Sistema automático de iluminación para las calles

Antonio Moya Toapanta

Resumen– En este trabajo he desarrollado un sistema IoT orientado a las ciudades inteligentes, específicamente para automatizar y controlar el alumbrado público mediante sensores PIR. El objetivo es crear un sistema integrado utilizando aplicaciones y herramientas ampliamente empleadas en sistemas IoT industriales, tales como el microcontrolador ESP32, el protocolo MQTT con el broker Mosquitto, Node-RED para el procesamiento de los datos, InfluxDB como base de datos no relacional y Grafana como herramienta de visualización. Este sistema podría convertirse en una solución eficaz para la gestión urbana, optimizando el consumo energético, reduciendo la contaminación lumínica, aumentando la seguridad ciudadana y mejorando el análisis de los patrones de movimiento de las personas.
El sistema propuesto ha sido validado en un entorno simulado, mostrando en Grafana tanto los datos actuales de movimiento como el histórico, además de probar exitosamente su comunicación con múltiples dispositivos ESP32. El sistema es compacto, asequible, escalable y sencillo de implementar, haciéndolo ideal para su uso en las futuras ciudades inteligentes.

Palabras clave– IoT, Ciudades inteligentes, ESP32, MQTT, sensores PIR, Node-RED, eficiencia energética, visualización de datos.

Abstract– In this final degree project, I have developed an IoT system aimed at Smart Cities, specifically designed to automate and control public lighting using PIR sensors. The goal is to create an integrated system by employing widely used applications and tools in industrial IoT environments, such as the ESP32 microcontroller, the MQTT protocol with the Mosquitto broker, Node-RED for data processing, InfluxDB as a non-relational time-series database, and Grafana as the visualization tool. This system could become an effective solution for urban management, optimizing energy consumption, reducing light pollution, increasing public safety, and improving the analysis of human movement patterns.
The proposed system has been validated in a simulated environment, displaying both real-time and historical movement data in Grafana, and successfully demonstrating communication with multiple ESP32 devices. The system is compact, affordable, scalable, and easy to implement, making it ideal to use in the Smart Cities of the future.

Keywords– IoT, Smart Cities, ESP32, MQTT, PIR sensors, Node-RED, energy efficiency, real-time monitoring.

1 Introducción

Las ciudades inteligentes son aquellas que integran tecnologías para gestionar de forma eficiente sus servicios (como el transporte, la energía, los residuos o la seguridad) con el objetivo de ser más sostenibles, eficaces y mejorando la vida de los ciudadanos.

Para lograrlo, se requiere una gran cantidad de microsistemas capaces de regular, de forma localizada, cada uno de los elementos mencionados. Este enfoque distribuido y automatizado es clave para avanzar hacia una gestión urbana más inteligente y adaptable a las necesidades reales de la población. El problema que planteo mitigar es el consumo energético excesivo y la contaminación lumínica que genera el alumbrado público al estar encendido de forma continua. Con un sistema IoT que regule automáticamente su funcionamiento, podemos dar solución a esta problemática.

• *E-mail de contacto:* 1665873@uab.cat
• *Trabajo tutorizado por:* Raúl Aragonés Ortíz (Área de Arquitectura y Tecnología de Computadores)
• *Curso 2024/25*

1.1. Estado del arte

Actualmente, el crecimiento urbano y la preocupación por la obtención de recursos, como el cambio de paradigma de cómo administrar una ciudad, ha supuesto la creación de las ciudades inteligentes o ”Smart Cities”. En la UE, ciudades como Oslo llevan implementando sistemas de redes inteligentes desde el 2006.

Casos de éxito:

En la capital de Columbus (EEUU) se implementaron los LEDs inteligentes, instalando 58 000 farolas con un control centralizado en tiempo real, reduciendo los costes operacionales significativamente.

En Bangladesh, un sistema IoT se utiliza en varias ciudades y reduce hasta un 60 por ciento del consumo eléctrico a través del control lumínico y la detección de los fallos.

En proyectos como en Edimburgo, Doncaster y Sheffield se demuestra que la adaptación al uso urbano es más eficiente que las regulaciones temporales.

Los sistemas previamente mencionados comparten las siguientes características:

- Ahorro energético de hasta un 80 por ciento.
- Mantenimiento activo.
- Mejora de la seguridad.
- Escalabilidad significativa y capacidad de mejoras continuas.

1.2. Motivación

La motivación de este trabajo recae en la evidente ineficiencia que presentan sistemas como el alumbrado público. Son notorias las carencias que provoca la falta de una gestión inteligente del alumbrado, tales como el excesivo consumo de electricidad y la gran contaminación lumínica que genera en el entorno. Las ciudades del mañana no solo deben ser eficientes, sino que deben llegar a ser un elemento más de la naturaleza.

1.3. Objetivo principal

El objetivo principal es crear un sistema IoT para detectar movimiento y automatizar el encendido del alumbrado

1.3.1. Objetivos específicos

- Diseñar una red basada en ESP32[4] y sensores de detección.
- Integrar el sistema con un protocolo de comunicación.
- Procesar los datos.
- Crear una base de datos.
- Visualizar los datos obtenidos a través de una ”dashboard”.
- Evaluar la viabilidad y escalabilidad del sistema.

