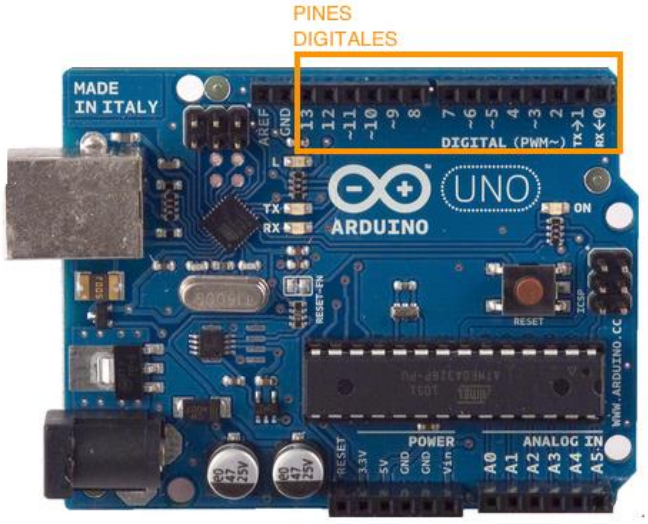

Figura 10. *Bloque de pines digitales.*

b. Pines analógicos.

El Uno tiene 6 entradas analógicas, etiquetadas desde A0 hasta A5, cada una de las cuales provee 10 bits de resolución (1024 diferentes valores). Por defecto estas miden de tierra a 5V, aunque es posible cambiar la salida alta de su rango usando el pin AREF y la función analogReference(). Además algunos de estos pines tienen funciones específicas como:
I2C: A4 o pin SDA, y A5 o pin SCL. Soporte para la comunicación I2C usando la libraría WIRE library.

Éstos dos últimos pines, A4 y A5, son los dos únicos pines analógicos que se usan en este proyecto para comunicarse con el RTC, como se detallará más adelante en el apartado 3.2. Todo el bloque de pines analógicos quedan enmarcados en la figura 11.

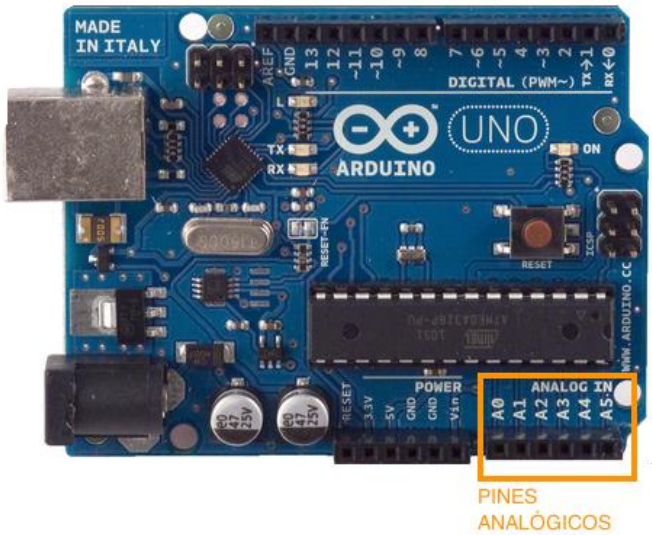

Figura 11. *Bloque de pines analógicos.*

c. Pines SPI.

Estos son los pines 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). Estos pines apoyan la comunicación SPI usando la librería SPI library.
Estos pines son los que se usan para la comunicación con el dispositivo SD.

En la figura 12 se muestran todos los pines utilizados en el proyecto.

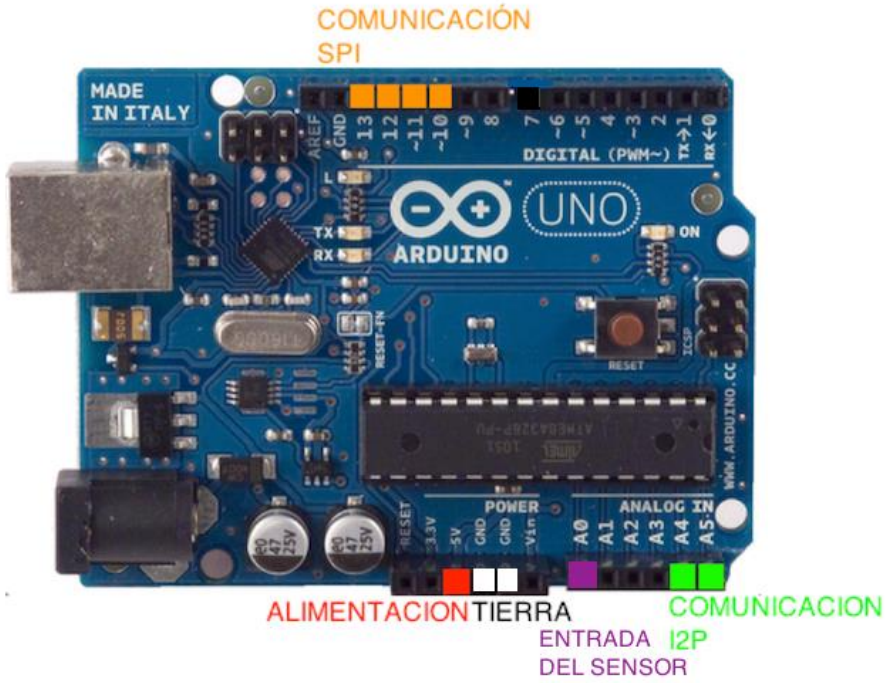

Figura 12. *Pines utilizados en la realización del proyecto.*

3.1.2 Protocolos de comunicación

Como ya se ha hablado en el apartado anterior, todas las placas Arduino, sean el modelo que sean, incluyen una serie de pines que soportan distintos tipos de comunicaciones. En los dos apartados siguientes se explican los dos protocolos de comunicación de los que se ha hecho uso en este proyecto. Éstos son el protocolo I2C y SPI.

3.1.2.1 Protocolo de comunicación I2C

El protocolo I2C/TWI, Two Wire Interface, define las reglas de cómo se pueden conectar diferentes dispositivos entre sí. Desarrollado por Philips en la década de los 80/90, convirtiéndose en la actualidad como un estándar. I2C crea un bus de comunicación entre los diferentes dispositivos en serie, esto nos permite conectar hasta 1000 dispositivos uno detrás de otro. La comunicación siempre se realizará entre dos dispositivos, uno actuará de maestro, que trasmitirá la señal para sincronizar la transferencia de datos, y otro de esclavo, en este caso, el maestro será Arduino y el esclavo el RTC. El que hace de maestro, no tiene por qué hacer esta función siempre, puede ir pasándose de uno a otro, aunque no todos los periféricos tienen esta funcionalidad, están diseñados para ser siempre esclavos. Aquí no se cambiará, Arduino siempre será el maestro.

Para identificar qué periférico ha enviado la información, cada uno de ellos tiene una dirección única, igual que una red de ordenadores donde cada ordenador tiene su propia dirección.

El bus I2C consta de 3 líneas, SDA (datos), SCL (reloj) y GND (masa)

- **SDA**. Es la línea por donde circulan los datos, formateada de la siguiente forma:

| start | A7 A6 A5 A4 A3 A2 A1 R/W | ACK | … DATA … | ACK | stop | idle |

- **SCL**. Por esta línea va la señal para sincronizar las transferencias de datos.
- **GND** o masa. Se utiliza como referencia de voltaje para el cálculo de las otras líneas.

Para el manejo de comunicaciones I2C en Arduino se debe importar la librería "Wire.h" al programa. Esta librería está incluida con las básicas que trae el entorno de Arduino.

3.1.2.2 Protocolo de comunicación SPI

El protocolo de comunicación SPI, Serial Peripheral Interface, es un método de comunicación síncrono. Este protocolo de comunicación requiere el uso de menos pins que el clásico método Bus SD, ya que con tan solo 4 PINS se puede comunicar con la tarjeta SD. Dichos pines son:

- **SLCK** (Clock). Es el pulso que marca la sincronización. Con cada pulso de reloj se lee un bit o se envía.
- **MOSI** (Master Output Slave Input). Salida de datos del Máster (Arduino) y entrada de datos al Esclavo (Shield SD).
- **MISO** (Master Input Slave Output). Salida de datos del Esclavo y entrada en el Máster.
- **SS**/Select. Para seleccionar un esclavo o para que el Máster le diga al Esclavo que se active.

Como los datos tan solo se envían por un terminal, no se puede llegar al potencial de utilización ofrecido por la comunicación BUS SD, pero es suficiente para este proyecto.

En la Tabla 4 se muestra el pineado de la SD.

Tabla 4. Pineado de la tarjeta SD

PIN	Nombre	Descripción
1	CS	Activación de tarjeta
2	DATA IN	Datos desde el host
3	Vss	GNd
4	Vcc	Alimentación
5	CLK	Clock
6	Vss	GND
7	DATA OUT	Salida de datos
8	RSV	Reservado
9	RSV	Reservado

3.1.3 Entorno de desarrollo para Arduino

Como ya ha sido mencionado anteriormente en el apartado 3.1.1., el entorno de desarrollo de Arduino se puede descargar de forma gratuita desde la página web de Arduino. La web proporciona distintas versiones en función del sistema operativo que estés utilizando (Windows, Mac OS X, Linux…). Una vez descargado se ha de ir actualizando la versión ya que éstas salen cada poco tiempo. Esto no supone mucho problema ya que el propio entorno muestra un mensaje cuando existe una nueva versión de forma que se pueda descargar de inmediato. En este proyecto la última versión utilizada fue la versión 1.0.5.

Se puede encontrar ayuda como manuales y ejemplos en la página web de Arduino. También se pueden encontrar librerías para descargar y agilizar la programación.

Para la realización del código de este proyecto se han descargado la librería "RTC.lib", para comunicarse con el dispositivo RTC, disponible en la página web. Las demás librerías utilizadas están incluidas dentro de las básicas que trae de serie el entorno de Arduino.

En el Anexo 1 se puede consultar todo el proceso de iniciación en este entorno así como cómo llevar a cabo la descarga y posterior instalación de librerías.

3.1.4 Monitorización Serie

El entorno Arduino incluye un monitor serie mediante el cual es posible escribir o recibir datos por puerto serie.

Este monitor muestra los datos enviados desde la placa Arduino ya sea vía USB o vía serie. Para descargar los datos en Arduino es tan simple como hacer click en cargar, desde la barra de comandos del sketch (representado con una flecha). Se debe además seleccionar la velocidad de la comunicación en el propio monitor, como se muestra en la Figura 13, e incluir en el programa la función *Serial.begin()* para indicar la velocidad. En este proyecto se trabaja a 9600 baudios.

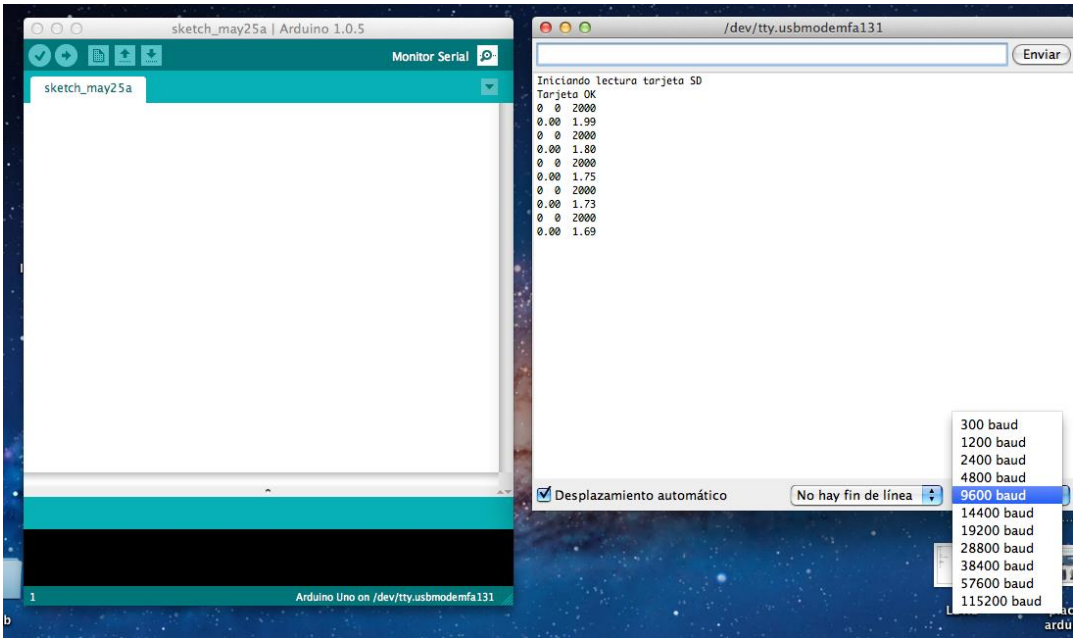

Figura 13. *Vista del monitor serie en entorno Arduino en MAC OS X.*

Todos los mensajes que desde el código se manden imprimir mediante la función *Serial.print()* se mostrarán en este monitor. También mostrará todos los datos recogidos de la placa Arduino, como datos de sensores, siempre que así se indique en el código mediante la función anteriormente mencionada, o leerá cualquier comando que se escriba.

El puerto serie por el que lee o envía es el puerto serie 0. Si está permitido el acceso a este monitor serie, es que el Arduino está conectado al ordenador a través del USB, y esta conexión está ligada al puerto serie 0.

3.2 RTC

El RTC es un reloj de tiempo real con batería trasera que permite que el microcontrolador siga manteniendo la hora incluso si es reprogramado o si se pierde la alimentación. Perfecto para registro de datos, toma de horas, temporizadores y alarmas, etc. En este proyecto se utiliza el DS1307 que es el RTC más popular, y es el que mejor trabaja con chips basados en 5V tales como el Arduino. Se puede observar este dispositivo en la Figura 14.

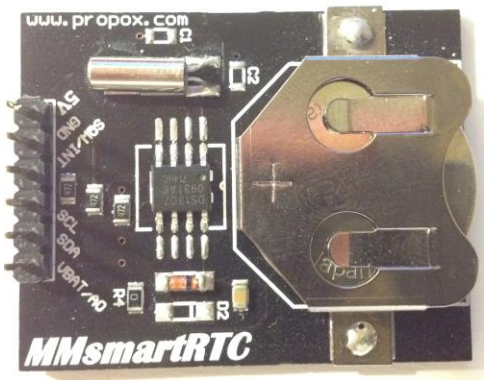

Figura 14. *Reloj RTC.*

Otros motivos por los que se ha elegido este dispositivo son:

- Todas las partes incluyendo PCB, cuerpo y batería están incluidas.
- Rápido de montar y usar.
- La instalación se hace enchufándolo en cualquier "breadboard" o usando cables.
- Tiene una vida útil de 5 años o más.

3.2.1 Características del RTC

El RTC (Real Time Clock) o en español, reloj de tiempo real, es básicamente un reloj de pulsera. Funciona con una pila y mantiene la hora incluso cuando hay un corte de alimentación. Usando un RTC, se puede mantener corriendo la hora durante largos periodos de tiempo, incluso si se reprograma el microcontrolador o se desconecta del USB o enchufe de alimentación.

La mayoría de los microcontroladores, incluyendo el Arduino, tienen un cronómetro integrado llamado millis() y hay también temporizadores dentro del chip que mantienen la hora durante largos periodos de tiempo como minutos o días. Debido a esto puede surgir la duda de por qué utilizar un RTC aparte, pero el problema fundamental del millis() es que sólo mantiene el paso del tiempo si Arduino fuera la última alimentación. Esto significa que cuando la alimentación se conecta, el milisegundero del temporizador es puesto a 0. El Arduino no sabe si es "Martes" u "8 de Marzo", todo lo que puede decirte es que "han pasado 14.000 milisegundos desde que fue puesto en marcha".

Entonces, si se quisiera poner la hora en marcha en Arduino, se tendría que programar en la hora y fecha y así tenerlo contando desde ese punto. Pero si se pierde la alimentación, se tendría que resetear la hora.

Mientras este tipo de temporizador es válido para algunos proyectos, otros proyectos como el aquí presente necesitarán tener un temporizador consistente que mantenga la hora que no se

resetee cuando la batería de Arduino se agote o sea reprogramado, ya que es fundamental para el proyecto llevar un seguimiento de la hora a la que se toman las medidas de luz constantemente. Por tanto, se incluye un RTC por separado. El chip RTC es un chip especializado que tan solo mantiene el paso del tiempo. Puede contar años bisiestos y sabe cuántos días hay en un mes.

El RTC que se ha usado es el DS1307. Se ha elegido éste dispositivo ya que ofrece una alta precisión a lo largo de los años, y a pesar de esto, este tiene muy bajo coste y puede funcionar durante años con una pequeña pila de botón. Además este RTC utiliza el protocolo I2C que da mucha flexibilidad de trabajo, al proporcionar una aplicación para poder controlar dispositivos o aplicaciones en las que el conocimiento de la hora es fundamental, como es el caso que de este proyecto.

Siempre y cuando tenga una pila para funcionar, el DS1307 marcará la hora durante largos periodos de tiempo, incluso cuando Arduino pierda la alimentación, o sea reprogramado.

3.3 Tarjeta de Memoria SD y Dispositivo SD

Para el desarrollo del proyecto se ha hecho uso de una tarjeta de memoria SD y un dispositivo SD para su conexión con Arduino. Estos son necesarios para guardar en un archivo de texto todas las muestras de luz tomadas con el sensor y la respectiva hora a las que se tomaron, para crear una base de datos gracias a la cual, podremos averiguar que tensión nos está proporcionando el sensor en la puesta de sol y la salida de sol y contribuir también a la base de datos de la plataforma SGPS para futuras ampliaciones e investigaciones.

