CAMPUS
DE EXCELENCIA
INTERNACIONAL

Graduado en Ingeniería Informática

Universidad Politécnica de Madrid

Escuela Técnica Superior de
Ingenieros Informáticos

TRABAJO FIN DE GRADO

Dispositivo de bajo coste para el control remoto y monitorización para cámara de acción vía WiFi.

Autor: Eva Mata Celaya

Director: Antonio Pérez Ambite

MADRID, JUNIO 2017

“One never notices what has been done;

one can only see what remains to be done…”

Marie Curie (1867-1934)

“Uno nunca se da cuenta de lo que se ha hecho;

Sólo se puede ver lo que queda por hacer…”

Marie Curie (1867-1934)

Agradecimientos

Me gustaría darles las gracias a mis padres por darme la oportunidad de estudiar lo que quería, por su paciencia infinita y su apoyo constante durante toda mi vida. **Sin vosotros nada de esto sería posible.**

Seguidamente darle las gracias a mi tutor, Antonio, por darme la oportunidad de hacer un trabajo tan especial para mí.

Gracias.

Resumen

El proyecto tiene como principal objetivo la realización un dispositivo bajo coste de control remoto y monitorización vía Wifi de una cámara de acción, por lo que con el mismo dispositivo podremos controlar el disparador de la cámara, ya sea para grabar video o bien para realizar fotografías. Además, el dispositivo contará con un servidor web que nos permitirá disfrutar el contenido de la cámara, ya este almacenado o en tiempo real.

El proyecto queda por tanto claramente enmarcado en el concepto de IoT o Internet of Things, conocido en castellano como el internet de las cosas, que es el que parece ser el concepto más revolucionario de nuestros tiempos, más incluso que el paso a la era digital. Se calcula que para 2020 haya 25000 millones de dispositivos conectados, para lo que será necesario realizar ciertos cambios a distintos niveles, como a nivel de comunicación dando el salto de IPv4 a IPv6, a nivel de seguridad mejorando ciertos aspectos relativos a ella o a nivel de usuario teniendo que renunciar a ciertos aspectos de la privacidad. No nos centraremos en los cambios que se necesitarán para la implantación de tal cantidad de dispositivos de IoT, sino en la definición del concepto de IoT. Al ser un concepto nuevo, no existe una única definición en la que los expertos coincidan, por lo que definiremos las dos definiciones que más se usan:

- Un dispositivo IoT es todo dispositivo capaz de conectarse y de obtener datos de su entorno.
- Un dispositivo IoT es todo dispositivo capaz de conectarse, obtener datos de su entorno y de tomar decisiones propias, en función de los datos obtenidos previamente.

Como se ve las definiciones difieren y entran en controversia, motivo por el que el proyecto solo lo enmarcaremos en el concepto en general de IoT, ya que algunos expertos considerarían el dispositivo que se va a desarrollar en este proyecto como 100% dispositivo IoT, ya que se va a conectar y va a obtener datos de su entorno, en formato de imágenes, ya sean fotografías o videos. Sin embargo, otros expertos no lo considerarían así, ya que no tiene una lógica propia para la toma de decisiones en función de esos datos obtenidos.

Abstract

The main objective of the project is the implementation of a low-cost remote and monitoring control device via Wi-Fi of an action-cam, so user would be able to control the camera trigger, either recording video or taking photographs. In addition, the device will have a web server that will allow us to enjoy the content of the camera, whether it is stored or real time.

The project is therefore clearly framed in the concept of IoT or Internet of Things, which is what seems to be the most revolutionary of our times, even more than the step into the digital age. It is estimated that for 2020 there are going to be 25billion connected devices, for which it will be necessary to make certain changes at different levels, such as the level of communication, making the leap from IPv4 to IPv6, at security level it would be necessary to improve certain aspects related to it or at user level giving up with certain aspect of privacy. We will not focus on these changes that will be required for implementation such amount of IoT devices, otherwise we will talk about IoT definition. Being such a new concept, it is not an unique definition in which the experts agree, therefore we define the most used:

- An IoT device is any device able to connecting and obtaining data from its environment.
- An IoT device is any device able to connecting, obtaining data from its environment and making its own decisions, depending on the data obtained previously.

As you can see, the definitions differ and come into controversy, which is why the project will only frame the concept of IoT, as some expert would consider the device to be developed in this project its 100% an IoT device, since it will connect and will get data about your environment, in the form of images, be they photographs or videos However, other experts would not consider it in that way, since it does not have its own logic for making decisions based on these data obtained.

Índice

Índice de figuras

Índice de tablas

1 INTRODUCCIÓN

Vivimos en un mundo conectado, lleno de redes sociales y de comunicaciones. Tuvimos un cambio histórico, el cambio de la era analógica a la digital, y parece que vamos a ser testigos de un segundo cambio aún más importante. Vamos a ver el comienzo del IoT y su crecimiento. Pero primero de todo, qué es eso del IoT.

IoT o Internet Of Things, en castellano conocido como el internet de las cosas se refiere a la interconexión de las cosas a internet, pero qué cosas, en cuanto a cosas se refiere a cualquier objeto del cual podamos obtener datos y tomar decisiones en función de dichos datos, ya sea de forma manual o autónoma. Y es en este último punto en el que difieren los expertos a la hora de dar una definición concreta sobre IoT. Pasemos a ver ambas definiciones:

- Un dispositivo IoT es todo dispositivo capaz de conectarse y de obtener datos de su entorno.
- Un dispositivo IoT es todo dispositivo capaz de conectarse, obtener datos de su entorno y de tomar decisiones propias, en función de los datos obtenidos previamente.

[1]Esta no es la única dificultad que va a encontrar IoT ya que para su implantación 100% en la sociedad van a tener que llegar cambios importantes a nivel tecnológico y sociológico.

En primer lugar, como ya todos sabemos a nivel de comunicaciones estamos usando un protocolo de comunicación que tiene unas limitaciones en cuanto al número de IP disponibles, IPv4. Se calcula que para 2020 haya 25000 millones de dispositivos conectados, con IPv4 tenemos 4,294,967,296 posibles IPs por lo que, para las previsiones de crecimiento, teniendo en cuenta que se cumplan, de dispositivos conectados se nos queda bastante corta en un plazo corto, mientras que con IPv6 tendremos $3.40x10^{38}$ posibles IPs una cifra muy superior a la estimación de crecimiento. No es el único problema dentro del campo de las comunicaciones, la cobertura de las redes existentes de comunicaciones va a tener que mejorar para la completa implantación de IoT

El segundo problema surge del lado de la seguridad de las comunicaciones y de los datos, ya que actualmente únicamente se protegía frente ataques los equipos más comunes, es decir, computadoras ya fueran de uso doméstico, industrial o empresarial, o algunos dispositivos móviles como tabletas o smartphones, pero son pocos los dispositivos del tipo vehículos, electrodomésticos, wearables, etc. que están actualmente protegidos.

El tercer punto crítico para su implantación es la carencia de estándares a la hora de realizar las comunicaciones, al ser algo nuevo y en auge todavía no se han determinado unos estándares para la gestión, tramitación y almacenamiento de los datos recopilados.

El quinto y último de los puntos a tratar, y el que quizás sea el más importante ya que este puede determinar el éxito o no de esta implantación ya que está muy relacionado con los usuarios de los dispositivos IoT, es un punto más social o de mentalidad de usuario, ya que toda esta gran recopilación de datos de carácter personal va a implicar de forma directa tener que renunciar a la privacidad esos datos personales, ya sean datos del tipo biométricos, de consumo, de ubicaciones, etc. Toda esta información es muy valiosa tanto para fines comerciales como para fines menos lícitos, y es en este punto es donde entra el juego el usuario y la seguridad que proporcionen las empresas de IoT, ya que en mi opinión es un punto clave para la implantación.

Este proyecto estará enmarcado en el ámbito del IoT, diremos enmarcado ya que como hemos visto no tenemos una definición única. Tiene como principal objetivo la realización un dispositivo bajo coste de control remoto y monitorización vía Wifi de una cámara de acción, por lo que con el mismo dispositivo podremos controlar el disparador de la cámara, ya sea para grabar video o bien para realizar fotografías. Además, el dispositivo contará con un servidor web que nos permitirá disfrutar el contenido de la cámara, ya este almacenado o en tiempo real.

1.1 Motivación

Este proyecto tiene varias motivaciones, una de ellas y la que quizás me llevo a tomar esta decisión fue la posibilidad de llegar a desarrollar yo sola un dispositivo partiendo de cero, solamente con una idea y finalmente tener físicamente dicho dispositivos funcionando. Creo que es una buena forma de finalizar una carrera, pudiendo comprobar que podemos hacer proyectos de forma autónoma de comienzo a fin.

Otra motivación para realizar este tipo de proyecto son mis ambiciones profesionales, ya que me quiero dedicar al sector del IoT, sector en auge, puntero y que parece ser algo inminente.

Esto no habría sido posible sin el acercamiento a este mundo por parte de una asignatura muy concreta de la carrera, “Informática Industrial”.

1.2 Objetivos

Una vez introducido el tema y desarrollada mis motivaciones personales por las cuales he elegido este proyecto. Pasaremos a definir los objetivos a alcanzar:

- Diseñar e implementar de un dispositivo funcional para activar de forma remota la captura de video e imagen en una cámara de acción. De forma que al menos tenga las mismas funcionalidades que el dispositivo de control remoto original.
- Implementar un servidor web que nos permita la visualización de contenido en tiempo real del video y de las imágenes almacenadas en la cámara.

1.3 Antecedentes

[2]El primer antecedente a mencionar son los controles remotos oficiales de la propia marca de la cámara de acción, que tienen en su página Web desde hace 5años (4/6/2012) [3]. El mando ha ido ganando funcionalidades con las siguientes versiones,

que se ha ido adaptando a las mejoras propuestas por la marca para sus cámaras. Actualmente la marca ha apostado por integrar el control remoto a la propia cámara haciendo uso del reconocimiento por voz del disparador, aunque también está comercializando un disparador sencillo con conexión Wifi para aquellos que no puedan o quieran usar dicha funcionalidad.

Los siguientes antecedentes son modelos no oficiales que han realizado la gente por su cuenta. [4]El más relevante para mí, por su parecido a mi proyecto, es el realizado por un chico alemán, el cual usa el mismo microcontrolador, el ESP8266-e12, y ha realizado varias versiones, aunque la definitiva no tiene pantalla y es únicamente un disparador Wifi.

Los siguientes modelos no es que sean menos relevantes, ya que entiendo que por la documentación leída cumplen con su cometido, pero uno está realizado con [5]un Arduino Genuino y otro con [6]un Arduino Yun.

