# 1. Introducción y objetivos

Este proyecto se trata del proyecto final de la asignatura de Informática Industrial del Grado en Ingeniería Electrónica, Robótica y Mecatrónica de la Universidad de Málaga, para el curso 2021-2022. El objetivo de este proyecto es el de afianzar los conocimientos adquiridos durante el desarrollo de la asignatura en lo relacionado con el Internet de las Cosas, IoT, utilizando tecnologías como la programación de microcontroladores con capacidad de conexión WiFi, el uso de MQTT para el paso de mensajes, la utilización de NodeRED para desarrollar un sistema SCADA, el uso de bases de datos como MongoDB o la utilización de sensores como el DHT11. Para eso este grupo ha pensado en desarrollar el siguiente proyecto.

El proyecto trata de realizar una maqueta o prueba de concepto de un garaje. Este garaje contará con una entrada con barrera, modelada como un servomotor, controlada por una placa ESP8266 y accionada mediante un botón. El levantamiento de la barrera se realizará tras analizar la matrícula del vehículo a la entrada. Esto se hará captando una imagen con una ESP32-CAM, y enviando la imagen a un servicio de análisis de matrículas mediante redes neuronales, para lo cual se utilizará NodeRED.

También se controlará la ocupación de cada plaza de garaje mediante un sensor de distancia conectado a una placa ESP8266 que recogerá los datos del sensor. También a esta placa se le conectará un DHT11 para monitorizar temperatura y humedad. Todos estos datos se empaquetarán en una estructura y se enviarán por ESP-Now a otra placa ESP8266 que actuará como pasarela con MQTT, de forma que los datos puedan enviarse a NodeRED y guardarse en una base de datos.

Finalmente se desarrollarán dos interfaces de usuario. Una de ellas se hará en un Dashboard de NodeRED, y servirá como interfaz del sistema SCADA del garaje. Otra se hará también en NodeRED, pero de forma que sea accesible mediante un bot de Telegram. Ambas interfaces serán capaces de visualizar datos y enviar comandos al sistema.

# 2. Diseño Hardware y esquemas de conexionado

A continuación, se presenta un listado del hardware utilizado en el proyecto.

* Placas de desarrollo NodeMCU ESP8266, 3 unidades.
* Placa de desarrollo ESP32-CAM, 1 unidad.
* Sensor de humedad y temperatura DHT11, 1 unidad.
* Sensor de distancia de ultrasonidos HC-SR04, 1 unidad.
* Servomotor SG90, 1 unidad.

En las siguientes imágenes se pueden ver los esquemas de conexionad de todo el hardware utilizado en el proyecto, realizados con Fritzing.