3.3.1 Tarjeta SD

La tarjeta SD es un dispositivo de almacenamiento. Es una memoria de tipo Flash, la cual nos permite almacenar una gran cantidad de datos en comparación a otro tipo de memorias tal y como podrían ser las memorias EEPROM.

Estas memorias poseen dos protocolos de comunicación. El protocolo estándar de nominado BUS SD, que requiere del uso de 4 líneas de paralelas para comunicarse con el dispositivo, y el protocolo SPI (Serial Peripheral Interface) que es el que se ha usado en este proyecto.

3.3.1.1 Características de las Tarjetas SD

Las tarjetas SD poseen 9 pines, los cuales se describen en la Tabla 2.

Tabla 2. *Resumen de pines de las tarjetas SD.*

PIN	*Nombre*	*Descripción*
1	*CS*	*Selección del chip*
2	*DI*	*Entrada de datos*
3	*Vss*	*GNd*
4	*Vcc*	*Alimentación*

5	*CLK*	*Clock (SPI)*
6	*Vss*	*GND*
7	*D0*	*Salida de datos*
8	*D1*	*Salida de datos*
9	*D2*	*Salida de datos*

En la Figura 15 se pueden ver algunos ejemplos de tarjetas SD en su variedad de tamaños.

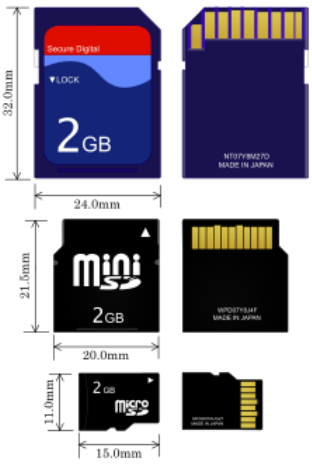

Figura 15. *Diferentes tipos de tarjetas SD.*

Internamente la tarjeta no sólo está dotada de chips de memoria flash, sino que también posee un microcontrolador para gestionar el tipo de protocolo utilizado para la comunicación y la autorización, o no, a la copia de datos almacenados en la tarjeta.

3.3.2 Dispositivo SD

Para conectar la tarjeta a Arduino es necesario el uso de un dispositivo donde se pueda introducir la tarjeta SD y a su vez conectarlo a Arduino.

Se eligió este dispositivo SD, SD Card Shield v3.1 de Seeed Studio, mostrado en la Figura 16, ya que soporta tarjetas SD, SDHC, o MicroSD TF y su instalación en Arduino es muy sencilla. Aunque se pueden conseguir otros dispositivos más baratos, esto ofrece sencillez de montaje, compatibilidad de distintos formatos de tarjeta de memoria y al llevarlo todo integrado se descartan problemas de hardware durante la instalación. Esta última razón es importante ya que se probó anteriormente otro dispositivo, más sencillo y barato, y tras diversas pruebas se comprobó que al montaje le faltaban componentes y conexiones y finalmente no fue posible su puesta en marcha.

Figura 16. *Dispositivo SD Card Shield v3.1 de Seeed Studio.*

Para seleccionar el tipo de tarjeta tan sólo hay que situar el interruptor de palanca que incorpora el dispositivo. En este trabajo se hace uso de una tarjeta SD, de modo que la palanca se situará en la posición izquierda.

Aunque se pueden conseguir otros dispositivos más baratos, éste ofrece sencillez de montaje, compatibilidad de distintos formatos de tarjeta de memoria y lleva integrado todo lo necesario para funcionar y así se descartan problemas de hardware durante la instalación. Esta última razón se tuvo mucho en cuenta, ya que anteriormente a éste se probó otro dispositivo, más sencillo y barato, pero tras diversas pruebas, se comprobó que al montaje le faltaban componentes y conexiones y finalmente no fue posible su puesta en marcha.

3.3.2.1 Características del Dispositivo SD

- Compatible con Arduino UNO y Seeeduino (No compatible con mega)
- Compatible grove (sistema parecido a lego mediante el cual se puede encajar sobre el Arduino y otros dispositivos)
- Soporta tarjetas SD, Micro SD y SDHC
- Compatible con tensión lógica de $3.3V$ *y* $5V$
- Suministra alimentación de $2.6{\sim}3.6V$ DC
-

3.3.2.2 Especificaciones

A continuación en la tabla 3, se muestran las especificaciones del dispositivo.

Tabla 3. *Especificaciones del dispositivo SD Card Shield v3.1 de Seeed Studio.*

Item	*Min*	*Typical*	*Max*	*Unit*
Voltage	*2.7*	*3.3*	*3.6*	*V*
Current	*0.159*	*40*	*200*	*mA*
Supported Card Type	*SD card (<=2G); Micro SD card (<=2G); SDHC card (<=16G) / /*			
Dimension	*57.15x44.70x19.00*			*mm*
Net Weight	*16.6*			*g*

3.4 SENSOR LDR

Un LDR (Light Dependent Resistor) es una resistencia que varía su valor en función de la luz recibida, cuanta más luz recibe, menor es su resistencia. Es la pieza clave del proyecto, junto con el Arduino, ya que la intención es medir la intensidad de luz solar con el objetivo de hallar las coordenadas en las que nos encontramos. Se puede observar en la figura 17.

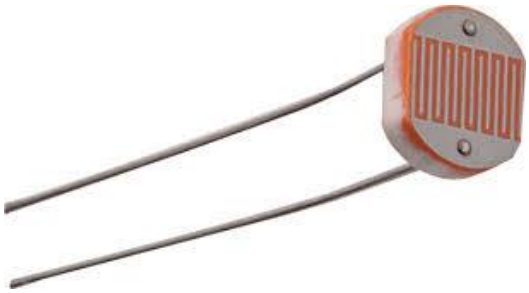

Figura 17. *Sensor LDR.*

Se escogió este sensor por ser el más sencillo dentro del mercado. Uno de los objetivos de este proyecto es fabricar el prototipo del SGPS de la manera más sencilla a la vez económica posible. Debido a esto, el prototipo se construyó con éste sensor, aunque, como se verá en apartados posteriores, una de las posibles mejoras a este proyecto, podría ser escoger un sensor más sofisticado.

3.4.1 Teoría de Funcionamiento

Un LDR está fabricado con un semiconductor de alta resistencia como puede ser el *sulfuro de cadmio*. Si la luz que incide en el dispositivo es de alta frecuencia, los fotones son absorbidos por la elasticidad del semiconductor dando a los electrones la suficiente energía para saltar la banda de conducción. El electrón libre que resulta (y su hueco asociado) conduce electricidad, de tal modo que disminuye la resistencia.
Las células de sulfuro del cadmio se basan en la capacidad del cadmio de variar su resistencia según la cantidad de luz que incide la célula. Cuanto más luz incide, más baja es la resistencia. Las células son también capaces de reaccionar a una amplia gama de frecuencias, incluyendo infrarrojo (IR), luz visible, y ultravioleta (UV).

3.4.2 Características del LDR

El rango de resistencia que nos puede dar un LDR desde la total oscuridad hasta la plena luz, nos va a variar de un modelo a otro, pero en general oscilan entre unos 50Ω a 1000Ω cuando están completamente iluminadas y entre 50KΩ y varios MΩ cuando está completamente a oscuras. Este sensor aunque no es extremadamente preciso, será suficiente, debido a que en este trabajo la linealidad no es un factor importante dado que lo único que se pretende distinguir cuándo se sobrepasa un determinado umbral de luz. Cuando esto ocurra, el programa identificará ese valor como valor de luz en el amanecer o en el ocaso.

4. IMPLEMENTACIÓN

Para este proyecto se requiere una implementación software y una implementación hardware. La combinación de ambas es lo que le permitirá funcionar como un sistema completo.

4.1 Implementación hardware

La implementación hardware se refiere a la unión física de los dispositivos, así pues tendremos 4 conexiones distintas, del ordenador a Arduino, y de Arduino a cada uno de los dispositivos anteriormente descritos, el sensor LDR, el RTC y el dispositivo para la tarjeta de memoria SD.

4.1.1 Conexión del Ordenador con Arduino

Arduino se conecta al ordenador mediante un cable USB 2.0 AM/BM. La conexión AM es la estándar que va conectada a cualquier puerto serie del ordenador ya que además de alimentar servirá para la comunicación serie con Arduino, mientras que a Arduino se conectara la BM en el puerto designado a tal efecto. Este cable se puede observar en la Figura 18.

Figura 18. *Cable USB 2.0 AM/BM*

Esta conexión proporciona alimentación a la placa y a cualquier dispositivo que esté conectado a ella. En este proyecto, el objetivo es que el sistema completo sea independiente del ordenador, de manera que pueda ser trasladado sin necesidad de llevar consigo un ordenador. Para ello usaremos un cable que nos permita conectar una pila común de petaca de 9 V a la placa de Arduino a través del puerto Jack y de este manera alimentar el sistema sin necesidad de conectarlo a un ordenador. Este cable se muestra en la figura 21 junto con la pila.

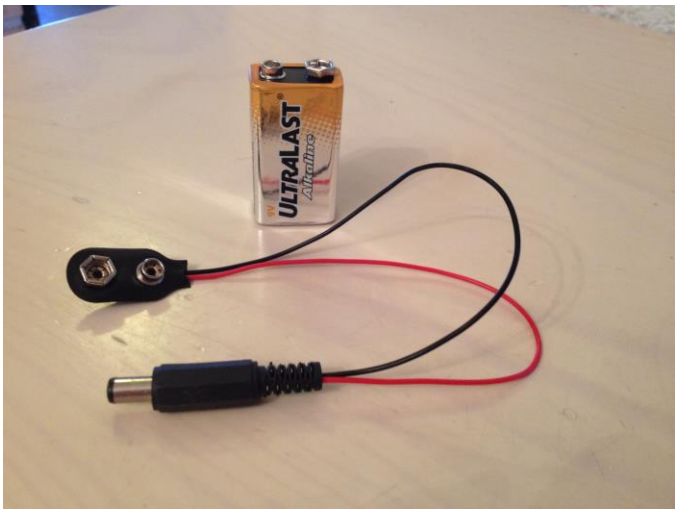

Figura 19. *Cable Jack para la alimentación de Arduino desde una pila de petaca 9 V.*

Por ultimo en la Figura 20 observamos la placa conectada, al ordenador, Figura 20.a, o la pila, Figura 20.b.

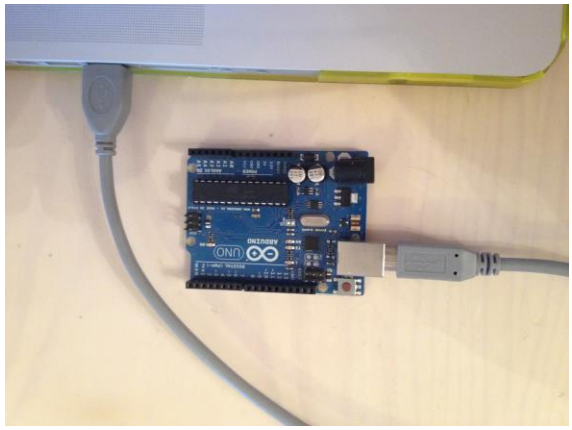

Figura 20.a. *Conexión a ordenador*

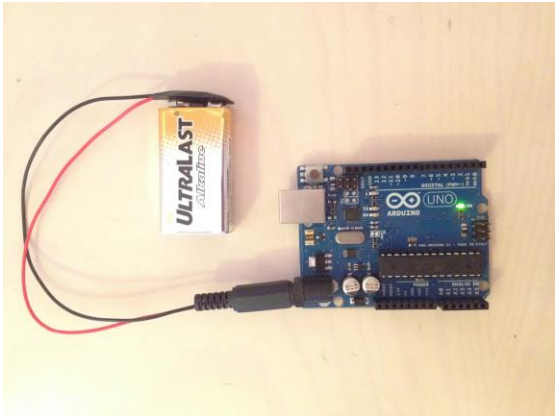

Figura 20.b. *Conexión a batería independiente.*

4.1.2 Conexión de Arduino con el Reloj RTC

El reloj RTC irá conectado a Arduino según el protocolo de comunicación I2C, como ya se ha explicado en el apartado 3.1.3.1, la placa Arduino trae incorporados los pines necesarios para llevar a cabo esta comunicación. En particular el modelo de RTC que se ha escogido, facilita la instalación, ya que tan sólo ha de ser "pinchado" en una protoboard y tirar los cables hasta los pines correspondientes de Arduino.

Según el modelo de Arduino que tengamos, los pines de SDA y SCL, varían. En el caso del Arduino UNO los pines son: para SDA el pin A4, y el SCL el pin A5, que son puertos analógicos. El cableado para la conexión se puede ver en la Figura 21.

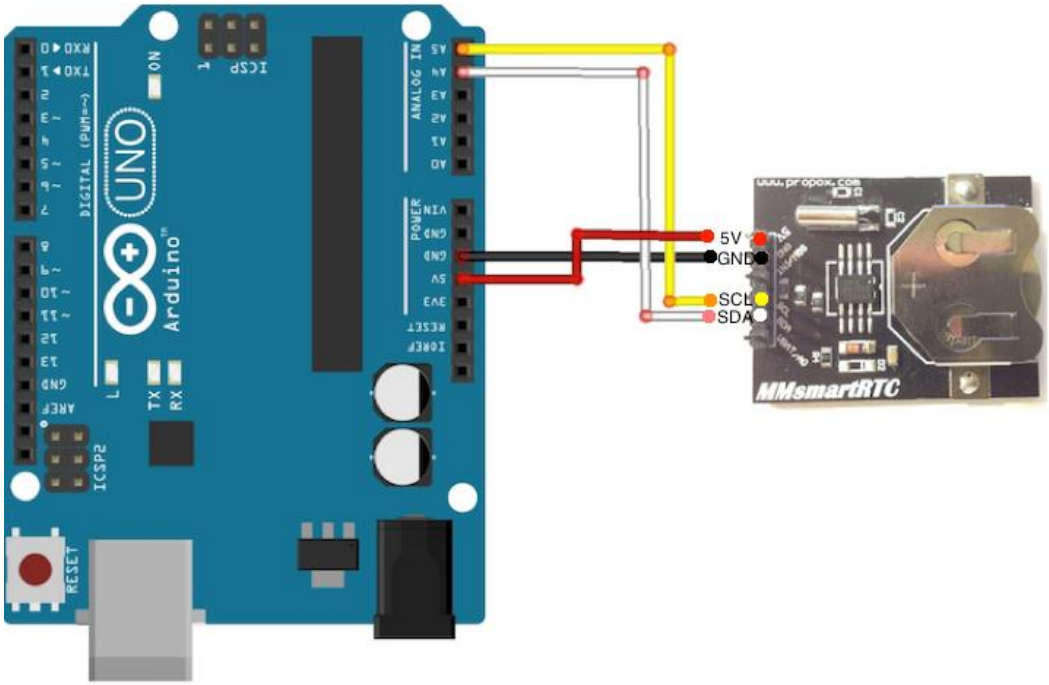

Figura 21. *Esquema de conexión de pines en Arduino UNO para comunicación I2C.*

Una vez se sabe cómo conectarlo por el protocolo de comunicación I2C, se conecta el reloj a una protoboard y cableamos como hemos visto en la figura 21. La conexión completa quedaría como se observa en la Figura 22.

Figura 22. *Cableado para la conexión del reloj RTC con la placa Arduino.*

4.1.3 Conexión Arduino con el SD Card Shield

La conexión entre el Arduino y la tarjeta SD se realizará mediante el protocolo de comunicación SPI, el cual queda explicado en el apartado 3.1.3.2.

Los pines de Arduino para conectarlo por el protocolo SPI, varían dependiendo del modelo de Arduino. En el caso de este proyecto, con Arduino UNO, estos pines son los que se observan en la tabla 5. El cableado para la conexión de Arduino UNO con la tarjeta SD se puede observar en la figura 23.

Tabla 5. Pines en Arduino UNO para protocolo de comunicación SPI.