2 DEFINICIÓN DE REQUISITOS DEL SISTEMA

[7]Para poder determinar las funcionalidades a desarrollar en el proyecto necesitamos definir los requisitos del mismo. Para ello definiremos unos requisitos generales a la cámara en su conjunto y unos requisitos por modo de funcionamiento, la cámara tiene 6 modos de funcionamiento:

- Requisitos generales
- Requisitos del modo cámara de fotos
- Requisitos del modo cámara de video
- Requisitos del modo ráfaga de fotos
- Requisitos del modo temporizador de fotos
- Requisitos del modo configuración.

2.1 Requisitos generales

1. Si el Wifi de la cámara está encendido, la cámara debe poder ser encendida y apagada desde el mando.
2. (Requisito opcional) Se debe poder controlar más de una cámara con un único mando, sin exceder el límite de 50 cámaras por mando.
3. El mando debe ser un servidor web a través del cual se podrá acceder al contenido de la cámara usando un navegador web:
    a. Visualización de imágenes capturadas
    b. Visualización de video capturados
    c. Visualización de configuración
4. El servidor web debe permitir descargar el contenido almacenado en la cámara.
    a. Descarga de imágenes capturadas
    b. Descarga de video capturado
5. El dispositivo permitirá controlar la cámara vía IP, por lo que podremos tener la cámara en otra ubicación y acceder al contenido haciendo uso del servidor web.
    a. Visualización en tiempo real de video.

2.1.1 Requisitos del modo cámara de fotos

6. El modo cámara de fotos permite capturar imágenes si apretamos el botón de captura, en el modo que este seleccionado y con la resolución seleccionada.

2.1.2 Requisitos del modo cámara de video

7. El modo cámara de video permite capturar video si apretamos el botón de captura una vez, en el modo que este seleccionado y con la resolución seleccionada, y si apretamos una segunda vez el botón de captura finalizaremos la captura de video.

2.1.3 Requisitos del modo ráfaga de fotos

8. El modo ráfaga de fotos permite capturar un numero de imágenes en un intervalo de tiempo si apretamos el botón de captura, en el modo que este seleccionado y con la resolución seleccionada.

   Ejemplo: 10/1s →la cámara capturará 10 imágenes en 1 segundo.

2.1.4 Requisitos del modo temporizador de fotos

9. El modo temporizador de fotos permite capturar una imagen cada cierto intervalo de tiempo de forma indefinida el botón de captura una vez, en el modo que este seleccionado y con la resolución seleccionada, y si apretamos una segunda vez el botón de captura finalizaremos la captura.

   Ejemplo: 1s →la cámara realizará una captura cada segundo que pase de forma indefinida hasta que volvamos a pulsar el botón de capturar.

2.1.5 Requisitos del modo configuración

10. (Requisito opcional) El modo configuración nos permite cambiar ciertos aspectos de la configuración a la hora de capturar imágenes:
    a. Resolución de video, FPS (frames per second) y FOV (Field of View) → permite elegir entre distintas configuraciones en cuanto a resolución, FPS y FOV.
    b. Resolución de fotos → permite elegir las distintas resoluciones en MP (Mega Pixeles), esto afectará a los modos de cámara de fotos, modo ráfaga de fotos y modo temporizador de fotos.
    c. Configuración del temporizador → permite elegir los intervalos de tiempo para el modo temporizador de fotos.
    d. Configuración del menú de capturas
        i. Orientación de la cámara → permite elegir la orientación de la cámara de forma que si colocas la cámara boca abajo no tengas que editar los videos posteriormente para verlos del derecho.
        ii. Spot Meter o exposímetro puntual → permite hacer uso de la cámara desde un lugar oscuro y grabar una zona luminosa con mayor calidad.
        iii. Looping Video o grabación en bucle → permite grabar en bucle y únicamente guardar los mejores momentos, ahorrando espacio de almacenamiento.
    e. Menú de configuración
        i. Modo predeterminado al encender la cámara → permite elegir cuál es el modo predeterminado al arrancar la cámara.
        ii. Un solo botón → permite, si está activo, comenzar a grabar nada más encender la cámara.

iii. NTSC/PAL → permite controlar la velocidad de fotogramas de grabación de video y a reproducción cuando se visualiza un video en un televisor normal o de alta definición.
iv. Presentación en pantalla u OSD→ permite mostrar u ocular los iconos de grabación y la información de archivo sobre el video de la pantalla de visualización durante la reproducción.
v. Luces indicadoras de estado de la cámara → permite elegir el estado de las luces indicadoras de la cámara.
vi. Indicador de sonido → permite ajustar el volumen del indicador de sonido.
vii. Apagado manual → permite configurar la cámara para que se apague automáticamente tras un periodo especifico de inactividad.
viii. Mes/Día/Año/Hora → Permite ajustar manualmente la fecha y hora.

f. Controles de la conexión inalámbrica → permite activar y desactiva la conexión inalámbrica.
g. Eliminar → permite eliminar archivos o formatear la tarjeta de almacenamiento.
h. Exit → permite salir del modo de configuración.

3 ARQUITECTURA DEL SISTEMA

De cara a satisfacer los requisitos del sistema y alcanzar los objetivos, se propone una solución compuesta de hardware y software, lo que nos dará lugar a un dispositivo capaz de satisfacerlos haciendo uso de la tecnología WiFi.

El dispositivo va a contar con dos botones que nos van a permitir realizar las acciones (encender y apagar la cámara, cambiar de modo y activar o desactivar la captura de imágenes o videos) sobre la cámara y una pantalla que va a permitir al usuario saber en qué modo se encuentra la cámara y si se están realizando o no acciones sobre la misma.

El dispositivo va a generar un servidor web para el usuario con su propia IP, la cual podrá visualizar en todo momento en la pantalla de su dispositivo, únicamente deberá introducir dicha IP en la sección URL de su navegador preferido, y tendrá acceso al contenido de su cámara, tanto lo almacenado como en tiempo real, y además podrá realizar todas las acciones que se puede realizar con su dispositivo, pero en este caso desde su navegador.

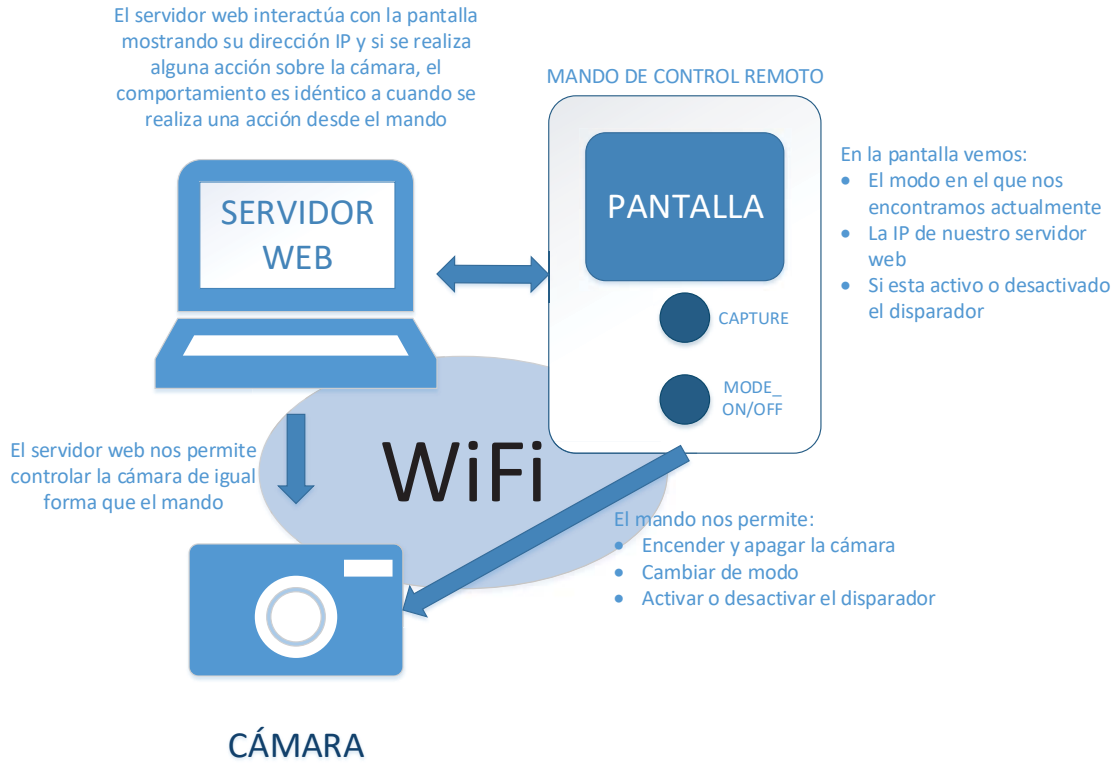


Figura 1 – Arquitectura del sistema

Como la propuesta para el proyecto es una solución compuesta de hardware y de software, en los siguientes capítulos veremos las arquitecturas correspondientes a cada una de esas partes.

4 DISEÑO DEL SISTEMA

4.1 Desarrollo del hardware

4.1.1 Arquitectura hardware

A nivel de hardware podemos dividir la arquitectura del mando en los siguientes grandes bloques:

- Bloque de comunicación, es el "cerebro" del proyecto, aquel que se va a encargar de realizar y gestionar las distintas acciones que se van a realizar sobre la cámara, va a comunicarse con la pantalla para ir refrescando correctamente la información y mostrando las acciones que están teniendo lugar, y por último va a generar y administrar el servidor web. Este bloque corresponde, por tanto, a un microcontrolador que nos permita gestionar todas estas cosas mencionadas con anterioridad haciendo uso de la tecnología WiFi.
- Bloque de accionadores, corresponde a una pequeña botonera a través de la cual el usuario va a interactuar con el dispositivo y consecuentemente con la cámara de acción. Compuesta de dos botones, que nos van a permitir realizar diversas acciones sobre la cámara.
- Bloque de visualización, corresponde a la pantalla, y los conectores necesarios, en la cual podremos observar distintos parámetros de configuración, saber cuántas fotos nos quedan por hacer o la batería restante de nuestro dispositivo.
- Bloque de energía o carga, es el bloque compuesto por la batería, sus conectores, un reductor de tensión que nos facilita la tensión necesaria para alimentar a la unidad microcontroladora y por último, un cargado de baterías.
- Bloque conector, es el bloque encargado de conectar y comunicar el mando de control remoto con nuestro ordenador, de tal forma que podamos cargar el firmware en el mismo, para posteriormente ejecutarlo, para lo cual necesitaremos un conversor de USB a serial UART TTL.