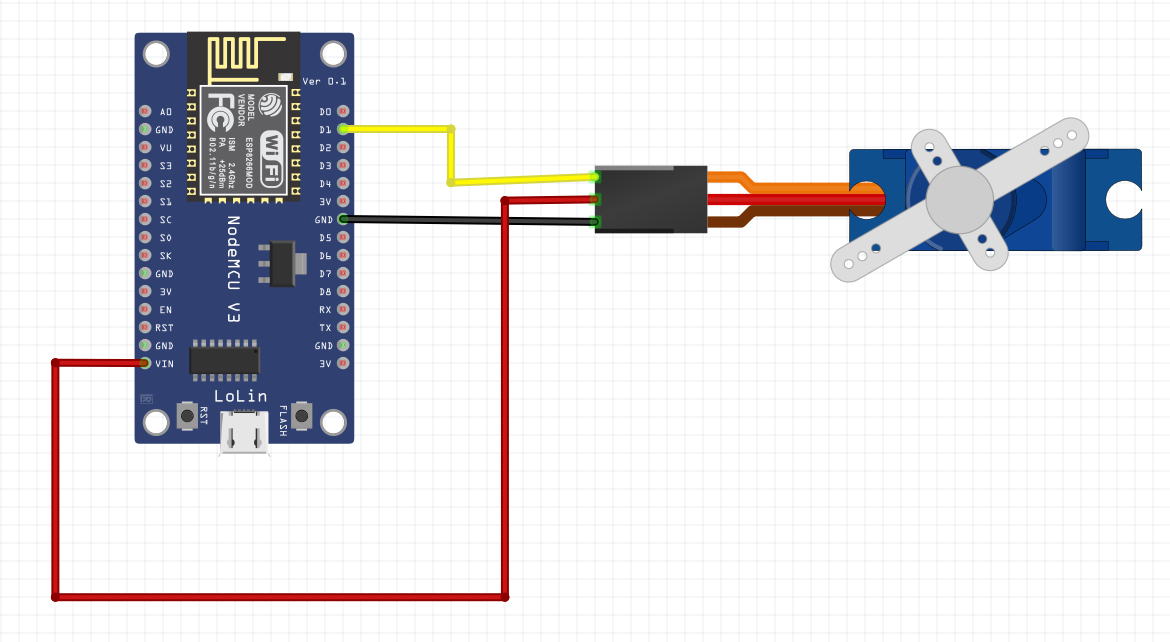


Ilustración 1. Placa de la entrada y servomotor de barrera

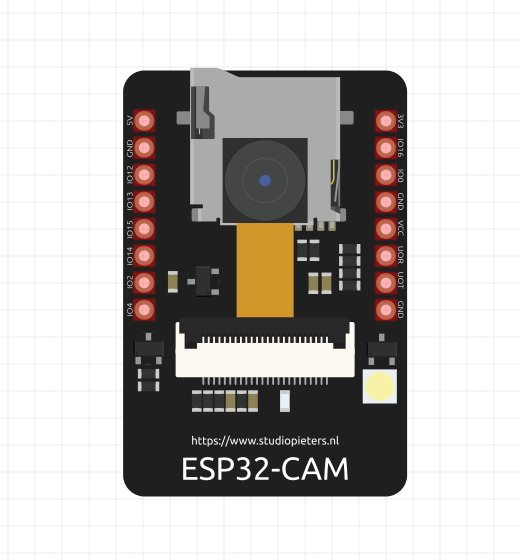


Ilustración 2. Cámara de la entrada

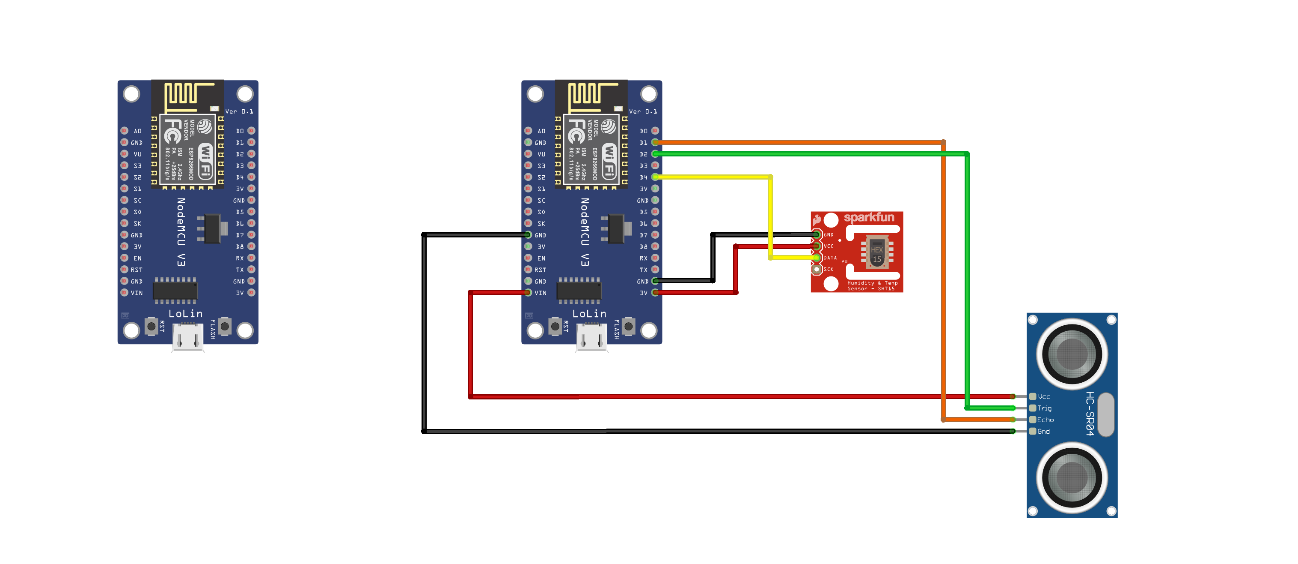


Ilustración 3. Placa de sensores y pasarela ESP-Now – MQTT

El conexionado de alimentación se ha obviado para todas las placas. Aparte de este, hay dos placas que no llevan ningún conexionado, como la de pasarela y la ESP32-CAM, mientras que la placa de sensores y la de control del motor de la entrada sí lo llevan. La placa de la entrada tiene conectada la alimentación del servomotor y la salida de datos. La de sensores lleva también la alimentación de dichos sensores, una conexión de datos para el DHT11, y dos para el sensor de distancia HC-SR04.