PIN	Nombre
10	SS
11	MOSI
12	MISO
13	SCK

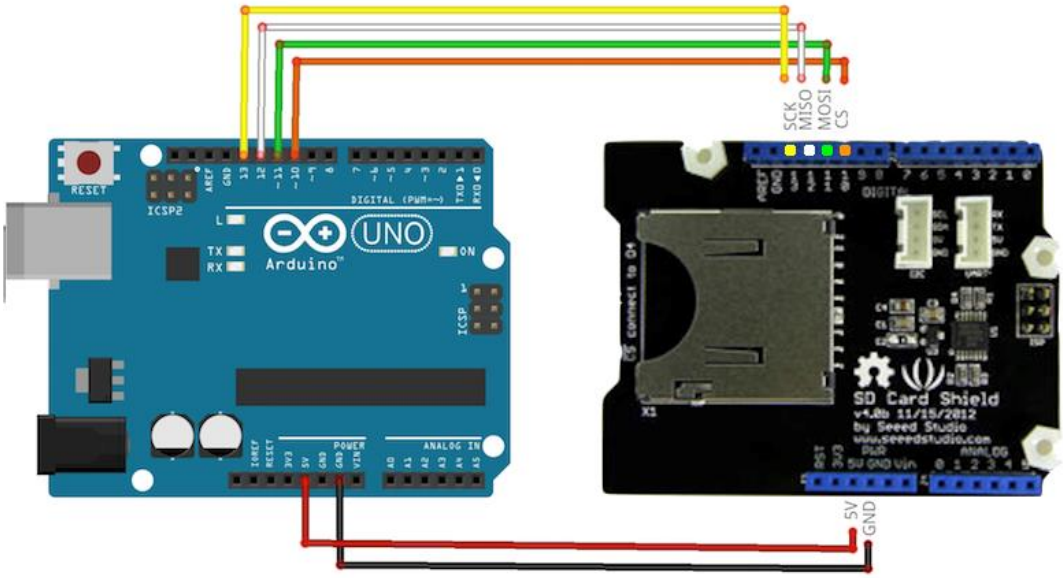

Figura 23. *Esquema para la conexión de pines en Arduino UNO para comunicación SPI.*

En este caso, el dispositivo elegido, viene ya montado sobre un zócalo, facilitando la instalación al máximo. Ésta se hace acoplándolo encima de la placa de Arduino haciendo coincidir los pines, y así poder acceder a ellos a través de la placa de la SD, que es la que queda encima. Así, este dispositivo, usa sólo los puertos de SPI de Arduino, dejando libres los demás sobre la paca del SD card shield. El resultado del montaje se muestra en la Figura 24.

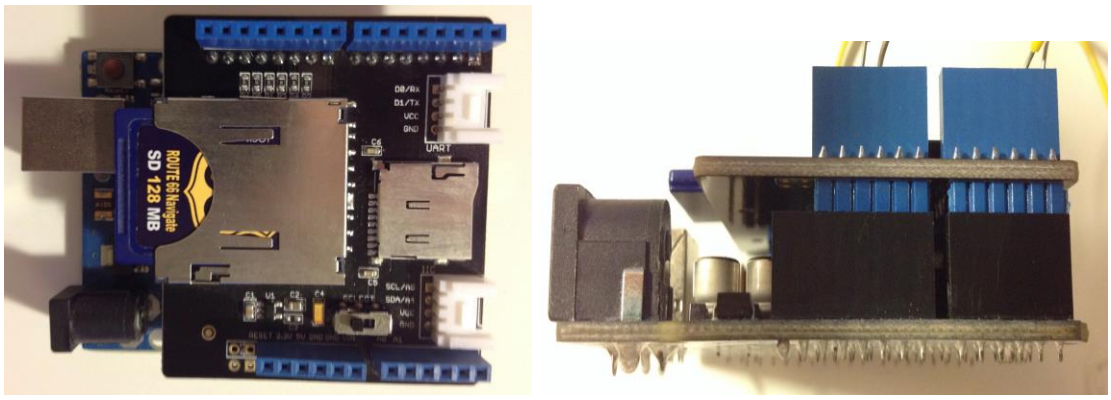

Figura 24. *Acoplamiento del dispositivo SD card shield sobre Arduino UNO.*

4.1.4 Conexión Arduino con el Sensor LDR

Para la conexión de un LDR a Arduino se debe hacer un divisor de tensión, de donde se sacará la señal para conectar a la entrada de Arduino. En este caso utilizaremos el pin número 10.

Se puede conectar de dos maneras diferentes, como se observa en la Figura 25.

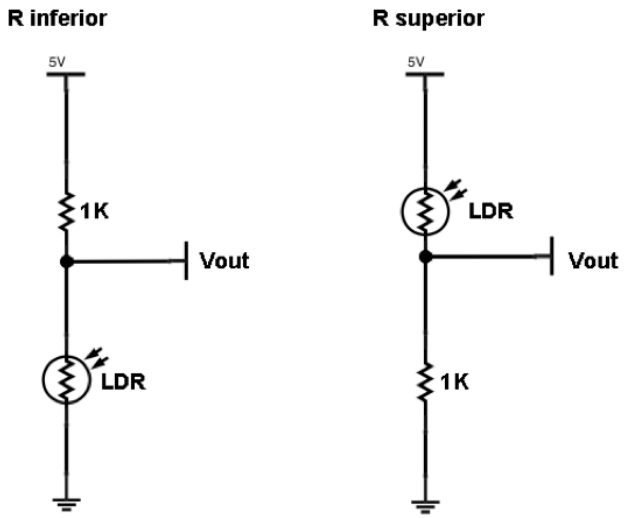

Figura 25. *Divisor de tensión para el conexionado del sensor LDR a Arduino UNO.*

Si utilizamos el LDR como resistencia inferior del divisor de tensión, proporcionará la tensión máxima cuando el LDR capte plena oscuridad, ya que estará oponiendo el máximo de su resistencia al paso de la corriente derivándose esta por Vout al completo, si lo utilizamos como resistencia superior, el resultado será el inverso, se obtendrá la tensión máxima cuando esté completamente iluminado, ya que se comportará prácticamente como un circuito abierto, con una resistencia de 50Ω o 100Ω. Este último será nuestro caso.

Finalmente en la figura 26 se muestra cómo sería la conexión del LDR con Arduino.

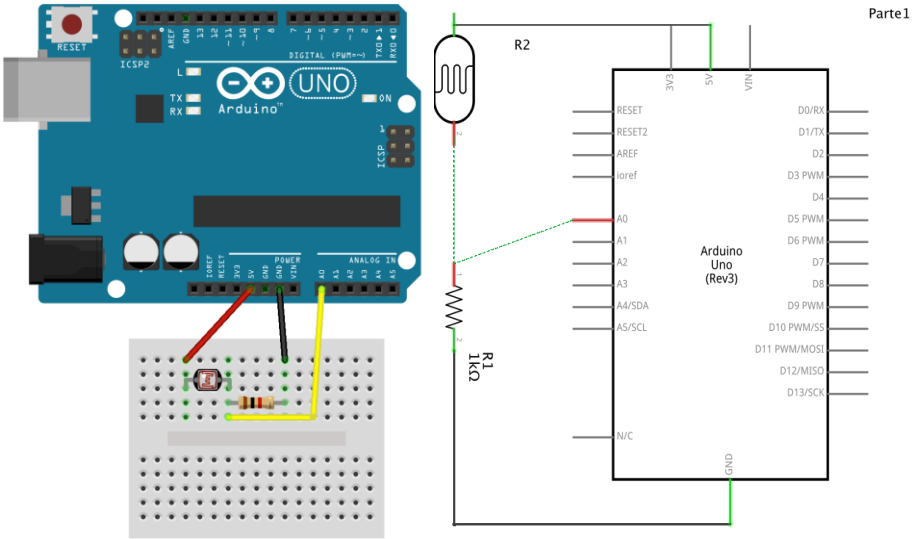

Figura 26. *Conexión sensor LDR a Arduino.*

4.1.5 Conexión Arduino-Sensor LDR-RTC-SD Card Shield

El montaje de todo el sistema quedará como se muestra en la figura 27.

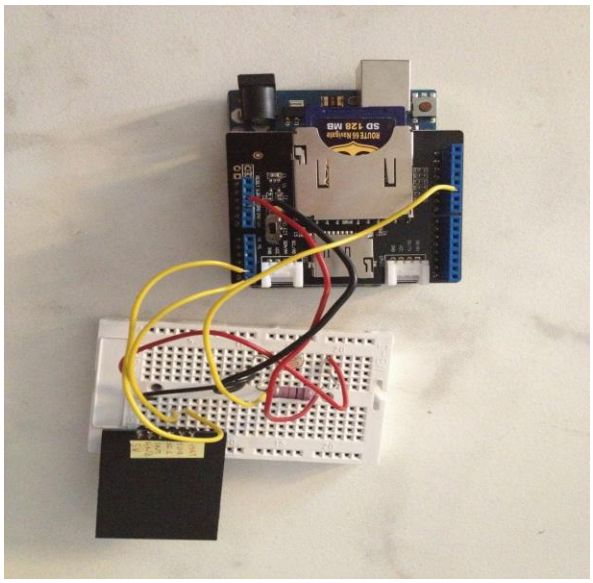

Figura 27. *Implementación hardware del sistema SGPS con Arduino UNO.*

4.2 Implementación software

Implementación software se refiere a la creación del código por el cual el sistema funcionará de manera autónoma. Este programa quedará grabado en el microcontrolador de Arduino de modo que el dispositivo podrá trabajar sin necesidad de estar conectado a ningún ordenador.

Antes de poner en marcha el SGPS o incluso crear el programa de funcionamiento, es necesario la calibración del sistema. Así, en este apartado se empezará explicando en qué consiste esta calibración y cómo se ha llevado a cabo.

4.2.1 Calibración del Sistema

Como ya ha sido comentado, antes de comenzar a programar el código mediante el que funcionará el SGPS, es necesario hacer una calibración del sistema. Esto es, que antes de empezar se necesita una base de datos con los que se determinará cuál es el umbral de tensión que se traspasa en el LDR tanto al amanecer como al anochecer, o lo que es lo mismo, qué cantidad de intensidad lumínica se produce durante los amaneceres y anocheceres.

Esto es necesario ya que el SGPS definitivo trabajará en base a esos valores de tensión. El dispositivo proporcionará una latitud y una longitud en base a la hora a la que estos fenómenos se producen. El SGPS sabrá qué horas ha de tomar para introducirlas en el modelo celestial cuando la intensidad de luz sobrepase dichos umbrales de tensión fijados durante la calibración.

Para elaborar la calibración del sensor LDR y del SGPS en general, se ha desarrollado un programa en el cuál se toman medidas de luz a una frecuencia de muestreo $\boldsymbol{Tsample}$. Cada tres medidas tomadas, se hace la media y se guarda dicho valor en la tarjeta de memoria SD. Junto a los valores de luz, se guarda la hora UTC a la que se tomó cada una de ellos, y todo en

un archivo de texto con un formato genérico. Cada día a las 0:00 se crea un archivo nuevo, de modo que se obtendrán tantos archivos como días haya estado el dispositivo midiendo.

Una vez se han recogido datos suficientes durante días, se analizan haciendo uso de la librería SGPS [13] en C++. Esta librería proporcionará unas coordenadas y el error que existe entre esas y las reales donde se tomaron las medidas indicándole por terminal, dónde está el archivo con los datos de luz y horas del día que se quiera analizar y, unos valores umbrales de luz para el amanecer y el anochecer. Para que la librería pueda leer estos archivos de texto es necesario que los archivos se generen con una determinada estructura y formato genéricos, los cuales se muestran a continuación en la figura 28 y un ejemplo en la figura 29.

```
genericdec
latitud longitud
dia mes año
hora_decimal intensidad
hora_decimal intensidad
hora_decimal intensidad
hora_decimal intensidad
hora_decimal intensidad
hora_decimal intensidad
hora_decimal intensidad
hora_decimal intensidad
hora_decimal intensidad
...
```

Figura 28. *Estructura de archivo genericdec para la ejecución de la librería SGPS en C++.*

```
genericdec
-3.60 40.15
5 5 2015
0.00 0.00
0.02 0.00
0.03 0.00
0.05 0.00
0.07 0.00
0.08 0.00
0.10 0.00
0.12 0.00
0.13 0.00
0.15 0.00
...
```

Figura 29. *Ejemplo de archivo genericdec generado el día 5 de Mayo de 2015*

De esta manera se pretende encontrar unos valores umbrales que minimicen el error acumulado. Para hacer esto, se han analizado 7 días por separado con la librería SGPS. En la figura 30 se pueden ver cómo imprime la librería los resultados por el terminal al introducirle distintos valores umbrales para el archivo genérico generado el día 7 de Mayo del 2015.

```
Day info: 127 2015. Transitions: 0. Sunrise: 5.07 Sunset: 19.17
Degrees error: Lat 0.450832 Long -0.872458
Percentage error: Lat 0.250462 Long -0.242349
MacBook-Pro-de-Teresa:build teresafernandez$ ./bin/sgps_apply ../../data/07:05:2015.txt -f genericdec
-s 0.7 0.52
Day info: 127 2015. Transitions: 0. Sunrise: 4.98 Sunset: 19.17
Degrees error: Lat -0.749149 Long -1.54746
Percentage error: Lat -0.416194 Long -0.42985
MacBook-Pro-de-Teresa:build teresafernandez$ ./bin/sgps_apply ../../data/07:05:2015.txt -f genericdec
-s 0.6 0.52
Day info: 127 2015. Transitions: 0. Sunrise: 4.95 Sunset: 19.17
Degrees error: Lat -1.14914 Long -1.77246
Percentage error: Lat -0.638413 Long -0.492351
```

Figura 30. *Ejemplo de uso de la librería SGPS para el día 07/05/2015*

En las siguientes tablas se muestran los valores de intensidad de luz con los que se ha probado y los errores que se han obtenido para cada valor. En morado se observan los valores negativos y en azul los valores positivos. De estas pruebas, se tomarán los valores para los que menor error se genere cada día.

Tabla 6. *Error en grados de latitud y longitud según diferentes valores de intensidad de luz solar para el día 7/5/2015.*

I luz anochecer / I luz amanecer	0.1		0.15		0.2		0.25		**0.3**		0.35		0.4		0.45	
	E latitud	E longitud	E latitud	E longitud	E latitud	E longitud	E latitud	E longitud	E latitud	E longitud	E latitud	E longitud	E latitud	E longitud	E latitud	E longitud
0.1	-6,44	-2,14	-6,04	-2,44	-5,64	-2,67	-5,34	-2,89	-5,14	-3,04	-4,94	-3,19	-4,74	-3,27	-4,34	-3,57
0.15	-6,14	-1,92	-5,64	-2,22	-5,34	-2,44	-5,04	-2,67	-4,74	-2,82	-4,54	-2,97	-4,44	-3,04	-3,94	-3,34
0.2	-5,64	-1,62	-5,24	-1,92	-4,94	-2,14	-4,54	-2,37	-4,34	-2,52	-4,14	-2,67	-3,94	-2,74	-3,54	-3,34
0.25	-5,54	-1,54	-5,14	-1,84	-4,74	-2,07	-4,44	-2,29	-4,24	-2,44	-3,94	-2,59	-3,84	-2,67	-3,34	-3,04
0.30	-5,14	-1,24	-4,64	-1,54	-4,34	-1,77	-3,94	-1,99	-3,74	-2,14	-3,54	-2,29	-3,34	-2,37	-2,94	-2,97
0.35	-5,04	-1,17	-4,54	-1,47	-4,24	-1,69	-3,84	-1,92	-3,64	-2,07	-3,34	-2,22	-3,24	-2,29	-2,74	-2,67
0.4	-4,54	-0,87	-4,14	-1,17	-3,74	-1,39	-3,34	-1,62	-3,14	-1,77	-2,94	-1,92	-2,74	-1,99	-2,24	-2,59
0.45	-4,44	-0,79	-3,94	-1,09	-3,64	-1,32	-3,24	-1,54	-3,04	-1,69	-2,74	-1,84	-2,64	-1,92	-2,14	-2,29
0.5	-4,24	-0,64	-3,74	-0,94	-3,34	-1,17	-3,04	-1,39	-2,74	-1,54	-2,54	-1,69	-2,44	-1,77	-1,94	-2,22
0.55	-3,94	-0,49	-3,54	-0,79	-3,14	-1,02	-2,74	-1,24	-2,54	-1,39	-2,24	-1,54	-2,14	-1,62	-1,64	-1,92
0.6	-3,64	-0,27	-3,14	-0,57	-2,74	-0,79	-2,44	-1,02	-2,14	-1,17	-1,94	-1,32	-1,74	-1,39	-1,24	-1,69
0.65	-3,34	-0,12	-2,94	-0,42	-2,54	-0,64	-2,14	-0,87	-1,94	-1,02	-1,64	-1,17	-1,54	-1,24	-1,04	-1,54
0.7	-3,24	-0,04	-2,74	-0,34	-2,44	-0,57	-2,04	-0,79	-1,74	-0,94	-1,54	-1,09	-1,44	-1,17	-0,84	-1,47
0.75	-3,04	0,1	-2,54	-0,19	-2,14	-0,42	-1,74	-0,64	-1,54	0,79	-1,24	-0,94	-1,14	-1,02	-0,64	-1,32
0.8	-2,74	0,25	-2,24	-0,04	-1,94	-0,27	-1,54	-0,49	-1,24	-0,64	-1,04	-0,79	-0,84	-0,87	-0,34	-1,17
0.85	-2,64	0,32	-2,14	0,02	-1,74	-0,19	-1,44	-0,42	-1,14	-0,57	-0,84	-0,72	-0,74	-0,79	-0,24	-1,09
0.9	-2,44	0,47	-1,94	0,17	-1,54	-0,04	-1,14	-0,27	-0,84	-0,42	-0,64	-0,57	-0,54	-0,64	-0,05	-0,94
0.95	-2,44	0,47	-1,94	0,17	-1,54	-0,04	-1,14	-0,27	-0,84	-0,42	-0,64	-0,57	-0,54	-0,64	-0,05	-0,94
1.0	-2,14	0,62	-1,64	0,32	-1,24	0,1	-0,84	-0,12	-0,64	-0,27	-0,34	-0,42	-0,24	-0,49	0,35	-0,79
1.05	-2,04	0,7	-1,54	0,4	-1,14	0,17	-0,74	-0,04	-0,54	-0,19	-0,24	-0,34	-0,14	-0,42	0,45	-0,72
1.1	-1,74	0,85	-1,24	0,55	-0,84	0,32	-0,54	0,1	-0,24	-0,04	0,05	-0,19	0,15	-0,27	0,75	-0,57
1.15	-1,54	1	-1,04	0,7	-0,64	0,4	-0,24	0,25	**0,05**	**0,1**	0,35	-0,04	0,45	-0,12	0,95	-0,42