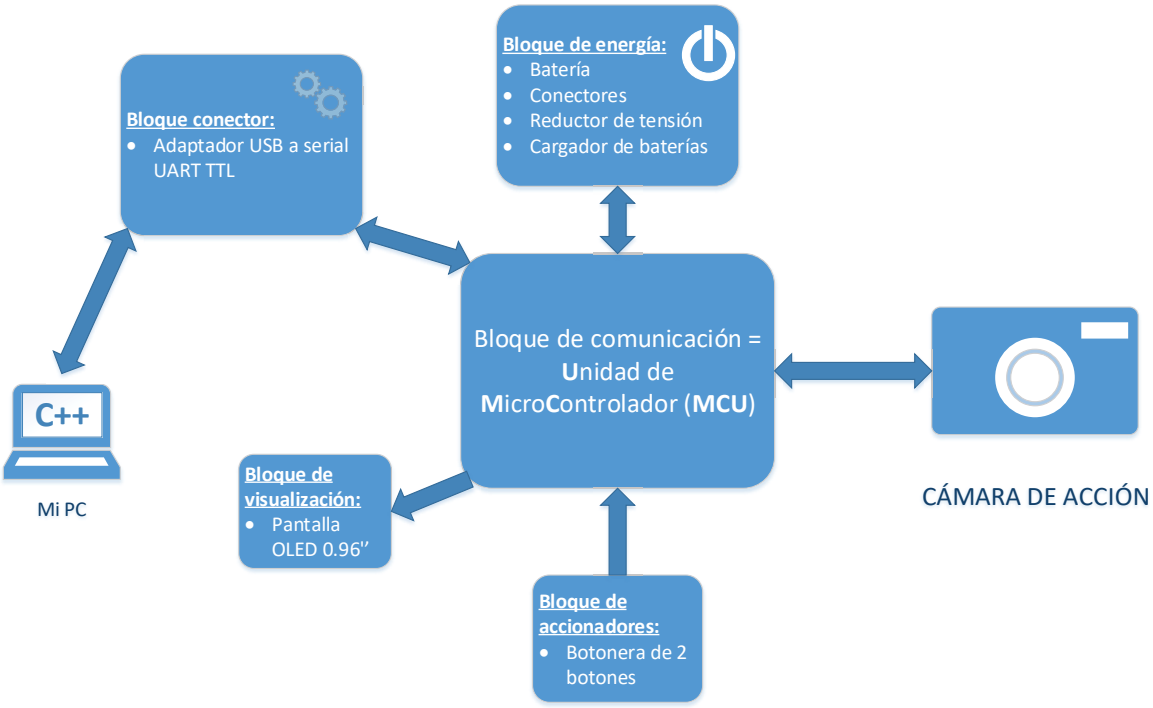


Figura 2 – Diagrama de bloques

4.1.2 Componentes del dispositivo

Los principales componentes que se van a usar en este proyecto van a ser:

- Un módulo de comunicación
- Conversor reductor de tensión
- Cargador de batería micro-USB
- Batería Li-Po
- Una botonera
- Una pantalla
- Conectores de pantalla y de batería
- USB Serial UART

4.1.2.1 Módulo de WiFi

[8]A la hora de elegir el módulo de comunicación hemos mirado tanto el tamaño del dispositivo, como el coste del mismo. [9] [10]Dentro de la gran variedad de módulos Wifi disponibles en el mercado de distintas marcas, hemos optado por elegir tanto por tamaño, como por especificaciones técnicas que necesita el proyecto, como por el tema económico, entre estos dos candidatos de microcontroladores:

- Estos dos microcontroladores son “pequeñas computadoras”, cuentan con una pequeña memoria RAM, memoria Flash, tarjeta de red WiFi y puertos de entrada/salida. La ventaja frente a otros, es que con estos no necesitas lidiar con un sistema operativo.
  [11]A continuación, vemos un esquema de bloques funcionales que los componen:

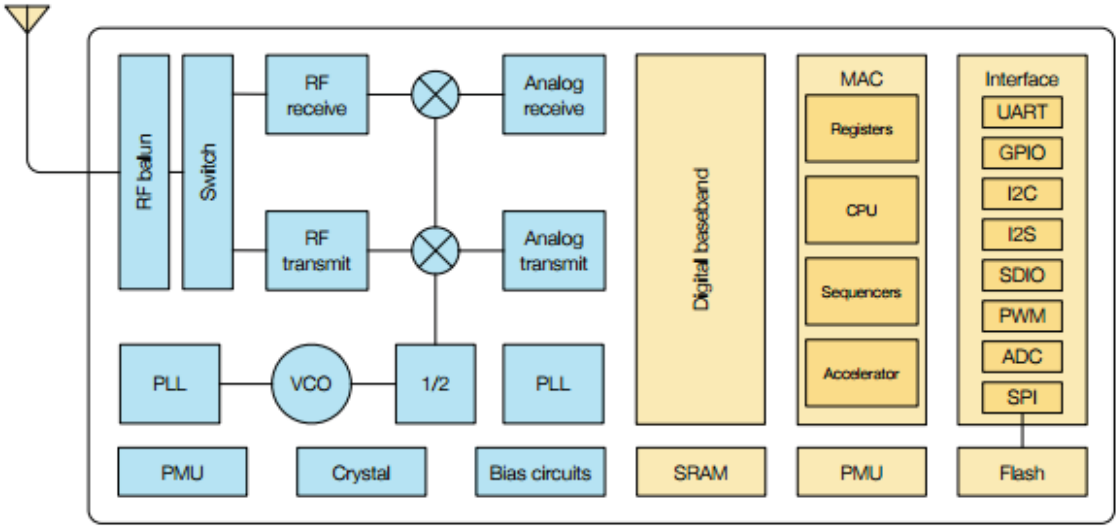


Figura 3 – Diagrama de bloques funcionales que componen los microcontroladores

Especificaciones	ESP8266	ESP32
MCU	Xtensa Single-Core 32bit L106	Xtensa Dual-Core 32bit LX6 600DMIPS
802.11 b/g/n Wi-Fi	Si, HT20	Si, HT40
Frecuencia típica	80MHz	160MHz
SRAM	160kBytes	512kBytes
Flash	SPI Flash, up to 16Mbytes	SPI Flash, up to 16Mbytes
GPIO	17	36
Hardware/Software	0 /8 canales	1 /16 canales
SPI/I2C/I2S/UART	2/1/2/2	4/2/2/2
ADC	10-bit	12-bit
CAN	No	Si
Ethernet MAC Interface	No	1
Touch Sensor	No	1
Temperature Sensor	No	Si
Temperatura de trabajo	-40°C – 125°C	-40°C – 125°C
Precio	[12]1.00€+2.99€ de envio	[13]8.62€ + 1.08€ de envio
Dimensiones	16.0mmX24.0mm (anchoXalto)	18.0mmX25.0mm (anchoXalto)

Tabla 1 – Tabla comparativa de especificaciones de los microcontroladores.

ESP8266-E12:

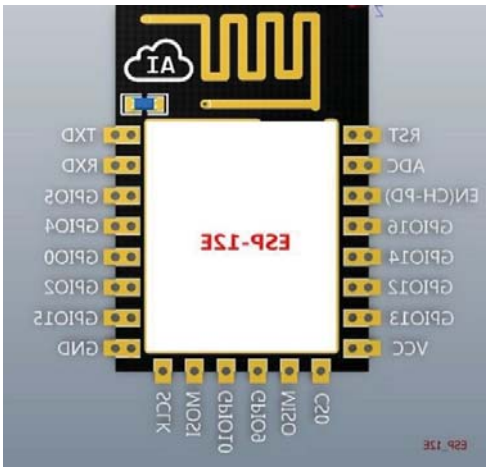


Figura 4 – Microcontrolador ESP8266-12E

ESP32:

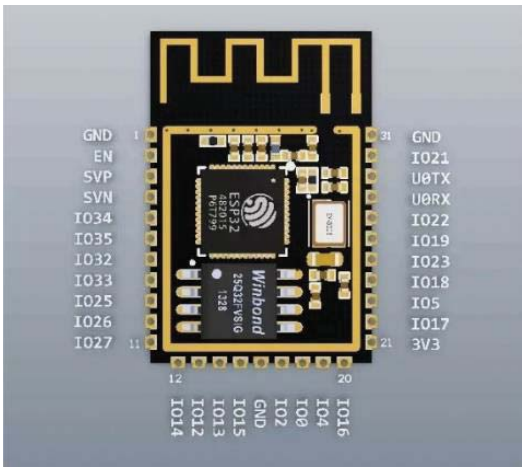


Figura 5 – Microcontrolador ESP32

El modelo ESP32 es la versión mejorada del ESP8266-12E, y aunque tiene unas características muy interesantes de cara a otros proyectos, con el modelo ESP8266-12E tenemos todo aquello que necesitamos: conexión Wifi, un puerto I2C para la pantalla, un puerto analógico para la botonera y un precio muy asequible.

4.1.2.2 Conversor reductor de tensión

[14]El microcontrolador del proyecto, funciona con 3.3V, por lo que, para suministrarle dicha tensión, ya que de lo contrario se quemaría, bajaremos la tensión con la que alimentamos al mismo. Para lo cual hemos seleccionado este conversor reductor de tensión DC-DC.

Figura 6-Conversor reductor de tensión

4.1.2.3 Batería

[15]El consumo de nuestro dispositivo se va a llevar a cabo, casi al 100%, en el microcontrolador. Según las especificaciones del mismo, tenemos un consumo máximo de unos 300mah, por lo que se ha elegido una batería LiPo de 800mah con una tensión de 3.7V, que tarda en cargar entre 50 y 70 minutos. Su peso aproximado es de unos 24.5g. Para cumplir con el requisito de tensión del conversor reductor de tensión

(mínimo necesitamos 4.75V en entrada), pondremos 2 baterías en serie del mismo modelo, en lugar de comprar una de mayor voltaje, ya que tienen unas dimensiones razonables, para un mando que debe caber en una mano, y además son baratas.

Figura 7 – Batería LiPo

4.1.2.4 Cargador de batería

[16]La batería LiPo seleccionada deberá ser cargada, para ello se ha seleccionado un dispositivo que nos permite cargar la batería vía micro USB.

