IMPLEMENTACIÓN DE UN SISTEMA MULTISENSORIAL DE BAJO COSTE

UNIVERSIDAD DE VALLADOLID Y SALAMANCA

FACULTAD DE CIENCIAS

Máster Universitario en Semiconductores y Tecnologías Electrónicas

Curso: 2024-2025

Autor: José Gabriel Nieto Ramos

Tutor/a: Helena Castán Lanaspa

Cotutor/a: Salvador Dueñas Carazo

Grupo de Caracterización de Materiales y Dispositivos Electrónicos

Agradecimientos

A esa persona que ha sido mi refugio en los días grises y mi alegría en los momentos de luz. Gracias por caminar a mi lado, por creer en mí incluso cuando yo dudaba, y por regalarme calma en medio de la tormenta. Tu apoyo silencioso, tu paciencia infinita y tu amor incondicional han sido el faro que me ha guiado hasta aquí.

Índice

Listado de figuras

Listado de tablas

Glosario

Arduino: es una plataforma de hardware y software de código abierto orientada al desarrollo de proyectos electrónicos que permite emplear microcontroladores de su propia marca o de otros.

Broker: es el servidor encargado de gestionar y distribuir todos los mensajes que circulan a través empleando el protocolo MQTT.

DeepSleep: es el estado de bajo consumo energético del ESP32 en el que la mayor parte del microcontrolador se apaga y solo permanece activo el reloj en tiempo real, permitiendo ahorrar batería hasta que se produzca un evento de activación.

ECU: son las siglas que reciben las unidades electrónicas de control utilizadas en automoción para gestionar y supervisar funciones del vehículo.

ESP32: es un microcontrolador de bajo consumo de la marca Espressif con conectividad WiFi y Bluetooth, muy utilizado en proyectos de IoT.

I2C: es un protocolo de comunicación serie que permite conectar varios dispositivos a un mismo bus usando solo dos cables, mediante los cuales se transmiten datos y reloj.

IoT: del inglés, Internet de las Cosas, este concepto hace referencia a una red de objetos y dispositivos conectados que pueden recopilar y transferir datos a través de Internet.

LoRaWAN: es un protocolo de red de largo alcance y bajo consumo diseñado para comunicaciones inalámbricas en aplicaciones IoT.

M2M: del inglés, Machine to Machine, se refiere a la comunicación directa entre dispositivos sin intervención humana.

MQTT: es un protocolo ligero de mensajería que permite la comunicación entre dispositivos en redes IoT mediante el intercambio de mensajes a través de un broker.

NTP: del inglés, Network Time Protocol, es un protocolo utilizado para sincronizar los relojes de los dispositivos conectados a una red con la hora correcta.

RTC: son las siglas de Real Time Clock, módulo o funcionalidad que mantiene la hora actual y permite almacenar datos durante los modos de bajo consumo en sistemas embebidos.

SPI: es un protocolo de comunicación serie síncrona que utiliza cuatro líneas para intercambiar datos rápidamente entre dispositivos.

VOC: son las siglas de Compuestos Orgánicos Volátiles, sustancias químicas presentes en el aire que pueden ser detectadas por sensores ambientales.

Resumen

Los sensores son dispositivos imprescindibles en la era digital en la que nos encontramos, donde todo y todos estamos conectados. La función de los sensores es capturar o percibir estímulos (físicos, químicos, biológicos…) y transformarlos en señales eléctricas, ya sean analógicas o digitales, pudiendo ser interpretadas dentro de los sistemas en los que existen. Gracias a la capacidad de sensorización digital que proporcionan, los sensores se han vuelto la base de la automatización, la eficiencia y la seguridad, no solo en entornos industriales, sino también en hogares.

Algunas de las funciones y las aplicaciones más importantes de los sensores son las siguientes:

- Automatización y eficiencia: el uso de sensores permite que los sistemas puedan funcionar de forma autónoma, siendo capaces de adaptarse al entorno sin necesidad de una intervención humana directa. Algunos ejemplos serían los sensores de presencia y temperatura, que permiten ajustar la iluminación y/o la temperatura de una sala de manera automática.
- Seguridad: fundamentales en la detección de intrusos, incendios, escapes de gases nocivos; aportan una respuesta rápida ante situaciones de emergencia.
- Salud y bienestar: capaces de leer la calidad del aire, la humedad y la temperatura, ayudan a mantener las condiciones óptimas en espacios y hogares.
- Productividad y reducción de costes: a nivel industrial, los sensores pueden monitorizar consumos, permitiendo conocer los puntos críticos de pérdidas de energía y pudiendo ajustar las líneas de producción en tiempo real, significando ahorros realmente importantes y rentabilidad a corto y medio plazo.

En la actualidad, para el Internet de las Cosas o IoT, los sensores juegan un papel clave y protagonista a la hora de recolectar datos y de plantear automatizaciones inteligentes, permitiendo la conexión entre personas, dispositivos y sistemas, así como la gestión y análisis de la información para poder optimizar los procesos. No será posible vislumbrar un futuro sin el uso de sensores a causa de la conectividad, la digitalización y la importancia para la inteligencia artificial.

¿Por qué se busca implementar sensores de bajo consumo? Esta tendencia sucede por las siguientes cuestiones o necesidades críticas de los sistemas:

1. Eficiencia energética y sostenibilidad: este tipo de sensores permiten operar durante periodos más largos de tiempo con baterías menores, redundando no solo en bajo consumo, sino también en el mantenimiento de los equipos, disminución del impacto ambiental y cumplimiento de normativas energéticas y medioambientales más estrictas.
2. Reducción de costes operativos: de la mano del primer punto, reducen significativamente los gastos de consumo de energía y de mantenimiento, sobre todo en redes extensas, permitiendo que sea más rentable de implementar una solución a gran escala sin tener muy en cuenta el factor del consumo.
3. Escalabilidad y flexibilidad: gracias a tecnologías como MQTT o LoRaWAN, se pueden conectar miles de sensores en grandes áreas, promoviendo el crecimiento de las redes IoT y la recolección de datos sin necesitar una infraestructura excesivamente costosa.
4. Innovación y nuevas aplicaciones: las aplicaciones de bajo consumo abren muchas posibilidades a nuevas innovaciones, sobre todo en zonas de difícil acceso

o donde la conexión eléctrica es inviable. También se promueve la evolución en desarrollar dispositivos portables.

Abstract

Sensors are essential devices in the digital age we live in, where everything and everyone is connected. The function of sensors is to capture or perceive stimuli (physical, chemical, biological, etc.) and transform them into electrical signals, whether analog or digital, so they can be interpreted within the systems in which they exist. Thanks to the digital sensing capability they provide, sensors have become the foundation of automation, efficiency, and safety, not only in industrial environments but also in homes.

Some of the most important functions and applications of sensors are as follows:

- Automation and efficiency: The use of sensors allows systems to operate autonomously, adapting to their environment without the need for direct human intervention. Examples include presence and temperature sensors, which can automatically adjust the lighting and/or temperature of a room.
- Safety: Sensors are fundamental for detecting intruders, fires, and hazardous gas leaks, providing a rapid response in emergency situations.
- Health and well-being: By measuring air quality, humidity, and temperature, sensors help maintain optimal conditions in spaces and homes.
- Productivity and cost reduction: In industrial settings, sensors can monitor consumption, identify critical points of energy loss, and allow real-time adjustments to production lines, resulting in significant savings and short- to medium-term profitability.

Today, in the context of the Internet of Things (IoT), sensors play a key and leading role in data collection and the implementation of intelligent automation. They enable the connection between people, devices, and systems, as well as the management and analysis of information to optimize processes. A future without sensors is unimaginable due to connectivity, digitalization, and their importance for artificial intelligence.

Why is there a drive to implement low-power sensors? This trend is driven by the following critical system needs:

1. Energy efficiency and sustainability: These types of sensors can operate for longer periods with smaller batteries, resulting not only in lower consumption but also in reduced equipment maintenance, decreased environmental impact, and compliance with stricter energy and environmental regulations.
2. Reduction of operating costs: Closely related to the first point, low-power sensors significantly reduce energy and maintenance expenses, especially in large networks, making large-scale solutions more cost-effective without having to worry much about power consumption.
3. Scalability and flexibility: Thanks to technologies such as MQTT or LoRaWAN, thousands of sensors can be connected across large areas, promoting the growth of IoT networks and data collection without the need for excessively expensive infrastructure.
4. Innovation and new applications: Low-power applications open up many possibilities for new innovations, especially in hard-to-reach areas or where electrical connections are unfeasible. They also encourage the development of portable devices.

Keywords

Air quality, connectivity, consumption control, energy efficiency, ESP32, flexibility, humidity sensor, innovation, Internet of Things (IoT), light intensity, MQTT, real-time monitoring, scalability, sensors, temperature sensor, wireless communication.