# 3. Diseño Software

## 3.1. Listado de topics MQTT

En el proyecto se ha decidido utilizar la siguiente estructura para los topics MQTT, utilizados para enviar mensajes entre dispositivos.

* II7
  + ESP32
    - conexion
    - config
    - FOTA
  + SensoresPlazas
    - conexion
    - config
    - FOTA
    - EstadoPlazas
  + Entrada
    - conexion
    - config
    - FOTA
    - BotonPulsado
    - BarreraEstado
    - BarreraCMD

La estructura utiliza una jerarquía que permite separar los topics en función de la placa a la que están destinados. Los que están bajo ESP32 son para la cámara, los que están bajo SensoresPlazas son para la pasarela MQTT – ESP-Now, que también conecta con las placas que monitoreen las plazas de aparcamiento, y los que estén bajo Entrada son para la placa que controla la barrera de entrada. Todas las placas tienen en común los topics acabados en /conexion, /config y /FOTA.

A continuación, se presenta el listado de los topics MQTT utilizados en el proyecto, junto con su descripción y ejemplos de uso.

* II7/ESP32/conexion
  + Indica el último estado de conexión del dispositivo. La placa es la que publica el mensaje.
  + Retenido: sí.
  + Ejemplo: {“CHIPID”: “ESP32”, “online”: “true”} o {“CHIPID”: “ESP32”, “online”: “false”}
* II7/ESP32/config
  + La placa se suscribe a este topic, en el que se pueden publicar mensajes con estructuras de datos para la configuración de parámetros en la placa. Principalmente se usa para especificar la frecuencia de comprobación de actualización FOTA.
  + Retenido: sí.
  + Ejemplo: {“frec\_FOTA”: 20000}
* II7/ESP32/FOTA
  + La placa se suscribe a este topic. Cuando llega un mensaje, debe comprobar la disponibilidad de actualizaciones FOTA. El mensaje que se pase por el topic es indiferente.
  + Retenido: no.
  + Ejemplo: indiferente.
* II7/SensoresPlazas/conexion
  + Indica el último estado de conexión del dispositivo. La placa es la que publica el mensaje.
  + Retenido: sí.
  + Ejemplo: {“CHIPID”: “ESP32”, “online”: “true”} o {“CHIPID”: “ESP32”, “online”: “false”}
* II7/SensoresPlazas/config
  + La placa se suscribe a este topic, en el que se pueden publicar mensajes con estructuras de datos para la configuración de parámetros en la placa. Principalmente se usa para especificar la frecuencia de comprobación de actualización FOTA.
  + Retenido: sí.
  + Ejemplo: {“frec\_FOTA”: 20000}
* II7/SensoresPlazas/FOTA
  + La placa se suscribe a este topic. Cuando llega un mensaje, debe comprobar la disponibilidad de actualizaciones FOTA. El mensaje que se pase por el topic es indiferente.
  + Retenido: no.
  + Ejemplo: indiferente.
* II7/SensoresPlazas/EstadoPlazas
  + La placa de pasarela MQTT – ESP-Now publica en este topic la estructura de datos que le llega por ESP-Now de la placa o placas de sensores.
  + Retenido: sí.
  + Ejemplo: {“chipId”: “ESP123456”, “numPlaza”: “1”, “ocupado”: “true”, “DHT11”: {“temperatura”: 23.4, “humedad”: 15.2}}
* II7/Entrada/conexion
  + Indica el último estado de conexión del dispositivo. La placa es la que publica el mensaje.
  + Retenido: sí.
  + Ejemplo: {“CHIPID”: “ESP32”, “online”: “true”} o {“CHIPID”: “ESP32”, “online”: “false”}
* II7/Entrada/config
  + La placa se suscribe a este topic, en el que se pueden publicar mensajes con estructuras de datos para la configuración de parámetros en la placa. Principalmente se usa para especificar la frecuencia de comprobación de actualización FOTA.
  + Retenido: sí.
  + Ejemplo: {“frec\_FOTA”: 20000}
* II7/Entrada/FOTA
  + La placa se suscribe a este topic. Cuando llega un mensaje, debe comprobar la disponibilidad de actualizaciones FOTA. El mensaje que se pase por el topic es indiferente.
  + Retenido: no.
  + Ejemplo: indiferente.
* II7/Entrada/BotonPulsado
  + La placa de la entrada publica en este topic cada vez que se pulse su botón Flash (pulsación corta). No se utiliza el formato JSON porque el mensaje enviado es indiferente.
  + Retenido: no.
  + Ejemplo: indiferente.
* II7/Entrada/BarreraEstado
  + La placa de la entrada publica en este topic cada vez que se alce o se baje la barrera.
  + Retenido: sí.
  + Ejemplo: {“estado”: “subido”} o {“estado”: “bajado”}
* II7/Entrada/BarreraCMD
  + La placa de entrada se suscribe a este topic y sube o baja la barrera de entrada en función del contenido del mensaje.
  + Retenido: sí.
  + Ejemplo: {“comando”: “subir”} o {“comando”: “bajar”}