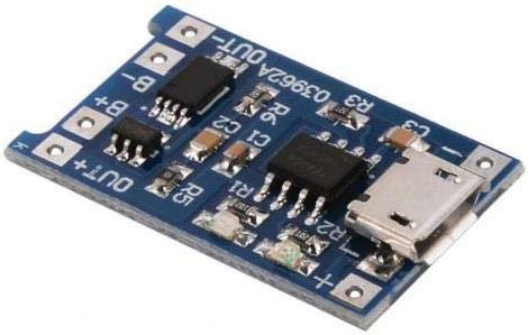


Figura 8 – Cargador de batería

4.1.2.5 Conectores para el conversor USB a TTL-UART

Para poder conectar de forma sencilla y cómoda el conversor USB a TTL-UART, se van a usar pines hembra estándar:

Figura 9 – Conectores para el conversor USB a TTL-UART

4.1.2.6 Conectores para la batería

[17]Para poder conectar de forma segura y cómoda la batería LiPo, [18] se ha elegido un conector crimps macho hembra de 2 pines:

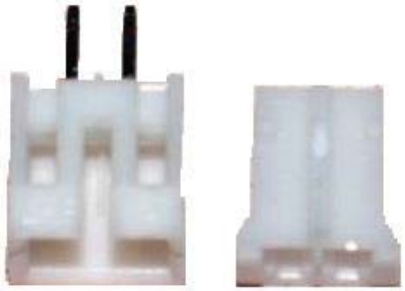


Figura 10 – Conectores para la batería

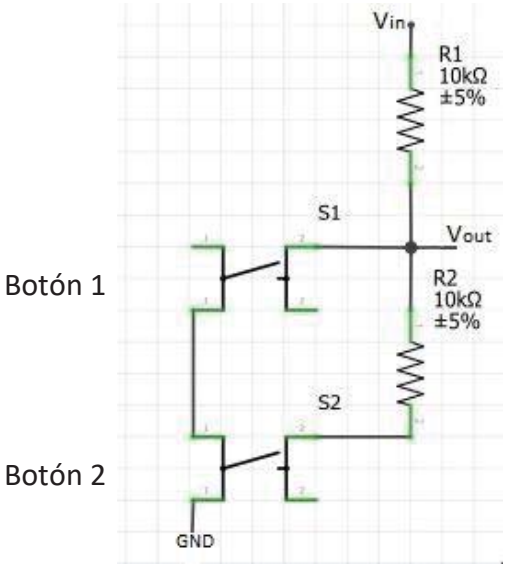
4.1.2.7 Conversor de USB a TTL-UART

[19]Será necesario un conversor USB a puerto serial TTL-UART, poder cargar el firmware en el dispositivo de forma sencilla, por lo que dejaremos 4 conectores libres para poder acceder de forma sencilla al puerto RXD, TXD, 3.3V y GND. Es importante recalcar que dicho conector no suministrará más que 3.3V de tensión al dispositivo, ya que como hemos comentado ya, el suministrar más de 3.3V podría estropear el microcontrolador.

Figura 11 – Conversor USB a TTL-UART

4.1.2.8 Botonera

La botonera se ha desarrollado usando un divisor de tensión. El divisor de tensión nos dará en un mismo punto eléctrico, V_{out}, distintas lecturas de tensión, por lo que necesitaremos conectar V_{out}al pin analógico en nuestro microcontrolador si queremos leer los valores del divisor. Para que el divisor de tensión funcione correctamente, y podamos leer los distintos valores de tensión en salida, vamos a necesitar dos resistencias de 10kΩ cada una, de esta forma podremos detectar fácilmente 3 estados: 0V (botón 2 pulsado), $½V_{in}$(botón 1 pulsado) y V_{in}(ningún botón pulsado). Veamos a continuación el esquema eléctrico:

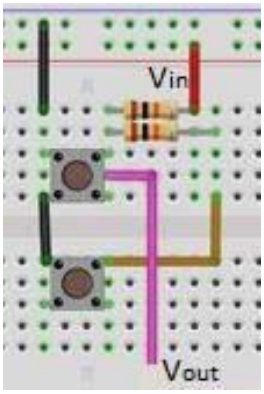


Botón 1

Botón 2

Figura 12 – Esquema unifilar de la botonera

Así quedaría montado nuestro divisor de tensión o botonera, en una protoboard:

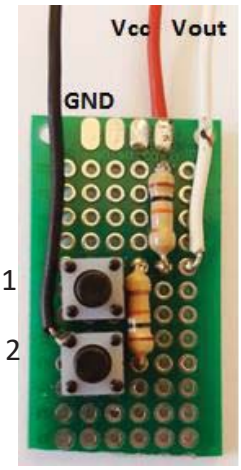


Botón 1 o Botón de Captura

Botón 2 o Botón de ON/OFF/Config

Figura 13 – Esquema de conexión de la botonera en una protoboard

Y así, soldada de forma casera en un PCB estándar perforado:



Botón 1

Botón 2

Figura 14 – Botonera soldada en un PCB estándar perforado

Para que la botonera propuesta sirva para nuestro prototipo PCB, hemos necesitado realizar unas pequeñas adaptaciones, para adaptar el rango de tensiones que genera la botonera (entre 0V y 3.3V) al rango de tensiones que utiliza el convertidor analógico digital del ESP8266, lo que significa la siguiente modificación respecto de la original, que es agregar un divisor de tensión formado por 2 resistencias de 220kΩ y de 100kΩ de la siguiente manera:

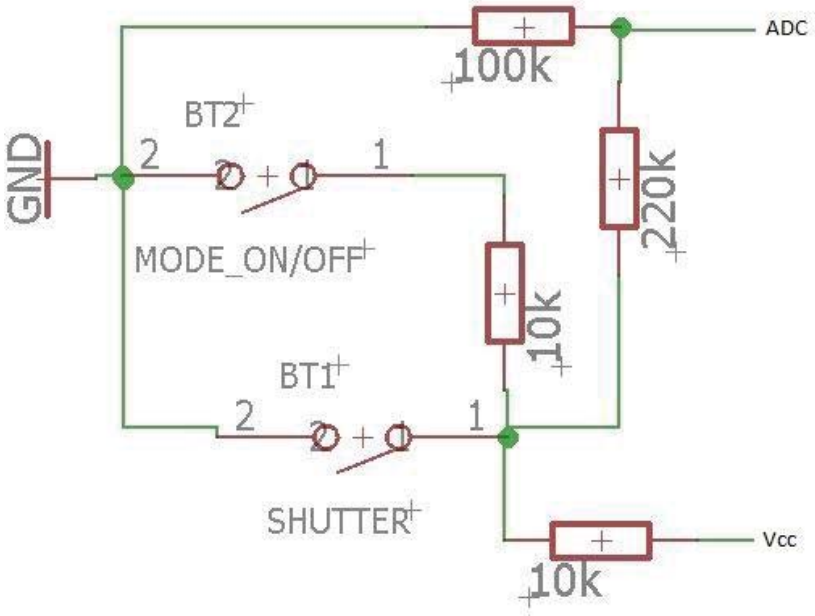


Figura 15 – Esquema eléctrico de la botonera para el prototipo PCB

Y así, soldada de forma casera en un PCB estándar perforado:

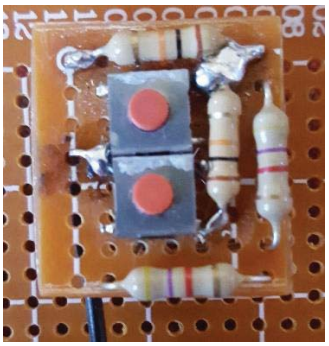


Figura 16 – Botonera para el prototipo PCB

4.1.2.9 Pantalla

[20]La pantalla va a ser una pantalla OLED de 0.96'' con conexión I2C, en la que podremos visualizar la información relevante de la cámara, como: capturas realizadas, estado de la batería, minutos de grabación, etc.

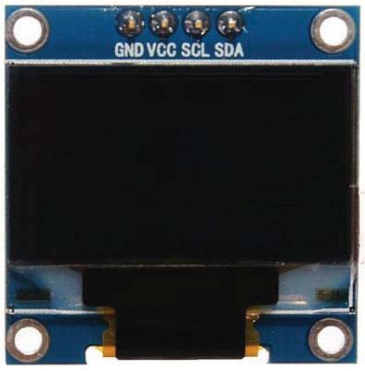


Figura 17 – Pantalla OLED de 0.96'' – 128x64pixeles

El tamaño de la pantalla, aunque pequeño, nos permite visualizar sin problemas la información importante y además nos permite reducir el tamaño del dispositivo, a fin de mantener un tamaño de dispositivo que nos permita manejar el mismo con una única mano.

4.1.3 Comunicación usada por la cámara

[21] La comunicación usada por la cámara es vía Wifi como hemos comentado anteriormente, y para poder acceder a los datos de la cámara, la propia marca de la cámara de acción que voy a usar en este proyecto, ofrece un SDK con su API para que los desarrolladores puedan hacer y probar sus proyectos, por desgracia la marca no ofrece este servicio de forma gratuita, por lo que se usará un repositorio de la comunidad de GitHub [22], que mediante ingeniería inversa ha creado su propia API para poder realizar proyectos similares al mío, y que está en continuo desarrollo. Al no ser una cámara de las más actuales, vamos a hacer uso de la parte del repositorio más relativamente estable, ya la fecha de última modificación de dicha parte es de 2012, esto no significa que todo vaya a funcionar al 100%.

Este repositorio recopila las peticiones HTTP que la cámara realiza para comunicarse con las aplicaciones móviles u otros dispositivos, esto quiere decir, que si conectamos nuestro microcontrolador a la cámara y lanzamos una de estas peticiones HTTP podremos hacer que la cámara realice acciones y obtener datos de la cámara, a través de la respuesta a la petición.

4.1.4 Herramientas usadas

A la hora de realizar los diseños tanto a nivel eléctrico, como a nivel de prototipado se han usado dos herramientas muy conocidas:

- [23]Fritzing, es un programa de software libre de automatización de diseño eléctrico que permite pasar los prototipos a productos finales. A pesar de que

este mismo programa nos permite pasar los prototipos a circuitos impresos, no es realmente una herramienta muy potente en este aspecto y viene bastante limitada, sin embargo, nos permite pasar nuestros prototipos de forma muy visual al formato digital para documentarlos y además realizar los esquemáticos de los mismos. Su uso es más de un ámbito educativo, más que profesional, debido a las limitaciones en los esquemático y al paso a circuitos impresos.

- [24]EAGLE, no nos permite realizar el diseño en protoboard de la forma tan visual que lo hace Fritzing, sin embargo, tiene otras características más interesantes. Es una herramienta muy potente y versátil a la hora de realizar diseño de esquemáticos ya que podemos no solo elegir encapsulados de las librerías incluidas en por la herramienta, sino que podemos crearnos nuestros propios encapsulado y crearnos nuestras librerías, además la gran mayoría de vendedores de material electrónico (SparkFun, Adafruit, Arduino o Dangerous Prototypes) ofrecen librerías con los encapsulados que tienen a la venta en sus tiendas. Además de permitirnos el paso a circuitos impresos, la herramienta tiene un algoritmo de optimizado para calcular distintas rutas para el mismo distinto circuito impreso. Todas estas características permiten un acabado mucho más profesional a la hora de realizar los proyectos.