1.4. Marco Teórico

Para poder implementar nuestro sistema necesitamos tener los siguientes conocimientos:

1.4.1. Conocimientos previos

- Las ciudades inteligentes son entornos urbanos que integran tecnologías de la información y la comunicación (TIC) para mejorar la eficiencia de los servicios públicos que brinda a sus habitantes.
- IoT es una red de dispositivos físicos (sensores, actuadores y microcontroladores) conectados a internet que recopilan y transmiten datos para automatizar tareas. Esta tecnología permite la comunicación máquina a máquina (M2M), permitiendo una interconexión entre dispositivos clave para la digitalización de las ciudades.
- Sensores PIR (passive infrared sensors) [17] son sensores que detectan cambios en la radiación infrarroja emitida por objetos en su campo de visión, normalmente utilizados para captar movimiento humano.
- Sensores de distancia por ultrasonidos son sensores que se basan en el principio de tiempo de vuelo. Este principio se refiere a la medición en tiempo que tarda una señal en viajar desde el emisor hasta un objeto y regresar al receptor.

$$\text{Distancia} = \frac{t \times v}{2}$$

donde:

  - t es el tiempo de vuelo (segundos),
  - v es la velocidad del sonido (metros/segundos)

- Los microcontroladores son dispositivos electrónicos integrados que contienen una unidad central de procesamiento, memoria y periféricos de entrada y salida, diseñados para controlar sistemas embebidos. Son el elemento capaz de permitir la conexión y el procesamiento local de datos provenientes de sensores y actuadores. Las características de bajo consumo energético, comunicación inalámbrica, flexibilidad en la programación y la fácil integración con distintos sensores hacen que sean perfectos para aplicaciones IoT.
- La comunicación entre dispositivos es crucial para sistemas IoT. MQTT (Message Queuing Telemetry Transport) [14] es un protocolo de comunicación ligero basado en modelos de publicación/suscripción, permitiendo una comunicación eficiente en redes de poca capacidad. El broker es el servidor intermediario que recibe los datos de los publicadores y los distribuye a los suscriptores según los tópicos a los que están suscritos. Esta herramienta permite una comunicación sencilla y descentralizada; es como estar suscrito a un periódico: recibes información constantemente, pero si te das de baja, dejan de enviarte el periódico.
- Necesitaremos una plataforma para la gestión y el procesamiento de los datos. Esta herramienta permite obtener información de los sensores, procesarla y transformarla mediante funciones, y enviarla a sistemas de almacenamiento y visualización, automatizando la comunicación entre elementos y creando flujos de datos complejos sin necesidad de programación avanzada.

- Una base de datos es un sistema organizado que permite almacenar, gestionar y recuperar datos de manera sencilla. En nuestro caso, necesitamos una base de datos especializada en series temporales, ideal para manejar una gran cantidad de datos secuenciales de forma eficiente, manteniendo la información temporal de los datos, permitiendo realizar consultas rápidas y análisis tanto en tiempo real como históricos, un tipo de base de datos perfecta para almacenar datos de sensores en un sistema IoT.

- Las plataformas de visualización de datos permiten crear paneles que muestran la información en diferentes formatos gráficos, facilitando la interpretación de los datos y ofreciendo monitorización continua y en tiempo real.

2 METODOLOGIA

Mi sistema IoT tiene las siguientes partes:

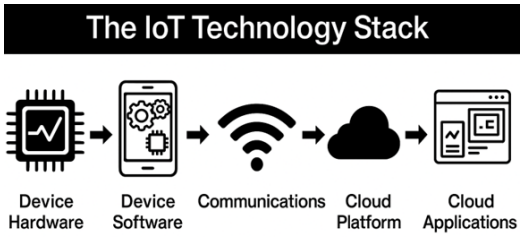

Fig. 1: Estructura utilitzada

2.1. Componentes

2.1.1. Device hardware

El microprocesador que he elegido para el sistema es el ESP32. Este microprocesador tiene bajo coste, wifi integrado, buena capacidad de procesamiento y es compatible con Arduino IDE [1]. La facilidad para capturar datos desde sensores y enviarlos de manera inalámbrica sin necesidad de módulos externos lo hace ideal para nuestro sistema IoT.

Fig. 2: ESP32-WROOM-32

Para poder detectar a los peatones utilizaré varios sensores, como el PIR(HC-SR501) [17], que es económico, fiable y fácil de implementar.

Fig. 3: Sensor HC-SR501

También utilizaré el sensor ultrasónico (HC-SR04)[16] [15] que permite una medición más precisa de la distancia. Su funcionamiento es por ultrasonido, manda un pulso acústico de 40 kHz y mide el tiempo que tarda en recorrer la distancia entre el módulo y el objeto, con un rango efectivo típico de 2 cm a 400 cm.