## 3.2. Listado de colecciones de la base de datos

En el proyecto se utiliza la base de datos MongoDB vista en clase, que permite su separación en colecciones. En concreto para este proyecto se han definido las siguientes colecciones.

* ~~matriculas\_validas~~
* registro\_matriculas\_entrada
  + En esta colección se guardan como documentos los resultados de analizar las imágenes de las matrículas de los vehículos a la entrada con la API para NodeRED, tras ser estas recibidas de la cámara ESP32-CAM.
* registro\_estado\_plazas
  + En esta colección se guardan los documentos recibidos desde la pasarela MQTT – ESP-Now con el estado de ocupación y meteorológico de cada plaza de aparcamiento.
* registro\_conexion
  + En esta colección se guardan los documentos recibidos por los topics acabados en /conexion.
* registro\_barrera\_entrada
  + En esta colección se guardan los documentos recibidos por el topic II7/Entrada/BarreraEstado.

En todas las colecciones, antes de guardar documentos en ellas, a cada documento se le adjunta también la fecha de guardado, como otro campo más del documento.

## 3.3. Camera Web Server

Esta parte del código se trata del proyecto Arduino que irá programado en la placa ESP32-Cam. Se encargará de montar un servidor http al que otros equipos conectados a la red podrán hacer peticiones de captura de imágenes. Además, la placa se conectará al sistema de intercambio de mensajes MQTT, principalmente para publicar sus datos de conexión y desconexión, además de para atender peticiones de otros dispositivos a través de la gestión de callbacks. Mediante diferentes topics MQTT se permitirá la configuración de parámetros relacionados con la actualización FOTA.

El código principal del proyecto viene del ejemplo CameraWebServer presente en el IDE de Arduino, que permite montar el servidor http de cámara. A los archivos de este proyecto se han añadido los archivos actualiza\_dual.cpp y actualiza\_dual.h, que son una modificación del proyecto actualiza\_dual.ino provisto al grupo por el profesor Andrés Rodríguez Moreno, que contiene un ejemplo de código que permite realizar la actualización FOTA tanto para placas ESP32 como para ESP8266, con sentencias especiales para que en función de la placa para la que se vaya a compilar el proyecto, se incluyan unas librerías y funciones u otras.

Cabe destacar que el programa cargado en esta placa le asignará una IP a esta al conectarla a la red WiFi, y esta IP se escribirá por la consola del puerto serie tras el arranque del programa. Teniendo esta IP que, en las pruebas que hemos hecho, es siempre la misma para cada red a la que se conecta, podemos acceder al servidor http de la cámara para realizar peticiones, y también para acceder a su interfaz gráfica mediante un navegador.

Al archivo principal del proyecto, CameraWebServer.ino, también se le han añadido las funciones pertinentes para realizar la conexión con MQTT.

Se utilizan las librerías:

* PubSubClient
* WiFi
* ArduinoJson
* HTTPUpdate
* HTTPClient

El fichero actualiza\_dual.h también incluye, aunque no se utiliza al compilar en la placa ESP32, las librerías:

* ESP8266WiFi
* ESP8266httpUpdate

Los demás ficheros del proyecto usan otras librerías que no se especificarán, pero que vienen todas en el ejemplo citado ya incorporadas.