4.1.5 Diseño de esquemáticos

De cara a este proyecto hemos planteado dos esquemáticos que difieren un poco entre sí. Por un lado, tenemos el esquemático correspondiente al prototipo implementado en una placa de pruebas o protoboard, y por otro lado tenemos el esquemático correspondiente a un prototipo para llevar a un circuito impreso o una placa PCB estándar perforada.

Tenemos dos diseños de circuitos, los cuales van a diferir un poco. El diseño en protoboard se va a montar en una protoboard, junto con una pantalla, una botonera y un módulo NodeMCU, que trae integrado: el módulo ESP8266, el conversor, el micro USB UART TTL y todas las soldaduras hechas. Sin embargo, el diseño del prototipo en PCB, va a tener que incluir por separado las partes que trae integradas el NodeMCU.

4.1.5.1 Diseño del prototipo en protoboard

Este primer esquemático está desarrollado en una protoboard o una placa de pruebas, usando el microprocesador el ESP8266-12E en un formato especial para pruebas, ya que viene soldado y preparado para insertarlos en la protoboard, de esta forma trae ya incluido en ese mismo pre-montaje el regulador de tensión y todas las conexiones necesarias para poder usarlo de forma “pulg&play” (conecta y juega en castellano).

Este primer esquemático se ha diseñado usando la herramienta Fritzing, ya que traía incluidos todos los módulos, incluida la protoboard.

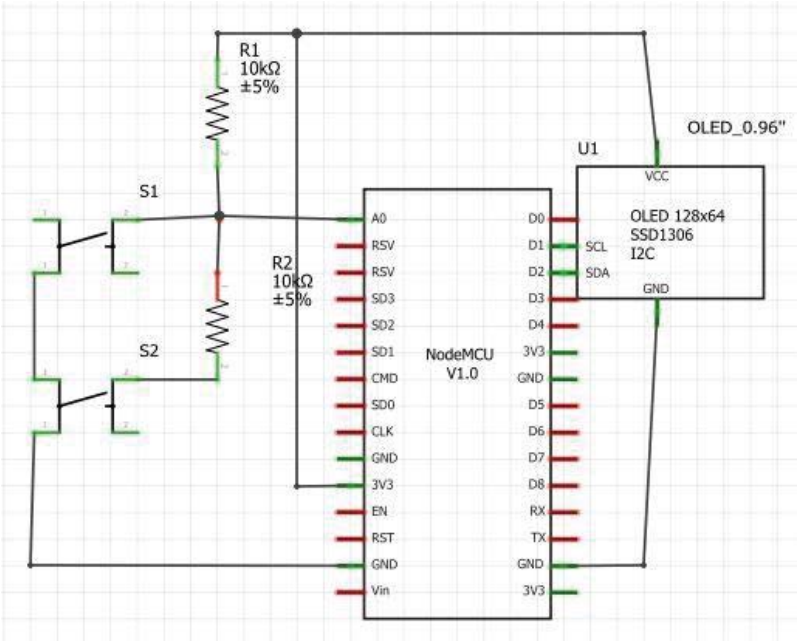


Figura 13 – Esquema eléctrico del prototipo en protoboard

4.1.5.1.1 Componentes específicos para el prototipo

A continuación, pasaremos a ver los dos componentes característicos de este esquemático para el prototipo de pruebas del proyecto.

4.1.5.1.1.1 Protoboard

Una protoboard es un tablero con orificios donde se pueden insertar los componentes electrónicos para poder realizar pruebas. La forma en la que están conectados estos orificios de la siguiente:

- Los nodos de conexión están unidos verticalmente, es decir, toda la vertical de puntos que corresponda a cada línea morada serán el mismo punto de conexión y tendrán, por tanto, el mismo valor de tensión. La protoboard tiene dos nodos de conexión separados por un canal central, que tendrá más o menos puntos de tensión según el tamaño de la protoboard
- El bus de conexión, al contrario que los nodos de conexión, están unidos horizontalmente, por tanto, toda la horizontal de puntos que corresponda a cada línea verde será el mismo punto de conexión y tendrán, por tanto, el mismo valor de tensión. La protoboard tiene 4 buses de alimentación,2 a cada extremo de la misma, típicamente corresponde respectivamente dos a dos a alimentación y masa.

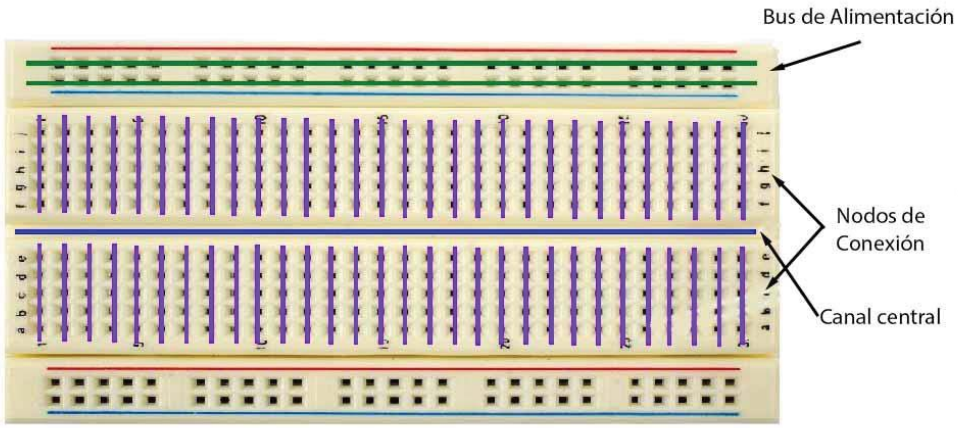


Figura 18 – Protoboard con su explicación de uso y funcionamiento

4.1.5.1.1.2 NodeMCU

El NodeMCU es una placa de desarrollo concebida para realizar pruebas con el módulo ESP8266-e12 sin tener que soldarlo en ninguna parte ya que el mismo modulo ya trae un conector micro USB para alimentar el dispositivo y conectarlo con un ordenador, así como el módulo conversor reductor de tensión. Al estar todo ya soldado únicamente tenemos que hacer uso de plug&play del dispositivo:

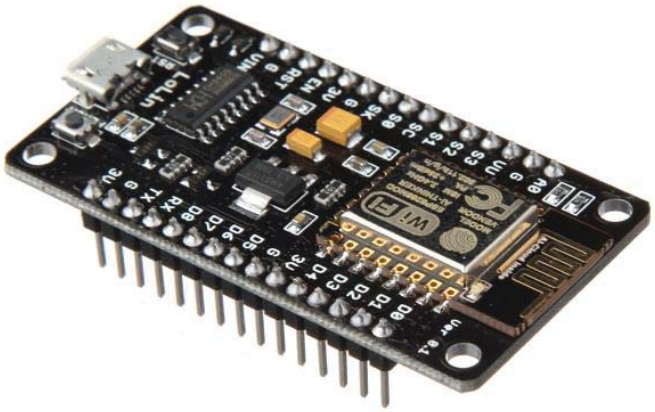


Figura 19 – Placa de desarrollo NodeMCU

Este tipo de módulos facilitan mucho la implementación y el desarrollo de proyectos de IoT, ya que permiten hacer pruebas sin la necesidad de estar soldando o rediseñando PCBs.

4.1.5.2 Diseño del prototipo PCB

Este segundo y último esquemático, es válido tanto para la implementación del prototipo en PCB estándar perforado, como, para el circuito PCB integrado, ya que ambos van a llevar los mismos componentes con la diferencia que el soporte donde se van a montar van a diferir y que los componentes que se vayan a montar, deben ser o bien de montaje superficial o SMD (Surface Mount Device) o bien para placas

perforadas. En esta ocasión este esquemático se ha desarrollado usando la herramienta EAGLE ya que muchos de los encapsulados no estaban disponibles en ninguna librería y se han tenido que desarrollar.

Los elementos para los cual hemos necesitado crear los encapsulados han sido los siguientes:

- Conversor de tensión DC-DC

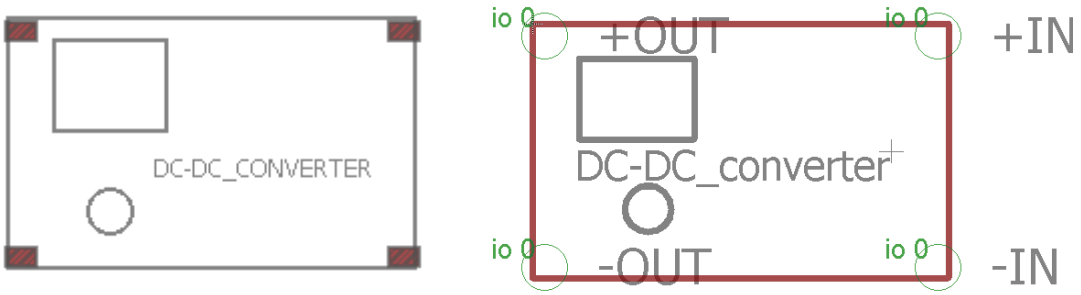


Figura 20 – Diseño del conversor de tensión

- Cargador de baterías LiPo

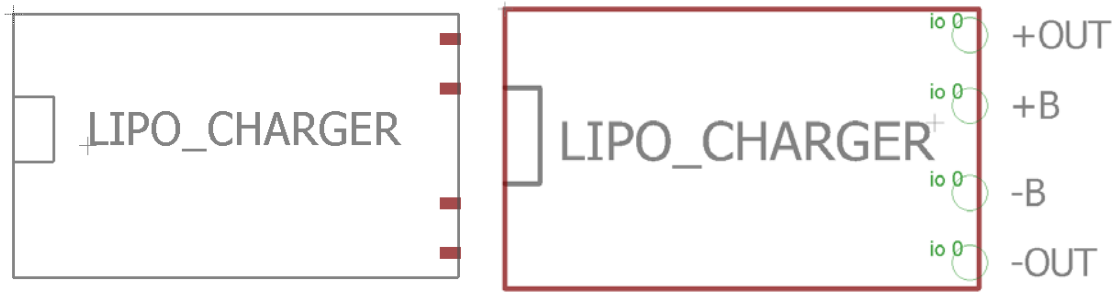


Figura 21 – Diseño del cargador de baterías LiPo

Una vez desarrollados los encapsulados que faltaban, se ha procedido a enlazarlos, dando lugar al siguiente esquemático:

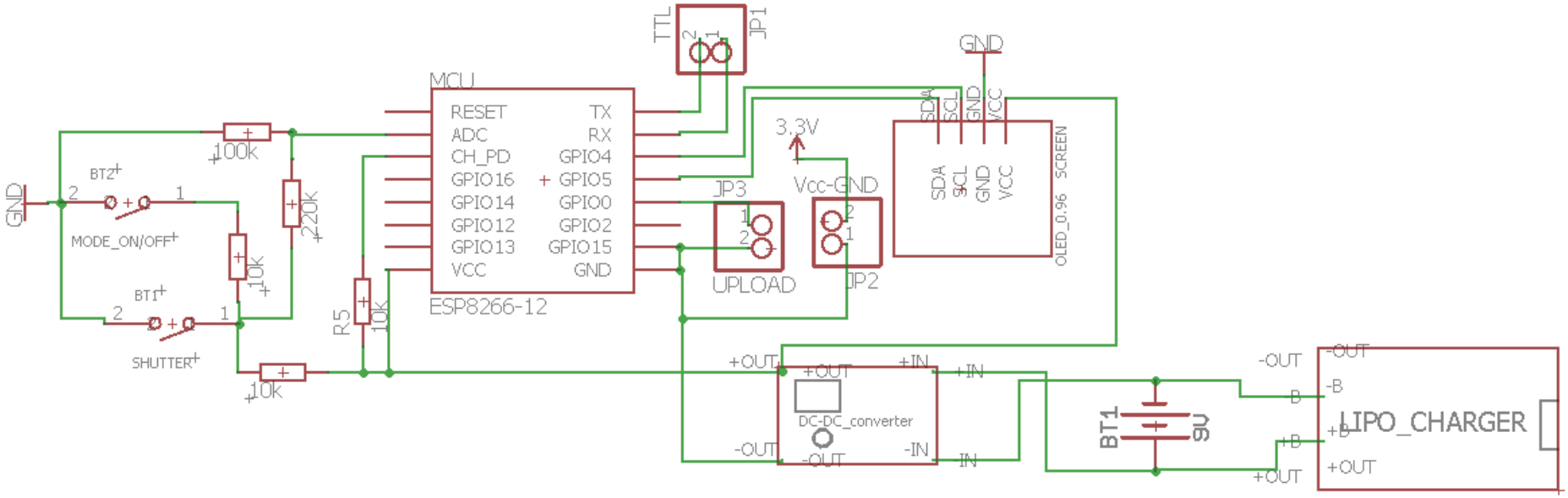


Figura 22 – Esquemático para el prototipo PCB

4.1.6 Prototipos implementados

Como hemos explicado en la sección anterior, a pesar de que únicamente disponemos de dos esquemáticos, sin embargo, estos dan lugar a 3 prototipos que veremos en las siguientes secciones.

4.1.6.1 Prototipo en protoboard

Para poder ir haciendo pruebas sin la necesidad de soldar una y otra vez los materiales, hasta estar seguros de que todo es correcto y funciona sin problemas, vamos a realizar un primer prototipo. Por ello he adquirido la siguiente placa de desarrollo, la NodeMCU. La cual conectaremos en una protoboard, en la que además enchufaremos la pantalla y una botonera "casera".

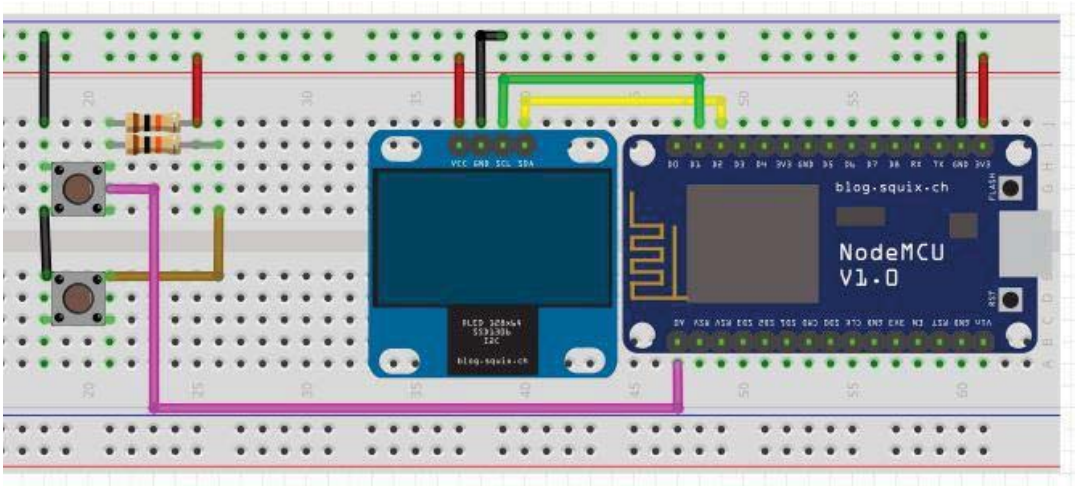


Figura 23 – Prototipo diseñado en una protoboard

4.1.6.2 Prototipo en PCB estándar perforado

A fin de poder mostrar un prototipo un poco más cercano a lo que sería el prototipo final, se ha soldado en un PCB estándar perforado todos los componentes necesarios para tener el 100% de las funcionalidades del sistema. Se han intentado respectar la idea de tener un dispositivo que se pueda manejar con una sola mano, por ese motivo todos los componentes se encuentran concentrados en 10cm^2 y por ambos lados del PCB. En la parte delantera del dispositivo nos encontramos con dos niveles de componentes, en el nivel superior encontramos la pantalla y la botonera, y en el nivel inferior de la parte delantera del dispositivo podemos ver el conversor de tensión, el cargador de batería y el microcontrolador. En la parte trasera se localiza el conector para la pila de 9v. Véase las figuras a continuación adjuntas:

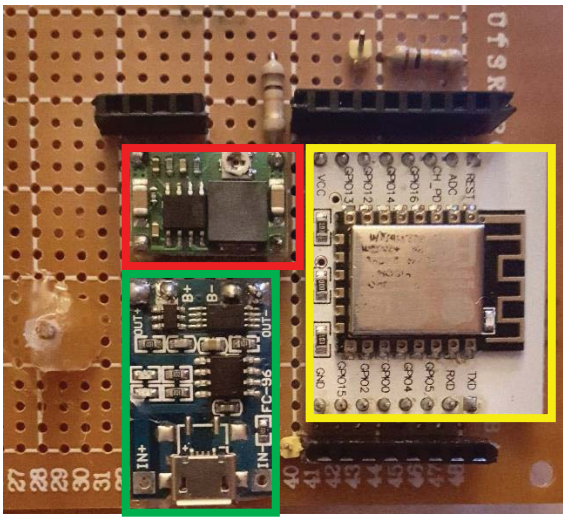


Figura 24 – Prototipo parte delantera parte baja

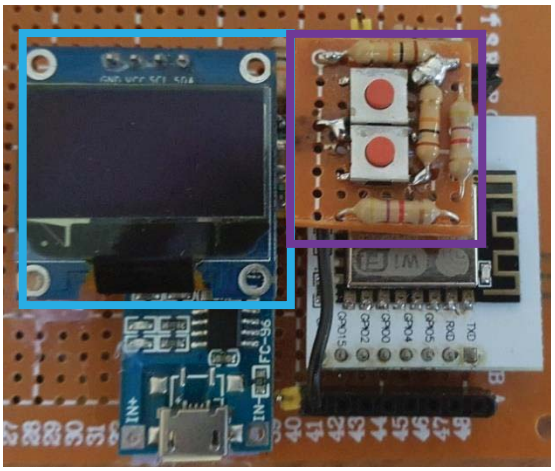
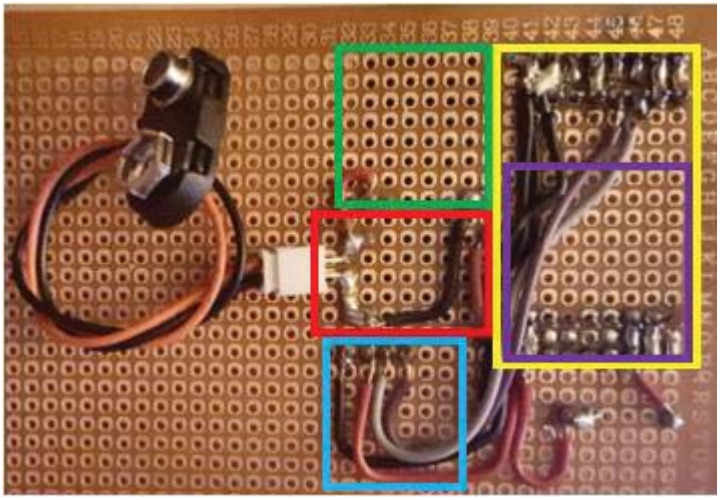


Figura 25 -Prototipo parte delantera parte alta



LEYENDA

MCU
BOTONERA
CONVERSOR DE TENSIÓN
CARGADOR DE BATERÍA
PANTALLA OLED 0.96''

Figura 26 – Prototipo parte trasera

4.1.6.3 Prototipo en PCB integrado

Los prototipos hasta el momento presentados no son útiles de cara a comercializar ningún producto, ya que el tiempo de ensamblaje y de pruebas no lo permiten. Sin embargo, este último prototipo desarrollado con un circuito impreso (PCB) específico, es completamente distinto. Los circuitos impresos específicos permiten realizar proyectos comercializables, ya que se diseñan una sola vez, pero se pueden fabricar tantos como se quiera. No solo eso, hay empresas que también se dedican al soldado de componentes, permitiendo a proyectos de gente amateur tener un acabado muy profesional.

Este prototipo cumple una vez más con la idea de lograr un dispositivo usable con una sola mano. Por lo que el uso de esta tecnología es ideal para ello, nos permite reducir aún más el tamaño y optimizar la distribución de los componentes.