Fig. 4: Sensor HC-SR04

Como último sensor, utilizaré el módulo de reloj en tiempo real(DS1302 RTC) [11] [10] que permite mantener la hora para sistemas sin alimentación ni acceso a la red wifi. Con este sensor el microcontrolador podrá obtener la hora exacta y en función de la hora regularse como es debido.

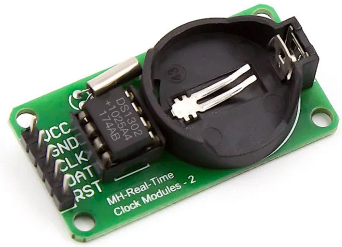

Fig. 5: Sensor DS1302 RTC

2.1.2. Device Software

Como software utilizaré Arduino IDE (Integrated Development Environment), una aplicación de código abierto que permite escribir, compilar y subir fácilmente código a microcontroladores Arduino y otros compatibles.

2.1.3. Communications

Como protocolo de comunicación utilizaré MQTT (con el broker Mosquitto), que es ligero, rápido e ideal para comunicaciones M2M. El broker Mosquitto actúa como servidor, recibiendo y distribuyendo los datos a los componentes necesarios.

2.1.4. Cloud Platform

Node-RED [12] es una plataforma de desarrollo visual basada en flujos. Estos flujos se crean a través de bloques o nodos que permiten conectar, procesar y transformar datos. En este sistema, se encargará de recibir los mensajes MQTT enviados por el ESP32, procesarlos según las funciones definidas y enviarlos a la base de datos (InfluxDB) [8].

Influxdb es una base de datos diseñada para almacenar datos con marca temporal, permitiendo un almacenamiento rápido de alta disponibilidad y la recuperación de datos de series temporales.

2.1.5. Cloud Aplication

Grafana [6] es una plataforma de código abierto que permite visualizar datos a través de dashboards interactivos, que monitorizan información en tiempo real mediante gráficos, indicadores y tablas personalizables.

2.1.6. Sinergia entre elementos del sistema

Esp32 + MQTT (Mosquitto), es la comunicación en tiempo real, fiable y desacoplada del software y hardware.

MQTT + Node-Red, es la integración visual sin código complejo y el flujo de datos automatizado desde los sensores hasta el almacenamiento.

Node-Red + InfluxDB, es el almacenamiento automático, estructurado y escalable de los datos.

InfluxDB + Grafana permite una visualización completa y comprensible de los datos obtenidos, siendo útil para el monitoreo y la toma de decisiones.

Para hacer más evidente la interacción entre los elementos del sistema IoT, el siguiente diagrama expone el flujo de información 6.

InfluxDB + Grafana permite una visualización completa y comprensible de los datos obtenidos, siendo útil para el monitoreo y la toma de decisiones.

3 IMPLEMENTACIÓN

Validaré el sistema IoT a través de una prueba en local desde mi dispositivo Windows 10, este entorno simulado me permitirá comprobar la correcta detección de los sensores, el envío de datos a través del protocolo MQTT, su procesamiento y almacenamiento y la posterior visualización del panel de monitoreo en tiempo real.

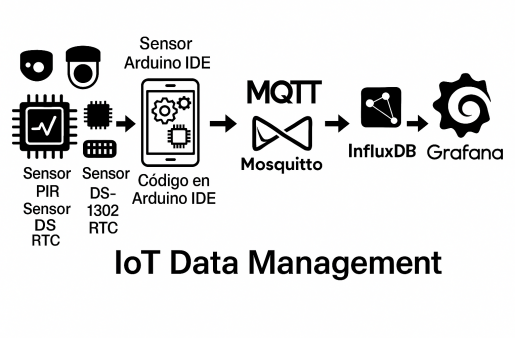

Fig. 6: Flujo de la información

Primero, explicaré el montaje del sistema físico en su totalidad.

El diagrama de los elementos físicos es:

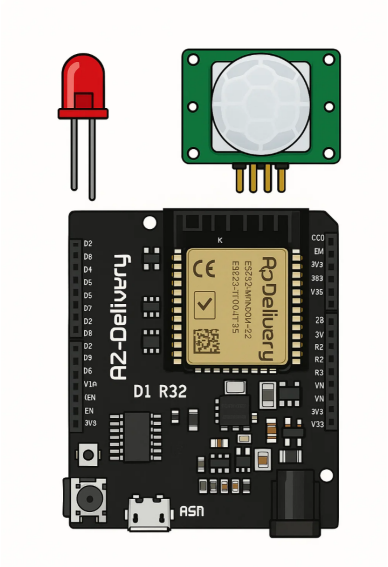

Fig. 7: Nodo A

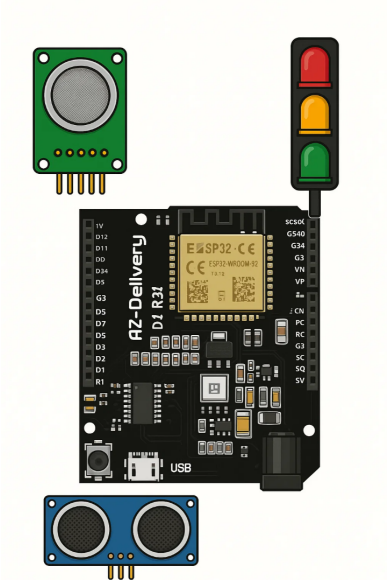

Fig. 8: Nodo B

El sistema tiene dos formas de detección de movimien-

to: el sensor HC-SR501 que funciona con infrarrojos donde la detección es inmediata, y, por otro lado, el sensor HC-SR04 que puede detectar el acercamiento o retroceso de un objeto. Al tener dos modos de detección, nuestra precisión aumenta. Por eso, tenemos un Nodo A y un Nodo B. La conectividad de los sensores es según esta tabla para mi sistema:

Nodo	Componente	Pin del componente	Pin en ESP32
A	LED (rojo)	GND	GND
	LED (rojo)	VCC (ánodo)	3 .3 V
	LED (rojo)	Señal (salida)	**GPIO 27**
	Sensor PIR HC-SR501	VCC	5 V
	Sensor PIR HC-SR501	GND	GND
	Sensor PIR HC-SR501	OUT	**GPIO 13**
B	DS1302 RTC	VCC	3 V
	DS1302 RTC	GND	GND
	DS1302 RTC	CLK	**GPIO 18**
	DS1302 RTC	DAT	**GPIO 19**
	DS1302 RTC	RST (CE)	**GPIO 23**
	HC-SR04	VCC	5 V
	HC-SR04	GND	GND
	HC-SR04	TRIG	**GPIO 26**
	HC-SR04	ECHO	**GPIO 25**
	LED semáforo	GND	GND
	LED semáforo	GREEN (ánodo)	**GPIO 27**
B	LED semáforo (amarillo)	ánodo	GPIO 12
B	LED semáforo (rojo)	ánodo	GPIO 13

Fig. 9: Conexiones entre los sensores y la placa ESP32

En mi dispositivo Windows descargué y configuré los programas MQTT mosquitto, InfluxDB (versión 1.8 funciona

(plug and play) con Node-Red), Node-Red (Node. Js) y Grafana.

Para inicializar los programas se utiliza el PoweShell o CMD y los siguientes comandos:

-Para inicializar MQTT mosquitto-

Abrimos el archivo (mosquitto.config) y escribimos ”listener 1883 allow-anonymous true”, este comando nos permitirá usar el puerto 1883 para la comunicación entre los elementos y permite su uso sin identificación usuario/contraseña. Seguidamente, usamos el comando `C:\Program Files\mosquitto` para verificar su correcto funcionamiento.

-Para iniciar InfluxDB-

Para iniciar la aplicación se utiliza el comando `cd C:\Users\toni\influxdb-1.8.10-1\influxd`, para crear la base de datos usaremos `cd C:\Users\toni\influxdb-1.8.10-1\influx`. Se abrirá un shell conectado a http://localhost:8086, en esta versión navegaremos por la base de datos a través de la terminal o CMD.

Crearemos la base de datos usando ”(CREATE DATABASE sensores)”. En la siguiente imagen podemos ver los elementos de la base de datos.

La estructura de la base de datos es: -Database:sensores Measurements: Movimiento es un valor integer 0/1. Distancia es un valor float, obteniendo distancia con decima-

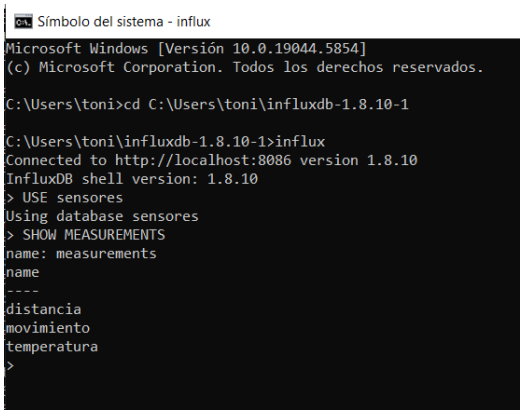

Fig. 10: Base de datos Utilizada

les. Cada entrada queda almacenada con un timestamp automático, permitiendo las consultas de series temporales.

-Para inicializar Node-Red-

Abriremos el terminal o CMD y ejecutaremos (npm install -g –unsafe-perm node-red) y lo iniciamos escribiendo ”(node-red)”(http://localhost:1880). En los nodos MQTT se conectan al tópico ESP32/movimiento y ESP32/distancia respectivamente, los nodos con las funciones porque InfluxDB únicamente entiende los datos como objetos con campos numéricos y nombre de medición; sin la transformación no sabrá como guardarlos ni interpretarlos. El nodo InfluxDB inserta en la base de datos sensores, la medición de movimiento y distancia. También he establecido una comunicación entre el nodo ESP32 y ESP32, donde el nodo A envía un valor=1 y cuando lo recibe el nodo B, lo muestra.