El diagrama de flujo del software programado en la ESP32-Cam es el siguiente:

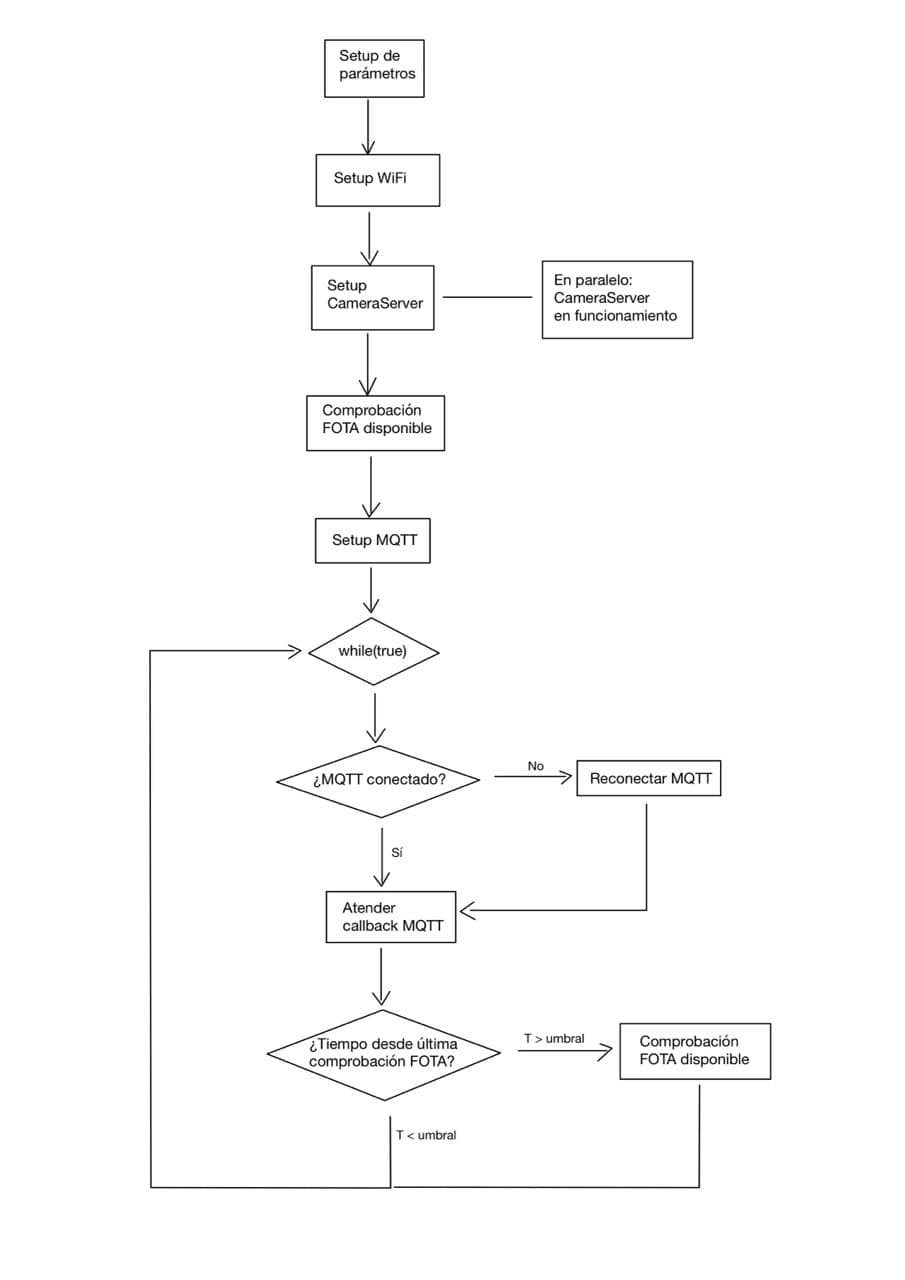


Ilustración 4. Diagrama de flujo de CameraWebServer

## 3.4. Entrada al garaje

## 3.5. Pasarela ESP-Now – MQTT

## 3.6. Sensores de aparcamiento

## 3.7. Flujo tal

## 3.8. Flujo cual

.

.

.