1.- Introducción

Los sensores son una herramienta crucial en el desarrollo de nuevos sistemas, pudiendo hacer que estos sean capaces de sentir lo que sucede en sus entornos y dando la posibilidad de implementar herramientas para que puedan autogestionarse y efectuar cambios por sí mismos. Además, la evolución de los sistemas de bajo consumo y de nuevas baterías, están permitiendo que los equipos no solo reduzcan su tamaño, sino que también sean capaces de ser portátiles y proporcionen a los usuarios la capacidad para ser colocados donde deseen.

Por otro lado, la evolución de los sensores ha ido muy ligada a la del IoT o Internet Of Things, puesto que proporcionan múltiples ventajas, no solo a los sensores, sino a todos los elementos de los sistemas. La principal ventaja que aporta el IoT es eliminar la comunicación cableada, evitando posibles errores de conexiones y reduciendo los costes de las instalaciones, mantenimientos y conocimientos de los técnicos en comunicaciones físicas.

1.1.- Marco del proyecto

Este proyecto tiene como intención estudiar, diseñar e implementar una estación multisensorial, la cual sea capaz de funcionar produciendo un bajo consumo y que suponga un coste reducido para poder ser replicable en cualquier lugar.

Para ello, se diseñarán dos dispositivos:

- Un sensor IoT que sea capaz de capturar diferentes datos, como temperatura, humedad, presión, composición de gases y luminosidad.
- Un broker o servidor MQTT que actúe como núcleo del sistema.

1.2.- Objetivos del trabajo

El objetivo principal del trabajo será realizar un microcontrolador con varios sensores y que sea capaz de funcionar en baja latencia, proporcionando no solo flexibilidad a futuro, sino que también cumpla con requisitos energéticos.

Para desarrollar el proyecto, debemos completar las siguientes tareas:

- Estudio del progreso de las tecnologías empleadas en sensores.
- Comprensión de los dispositivos y los sistemas de baja latencia.
- Estudio del protocolo Message Queuing Telemetry Transport (MQTT) para su implementación en un broker externo y la comunicación con el dispositivo con una interfaz destinada al usuario.
- Realización de pruebas y evaluación de los resultados obtenidos.
- Estudio y análisis económico del proyecto.

1.3.- Estructura del documento

A lo largo de este documento, se tratarán diferentes conceptos relevantes a los sensores de bajo coste y bajo consumo.

En el segundo capítulo, se pondrá en contexto el estado del arte sobre los sensores de bajo consumo, puesto que es otra vía de crear dispositivos duraderos y económicos, y la unión e importancia de las tecnologías IoT para economizar en mantenimiento de equipos de comunicación y cableado.

En el tercer capítulo se definirá el problema, y se llevará a cabo su análisis y la justificación de algunas modificaciones o aportaciones adicionales incluidas para darle mayor valor al proyecto.

Posteriormente, el cuarto capítulo tratará sobre la implementación; describirá cómo se ha logrado crear el sistema que cumpla con los requisitos establecidos y cómo se han implementado las tecnologías justificadas en capítulos anteriores.

Ligado al capítulo anterior, encontraremos el capítulo quinto, donde se explicará cómo se han realizado diversas pruebas para testear la funcionalidad y la robustez del sistema.

El sexto capítulo contendrá el análisis económico de crear el sistema y un dispositivo, así como el tiempo empleado, documentaciones, manuales…

Finalmente, encontraremos la conclusión y la definición de las líneas futuras en el capítulo séptimo.

2.- Sensores de bajo consumo

2.1.- La evolución de los sensores

El desarrollo de los sensores ha ido muy ligado al desarrollo de la humanidad, partiendo de sensores basados en mecanismos simples y rudimentarios para medir magnitudes físicas básicas (temperatura, presión o luz). Los primeros dispositivos eran muy imprecisos y estaban limitados a funciones como activar mecanismos, alarmas, etc…siendo capaces de detectar estímulos sin distinción entre su procedencia o importancia.

Con la llegada del siglo XX y el advenimiento de los transistores y de los circuitos integrados, los sensores sufrieron una evolución radical. Gracias a la miniaturización de los componentes y la capacidad de los chips de albergar múltiples funciones, se logró crear sensores que ocuparan menor espacio, con mayor fiabilidad y precisión. Un hecho muy importante fue el descubrimiento del efecto piezorresistivo en Si-Ge, favoreciendo la aparición de sensores más precisos.

Durante los años 60 y 70, se mejoró la robustez y se comenzó el desarrollo especializado de los sensores, como en los ámbitos de seguridad, donde los sensores de vibración evolucionaron para ser capaces de detectar si la actividad detectada era común o si era de un intruso real, reduciendo el número de alarmas falsas y mejorando su eficiencia, o de industria, especialmente en control de procesos y en automatización.

Ya llegada la década de los 80, surgen los sensores inteligentes, que eran capaces de autocalibrarse gracias al procesamiento de la información detectada y medida, siendo muy flexibles en su implementación en ambientes diversos y siendo capaces de diagnosticar fallos, lo cual permitía mayor control y seguridad en tiempo real.

A modo de resumen, gracias al progreso que han ido experimentando los sensores con el paso de las décadas, se ha logrado mayor precisión y fiabilidad permitiendo optimizar recursos, reducir errores humanos en puntos de alta criticidad y mejorar la calidad de vida.

2.2.- Ventajas del uso de tecnologías IoT y MQTT

Desde los años 80, cuando apareció el primer dispositivo conectado a Internet, una máquina de la empresa CocaCola, las comunicaciones IoT han ido tomando fuerza, aunque no sería hasta 1999 que se comenzaría a usar el término como tal. Posteriormente, se buscó aportar cierta inteligencia y autonomía a estos dispositivos mediante la implementación de sensores y actuadores.

La capacidad de los sensores de disponer del IoT ha permitido su integración en sistemas más complejos y el control de las variables medidas en tiempo real, siendo clave en campos como la medicina, para el monitoreo de pacientes; en automoción, para ayudas a la conducción o conducción autónoma; en industria, para la regulación semi o completamente automática de ciertos sistemas en procesos complejos; en agricultura, con el control de riego o análisis visual de los cultivos; y en domótica, permitiendo automatizar tareas del hogar.

Por ello, el uso de estas tecnologías se ha visto incrementado exponencialmente, llegando al punto de que en el 2010 superó a la población mundial. La llegada de las redes 5G también permitió un gran avance en este tipo de redes, puesto que mejoró las velocidades

de las comunicaciones WiFi. En la figura 1, se puede observar el constante crecimiento del número de dispositivos IoT.

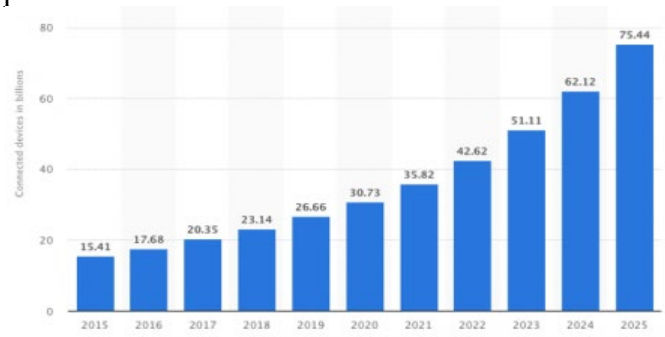

Figura 1. Número de dispositivos IoT existentes (en billones). [6]

Entrando más en detalle en el uso del protocolo MQTT, este se emplea en el control de dispositivos IoT gracias a su simplicidad, la ligereza de los mensajes enviados y su escalabilidad.

Navegando por internet, se pueden encontrar muchos proyectos que implementan esta tecnología, no solo en el entorno industrial, sino que también se puede emplear en hogares o en proyectos de automatización. Se pueden recurrir a los enlaces [8], [11], [14] o [18], para ver algunos ejemplos.

2.3.- Áreas de aplicación de los sensores

En la actualidad, los sensores han tomado fuerza en la automatización de hogares o, incluso, edificios inteligentes, en la agricultura y en el entorno de la salud. Gracias a que aportan datos en tiempo real, permiten optimizar las condiciones para lograr la máxima eficiencia energética, garantizar la seguridad o mantener condiciones favorables para la salud.

Algunas de las áreas de aplicación, junto a las tareas y parámetros a medir, son:

- Hogares y edificios inteligentes: se emplean para controlar el clima, la iluminación adaptativa, detección de intrusos y, por supuesto, optimizar el consumo energético. Algunos parámetros a controlar son la temperatura, la humedad, la luminosidad, el movimiento, los ruidos y/o el CO_2.
- Salud: su uso está ligado al mantenimiento y control de ambientes en hospitales, monitorización del aire (como CO_2 y VOC), temperatura y humedad en salas con pacientes críticos.
- Agricultura inteligente: en este ámbito, ha supuesto un avance importantísimo, gracias a poder optimizar y gestionar bien los recursos, poder conocer las necesidades de los diferentes terrenos y así poseer un control de calidad de los productos. Para ello, controlan la humedad de los suelos, las temperaturas, la humedad relativa, la intensidad lumínica, la presión o con sensores visuales saber el estado de las plantaciones.
- Automatizaciones industriales: con objetivo de lograr trazabilidad y asegurar la calidad de los productos, medir temperaturas, humedades, concentraciones de gases y partículas en suspensión sirven para determinar parámetros para determinar cómo saldrán las piezas o incluso averiguar dónde se ha podido producir un error.
- Automoción: con la continua evolución eléctrica-electrónica en los vehículos, no solo son necesarios más sistemas controlados por unidades de control electrónicas

o UCEs (ECUs en inglés), sino que estos sistemas necesitan múltiples sensores para poder proteger a los integrantes del vehículo, desde los sistemas de aparcamiento, ayudas a la conducción, sistemas de control de tracción, hasta la monitorización de la calidad del aire hacia el interior del habitáculo, el control de las luces según el momento del día… La temperatura, la humedad, la luz, la detección de lluvia o la calidad del aire son parámetros que miden continuamente los sistemas de los vehículos.