0.5		0.55		0.6		0.65		0.7		0.75		0.8		0.85		0.9	
E latitud	E longitud	E latitud	E longitud	E latitud	E longitud	E latitud	E longitud	E latitud	E longitud	E latitud	E longitud	E latitud	E longitud	E latitud	E longitud	E latitud	E longitud
-4,24	-3,64	-3,94	-3,79	-3,74	-3,94	-3,64	-4,02	-3,34	-4,17	-3,14	-4,32	-3,04	-4,39	-2,74	-4,54	-2,54	-4,69
-3,84	-3,42	-3,64	-3,57	-3,34	-3,72	-3,24	-3,79	-3,04	-3,94	-2,74	-4,09	-2,64	-4,17	-2,44	-4,32	-2,14	-4,47
-3,34	-3,12	-3,14	-3,27	-2,94	-3,42	-2,74	-3,49	-2,54	-3,64	-2,24	-3,79	-2,14	-3,87	-1,94	-4,02	-1,64	-4,17
-3,24	-3,04	-3,04	-3,19	-2,74	-3,34	-2,64	-3,42	-2,44	-3,57	-2,14	-3,72	-2,04	-3,79	-1,74	-3,94	-1,54	-4,09
-2,74	-2,74	-2,54	-2,89	-2,24	-3,04	-2,14	-3,12	-1,94	-3,27	-1,64	-3,42	-1,54	-3,49	-1,24	-3,64	-1,04	-3,79
-2,64	-2,67	-2,44	-2,82	-2,14	-2,97	-2,04	-3,04	-1,74	-3,19	-1,54	-3,34	-1,44	-3,42	-1,14	-3,57	-0,84	-3,72
-2,14	-2,37	-1,94	-2,52	-1,64	-2,67	-1,54	-2,74	-1,24	-2,89	-1,04	-3,04	-0,84	-3,12	-0,64	-3,27	-0,34	-3,42
-2,04	-2,29	-1,74	-2,44	-1,54	-2,59	-1,44	-2,67	-1,14	-2,82	-0,84	-2,97	-0,74	-3,04	-0,54	-3,19	-0,24	-3,34
-1,79	-2,14	-1,54	-2,29	-1,24	-2,44	-1,14	-2,52	-0,84	-2,67	-0,64	-2,82	-0,54	-2,89	-0,24	-3,04	0,05	-3,19
-1,54	-1,99	-1,24	-2,14	-1,04	-2,29	-0,84	-2,37	-0,64	-2,52	-0,34	-2,67	-0,24	-2,74	0,05	-2,89	0,35	-3,04
-1,14	-1,77	-0,84	-1,92	-0,64	-2,07	-0,54	-2,14	-0,24	-2,29	0,05	-2,44	0,15	-2,52	0,45	-2,67	0,75	-2,82
-0,84	-1,62	-0,64	-1,77	-0,34	-1,92	-0,24	-1,99	0,05	-2,14	0,35	-2,29	0,45	-2,37	0,75	-2,52	0,95	-2,67
-0,74	-1,54	-0,54	-1,69	-0,24	-1,84	-0,14	-1,92	0,15	-2,07	0,45	-2,22	0,55	-2,29	0,85	-2,44	1,15	-2,59
-0,54	-1,39	-0,24	-1,54	0,05	-1,69	0,15	-1,77	0,45	-1,92	0,75	-2,07	0,85	-2,14	1,15	-2,29	1,45	-2,44
-0,24	-1,24	0,05	-1,39	0,35	-1,54	0,45	-1,62	0,75	-1,77	0,95	-1,92	1,15	-1,99	1,45	-2,14	1,65	-2,29
-0,14	-1,17	0,15	-1,32	0,45	-1,47	0,55	-1,54	0,85	-1,69	1,15	-1,84	1,25	-1,92	1,55	-2,07	1,85	-2,22
0,15	-1,02	0,45	-1,17	0,75	-1,32	0,85	-1,39	1,15	-1,54	1,45	-1,69	1,55	-1,77	1,85	-1,92	2,15	-2,07
0,15	-1,02	0,45	-1,17	0,75	-1,32	0,85	-1,39	1,15	-1,54	1,45	-1,69	1,55	-1,77	1,85	-1,92	2,15	-2,07
0,45	-0,87	0,75	-1,02	0,95	-1,17	1,15	-1,24	1,45	-1,39	1,65	-1,54	1,85	-1,62	2,15	-1,77	2,45	-1,92
0,55	-0,79	0,85	-0,94	1,15	-1,09	1,25	-1,17	1,55	-1,32	1,85	-1,47	1,95	-1,54	2,25	-1,69	2,55	-1,84
0,85	-0,64	1,15	-0,79	1,45	-0,94	1,55	-1,02	1,85	-1,17	2,15	-1,32	2,25	-1,39	2,55	-1,54	2,85	-1,69
1,15	-0,49	1,45	-0,64	1,65	-0,79	1,85	-0,87	2,15	-1,02	2,45	-1,17	2,55	-1,24	2,85	-1,39	3,15	-1,54

Como se puede observar en la tabla 6, el menor error de latitud y longitud se produce para un valor de intensidad de luz solar de ***1,15 V*** al amanecer y ***0,3V*** para el anochecer. De este modo los siguientes días que usaremos para calibrar se probará con valores más cercanos a éstos, tomando, al igual que se ha hecho con el anterior, los dos valores de intensidad que menor error produzcan. Se pueden observar los resultados en las siguientes tablas.

Tabla 7. *Error en grados de latitud y longitud según diferentes valores de intensidad de luz solar para el día 6/5/2015*

I Luz anochecer / I luz amanecer	0.2		0.25		**0.3**		0.35		0.4	
	E latitud	E longitud	E latitud	E longitud	E latitud	E longitud	E latitud	E longitud	E latitud	E longitud
1.0	-2,04	-0,36	-1,64	-0,59	-1,34	-0,74	-0,94	-0,96	-0,74	-1,11
1.05	-1,74	-0,21	-1,34	-0,44	-1,14	-0,59	-0,74	-0,81	-0,44	-0,96
1.1	-1,54	-0,06	-1,14	-0,29	-0,84	-0,44	-0,44	-0,66	-0,14	-0,81
1.15	-1,54	-0,06	-1,14	-0,29	-0,84	-0,44	-0,44	-0,66	-0,14	-0,81
1.2	-1,34	0,01	-0,94	-0,21	-0,74	-0,36	-0,34	-0,59	-0,04	-0,74
1.25	-1,14	0,15	-0,74	-0,06	-0,44	-0,21	-0,04	-0,44	0,25	-0,59
1.3	-0,84	0,3	-0,58	0,08	-0,14	-0,06	0,25	-0,29	0,55	-0,44
1.35	-0,74	0,38	-0,34	0,15	**-0,04**	**0,01**	0,35	-0,21	0,65	-0,36
1.4	-0,74	0,38	-0,34	0,15	-0,04	0,01	0,35	-0,21	0,65	-0,36
1.45	-0,44	0,53	-0,04	0,3	0,25	0,15	0,65	-0,06	0,95	-0,21

Tabla 8. *Error en grados de latitud y longitud según diferentes valores de intensidad de luz solar para el día 8/5/2015*

I Luz anochecer / I luz amanecer	0.25		**0.3**		0.35	
	E latitud	E longitud	E latitud	E longitud	E latitud	E longitud
1.1	-0,84	0,11	-0,44	-0,1	-0,14	-0,25
1.15	-0,64	0,19	-0,24	-0,03	-0,04	-0,18
1.2	-0,44	0,34	**-0,04**	**0,11**	0,25	-0,03
1.25	-0,14	0,49	0,25	0,26	0,55	0,11

Tabla 9. *Error en grados de latitud y longitud según diferentes valores de intensidad de luz solar para el día 5/5/2015*

I Luz anochecer / I luz amanecer	0.25		**0.3**		0.35	
	E latitud	E longitud	E latitud	E longitud	E latitud	E longitud
0.55	-0,5	-0,01	-0,44	-0,08	0,15	-0,38
0.6	-0,24	0,13	**-0,14**	**0,06**	0,45	-0,2
0.65	0,05	0,28	0,15	0,21	0,75	-0,08

Tabla 10. *Error en grados de latitud y longitud según diferentes valores de intensidad de luz solar para el día 12/5/2015*

I Luz anochecer / I luz amanecer	0.3	
	E latitud	E longitud
1.1	-0,34	-0,14
1.15	**-0,04**	**0,003**
1.2	0,05	0,07
1.25	0,25	0,22

Tabla 11. *Error en grados de latitud y longitud según diferentes valores de intensidad de luz solar para el día 13/5/2015*

I Luz anochecer / I luz amanecer	0.25		0.3	
	E latitud	E longitud	E latitud	E longitud
0.9	-0,44	-0,14	-0,04	-0,36
0.95	**-0,14**	**0,006**	0,15	-0,21
1.0	0,05	0,15	0,45	-0,06
1.05	0,15	1,23	0,55	0,006

Tabla 12. *Error en grados de latitud y longitud según diferentes valores de intensidad de luz solar para el día 14/5/2015*

I Luz anochecer / I luz amanecer	0.35		0.4	
	E latitud	E longitud	E latitud	E longitud
0.7	-0,64	0,23	**-0,04**	**-0,14**
0.75	-0,14	0,53	0,45	0,15
0.8	-0,04	0,6	0,55	0,23

Una vez se han obtenido para cada día los valores de intensidad de luz que menos error generaba en latitud y en longitud, marcados en rojo en las tablas anteriores, se ha hecho una media de dichos valores, cuyo resultado serán los umbrales de luz solar que el dispositivo SGPS utilizará para tomar la hora de amanecer y anochecer. Se puede observar cómo se ha calculado a continuación en la ecuación 14 el valor de intensidad de luz en voltios en el amanecer y en la ecuación 15 el de intensidad de luz en voltios al anochecer.

$$\bar{V}_{amanecer} = \frac{1{,}5 + 1{,}35 + 1{,}2 + 0{,}6 + 1{,}15 + 0{,}95 + 0{,}7}{7} = \mathbf{1{,}01V}$$

Ecuación 14. *Cálculo de la media de intensidad de luz solar al amanecer.*

$$\bar{V}_{anochecer} = \frac{0{,}3 + 0{,}3 + 0{,}3 + 0{,}35 + 0{,}3 + 0{,}25 + 0{,}4}{7} = \mathbf{0{,}31V}$$

Ecuación 15. *Cálculo de la media de intensidad de luz solar al anochecer.*

Se puede observar que el valor es bastante más alto comparativamente para el amanecer. Esto puede ser debido a que las medidas se tomaron en un lugar orientado hacia el este de modo que se registra mucha más intensidad al amanecer que al anochecer. De modo que se podría decir que este dispositivo tiene una alta dependencia de su calibración, lo cual no es favorable. Usando un sensor de luz más preciso esto mejoraría ya que daría unos valores umbrales más cercanos a 0, como se comprueba en el artículo Evolutionary Optimization Algorithms for Sunlight-based Positioning Sensor Networks [14].

Hay que recalcar que para la calibración es importante tener en cuenta distintos días, dispersos en el tiempo, cuantos más días y más dispersos en el tiempo sean, más ajustados estarán los valores. Una razón es que debido a la sensibilidad limitada del sensor LDR, habrá bastante disparidad entre días soleados y días nublados, obteniendo datos estables en días soleados, figura 31, y más escarpados e inestables en nublados, figura 32.

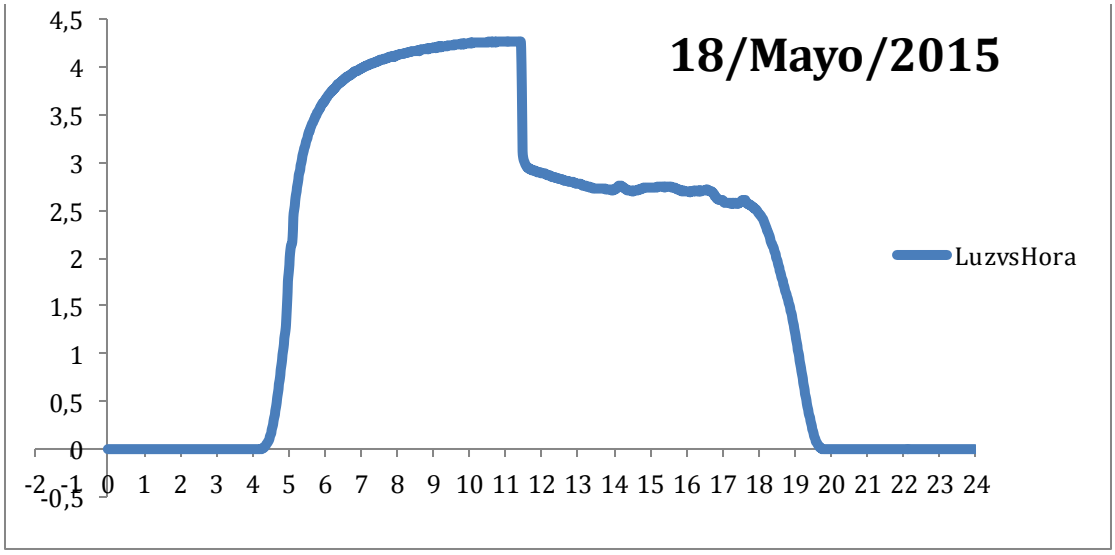

Figura 31. *Datos de luz en un día soleado*

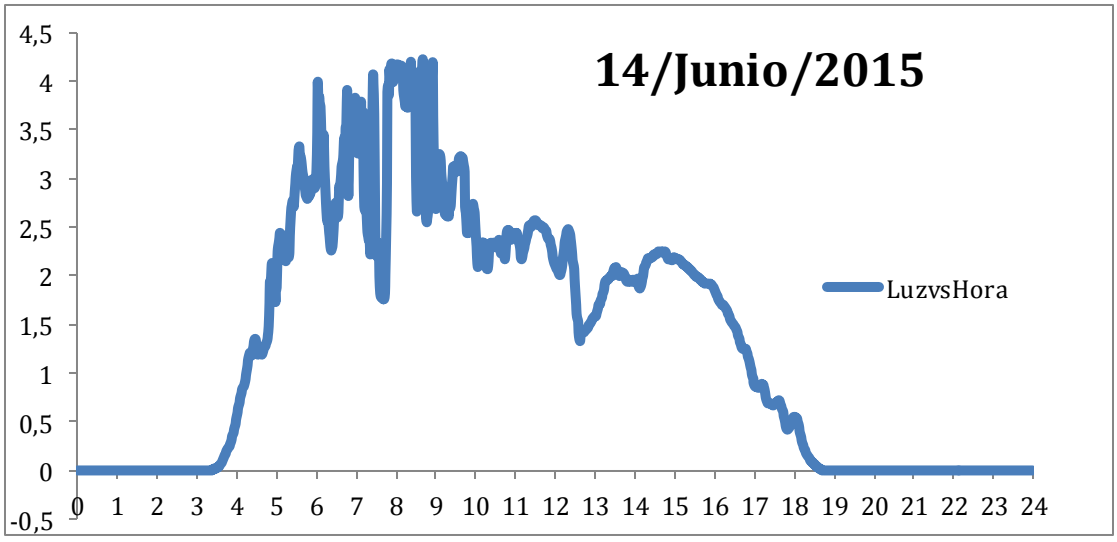

Figura 32. *Datos de luz en un día nublado*

4.2.2 Sistema SGPS

El programa SGPS se ha programado en Arduino, y su funcionamiento depende directamente de la calibración que se ha elaborado y explicado en el apartado anterior, ya que suministrará las coordenadas geográficas partiendo de la hora a la que se dan los valores de intensidad lumínica de amanecer y anochecer calculados durante la calibración. A continuación se detalla cómo funciona el programa.

En primer lugar, al programa SGPS se le introducen como constantes los dos valores de intensidad lumínica obtenidos durante la calibración, $light\_sr$ para la intensidad de luz al amanecer, y $light\_ss$ para la intensidad al anochecer.

El SGPS lee la intensidad de luz del LDR cada dos segundos, y cada tres lecturas hace una media. Esta media y la hora a la que se calculó, tomada del RTC, se almacenan en las variables $sensorval$ y $hora\_dec$ respectivamente.

En ese momento el programa comparará el valor almacenado en $sensorval$ con los valores umbrales que se introdujeron como constantes. Cuando el valor $sensorval$ iguale o supere el valor umbral para el amanecer e iguale, o caiga por debajo del valor umbral de anochecer, será el momento de tomar la hora del RTC y almacenarla como o bien hora de amanecer sr o bien hora de anochecer ss.

A continuación en la imagen 32, se observa un ejemplo gráfico de la variable $sensorval$ durante todo un día en mayo.