Dentro de cada subapartado:

Diagrama

Descripción

Robustez

Librerías

## 3.fin. Estudio de consumo

El estudio de consumo se ha hecho para dos de las placas. La primera es la ESP32-Cam. El estudio de autonomía para esta placa no es fiable porque, aunque teóricamente el montaje de medición del laboratorio debía alimentar la placa a 5 V, en la práctica, midiendo con un voltímetro, solo se estaba alimentando con 4.2 V la placa, por lo que no era capaz de funcionar correctamente y no llegaba a arrancar el programa de forma adecuada, no respondiendo a las peticiones de captura de imagen, aunque sí se conectaba a la red WiFi. Se intentó solucionar alimentando el montaje con una fuente de alimentación de PC, pero el problema persistió y la alimentación que llegaba a la placa no llegaba a superar los 4.2 V. De todas formas, se midió una corriente de 129 mA.

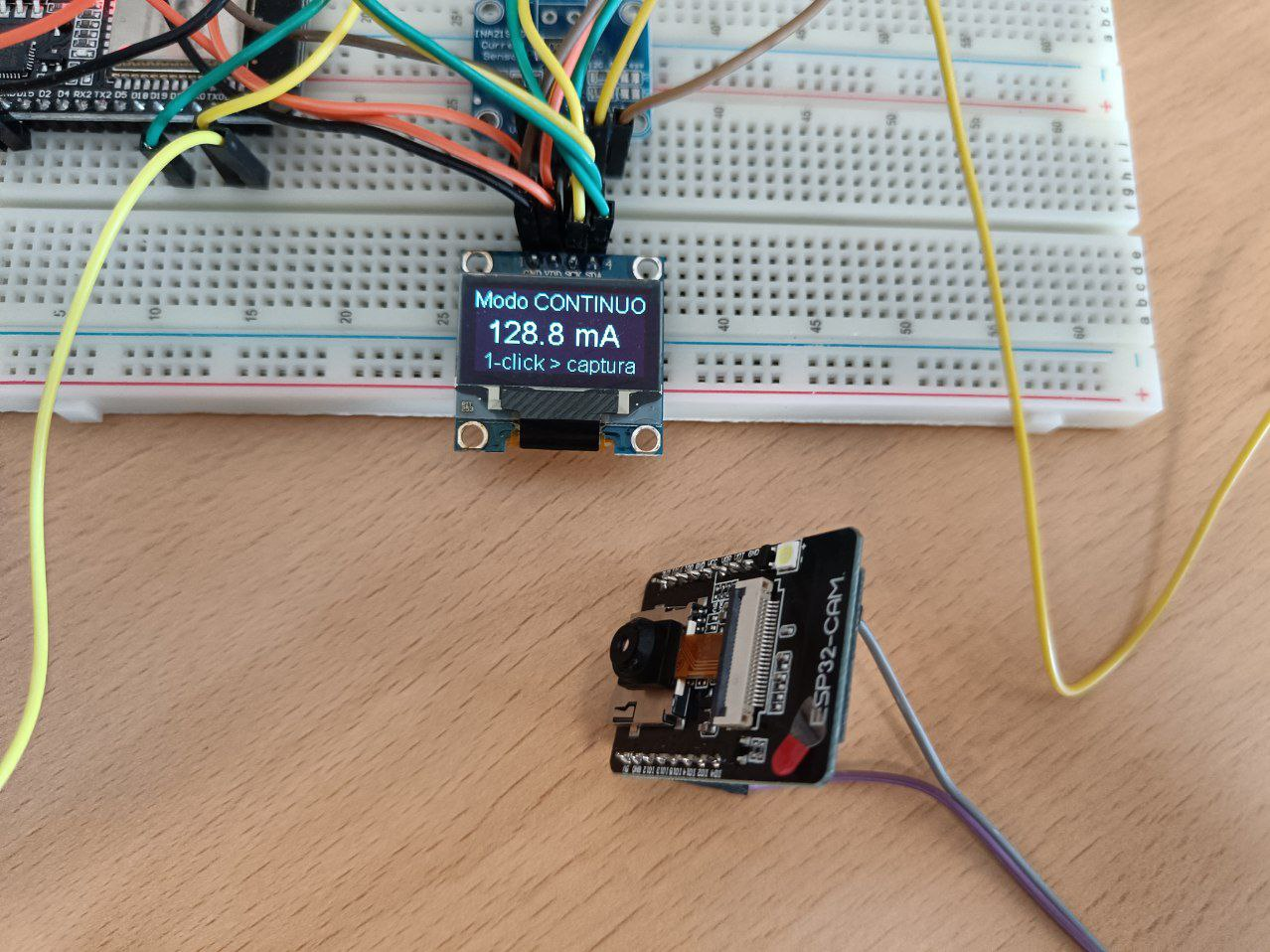


Ilustración . Estudio de consumo. ESP32-Cam

La otra placa para la que se realizó la medición de consumo fue la ESP8266 con los sensores de la plaza de aparcamiento (DHT11 y sensor de distancia de ultrasonidos HC-SR04). Esta placa sí arrancó correctamente y estableció la conexión ESP-Now con la placa pasarela. Se observó una corriente requerida de 87 mA.