- Control meteorológico: con objeto de crear mapas predictivos y poder seguir los fenómenos climáticos, analizar los niveles de contaminación en entornos urbanos o para estudiar y gestionar las alertas meteorológicas. Los parámetros medidos suelen ser temperatura, humedad, concentración de contaminantes (CO_2, NOx…) o radiación UV.
- Gestión de residuos: para detectar si los contenedores se han llenado, análisis de zonas y los residuos generados, monitorizar los gases emitidos en los vertederos u optimizar las rutas de recogida. Los sensores empleados para este ámbito se dedican a medir temperaturas, presencia de gases (CO_2 o metano) y niveles de llenado.

3.- Análisis Previo

3.1.- Definición del problema

El objetivo propuesto fue desarrollar un dispositivo multisensorial independiente en un Arduino, el cual fuese capaz de almacenar la información de los diferentes sensores y que su implementación fuese relativamente económica.

Con la intención de aportar un mayor valor añadido al proyecto, se han modificado o añadido algunos objetivos, como implementar el dispositivo en un ESP32, gracias a sus bajos consumos, su bajo precio y sus funciones WiFi y de baja latencia embebidas en el chip, y el diseño de un sistema IoT basado en MQTT, que aporte mucha flexibilidad en la implementación para ser funcional en cualquier lugar o poder añadir nuevas funciones en la misma a futuro.

3.2.- Requisitos de la solución

Como requisitos principales, tendríamos los siguientes:

- Sensor/sensores suficientes para poder caracterizar un espacio.
- Implementación en un microcontrolador.
- Estudio de viabilidad económica, búsqueda de una solución funcional y robusta, pero sin excederse en el presupuesto.

Por contraparte, tendríamos como requisitos adicionales:

- Implementación del dispositivo a desarrollar en un ESP32.
- Desarrollo e implementación de una red MQTT.
- Implementación de funciones de baja latencia para reducir los consumos considerablemente.

3.3.- Microcontroladores, ¿por qué ESP32?

El dispositivo creado podría implementarse en cualquier Arduino, pero a nivel de costes, un Arduino puede valer desde los 30 euros, además de necesitar implementar un módulo adicional WiFi para la integración del Arduino en la red MQTT.

Por ello, se ha recurrido a la implementación del mismo en un ESP Wroom 32, un dispositivo que emplea un chip de ESPRESSIF Systems que integra funciones típicas de microcontroladores, con módulos WiFi, Bluetooth y de baja latencia, incluso dispone de dos núcleos. Adicionalmente, el coste de estos dispositivos ronda desde los 5 hasta los 10 euros si son adquiridos de forma individual, aunque al comprar más cantidad el precio se reduce.

Enumerando algunas de las características del ESP32, este dispone de un procesador Tensilica Xtensa LX6 de doble núcleo con frecuencias de reloj de hasta 240 GHz y opciones de funcionamiento en bajo consumo, posee WiFi 802.11, Bluetooth V4.2 y BLE, memoria ROM de 384KB, SRAM de 520KB (adicionalmente tiene una SRAM RTC de 16KB y una PSRAM de 8MB) y, para finalizar, tiene pines para comunicaciones con protocolos físicos (como PWM, SPI, I2S e I2C, CAN o UART). En la tabla 1 se indican todas las características del fabricante sobre el ESP32.

Características	ESP-32
Procesador	Tensilica Xtensa LX6 32 bit Dual Core a 160MHz
Memoria RAM	520kB
Memoria Flash	<16MB
ROM	448kB
Alimentación	[2.2, 3.6] V
Rango de temperaturas	[-40, 125] ºC
Consumo de corriente	80 mA (promedio) 225 mA (máximo)
Consumo en modo Deep-sleep	2.5uA (RTC + memoria RTC)
Cooprocesador	ULP (consumo inferior a 150uA)
WiFi	802.11 b/g/n
Bluetooth	V4.2 BR/EDR y BLE
UART	3 puertos
I2C	2 interfaces
SPI	4 interfaces
GPIO (utilizables)	11 pines
PWM	16 canales
ADC	2 (<18 canales) (12 bits)
ADC preamplificador	Sí (bajo ruido hasta 60dB)
DAC	2 canales de 8 bits
I2S	2 interfaces
CAN 2.0	1 bus
Ethernet	10/100 Mbps MAC
Sensor de temperatura	Sí
Sensor de efecto Hall	Sí
Infrarrojos	Sí
Timers	Sí
Encriptación	AES, SHA, RSA, ECC
RNG	Sí
Encriptación flash	Sí
Secure Boot	Sí

Tabla 1. Características Hardware del ESP 32

Gracias a todas estas propiedades, los ESP32 (figura 2) aportan versatilidad, reducido espacio y bajo coste, lo cual ha favorecido que este microcontrolador haya tomado mucha fuerza en el mundo IoT.

Figura 2. Microcontrolador ESP32

3.4.- Comunicación IoT, ¿por qué MQTT?

Algunos ejemplos de protocolos IoT son OPC-UA, MQTT, … De entre todos ellos, se ha seleccionado el protocolo MQTT.

Este protocolo se basa en un sistema de publicadores y subscriptores enlazados por una mensajería ligera, siendo los publicadores quienes dejan la información en un tema o topic y los subscriptores están a la espera de modificaciones para recibir las informaciones que les interesan. Otros de los motivos para seleccionarlo son por ser uno de los protocolos más populares en el ámbito del IoT y en los sistemas de comunicación M2M, promovido por su simplificad y eficiencia.

Toda la red de publicadores/subscriptores esta enlazada entre sí mediante un servidor o un broker, el cual se encarga de gestionar todos los flujos de información entre los clientes, siendo una capa de aplicación en las arquitecturas de redes (puede verse en la figura 3).

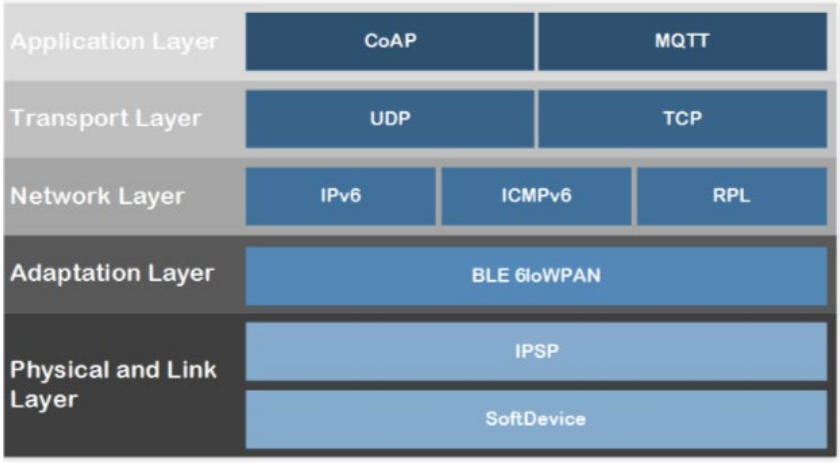

Figura 3. Esquema de niveles de red de comunicaciones [3]

Una ventaja general de los protocolos de comunicación MQTT, es que las líneas que unen sus clientes y sus subscriptores con el servidor no son físicas, sino que se realiza por WiFi (figura 4).

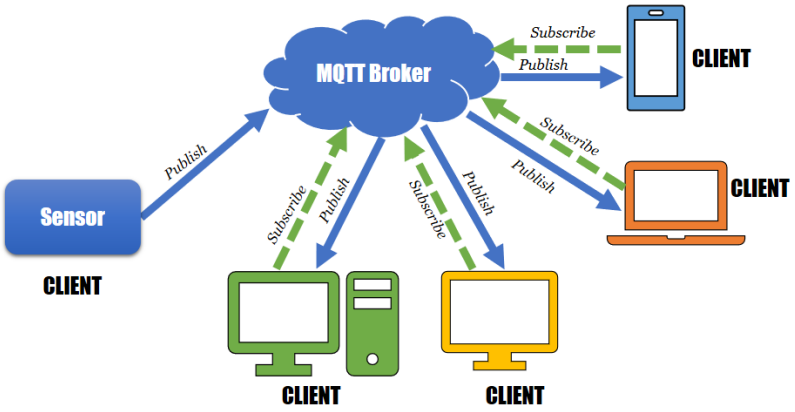

Figura 4. Visión global de un entorno MQTT [1]

Algunos conceptos importantes del protocolo MQTT son los tres niveles de QoS o Calidad del Servicio y los topics o temas.