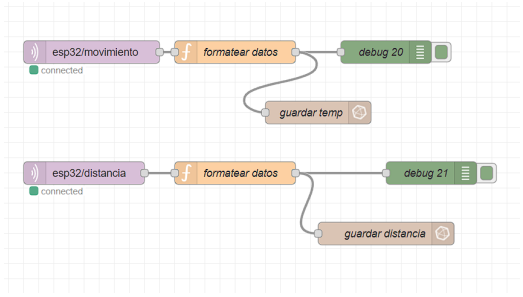

Fig. 11: Diseño Nodos A y B Node-Red

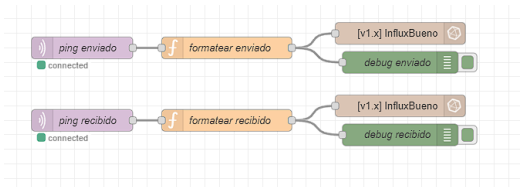

Fig. 12: Diseño Comunicación entre ESP32 y ESP32

-Para inicializar Grafana- Para acceder a la interfaz es a través de http://localhost:3000. Para que obtenga la información de la base de datos, vamos a ”(DATA SOURCES)”luego ”(ADD DATA SOURCE)z la base de datos se llama ”(Influxdb-1)”. Para acceder a los datos, utilizamos las peticiones o querys . Seguidamente, creamos un panel

y lo configuramos para nuestro sistema. Los paneles creados son cuatro ”stat”, donde muestran un único número, con dos paneles que muestran el historial de detección de movimiento y distancia. En la siguiente imagen, se ve el panel que indica si hay movimiento que recibe información del sensor de infrarrojos y la distancia en cm del sensor de ultrasonidos. Debajo de esos paneles, el historial de detección de movimiento y distancia respectivamente. Los últimos dos paneles son dos valores que muestran que ha habido una comunicación entre los ESP32 correcta.

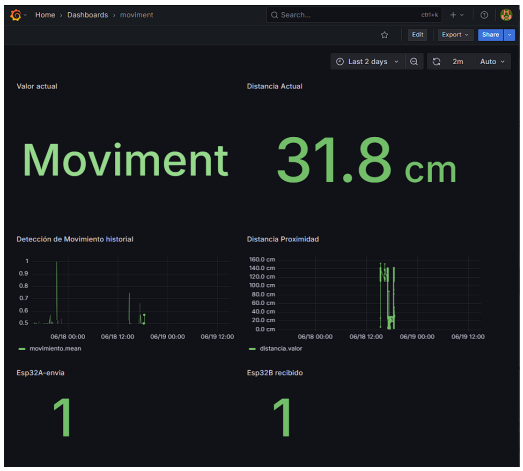

Fig. 13: Visualización de los datos obtenidos

-Código Arduino- He desarrollado dos scripts en Arduino, .^esp32A.ino”para el nodo A y .^esp32Bsem.ino”para el nodo B. Los códigos están escritos en C++ a través del entorno de Arduino IDE. Debido a su extensión esta la imagen completa con comentario en el anexo.

3.1. Desafíos Superados

El desafío constante ha sido el deterioro de los ”drivers”de los puertos para el Arduino IDE. Al no saber que era culpa de un problema de software hizo que dedicase bastante tiempo para diagnosticar el problema. Su solución es la reinstalación del puerto CH340.

El siguiente gran problema ha sido el mal funcionamiento del sensor DS1302 RTC, que marca un tiempo incoherente debido a problemas físicos del sensor, he podido solucionarlo utilizando un servicio NTP (Network Time Protocol), el cual es capacidad integrada en el ESP32 obteniendo la hora exacta a través de internet.

3.2. Aplicación Hipotética

Realizaremos una implementación en una hipotética calle estándar de un barrio de Terrassa. Calle con 25 farolas.

El coste del sistema (Nodo B para 25 farolas) es:

- ESP32 DevKit c4: 200 € (AZ-Delivery tienda oficial).
- Sensor HC-SR04: 50 € (Naylamp Mechatronics).
- Sensor HC-SR501: 62,5 € (Naylamp Mechatronics)
- Módulo DS1302 RTC: 37,5 € (Naylamp Mechatronics).
- Cableado/tornillería: 20 € (Amazon).
- Mano de obra (Instalación y mantenimiento): 750 €.
- Total elementos e instalación: 1120 €.

  Ahorro energético estimado:

- Algoritmo actual (LED al 100 %, si la distancia medida es menor de 20 cm).
- Tiempo activo aproximadamente un 30 por ciento.
- Consumo medio estimado: 63 kWh/año.
- Ahorro por farola en electricidad y precio: 168 kWh - 63 kWh = 105 kWh/año x 0,07 € = 7,4 €/año por farola.
- Ahorro para 25 farolas: 185 €/año.

El (pay-back) o ROI(return of investment) es inversión inicial 1057 € entre el ahorro anual 185 € que da un aproximado de 6 años de amortización [7] [5] [13] [9] [3] [18] [2].