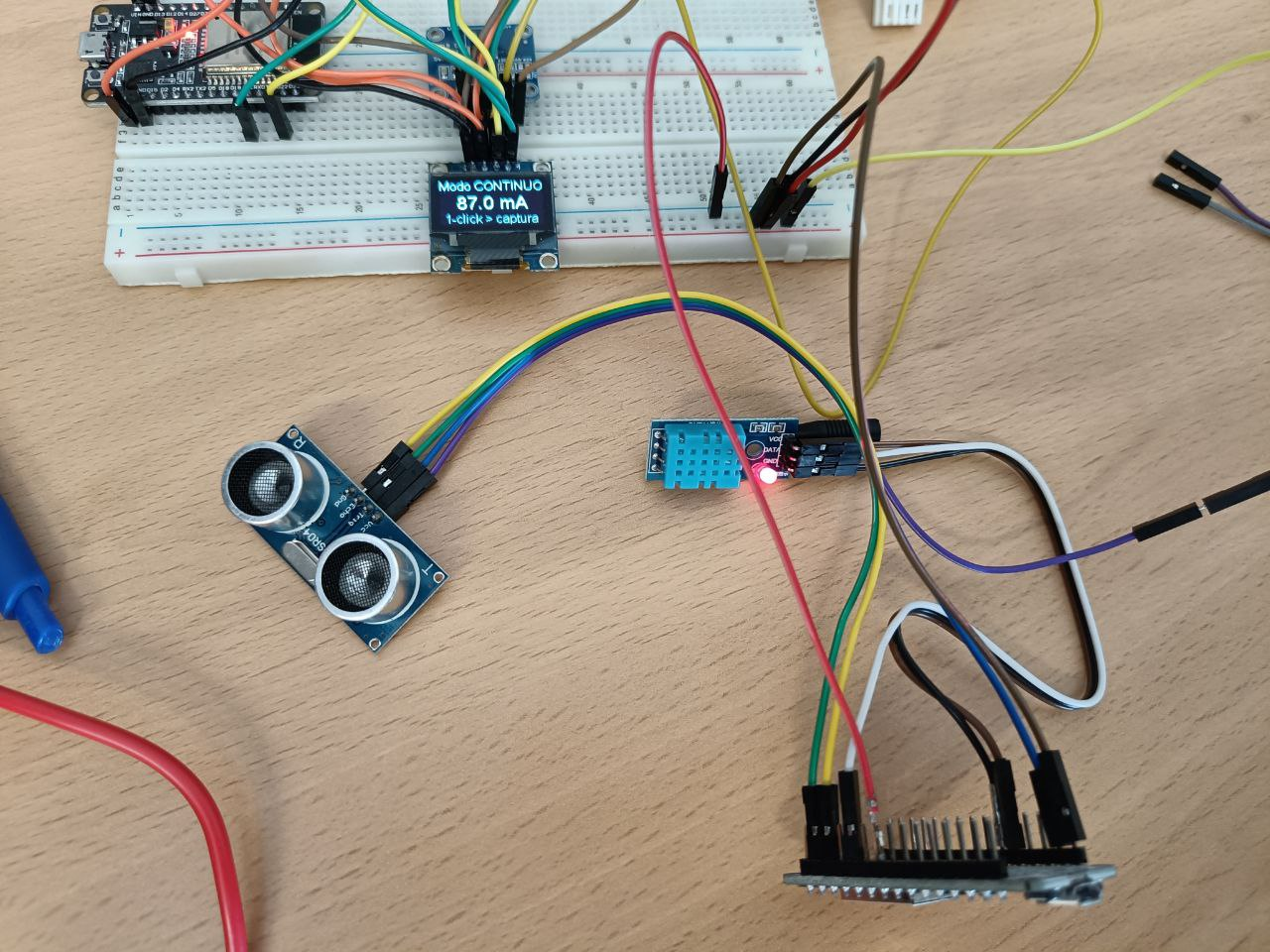


Ilustración . Estudio de consumo. ESP8266 con sensores DHT11 y HC-SR04

Al haber planteado como proyecto un parking, con fácil acceso a corriente, no se ha planteado un estudio de duración de batería exhaustivo, pero sí que se puede realizar la siguiente consideración. Si junto a cada placa tuviéramos una batería para emergencias de 1200 mAh, con los datos calculados, la ESP32-Cam podría funcionar con esa batería durante 9.3 h, y la placa de sensores, durante 13.8 h. Estos resultados son más que suficientes para resolver una emergencia en la alimentación normal de las placas. Sin embargo, como ya se ha discutido más arriba, los datos para la ESP32-Cam no son del todo fiables.

# 4. Resultados y conclusiones

Como conclusión, se han cumplido todos los objetivos que se marcaron al plantear el proyecto, consiguiendo poner en marcha una prueba de concepto de garaje con dispositivos IoT. Como consecuencia de esto, el grupo ha afianzado los conocimientos obtenidos en la asignatura sobre programación de microcontroladores con Arduino, el uso de MQTT para paso de mensajes, el uso de las bases de datos con MongoDB y la interconexión de todos estos componentes con NodeRED. También se ha conseguido hacer una interfaz de usuario, tanto en Telegram como en el Dashboard de NodeRED, que permite abstraer el funcionamiento del sistema a nivel de programación para poder utilizar el proyecto sin preocuparse de entender lo que hay por debajo de la interfaz. Aun así, con la documentación recogida en esta memoria y en los comentarios de los códigos, es perfectamente factible replicar el proyecto y seguir construyendo sobre lo que se ha hecho.

Como objetivos extra, el grupo ha podido profundizar su conocimiento sobre el funcionamiento de ESP-Now, y sobre cómo establecer una pasarela entre esta herramienta y MQTT, lo cual es realmente útil en nuestro proyecto, al tratarse un garaje con muchas plazas normalmente de un recinto bastante extenso.