Los tres niveles de QoS van desde el 0 hasta el 2, yendo desde el nivel de menor fiabilidad (0) hasta el de mayor fiabilidad (2):

- QoS 0: se envía un mensaje al subscriptor solo una vez, no existe ningún tipo de confirmación de recepción del mensaje.
- QoS 1: tras en el envío de información al subscriptor, el broker queda a la espera de recibir una confirmación de recepción del subscriptor; si no recibe el mensaje de confirmación, reenvía la información.
- QoS 2: mediante 4 etapas se garantiza que la información se ha recibido una única vez por el subscriptor.

Para esta aplicación, nuestro único subscriptor será el broker, pero se emplea un QoS 0, sin necesidad de confirmaciones de recepción, puesto que no disponemos de comunicaciones críticas.

Los topics o temas son las estructuras que emplea MQTT para organizar la información. Este método es muy sencillo, puesto que es una distribución similar a directorios o carpetas, donde ramificamos empleando "/" para distinguir las informaciones de un mismo lugar (en nuestro caso podríamos separar Sensor/Temperatura de Sensor/Humedad, pero seguir disponiendo de toda la información en Sensor).

3.5.- Sensores empleados

Para la creación del dispositivo sensorial con el ESP32, se han empleado dos sensores:

- Sensor LDR KY-018 de AZDelivery:
- Sensor BME680 de SEENGREAT.

A continuación, explicaremos cómo funcionan los sensores en detalle.

3.5.1.- Sensor LDR KY-018

Este módulo es muy utilizado en proyectos de electrónica con el objetivo de medir la intensidad de luz ambiental. En la figura 5 se puede observar el sensor cuestión:

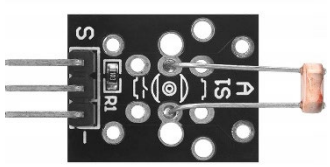

Figura 5. Sensor LDR KY-018 de AzDelivery [4]

Su funcionamiento se basa en un divisor de tensión, donde una de las resistencias es conocida (de 10 kΩ) y la otra es una fotorresistencia sensible a la luz (mayor luz aplicada supone menor resistencia, y menor luz supone mayor resistencia).

El sensor dispone de 3 pines: señal, positivo y tierra. La alimentación del sensor va desde los 3.3V hasta los 5V y la señal proporcionada es analógica, dando como resultado un número del rango [0, 4055] (de más a menos luz).

El sensor KY-018 no es solo sensible a la luz visible, sino que también se ve afectado por la luz ultravioleta e infrarroja, lo cual le hace muy atractivo para aplicaciones de mediciones ambiente, seguidores de luz, detectores crepusculares…

La respuesta de este sensor no es lineal, sino que la relación entre resistencia y luz incidente dependerá del material de la resistencia LDR y se basa en $R = A * L * \alpha$, donde A y α son constantes que dependen del material del sensor y L será la intensidad lumínica.

3.5.2.- Sensor BME680

Este sensor ha sido desarrollado por BOSCH, quienes han creado un sensor ambiental digital avanzado que, gracias a los avances de miniaturización de los sensores, alberga en un pequeño chip la capacidad de medir temperatura, humedad relativa, presión barométrica y calidad del aire. El ensamblado del sensor se ve en la figura 6:

Figura 6. Sensor BME680 de SEENGREAT [23][24][25]

Gracias a la precisión que aporta, se emplea en aplicaciones donde se necesita de precisión y eficiencia, como pueden ser las estaciones meteorológicas o domótica.

Como hemos indicado en el principio, este sensor incorpora 4 sensores internamente:

- Sensor de temperatura: mediante un principio resistivo, logra medir la temperatura del ambiente, logrando una precisión de ±1ºC y una resolución de 0.01ºC, con un rango de [-40, 85] ºC.
- Sensor de humedad: con el uso de tecnología capacitiva, es capaz de medir la humedad relativa del ambiente con una precisión del ±3% y una resolución de 0.008%, aportando un rango útil de [0, 100] %.
- Sensor de presión: para medir la presión atmosférica, emplea un elemento piezoresistivo de alta precisión, aproximadamente ±0,12hPa, para medir en un rango de [300, 1100] hPa. Una peculiaridad de este sensor es que, con la presión de un metro que aporta, permite calcular la altitud.
- Sensor de gases: mediante un sensor MOx (Metal-Óxido), el sensor es capaz de determinar la calidad del aire al medir los compuestos orgánicos volátiles (VOC), como pueden ser el monóxido de carbono, el etanol, la acetona… El sensor MOx se calienta internamente y modifica su conductividad eléctrica en función de las moléculas VOC que absorba del ambiente. El valor transmitido se denomina índice de calidad del aire, puesto que no es capaz de diferenciar entre los gases de forma individual.

Este sensor comunica de forma digital utilizando los protocolos I2C (con una frecuencia de hasta 3.4MHz) o SPI (hasta 10MHz). En esta aplicación se empleará I2C por no desempeñar una tarea crítica, a causa del funcionamiento con la baja latencia, y por la búsqueda de reducir los consumos.

El chip funciona con una tensión de alimentación de [1.71, 3.6]V, pero en el módulo comercial empleado, permite alimentaciones de 3.3V a 5V. En relación con esto, el consumo energético del sensor será de 0,15µA en reposo y de [0.09, 12]mA en el periodo activo de medición (según el modo de comunicación y los sensores necesarios).

Una recomendación del fabricante es el tiempo de precalentamiento, el cual consiste en 30 minutos con el objetivo de estabilizar las mediciones de gases, y de 48 horas si el sensor se cambia de entorno.

3.6.- Funcionamiento de la solución implementada

Aquí se expondrán los funcionamientos tanto del broker como del sensor en cuestión.

Comenzando con el broker, este dispositivo al ser alimentado hará lo siguiente:

1. Conectarse a la red WiFi preconfigurada y verificar la conectividad.
2. Inicializar el broker en una dirección IP.
3. Mantener activo el broker.

Adicionalmente, se ha incluido una publicación vía serial de la dirección IP, para que se pueda conocer la dirección para conectar los sensores o actuadores a la red MQTT.

El sensor está programado para que actúe de forma cíclica, puesto que incluirá funciones de baja latencia. Los pasos cíclicos del sensor serán:

1. Despertado del dispositivo (paso del estado de baja latencia al estado de actividad).
2. Conectarse a la red WiFi y al broker MQTT, verificando la conectividad.
3. Captación de los valores medidos por los sensores.
4. Publicación de los valores en los respectivos topics.
5. Dormido del dispositivo (paso del estado de actividad al estado de baja latencia).

El objetivo del estado de baja latencia es reducir lo máximo posible el consumo del dispositivo, puesto que no será necesario obtener valores de los sensores contantemente, sino que con valores equiespaciados controlados será suficiente.

Con el siguiente diagrama de flujos de la figura 7, se expone el funcionamiento de ambos dispositivos:

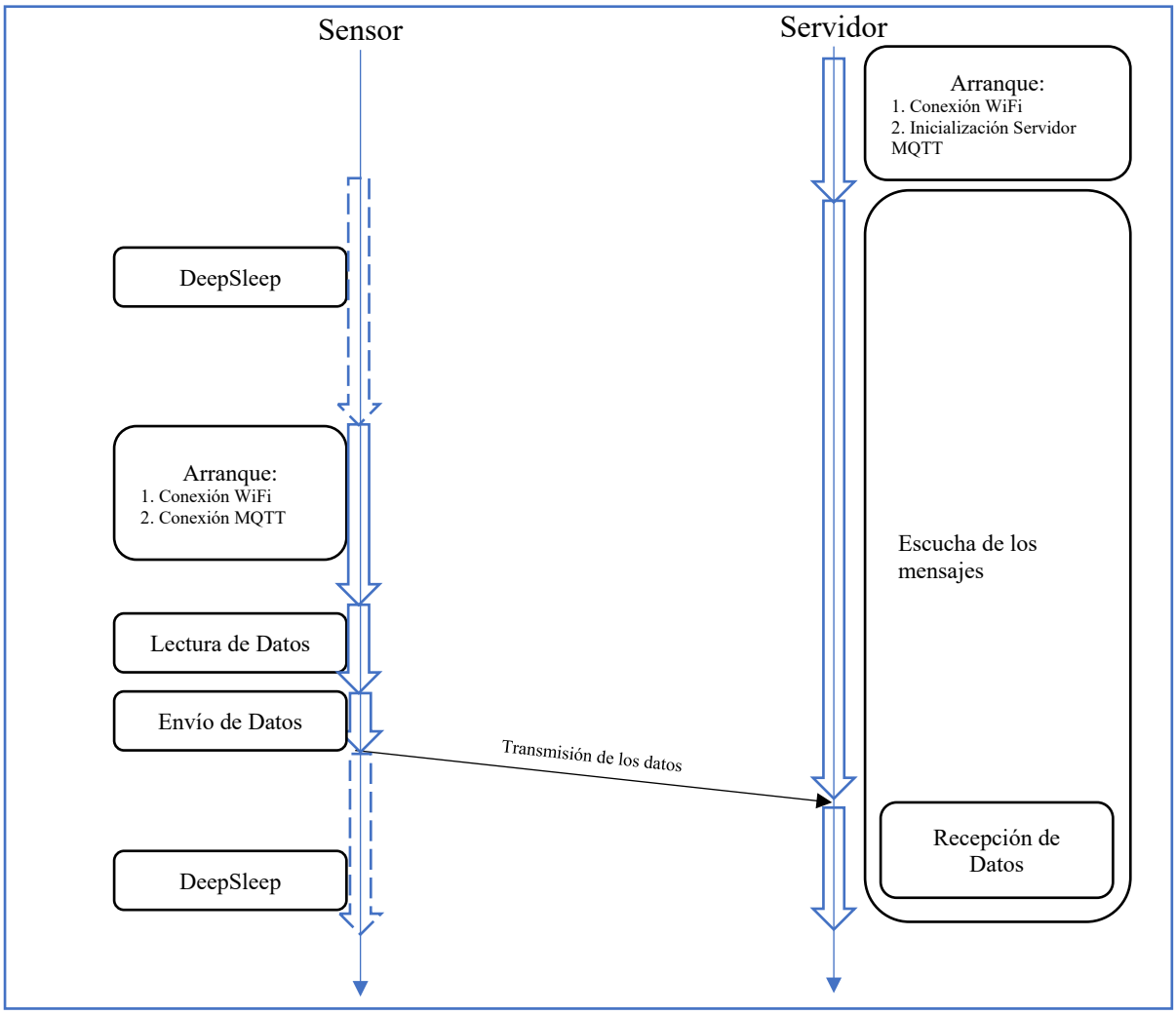

Figura 7. Diagrama de flujo del funcionamiento del broker y del sensor

4.- Implementación y desarrollo

En este apartado, se van a describir las diferentes funciones y dispositivos creados durante este TFM, con el objetivo de explicar completamente todos sus montajes y programaciones.

En la figura 8 se observan ambos dispositivos: el broker MQTT (a la izquierda) y el dispositivo sensor completo (a la derecha).

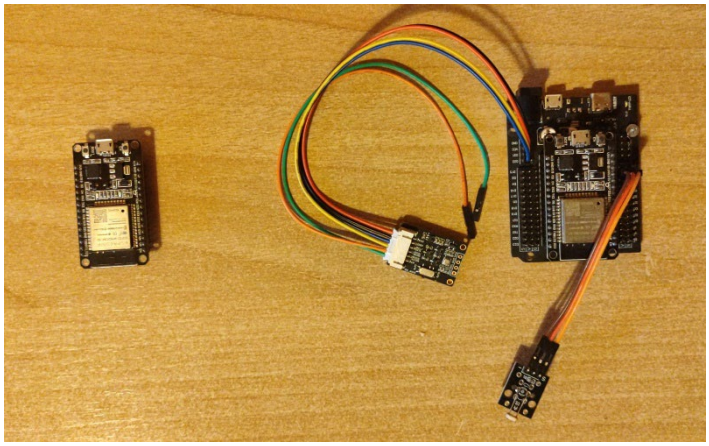

Figura 8. Broker MQTT y dispositivo sensor

4.1.- Broker MQTT

La programación del bróker consiste en las siguientes partes:

1. Declaración de librerías: para este primer dispositivo, solo serán necesarias tres librerías. La librería "WiFi.h" se emplea para conectarse a una red WiFi y disponer de servicios red y "PicoMQTT.h" será la librería empleada para crear el servidor o broker MQTT.

```
#include <WiFi.h>
#include <PicoMQTT.h>
```

2. Inicialización de variables: aquí declararemos el SSID y la clave de la red WiFi a la que vayamos a conectar y la variable de arranque del broker.

```
// Configuración WiFi
const char* ssid = "GabiWiFi";
const char* password = "Valladolid01!";


// Crea instancia del broker
PicoMQTT::Server broker_mqtt;
```

3. Función setup: función ejecutada tras iniciarse las variables al conectar el dispositivo, en este caso, configura la frecuencia de los mensajes a través del puerto Serial (vía cable), inicia el proceso de conexión WiFi, un bucle mediante el cual continúa buscando hasta lograr conectarse a la red configurada, tras la conexión, se emite un mensaje Serial con la dirección IP del ESP32 donde se generará el broker y se inicia el servicio MQTT (las líneas comentadas encima son para iniciar un servicio MQTT con autenticación de usuario/contraseña).

```
void setup() {
  Serial.begin(115200);

  // Conexión WiFi
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("\nConectado a WiFi: " + WiFi.localIP().toString());

  // Configuración segura del broker
  //broker_mqtt.config.allow_unauthorized = false;  // Habilita autenticación
  //broker_mqtt.set_credentials("usuario", "contraseña");  // Credenciales MQTT
  mqtt.begin();  // Inicia el broker
}
```

4. Función loop: en esta función se emplea para programar las acciones del dispositivo que se desean realizar en bucle tras la inicialización, para el broker solo será necesario llamar en bucle a la función, de la librería "PicoMQTT.h", loop(), la cual permite que el servicio o broker creados en la función setup se mantenga activa de forma permanente.

```
void loop() {
  broker_mqtt.loop();  // Mantiene activo el broker
}
```

Gracias a todo esto y a que el propio chip del ESP32 incluyen componentes para la conectividad WiFi, este dispositivo es capad de aportar el servicio MQTT simplemente conectando el microcontrolador a la red eléctrica.

4.2.- Dispositivo sensor

El dispositivo principal de este proyecto está formado por, como ya se ha comentado con anterioridad, por un ESP32, un sensor LDR KY-018 y un sensor BME680. El primer sensor mide la diferencia de potencial en el divisor de tensión y lo convierte en un valor analógico, mientras que el segundo envía los datos vía I2C, por lo que el ESP32 debe configurar los puertos dedicados para este caso y así capturar toda la información captada por este.

Su programación consta de:

1. Declaración de librerías: para este dispositivo, también se empleará la librería "WiFi.h" con el mismo fin que en el broker MQTT. Por otro lado, usaremos "PubSubClient.h" para hacer que este dispositivo sea un cliente MQTT publicador de información, "bme68xLibrary.h" para poder sacar funciones únicas de nuestro sensor BME680 y "time.h" para obtener de internet la fecha y hora actuales.

```
#include "Arduino.h"
#include "bme68xLibrary.h"
#include <WiFi.h>
#include <PubSubClient.h>
#include "time.h"
```

2. Inicialización de variables del DeepSleep: con estas variables iniciaremos el modo de baja latencia, mediante el cual nuestro dispositivo lanza en el segundo núcleo un ciclo de reloj para que, al cumplirse el tiempo, este se apague y, tras otro ciclo, este se inicie de nuevo.

```
// Definición del DeepSleep:
#define TIEMPO_SLEEP 20 * 1000000  // Microsegundos (20 segundos)

// Variables RTC (conservan valores entre despertares)
RTC_DATA_ATTR int bootCount = 0;  // Contador de despertares
```

3. Inicialización de parámetros de los sensores: aquí se definirán los pines de comunicación con los sensores, así como el tipo de comunicación con el BME680 (en este caso I2C) o el modo de funcionamiento.

```
// Inicialización de los sensores:
#define USE_IIC 1
#if USE_IIC
#define ADDR_I2C  0x77 // I2C
#else
#define PIN_CS 27
#endif
Bme68x bme;
const int pinLDR = 33; // D33 en ESP32
```

4. Inicialización de variables WiFi y MQTT: se definen los parámetros de la red WiFi a conectar, así como la IP del broker para enlazar el publicador al servicio y se crea la variable cliente MQTT.

```
// Configuración de red WiFi y dirección del broker MQTT a forzar:
const char* ssid = "GabiWiFi";
const char* password = "Valladolid01!";
const char* mqtt_server = "192.168.157.169";

// Inicialización del cliente WiFi:
WiFiClient espClient;
PubSubClient client(espClient);
```

5. Definición de parámetros de los mensajes: se definirá que los mensajes no serán más extensos de 50 caracteres y se inicializa una variable que almacenará la información del último mensaje.

```
// Definición de mensajes
long lastMsg = 0;
char msg[50];
```

6. Definición del servicio NTP: NTP significa Network Time Protocol, siendo un protocolo estándar utilizado para sincronizar los relojes de los sistemas conectados a la red.

```
// Configuración NTP
const char* ntpServer = "pool.ntp.org";
const long gmtOffset_sec = 3600;      // Ajusta según tu zona horaria
const int daylightOffset_sec = 3600;  // Ajusta según horario de verano
```

7. Función setup_wifi: esta función lanza el ciclo de intentos para conectar los datos de la red WiFi preconfigurada.

```
void setup_wifi() {
  delay(10);
  Serial.println("\nConectando a: " + String(ssid));
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("\nConectado a WiFi\nIP: " + WiFi.localIP());
}
```