3.3. Mejoras Futuras

Las mejoras futuras del sistema IoT son significativas, mejoras como:

- Integrar el uso de paneles solares al sistema para tener un consumo casi inexistente y poder agregar energía limpia a la red, facilitando una implementación sostenible y autosuficiente.
- Integración con plataformas en la nube. La vinculación con plataformas como AWS IoT, Azure IoT o Google cloud IoT, permitiría una monitorización remota, análisis predictivo y administración centralizada facilitando la toma de decisiones con entes públicos superiores.
- Integracion con IA. Los algoritmos de inteligencia artificial podrían obtener patrones, detectar anomalías y mejorar la información obtenida a los entes públicos para la toma de decisiones.
- Mejora la interfaz gráfica. Desarrollar interfaces gráficas más interactivas y personalizadas, mejorando la gestión del sistema.
- Seguridad y privacidad de datos. La implementación en una ciudad real implicaría la necesidad de tener un protocolo mas robusto cifrando las capas del sistema para garantizar la protección integral de datos.

4 Conclusiones

Nuestro sistema IoT demostró ser funcional y proporcionar una respuesta estable entre los diferentes elementos. Con los sensores elegidos podemos obtener un funcionamiento óptimo para nuestro sistema. La sinergia que obtenemos de parte de los sensores de proximidad, donde el HC-SR501 tiene una detección instantánea de la presencia, el HC-SR04 nos da una medición más cuantitativa permitiendo una información más precisa junto con el DS1302

RTC que nos informa de la temporalidad de los datos. Nuestro sistema es completo, nos da rapidez de respuesta, información urbanística útil, bajo coste, eficiencia energética y escalabilidad, siendo muy adecuado para la automatización inteligente del alumbrado público.

El leve consumo energético del sistema, de aproximadamente 80 mA en reposo y picos de más de 200 mA al enviar información y activar los actuadores, refleja una poca aportación de consumo a la red eléctrica de los ayuntamientos. Al ser elementos pequeños, la instalación en el alumbrado sería sencilla y sin grandes complicaciones.

Para escalar el sistema, tan solo requeriría añadir los elementos físicos, definir nuevos tópicos en MQTT y clonar las funciones de Node-Red. También requerirían la implantación de seguridad en el MQTT para que no se puedan alterar los datos enviados.

Con este sistema, las contribuciones a la Smart City son:

-Eficiencia energética: el alumbrado adaptativo reduciría el consumo energético y de mantenimiento en la red de iluminación de la ciudad.

-Reducción en la contaminación lumínica: la luz únicamente encendida cuando hay tránsito minimiza la contaminación lumínica respetando la biodiversidad y el bienestar humano.

-Planificación urbana: los datos obtenidos de los movimientos concreta una actividad peatonal en fecha, lugar y hora. Con esta información podemos evaluar el tipo de movilidad, posibles mejoras en las rutas urbanas menos transitadas, una modificación en los horarios del transporte público o la ampliación o reducción de transportes como el "Bicing"de Barcelona.

El sistema IoT realizado es uno de los cientos o incluso miles de subsistemas que tendrá una ciudad inteligente. Cada uno de ellos será fundamental para tener una infraestructura con la que sea viable el ideal de la Smart City. Su éxito dependerá de la colaboración transversal de técnicos y políticos en las fases de concepción, desarrollo, despliegue y monitorización de las ciudades del futuro.

AGRADECIMIENTOS

Quiero dar las gracias a todas las personas que me han acompañado en este camino. En especial, gracias a Raúl Aragonés, mi tutor, por su ayuda y seguimiento al realizar mi proyecto, resolviendo mis dudas y proponiendo mejoras a mi sistema.

También quiero agradecer a mi familia y compañeros por su apoyo, paciencia y ánimos, sobre todo cuando las cosas no salían como esperaba.

REFERENCIAS

[1] *Arduino IDE and ESP32 Libraries*. Instalación: https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json, ESP32 GitHub: https://github.com/espressif/arduino-esp32, Driver CH430 Windows 10: http://www.wch.cn/downloads/CH341SER_EXE.html, Librerías: WIFI.h (core ESP32), PubSubClient.h https://github.com/knolleary/pubsubclient, Wire.h https://www.arduino.cc/en/reference/wire, RtcDS1302.h https://github.com/Makuna/Rtc. 2025. URL: https://www.arduino.cc/en/software.

[2] *Artículos relacionados sobre alumbrado público inteligente*. IEEE: https://ieeexplore.ieee.org/document/8317037, CNN Innovation: https://edition.cnn.com/2013/07/18/tech/innovation/tvilight-street-lamps-roosegarde/, Dimonoff case study: https://www.dimonoff.com/resources/case-studies/city-smart-street-lighting-journey-columbus/?utm_source=chatgpt.com, Arxiv: https://arxiv.org/abs/2211.00074?utm_source=chatgpt.com, ResearchGate Sheffield: https://www.researchgate.net/publication/349587224_Smart_streetlights_in_Smart_City_a_case_study_of_Sheffield. 2025. URL: https://en.wikipedia.org/wiki/Intelligent_street_lighting?utm_source=chatgpt.com.