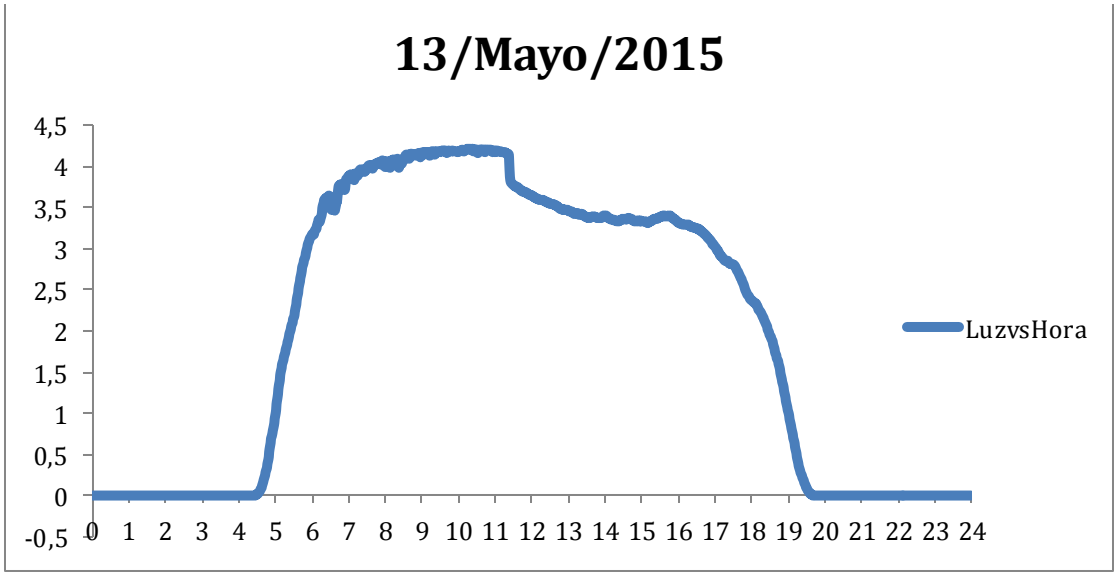

Figura 33. *Datos de luz tomados durante el día 13 de mayo del 2015.*

Durante el amanecer se observa una fuerte pendiente ascendente, y como se estabiliza durante las horas de luz del día, hasta que vuelve a caer bruscamente para dar lugar al anochecer. De ambas pendientes lo que interesa es el primer instante en el que se igualan o traspasan el valor umbral obtenido en la calibración. Por ello, el hecho de traspasar dicho umbral, no será el único indicador de que el amanecer o el anochecer está ocurriendo. Para el amanecer, una vez se sobrepasa por primera vez el umbral, el valor de luz va a ser mayor durante todas las horas de sol hasta que vuelva a caer por el anochecer. Lo mismo ocurre al anochecer, una vez se traspasa el umbral, durante toda la noche la luz será menor.

Sin embargo, sólo ocurre una vez en todo el día en que no habiendo luz durante varias horas, los valores de luz empiecen a ascender bruscamente hasta traspasar $light\_sr$, y a la inversa para el anochecer, los valores caen en picado hasta quedar sin luz. Por tanto, para seleccionar los valores de amanecer y anochecer se tiene en cuenta el valor umbral pero también el patrón de los datos que circundan dicho valor.

De esta manera se ha hecho que sí $sensorval$ supera el valor $light\_sr$ por primera vez en el día, y los cinco valores anteriores a éste son menores gradualmente, es decir, van creciendo directamente con la hora, tomará la hora del RTC. Esta hora la identificará como la hora del amanecer y la almacenará en la variable sr. De igual modo pero a la inversa se opera para identificar la hora del anochecer. Cuando el valor $sensorval$ sea menor o igual que el umbral $light\_ss$, y los 5 valores anteriores sean mayores que este por primera vez en el día, es decir decrezcan en función de la hora, se tomará la hora del RTC identificándola como la hora de anochecer y se almacenará en la variable ss. Estas dos variables son las que se introducirán en el algoritmo del modelo celestial.

Todas las medidas de luz con su correspondiente hora de medición, se van almacenando en un archivo que se crea en la tarjeta de memoria SD que incorpora el dispositivo. El nombre de los archivos, será la fecha del día que se hicieron las mediciones. Cuando un día finaliza, a las 0:00, el dispositivo abre otro archivo del día nuevo. Es en éste nuevo archivo donde imprimirá las coordenadas geográficas resultado de todo el desarrollo del algoritmo del modelo celestial.

Aunque el objetivo último de este proyecto es la obtención de las coordenadas geográficas, se ha querido además guardar los datos de luz y horas para aumentar la base de datos de la plataforma SGPS, para posibles mejoras y nuevas vías de investigación. Esto servirá de ayuda ya que crear una base de datos sólida para el SGPS conlleva mucho tiempo por tener que dejar el dispositivo días enteros midiendo.

5. RESULTADOS

Para evaluar la viabilidad, la fiabilidad y precisión del SGPS implementado, se han llevado a cabo algunas pruebas. Dichas pruebas consisten en la medición de la intensidad de luz a lo largo de días completos, y después el análisis de los datos recogidos, obteniendo la hora de amanecer y anochecer para después aplicar el algoritmo explicado en la sección 2.

Todas las pruebas realizadas con el SGPS fueron realizadas en franja horaria UTC+2 en Ciempozuelos, Madrid, España con longitud -3.60º y latitud 40,15º durante el mes de Septiembre de 2015. El dispositivo se situó en una ventana orientada al este durante días sin ser movido.

Se ha de subrayar que el mes en que se realizaron las pruebas del dispositivo es especial, tal y como se indica en estudios anteriores [4]. Se demuestra que aun tomando la hora exacta de amanecer y anochecer durante los días 21 y 22 de septiembre y anexos, se produce un error impredecible, luego insalvable, en el cálculo de las coordenadas. Este error se manifiesta sobre todo en las coordenadas de latitud, sin embargo no es muy significativo en cuanto a la longitud.

Esto ocurre tanto en septiembre como en marzo, debido a que durante estas fechas tienen lugar los equinoccios en los que la inclinación solar, δ, es 0. No obstante se mostrarán razones lógicas para pensar que el prototipo construido funciona correctamente.

5.1 Obtención de Horas de Amanecer y Anochecer

Durante la calibración se fijaron dos umbrales de intensidad lumínica, 1,01V para el amanecer y 0,3V para el anochecer. Traspasados dichos umbrales el dispositivo debe saber que se está produciendo el amanecer o el anochecer y tomar la hora a la que se producen dichos fenómenos. A continuación se muestran las horas que el dispositivo tomó como de amanecer y anochecer a lo largo del mes de septiembre del 2015 comparadas con las horas reales a las que se dieron tales fenómenos. Se han separado en dos gráficas, en la gráfica 34 se observan las distintas horas del amanecer según el día y en la figura 35 las de anochecer.

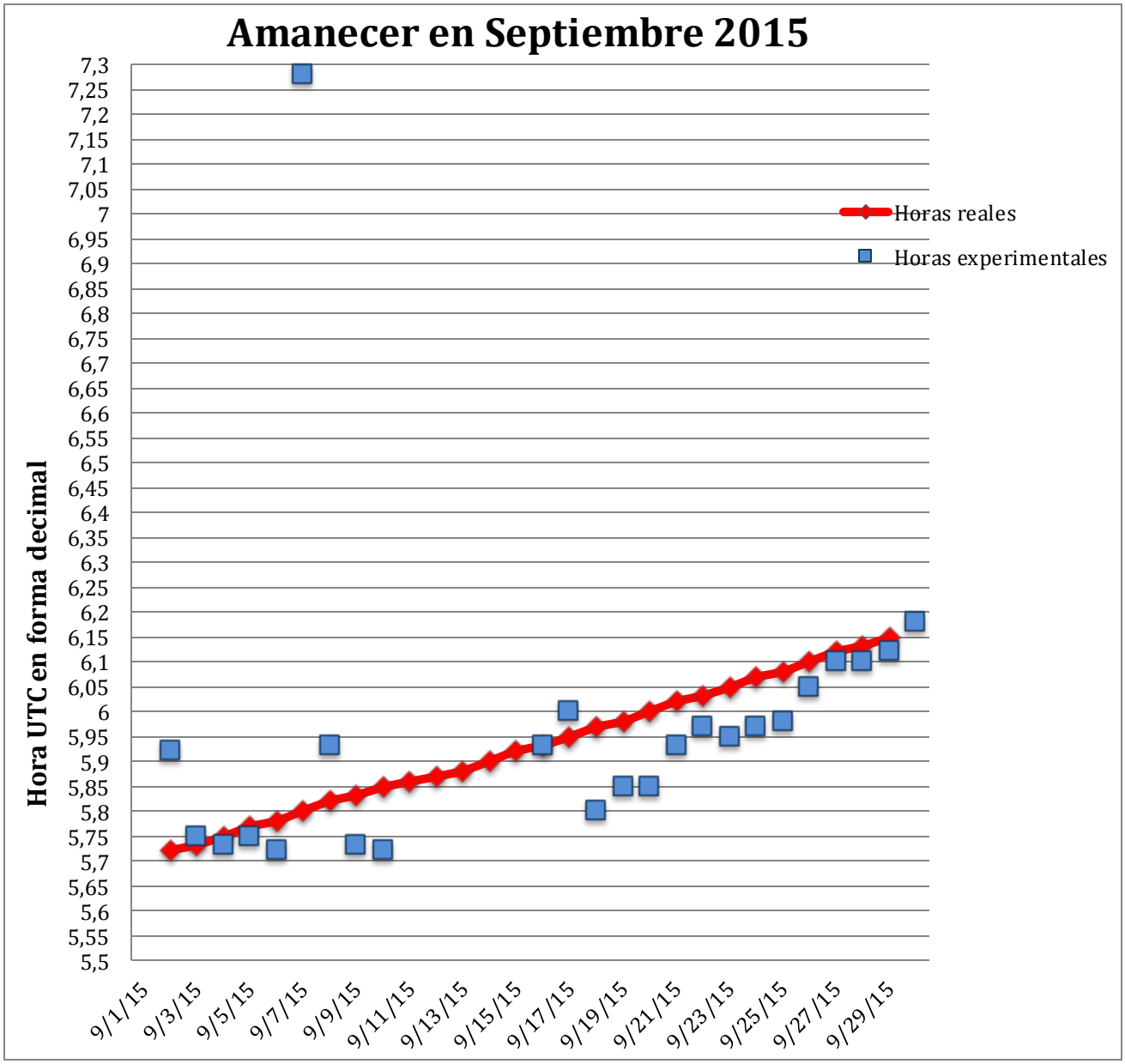

Figura 34. *Horas amanecer septiembre del 2015.*

Tal y como se muestra en la gráfica 34, las horas de amanecer reales en septiembre van creciendo gradualmente con los días debido a que como ya se sabe, según se va a acercando el otoño las horas de luz son cada vez menos a lo largo de un día. A su vez, también se puede observar ese aumento gradual en las horas experimentales tomadas.

También se observa cómo casi todas las horas que tomó el dispositivo en general fueron anteriores a las reales. Esto significa que se podría ajustar el valor de la calibración, aumentando el umbral de luz fijado para el amanecer, de modo que tome la hora de amanecer con un poco más de luz, luego un poco más tarde en el tiempo.

A pesar de esto, las horas que el SGPS tomó para el amanecer son muy semejantes a las reales. El margen entre las horas reales y las horas experimentales anda entre centésimas, lo cual en minutos significan entre 4 y 5 minutos máximo.

No obstante, se pueden observar algunas desviaciones. En particular hay una desviación especialmente grande, el día 7 de septiembre, el dispositivo tomó como hora de amanecer las 7,28 horas, existiendo 1 hora y media de diferencia con la real, las 5,8 horas. Bien, este día se

registró que amaneció nublado, lo cual nos demuestra que los días nublados el sensor escogido no opera con la precisión deseada.

Para los cálculos venideros, este dato se descartó ya que se sabe a qué fue debida su desviación, pero se tuvo en cuenta para elaborar las conclusiones.

A continuación se muestran las horas tomadas para el anochecer en la figura 35.

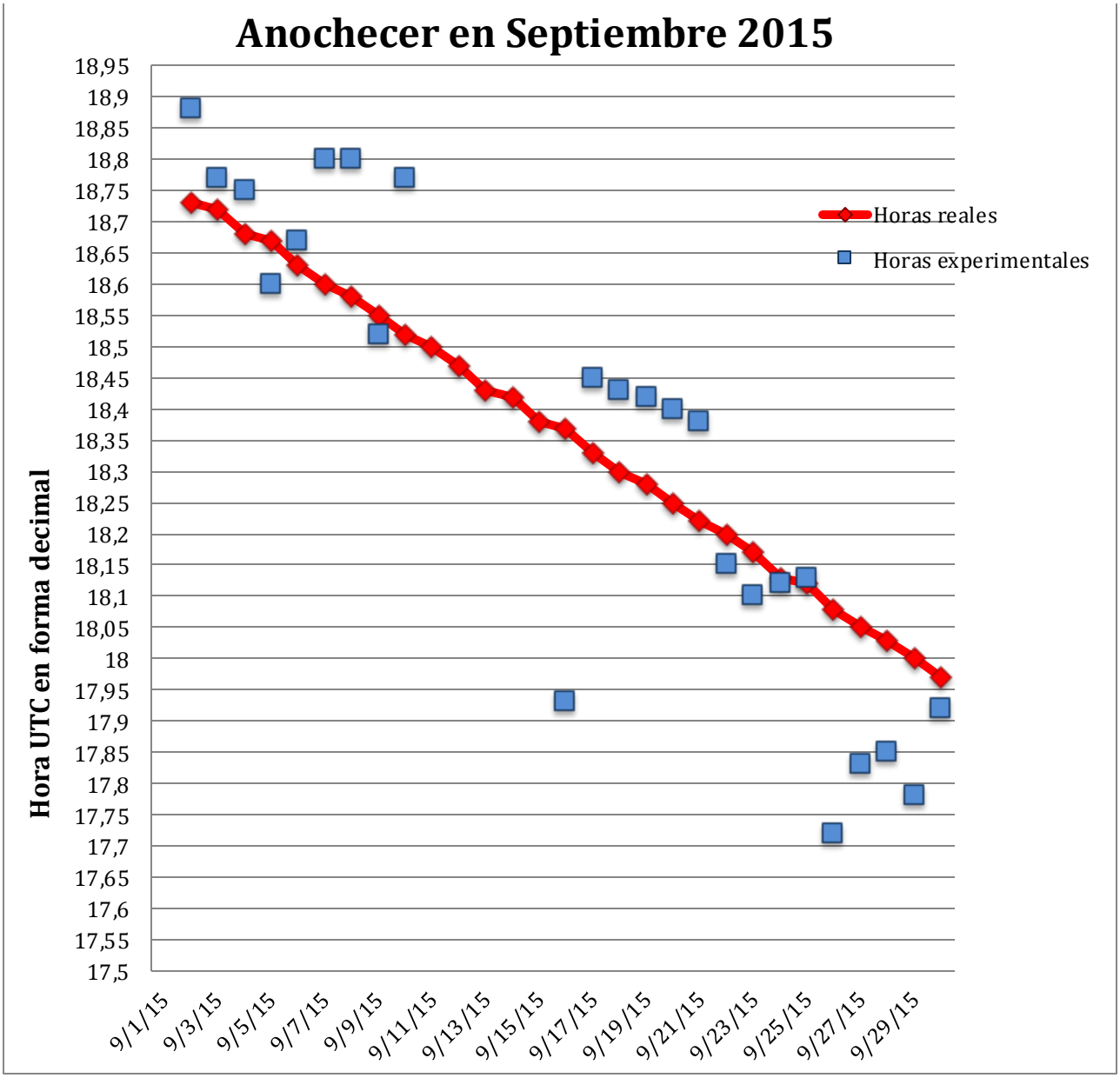

Figura 35. *Horas anochecer septiembre del 2015.*

Para el caso del anochecer se puede observar que existen algunas desviaciones más en general, pero ninguna como en el caso del amanecer. Esto se debe a que como se ha indicado al inicio de esta sección, el dispositivo se situó orientado al este. Esto significa que durante el amanecer, el sol incide directamente sobre el prototipo mientras que al anochecer no, lo cual hacer que no existan saltos desmesurados, pero genera un poco más de diferencia entre las horas tomadas y las reales.

No obstante, en general las diferencias son también de centésimas, que significan diferencias entre 5 y 6 minutos máximo. También se observan un par de días en los que se produjeron

atardeceres nublados, aunque influyeron menos que para el amanecer, como el día 16 y el 26, en los que se observa un anochecer adelantado en el tiempo.

Por otro lado, se observa como las horas reales se producen cada vez más pronto debido al acortamiento de las horas de luz en función del avance de los días.

En general se puede decir que las horas tomadas para el amanecer y el anochecer son bastante precisas tratándose de un sensor no muy sofisticado, y que la calibración se podría ajustar un poco más usando algún otro sensor que aporte una variable extra para la detección del amanecer y anochecer, como podría ser el color.

5.2 Obtención de Coordenadas

Una vez el SGPS ha identificado las horas de amanecer y anochecer de un día, procede a introducirlas en el modelo celestial, mostrado en la sección 2 de este documento, obteniendo así las coordenadas de longitud y latitud en grados. En la figura 36 se muestra los resultados obtenidos para el mes de septiembre del 2015, donde en azul se observan las coordenadas experimentales y en rojo las coordenadas reales del lugar donde se llevó a cabo el experimento.