8. Función reconnect: se llama a esta función cuando se quiere conectar al broker MQTT, asegurando que el dispositivo puede conectarse, si no, ejecuta una excepción de error hasta lograr conectar con el servicio.

```
void reconnect() {
  while (!client.connected()) {
    Serial.print("Intentando conexión MQTT...");
    if (client.connect("ESP32Client")) {
      Serial.println("Conectado");
      client.subscribe("esp32/entrada");
    } else {
      Serial.print("Error, rc = ");
      Serial.print(client.state());
      Serial.println("Reintento en 5s");
      delay(5000);
    }
  }
}
```

9. Función getDateTime: esta función necesita que se la inicialice con un mensaje y la longitud del mismo, en este caso vamos a emplearla para reducir el formato de fecha contenido en la variable "buffer" para que solo contenga "aaaa-mm-dd HH:MM", eliminando los segundos, puesto que el objetivo del dispositivo es que pasen minutos e incluso horas entre cada ciclo de operación.

```
// Función para obtener la fecha y hora sin segundos
void getDateTime(char* buffer, size_t len) {
    time_t now = time(nullptr);
    struct tm* timeinfo = localtime(&now);
    strftime(buffer, len, "%Y-%m-%d %H:%M", timeinfo);
}
```

10. Función setup: en esta función principal, al igual que en el broker, se iniciarán todos los parámetros y conexiones para que el dispositivo sensor sea capaz de funcionar. Se configuran el puerto Serial, el tiempo de funcionamiento del DeepSleep, los pines y verificaciones de los sensores, se conectan y sincronizan la conexión WiFi, la conexión MQTT y la hora por NTP, y finalmente se añade

uno al contador de despertados, para poder conocer cuantos ciclos se han ejecutado.

```
void setup() {
  Serial.begin(115200);

  esp_sleep_enable_timer_wakeup(TIEMPO_SLEEP);

  setup_wifi();
  client.setServer(mqtt_server, 1883);

  // Inicialización NTP
  configTime(gmtOffset_sec, daylightOffset_sec, ntpServer);
  struct tm timeinfo;
  while (!getLocalTime(&timeinfo)) {
    delay(500);
    Serial.println("Esperando sincronización NTP...");
  }

  pinMode(pinLDR, INPUT);

  while (!Serial)
    delay(10);
  #if USE_IIC
    Wire.begin();
  #else
    SPI.begin();
  #endif

  #if USE_IIC
    Serial.print("use I2C\n");
    bme.begin(ADDR_I2C, Wire);
  #else
    Serial.print("use SPI\n");
    bme.begin(PIN_CS, SPI);
  #endif
    Serial.print("run in forced_mode\n");
    if(bme.checkStatus())
    {
      if (bme.checkStatus() == BME68X_ERROR)
      {
        Serial.println("Sensor error:" + bme.statusString());
        return;
      }
      else if (bme.checkStatus() == BME68X_WARNING)
      {
        Serial.println("Sensor Warning:" + bme.statusString());
      }
    }
```

```
  bme.setTPH();
  bme.setHeaterProf(300, 100);

  bootCount++;
  Serial.print("\n\nWakeup número: ");
  Serial.println(bootCount);
}
```

11. Función loop: en esta función se definen todos los parámetros a leer, la conexión del publicador MQTT al broker y se dará formato a los datos que se desean enviar vía MQTT para que, finalmente, se bloquee el bucle y se inicie el modo de baja latencia, donde el dispositivo dejará de ejecutar las funciones hasta ahora comentadas.

```
void loop() {
  bme68xData data;

  if (!client.connected()) {
    reconnect();
  }
  client.loop();

  long now = millis();
  if (now - lastMsg > 10000) {
    lastMsg = now;
    bme.setOpMode(BME68X_FORCED_MODE);
    delay(1000+bme.getMeasDur()/200);

    if (bme.fetchData())
    {
      bme.getData(data);
      char buffer[32];

      // Publica la fecha y hora actual sin segundos
      getDateTime(buffer, 32);
      client.publish("Sensor/FechaHora", buffer);

      // Para temperatura (float)
      dtostrf(data.temperature, 1, 2, buffer);
      client.publish("Sensor/Temperatura", buffer);

      // Para presión (float)
      dtostrf(data.pressure, 1, 2, buffer);
      client.publish("Sensor/Presion", buffer);

      // Para humedad (float)
      dtostrf(data.humidity, 1, 2, buffer);
      client.publish("Sensor/Humedad", buffer);

      // Para gas resistance (float)
```

```
      dtostrf(data.gas_resistance, 1, 2, buffer);
      client.publish("Sensor/GasResistance", buffer);

      // Para status (int)
      sprintf(buffer, "%02X", data.status);
      client.publish("Sensor/Status", buffer);

      // Para LDR (int)
      sprintf(buffer, "%d", analogRead(pinLDR));
      client.publish("Sensor/Luz", buffer);

      Serial.println("Mensaje publicado");

      delay(5000);
      Serial.println("Entrando en Deep Sleep...");
      esp_deep_sleep_start();
    }
  }
}
```

5.- Evaluación y pruebas

A lo largo del desarrollo del sistema, se ha empleado un método de trabajo el cual emplea un enfoque paralelo entre el desarrollo de las funciones y los test a realizar, permitiendo una detección temprana de errores y una validación continua de cada función implementada. Gracias al uso de este método, se logra identificar y corregir incidencias a medida que el sistema va creciendo, garantizando estabilidad y funcionalidad.

Las pruebas se centraron en lograr la interoperabilidad de los sensores con servicios WiFi y comunicación MQTT. Gracias al empleo de Arduino IDE y a la aplicación MQTTExplorer, se ha podido verificar en todas las etapas del proyecto la funcionalidad. Arduino IDE se usó el puerto Serial para comprobar de forma inicial que los datos se leían con el formato apropiado, los ciclos del DeepSleep o datos de conectividades, y MQTTExplorer proporciona una visión global del tráfico en tiempo real de los mensajes que llegan al broker y la operatividad del mismo, además de ser útil para ver el formato de los datos enviados. Puede verse en la figura 9 como se puede ver el Serial del Arduino IDE a la izquierda y los datos publicados en el broker en MQTTExplorer a la derecha:

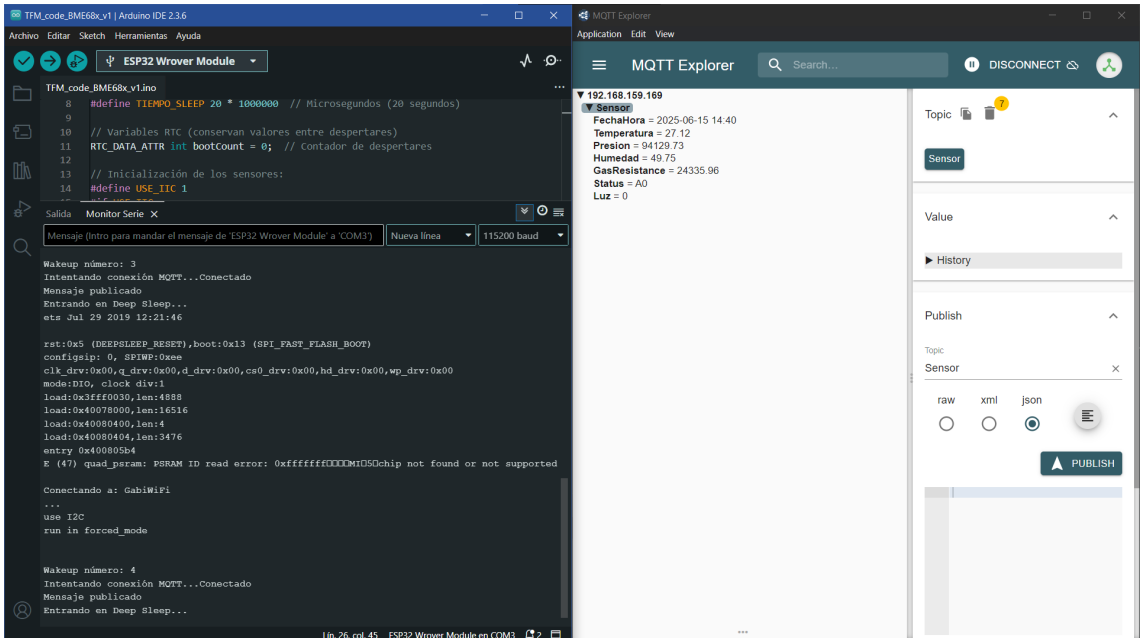

Figura 9. Interfaz de test: Arduino IDE y MQTTExplorer

Dos puntos críticos han sido a causa de la implementación del DeepSleep y del fijado de la dirección IP del broker MQTT:

- A la hora de incluir las funciones del DeepSleep, se observó tras ejecutar varios ciclos que la información se perdía entre ellos, por lo que se empleó la memoria RTC del ESP32 para almacenar la información entre ciclos. Por otro lado, se usó la sincronización NTP para que existiera linealidad entre las diferentes lecturas.
- Con respecto a la dirección IP del broker MQTT, el propio dispositivo la genera en su misma dirección IP y en el puerto 1883, pero a lo largo del proyecto, se ha observado que a lo largo del tiempo, si el dispositivo se apaga y enciende en diferentes ubicaciones, aunque esté conectado a la misma red, este es capaz de modificar su IP, teniendo que ajustar después el dispositivo sensor. Se han probado varias formas, pero finalmente ninguna prueba ha sido concluyente, puesto que el ESP32 si que bloqueaba la dirección, pero no era capaz de lanzar el servicio.