[3] Ayuntamiento de Terrassa. *Pliego técnico expediente 15846/2023 (mantenimiento y mejora del alumbrado público)*. 2023. URL: https://contractaciopublica.cat/ca/perfils-contractant/detall/9205334/documents#contractacio-programada.

[4] Espressif Systems. *Microcontrolador ESP32 (ESP32 Overview)*. 2025. URL: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/.

[5] *Factor de emisión CO_2/kWh en España (2024)*. 2024. URL: https://www.miteco.gob.es/content/dam/miteco/es/calidad-y-evaluacion-ambiental/temas/sistema-espanol-de-inventario-sei-/es-nir-edicion-2024.pdf.

[6] *Grafana Documentation*. Documentación: https://grafana.com/docs/grafana/latest/. 2025. URL: https://grafana.com/.

[7] *Horas de encendido anual (calle no regulada)*. 2025. URL: https://industria.gob.es/Calidad-Industrial/seguridadindustrial/instalacionesindustriales/baja-tension/Paginas/guia-tecnica-aplicacion.aspx.

[8] *InfluxDB versión 1.8*. Guía usuario: https://docs.influxdata.com/influxdb/v1.8/introduction/install/. 2025. URL: https://dl.influxdata.com/influxdb/releases/influxdb-1.8.10_windows_amd64.zip.

[9] LEDBOX. *Farola Villa LED 40 W (Ficha técnica Ref. LD1010054)*. 2025. URL: https://www.ledbox.es/.

[10] *Módulo DS1302 RTC Datasheet*. 2025. URL: https://datasheets.maximintegrated.com/en/ds/DS1302.pdf.

[11] *Módulo DS1302 RTC Imagen (Fig. 5)*. 2025. URL: https://www.botnroll.com/en/others/3050-ds1302-rtc-real-time-clock-module.html.

[12] *Node-Red Documentation*. Proyecto open source: https://github.com/node-red/node-red. 2025. URL: https://nodered.org/docs/.

[13] *Precio electricidad 2025 (tarifa ayuntamiento)*. 2025. URL: https://tradingeconomics.com/spain/electricity-price.

[14] *Protocolo MQTT y Mosquitto Broker*. 2025. URL: https://mqtt.org.

[15] *Sensor HC-SR04 Datasheet*. 2025. URL: https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf.

[16] *Sensor HC-SR04 Imagen (Fig. 4)*. 2025. URL: https://naylampmechatronics.com/sensores-proximidad/10-sensor-ultrasonido-hc-sr04.html.

[17] *Sensor PIR HC-SR501 Manual del Usuario*. 2025. URL: https://www.scribd.com/document/357270269/Manual-Del-Usuario-Sensor-de-Movimiento-Pir-Hc-Sr501.

[18] *Separación media y número de farolas por km en vía pública urbana (2015)*. 2015. URL: https://industria.gob.es/Calidad-Industrial/seguridadindustrial/instalacionesindustriales/baja-tension/Paginas/guia-tecnica-aplicacion.aspx.

APÉNDICE

A.1. Código

-Nodo A- -Nodo B-

A.2. Maqueta

La maqueta fabricada tiene el objetivo de representar la disposición real de mi sistema en una ciudad. Se muestra visual y claramente el funcionamiento del sistema IoT. La maqueta dispone todos los elementos previamente mencionados. Para el correcto funcionamiento de los sensores, la disposición es muy importante. Para el sensor HC-SR501, su posición tiene que ser en perpendicular a la carretera y en el sensor HC-SR04 tiene que ser de cara a la acera transitada.

Gracias a esta maqueta, he podido comprobar la eficacia de la solución planteada, validar el correcto funcionamiento del sistema completo, y ofrecer una demostración práctica

Fig. 14: Código Nodo A-1

Fig. 15: Código Nodo A-2

que complementa y facilita la explicación teórica desarrollada en este trabajo.

esp32bsem.ino

```
// ESP32-B: mido distancia con HC-SR04 y respondo pings MQTT

#include <WiFi.h>
#include <PubSubClient.h>
#include <time.h>

// — Mis datos de Wi-Fi y MQTT —
const char* SSID        = "tfgprueba";
const char* PASS        = "12345678910";
const char* MQTT_BROKER = "192.168.176.243";

// — Pines HC-SR04 —
#define TRIG_PIN  26
#define ECHO_PIN  25

// — Pines semáforo —
#define LED_RED     13
#define LED_YELLOW  12
#define LED_GREEN   27

// — Rangos en cm —
#define RED_MIN     5
#define RED_MAX     20
#define YELLOW_MAX  40

WiFiClient    netclient;
PubSubClient  mqtt(netclient);

// Devuelvo HH:MM:SS vía NTP
String getTime() {
  time_t now = time(nullptr);
  struct tm tm;
  localtime_r(&now, &tm);
  char buf[9];
  strftime(buf, sizeof(buf), "%H:%M:%S", &tm);
  return String(buf);
}

// Cuando llega un ping de A, mando ack
void mqttCallback(char* topic, byte* payload, unsigned int len) {
  String msg;
  for (unsigned i = 0; i < len; i++) msg += (char)payload[i];
  if (String(topic) == "esp32/comms/enviado") {
    mqtt.publish("esp32/comms/recibido", msg.c_str());
    Serial.println("Ping recibido, respondo");
  }
}

// Me suscribo y aseguro la conexión MQTT
void ensureMqtt() {
  if (mqtt.connected()) return;
  Serial.print("Conecto MQTT…");
  while (!mqtt.connected()) {
    if (mqtt.connect("ESP32_B")) {
      Serial.println("¡conectado!");
      mqtt.subscribe("esp32/comms/enviado");
    } else {
      Serial.print("fallo rc=");
      Serial.print(mqtt.state());
      Serial.println(", reintento 2s");
      delay(2000);
    }
  }
}
```