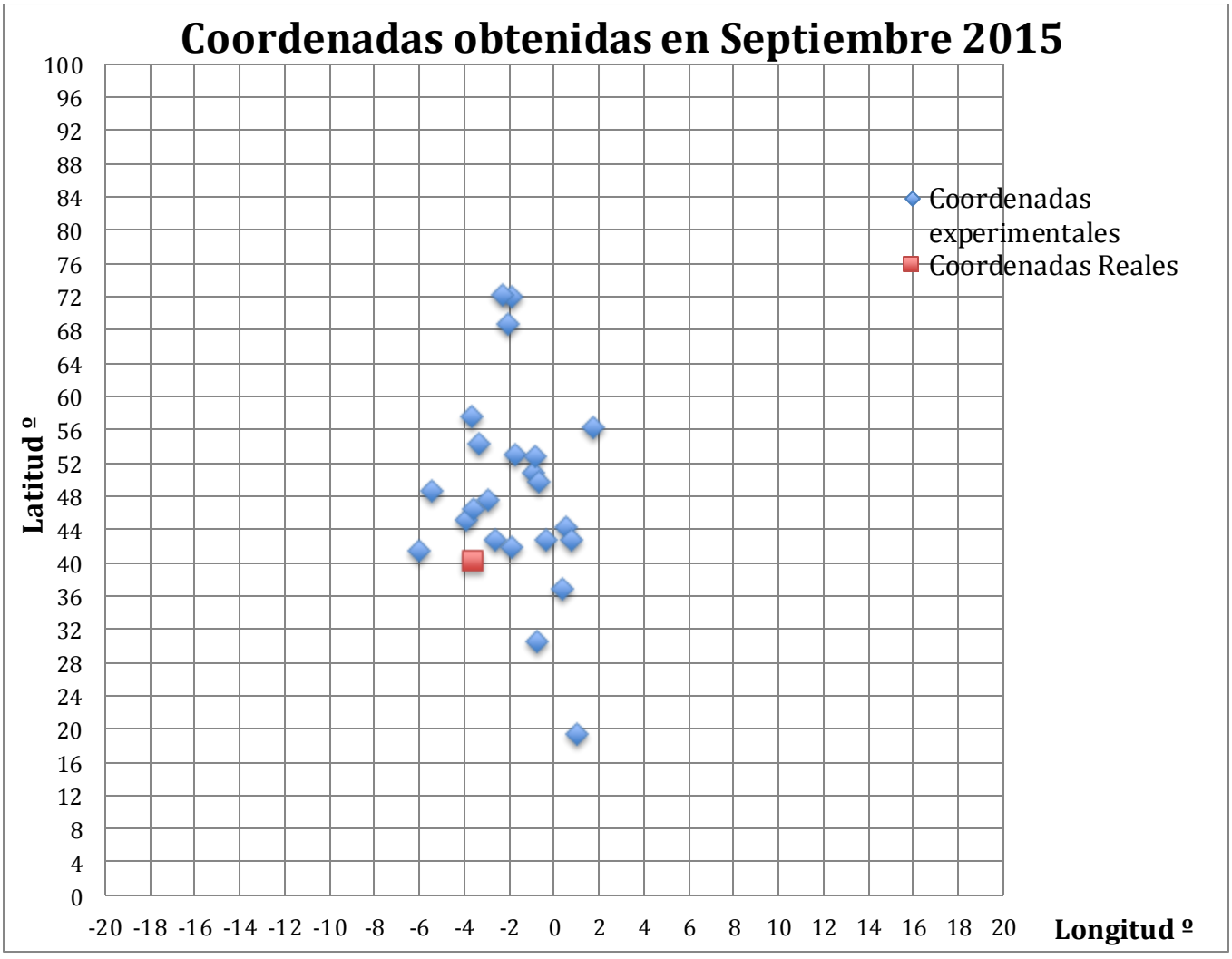

Figura 36. *Coordenadas obtenidas por el prototipo SGPS durante el mes de septiembre del 2015.*

Observando la gráfica 36, se pueden deducir varias cosas. Las coordenadas obtenidas en longitud parecen más precisas, rondando en un rango de entre -6º y 2º, siendo la real -3,60º. Sin embargo en latitud se observan más desviaciones, habiendo algunas coordenadas que difieren con la real en más de 20º.

Para analizar mejor estos datos obtenidos, se han elaborado dos gráficas en las que se muestran la latitud en función del día, y en otra la longitud en función del día. La figura 37 corresponde a las longitudes calculadas por el SGPS y la figura 38 las latitudes. En dichas figuras la línea roja corresponde a la latitud real y los puntos azules a las calculadas.

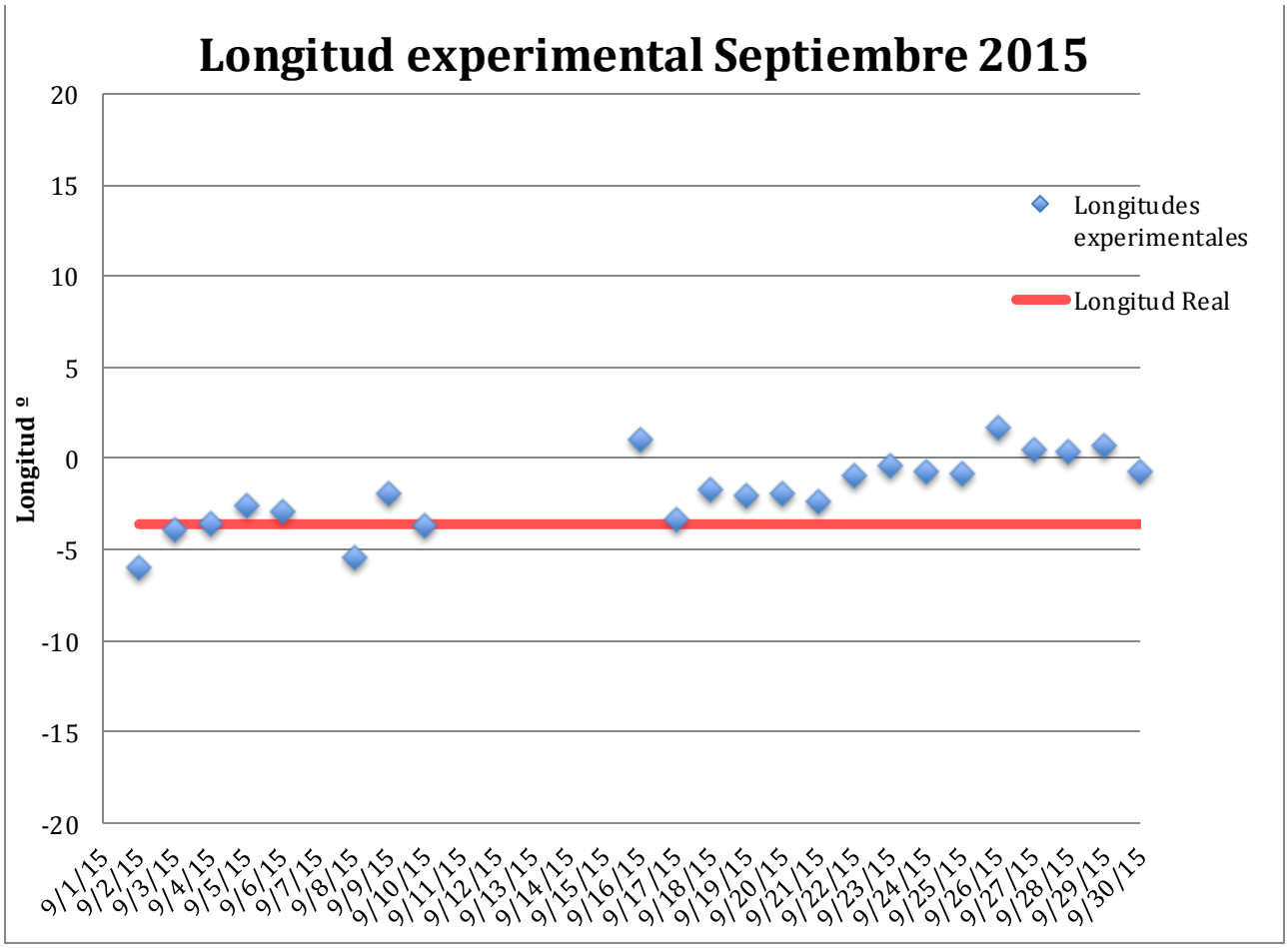

Figura 37. *Coordenadas de longitud obtenidas por el prototipo SGPS durante el mes de septiembre del 2015.*

Se puede comprobar en la figura 37 que los cálculos para la longitud son muy precisos como ya se intuía por la figura 36. También se puede decir que a pesar de que en septiembre se produce el equinoccio de otoño, esto no ha afectado al cálculo de la longitud como se puede observar en las dos anteriores gráficas. Esto se debe a que el cálculo de longitud depende directamente de las horas que el dispositivo tomó como de amanecer y de anochecer, y como se ha visto en el apartado anterior éstas fueron bastante precisas.

A continuación se muestra las latitudes que el SGPS calculó para los días de septiembre.

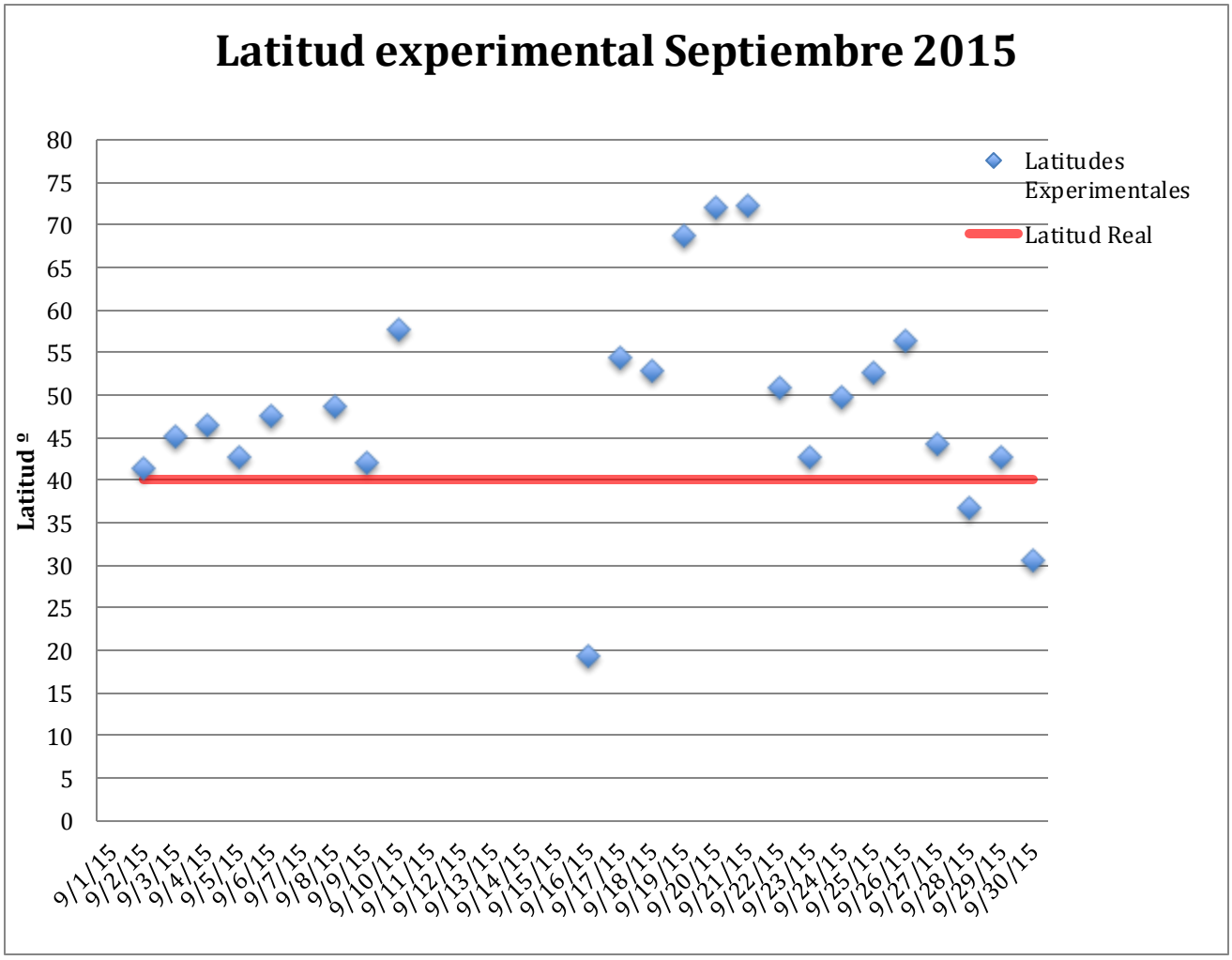

Figura 38. *Coordenadas de latitud obtenidas por el prototipo SGPS durante el mes de septiembre del 2015.*

Lo primero que se observa atendiendo a la figura 38 es que los cálculos no son tan precisos como los obtenidos de longitud, especialmente se aprecia mucha desviación en el tramo entre día 16 y el 21 de septiembre, produciéndose la máxima desviación el 21 de septiembre. Esto, como ya se ha comentado repetidamente, es lo que se esperaba por ser los días próximos al equinoccio y finalmente el 21 el día del mismo. El día 21 de septiembre la latitud calculada por el SGPS es de 72,3º siendo la latitud real de 40,15º, es decir una diferencia de 32,15º que es casi el doble de la real.

Tras esos días se observa como la latitud va estabilizándose de nuevo, de manera que si se tomaran datos de octubre cabría esperar menos error aún en los cálculos.

Para poder ver qué significan estas diferencias en cuestión de espacio y de localización, se muestran las coordenadas obtenidas y la situación real del dispositivo situadas en un mapa geográfico, figura 39. El mapa se ha obtenido introduciendo las coordenadas obtenidas en Google earth[6].

[6] *https://www.google.es/intl/es/earth/index.html*

Figura 39. *Vista en Google earth de las coordenadas obtenidas por el SGPS para el mes de septiembre del 2015.*

Una vez más se observa cómo en cuanto a longitud las coordenadas no irían más allá de la península Ibérica mientras que en latitud se ve como los días cercanos al equinoccio y en el día del mismo se aleja hasta llegar a la altura de Noruega.

Cabe pensar que estos errores se crean por el equinoccio, y por lo tanto que continuando con las pruebas durante más tiempo, la diferencia entre las coordenadas obtenidas y las reales fuera cada vez más pequeña.

En general, se puede decir que se ha conseguido que las mediciones y los cálculos sean bastante precisos con la tecnología que se ha escogido y que con un sensor más preciso y eligiendo una época del año diferente a los equinoccios, el dispositivo puede localizar un objeto estacionario dentro de un perímetro similar en espacio a la península ibérica.

Con un sensor más sofisticado, se cree que los resultados podría evitar las desviaciones a causa de días nublados u orientaciones del dispositivo.

5.3 Cálculo de Errores

Para analizar mejor los resultados desde el punto de vista de los errores, se han calculado los errores entre las coordenadas reales y las obtenidas por el prototipo de SGPS de distintas formas, el error absoluto en grados, en porcentaje y en kilómetros.

El error absoluto se ha calculado de forma independiente para longitud y latitud en grados, mediante la ecuación 10.

$$e = |coordenada\,real - coordenada\,experimental|$$

Ecuación 10.

Partiendo del error en grados para cada coordenada, se podrán calcular los errores en porcentaje mediante las ecuaciones 11 y 12 a continuación.

$$e_{longitud}\% = \frac{e_{longitud}}{360}100$$

Ecuación 11.

$$e_{latitud}\% = \frac{e_{latitud}}{180}100$$

Ecuación 12.

El error en kilómetros de longitud se puede hallar a través de la ecuación 13,

$$e_{longitud}\,km = e_{longitud}\;x\;111{,}12cos\varphi$$

Ecuación 13.

mientras que el error en kilómetros de latitud se obtendrá utilizando la fórmula 14.

$$e_{latitud}\,km = e_{latitud}\;x\;111{,}12$$

Ecuación 14.

Aplicando estos cálculos, se muestran en la tabla 13 todos los errores.