Con las últimas pruebas y resultados, se ha demostrado que el sistema es estable, capaz de publicar los datos y mantener linealidad entre ellos, incluso entre apagados del dispositivo por entrar en el estado DeepSleep o porque se apague la fuente de alimentación.

6.- Estudio económico

6.1.- Introducción

En el presente capítulo se realizará un estudio del coste económico que supone la realización del proyecto desarrollado.

Hasta este momento se ha estudiado la viabilidad de este proyecto desde un punto de vista técnico, es decir, si cumple o no el objetivo marcado desde el comienzo. Habiendo concluido este punto, queda pendiente comprobar la viabilidad económica del mismo.
En los siguientes apartados se estudiarán por separado costes directos e indirectos y finalmente se mostrarán los costes totales del proyecto.

A la hora de comenzar un proyecto, lo primero que se debe realizar es una planificación temporal del mismo, incluso antes de diseñar, programar o idear algo. Para ello, nos hemos basado en el siguiente índice, aunque hemos seguido un modelo de trabajo cíclico para verificar errores y añadir nuevas especificaciones:

1. Formación y documentación: en este primer apartado, clave a la hora de desarrollar cualquier proyecto, se incluirán la búsqueda de documentación, formación sobre los temas a desarrollar durante el proyecto y selección de los componentes necesarios. El tiempo de análisis de los componentes y tecnologías es muy importante dado que determina que, de aparecer contratiempos durante la duración del proyecto, no haya efectos en los plazos, puesto que se disponen de los útiles y conocimientos necesarios.

2. Estudio del problema: aquí se incluye el análisis del problema planteado y de la posible solución inicial propuesta para resolver el mismo. Ligado con el primer apartado, se evaluará el proyecto para buscar una solución óptima, funcional, que cumpla todos los requisitos y con aquellos requisitos adicionales.

3. Desarrollo de la aplicación: aquí tenemos que discernir dos partes, puesto que la solución final propuesta consta de dos dispositivos:
   - Programación del broker MQTT.
   - Diseño y programación del sensor, así como la disposición de los datos en el broker.

4. Puesta a punto del sistema (detección de errores): otro apartado clave dentro de los proyectos, comprobar que lo que hemos desarrollado funciona y cumple completamente con lo especificado en el primer punto. Para ello, será necesario testear el equipo en diferentes ubicaciones bajo diferentes condiciones.

5. Elaboración de la documentación: aquí se archivan las hojas de documentaciones y de la creación de esta memoria del proyecto. Adicionalmente, también se creará un manual de usuario del sistema.

	Mayo					Junio				
	S2518	S2519	S2520	S2521	S2522	S2523	S2524	S2525	S2526	S2527
Formación y documentación										
Estudio del problema										
Desarrollo del sistema										
Pruebas y puesta a punto del sistema										
Elaboración de documentación										

Tabla 2. Diagrama de Gantt del proyecto

6.2.- Recursos empleados

A continuación, se muestra un resumen de los recursos hardware y software empleados en el desarrollo de la aplicación. Es importante tener en cuenta únicamente la amortización del material durante el período de tiempo que ha sido utilizado en el proyecto, para calcular el coste real.

Software:

- Sistema operativo: Windows 10.
- Softwares de programación: Arduino IDE.

Hardware:

- Un ordenador portátil: Asus Zenbook 13.
- 2 Kits PCB con ESP-32.
- Sensores LDR KY-018 de AZDelivery.
- Sensor BME680 de SEENGREAT.

Material ofimático:

- Libros de consulta.
- Otros consumibles.

6.3.- Costes directos

En cuanto a los costes directos, se evalúan:

- Coste del personal.
- Cortes amortizables de programas y equipos.
- Coste de materiales directos empleados.

6.3.1.- Costes del personal

La realización del presente proyecto ha sido llevada a cabo por un ingeniero encargado del diseño y puesta a punto del sistema y aplicaciones correspondientes.

Se calcula el coste anual para un ingeniero para posteriormente adecuarlo al número de horas trabajadas por el mismo. Este coste anual incluye:

- Sueldo bruto anual, así como los posibles incentivos por su trabajo.
- Cotización a la Seguridad Social, que es un 35% del sueldo bruto.

Teniendo en cuenta esto, el coste anual del ingeniero será:

COSTE ANUAL	
Sueldo bruto más incentivos	32 744,44 €
Seguridad Social (35% sueldo bruto)	11 453,56 €
Coste total	**44 178,00 €**

Tabla 3. Coste anual del personal

Se calcula a continuación una estimación de los días efectivos trabajados al año:

DÍAS EFECTIVOS POR AÑO	
Año medio	365,25 días
Sábados y Domingos	-104,36 días
Días de vacaciones efectivos	-20,00 días
Días festivos reconocidos	-15,00 días
Días perdidos estimados	-5,00 días
Total días efectivos estimados	**220,89 días**

Tabla 4. Días efectivos por año

Conociendo ya el número total de días efectivos de trabajo, y que la jornada laboral es de 8 horas, obtenemos el total de horas efectivas de trabajo:

220,89 días/año x 8 horas/día = 1 767,12 horas/año

El coste por hora de un ingeniero se calcula como la división del sueldo anual entre las horas efectivas trabajadas al año:

Coste hora = 44 178[€/año] / 1 767,12[h/año] = **25 €/hora**

En la siguiente tabla (Tabla 5) se muestra una distribución temporal aproximada del trabajo empleado por el ingeniero para el desarrollo del proyecto:

DISTRIBUCIÓN TEMPORAL DE TRABAJO	
Formación y documentación	45 horas
Estudio del problema	35 horas
Desarrollo de la aplicación	120 horas
Pruebas y puesta a punto del sistema	140 horas
Elaboración de la documentación	60 horas
Total de horas empleadas	**400 horas**

Tabla 5. Distribución temporal de trabajo

El coste personal directo es calculado como la multiplicación de las horas y el coste efectivo de una hora de trabajo del ingeniero:

400 horas x 25 €/hora = 10000 €

COSTE PERSONAL DIRECTO	**10 000 €**

6.3.2- Costes de amortización de equipos y programas

Para el cálculo de estos costes se debe realizar previamente la inversión total y calcular la amortización lineal correspondiente según los criterios aconsejados por la ley. En este apartado estudiaremos tanto los costes de la amortización del material de oficina como

los costes de amortización de los equipos. En este caso, los softwares empleados son gratuitos.

Se estima como tiempo de amortización un periodo de **3 años**, ya que es el considerado como vida útil de dicho material. De forma que al calcular el coste hay que multiplicar por un factor del [p.ej. 0.33 para 3 años] los precios mostrados:

MATERIAL	IMPORTE (aprox.)	AMORTIZACIÓN (33,3%)
S.O. Windows 10	129,99 €	43,33 €
Ordenador Asus Zenbook	800 €	266,67 €
Paquete ofimático Office 365	149 €	49,67 €
PCBs y ESP-32	14,99 €	5 €
Sensores LDR (x3)	5,99 €	2 €
Sensor BME680	18,15 €	6,05 €
TOTAL MATERIAL	**1 118,12 €**	**372,72 €**

Tabla 6. Amortización del material empleado

Hay que apuntar que los programas informáticos no incluidos anteriormente son de libre distribución y su coste, por tanto, es cero.

El coste final por hora de utilización del material es calculado mediante la división de la amortización anual entre el número de horas de uso en dichos equipos.

Coste final/hora = 372,72[€/año] / 1767,12[horas/año] ≃ **0,21 €/hora**

Se considera el tiempo de uso como el tiempo total necesario para la realización del proyecto, por ser necesario en las etapas de análisis, diseño, programación y documentación. Por tanto, el coste de amortización de material será:

400 horas x 0,21 €/hora = 84,38 €

COSTE DE AMORTIZACIÓN DE MATERIAL DE OFICINA	**84,38 €**

6.3.3.- Costes derivados de otros materiales

Los costes recogidos a continuación se denominan consumibles, e incluyen, por ejemplo, libros de consulta, papel de impresora, fotocopias, cartuchos de tinta, etc.

Este tipo de material es necesario para la realización de los diferentes trabajos, tanto en la fase de desarrollo y edición, impresión de listados, manuales e informes, almacenamiento de programas y documentos, etc.
El coste total de este material es de 140 €.