Fig. 16: Código Nodo B-1

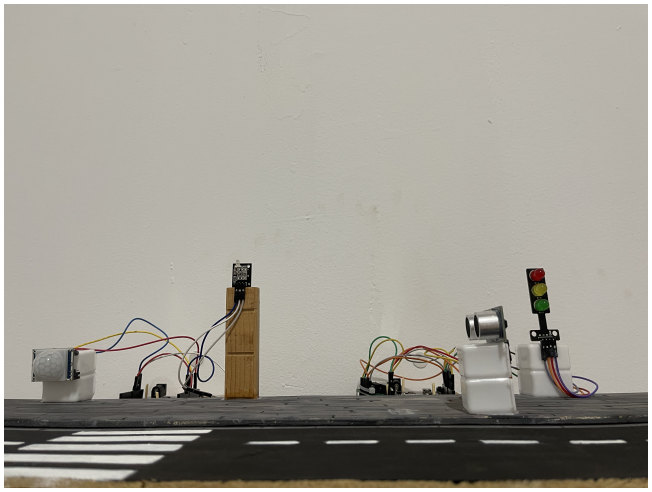

Fig. 18: FOTO MAQUETA 1

```

void setup() {
  Serial.begin(115200);
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  pinMode(LED_RED,    OUTPUT);
  pinMode(LED_YELLOW, OUTPUT);
  pinMode(LED_GREEN,  OUTPUT);

  WiFi.begin(SSID, PASS);
  Serial.print("Conecto Wi-Fi");
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(500);
  }
  Serial.println(" ¡Wi-Fi OK!");

  configTime(0, 0, "pool.ntp.org","time.nist.gov");
  while (!time(nullptr)) delay(100);
  Serial.println("Hora NTP OK");

  mqtt.setServer(MQTT_BROKER, 1883);
  mqtt.setCallback(mqttCallback);

  Serial.println("Setup OK\n");
}

void loop() {
  ensureMqtt();
  mqtt.loop();

  // Mido distancia
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);

  long dur = pulseIn(ECHO_PIN, HIGH, 20000UL);
  float distance = dur > 0 ? dur * 0.034f / 2.0f : -1.0f;

  String ts = getTime();
  if (distance >= 0) {
    Serial.printf("%s  %.1f cm\n", ts.c_str(), distance);
    char msg[64];
    snprintf(msg, sizeof(msg),
             "{\"distancia\":%.1f,\"tiempo\":\"%s\"}",
             distance, ts.c_str());
    mqtt.publish("esp32/distancia", msg);
  } else {
    Serial.printf("%s  -- cm\n", ts.c_str());
  }

  // Semáforo: rojo (cerca), amarillo (medio), verde (lejos)
  digitalWrite(LED_RED,    distance>=RED_MIN   && distance<RED_MAX);
  digitalWrite(LED_YELLOW, distance>=RED_MAX   && distance<YELLOW_MAX);
  digitalWrite(LED_GREEN,  distance>=YELLOW_MAX);

  delay(200);
}
```

Fig. 17: Código Nodo B-2

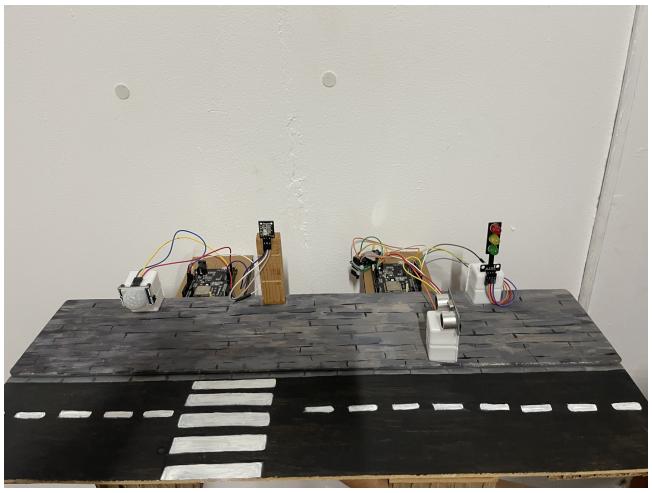

Fig. 19: FOTO MAQUETA 2