También se ha aprendido a utilizar sensores nuevos como el sensor de distancia HC-SR04, y actuadores como el servomotor de la barrera de entrada, cuyo funcionamiento se facilita mucho gracias al uso de librerías diseñadas para ello. Además, también se ha aprendido el funcionamiento de APIs que permiten conectar nuestro código con otras aplicaciones que proveen servicio online como [platerecognizer.com](https://platerecognizer.com/), cuyo servicio está integrado con NodeRED mediante la librería node-red-contrib-plate-recognizer.

Otro elemento aprendido ha sido el desarrollo del código para una placa distinta a la que veníamos usando durante toda la asignatura, que es la ESP32, aprendiendo sobre las diferentes compatibilidades e incompatibilidades a la hora de usar librerías específicas para cada placa. Esto se pone principalmente de manifiesto en el uso del código para la actualización FOTA, que sirve tanto para la ESP32 como para la ESP8266. Mediante el estudio de este código también se ha aprendido sobre el funcionamiento de las directivas de pre-procesadores que han permitido que el código sea compatible con los dos tipos de microcontrolador sin cambio alguno.

Finalmente, el grupo también ha aprendido a utilizar repositorios mediante Git y Github, lo cual ha ayudado mucho en el proceso de desarrollar un proyecto de código en grupo, en el que se ha intentado desarrollar todo el código de la forma más modular posible, distribuyendo las partes reutilizables en ficheros que luego pudieran usarse en los proyectos para otras placas. Esto se ha hecho por ejemplo con los ficheros de actualización FOTA, que precisamente por ser válidos tanto para ESP32 como ESP8266, se han podido reutilizar para todas las placas que lo necesitaran.

Para acabar, ya que el proyecto se ha realizado teniendo en cuenta todo lo posible la escalabilidad del mismo, nos habría gustado hacer una prueba de esta escalabilidad con más placas para los sensores de cada plaza de garaje, así como con la posibilidad de tener salida con barrera motorizada en el mismo. Esto, en un futuro, nos permitirá probar si se ha planteado bien la escalabilidad del proyecto.

# Anexo 1. Lista y descripción de ficheros entregados

# Anexo 2. Manual de usuario