Tabla 13. *Errores de longitud y latitud.*

Día	Longitud Real (º)	Longitud Exp (º)	Latitud Real (º)	Latitud Exp (º)	$e_{longitud}$ (º)	$e_{latitud}$ (º)	$e_{longitud}$ (%)	$e_{latitud}$ (%)	$e_{longitud}$ (km)	$e_{latitud}$ (km)
2/9/15	-3,6	-6	40,15	41,43	2,4	1,28	0,67	0,71	-221,71	142,23
3/9/15	-3,6	-3,9	40,15	45,04	0,3	4,89	0,08	2,72	16,36	543,38
4/9/15	-3,6	-3,6	40,15	46,48	0	6,33	0	3,52	0	703,39
5/9/15	-3,6	-2,625	40,15	42,73	0,975	2,58	0,27	1,43	33,93	286,69
6/9/15	-3,6	-2,925	40,15	47,56	0,675	7,41	0,19	4,12	-67,99	823,4
8/9/15	-3,6	-5,475	40,15	48,55	1,875	8,4	0,52	4,67	-30,04	933,41
9/9/15	-3,6	-1,875	40,15	41,97	1,725	1,82	0,48	1,01	-81,91	202,24
10/9/15	-3,6	-3,675	40,15	57,61	0,075	17,46	0,02	9,7	4,06	1940,16
16/9/15	-3,6	1,05	40,15	19,36	4,65	20,79	1,29	11,55	450,84	2310,18
17/9/15	-3,6	-3,375	40,15	54,38	0,225	14,23	0,06	7,91	-14,07	1581,24
18/9/15	-3,6	-1,725	40,15	52,94	1,875	12,79	0,52	7,11	-186,03	1421,22
19/9/15	-3,6	-2,025	40,15	68,66	1,575	28,51	0,44	15,84	157,21	3168,03
20/9/15	-3,6	-1,875	40,15	71,95	1,725	31,8	0,48	17,67	-182,74	3533,62
21/9/15	-3,6	-2,325	40,15	72,3	1,275	32,15	0,35	17,86	-141,54	3572,51
22/9/15	-3,6	-0,9	40,15	50,82	2,7	10,67	0,75	5,93	255,07	1185,65
23/9/15	-3,6	-0,375	40,15	42,81	3,225	2,66	0,9	1,48	139,06	295,58
24/9/15	-3,6	-0,675	40,15	49,84	2,925	9,69	0,81	5,38	296,05	1076,75
25/9/15	-3,6	-0,825	40,15	52,71	2,775	12,56	0,77	6,98	-236,43	1395,67
26/9/15	-3,6	1,725	40,15	56,36	5,325	16,21	1,48	9,01	581,21	1801,26
27/9/15	-3,6	0,525	40,15	44,31	4,125	4,16	1,15	2,31	433,98	462,26
28/9/15	-3,6	0,375	40,15	36,77	3,975	3,38	1,1	1,88	264,38	375,59
29/9/15	-3,6	0,75	40,15	42,68	4,35	2,53	1,21	1,41	128,23	281,13
30/9/15	-3,6	-0,75	40,15	30,58	2,85	9,57	0,79	5,32	212,34	1063,42

Del cálculo de errores se deduce que hay que debe intentar ajustar la calibración al máximo pues unos solos grados de diferencia entre las coordenadas reales y las obtenidas significan muchos kilómetros de distancia.

Como se ha comentado anteriormente, si se presta atención tan sólo a la longitud, el mayor error en kilómetros apenas supera la superficie de la comunidad de Madrid. Sin embargo en cuanto a latitud encontramos varios miles de kilómetros de error en algunos casos como en los días próximos al equinoccio, pero en general se puede observar que el error no supera los mil kilómetros. Aun siendo éstos muchos kilómetros los resultados en porcentaje dan una visión más satisfactoria.

Si se presta atención al mencionado error en porcentaje, se observa que para la longitud apenas se supera el 1% de error y para la latitud, aunque es mayor, tampoco se puede hablar de grandes cifras siendo el mayor error de casi el 18%, y dado que el dato se registró el día del equinoccio no se puede decir que no sean resultados satisfactorios.

Una vez más se debe decir que este error es esperado tratándose de la época, pero gracias a la coordenada de longitud, a la tendencia que se observa pasado el equinoccio y atendiendo a las horas que el dispositivo tomó como horas de amanecer y horas de anochecer, se puede afirmar que el prototipo de SGPS podría localizar geológicamente un objeto estacionario con un error inferior al 1% en longitud y un error del 0% al 5% en épocas no cercanas a los equinoccios.

6. CONCLUSIONES Y FUTUROS TRABAJOS

Como conclusión al trabajo realizado se puede decir que los objetivos se han cumplido con éxito: se ha creado un prototipo basado en el modelo celestial que es capaz de proporcionar las coordenadas de un objeto exterior estacionario, tan sólo dándole la información de fecha y hora al iniciarlo. Además, se ha construido con el menor presupuesto posible, y como se ha demostrado, esto no ha afectado significativamente a la precisión.

Sin embargo, sí que se ha observado que el prototipo presenta dificultades en cuanto a la obtención de la coordenada de latitud, durante los equinoccios y días cercanos. Así mismo, los días nublados también le hacen perder precisión.

Esto lleva a plantear como futuras vías de estudio, el análisis detallado de estos momentos tratando de encontrar un patrón de error en estos días y así intentar salvar el error. También sería interesante probar el dispositivo con un sensor más sofisticado que sea capaz de identificar espectros de colores. De esta manera, el dispositivo podría intuir el amanecer y anochecer cuando los colores sean más rojizos y así, por ejemplo, ahorrar energía de la batería y eliminar el error en días nublados.

Como primer prototipo, se han logrado los objetivos, pero como se ha mencionado, se puede mejorar y abre la puerta a futuros trabajos, sirviendo como base sólida para llevar la tecnología de SGPS al mundo real.

ANEXO

A. Tutorial básico entorno Arduino

I. Primeros pasos en entorno Arduino

El entorno consta de un editor de texto en el que se escribe el código, una ventana de mensajes en la parte inferior, un monitor serie en el que se puede visualizar resultados del programa mediante la función *Serial.print()*, una barra de herramientas con botones para acciones comunes como descarga a la placa, verificación del código o puesta en marcha.

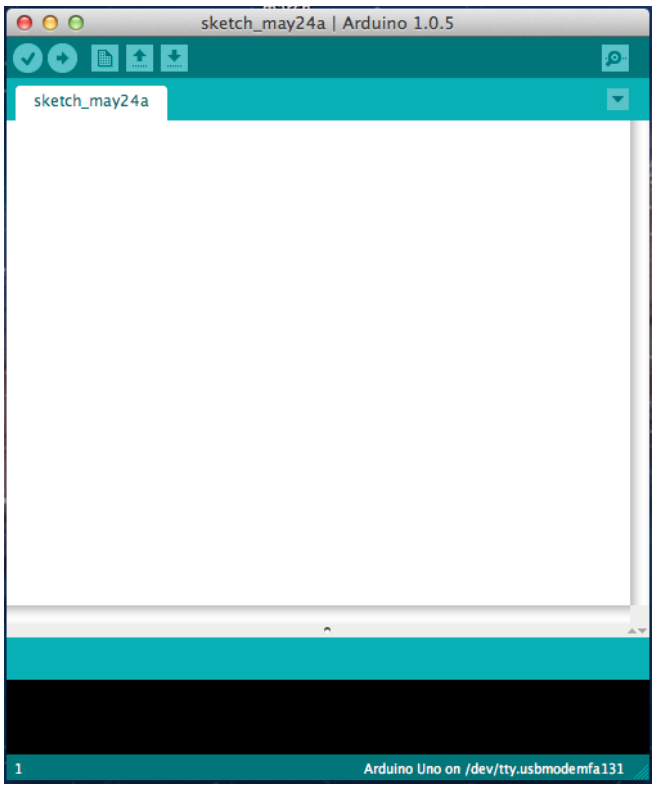

Figura 40. *Datos de luz tomados durante el día 13 de mayo del 2015.*

Lo primero que hay que hacer para comunicarse con la placa desde el entorno es seleccionar la placa que vamos a usar (Figura XX) desde el Menú>Herramientas.

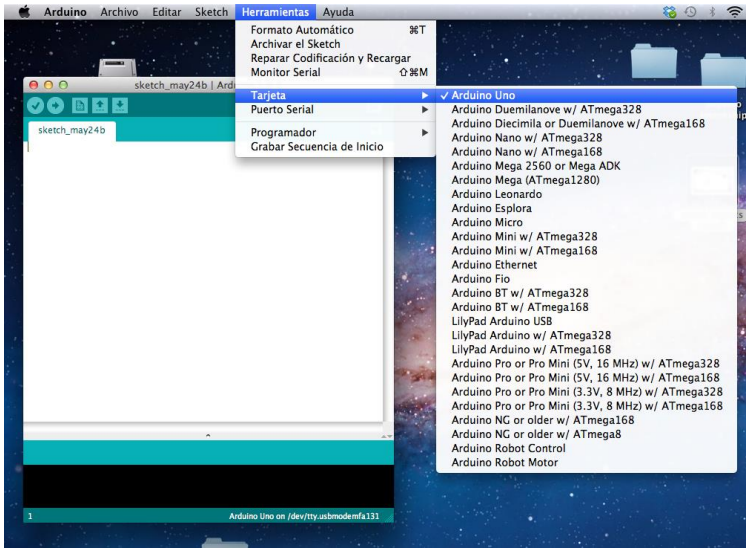

Figura 41. *Datos de luz tomados durante el día 13 de mayo del 2015.*

Una vez seleccionada la placa hemos de seleccionar el puerto serie en el que se encuentra conectada la placa para comenzar a comunicarte. Este puerto va a variar dependiendo del sistema operativo que estés utilizando, en este caso se está utilizando MAC OS X y el puerto es /dev/tty.usbmodemfa131 como se puede ver en la Figura XX.

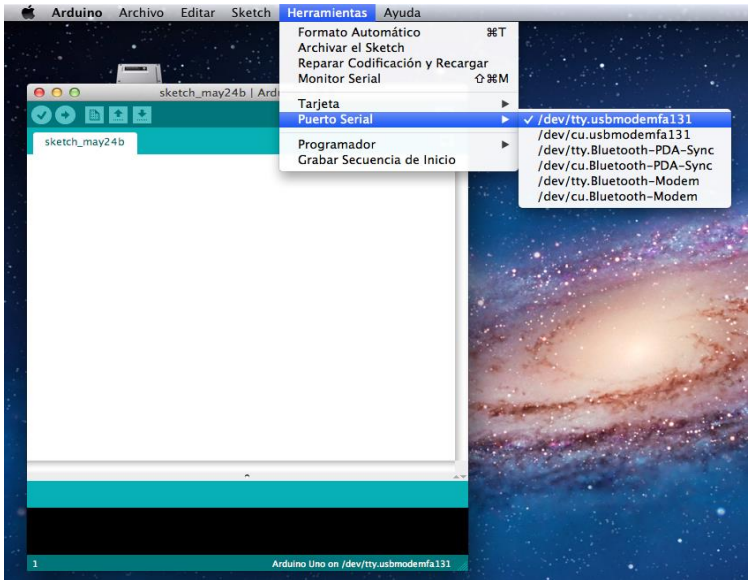

Figura 42. *Datos de luz tomados durante el día 13 de mayo del 2015.*

Una vez seleccionadas estas opciones es momento de comenzar a programar en el sketch. Se puede encontrar ayuda como manuales y ejemplos en la página web de Arduino. También se pueden encontrar librerías para descargar y agilizar la programación. Una vez que se han descargado, deben introducirse en el entorno de programación, en la carpeta destinada para tal efecto, además de introducirlas en el hardware de Arduino. Todo este proceso se encuentra explicado en el apartado de librerías de la página web[7].

[7] *http://arduino.cc/en/Guide/Libraries*

B. Programa de Calibración del SGPS en Arduino

A continuación se pasa a desglosar el código realizado para la calibración del SGPS, y las funciones y librerías utilizadas en dicho programa.

I. Librerías usadas en el programa de calibración.

Las librerías utilizadas en el programa creado para la calibración son las que se citan a continuación.

- *Wire.h*

Esta librería viene ya incorporada en el entorno Arduino. Es ésta la que permite la comunicación mediante el protocolo I2C. Permite el envío y la recepción de datos entre dos dispositivos o sensores conectados mediante I2C/ TWI (Two Wire Interface), es decir, en este caso, mediante el RTC y Arduino.

- *SD.h*

Al igual que la anterior, ésta viene ya incorporada en Arduino. Permite la lectura y escritura en la tarjeta SD, que se ha incorporado en el dispositivo, mediante comunicación SPI.

- *SPI.h*

Usada para comunicaciones entre dispositivos usando el Bus SPI. Es un protocolo de datos serie síncrono usado por los microcontroladores para la comunicación con uno o más dispositivos periféricos rápido y a corta distancia. También se puede usar para comunicaciones entre dos microcontroladores. En una conexión SPI siempre existe un dispositivo Máster, normalmente el microcontrolador, el cuál controla dispositivos periféricos o esclavos.

- *RTClib.h*

Esta librería no viene incluida en Arduino, luego es necesario descargarla e instalarla como se explica en el Anexo 1. Fue creada por JeeLab y es gratuita y fácilmente descargable. La última versión esta mantenida en GitHub [8], desde donde se podrá descargar, obtener información de funciones y proyectos de ejemplo.

Es una librería que permite obtener y ajustar la hora desde el chip DS1307. En este trabajo se usó para poner en hora el RTC ajustándolo a la hora que marcaba el ordenador durante la carga del código, y obtener la hora del RTC cuando fuere necesario.

Todas las librerías incluidas en Arduino vienen detallas en la página web[9], donde se podrán consultar las funciones que incluyen así como ejemplos y discusiones abiertas.

[8] *https://github.com/jcw/rtclib*
[9] *http://arduino.cc/en/Reference/Libraries*

II. Funciones Creadas en el Programa Principal

Las funciones que aquí se detallan se encuentran declaradas al inicio del programa e implementadas al final del mismo. Estas funciones se detallan a continuación.

- *int bisiesto(int anno)*

Esta función analiza si un año en concreto es bisiesto o no. Es necesaria ya que en el programa los días del año se tratan en números de 1 a 365 o 366 dependiendo de si el año es bisiesto. Si el año es bisiesto la función devolverá un 1, si por el contrario el año introducido no es bisiesto, devolverá un 0.

Tan sólo tiene un argumento, *anno,* el cual se obtiene del RTC. Aquí será sometido a una serie de condiciones que determinarán si el año es bisiesto o no.

- *int días_de_mes(int m, int anno)*

Es la función encargada de devolver el número de días que tiene el mes que está proporcionando el RTC, es decir, el mes en el que se están tomando medidas. Dentro de esta función se hace uso de la anterior función, *int bisiesto(int ano)*.

Tiene dos argumentos, éstos son; m, el mes en el que se están tomando mediciones, y, anno, el año, ambos extraídos en el programa principal del RTC.

III. Funciones Contenidas en las Librerías

Éstas son las funciones contenidas en las librerías anteriormente mostradas que se usan el programa, en orden según aparecen en el programa:

- *Wire.begin()*

Inicia la librería Wire e incorpora el bus I2C como máster o esclavo. Normalmente solo es llamada una vez y así ha sido aquí.

- *RTC.begin()*

Inicia la librería RTC. Sólo es necesario llamar esta función una vez durante el programa.

- *RTC.adjust(DateTime(__DATA__,__TIME__))*

Esta función toma la fecha y la hora acorde a la del ordenador que se está usando, justo en el momento en que se compila el código, y la usa para programar el RTC. Si el ordenador no está bien configurado la hora no se establecerá correctamente, luego es importante fijar la hora correctamente antes de compilar el programa.

- *SD.begin(CS))*

Inicializa la librería SD y la tarjeta. Ésta comienza el uso del bus SPI y el chip select pin, el chip de selección de pin, que por defecto es el pin SS del hardware, pin 10 para Arduino UNO. Es importante que incluso si se usa otro chip select pin, el pin SS del hardware debe ser mantenido como salida o las funciones de la librería no funcionarán.

- *RTC.now()*

Esta es la manera de obtener el tiempo de la RTClib, la función devuelve un objeto DateTime que describe el año, mes, día, hora, minutos y segundos.

- *now.year(), now.month(), now.day(), now.hour(), now.month(), now.day(), now.hour(), now.minute()*

Como se ha dicho en la anterior, la función *now()* se usa para obtener la fecha y hora, pero si lo que se pretende es obtener cada una de ellas por separado, se habrán de llamar por separado. Así estas son las funciones que se usan para obtener cada una de ellas.

- *SD.open("log.txt", FILE_WRITE)*

Abre un archivo para escritura en la SD con el nombre log.txt. Si el archivo no existe, la función lo creará, pero si por el contrario éste ya existe, tan solo lo abrirá. El directorio que contiene este archivo sí debe existir previamente.

- *dataFile.print(data)*

Imprime los datos especificados en el archivo que ha abierto previamente mediante la función *SD.open()*. En este programa se imprime la hora en formato decimal y el valor medio de tensión de cada tres muestras tomadas del LDR.

- *dataFile.close()*

Cierra el archivo de texto creado en la SD y se asegura que cualquier data que haya sido escrito en la SD está físicamente guardado en la tarjeta SD.

IV. Funciones de Arduino

El lenguaje Arduino consta de una serie de funciones básicas para crear programas. En el programa de calibración se ha hecho uso de las siguientes:

- *void setup()*

La función es llamada cuando cuando un sketch empieza. Se utiliza para inicializar las variables, los pin modes, las inicializaciones de librerías, etc. Esta función solo se ejecutará una vez, después de cada puesta en marcha o reseteo. Debe ser incluida siempre en los programas de Arduino, aunque no tenga ningún contenido.

- *Serial.begin(9600)*

Inicializa el puerto serie, y establece la velocidad en baudios de transmisión de datos vía puerto serie. La velocidad es la que se especifica entre paréntesis, 9600 baudios en este caso.

- *Serial.println()*

Imprime los datos o un mensaje de texto, entre comillas, especificados en el paréntesis por el puerto serie, los cuáles pueden ser vistos en el monitor puerto serie que incorpora el entorno. Incorpora un salto de línea de modo que lo próximo que se escriba será en la línea siguiente.

- *pinMode(CS, OUTPUT)*

Configura un pin específico para comportarse, ya sea como entrada o salida. En este caso está configurando el pin CS, como salida.

- *void.loop()*

La función loop() hace precisamente lo que su nombre sugiere, un bucle del progama principal, es imprescindible y es donde permite a tu programa cambiar y responder. Se usa para activar el control de la placa Arduino.

- *analogRead(sensorReading)*

Esta función lee un valor de un determinado pin analógico, especificado entre paréntesis. Este pin en este proyecto es el pin A0, de donde obtendremos el valor de tensión del sensor.

V. Código de la Calibración

El código desarrollado para llevar a cabo la calibración del SGPS se ha subido a la plataforma GitHub, en la que se ha creado un repositorio con el nombre Arduino_SGPS.

GitHub es una plataforma de desarrollo colaborativo de software para alojar proyectos utilizando el software de control de versiones Git. Esto significa que se ha publicado este proyecto de forma pública de manera que cualquier persona pueda acceder a él de manera gratuita y fácilmente, tanto para bajárselo o para modificarlo o aportar nuevas ideas.

Para descargarlo tan sólo es necesario ir a la página de GitHub[10] e introducir en el buscador el nombre del repositorio, Arduino_SGPS, como se muestra en la figura a continuación.

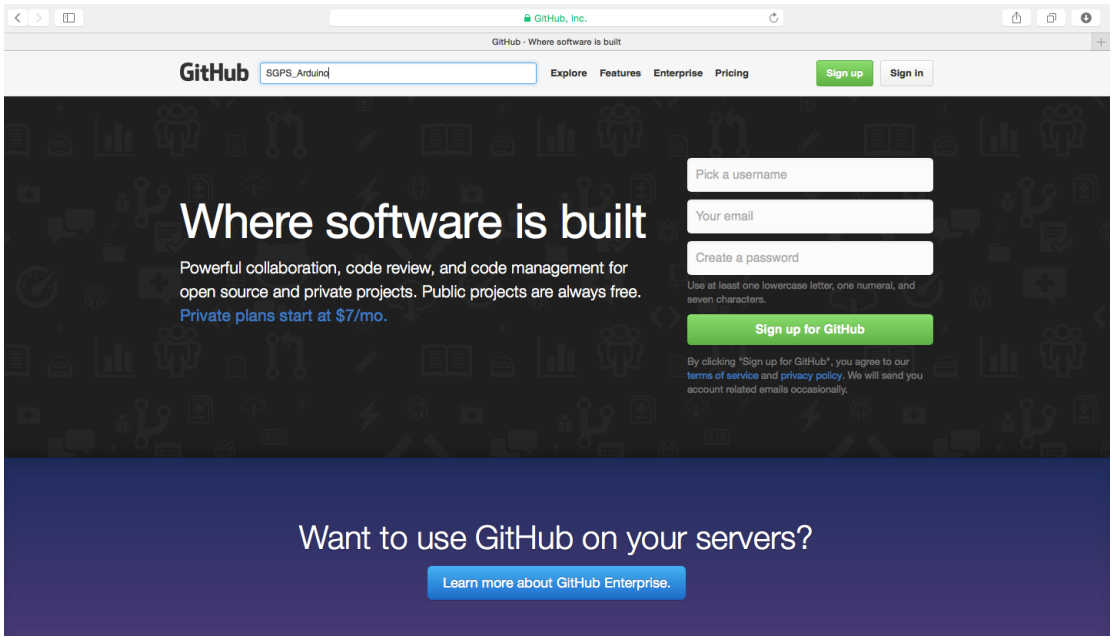

Figura 43. *Página de inicio de GitHub.*

[10] *https://github.com*

En los resultados de la búsqueda se puede observar el repositorio, figura 40.

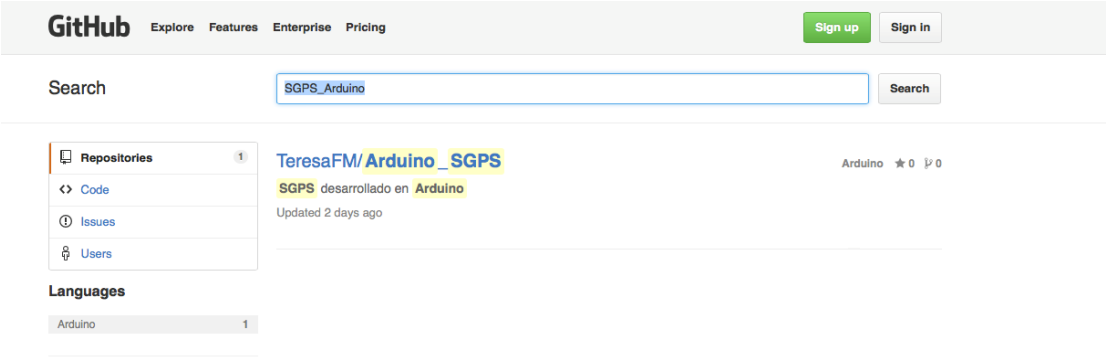

Figura 44. *Resultados de la búsqueda SGPS_Arduino en GitHub.*

Se pincha sobre TeresaFM/Arduino_SGPS y esto lleva a la carpeta donde se encuentra el proyecto completo, el código en Arduino de la calibración, el código final del SGPS, y una carpeta con todos los archivos de texto en formato genérico con datos de luz y hora para diferentes días, tomados durante la realización del presente proyecto.

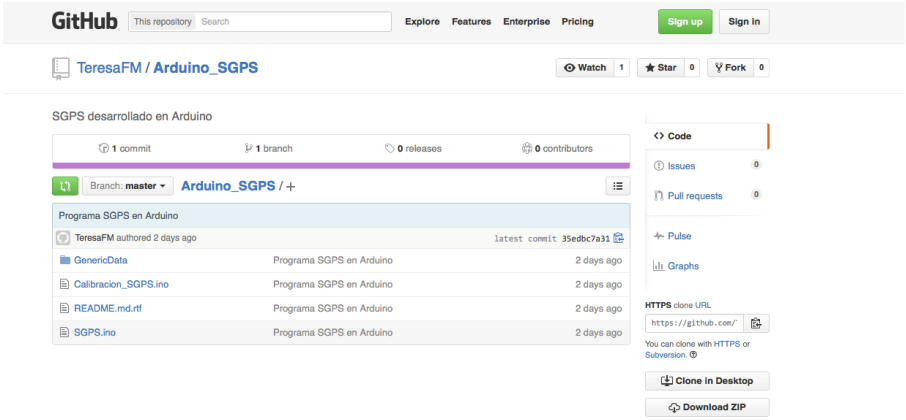

Figura 45. *Carpetas del repositorio o proyecto SGPS_Arduino.*

Si no se desean seguir todos estos pasos, es posible acceder directamente a la carpeta del proyecto siguiendo esta dirección https://github.com/TeresaFM/Arduino_SGPS.git que lleva directamente a la pantalla que se muestra arriba, en la figura 41.

Para descargar la carpeta del proyecto, es tan sencillo como pulsar en el botón Download ZIP, situado en la parte baja a la derecha como se muestra en la figura 42 recuadrado en rojo. De esta manera se descargará una carpeta con el nombre del proyecto en la carpeta de descargas del ordenador desde donde se esté realizando.

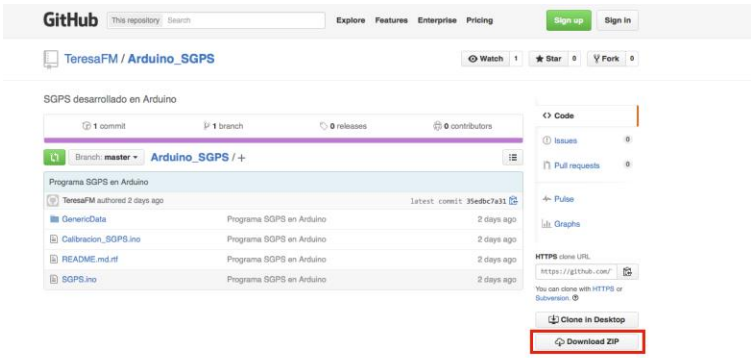

Figura 46. *Muestra del botón Download ZIP*

En caso de ya disponer de una cuenta en la plataforma GitHub, y del software para escritorio de GitHub, es posible clonarse la carpeta del proyecto en nuestro escritorio pulsando el botón Clone to Desktop, situado justo encima de Download ZIP.

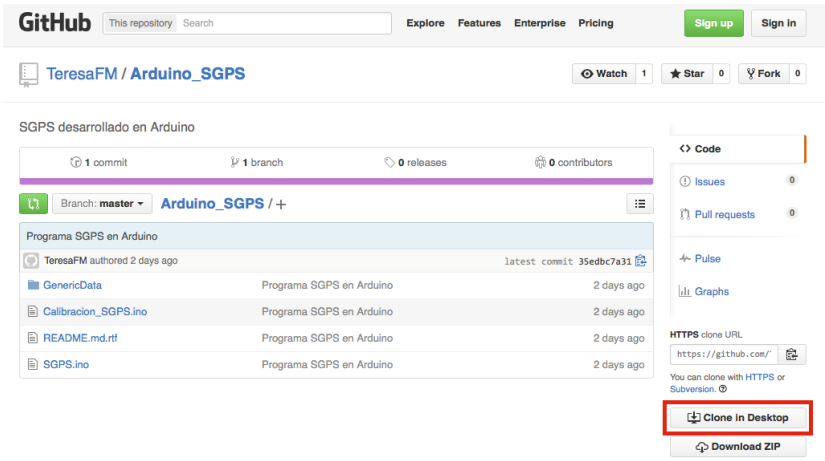

Figura 47. *Muestra del botón Clone in Desktop.*

Una vez se tiene esta carpeta en el ordenador, si se tiene el software de Arduino instalado, los programas los reconocerá como archivos de Arduino y se abrirán directamente en este entorno.

C. Programa SGPS en Arduino

En este apartado se desglosa el código programado para el funcionamiento del dispositivo SGPS haciendo una descripción de las librerías que se han utilizado para programarlo.

Algunas de estas librerías ya se han descrito en el apartado anterior, apartado 2 de anexos, de modo que no se volverán a explicar sino que se indicará en qué apartado se hizo.

I. Librerías

- *SD.h, RTClib.h, SPI.h*

Detalladas en el apartado 2.1 del anexo.

- *Math.h*

Esta librería incluye un gran número de funciones matemáticas muy útiles para el uso de números en coma flotante (float). El listado completo de funciones de esta librería se puede encontrar en la web[11], desde donde también se podrá descargar gratuitamente.

II. Funciones Creadas en el Programa Principal

Para el programa SGPS definitivo se crearon funciones para desarrollar el modelo celestial en el que se basa su funcionamiento. Para esto, se ha creado una función para cada ecuación del modelo. Estas funciones son las que se listan a continuación además de otras que fueron necesarias.

- *float anioFrac(const unsigned int ndias)*

Se calcula la ecuación 4 para obtener β.

- *float EquationOfTime(const float beta)*

Se calcula EqT según ecuación 11.

- *float cLongitud(const float mDay)*

Se resuelve la ecuación 5 y devuelve el valor de la coordenada de longitud, λ.

- *float cLatitud(const float delt, const float asunset)*

Resuelve la ecuación 6 para obtener el la coordenada de latitud, φ.

- *float sunDeclination(const float beta)*

Se obtiene la declinación solar, δ, según ecuación 3.

- *float angularSunset(const float sunset, const float mDay)*

Devuelve la puesta de sol angular, α_{sunset}, haciendo los cálculos de la ecuación 2.

- *int bisiesto(int anno), int dias_de_mes(int m, int anno)*

[11] *http://www.nongnu.org/avr-libc/user-manual/group__avr__math.html*

Estas dos librerías están explicadas en el apartado 2.2 del anexo.

III. Funciones Contenidas en las Librerías

Todas las descritas en el apartado 2.3 de anexo.

IV. Funciones de Arduino

Las mismas que se explican y detallan en el apartado 2.4 del anexo.

V. Código del SGPS

El código del SGPS en Arduino se encuentra en la plataforma GitHub, y se podrá obtener operando de la misma manera que para obtener el código de la calibración, como se explica en el apartado V del anexo B.

PRESUPUESTO

En el siguiente anexo se detallará el coste total del presente proyecto, calculado y detallado cada coste por separado y finalmente el coste total.

VI. Costes de Personal

En el coste de personal se diferencia el tipo de personal y las fases del proyecto. Como personal se tiene en cuenta al director de proyecto por un lado y al ingeniero de proyectos por otro ya que existe una gran diferencia de salario entre ambos.

Las fases del proyecto serán:

- Planificación. Incluye la familiarización con Arduino y su entorno, así como de la plataforma de SGPS, el estudio del modelo celestial y el diseño del prototipo de SGPS.
- Desarrollo. Esta fase incluye la construcción del prototipo, el desarrollo del software para la calibración del SGPS, y el desarrollo del programa del SGPS. Además incluye el análisis de resultados e investigación requerida para corregir errores.
- Documentación. Incluye todo el tiempo requerido para la redacción de la memoria del proyecto y correcciones.

Tabla 14. *Costes de personal desglosado*

		Director de proyecto	Ingeniero de proyecto
Planificación	Familiarización (h)	0	20
	Comprensión del modelo teórico (h)	0	20
	Diseño del prototipo (h)	6	16
Desarrollo	Montaje del prototipo (h)	1	4
	Desarrollo programa de calibración (h)	5	100
	Desarrollo programa de SGPS (h)	10	200
	Análisis de datos e investigación (h)	5	100
Documentación	Redacción (h)	0	70
	Corrección (h)	10	20
Total	Total (h)	37	550
	Salario/h	50	30
	Salario total (€)	1850	16500
	Total (€)	**18350**	

VII. Costes de Material

Como costes materiales sólo se ha incluido el coste de los componentes requeridos para construir el prototipo, ya que esto es lo más interesante para la valoración del cumplimiento de uno de los objetivos del proyecto, construir un prototipo por un coste aproximado de 50 euros.

El coste de materiales se desglosa a continuación en la tabla 15.

Tabla 15. *Costes de material desglosados.*

			Coste (€)
Componentes	Microprocesador	Arduino UNO	24.20
	Reloj digital	Mmsmart RTC	14.40
	Dispositivo para tarjeta de memoria SD	SD Card Shield Seeed studio	6.90
	Tarjeta de memoria SD	Tarjeta SD $500MB$	2.50
	Sensor intensidad de luz	KE-10720 LDR	3.00
	Cable	$1m$ cable unipolar $76mm^2$	0.20
	Batería	Pila Alcalina $9V$ Panasonic	3.22
	Resistencia	Resistencia $1K\Omega$	0.25
	Cable de pila		1.00
Total (€)			**55.67**

Aunque el precio total del prototipo supera los 50€, es importante recalcar que un dispositivo como éste fabricado en grandes lotes, rebajaría el precio considerablemente, haciéndolo un artículo más que asequible.

VIII. Coste Total del Proyecto

El coste total se obtendrá de la suma del total de los costes de los apartados anteriores añadiéndole el IVA (veintiuno por ciento). De este modo el coste total del proyecto asciende a ***veintidós mil doscientos setenta euros con ochenta y seis céntimos.***

Tabla 16. *Coste total del proyecto desglosado.*

Concepto	Coste (€)
Personal	18350
Material	55.67
Subtotal	18405.67
IVA (21%)	3865.19
Total	**22270.86**

REFERENCIAS

[1] B. Hofmann-Wellenhof, H. Lichtenegger, J. Collins, Global Positioning System: Theory and Practice, 5 ed., Springer Verlag/Wien, 2001.

[2] R. D. Hill, M. J. Braun, Geolocation by light level. The next step: Lat- itude, 2001, pp. 315–330.

[3] J.-B. Thiebot, D. Pinaud, Quantitative method to estimate species habitat use from light-based geolocation data, Endanger. Species Res. 10 (2010) 341–353.

[4] Jose Pardeiro, Javier V. Gómez, Alberto Brunete, Frode Eika Sadnes: Evolutionary Optimization Algorithms for Sunlight-based Positioning Sensor Network, en: International Journal of Distributed sensor Networks, 17 Junio 2014.

[5] J. V. Gómez, F. E. Sandnes, B. Fernández, Sunlight Intensity Based Global Positioning System for Near-Surface Underwater Sensors, Sensors 12 (2012) 1930–1949.

[6] ANSI. American National Standard Photographic Exposure Guide; ANSI PH2.7-1973; American National Standards Institute: New York, NY, USA, 1973.

[7] ANSI. ANSI PH2.7-1986. American National Standard for Photography—Photographic Exposure Guide. American National Standards Institute: New York, NY, USA, 1986.

[8] Ray, S.F.; Jacobson, R.E.; Atteridge, G.G.; Axford, N.R. The Manual of Photography: Photographic and Digital Imaging; Focal Press: Waltham, MA, USA, 2000.

[9] Polastre, J.; Szewczyk, R.; Sharp, C.; Culler, D. The Mote Revolution: Low Power Wireless Sensor Networks. Proceedings of the 16th Symposium on High Performance Chips (HotChips), San Francisco, CA, USA, 22–24 August 2004.

[10] Meeus, J. Astronomical Algorithms, 2nd ed.; Willmann-Bell, Inc.: Richmond, VA, USA, 1999.

[11] Spencer, J.W. Fourier series representation of the position of the sun. Search 1971, 2, 172.

[12] National Oceanic and Atmospheric Administration (NOAA). National Oceanic and Atmospheric Administration Solar Calculations Spreadsheet; NOAA: Silver Spring, MD, USA, 2011.

[13] Isaac Rivero Guisado. Desarrollo de librería open-source para el proyecto SGPS, Universidad Carlos III de Madrid, *http://javiervgomez.com/pages/sunlight-intensity-based-global-positioning-system-library.html*

[14] Jose Pardeiro, Javier V. Gómez, Alberto Brunete, Frode Eika Sandnes. Evolutionary Optimization Algorithms for Sunlight-based Positioning Sensor Networks, International Journal of Distributed Sensor Networks, 17 Junio, 2014