COSTE DE CONSUMIBLES	**140,00 €**

6.3.4.- Costes directos totales

De todos los costes obtenidos anteriormente concluimos que los costes directos totales son los derivados de la suma de los costes de personal, amortización de material y de consumibles. Por tanto, será:

10000 € + 1118,12 € + 84,37 € + 140,00 € = 11342,49 €

COSTES DIRECTOS	**11 342,49 €**

6.4.- Costes indirectos

Los costes indirectos son los gastos producidos por la actividad requerida para la elaboración del proyecto y que no se pueden incluir en ninguno de los gastos directos.

COSTES INDIRECTOS PARCIALES	
Dirección y servicios administrativos	150,00 €
Consumo de electricidad	140,00 €
Consumo de telefonía	20,00 €
Consumo de desplazamiento	100,00 €
Total gastos indirectos	**410,00 €**

Tabla 7. Costes indirectos

Por tanto, los costes indirectos totales ascienden a:

COSTES INDIRECTOS	**410,00 €**

6.5.- Costes totales

Los costes totales son el resultado de sumar los gastos directos e indirectos, siendo el montante total para este proyecto:

COSTES TOTALES	
Costes directos	11 342,49 €
Costes indirectos	410,00 €
Coste total del proyecto	**11 752,49 €**

Tabla 8. Costes totales

En conclusión, el coste total del proyecto asciende a la cantidad de:

COSTES TOTALES DEL PROYECTO	**11 752,49 €**

7.- Conclusiones y líneas futuras

7.1.- Conclusiones

El proyecto cumple con todos los objetivos inicialmente determinados, logrando un final satisfactorio gracias a, como ya se ha demostrado, disponer un funcionamiento estable y fiable.

A lo largo del proyecto se han ido generando diferentes retos que han necesitado de análisis, pruebas y ajustes para lograr corregir errores o implementar nuevas funciones. Los sensores y la comunicación MQTT, unidos a la necesidad de garantizar la fiabilidad de los datos enviados, han necesitado especial atención a la hora de diseñar el hardware y el software. .

Finalmente, este proyecto ha supuesto una oportunidad para poner en práctica los conocimientos adquiridos a lo largo de mi formación académica. Mediante la implementación de sensores con comunicación inalámbrica y la gestión eficiente de la energía, se ha logrado construir un sistema capaz de capturar y transferir datos con un bajo coste de producción, de consumo y de mantenimiento, permitiéndome afianzar conceptos teóricos junto a mejorar mis habilidades de planificación, adaptación, análisis y búsqueda de soluciones.

7.2.- Líneas futuras

Gracias a la comunicación IoT, el sistema es fácilmente escalable, pudiendo añadir otros sensores, que midan los mismos parámetros u otros, pero en otras ubicaciones, e incluso implementar actuadores, que funcionen de forma automática y, en consecuencia, de variables en los entornos, con el objetivo de crear un entorno domótico.

Como líneas a investigar para el sensor, podrían tomarse las siguientes:

- Se podría estudiar en detalle los consumos del mismo con el objetivo de optimizar los tiempos de activación.
- Se podría analizar el consumo medio del dispositivo para crear un sistema portátil, con una batería y una pequeña placa solar, prolongando la autonomía del dispositivo y pudiendo ser implementado en cualquier lugar.
- Ligado con el anterior, podría implementarse un sistema de motores y un PID para optimizar la carga de la placa solar.
- Implementación sobre un ESP32 con un módulo GSM, sustituyendo la red MQTT por la red telefónica, evitando los costes de mantenimiento del broker y el límite del área de aplicación, así como problemas con las direcciones IPs variables.

8. Bibliografía

[01] MQTT amb entorns gràfics de programació - https://www.scoop.it/topic/ipee/?&tag=Internet+de+las+cosas - Acción Lamas, Angel (18 de noviembre de 2023).

[02] Página principal de Arduino - https://www.arduino.cc/ - Arduino.

[03] Protocolo MQTT - https://aprendiendoarduino.wordpress.com/tag/mqtt-qos/ - Arduino (20 de noviembre de 2023).

[04] Página principal de AzDelivery - https://www.az-delivery.de/es - AzDelivery.

[05] La evolución de la tecnología de sensores) - https://www.balluff.com/es-es/historia – Balluff.

[06] Internet de las cosas: cuando todo está conectado - https://www.lavanguardia.com/vida/junior-report/20190301/46752655177/internet-cosas-dispositivos-conectados-iot.html - Barchilón, Miriam (13 de octubre de 2023).

[07] ¿Qué es MQTT? - https://www.youtube.com/watch?v=T1_w8-8Y5kc - ChepeCarlos (21 de septiembre de 2023).

[08] Ejemplo MQTT Python - https://www.youtube.com/watch?v=T362losqJys - ChepeCarlos (21 de septiembre de 2023).

[09] “Sensores: los pioneros de la digitalización” - https://convertronic.net/noticias/tecnologia/10729-sensores-los-pioneros-de-la-digitalizacion.html - Convertronic (8 de septiembre de 2022).

[10] “Sensores inteligentes : una historia con futuro” - https://upcommons.upc.edu/handle/2099/9553 - Custodio Ruiz, Ángel/Bragós Bardia, Ramón/Pallàs-Areny, Ramón.

[11] Cómo comunicar un ESP32 con una página web a través de MQTT - https://www.electrogeekshop.com/como-comunicar-un-esp32-con-una-pagina-web-a-traves-de-mqtt/ - Damián, Juan (30de septiembre de 2023).

[12] How to Enable WebSockets for Mosquitto MQTT Broker - https://cedalo.com/blog/enabling-websockets-over-mqtt-with-mosquitto/ - Domin, Bohdan (30 de septiembre de 2023).

[13] Página principal de Espressif - https://www.espressif.com/en – Espressif.

[14] TLS connection with client certificate validation issue between MQTT bróker and ESP32 client - https://github.com/espressif/arduino-esp32/issues/5021 - Gal, Norbert (30 de septiembre de 2023).

[15] Protocolo MQTT: conceptos que debes saber - https://www.youtube.com/watch?v=-fQzbyLqsMc - INNOVA DOMOTICS (4 de octubre de 2023).

[16] Uso de MQTT para control de dispositivos de IoT - https://riunet.upv.es/bitstream/handle/10251/173841/Lliso%20-%20Uso%20de%20MQTT%20para%20el%20control%20de%20dispositivos%20de%20IoT.pdf?sequence=1 - Lliso Cosin, Alejandro (29 de junio de 2023).

[17] Universidad Isabel I: Historia y origen del IoT - https://www.ui1.es/blog-ui1/historia-y-origen-del-iot - Martinez Martinez, Victor (5 de abril del 2022).

[18] Using MQTT with Robots (and Home Automation) - https://www.youtube.com/watch?v=4UIUC1YU_Sk - McAleer, Kevin (27 de septiembre de 2023).

[19] ESP32 MQTT Publish Subscribe with Arduino IDE - https://microcontrollerslab.com/esp32-mqtt-publish-subscribe-arduino-ide/ - MicrocontrollersLab (22 de septiembre de 2023).

[20] Explorador de brokers MQTT - https://mqtt-explorer.com/ - MQTTExplorer (7 de diciembre de 2023).

[20] "Desarrollo de sistema de asistente por voz para control de un robot móvil" - https://uvadoc.uva.es/handle/10324/852/browse?authority=4753e2aa-efbd-4e2a-830c-6ca27b85b12b&type=author - Nieto Ramos, José Gabriel (2 de febrero de 2024).

[21] ¿Qué es MQTT? - https://www.paessler.com/es/it-explained/mqtt - Paessler (1 de septiembre de 2023).

[22] How to Use the Paho MQTT Client in Python with Examples - https://cedalo.com/blog/configuring-paho-mqtt-python-client-with-examples/ - Schiffler, Andreas (10 de octubre de 2023).

[23] Página principal de SeenGreat - https://seengreat.com/ - SeenGreat.

[24] Github de SeenGreat - https://github.com/seengreat – SeenGreat.

[25] Github de SeenGreat para el BME680 - https://github.com/seengreat/BME68X-Environmental-Sensor – SeenGreat.

[26] Connecting ESP32 to a MQTT Broker with SSL Certificate - https://forum.arduino.cc/t/need-of-a-guidance-of-securely-connecting-a-esp32-to-a-mqtt-broker-with-ssl-certificate-with-a-static-ip/996519 - Sidsrihari (7 de septiembre de 2023).

[27] Paho Python MQTT Client Subscribe With Examples - http://www.steves-internet-guide.com/subscribing-topics-mqtt-client/ - Steve's Internet Guide (2 de octubre de 2023).

[28] How to use ESP32 MQTTS with MQTTS Mosquitto Broker (TLS/SSL) - http://www.iotsharing.com/2017/08/how-to-use-esp32-mqtts-with-mqtts-mosquitto-broker-tls-ssl.html - Tech It Yourself (28 de agosto de 2023).

[29] Python: Publisching messages to MQTT topic - https://techtutorialsx.com/2017/04/14/python-publishing-messages-to-mqtt-topic/ - techtutorialsx (19 de octubre de 2023).