Trabajo Fin de Grado

Sistema de seguridad para el hogar

Home security system

Autor

David Escosa Galve

Director

Fernando Tricas Lamana

Ponente

José Luis Briz Velasco

Escuela de Ingeniería y Arquitectura

2016

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe acompañar al Trabajo Fin de Grado (TFG)/Trabajo Fin de Máster (TFM) cuando sea depositado para su evaluación).

D./Dª. David Escosa Galve,

con nº de DNI 76973510-P en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza,

Declaro que el presente Trabajo de Fin de (Grado/Máster) Grado en Ingeniería Informática, (Título del Trabajo) Sistema de seguridad para el hogar,

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, 18 de Enero de 2017

Fdo: David Escosa Galve

Sistema de seguridad para el hogar

Resumen

En este proyecto se han desarrollado dos prototipos de un sistema de seguridad para el hogar autoinstalables por el usuario. Ambos almacenan la información en un servidor con acceso a través de la web para la visualización de los datos mediante consultas y gráficas.

El trabajo desarrollado en el primer prototipo se ha centrado en la validación y puesta a punto de cada uno de los sensores mientras que con el segundo se ha buscado minimizar todo el conjunto, tanto *hardware* como *software*, sin perder ninguna funcionalidad y buscando un producto más próximo al mercado.

Para ello se hizo un estudio de los principales productos del sector para conocer las funcionalidades ofrecidas. Seguidamente, se plantearon los requisitos y se buscaron los componentes necesarios para cumplirlos. Una vez adquiridos cada uno de ellos, se buscó la información técnica necesaria bien sobre librerías disponibles o bien para desarrollarlas validando así cada componente de forma unitaria. Por último, se sometió el sistema a pruebas globales para comprobar que actúa como se especificó.

Tanto para la validación de cada componente como para la del montaje conjunto hubieron de resolverse problemas tanto de hardware como de *software*.

El resultado ha sido un sistema capaz de registrar la información procedente de los sensores, que permite monitorizar la actividad doméstica, enviando la información a un sistema alojado en un servidor web –desarrollado en Pariver y adaptado a este caso como complemento del proyecto- que permite su consulta mediante un navegador, y que envía mensajes a través del correo o el teléfono móvil.

Índice de contenido

Índice de figuras

Índice de tablas

1. Introducción

La seguridad en el hogar es un tema de gran preocupación social. Por ello han ido surgiendo en el mercado nuevos sistemas que buscan dar cobertura a esa preocupación a través de diferentes dispositivos y funcionalidades. Entre ellos están el Canary, producto de Canary Connect [1], o los diferentes productos de la empresa Nest [2]. Estos nuevos sistemas se diferencian de los clásicos sistemas de seguridad en el modo de instalación (preparados para una autoinstalación y configuración), en el soporte directo al usuario desde la web con o sin una aplicación en dispositivo móvil, y la ausencia de conexión a un sistema de central de alarmas y de su correspondiente cuota periódica en el servicio básico (requerido en función de los servicios en *cloud* con los que se desee contar).

En esa línea, PARIVER S.A., empresa en la que he realizado prácticas durante el Grado, se plantea la posibilidad de desarrollar un dispositivo similar y me propone realizar un prototipo como Trabajo de Fin de Grado (TFG).

1.1. Objetivo y alcance del proyecto

El objetivo del proyecto es desarrollar un dispositivo de seguridad doméstica. En él se deben integrar una serie de sensores gestionados por un procesador en un sistema encapsulado con la finalidad de dar cobertura a las funcionalidades básicas de seguridad en el hogar. Los datos obtenidos por el sistema serán accesibles a través de un sistema de tipo servidor accesible mediante un navegador web o dispositivo móvil, cuyo desarrollo queda al margen de este proyecto.

1.2. Contexto de desarrollo

Pariver se fundó en 1985 como una empresa pionera en España en el desarrollo de servicios TIC (Tecnologías de la Información y Comunicación) y consultoría para empresas e instituciones. En 2010 el Gobierno de Aragón aprobó el proyecto IDAS Blackbox con el que se desarrolló un sistema automatizado de adquisición de datos, en el que se apoya una parte de este TFG.

1.3. Métodos y técnicas

Algunos de los sensores que se han utilizado, y que fueron elegidos por disponibilidad, coste y rangos operativos, no disponían de documentación técnica suficiente, o bien aparecían en instalaciones que modificaban las condiciones de funcionamiento recogidas en el *datasheet*, o se distribuían en *shields* deficientemente o insuficientemente implementados. Para estudiar su comportamiento y conseguir un funcionamiento correcto se utilizó un polímetro y un osciloscopio, sometiéndolos a pruebas específicas. Adicionalmente, hubo que diseñar y construir interfaces electrónicos específicos para algunos de ellos La descripción de los problemas, pruebas e interfaces se detalla más adelante en las secciones correspondientes.

Para la programación del sistema se utilizó el entorno de programación nativo de Arduino [3]. Se aprovecharon las librerías que éste aporta, se implementaron las que no existían y eran requeridas para el funcionamiento del sistema y se adaptaron a las necesidades del proyecto algunas otras, todo ello en lenguaje C++. También se utilizó KiCad [4] para elaborar un esquema del montaje correcto de todos los componentes.

1.4. Planificación

En la *Figura 1* se recoge el diagrama temporal, con granularidad semanal, que se estimó para llevar a cabo el proyecto. Se preveía una duración total de 41 semanas.

Sin embargo, surgieron dificultades que alargaron el tiempo previsto para algunas de las tareas, e implicaron la adición de otras nuevas. En la *Figura 2* se recogen las tareas desarrolladas finalmente con su dedicación real. La duración total del proyecto ha sido de 48 semanas.

A continuación se explican los motivos por los que el tiempo para abordar las diferentes partes del proyecto no fue el previsto.

En primer lugar hubo que buscar alternativas a algunos de los dispositivos inicialmente seleccionados, sea por necesidades funcionales o por problemas técnicos. En un último momento también se decidió sustituir la cámara para permitir reproducir vídeo en *streaming* (una funcionalidad no prevista al comienzo del desarrollo del TFG).

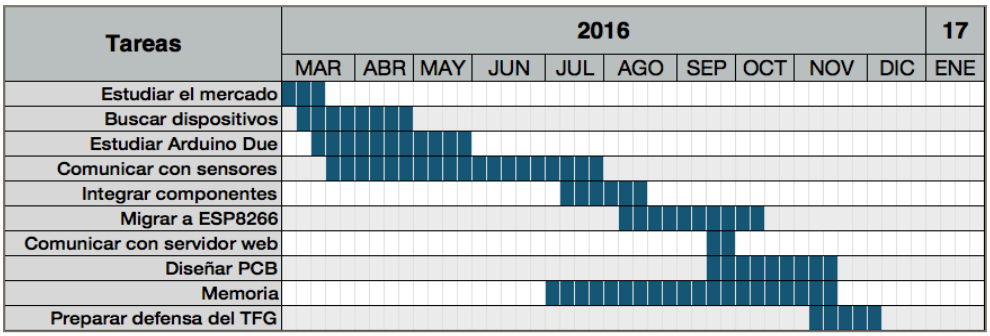
En segundo lugar, el estudio del microcontrolador Arduino Due se extendió significativamente. El motivo fue tener que profundizar en su funcionamiento interno para resolver requisitos muy estrictos marcados por algunos dispositivos. Además fue necesario para comprender por qué se estaban produciendo condiciones de carrera que provocaban un mal funcionamiento del sistema.

Cabe destacar la decisión tomada a mitad de proyecto de preparar un prototipo totalmente funcional con los elementos de *hardware* y *software* disponibles en ese momento. De este modo se conseguía validar todos los dispositivos y disponer de un prototipo que cumplía todas las funcionalidades que se planteaban al comienzo del TFG. La dificultad que apareció con la comunicación y control de algunos de los sensores hizo interesante esta opción que permitía tener un primer prototipo válido (aunque no definitivo) que ayudara en la decisión final (o descarte) de algunos sensores.

Esto pospuso el desarrollo de la placa de circuito impreso para el producto definitivo. Aunque ya se había preparado su esquema en KiCad, hubo que modificarlo con posterioridad en función del resultado del primer prototipo.

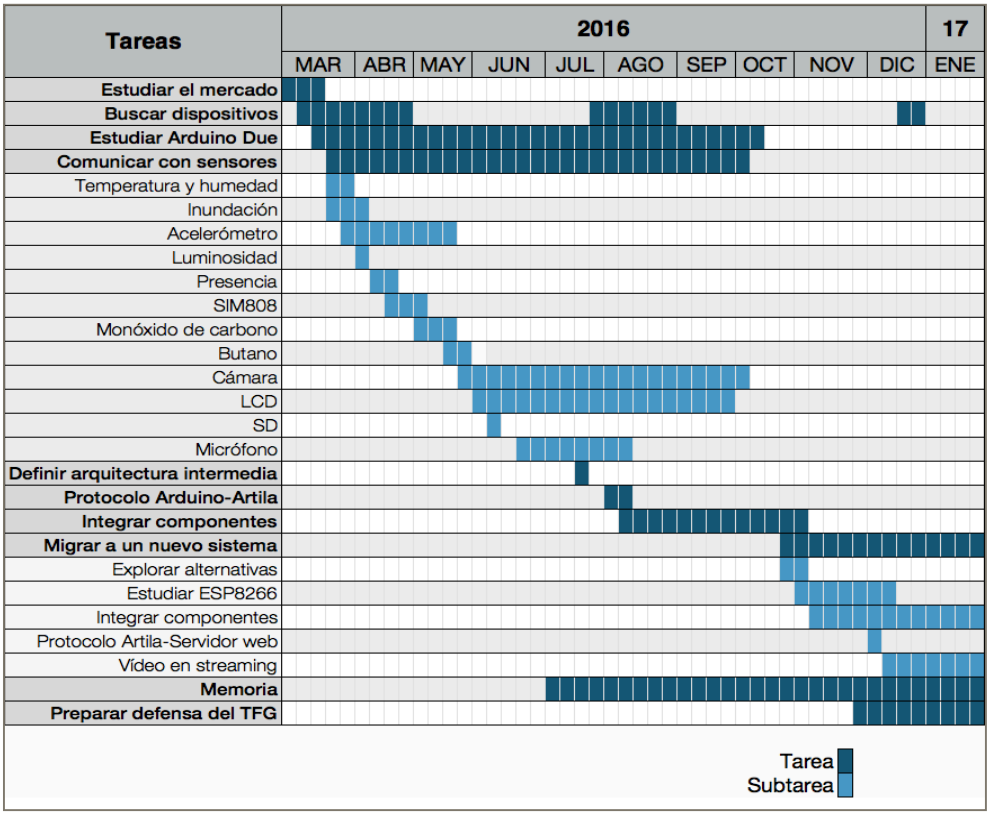
Finalmente, surgieron algunos problemas en diciembre que me impidieron presentar el TFG. Esto me permitió incorporar trabajos que se habían estado continuando y que han derivado en un nuevo prototipo completamente funcional basado en el ESP8266 [5] y una arquitectura más simplificada.

Figura 1. Planificación temporal inicial



*La figura ha sido adaptada a las dimensiones de la *Figura 2* para facilitar su comparación.

Fuente: Elaboración propia. Figura 2. Coste real de las actividades del TFG



Fuente: Elaboración propia.

Tabla 1. Dispositivos que implementan los requisitos.

Requisitos	Dispositivos
Conocer la calidad del aire	Sensor de temperatura y humedad Sensor de gas butano Sensor de CO
Conocer si hay actividad en casa	Sensor de luminosidad Sensor de movimiento Cámara de vídeo Micrófono
Conocer si hay fugas de agua	Sensor de inundación
Asegurar que no están manipulando el sistema	Acelerómetro
Poder recibir alertas en todo momento	SIM808
Poder consultar información o modificar algún parámetro en el propio sistema	Pantalla táctil
Disponer de una fuente de fecha y hora en tiempo real	RTC

Fuente: Elaboración propia.

2. Análisis

En las siguientes secciones se señalan los requisitos funcionales del sistema, se proporcionan más datos sobre algunas de las soluciones actuales del mercado y la solución que se propone para desarrollar en el ámbito del TFG.

2.1. Análisis de requisitos

En la *Tabla 1* se detallan los requisitos que el sistema debe satisfacer, de acuerdo a los objetivos que se plantearon al comienzo del TFG, y los sensores/dispositivos necesarios para el cumplimiento de cada uno. Aunque se relaciona cada requisito con una serie de dispositivos, no es una relación estricta ya que en algunos casos un dispositivo puede servir para cumplir o complementar varios de ellos.

2.2. Análisis de soluciones

En este capítulo se exponen algunas de las soluciones existentes en el mercado, la propuesta que se desea desarrollar en el TFG así como las diferentes fases en las que se abordó.

2.2.1. Soluciones existentes

Se analizaron dos soluciones presentes en el mercado: Canary y Nest.

El sistema Canary cuenta con una cámara HD con visión nocturna, una sirena, un sensor de temperatura y humedad y detectores de gases. Ofrece una aplicación

para *smartphone* para consultar los valores de los sensores y para visualizar video en *streaming*. Podemos adquirirlo por un precio de 199$.

Nest proporciona diferentes productos independientes para cubrir cada una de las necesidades. Dispone de cámara para exterior e interior, ambas por 199$ cada una, detector de humo y CO por 99$, y de un termostato para el control del sistema de calefacción a un precio de 249$. Por supuesto, todos ellos disponen de una aplicación para *smartphone* desde la que recibir alertas, visualizar la cámara y regular el termostato.

2.2.2. Solución propuesta

En la solución que se plantea se garantizan los requisitos de la *Tabla 1*, los cuales cubren todas las funcionalidades ofrecidas por las soluciones analizadas e incorporan algunas más como sensor de inundación o detección de manipulación del sistema. Otra de las ventajas de esta solución es la posibilidad de poder introducir una tarjeta SIM de telefonía para permitir recibir alertas en el teléfono móvil aun cuando no haya conexión a Internet en el dispositivo.

2.2.3. Fases

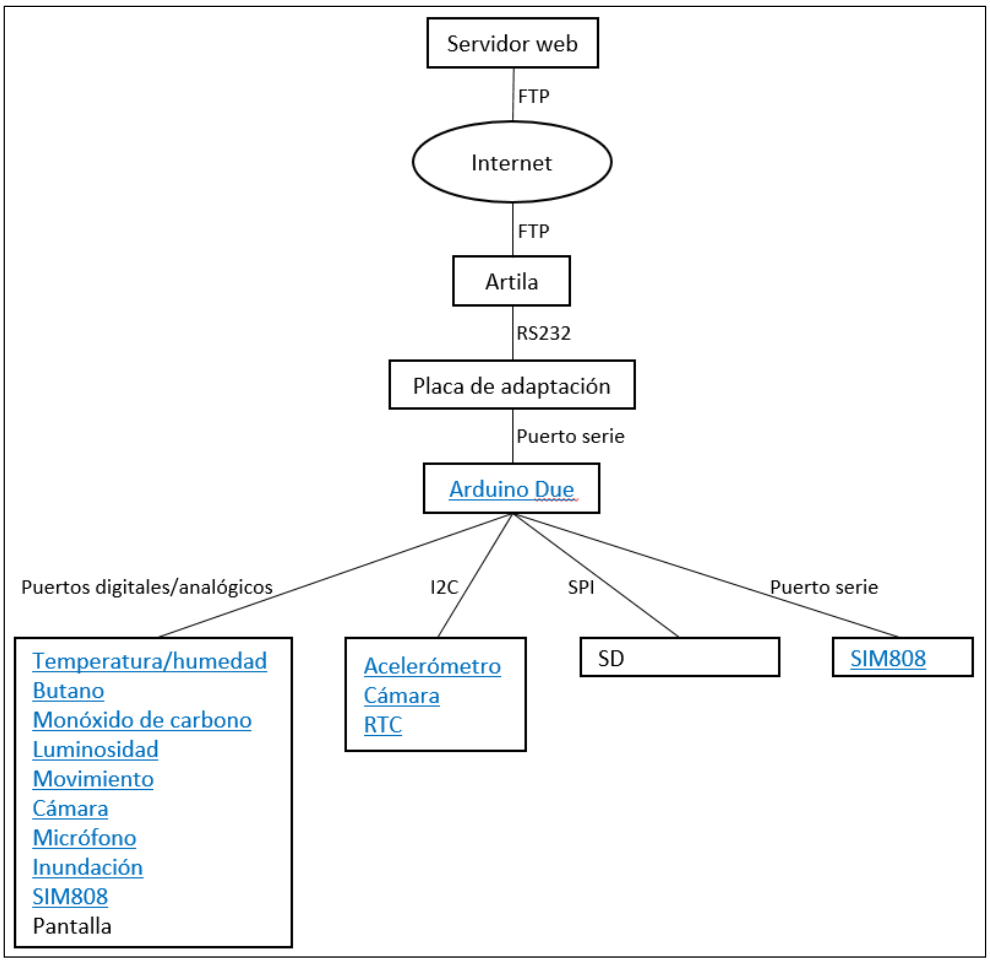
El proyecto se divide en dos fases, culminando cada una de ellas con un prototipo válido y totalmente funcional. La diferencia básica entre los dos prototipos es que en el primero la parte de control se compone de un servidor local y un microcontrolador, mientras que en el segundo se hace uso exclusivamente de un microcontrolador.

En el desarrollo del primer prototipo, abordado en una primera fase, se ven involucrados un micro PC (Artila, Matrix 500 [6]), un servidor web y un sistema Arduino Due. En el micro PC Artila se reutiliza el software del proyecto Blackbox (ver Sec. 1.2), que interroga al sistema Arduino por los valores de sus sensores y los envía al servidor web para su visualización mediante programas de consulta y gráficas desde un navegador convencional.

Para desarrollar este prototipo, las tareas principales llevadas a cabo comprenden el estudio en profundidad del microcontrolador empleado, la puesta en funcionamiento de cada uno de los sensores y dispositivos individualmente, su integración en un único software y su validación.

En el desarrollo del segundo prototipo, se sustituyeron las placas Arduino y Artila por el módulo ESP8266 que comunica directamente con el servidor web. Para subir los datos capturados desde los sensores por el ESP8266 hacia el servidor web, se estudió el protocolo que sigue el Artila para poder implementar un sistema similar en el ESP8266. Adicionalmente se buscaron alternativas para implementar la funcionalidad de visualización en *streaming* desde la cámara.

Figura 3. Arquitectura *hardware* del primer prototipo



* Los componentes subrayados se han desarrollado íntegramente. El resto se han implementado usando librerías ya disponibles.
Fuente: Elaboración propia.

3. Diseño

En este capítulo se recogen los aspectos de diseño general del sistema y se detallan en profundidad todos los componentes utilizados.

3.1. Arquitectura hardware

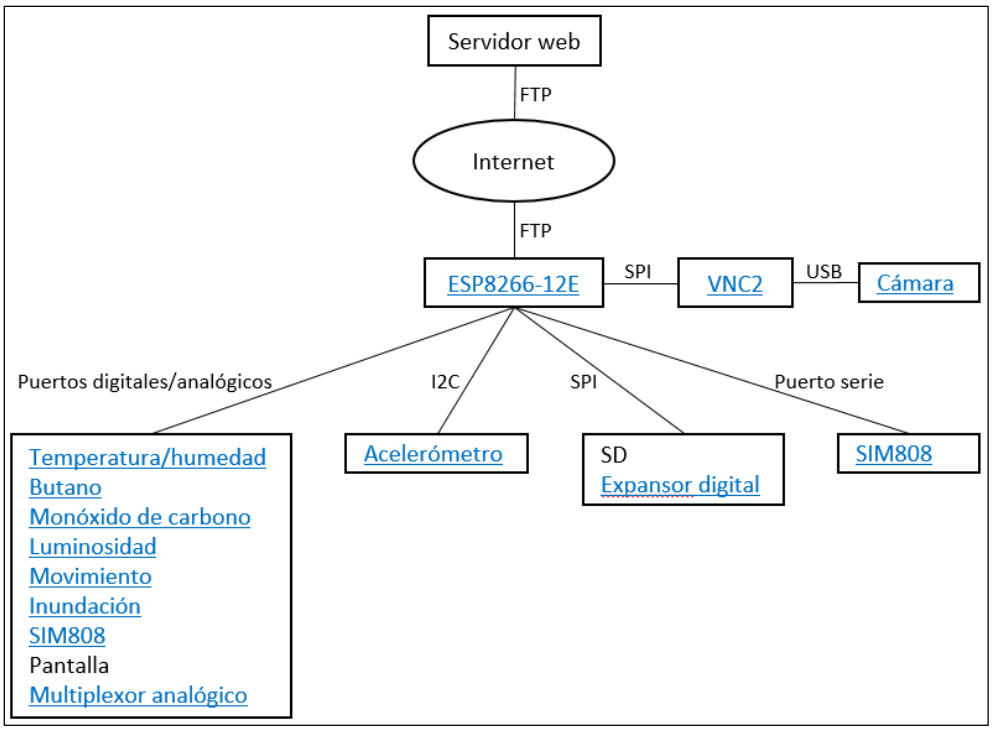
En la *Figuras 3 y 4* se reflejan los elementos utilizados en el desarrollo del primer y segundo prototipos respectivamente. A continuación se describen todos los componentes diferenciando si se usaron en el primer prototipo, en el segundo o en ambos.

3.1.1. Componentes de la fase de desarrollo

A continuación se relacionan los componentes incluidos exclusivamente en el primer prototipo. Estos componentes fueron sustituidos por otros en la implementación del prototipo final.

Diversos problemas aparecieron con varios sensores o componentes de modo que además de las dificultades técnicas inherentes a la falta de documentación o de

Figura 4. Arquitectura *hardware* del segundo prototipo



* Los componentes subrayados se han desarrollado íntegramente. El resto se han implementado usando librerías ya disponibles.
Fuente: Elaboración propia.

especificaciones técnicas correctas, surgieron problemas con el suministro de equipos averiados. Hubo que comprobar cada componente, su correspondencia con el pedido original, el correcto funcionamiento, la equivalencia con el *datasheet* publicado etc. De modo que cada nuevo sensor o componente utilizado en el prototipo supuso un esfuerzo adicional de chequeo y validación de su funcionamiento para incorporarlo al prototipo o desecharlo y buscar una nueva alternativa.

3.1.1.1. Arduino Due

Arduino es una compañía de *hardware* libre que ha desarrollado placas de control de bajo coste que se programan mediante su entorno de desarrollo (Arduino IDE). Gracias a su característica de "*hardware* libre", el florecimiento de una importante comunidad y el bajo precio de venta han conseguido un gran respaldo por los desarrolladores interesados en el uso de la electrónica y la programación para todo tipo de proyectos.

La placa de control que se eligió para el desarrollo del proyecto es la Arduino Due. Ésta incorpora un microcontrolador basado en ARM Cortex-M3 [7] y se caracteriza principalmente por ser la primera de su familia en utilizar un procesador de 32 bits, por funcionar a 3V3 y por alcanzar los 84 MHz. Estos datos se traducen en un aumento significativo de prestaciones frente a sus predecesores, que integraban microcontroladores AVR de 8 bits a 5V y a una frecuencia máxima de 16 MHz.

Cuenta con 54 puertos digitales, 12 analógicos de entrada, 2 analógicos de salida y líneas específicas para comunicación serial, I2C [8] y SPI [9]. Además, 12 de los puertos digitales se pueden configurar como PWM (*Pulse Width Modulation*).

Es importante destacar que la placa trabaja a 3V3, ya que desde un primer momento se tenía pensado sustituirla por el módulo ESP8266, que utiliza la misma tensión. Se dan más datos sobre este módulo en el punto 3.1.3.1.

3.1.1.2. Artila (Matrix 500)

Artila Electronics es una empresa cuya actividad se centra principalmente en el desarrollo de soluciones ARM para sistemas empotrados basados en Linux.

Se ha utilizado el modelo Matrix 500, que es el que se usa en Pariver con el software Blackbox mencionado en la sección 1.2. para la captura distribuida de datos. Para la comunicación se conecta uno de sus puertos RS-232 [10] al puerto serie del Arduino mediante una placa de adaptación intermedia (desarrollada y fabricada en Pariver).

El Artila ejecuta el software Blackbox que interroga al Arduino bajo condiciones especificadas en varios ficheros de configuración, y envía la información recopilada a un servidor web remoto mediante FTP (*File Transfer Protocol*). Los ficheros de configuración se utilizan para establecer diversos parámetros de funcionamiento del sistema como la velocidad de comunicación entre Artila y Arduino, o para indicar los sensores sobre los que recopilar información.

3.1.1.3. Cámara

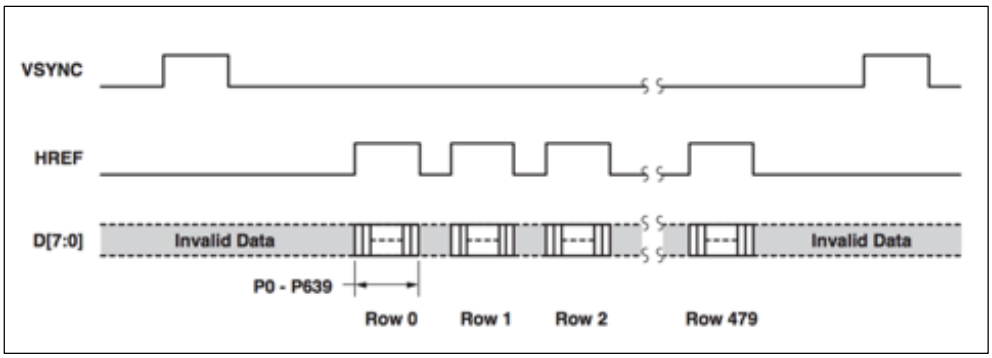
La cámara elegida inicialmente corresponde con el modelo OV7670 [11] de Omnivison. Se pueden encontrar dos versiones de esta cámara, una con memoria FIFO (*First In First Out*) integrada que requiere de menos prestaciones por parte del controlador para la extracción de una imagen y otra sin esa memoria FIFO.

La cámara funciona mediante un protocolo paralelo entregando los datos por sus pines D7-D0 y en sincronía con las señales HREF, VSYNQ y CLK. El contenido de los datos depende de la configuración de sus registros internos a los que podemos acceder en lectura y escritura mediante protocolo I2C.

El primer modelo con el que se trabajó no incorporaba la memoria FIFO. Se configuró de tal modo que la frecuencia del reloj de la cámara se dividiera al máximo manteniendo la resolución más alta (VGA). No sólo se detectaron problemas de velocidad para acceder a los datos, sino que el tamaño de una captura completa superaba el de la memoria del controlador, obligando a ir escribiendo en la tarjeta SD por bloques, aumentando el coste temporal del proceso.

Cuando se detectaron estos problemas, se buscó información adicional entre diferentes fabricantes y distribuidores para localizar una solución aceptable a nuestro caso. Se pudo comprobar que gran cantidad de desarrolladores se habían enfrentado a este mismo problema. Para solucionarlo se citaba una implementación diferente del mismo modelo de cámara que incluía una memoria FIFO. Se adquirió la que incorporaba el modelo de memoria AL422B [12] que ofrece una capacidad de 393.120 bytes. En las primeras pruebas fallaba la comunicación por I2C, circunstancia que no se había dado con el primer modelo. Se encontraron librerías de otros desarrolladores que habían trabajado con ambas. En ellas no quedaba bien definido como se tenía que realizar la conexión ni qué modelo de cámara resolvían por lo que no se consiguió nada con ellas. Aunque se pudo consultar el *datasheet* de la cámara y el de la memoria,

Figura 5. Protocolo de extracción de la imagen del modelo de cámara OV7670



Fuente: http://www.voti.nl/docs/OV7670.pdf

no se encontró uno del montaje conjunto. Finalmente, se localizó el esquema electrónico desarrollado en KiCad en el que se podía examinar cómo se conectaban las líneas internamente.

El proceso desde que la cámara toma una imagen hasta que la recibe el microcontrolador consta de dos partes totalmente separadas: almacenamiento de la imagen en la memoria FIFO y su posterior recuperación.

Almacenamiento en la memoria FIFO: Corresponde con la grabación de la imagen tomada en la memoria FIFO, en ella participan dos protocolos: el protocolo paralelo de extracción de la imagen de la cámara y el de escritura en la memoria. En el primero, representado en la *Figura 5*, la cámara facilita una captura completa durante el tiempo que la señal VSYNQ permanezca en *low*, cada una de las filas mientras HREF este en *high*, y cada byte de la fila en cada flanco de subida de su reloj interno a través de sus señales de datos (D[7:0]). El protocolo de grabación en la memoria se sirve de la señal de reinicio del puntero de escritura al origen (conectada a VSYNQ), de la de habilitar escritura (conectada a HREF si activamos un pin externo), de otras ocho que representan el byte (conectadas a las ocho de datos de la cámara) y de una señal de reloj compartida por ambas. Por tanto, en esta fase, se deberá activar el pin externo mencionado tras el flanco de bajada de la señal VSYNQ y desactivar el pin tras el flanco de subida de VSYNQ, en este momento habrá una captura en la memoria FIFO.

Recuperación de la imagen: el microcontrolador extrae la captura de la memoria. Se dispondrá nuevamente de ocho señales de datos (distintas a las de entrada), otra para reiniciar el puntero de lectura al origen y una de reloj que se conecta a un pin digital del Arduino. Así, se conduce un *high* y posteriormente un *low* al pin del reloj para completar cada ciclo. En cada ciclo la memoria dispondrá un byte en sus líneas de datos e incrementará el puntero de lectura. Aunque la memoria por separado sí que dispone de una señal para habilitar la lectura, en esta implementación está conectada a masa (activa).

La cámara permite generar los formatos de salida YCbCr [13] y RGB. Se seleccionó YCbCr porque requiere menos bytes para una misma captura. Una imagen con resolución 640x480 está formada por 614.400 bytes, demasiado para la memoria FIFO. Para mantener la calidad de imagen y el ángulo visión fue necesario modificar un par de registros de la cámara que permitían ajustar la resolución manualmente. Se redujo hasta 640x307, obteniendo para cada captura en formato YCbCr 392.960 (640*307*4/2) bytes que ya se pueden almacenar en la memoria.

Figura 6. Fórmula de conversión YCbCr a RGB

$$
\begin{aligned}
R &= Y & &+ 1.402 & &\cdot (C_R - 128) \\
G &= Y - 0.34414 & &\cdot (C_B - 128) - 0.71414 & &\cdot (C_R - 128) \\
B &= Y + 1.772 & &\cdot (C_B - 128)
\end{aligned}
$$

Fuente: https://es.wikipedia.org/wiki/YCbCr

Para guardar las capturas se escogió el formato Windows bitmap (extensión .bmp) porque trabaja con datos sin comprimir. El formato exige unas cabeceras de configuración sencillas para saber cómo interpretar los bytes de la imagen. La composición de color admitida es RGB por lo que se usaron las fórmulas de la *Figura 6* para transformar los datos obtenidos.

3.1.1.4. Micrófono

Para utilizar este sensor se tuvo que deducir su funcionamiento de forma experimental dado que no se pudo acceder a información técnica suficiente. Inicialmente se supuso que sería capaz exclusivamente de detectar el nivel de ruido (la especificación recogida en la placa indica "*sound sensor*"), por lo que se conectó VDD a 3V3, GND a masa y el pin restante a un puerto analógico, obteniendo así el valor sonoro detectado.

Posteriormente, y para satisfacer los requisitos iniciales del proyecto de grabar sonido ambiente, hubo que muestrear la señal de entrada a una determinada frecuencia y almacenarla en un formato adecuado.

Buscando formatos que trabajasen con datos sin compresión, para evitar la introducción de nuevo hardware, se optó por los ficheros de audio WAV [14]. Este formato define unas cabeceras en las que indican parámetros como el tamaño de muestra, frecuencia de muestreo, bits por muestra, etc.

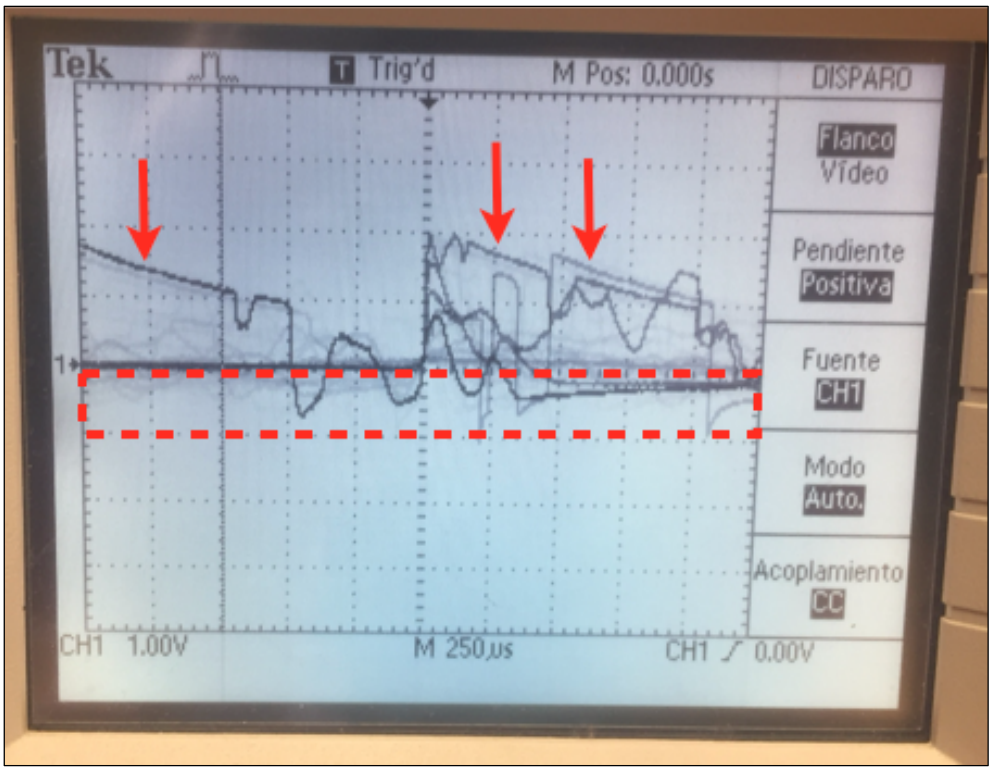
Se desarrolló un código que mediante tres parámetros de entrada generaban el fichero completo. Esto ayudó a la hora de agilizar las pruebas con diferentes frecuencias y bits por muestra. En todas las pruebas se obtenía mucho ruido de fondo.

Observando con el osciloscopio la señal de salida del micrófono se pudo percibir que dicha señal no era la esperada. La señal ideal debería permanecer entre 0 y 3V3, manteniéndose en un punto intermedio cuando estuviera en silencio. En la *Figura 7* se puede ver como la señal media oscila en torno a 0 V, produce picos antes de llegar a 2 V y baja por debajo de 0 V (valor que no puede detectar el Arduino).

La reoperación del micrófono, mediante su desoldadura del *shield*, no fue posible debido a que se trata de un sensor de una sola soldadura. Por ese motivo, se sustituyó por otro con el que finalmente se consiguió una señal coherente.

El Arduino realiza lecturas en sus pines analógicos con diez bits de precisión, lo que se traduce en valores en el rango [0-1023], cuyo equivalente en voltios es [0-3,3]. Una señal ideal oscilaría en torno a 512 sin sobrepasar el rango mencionado. Se obtuvo una señal que oscilaba en torno a 210 sin sobrepasar los márgenes. Para alinear la señal se tenía que desplazar ese punto medio, sumando 312 a los valores obtenidos (312+210=512). Se planteó probar dos configuraciones, ambas tomando muestras a 8 KHz, que es lo necesario para un canal telefónico de voz. Una con 8 bits por muestra

Figura 7. Señal original del micrófono



Fuente: Elaboración propia.

y otra con 16. En la primera se divide la señal leída (10 bits) por 4 perdiendo 2 bits de precisión. En la segunda se añaden 6 bits de relleno para completar los 16 bits. Como no se apreciaba diferencia en los sonidos captados y teniendo en cuenta que con ocho bits por muestra se necesitaría la mitad de almacenamiento, se seleccionó dicha configuración. Se logró reconocer la voz hasta unos 4-5 metros de distancia desde el individuo al micrófono.

3.1.1.5. RTC (*Real Time Clock*)

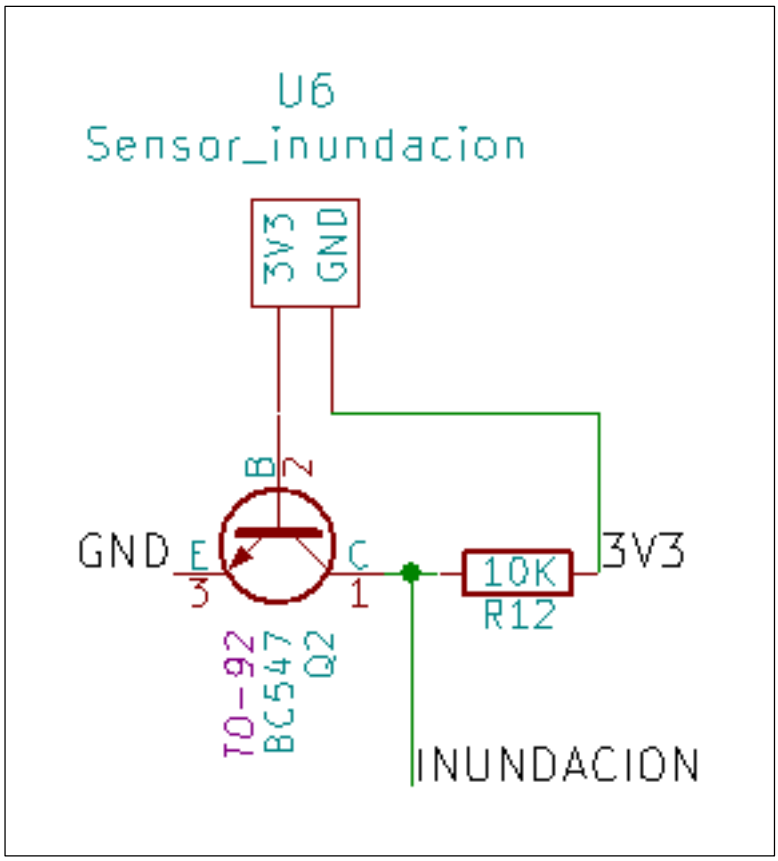
Este componente surgió de la necesidad de almacenar los ficheros de audio en la tarjeta SD. En ese sistema de almacenamiento se desarrolló un sistema de nombres que se generan dependiendo de la hora del sistema.

Se eligió una placa que incorpora el modelo DS3231 y una bahía para colocar una pila de forma que no se desincronice la hora ante un corte del suministro eléctrico. En el funcionamiento normal se alimenta por el pin de 3V3 del Arduino y se comunica mediante I2C con el microcontrolador.

Sometiendo el sistema a reinicios y cortes en la alimentación nunca perdió el valor correcto de la hora y el sistema completo siempre inició con normalidad.

Este reloj se descartó para el segundo prototipo basado en el ESP8266, en el que se consultará un servidor NTP (*Network Time Protocol*) para sincronizar la hora del sistema.

Figura 8. Circuito de adaptación del sensor de inundación.



Fuente: Elaboración propia.

3.1.2. Componentes comunes

A continuación se detallan los dispositivos incluidos en el desarrollado con la placa Arduino (primer prototipo) y que han sido mantenidos posteriormente con el ESP8266 (segundo prototipo).

3.1.2.1. Sensor de inundación

La documentación de este sensor sólo especificaba que tiene dos pines (polos positivo y negativo). Se supuso que cuando el agua toca las dos patillas del sensor habría paso de corriente (uno lógico), detectando así la presencia de agua. Se conectó un pin a 3V3 y el otro a masa pero esto no ocurría.

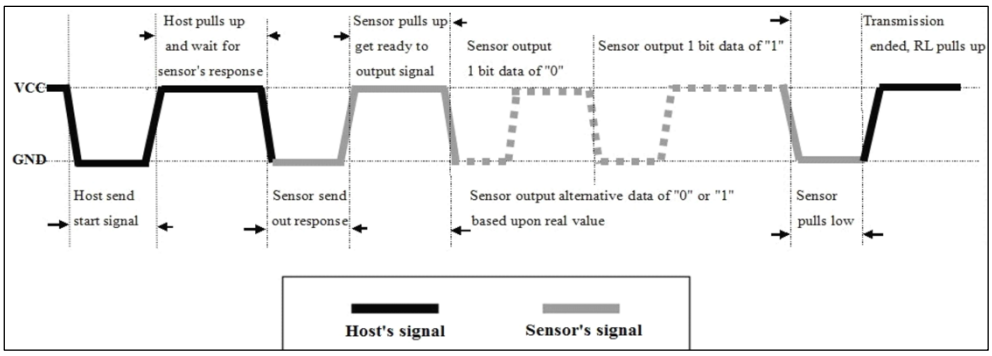
Mediante un polímetro se comprobó que el sensor tenía una resistencia interna de 17 Millones de ohmios, por lo que se diseñó el circuito representado en la *Figura 8*, que incorpora un transistor BC547, y que permite leer un cero lógico en presencia de agua y un uno lógico en caso contrario.

3.1.2.2. Sensor de temperatura y humedad

Para este cometido se eligió el sensor DHT22. Puede funcionar a 3V3 o a 5V, implementa su protocolo de comunicación mediante un pin digital y tiene precisión de un decimal tanto para temperatura como para humedad.

El protocolo implementado corresponde con la *Figura 9*. El microcontrolador y el sensor van intercambiando los *rolles* de *master* y *slave* para transmitir las señales de

Figura 9. Protocolo de comunicación con el sensor DHT22



Fuente: https://cdn-shop.adafruit.com/datasheets/Digital+humidity+and+temperature+sensor+AM2302.pdf

control y la información. El microcontrolador debe mantener la señal en *high* cuando no haga uso del dispositivo. Para comenzar la transmisión deberá dirigir un *low* entre 1 y 10 ms, volver a *high* y esperar la respuesta del sensor. El sensor transmitirá un ciclo (*low* + *high*) para indicar que está preparado, y en ese momento tendremos que leer 40 bits de datos.

Para cada bit, el sensor conducirá una señal *low* y una *high*, siendo el bit un uno o un cero dependiendo del tiempo que mantenga la línea en *high*. Los dos primeros bytes formarán el valor de humedad y los dos siguientes el valor de temperatura (el primer bit de la temperatura indica el signo del valor). Por último se recibe un byte de *Check-sum*. Para comprobar que hemos recibido correctamente se suman todos los bytes recibidos, y se compara el byte menos significativo de la suma con el *Check-sum*.

3.1.2.3. Sensor de movimiento

Este sensor no presentó ninguna complicación. Dispone de tres pines: VCC, GND y uno digital. Se conecta VCC a 3V3, GND a masa y se obtiene un uno lógico por la salida digital cuando se detecta movimiento. Además, la señal se mantiene durante aproximadamente dos segundos, por lo que se puede leer mediante encuesta.

El bucle principal del software es capaz de leer todos los sensores en el peor caso una vez por segundo, por lo que no se pierde la señal. Para el montaje con el Artila, el valor correspondiente a que se ha detectado movimiento no se limpia hasta que este haya preguntado por el valor.

3.1.2.4. Sensor de luminosidad

Se pudo incorporar sin dificultades. Se alimenta a 3V3 y devuelve un valor por su pin analógico entre 0 y 1023 que podemos registrar.

3.1.2.5. Sensor de gas butano

Para cubrir el requerimiento de detección de gas butano se optó por un sensor del tipo MQ6. Dispone de un *datasheet* [15] que incluye el esquema de conexión que hay que seguir. Devuelve un valor analógico que se incrementa con la presencia de butano. Para comprobar su funcionamiento se utilizó la aportación de gas en proximidad con una pequeña fuente emisora del tipo de un mechero.

3.1.2.6. Sensor de monóxido de carbono

Este sensor es muy importante. Se encarga de detectar la presencia de monóxido de carbono, un gas que puede ser mortal en espacios cerrados además de ser imperceptible para el ser humano.

Pertenece a la familia del sensor de gas butano y por el que se optó es del tipo MQ7. El problema que plantea es que a diferencia del MQ6 tiene dos fases que se van alternando constantemente:

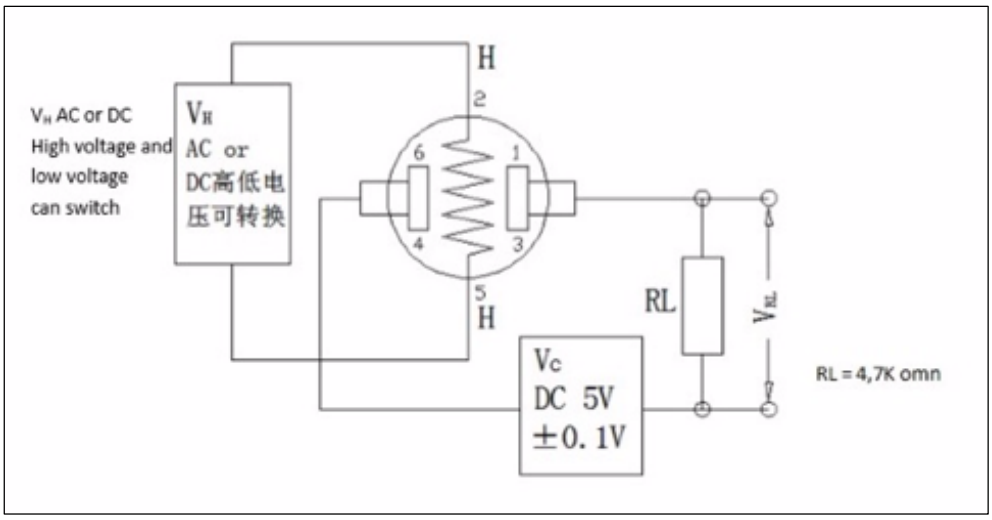
- En la primera (calentamiento) se debe suministrar 5V por una de sus patas durante 60 segundos.

- En la segunda fase (lectura) se tiene que alimentar por esa misma pata con 1,5V durante 90 segundos y es en esta fase cuando se leen valores válidos.

Para cambiar la alimentación que se suministra por un pin se necesita un PWM, pero como el Arduino Due trabaja a 3V3, el máximo valor que podrá tener en esa salida está limitado a esa tensión. Para solucionarlo se adaptó el circuito original a uno nuevo el que se incluye un transistor BD139 para poder trabajar con los voltajes necesarios. Podemos ver el circuito original y su adaptación en las *Figuras 10 y 11* respectivamente.

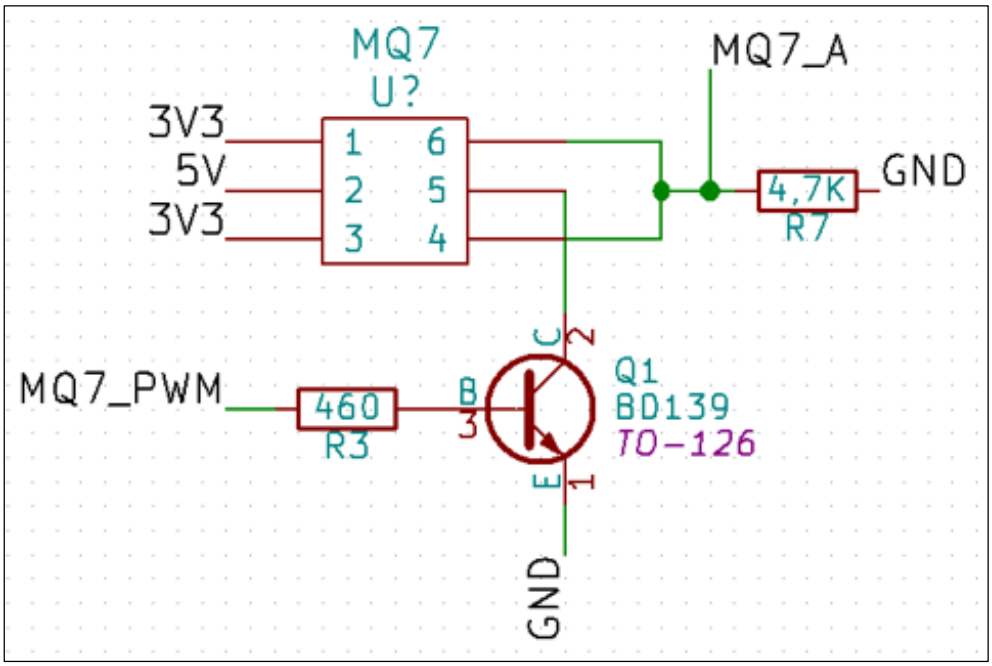
Para comprobar el correcto funcionamiento se expuso el sensor a la salida del tubo de escape de una motocicleta que entre otros gases desprende monóxido de carbono.

Figura 10. Circuito de conexión original del sensor MQ7.



Fuente: http://www.winsen-sensor.com/d/files/PDF/Semiconductor%20Gas%20Sensor/MQ-7B%20(Ver1.4)%20-%20Manual.pdf

Figura 11. Circuito de adaptación el sensor MQ7.



Fuente: Elaboración propia.

3.1.2.7. Acelerómetro

Este dispositivo dispone de dos líneas para la comunicación mediante el protocolo I2C, además de las dos de alimentación.

Este chip está preparado para funcionar a 3V3. Se optó por uno suministrado en un *shield* que incluía un regulador de tensión. Éste bajaba la tensión de 5V a 3V3 para hacerlo eléctricamente compatible con el resto de Arduinos del mercado que funcionan a 5V. Si la entrada de alimentación es inferior o igual a 3V3 la mantiene en ese nivel por lo que no provocó problemas. Sin embargo, incorpora un componente bidireccional adicional para las líneas SCL y SDA que rebaja la tensión de 5V a 3V3. Como estas líneas ya operan a 3V3 en el Arduino Due, cuando atraviesan el componente quedan reducidas en un valor que el chip no es capaz de detectar, impidiendo la comunicación. Se solucionó eliminando dicho componente y uniendo sus entradas con sus salidas para que las líneas SCL y SDA entre el microcontrolador y el chip no se vieran alteradas.

Para este proyecto no necesitamos un acelerómetro de gran precisión porque basta con saber que ha habido movimiento. Moviéndolo en cada una de las direcciones es capaz de detectarlo satisfactoriamente leyendo sus registros correspondientes.

3.1.2.8. LCD táctil

El modelo que se adquirió incluye un *shield* para conectar directamente a un Arduino Due o Mega (tienen las mismas dimensiones). Además, hay disponible una librería para manejar tanto el LCD como el panel táctil.

Dispone de registros mapeados en memoria para realizar las configuraciones oportunas. Para acceder a ellos se usa el protocolo paralelo 8080 de Intel.

La librería utilizada dispone de ejemplos para su utilización. Todos ellos funcionaron correctamente, tanto los que involucraban a la parte de mostrar imágenes/texto como los que utilizaban el control táctil.

Esta pantalla también incluye una ranura para conectar una tarjeta SD, muy útil para ampliar el almacenamiento interno de nuestro microcontrolador. Hay disponible una librería oficial para trabajar con FAT16 y FAT32 que funciona correctamente.

En cuanto se tuvieron las tres partes funcionando, se eliminó el *shield* y se conectó cada señal individualmente, minimizando el espacio y dejando puertos para conectar el resto de componentes.

3.1.2.9. SIM808

Este chip proporciona conectividad GSM, GPRS, GPS y *bluetooth*. Se comunica con el microcontrolador a través de un puerto serie mediante comandos AT. Los comandos AT son un estándar abierto desarrollado por la compañía Hayes Communications para configurar/interactuar con módems [16].

El puerto serie se debe configurar para trabajar en ambos extremos a la misma velocidad. Por defecto comunica a 9600 baudios, así que se configura el controlador de este modo y entonces se puede subir la velocidad. Como todo puerto serie, requiere dos pines para la comunicación: TX para la transmisión y RX para la recepción, de manera que se conecta el pin TX de uno con el RX del otro.

Para obtener información del módulo se le envía un comando que devuelve todos los datos periódicamente. El periodo se indica en segundos como parámetro del comando. Los datos vienen en una cadena de caracteres separados por ",", de manera que se puede extraer cada uno de ellos sencillamente.

Aunque se comprobó que el protocolo Bluetooth funcionaba correctamente no se le ha asignado ninguna funcionalidad para el TFG. El resto de datos se almacena en el servidor web para su consulta.

Para encender/apagar el módulo, se le aplica en un pin un pulso de dos segundos aproximadamente. En las pruebas, el módulo no se apagaba en los reinicios del sistema completo porque no perdía la alimentación, y sí lo hacía en el proceso de puesta en marcha. Se modificó el código para que antes de mandar el pulso de activación del módulo se le mandará un comando para comprobar si estaba ya encendido.

3.1.3. Componentes del sistema final

En este apartado se introduce el nuevo microcontrolador utilizado, así como los componentes necesarios para trabajar con él.

3.1.3.1. ESP8266-12E

El sistema final es gobernado por el módulo ESP8266-12E [17]. La peculiaridad de este módulo es que implementa el protocolo de pila TCP/IP para conexión WiFi a 2,4 GHz. Su reloj interno soporta tanto 80 como 160 MHz. Nos proporciona líneas de I/O para comunicación serie, I2C y SPI. Si se utilizan todas ellas, quedaran disponibles para el usuario cuatro GPIOs (*General Purpose Input/Output*) y un último pin conectado a un ADC (*Analog Digital Converter*) para obtener los valores analógicos. Además funciona a 3V3 por lo que los circuitos preparados para solucionar los problemas en el Arduino Due se pueden aprovechar para este controlador.

Concretamente se utilizó la placa NodeMCU v1.0. [18], que implementa dicho módulo. Esta versión exporta todos los pines al exterior de manera que se puede conectar fácilmente a una placa de prototipado para las pruebas. Además dispone del convertidor de tensión necesario para alimentarlo y a su vez programarlo a través de un puerto USB del ordenador.

Se puede programar mediante un lenguaje de *scripting* (LUA) o mediante el entorno de desarrollo de Arduino instalando un *plugin*. En este proyecto se utilizó el IDE de Arduino.

3.1.3.2. Expansor de puertos digitales

Como se detalla en el apartado anterior, además de las líneas de comunicación, sólo se dispone de cuatro GPIOs adicionales. Como consecuencia se necesitó un sistema que ampliara esta capacidad. Existen en el mercado expansores digitales que satisfacen esta necesidad.

El expansor escogido para el proyecto corresponde con el modelo MPC23S17 [19]. Añade 16 GPIOs y se controla mediante el protocolo SPI a 10 MHz. Éstos se dividen en dos puertos de manera cada uno controla ocho líneas. Se sirve de once registros mapeados en memoria para cada puerto mediante los que configurar si el pin

Figura 12. Funciones que abstraen el funcionamiento del expansor MPC23S17

```
// Funciones genéricas
void Expander::pinDir(unsigned char _id, unsigned char _port, unsigned char _pin, unsigned char _dir){

    unsigned char _aux = readRegister(_id, _port, IODIR);

    if (_dir == 1)      _aux = _aux & (~(1 << _pin)); // OUTPUT
    else if (_dir == 0) _aux = _aux | (1 << _pin); // INPUT

    writeRegister(_id, _port, IODIR, _aux);
}
void Expander::write(unsigned char _id, unsigned char _port, unsigned char _pin, unsigned char _value){

    unsigned char _aux = readRegister(_id, _port, GPIO);

    if (_value == HIGH)     _aux = _aux | (1 << _pin);
    else if (_value == LOW) _aux = _aux & (~(1 << _pin));

    writeRegister(_id, _port, GPIO, _aux);
}
unsigned char Expander::read(unsigned char _id, unsigned char _port, unsigned char _pin){

    return (readRegister(_id, _port, GPIO) >> _pin) & 0x01;
}


// Funciones para uso intensivo
void Expander::pinDirAll(unsigned char _id, unsigned char _port, unsigned char _dir){
    if (_dir == 1)      writeRegister(_id, _port, IODIR, 0x00); // OUTPUT
    else if (_dir == 0) writeRegister(_id, _port, IODIR, 0xFF);// INPUT
}
void Expander::writeAll(unsigned char _id, unsigned char _port, unsigned char _value){
    writeRegister(_id, _port, GPIO, _value);
}
```

Fuente: Elaboración propia.

será de entrada o salida, el valor por defecto, si se quiere que provoque interrupción, etc.

El protocolo SPI se implementa a través de tres líneas. Dos de ellas son comunes para todos los dispositivos que comuniquen a través de él, la tercera (*chip select*) es individual por cada *slave* y sirve para marcar con quién comunica el controlador.

Un dato importante es que podemos utilizar hasta ocho expansores de este tipo con una única línea de *chip select*. Se direccionan mediante tres bits de direccionamiento en el primer byte del protocolo de comunicación, que indica el código de operación del dispositivo. Estos tres bits corresponden con las líneas S2-S0 del expansor que se deben llevar a GND o VCC para fijar dicha dirección. No es una funcionalidad por defecto, por lo que hay que activarla escribiendo uno de sus registros.

Se incluyeron dos de estos expansores en el proyecto, uno exclusivo para el control de la LCD y otro para el resto de dispositivos digitales. Se implementaron una serie de funciones para facilitar el uso de cada GPIO de manera individual. Sin embargo, para la LCD, que requiere de un uso más intensivo, se añadieron dos funciones más que permitieran escribir/leer un puerto (ocho GPIOs) en una sola interacción. Dichas funciones aparecen en la *Figura 12*.

3.1.3.3. Multiplexor de puertos analógicos

Un multiplexor es un componente electrónico que exporta varias entradas de datos y una única de salida. Sirven para conducir uno de los datos entrantes a la señal saliente mediante una o varias líneas de control. Concretamente utilizaremos el modelo 74HC4051 [20] que dispone de ocho entradas.

Su uso es necesario ya que disponemos de varios dispositivos analógicos que queremos conservar y una única entrada del ESP8266 conectada a su ADC. Se conecta esta entrada a la salida del multiplexor y se gobiernan las tres líneas necesarias para conducir las ocho entradas analógicas mediante el expansor digital.

Del mismo modo que para el expansor se ha desarrollado un código capaz de abstraer el comportamiento del multiplexor, permitiendo hacer la lectura de cada sensor mediante un único parámetro. El parámetro corresponde con el valor numérico grabado en la placa de adaptación donde se conecta la línea a leer.

3.1.3.4. Cámara

Para este segundo prototipo se quiere ser capaz de visualizar video en *streaming*, una característica que muchos sistemas incorporan y que resulta muy atractiva de cara al usuario final.

El modelo de cámara elegido es el USBFHD04H de Surveillance Equipment CCTV Systems [21]. Permite transmitir video a 1080P en formato comprimido H.264 el cual minimiza considerablemente el tamaño de los datos. Esto es fundamental para la transmisión de video en *streaming*. Además, cuenta con 180º de ángulo de visión, una especificación alta si la comparamos con el resto de cámaras del mercado.

El problema que plantea es que se debe conectar mediante USB 2.0. Como el ESP8266 no tiene la capacidad de realizar este tipo de comunicación se han buscado alternativas. La elegida consiste en añadir un nuevo microcontrolador (VNC2) que permite esta comunicación e incluso incluye drivers de ejemplo para facilitar el desarrollo de la solución específica en cada caso. De este modo, el nuevo microcontrolador ejecutará un software que aceptará comandos del ESP8266 vía SPI y realizará la comunicación con la cámara. Cuando la cámara tenga que capturar imágenes, los datos serán transmitidos por USB al VNC2 y de éste al ESP8266 por SPI.

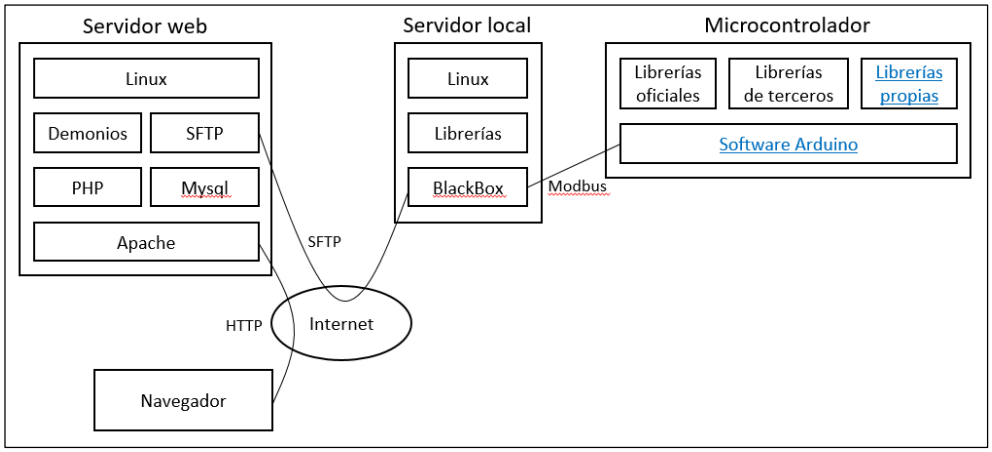
Así, el ESP8266 puede ejecutar comandos para tomar capturas de la cámara o para comenzar a retransmitir en *streaming*. Para esta segunda opción, si un cliente quisiera acceder desde fuera de la red doméstica, necesitaría configurar un puerto del router por el que aceptar la petición además de disponer de una IP estática. Para solucionarlo, ya que el objetivo es un sistema de fácil instalación, el controlador enviará los datos de video a un servidor externo que será al que realmente accederá el cliente.

Para la programación del VNC2 se utiliza en entorno de desarrollo Vinculum II a través del chip de programación/*debug* FT232.

3.2. Arquitectura software

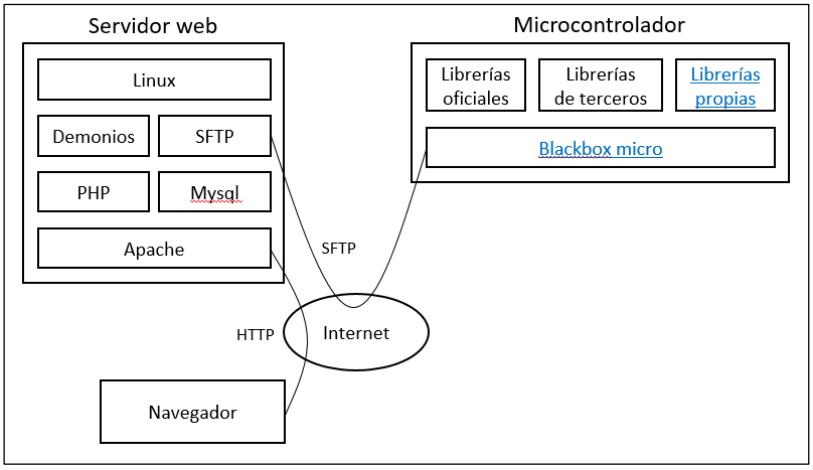
En ambos prototipos los controladores, Arduino y ESP8266, carecen de sistema operativo mientras que en el primero, el sistema de interrogación del Artila corre bajo un Linux. Se puede ver el esquema de arquitectura software de los dos prototipos desarrollados en las *Figuras 13 y 14*.

Figura 13. Arquitectura *software* del primer prototipo



* Los componentes subrayados se han desarrollado íntegramente. El resto se han implementado usando librerías ya disponibles.
Fuente: Elaboración propia.

Figura 14. Arquitectura *software* del segundo prototipo



* Los componentes subrayados se han desarrollado íntegramente. El resto se han implementado usando librerías ya disponibles.
Fuente: Elaboración propia.

Tabla 2. Librerías oficiales y su descripción

Librería	Descripción
Wire	Implementa la comunicación I2C
SD	Implementa la comunicación con tarjetas SD. Facilita funciones para la creación de ficheros, directorios, así como para su borrado o consulta.

Fuente: Elaboración propia.

Tabla 3. Librerías de otros desarrolladores

Librería	Descripción
UTouch	Implementa la interacción táctil con la pantalla LCD. Se tuvo que adaptar para manejarla como una interrupción externa y no mediante encuesta.
UTFT	Implementa el control de la parte visual de la LCD. Facilita funciones para mostrar figuras, líneas, archivos de imagen, cadenas de texto y números en formato 7-segmentos.
DHTLib	Implementa el protocolo de comunicación con el sensor de temperatura y humedad, permitiendo obtener dichos valores.

Fuente: Elaboración propia.

Tanto el Arduino como el ESP8266 se pueden programar desde el sistema de desarrollo de Arduino de manera directa para el primero e instalando un *plugin* en el caso del segundo. La versión del entorno de desarrollo que se utilizó es la 1.6.12. Las librerías que éste ofrece se recogen en la *Tabla 2*. Todas ellas se pueden usar tanto por el Arduino como por el ESP8266 siempre que haya suficientes pines disponibles.

Además de las oficiales se incorporaron al proyecto librerías de otros desarrolladores, quedan recogidas en la *Tabla 3*.

Lamentablemente, no en todos los casos las librerías satisfacían los requerimientos del proyecto. Por ello, hubo que desarrollar algunas librerías ex novo. A continuación se señalan las librerías desarrolladas para cubrir el funcionamiento del resto de sensores y componentes.

Se desarrolló una librería denominada Microphone ya que no se tuvo acceso a ninguna que se pudiera implementar directamente en el proyecto. Su funcionamiento depende de un *timer* que toma muestras del pin analógico 8.000 veces por segundo y las almacena en un buffer circular. De este punto derivan los dos requisitos de la librería: disponer de una SD y guardar el buffer en ella con un periodo inferior a dos segundos para que no desborde. Se mantiene otro buffer circular en la SD con capacidad para 20 ficheros que corresponden con los últimos 20 segundos, de forma que se crea uno y se borra el último una vez por segundo.

También se tuvo que desarrollar la librería para gestionar las fases de lectura y calentamiento del sensor de monóxido de carbono. En ella se mantiene el valor devuelto por *millis()* (milisegundos transcurridos desde la puesta en marcha del sistema) y se compara con cada llamada a la función de lectura para comprobar en qué fase está y si hay que cambiarla. El valor devuelto corresponde con el valor tomado del pin analógico si está en fase de lectura, en caso contrario devuelve el último valor válido.

Para utilizar el módulo SIM808 se implementó una librería que permitiera utilizar tanto el GPS como la red GSM. En relación con el GPS se envía en su inicialización un comando que se encarga de activar una función en el GPS que devuelve los valore que éste registra cada diez segundos. De este modo, se debe comprobar periódicamente si hay mensajes a la entrada del puerto serie y almacenarlos debidamente. En relación con la red GSM, se desarrolló una función que envía un mensaje de texto indicado como parámetro mediante una cadena de caracteres que envía la aplicación. El teléfono al que va dirigido el mensaje se toma de un fichero de configuración.

La tarea de capturar y enviar una imagen tiene un coste temporal elevado. Por ello, en la librería que se desarrolló, se divide su envío en trozos de 700 milisegundos para no transmitir la sensación de funcionamiento detenido. Las capturas son tomadas y enviadas al Artila cuando éste lo solicita. Para conseguir unos tonos de color próximos a la realidad se tuvieron que realizar numerosas pruebas con diferentes configuraciones de los registros internos de la cámara.

Para el reloj de tiempo real también se tuvo que desarrollar una librería que leyera sus registros internos y los transformara en los valores correctos a través de una serie de operaciones. El reloj proporciona información sobre el año, el mes, el día, el día de la semana, la hora, los minutos y los segundos.

En la *Tabla 4* se recogen las funciones desarrolladas para el correcto funcionamiento de cada componente asi como una breve descripción. El resto de librerías no desarrolladas integramente también se han modificado en la medida de lo posible para adaptarse al funcionamiento general del sistema.

Tabla 4. Funciones desarrolladas integramente para cada dispositivo

Funciones	Librerías	Descripción
init	Accelerometer Expander ILI9341_LCD MQ6 MQ7 Light Multiplexer PIR SIM808 Water DateTime Microphone Camera	Función que inicializa el componente y lo deja en estado operativo. Sus parámetros son un puntero a un objeto que permite enviar mensajes a un servidor FTP para realizar las labores de depuración y los pines a los que se conecta cada uno. Estos pueden ser uno o varios dependiendo del componente. Además, si se conectan a través de expansor se deberá especificar a cual de ellos.
read	Accelerometer MQ6 MQ7 Light PIR SIM808 Water DateTime Microphone*	Implementa el protocolo de comunicación correspondiente a cada dispositivo. Tiene tantos parámetros como valores puede devolver el sensor. Cada uno es un puntero y es rellenado durante la ejecucicón de la función. *Añade el audio generado desde la última llamada a dicha función al sistema de ficheros de la SD correspondiente al segundo de la invocación.
	Multiplexer Expander	Devuelve el valor del pin indicado como parámetro.
write	Expander	Recibe dos parámetros que indican el valor que mantendrá un pin tras la ejecución de la función. Dispone de una versión para uso intensivo que escribe un puerto completo (ocho pines) en una interacción.
pinDir	Expander	Recibe dos parámetros que indican si un pin se configura como *INPUT* o *OUTPUT*. Dispone de una versión para uso intensivo que escribe un puerto completo (ocho pines) en una interacción.
writeDigit	ILI9341_LCD	Escribe un número de una cifra en formato 7-seg. Mediante los parámetros se indica el valor a mostrar, las coordenadas y el color en formato RGB. La dimensión del dígito es de 60x40 píxeles.
writePixel	ILI9341_LCD	Pinta un pixel en la LCD. Mediante los parámetros se indican las coordenadas del pixel y su color en formato RGB.
remove_old_audio	Microphone	Elimina el fichero de audio más antiguo de la LCD manteniendo un máximo de 20 ficheros (20 segundos).
send_audio	Microphone	Función que implementa el protocolo de envío de audio. Cada vez que se llama a la función se envía una parte de un fichero de audio al Artila. Se debe de llamar continuamente hasta que la función devuelva false. En ese momento el Artila habrá recibido los 20 segundos de audio solicitados. Los 20 segundos corresponden con los 10 anteriores a la petición del Artila y los 10 posteriores.
send_image	Camera	Función que implementa la extracción de la imagen de la memoria FIFO y la envía al Artila. En cada llamada la función envía un trozo de la imagen durante 700 ms y retorna un valor que indica si todavía queda imagen para enviar. La función debe ser llamada hasta que se complete el envío.

Fuente: Elaboración propia.

4. Pruebas

La realización de las pruebas unitarias se fue efectuando conforme se fueron implementando los distintos sensores en uno u otro prototipo. Las dificultades e incidencias detectadas y las acciones desplegadas en consecuencia se han descrito ya en la sección precedente, en la que se detallan cada uno de los sensores y dispositivos implementados.

Las pruebas globales se realizaron una vez dispuestos cada uno de los dos prototipos sobre el montaje final. En ese punto aparecieron, como era de esperar, algunos problemas adicionales que se exponen a continuación.

Aunque el sensor de temperatura y humedad funcionaba correctamente en las pruebas unitarias, en las pruebas conjuntas ocasionalmente se recibía un valor no válido. Conociendo el protocolo y mediante un osciloscopio se pudo detectar que el sensor no estaba contestando debido a problemas con las funciones de interrogación. En concreto, la función *delay(1)* debía generar el pulso inicial de 1 ms pero en ocasiones el valor recogido en el osciloscopio era muy inferior, no dando tiempo al sensor a captarlo. Por ello, el sensor no emitía valor alguno al no detectar la petición. La función citada utilizaba la llamada a otra denominada *millis()* que devuelve el número de milisegundos desde el arranque del Arduino. Al tener precisión de ms, la espera que hace *delay(1)* es cualquier valor entre 0 y 1 ms. Para solucionarlo, se hubo de cambiar la llamada a *delay()* con un 2 como argumento y así conseguir una espera entre 1 y 2 ms (según la documentación debe ser entre 1 y 10 ms). Una vez efectuados los cambios en la librería se realizaron nuevas pruebas y se logró un funcionamiento correcto en todos los casos.

Como en el caso del sensor anterior, todas las funciones de la LCD funcionaban correctamente en las pruebas individuales, pero en el montaje conjunto se dieron problemas con la parte táctil. Su controlador funciona de tal forma que cuando detecta presión provoca un cambio de valor a través de un pin. La librería disponible gestiona ese pin mediante encuesta. Se detectó que la pantalla no captaba en muchas ocasiones las pulsaciones o sufría retardos molestos. Para solucionarlo se estudió el funcionamiento de la librería y se modificó para que gestionara la señal como una interrupción externa.

Adicionalmente, se detectó que algunos otros sensores como el DHT22 o el acelerómetro requieren de tiempos mínimos entre lecturas que no se contemplan en las librerías provocando un funcionamiento erróneo en las lecturas del software principal. Se ha consideró que esta especificación debería estar implementada en la librería que abstrae el comportamiento del sensor. Por tanto, se modificaron dichas librerías con una misma estructura para que devuelvan el último valor obtenido siempre que el lapso de tiempo requerido para la siguiente lectura no haya sido superado.

Aparecieron, así mismo, problemas con la comunicación I2C, ya que era utilizada por varios dispositivos en un intervalo muy reducido. Se solucionó intercalando las lecturas entre los diferentes protocolos de modo que se eludieran las posibles colisiones o saturación del bus.

El resto de componentes no aportaron problemas adicionales en relación a la validación individual de cada uno.

5. Conclusiones y próximos pasos

Se ha obtenido un producto final de reducidas dimensiones y con una alta funcionalidad siendo capaz de cumplir todos los requisitos planteados.

La plataforma Arduino ha resultado muy apropiada para la primera toma de contacto con un microcontrolador en un entorno profesional. Se ha tenido acceso a abundante información además de suficientes líneas para conectar todos los componentes sin necesidad de elementos que pudieran añadir dificultades.

La plataforma utilizada en el segundo prototipo (ESP8266) también ha sido muy apropiada ya que manteniendo todas las funcionalidades del primer prototipo (incluyendo las propias del Artila) y reduciendo considerablemente las dimensiones del producto consigue un rendimiento superior.

La experiencia de trabajo en la empresa ha sido muy enriquecedora y me ha permitido realizar la transición del mundo académico al profesional de manera muy satisfactoria. Además ha resultado muy gratificante el poder ayudar a compañeros que tenían problemas con sensores con los que yo había profundizado y que se enfrentaban a situaciones similares.

Si algo he aprendido es que aunque haya librerías disponibles, es muy necesario hacer al menos una lectura rápida del manual del componente para extraer la información más relevante ya que en muchas ocasiones la librería no está completa o no está implementada de la forma que se espera. Esto ha provocado en alguna ocasión perder tiempo buscando problemas donde realmente no los había.

Como próximos pasos se plantea añadir una sirena que se active ante determinadas situaciones como medida de alarma local. Se valorará también aprovechar la parte táctil de la LCD para que el usuario pueda modificar algún parámetro de configuración.

Finalmente la idea es implementar este segundo prototipo dentro de un diseño que permita contar con un producto final de fácil instalación y coste ajustado. En la actualidad, continúo trabajando en estas extensiones ya como contratado de la empresa Pariver dentro del equipo de desarrollo de sistemas empotrados.

6. Bibliografía

[1] Canary Connect, «Canary,» 2014. [En línea]. Available: https://canary.is. [Último acceso: 1 Marzo 2016].

[2] Nest, «Nest,» Mayo 2010. [En línea]. Available: https://nest.com.

[3] Arduino, «Arduino,» 2005. [En línea]. Available: https://www.arduino.cc. [Último acceso: 1 Marzo 2016].

[4] J.-P. Charras, «KiCad EDA,» 1992. [En línea]. Available: http://kicad-pcb.org. [Último acceso: 9 Mayo 2016].

[5] Espressif, «ESP8266,» 2016. [En línea]. Available: https://espressif.com/en/products/hardware/esp8266ex/overview.

[6] Artila Electronics, «Embedded Networking and Computing,» [En línea]. Available: http://www.artila.com/en/p_matrix.html. [Último acceso: 20 Julio 2016].

[7] Atmel, «SAM3X / SAM3A Series,» 23 Marzo 2015. [En línea]. Available: http://www.atmel.com/Images/Atmel-11057-32-bit-Cortex-M3-Microcontroller-SAM3X-SAM3A_Datasheet.pdf.

[8] Wikipedia, «I2C,» 30 Diciembre 2016. [En línea]. Available: https://es.wikipedia.org/wiki/I²C.

[9] Wikipedia, «Serial Peripheral Interface,» 22 Marzo 2016. [En línea]. Available: https://es.wikipedia.org/wiki/Serial_Peripheral_Interface.

[10] Wikipedia, «RS-232,» 2 Noviembre 2016. [En línea]. Available: https://es.wikipedia.org/wiki/RS-232.

[11] OmniVision, «OV7670/OV7171 CMOS VGA (640x480) CameraChip with OmniPixel Texhnology,» 8 Julio 2005. [En línea]. Available: http://www.voti.nl/docs/OV7670.pdf. [Último acceso: 20 Mayo 2016].

[12] Averlogic, «AL422B Data Sheets,» 3 Enero 2001. [En línea]. Available: http://www.51hei.com/f/AL422B_Data_Sheets.pdf. [Último acceso: 27 Septiembre 2016].

[13] Wikipedia, «YCbCr,» 4 Septiembre 2016. [En línea]. Available: https://es.wikipedia.org/wiki/YCbCr.

[14] craig@ccrma.stanford.edu, «WAVE PCM sounfile format,» [En línea]. Available: http://soundfile.sapp.org/doc/WaveFormat/. [Último acceso: 15 Julio 2016].

[15] Zhengzhou Winsen Electronics Technology Co., «Winsen,» 1 Marzo 2015. [En línea]. Available: http://www.winsen-

sensor.com/d/files/PDF/Semiconductor%20Gas%20Sensor/MQ-6%20(Ver1.4)%20-%20Manual.pdf.

[16] Wikipedia, «Conjunto de comandos Hayes,» 15 Abril 2016. [En línea]. Available: https://es.wikipedia.org/wiki/Conjunto_de_comandos_Hayes#AT.

[17] Espressif, «ESP8266EX Datasheet,» Diciembre 2016. [En línea]. Available: http://espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf.

[18] NodeMcu Team, «NodeMcu, Connect Things Easy,» 2014. [En línea]. Available: http://www.nodemcu.com/index_en.html.

[19] Microchip, «MCP23017/MCP23S17,» 2016. [En línea]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/20001952C.pdf.

[20] NXP Semiconductors, «74HC4051; 74HCT4051,» [En línea]. Available: http://www.nxp.com/documents/data_sheet/74HC_HCT4051.pdf.

[21] Surveillance Equipment CCTV Systems, «ELP,» 2004. [En línea]. Available: http://www.elpcctv.com.

[22] Zhengzhou Winsen Electronics Technology Co., «Winsen,» 10 Marzo 2015. [En línea]. Available: http://www.winsen-sensor.com/d/files/PDF/Semiconductor%20Gas%20Sensor/MQ-7B%20(Ver1.4)%20-%20Manual.pdf.

[23] Aosong Electronics Co., «Digital-output relative humidity & temperature sensor/module,» [En línea]. Available: https://cdn-shop.adafruit.com/datasheets/Digital+humidity+and+temperature+sensor+AM2302.pdf.

[24] RobTillart, «DHTlib,» 3 Enero 2017. [En línea]. Available: https://github.com/RobTillaart/Arduino/tree/master/libraries/DHTlib.

[25] Freescale Semiconductor, «3-Axis Orientation/Motion Detection Sensor,» 1 Marzo 2009. [En línea]. Available: http://www.freescale.com.cn/files/sensors/doc/data_sheet/MMA7660FC.pdf?fpsp=1.

[26] ElecFreaks, «3.2" TFT LCD Screen Module,» 13 Mayo 2015. [En línea]. Available: http://www.elecfreaks.com/wiki/index.php?title=3.2%22_TFT_LCD_Screen_Module.

[27] SIMCom, «SIM808_Hardware Design_V1.02,» 20 Marzo 2015. [En línea]. Available: http://simcom.ee/documents/SIM808/SIM808_Hardware%20Design_V1.02.pdf.

[28] SIMCom, «SIM800 Series_AT Command Manual_V1.09,» 3 Agosto 2015. [En línea]. Available:

http://simcom.ee/documents/SIM808/SIM800%20Series_AT%20Command%20Manual_V1.09.pdf.

[29] Maxim Integrated, «Extremely Accurate I2C-Integrated RTC/TCXO/Crystal,» Marzo 2015. [En línea]. Available: https://datasheets.maximintegrated.com/en/ds/DS3231.pdf.