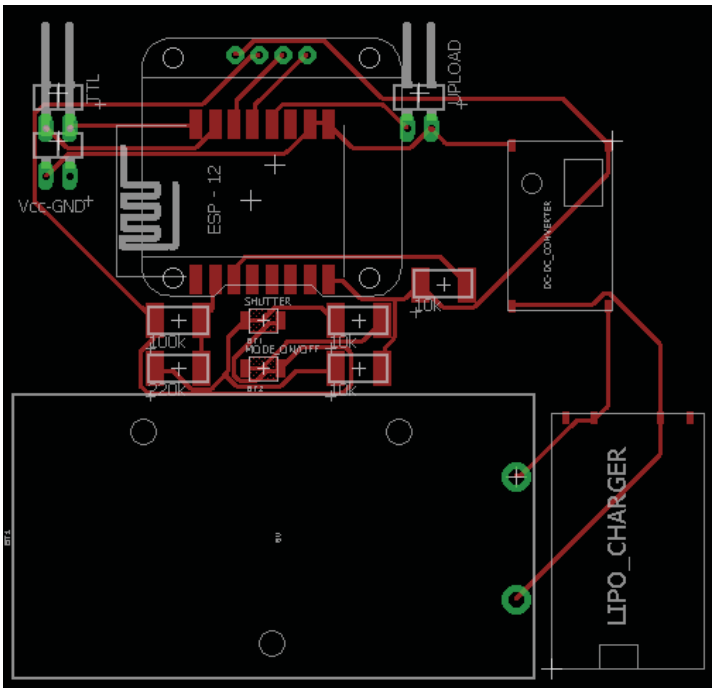


Figura 27 – Circuito impreso del prototipo

4.2 Desarrollo del software

4.2.1 Arquitectura software

La arquitectura software implementada en 3 partes diferenciadas interrelacionadas entre sí de la siguiente manera:

- En primer lugar, tenemos un módulo pantalla que reúne todo lo referenciado a este componente con el sistema, como, dijimos la pantalla muestra los modos de la cámara, las acciones que se están realizando y la dirección IP del servidor web, por tanto, el módulo está relacionado tanto con el módulo del servidor web para obtener la dirección IP del mismo, como, con el módulo de control remoto que nos facilita la información relativa al modo en el que se encuentra la cámara como la acción que se está llevando a cabo.
- En segundo lugar, tenemos el módulo central de nuestra arquitectura software, el módulo del control remoto. Este módulo se encarga de atender las interrupciones procedentes del bloque de accionadores del hardware, es decir, de la botonera, y realizar las peticiones HTTP necesarias en función de la acción solicitada por el usuario. Una vez realizadas las distintas acciones se encarga de comunicarse con el módulo pantalla para refrescar la misma y tener al usuario informado del estado del sistema. No solo realiza estas funciones, sino que además se encarga de tramitar gran parte de las peticiones HTTP solicitadas por el servidor web, es decir, desde el servidor web identificamos que acción quiere realizar el usuario, pero es el módulo de control remoto quien se encarga de gestionar y realizar dicha petición HTTP. Por qué se tramitan vía el módulo de control únicamente gran parte de las peticiones HTTP, se explicará en la documentación software en la parte relacionada con el servidor web.

- Finalmente, el tercer módulo, es el módulo del servidor web. Genera y gestiona el servidor web, mostrándose lo al usuario usando un navegador o explorador de su elección. Una vez el usuario ha seleccionado una acción, pueden pasar dos cosas, o bien la petición se tramita desde el propio servidor web o bien le indica al módulo de control remoto, que peticiones HTTP se quieren llevará a cabo en función de la elección del usuario. De forma simultánea, el propio módulo de servidor web se encarga de refrescar el módulo de pantalla, si es necesario, cuando el usuario realiza alguna acción desde el mismo.

A fin de hacer esta arquitectura más visual tenemos la siguiente figura, que nos muestra los distintos módulos descritos anteriormente, las variables globales que se usan en cada uno y los distintos métodos desarrollados para realizar las tareas arriba descritas.

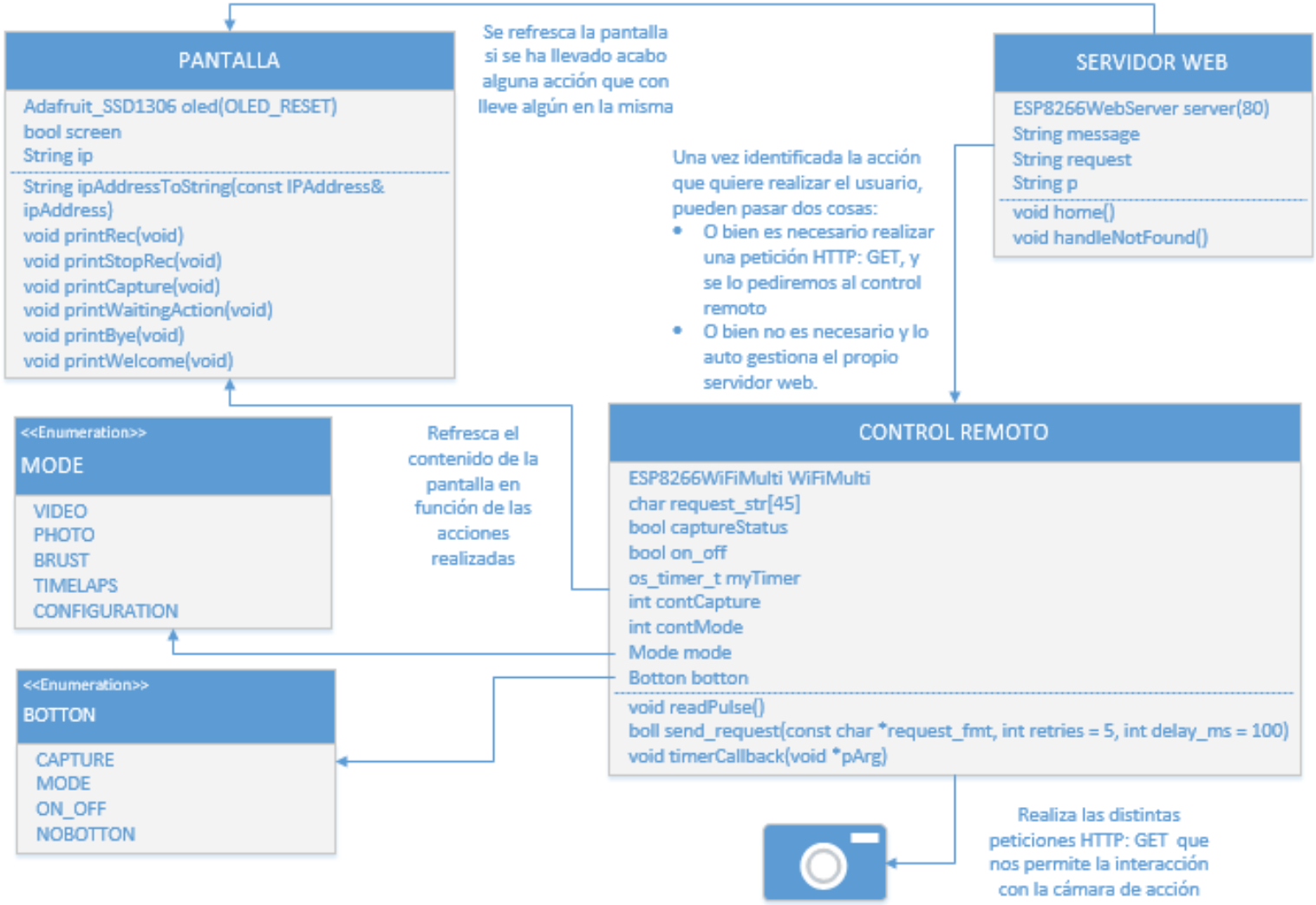


Figura 28 – Arquitectura software

4.2.2 Herramientas usadas

Durante la parte de desarrollo software se han usado tres herramientas:

- Arduino v1.8 IDE, es un entorno de desarrollo sencillo que permite programar, compilar los programar y cargarlos en el MCU, es importante saber que carece de debugger.
- Notepad++, es un editor de texto plano, que permite una visualización más amena del código en función del lenguaje de programación. Además, facilita las tareas de identado, ya que el entorno de desarrollo proporcionado por Arduino, es muy sencillo como hemos dicho.
- Navegador web, en este caso se ha usado el conocido Google Chrome para realizar las pruebas con el servidor web. Pero sirve cualquier navegador.

4.2.3 Documentación software

De cara a realizar la documentación software de nuestro sistema, vamos a proceder de forma que estuviéramos ejecutando el programa en nuestro MCU, por lo que comenzaremos ejecutando el `void setup():`

```
/*
 *Funciones: setup()
 *Entradas:           -
 *Salidas:            -
 *Descripcion: funcion de configuracion, solo se ejecuta una unica
 *            vez.
 */
void setup();
```

y seguidamente el `void loop()`:

```
/*
 *Funciones: loop()
 *Entradas:           -
 *Salidas:            -
 *Descripcion: funcion principal, que se ejecuta de forma ciclica
 *            de forma indefinida.
 */
void loop();
```

de forma indefinida, son dos funciones por defecto para ejecutar cualquier fichero con extensión ".ino". Arduino ejecuta de forma secuencial cada sentencia, pero con una peculiaridad las sentencias contenidas en `setup` solo se ejecutan una única vez al inicio de la ejecución, sin embargo, las sentencias contenidas en loop se ejecutan como un bucle infinito, repitiéndose indefinidamente.

En primer lugar comenzaremos ejecutando el void setup() que contiene la configuración inicial de nuestro sistema, por lo que configuraremos el WiFi para buscar una red con determinado nombre y contraseña (SSID y PASSWORD) que serán las credenciales de nuestra propia cámara de acción. Seguidamente iniciamos la conexión y encendemos la cámara. Para poder tratar los distintos tipos de pulsaciones que puede realizar un usuario en el módulo hardware de la botonera, es necesario realizar tratamiento de interrupciones, para lo cual armaremos las interrupciones con su función de tratamiento de interrupciones estableceremos un temporizador para capturar dichas interrupciones, cada 10ms. Por lo que cada 10 ms segundos el procesador dejará de hacer aquello que le tenga ocupado para irse a la función:

```
/*
 *    Funcion:        timerCallback(void *pArg)
 *    Entrada:        -
 *    Salida:         -
 *    Descripcion:    se llama a la funcion donde se trata la
 *                    interrupcion.
 */
void timerCallback(void *pArg);
```

Dentro de esta función tenemos una única sentencia, que es la llamada a la función:

```
/*
 *    Funcion:        readPulse(const IPAddress& ipAddress)
 *    Entrada:        -
 *    Salida:         -
 *    Descripcion: lee los valores del teclado conectado en la
 *          entrada analogica, determina cual de los dos botones
 *          se han pulsado, y asi como si la pulsacion ha sido
 *          larga o corta, para en funcion de esto, determinar
 *           que accion se quiere realizar, si accionar o deterner
 *          el disparador, cambiar de modo, enceder o apagar la
 *          camara o si no se ha apretado ningun boton.
 */
void readPulse();
```

Es una función en la que miramos la duración de la pulsación de cada botón haciendo uso de unos contadores, en función de esto podremos definir qué tipo de acción queremos hacer. El motivo de hacer esto es simple, aunque nosotros intentemos hacer una pulsación simple de cualquier botón en el contacto se producen rebotes, por lo que se genera un ruido eléctrico que puede falsear la pulsación. Como nuestro microcontrolador tiene un ciclo de reloj mucho más veloz que lo que nosotros como humanos podemos apreciar, lo que para nosotros es una única pulsación para el microcontrolador implica repetidas lecturas del mismo valor, esto se aprovecha para eliminar el *"Efecto rebote"*. Con esta función, implementamos un filtro anti rebote y

además ganamos la posibilidad de poder usar un único botón para programar varias funcionalidades, dependiendo de la duración de la pulsación del mismo.

Una vez establecido el temporizador de nuestras interrupciones y explicado su tratamiento, pasamos a configurar el servidor web, que se componen de dos rutas. En primer lugar, la ruta creada por la función:

```
/*
 *    Funcion:        home()
 *    Entrada:        -
 *    Salida:         -
 *    Descripcion: genera la pagina principal del servidor web
 *          y gestiona los botones del mismo, realizando las
 *          acciones que estos representan.
 */
void home();
```

Que monta la página principal del servidor web y se realiza el tratamiento de los botones que aparecen en dicha página, por lo que se pueden dar dos situaciones:

- O bien se no es necesario identificar los botones ya que al apretar el usuario el botón se realiza directamente una petición GET a la cámara sin usar ningún intermediario.
- O bien se llama a la función:

```
/*
 *    Funcion: send_request(const char *request_fmt, int retries,
 *                          int delay_ms)
 *    Entrada:
 *         const char *request_fmt
 *              => peticion
 *         int retries
 *              => número de intentos en caso de fallo
 *         int delay_ms
 *              => retraso entre intentos, en milisegundos
 *
 *    Salida: devuelve true si se pudo realizar la peticion,
 *           devuelve false si fallo
 *    Descripcion: manda una peticion HTTP tantas veces como
 *           <retries> con un retraso <delay_ms>. Si retries<0,
 *           entonces lo intentara continuamente. Solo se imprime
 *           un error, pasado los 10 errores.
 * Ejemplo:
 *    Formato de la petición HTTP
 *         http://10.5.5.9/param1/PARAM2?t=PASSWORD&p=%OPTION
 *    En donde:
 *         param1 es el lugar en donde va a tener lugar la acción
 *                en la cámara o en bacpac(WiFi)
```

```
 *         param2 es el tipo de acción a realizar
 *         password es la contraseña de la GoPro
 *         option son los parámetros para dicha acción
 *    request_fmt es esta parte de la peticion:
 *    /param1/PARAM2?t=PASSWORD&p=%OPTION
 *    La password se rellena dentro de la función
 *         request_fmt: "/bacpac/PW?t=%s&p=%%01"
 *         WIFI_PASSWORD: Villares82
 *         request_str: "/bacpac/PW?t=Villares82&p=%%01"
 */
bool send_request(const char *request_fmt, int retries = 5, int
delay_ms = 100);
```

Que es la encargada de realizar las peticiones HTTP: GET en función de los parámetros que recibe, el más importante es `request_fmt`, ya que corresponde al cuerpo de la petición. En primer lugar, se genera el cliente HTTP con el que se van a llevar a cabo las distintas peticiones. La variable `request_fmt` trae un campo sin cumplimentar que corresponde a la contraseña del dispositivo, si bien es cierto que se podría definir el `string` de la petición desde un principio con la variable contraseña ya cumplimentada, pero esto únicamente nos serviría para una misma contraseña, por tanto, si por cualquier motivo modificáramos la contraseña debería modificar una a una la definición de las distintas peticiones de forma manual.

Para evitar todo esto, lo primero que se hace es usar la función `snprintf()`, con los parámetros correspondientes, para cumplimentar el campo contraseña en la petición de tal forma que si la petición entra en la función con el siguiente formato: "/bacpac/PW?t=%s&p=%%01" a la salida de la función `snprintf()` tendremos, si la contraseña fuera “PASSWORD”, lo siguiente: "/bacpac/PW?t=PASSWORD&p=%%01".

Seguidamente se realizan intentos de petición, tantos como el parámetro `retries` nos indique y con un tiempo `delay_ms` entre cada intento de petición. Dentro de cada intento iniciamos el cliente http y una vez esté conectado el cliente, lanzamos la petición indicando el host destino, el puerto y la petición, seguidamente recogemos el código http de dicha petición:

- Si el código de la petición es 200, es que la petición se ha realizado correctamente y la función retorna `true`.
- Si el código de la petición es 403, es que la petición realizada llevaba mal la contraseña por lo que volverá a intentar la petición.
- Si el código de la petición es 410, es que la petición realizada llevaba un comando no valido, ya sea un error tipográfico o simplemente porque dicho comando no exista en el host destino.

En caso de que se alcance el número máximo de `retries` se informa al usuario de que la petición no se llevó a cabo, el numero de intentos que se llevaron a cabo y el tipo de petición que ha fallado, además retorna el valor false.

La razón por la que se dan estas dos situaciones en la página principal de nuestro servidor web es el siguiente, las peticiones que se deben mandar haciendo uso de HTML contienen, en uno de los parámetros, el carácter especial '%', y aunque se puede codificar con una de las siguientes notaciones '%25' o '%', no es un parámetro valido para mandar vía URL.

Una vez configurado el servidor web, se procede a guardar la IP del mismo, para que el formato nos sea útil para poderlo imprimir por nuestra pantalla y mostrársela al usuario en todo momento es necesario realizar una conversión de formatos, para ello se llama a la función:

```
/*
 *    Funcion: ipAddressToString(const IPAddress& ipAddress)
 *    Entrada:        recibe una IPAddress ip
 *    Salida:         devuelve la ip en formato string
 *    Descripcion: realiza el casting a string y concatenación
 *                 de los elementos de una ip
 */
String ipAddressToString(const IPAddress& ipAddress);
```

Esta función, recibe por la entrada una IP en formato `IPAddress` y la convierte al formato `string`, que si podemos imprimir por nuestra pantalla.

Finalmente, una vez realizada la petición se procede a refrescar la pantalla haciendo uso de la función necesaria para ello, en función de la acción realizada será una u otra.

En segundo lugar, tenemos la función:

```
/*
 *    Funcion:        handleNotFound()
 *    Entrada:        -
 *    Salida:         -
 *    Descripcion: en caso de que alguien intente entrar en una
 *                 pagina que no exista en el servidor web, genera
 *                 el error.
 */
void handleNotFound();
```

Que es una función para que en caso de que intenten alguien intente acceder a algo que no existe en el servidor web le muestre el error correspondiente en el navegador.

Por último en el `setup()`, arrancamos el módulo de la pantalla e imprimimos un mensaje de bienvenida.

En segundo lugar, explicaremos el contenido de la función `void loop()`. La función `void loop()` comienza arrancando el servidor web, para que este en todo momento disponible y se refresque continuamente. Seguidamente se pasa a diferenciar las

distintas acciones a seguir en función del valor de la variable `botton`, que corresponde a la lectura del teclado realizada durante el tratamiento de interrupciones. Por lo que podremos diferenciar las siguientes situaciones:

- No se ha apretado ningún botón, por lo que únicamente refrescaremos la pantalla de forma que informe al usuario que se está esperando a que pulse algún botón.
- Se apretó el botón de captura, por lo que se tiene que mirar en el modo en el que se encuentra el sistema, por defecto la cámara de acción viene configurada con el modo video al inicio. Si el modo es foto o ráfaga únicamente será necesario apretar una única vez el botón para que la acción se lleve a cabo. Sin embargo, si estamos en modo video o temporizador, una primera pulsación del botón de captura inicia la grabación o toma de fotos y una segunda pulsación finaliza la acción.
- Se hizo una pulsación sostenida el botón de modo u on/off, por lo que si la cámara está encendida se procede apagarla y viceversa.
- Se hizo una pulsación corta en el botón de modo u on/off, por lo que se irá cambiando el modo de la cámara de forma secuencial y cíclica en el siguiente orden: VIDEO-FOTO-RÁFAGA-TEMPORIZADOR.

Las acciones que se llevan a cabo en cada situación, corresponden a una llamada a la función `bool send_request(const char *request_fmt, int retries = 5, int delay_ms = 100)`, explicada en esta misma sección.

Durante toda la documentación software se ha hablado de refrescar el contenido de la pantalla, pero no se ha hecho referencia a las funciones que pueden hacer el refresco. El módulo de la pantalla, es llamado desde distintas funciones y está compuesto por 6 funciones de tipo `void`:

- `void printRec(void)`, informa al usuario por pantalla de se está comenzando a grabar un video o capturar imágenes si estamos en modo temporizador.

```
/*
 *Funciones:printRec()
 *Entradas:       -
 *Salidas:        -
 *Descripcion: imprime el mensaje en la pantalla OLED,
 *          para informar de que esta comenzando a grabar un
 *          video o capturar imágenes si estamos en modo
 *          temporizador.
 */
 void printRec(void);
```

- `void printStopRec(void)`, informa al usuario por pantalla de se está finalizado la grabación un video o la captura de imágenes si estamos en modo temporizador.

```
/*
 * Funciones: printStopRec ()
 * Entradas:        -
 * Salidas:         -
 * Descripcion: imprime el mensaje en la pantalla OLED,
 *         para informar de que esta finalizando a grabar un
 *         video o capturar imágenes si estamos en modo
 *         temporizador.
 */
 void printStopRec(void);
```

- `void printCapture(void)`, informa al usuario de que se está tomando una foto o se está capturando las imagenes en modo ráfaga.

```
 /*
  *Funciones: printCapture ()
  *Entradas:       -
  *Salidas:        -
  *Descripcion: imprime el mensaje en la pantalla
  *           OLED, para informar de que se esta tomando una
  *           foto o se está capturando las imagenes en modo
  *           ráfaga.
  */
  void printCapture (void);
```

- `void printWaitingAction(void)`, informa al usuario de que el sistema está disponible y esperando a que se realice una acción.

```
 /*
  *Funciones: printWaitingAction ()
  *Entradas:       -
  *Salidas:        -
  *Descripcion: imprime el mensaje en la pantalla OLED de que
  *         se está disponible y esperando a que se realice una
  *         acción.
  */
  void printWaitingAction (void);
```

- `void printBye(void)`, informa al usuario de que se está apagando la cámara.

```
/*
 *Funciones: printBye ()
 *Entradas:            -
 *Salidas:       -
 *Descripcion: imprime el mensaje en la pantalla OLED de que
 *             se está apagando la cámara.
 */
 void printBye (void);
```

- `void printWelcome(void)`, informa al usuario de que se está encendiendo la cámara.

```
/*
 *Funciones: printWelcome ()
 *Entradas:            -
 *Salidas:       -
 *Descripcion: imprime el mensaje en la pantalla OLED de que
 *             se está encendiendo la cámara.
 */
 void printWelcome (void);
```

De cara a ver de una forma más visual la relaciones entre las distintas funciones del sistema, adjunto este esquema: