UNIVERSITAT POLITÈCNICA DE VALÈNCIA

_ TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

SISTEMA DE AUTOMATIZACIÓN Y CONTROL DE LUMINARIAS MEDIANTE TECNOLOGÍA LoRaWAN.

Ignacio García Plaza

Tutor: Víctor Miguel Sempere Paya

Trabajo Fin de Grado presentado en la Escuela Técnica Superior de Ingeniería de Telecomunicación de la Universitat Politècnica de València, para la obtención del Título de Graduado en Ingeniería de Tecnologías y Servicios de Telecomunicación

Curso 2020-21

Valencia, 11 de septiembre de 2021

Escuela Técnica Superior de Ingeniería de Telecomunicación
Universitat Politècnica de València
Edificio 4D. Camino de Vera, s/n, 46022 Valencia
Tel. +34 96 387 71 90, ext. 77190
www.etsit.upv.es

VLC/ CAMPUS VALENCIA, INTERNATIONAL CAMPUS OF EXCELLENCE

EMAS Gestión ambiental verificada REG.NO. ES-CV- 000030

Resumen

El proyecto trata de realizar un sistema capaz de controlar la iluminación de una serie de luminarias estableciendo las comunicaciones mediante el uso de tecnología LoRa.

El sistema está compuesto por una serie de autómatas. Funcionando como *'master'*, hay una Raspberry Pi 3 y funcionando como *'slave'* una Raspberry Pi Pico en cada una de las luminarias. La comunicación entre ellos es bidireccional. *Master* puede hacer dos tipos de envíos, forzar una determinada iluminación de la luminaria y consultar ese porcentaje de iluminación. Así pues, el *slave* recibe ese porcentaje de iluminación y la regula o bien responde enviando el estado de dicha luminaria si se le consulta.

Las comunicaciones a destacar del sistema son dos: el protocolo SSH, para conectarse al *master* desde el ordenador y tecnología LoRa para la comunicación final entre *master* y *slaves*.

Por último, por cada consulta, el estado de cada luminaria se almacena en una tabla SQL en la memoria interna del máster. Así pues, cada 3 minutos se suben dichos datos desde el máster a un servidor propio para la elaboración de gráficas o el estudio de mejoras.

Palabras clave

LoRa, LoRaWAN, comunicación inalámbrica, master, slave, módulo LoRa, Raspberry Pi 3, Raspberry Pi Pico, servidor, MySQL, luminaria, control, automatización.

Resum

El projecte tracta de realitzar un sistema capaç de controlar la il·luminació d'una sèrie de lluminàries establint les comunicacions mitjançant tecnología LoRa.

El sistema està compost per una sèrie d'autòmats. Funcionant com 'master', tinc una Raspberry Pi 3 i funcionant de 'slave', utilize una Raspberry Pi Pico en cada una de les lluminàries. La comunicació entre ells és bidireccional. 'Master' pot fer dos tipus d'envios, forçar una determinada il·lumnació de la lluminària i consultar eixe percentatge d'il·luminació. D'eixa manera el 'slave' rep eixe percentatge d'il·luminació i regula la lluminària o bé respon enviant l'estat de la dita lluminària si se li consulta.

Les comunicacions a destacar del sistema son dos: el protocol SSH, per a connectar-se al 'master' des de l'ordinador i tecnologia LoRa per a la comunicació final entre 'master' i 'slaves'.

Finalment, per cada consulta que es fa, l'estat de cada lluminària s'emmagatzema en una taula SQL en la memòria interna del 'master'. Aixina, cada XXX es pugen les dades des del 'master' a un servidor propi per a l'elaboració de gràfiques o l'estudi de millores.

Paraules clau

LoRa, LoRaWAN, comunicació inalàmbrica, màster, esclau, módul LoRa, Raspberry Pi 3, Raspberry Pi Pico, servidor, MySQL, lluminaria, control, automatització.

Abstract

The project consists of creating a system able to control the lighting of a group of luminaires establishing communications through the use of LoRa technology.

The system is made up of some automatas. Working as a master, I have a Raspberry Pi 3 and working as slaves, I use a Raspberry Pi Pico in each luminaire. Communication between them is bidirectional. Master can make two types of message, force a certain illumination of the luminaire and consult that percentage of illumination. So, the slave receives that percentage of illumination and regulates the luminaire or responds by sending the status of the luminaire if it is consulted.

There are two outstanding communications in the system: the SSH protocol, to connect from the computer to the master and LoRa technology for the final communication between master and slaves.

Finally, for each query made, the status of each fixture is stored in an SQL table in the internal memory of the master. In addition, every 3 minutes data is uploaded from the master to an own server to make graphs or improvements studies.

Keywords

LoRa, LoRaWAN, wireless communication, master, slave, LoRa module, Raspberry Pi 3, Raspberry Pi Pico, My SQL tables, lighting, control, automation.

AGRADECIMIENTOS.

Muchas gracias a familiares y amigos por haberme apoyado y aguantado a partes iguales durante todos estos años. Nunca hubiera sido posible sin vuestro esfuerzo por verme crecer como persona. Sé que podré contar con todos vosotros durante el resto de mi vida.

Índice

Introducción.

Este proyecto se ha realizado en la empresa PAVENER SERVICIOS ENERGÉTICOS SLU para fines profesionales. En esta empresa es donde he realizado las prácticas durante el último curso del grado. PAVENER es una empresa que se encarga de la gestión de grandes instalaciones consumidoras de energía, y sus servicios comprenden desde la monitorización, el control y la automatización de los sistemas y procesos, hasta la compra de la energía.

PAVENER persigue una solución para el desarrollo de un sistema de comunicación punto a punto fiable y económico. En un principio se planteó utilizar módulos de comunicaciones del fabricante Siemens, pero haciendo un estudio económico sobre su uso en una hipotética obra no resultaba eficiente utilizarlos.

Dicho plan casa con el objetivo por parte de PAVENER de construir instalaciones de alta eficiencia energética. De este modo, se plantea el uso de dicho sistema para controlar la iluminación de las luminarias pertenecientes a un cuadro eléctrico. Si podemos regular la iluminación disminuyéndola durante algunas horas en la madrugada, que es cuando menos afluencia de tráfico y peatones hay, podemos observar un ahorro de energía notable al cabo de un tiempo.

Mi tutor de prácticas en empresa y, ahora, compañero Fco. Javier Reig Montesinos comenzó a analizar el modo de realizar esta conexión mediante controladores como Arduino o Raspberry, más económicos que los módulos fabricados para ello. Estos dispositivos son versátiles, sencillos de utilizar y baratos.

Objetivos.

El objetivo principal a la hora de desarrollar el sistema de control y automatización de luminarias es ahorrar energía. Con el sistema se tendría que conseguir automatizar la regulación de las luminarias desde las 12 horas de la noche hasta las 6:30 horas de la mañana. Podemos conseguir una disminución del consumo energético público si durante esas horas las luminarias pasan de estar al 100% de su capacidad lumínica al 40%.

El objetivo secundario del proyecto es realizar dicho sistema de la manera más económica posible. Nos hemos fijado como objetivo no superar un presupuesto de 100 € para el desarrollo del sistema (valorando solamente la compra de dispositivos para la comunicación y regulación, dado que las luminarias para realizar las pruebas pertinentes las ha proporcionado PAVENER). Si se utilizaran módulos LoRa fabricados para ese fin, se superarían los 300 euros.

El objetivo final es que PAVENER pueda industrializar dicho sistema para obras relacionadas con instalaciones eléctricas.

Capítulo 1. Estado actual de LoRa.

Hoy en día tanto desde instituciones públicas como desde empresas privadas se fomenta la creación y desarrollo del Internet of Things (IoT). La creación de espacios inteligentes como Smart Cities, Smart Campus, Smart Buildings y Smart houses o el análisis del tráfico de una ciudad mediante múltiples sensores son un ejemplo. Lo que hace unos años eran proyectos piloto, en la actualidad son sistemas reales que funcionan y que tienen múltiples aplicaciones.

1.1. Definiciones de LoRa y LoRaWAN.

LoRa y LoRaWAN son dos cosas diferentes. LoRa (Long Range) es el tipo de modulación en radiofrecuencia, patentado por Semtech, que se utiliza en las comunicaciones de las redes LPWAN (Low Power Wide Area Network). En cambio, LoRaWAN (Long Range Wide Area Network) es un protocolo de red que utiliza la tecnología LoRa para comunicar y administrar varios dispositivos. LoRa tiene las siguientes características:

- Sensibilidad elevada, en torno a -168 dB.
- Tiene un alcance teórico de entre 8 y 15 kilómetros (LOS).
- Conexión punto a punto.
- Baja transferencia de datos (hasta 255 bytes).
- Funciona a unas frecuencias de trabajo determinadas dependiendo de la legislación de cada país. En España puede trabajar a 433 MHz o a 868 MHz.

Estas características hacen que LoRa sea una muy buena solución para conexiones con poca carga de datos y entre varios dispositivos, como las redes IoT.

Las LPWAN son redes con topología estrella. Estas redes se componen de dispositivos finales (sensores) conectados a *gateways* o puertas de enlace, desde los cuales se envían todos los datos a un servidor para su posterior tratamiento. Según los niveles OSI, en las LPWAN, LoRaWAN es nivel 2 (red) y LoRa el nivel 1 (físico). En la siguiente figura podemos observar un diagrama que representa la arquitectura del protocolo de comunicación.

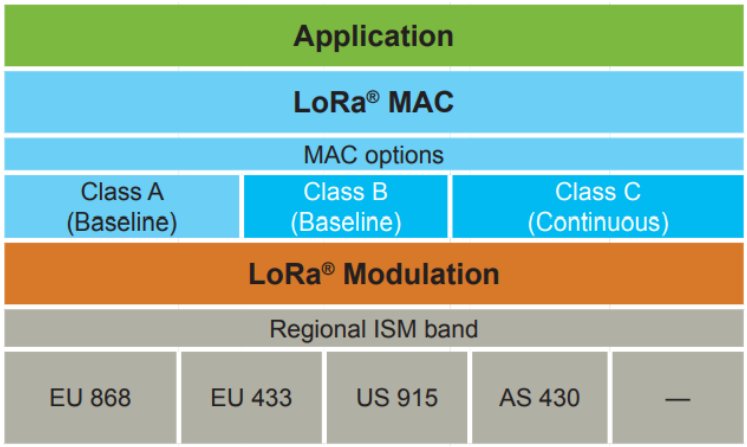

Figura 1. Esquema de la arquitectura del protocolo de comunicación.[1]

1.2. Tipos de comunicación.

Existen dos modos de comunicación. El modo LoRa, que es la conexión punto a punto entre los diferentes sensores o nodos y el modo LoRaWAN, donde los nodos se comunican con el gateway.

La principal característica del modo LoRa es que no es necesario que haya un dispositivo intermediario gestionando la comunicación entre nodos. Este tipo de comunicación es ideal para aplicaciones sencillas donde se quiere controlar el estado de un dispositivo en concreto, por ejemplo, una puerta de un garaje o una lámpara. Otra manera de comunicarse en este modo es la denominada *Mesh*. De esta manera existe un dispositivo que actúa de *master* y es el que se encarga de comunicarse con los otros nodos o *slaves*.

En el modo LoRaWAN es necesario que los nodos estén conectados a un gateway. Este gateway admite como máximo 62.500 nodos (límite teórico asumiendo una baja tasa, de 1 paquete/día por nodo y dependiendo de la configuración del gateway). Cada nodo tiene que enviar una serie de identificadores y claves de seguridad para conectarse a la red. Los nodos pueden estar en movimiento, y cuando esto ocurre, deben conectarse al gateway con mejor calidad y más próximo. En estos casos, se dice que la topología puede llamarse “estrella de estrellas”, pues son distintos gateway los que se conectan con un solo servidor.

1.3. Clases de nodos.

Existen tres clases de nodos:

- Clase A. Esta clase de nodos son los que están alimentados con baterías ya que tienen un consumo energético muy bajo. Esta clase de nodos, después de haber comunicado con el gateway, habilitan el modo escucha.
- Clase B. Los nodos pueden utilizar una batería o una fuente de alimentación externa. Dependerá de la cantidad de tiempo que permanece en modo escucha, determinado por el gateway, si es necesario colocarle una batería al sensor o directamente alimentarlo de forma externa.
- Clase C. Estos nodos deben ir conectados a una fuente de alimentación externa debido a su consumo, pues permanecen la mayor parte del tiempo en modo escucha y solo transmiten cuando es necesario.

1.4. Tipos de gateway.

El gateway o puerta de enlace en español se ocupa de transmitir y recibir la información a un servidor. En las redes LPWAN, los nodos se conectan al gateway. Tiene un canal de transmisión. Su sistema operativo suele estar basado en Linux. Otra de sus características es que puede tener un canal en recepción (*‘Single Channel’*) o varios (*‘Multichannel’*). Los gateway multicanal más comunes suelen ser los que tienen 8 canales en recepción de forma simultánea. Como se menciona en la sección 1.2 el número máximo de nodos soportados depende de la cantidad de paquetes que se transmitan por la red.

1.5 Características del proyecto.

Después de haber definido lo que es una LPWAN y los diferentes dispositivos que la componen, se puede decir que para este proyecto se va a utilizar el tipo de comunicación LoRa 'mesh'.

Como gateway actúa una Raspberry Pi 3. Este dispositivo se encarga de enviar los datos de regulación a los distintos sensores. Éste también se encarga de recibir los datos solicitados de casa sensor y de almacenarlos en un servidor interno y externo.

Los sensores que se utilizan en el proyecto son de tipo B. Utilizamos como sensores Raspberry Pi Pico, los cuales se alimentan con una toma de 3 voltios, con batería o con alimentación externa.

Capítulo 2. Desarrollo del sistema de control de luminarias.

En este capítulo se explica cómo está distribuido el sistema físicamente, qué dispositivos lo componen y por qué, qué protocolos de comunicación se utilizan para las distintas conexiones y que tipo de conexiones existen entre los dispositivos.

En primer lugar, en el siguiente diagrama se muestra la distribución del sistema que hemos estado estudiando.

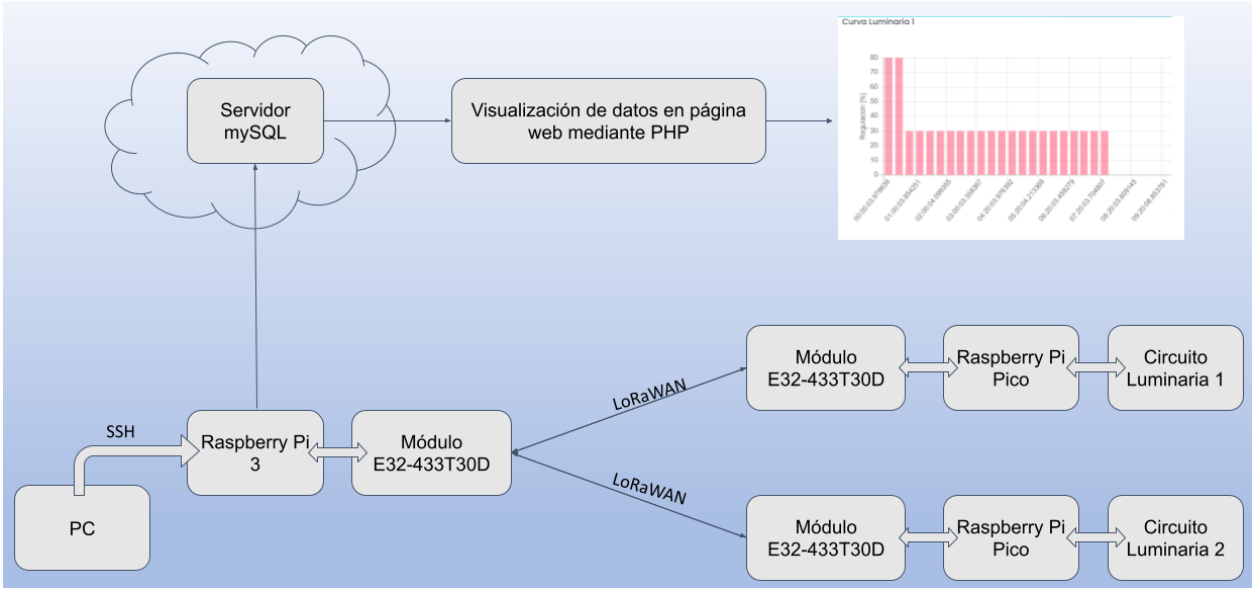

Figura 2. Diagrama de la distribución de los dispositivos y conexiones del sistema.

2.1. Distribución de los elementos.

Como se puede ver en la figura 2 el sistema se compone principalmente de tres bloques:

- En la parte inferior izquierda del diagrama, el máster. Es desde donde se va a controlar a los distintos sensores y desde donde se van a almacenar todos los datos recopilados por éstos en un servidor MySQL (Raspberry Pi 3 con el módulo de comunicaciones LoRa).

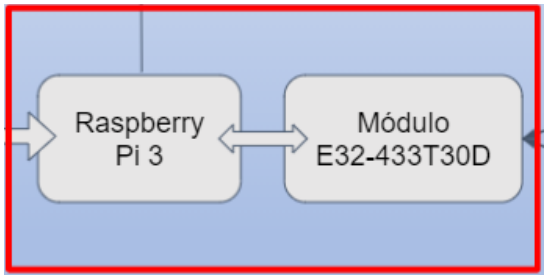

Figura 3. Recorte de la parte que controla el sistema, llamada máster.

- En la parte inferior derecha del diagrama podemos ver los sensores y sus respectivos circuitos en cada luminaria. Se encargan de regular la iluminación de su luminaria y de

enviar el estado de ésta cuando el máster se lo solicite (Raspberry Pi Pico con el módulo de comunicaciones LoRa).

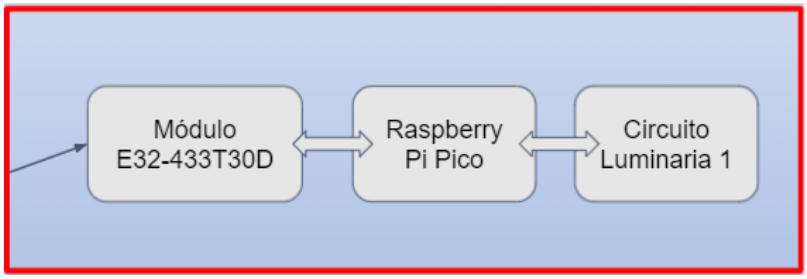

Figura 4. Recorte de la parte de los sensores (este pertenece al sensor de la luminaria 1, pero son todos idénticos).

➢ El servidor MySQL de la empresa PAVENER. Es donde vamos a almacenar todos los estados de las luminarias para realizar el posterior análisis y tratamiento de datos.

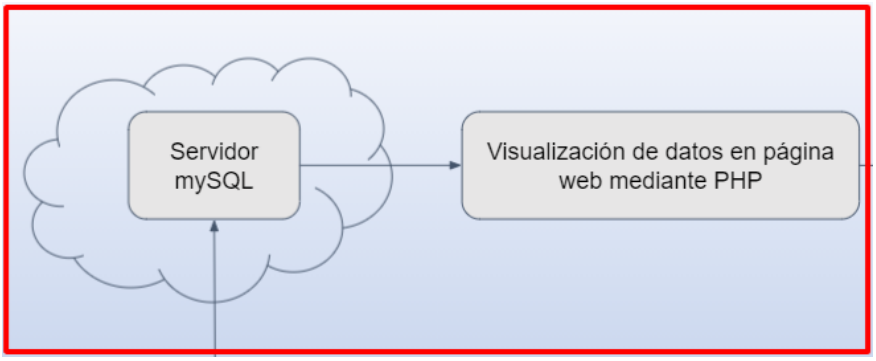

Figura 5. Recorte de la parte del servidor donde se almacenan todos los datos de las luminarias.

La idea inicial es que, en una instalación de varias luminarias a lo largo de una calle, el máster esté ubicado en el cuadro eléctrico junto a los interruptores automáticos, relés y demás componentes del mismo.

En cambio, los sensores van ubicados en la misma luminaria. En un principio queremos que la Raspberry Pi Pico, el circuito regulador de iluminación y el módulo de comunicaciones LoRa queden en cada luminaria.

En la práctica, cada cuadro eléctrico con sus respectivas luminarias es un mundo. El número de luminarias que controla o la distancia entre el cuadro y la luminaria más alejada dependen de la calle, avenida o parque donde estén. De momento, hasta que comprobemos el perfecto funcionamiento del sistema en las oficinas de la empresa, el máster dista de los sensores aproximadamente 12 metros.

A continuación, en el siguiente punto se describen cada uno de los dispositivos que se utilizan en el proyecto.

2.2. Dispositivos y tecnologías.

Para la construcción del proyecto hemos necesitado una Raspberry Pi 3, dos Raspberry Pi Pico, tres módulos transceptores de radiofrecuencia de E-byte E32-433T30D con tres antenas tipo monopolo, el servidor de PAVENER, dos placas de pruebas (las placas típicas que se usan en el laboratorio de electrónica) y una serie de componentes electrónicos que se describirán más adelante.

2.2.1. Raspberry Pi 3.

Una Raspberry Pi 3 es un ordenador de bajo coste y tamaño reducido. Es conocida por ser muy utilizada para desarrollar prototipos y para la formación en la informática. El sistema operativo por defecto de la Raspberry Pi 3 es Raspbian, basado en Linux. Sus especificaciones son las siguientes:

- CPU de cuatro núcleos a 1,2 GHz Broadcom BCM2837 de 64 bits.
- 1 GB de RAM.
- LAN inalámbrica y Bluetooth de baja energía (BLE) a bordo.
- Ethernet.
- GPIO extendido de 40 pines.
- 4 puertos USB tipo 2.
- Salida estéreo de 4 polos y puerto de video compuesto.
- HDMI de tamaño completo.
- Puerto de cámara CSI para conectar una cámara Raspberry Pi
- Puerto de pantalla DSI para conectar una pantalla táctil Raspberry Pi.
- Puerto microSD para cargar su sistema operativo y almacenar datos.
- Fuente de alimentación micro USB conmutada de hasta 2,5 A.

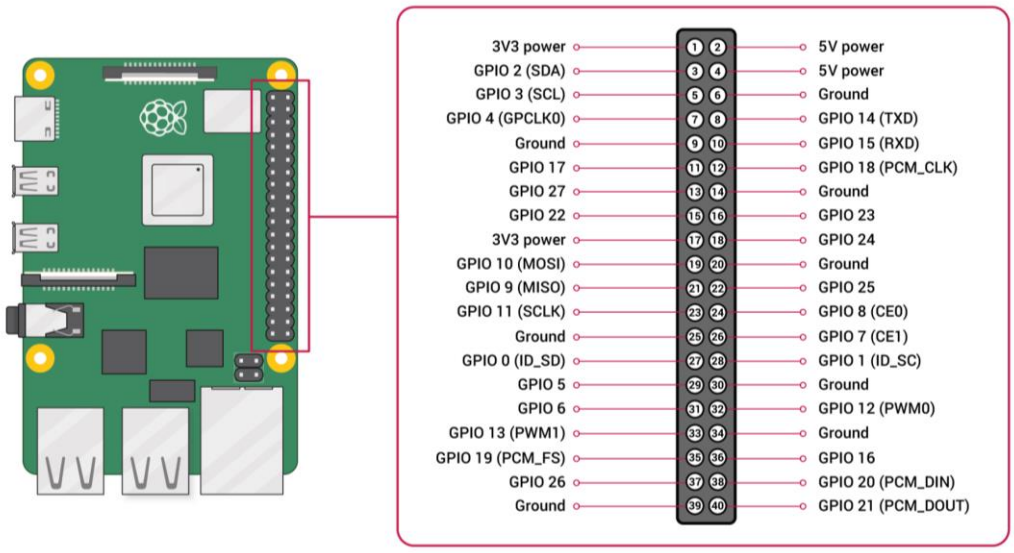

Figura 6. Diagrama de los pines de la Raspberry Pi 3. [4]

Las entradas que utilizamos en este proyecto son la entrada micro USB para alimentar la placa, los pines 4 (+5V) y 6 (GND) para alimentar el módulo transceptor LoRa y los pines 8

(UART-TXD) y 10 (UART-RXD) para la transmisión de los datos desde la placa al módulo transceptor.

2.2.2. Raspberry Pi Pico.

La Raspberry Pi Pico es una placa de microcontrolador de alto rendimiento y bajo coste con interfaces digitales flexibles. Las características principales son estas:

- Chip microcontrolador RP2040 diseñado por Raspberry Pi.
- Procesador Arm Cortex M0 + de doble núcleo, reloj flexible que funciona hasta 133 MHz.
- 264 KB de SRAM y 2 MB de memoria flash incorporada.
- El módulo almenado permite soldar directamente a las placas de soporte.
- USB 1.1 con soporte para dispositivo y host.
- Modos de reposo y de reposo de bajo consumo.
- Programación de arrastrar y soltar usando almacenamiento masivo a través de USB.
- 26 × pines GPIO multifunción (2 × SPI, 2 × I2C, 2 × UART, 3 × ADC de 12 bits, 16 × canales PWM controlables).
- Sensor de temperatura.
- Bibliotecas de punto flotante aceleradas en chip.
- 8 × máquinas de estado de E/S programables (PIO) para soporte periférico personalizado.

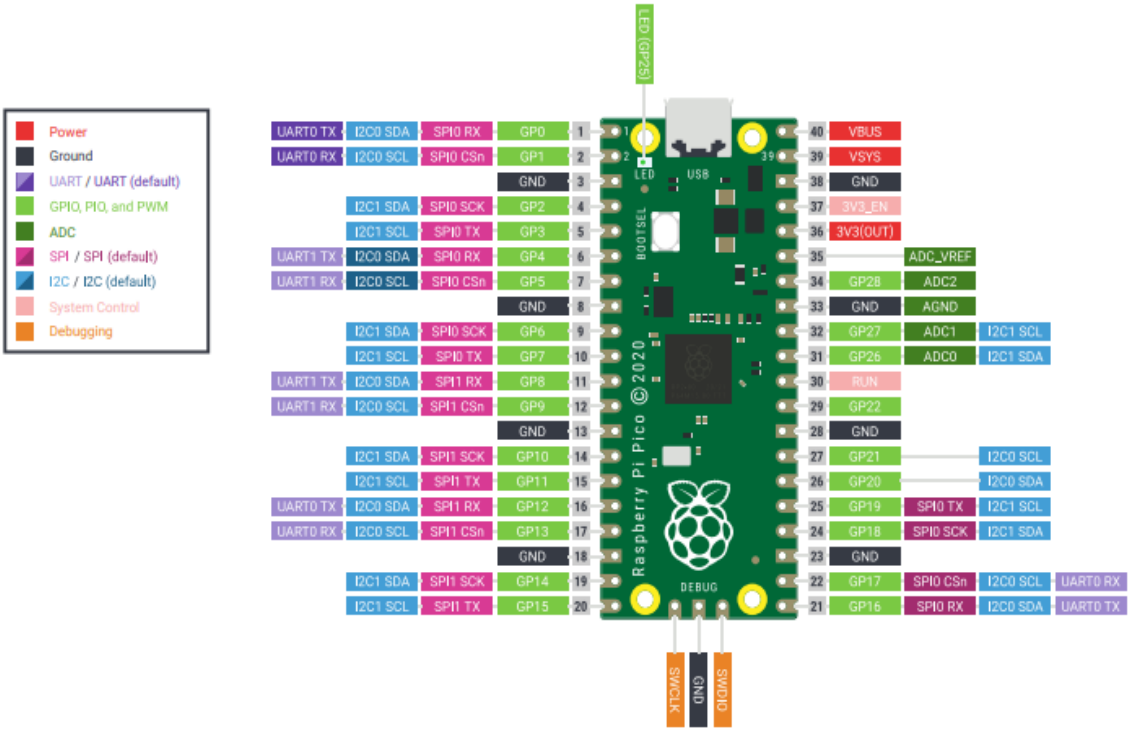

Figura 7. Diagrama de los pines de la Raspberry Pi Pico. [5]

Los pines que utilizamos en este proyecto de la Raspberry Pi Pico son el 39 (+3.3V) y el 38 (GND) para la alimentación de la placa, el 1 (UART0-TX) y 2 (UART0-RX) para recibir y transmitir los datos al módulo transceptor Lora, el pin 21 (PWM) va conectado a la entrada de 'regulación' del circuito impreso y el 22 (PWM) va conectado a la entrada 'relé' del circuito impreso.

2.2.3. Módulo transceptor E32-433T30D de E byte.

E32-433T30D es un módulo con un puerto serie inalámbrico (UART) basado en el chip de radiofrecuencia SX1278 de SEMTECH. Tiene varios modos de transmisión y su frecuencia de trabajo puede ser entre los 410 MHz hasta los 441 MHz, aunque por defecto su frecuencia de trabajo es 433 MHz, canal 32 (espectro de LoRa). Este módulo tiene las siguientes características técnicas:

- Puede ser alimentado desde 2.8V a 5.5V, aunque el fabricante asegura un mejor funcionamiento si la alimentación supera los 5V.
- Distancia máxima de comunicación probada superior a 8 kilómetros.
- Potencia de transmisión máxima de 1W (entre 21 y 30 dBm).
- Sensibilidad de -147 dBm.
- Tasa en baudios entre 1200 y 115200, por defecto 9600 baudios.
- Temperatura de funcionamiento entre -40ºC y 85ºC.

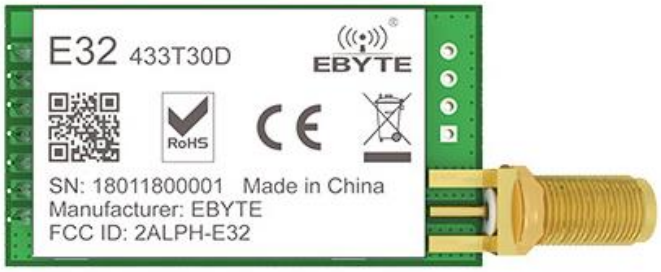

Figura 8. Imagen del módulo E32-433T30D.

La figura 8 representa solo el módulo transceptor, pero éste, así como sale en la imagen no se puede conectar con ninguna de las Raspberry que estamos utilizando. Para poder transmitir y recibir datos a través de él hay que implementar un circuito como el de la figura 9. En este caso compramos conjuntamente tanto el módulo E32 como el circuito integrado.

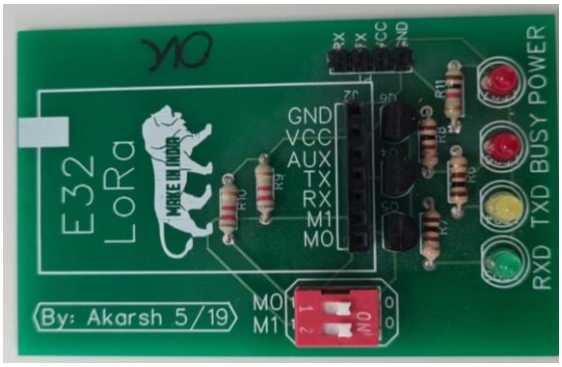

Figura 9. Imagen del circuito integrado auxiliar para el módulo E32.

Aquí vemos que los pines GND, VCC, TX y RX del módulo E32 quedan como entradas y salidas de la misma placa. El pin AUX sirve para indicar si el buffer del módulo está vacío o lleno. Como podemos observar en la figura 9, en la parte inferior hay dos switches, M0 y M1. Estos sirven para seleccionar el modo de funcionamiento del módulo.

En la siguiente tabla se pueden ver cómo configurar los diferentes modos:

Mode（0-3）	M0	M1	Mode introduction
0 Normal	0	0	UART and wireless channel are open, transparent transmission is on
1 Wake up	1	0	UART and wireless channel are open, the only difference with mode 0 is that before transmitting data, increasing the wake up code automatically, so that it can awake the receiver under mode 3.
2 Power saving	0	1	UART close, wireless is under air-awaken mode, after receiving data, UART open and send data.
3 Sleep	1	1	sleep mode, receiving parameter setting command is available.

Tabla 1. Configuración de los pines M0 y M1 para las distintas funciones del módulo E32. [6]

Nuestra configuración de estos pines es M0=0 y M1=0.

Finalmente, así es como queda cuando acoplas el módulo E32 al circuito integrado (figura 10).

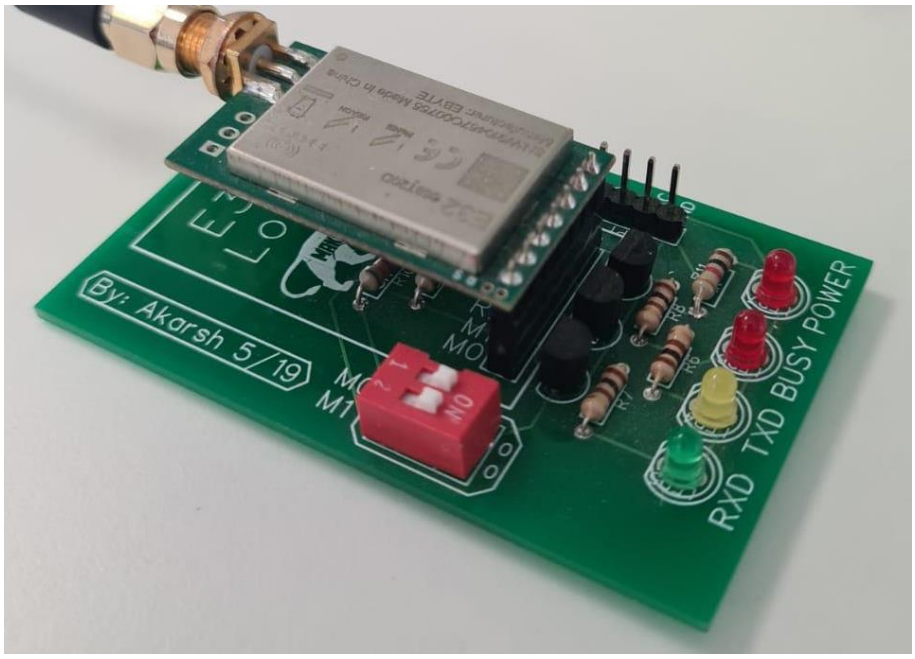

Figura 10. Fotografía de la configuración final del módulo de comunicaciones LoRa del proyecto.

Las antenas que usamos son unas antenas tipo monopolo con las siguientes especificaciones de fábrica:

- Impedancia de 50 ohmios.

- Ganancia de -9.5dBi.
- Longitud física de 142 milímetros.
- Conector RP-SMA-J.
- Eficiencia máxima del 12%.

2.2.4. Circuito regulador de iluminación.

En un principio todos los circuitos que se exponen a continuación, pequeños circuitos que componen el circuito regulador de iluminación de las luminarias, se implementaron en placas de prueba como las que se pueden utilizar en un laboratorio de electrónica. Su manipulación resulta fácil al igual que poder implementar en ellas el circuito que desees. Si cuando estuviera todo bien montado y funcionando correctamente no tuviera ningún problema, realizaremos una placa pcb con todo el circuito impreso en ella.

Los diseños de los circuitos me los proporcionó la empresa PAVENER, que los realizó con el software Eagle. En la siguiente imagen se pueden observar los circuitos que componen el circuito regulador de iluminación.

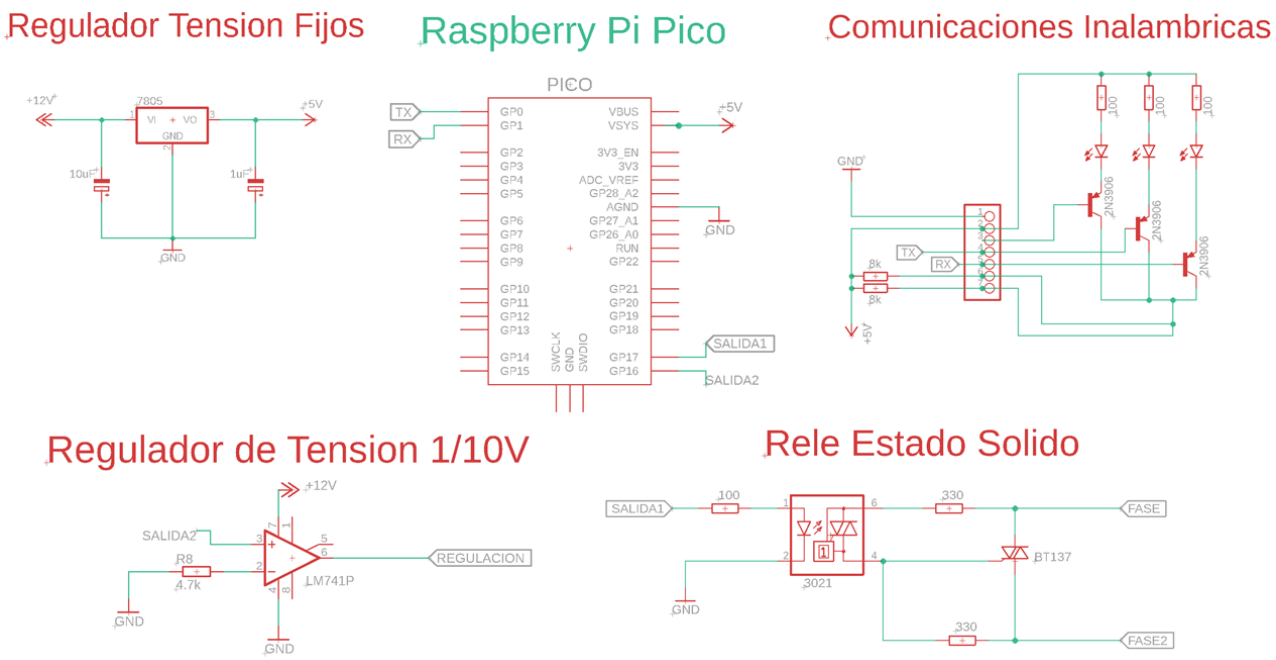

Figura 11. Esquema general de los circuitos en software Eagle.

Como se menciona anteriormente en cada luminaria se coloca un circuito regulador de iluminación. Dicho circuito está compuesto principalmente de 5 partes: un regulador de tensión, un relé, un regulador de tensión fijo, el módulo E32 y la Raspberry Pi Pico. A continuación, se describen los componentes y función de cada uno de los circuitos.

2.2.4.1. Regulador de tensión entre 1 y 10 voltios.

El circuito 'Regulador de tensión 1/10v' está formado por un amplificador operacional LM741P alimentado con 12 voltios por el pin 7 y conectado a tierra por el pin 4. En el pin número 3 se introduce la señal 'PWM' saliente de la Raspberry Pi Pico, entre 0 y 3 voltios. De esa forma, en el pin número 6 se manda la señal, entre 1 y 10 voltios, que regula la iluminación de la luminaria.

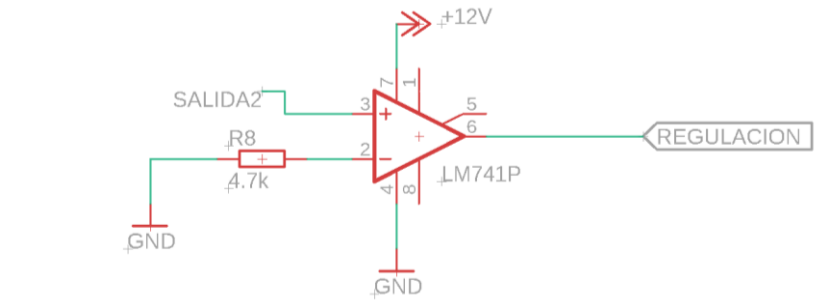

Figura 12. Esquema del circuito regulador de tensión en software Eagle.

2.2.4.2. Regulador de tensión fijo.

El circuito 'Regulador de tensión fijo' es muy simple. Se compone de un regulador de voltaje 7805 de 5 voltios. A la entrada de este tenemos 12 voltios, y a su salida 5 voltios. Hay que colocar los condensadores que se indican en el esquema de esa manera para conseguirlo.

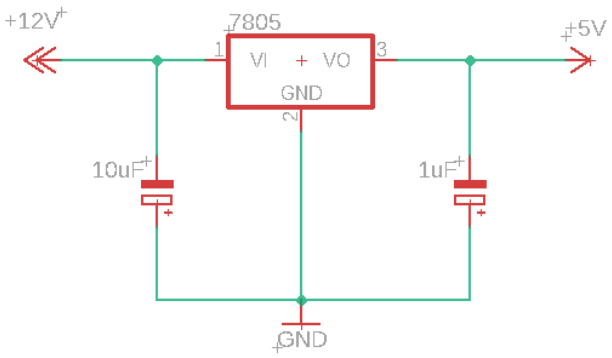

Figura 13. Esquema del circuito regulador de tensión de 12 a 5 voltios en software Eagle.

2.2.4.3. Relé estado sólido.

En el circuito de 'Relé estado sólido' la entrada 1 al optoacoplador es una señal digital que sale de la Raspberry Pi Pico. Mediante el uso de un triac BT137 conseguimos poder encender o apagar la luminaria. 'FASE' y 'FASE2' son las dos fases de la toma de corriente trifásica, de esta manera podemos encender o apagar la luminaria, no solo regular su iluminación.

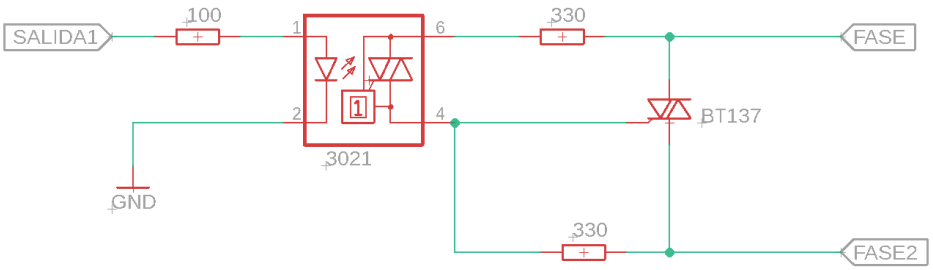

Figura 14. Esquema del relé estado sólido en software Eagle.

2.2.4.4. Comunicaciones inalámbricas.

Como ya se introdujo en el apartado 2.2.3 del proyecto, este circuito es el encargado de unir la Raspberry Pi Pico con el módulo ya fabricado de LoRa. Los pines que se utilizan del autómata son el 1 (como transmisor) y el 2 (como receptor). Este circuito puede alimentarse con una tensión de entre 3.3 voltios y 5.5 voltios. En este proyecto lo alimentamos con la misma fuente externa con la que se alimenta la Raspberry Pi Pico, una fuente de 5 voltios.

La siguiente imagen, la figura 16, muestra el esquemático del circuito de la placa pcb de la figura 9.

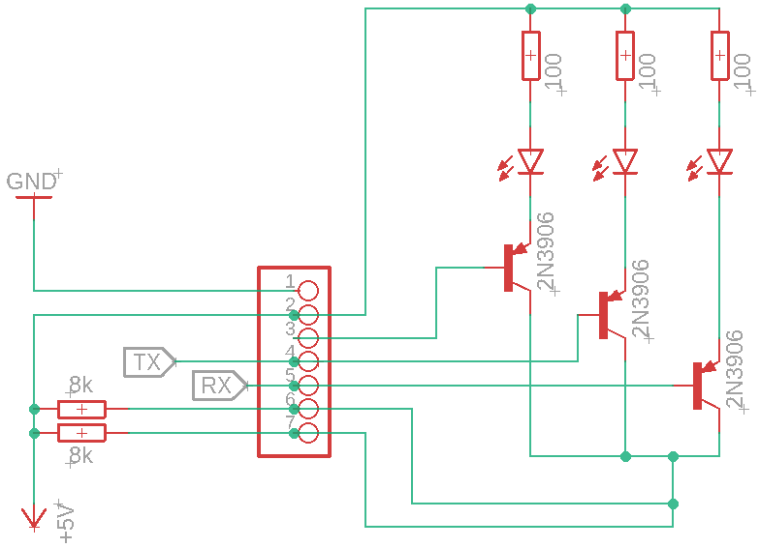

Figura 15. Esquema del circuito de comunicaciones LoRa en software Eagle.

2.2.4.5. Raspberry Pi Pico.

Observamos los diferentes pines que utilizamos del autómata. Está alimentado con 5 voltios. Podemos observar las diferentes entradas y salidas que tiene con esta configuración. La salida llamada 'SALIDA 1' es la entrada en el circuito del relé y la salida 'SALIDA 2' es la entrada al circuito de regulación de la tensión entre 1 voltio y 10 voltios.

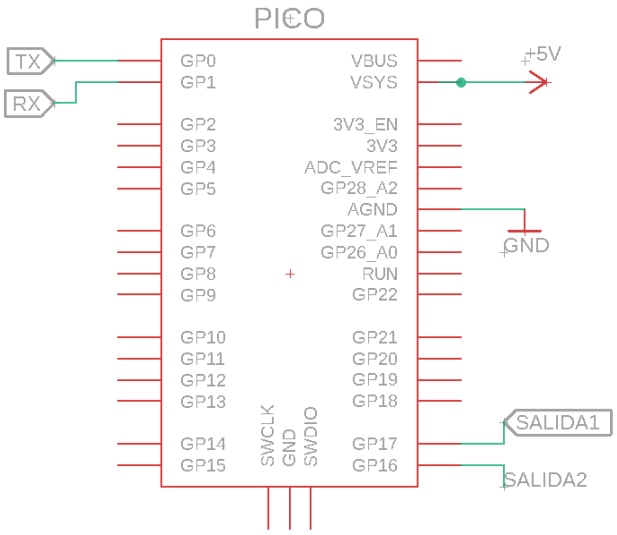

Figura 16. Esquema del autómata Raspberry Pi Pico en software Eagle.

La realización de todos estos esquemáticos con este software fue necesaria para que más tarde pudiéramos crear la placa pcb con todos los circuitos integrados en ella. El hecho de hacer todo en una misma placa pcb es por ser algo mucho más práctico que con las placas de pruebas.

2.2.4.6. Diseño de la placa pcb.

Así es como quedaría la placa pcb del circuito regulador de iluminación, compuesta por los anteriores circuitos descritos, la Raspberry Pi Pico y el módulo E32.

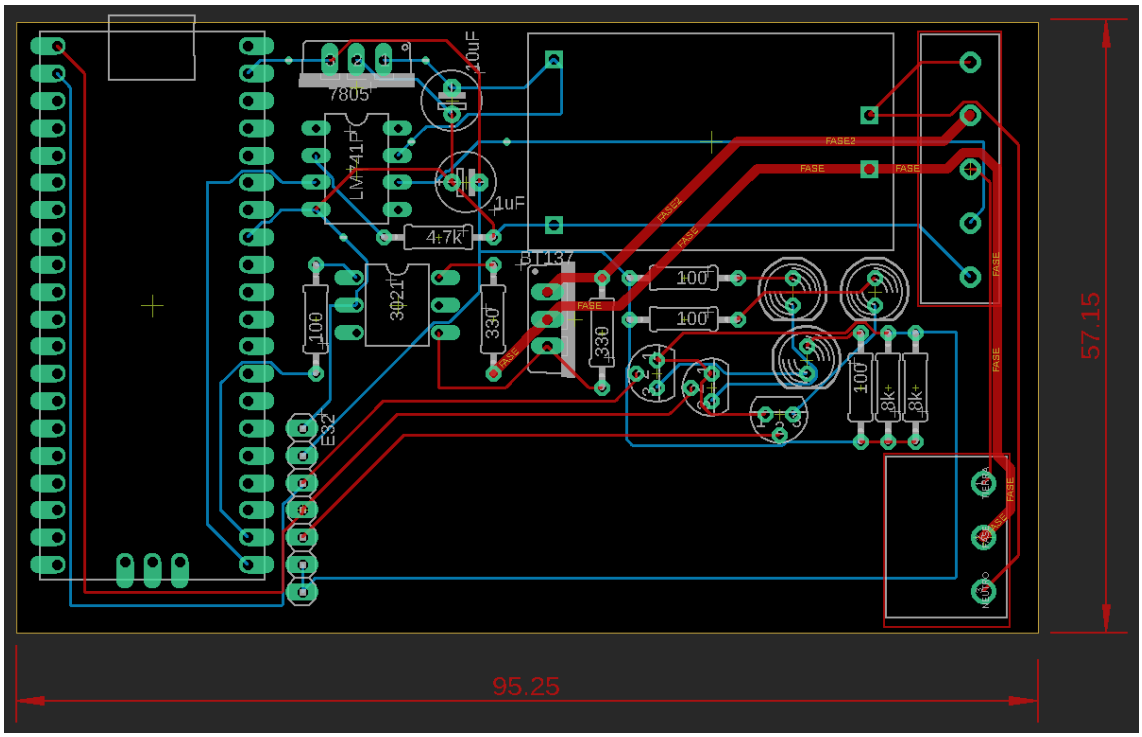

Figura 17. Diseño de la placa pcb del circuito integrado en la luminaria.

Ésta tendría como entradas la alimentación de 5 voltios, una toma de tierra, las dos fases y el neutro de la toma de corriente trifásica. Las cotas de la longitud y el ancho de la placa están en milímetros.

2.2.5. Luminarias Philips ClearWay gen2.

Es una luminaria de alumbrado público que la empresa PAVENER tenía en un almacén junto a otras. Para el desarrollo de este proyecto hemos usado dos luminarias como esta. Tiene las siguientes especificaciones técnicas:

- Usa un módulo led de 6900 lúmenes.
- Tensión de entrada entre 220 V y 240 V.
- Frecuencia de entrada de 50 Hz a 60 Hz.
- Corriente de arranque de 47 A.
- Iluminación regulable.
- Potencia de entrada inicial 40.5 W.

Figura 18. Aspecto de luminaria led de Philips de 40 W de potencia.

Una vez descritos todos los componentes del sistema y sus características es necesario explicar cómo funciona el sistema y qué comunicaciones existen entre las diferentes partes.

2.3. Funcionamiento del sistema.

El sistema tiene dos funciones: regular la iluminación de la luminaria y conocer el estado en el que ésta se encuentra. A continuación, se explica estas dos funciones mediante el uso de dos diagramas de flujo.

2.3.1. Regulación de la luminaria.

Para regular la luminaria se sigue el proceso descrito en este diagrama de flujo.

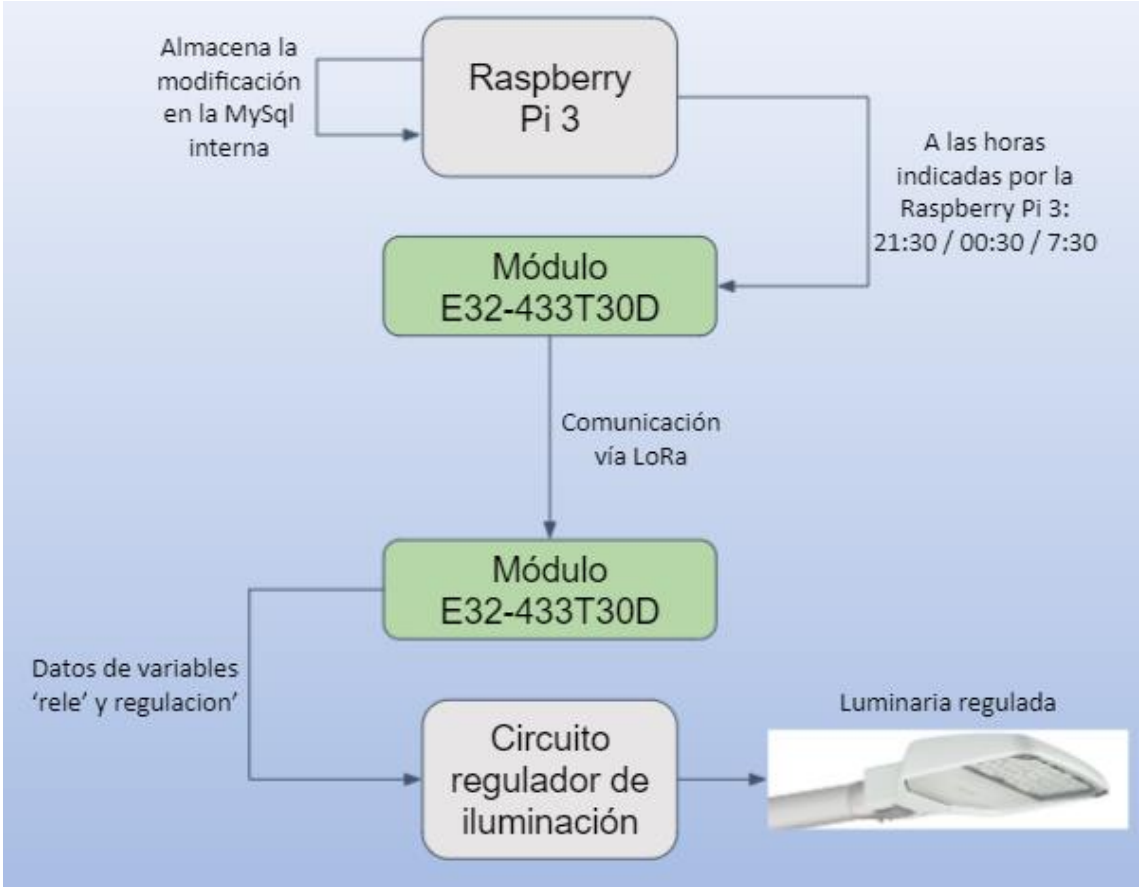

Figura 19. Diagrama de flujo que describe el funcionamiento de la regulación de la luminaria.

Siguiendo el diagrama, todo comienza en la Raspberry Pi 3. En ella se encuentra una serie de scripts, que se describirán más adelante, en los cuales se puede configurar que regulen un porcentaje de iluminación cualquiera de las luminarias a una hora exacta.

Por ejemplo, en horario de verano, queremos que las luminarias se enciendan automáticamente a las 21:30 horas. Cuando la Raspberry Pi 3 lea esa hora de su sistema, asignará un valor de '1' a la variable 'rele' y un valor '100' a la variable 'regulacion' y enviará dichos datos vía LoRa a las luminarias que se le especifique (en este caso al ser un encendido general, a todas) y estos datos quedan almacenados en la Raspberry Pi Pico. Dichos valores, así como la fecha y la luminaria a la que se le envían, queda registrado en la base de datos MySql interna de la Raspberry Pi 3.

Una vez la Raspberry Pi Pico recibe dichos datos, a través del circuito regulador de iluminación, se configura el encendido de la luminaria.

2.3.1. Consulta del estado de la luminaria.

Para describir la segunda función que tiene el sistema, conocer el estado de cada luminaria, podemos utilizar el siguiente diagrama de flujo.

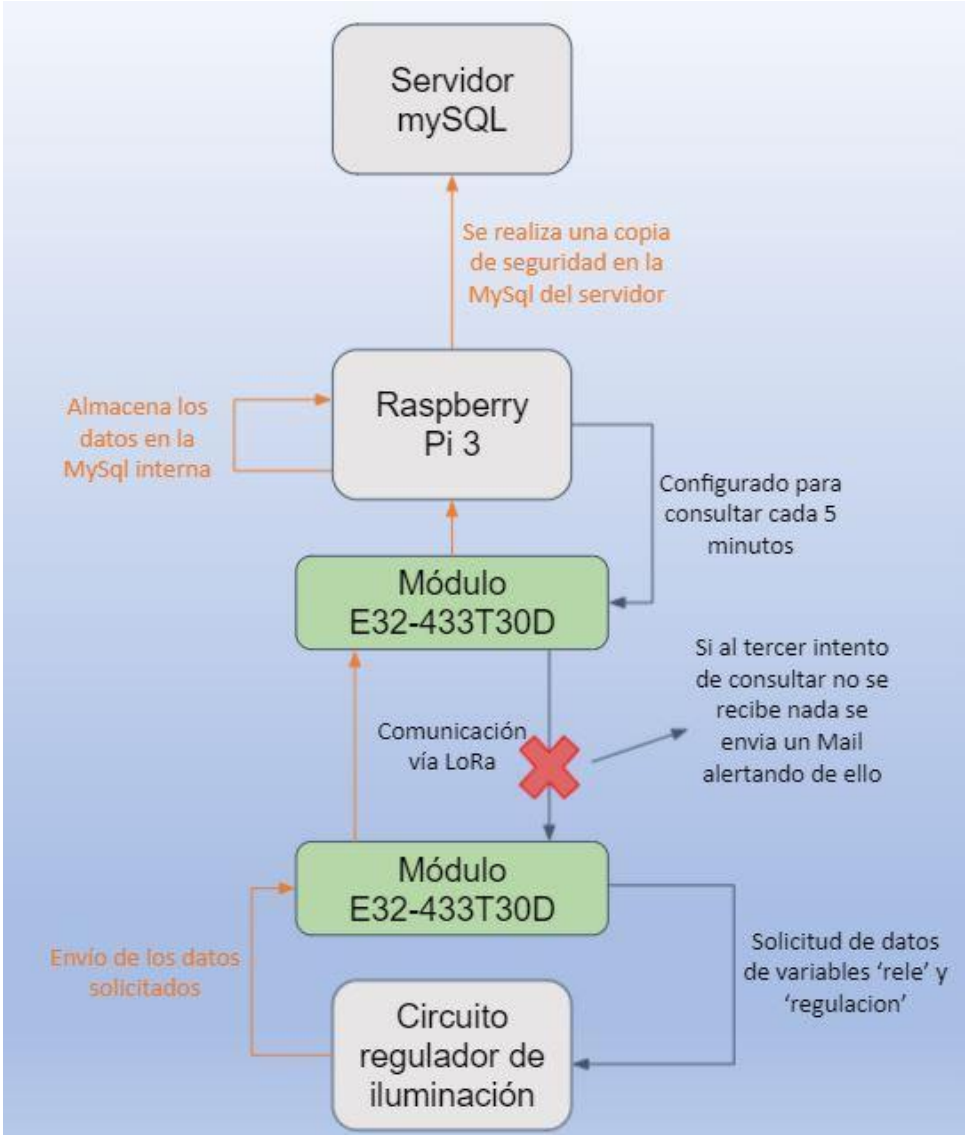

Figura 20. Diagrama de flujo que describe el funcionamiento de la consulta a los sensores para conocer el estado de las luminarias.

El proceso de la consulta del estado de las luminarias comienza en la Raspberry Pi 3. Ésta está configurada de tal forma que cada 5 minutos emite, a través del E32, un mensaje de consulta. Dicho mensaje se envía a todas las luminarias. Una vez la Raspberry Pi Pico situada en cada luminaria lee el mensaje, observa qué valor almacenado tienen las variables 'rele' y 'regulacion' y los envía a la Raspberry Pi 3. Una vez ésta lee dichos datos, almacena la fecha, el número de luminaria y los datos de 'rele' y 'regulacion' en su base de datos MySQL interna.

Cuando ya se han almacenado los datos de todas las luminarias se realiza el vuelco de datos de la MySQL interna de la Raspberry Pi 3 a la base de datos MySQL del servidor de PAVENER.

Cabe la posibilidad de que al enviar el mensaje de consulta no se comuniquen entre los módulos E32 o que se haya perdido la comunicación. En ese caso, cuando no hay respuesta por parte de alguna de las luminarias en tres ocasiones seguidas, se envía un correo electrónico alertando de qué luminaria ha perdido la comunicación y a qué hora.

Estas son las dos funciones que tiene el sistema de regulación de luminarias grosso modo. En el siguiente capítulo se entrará en detalle de cómo funcionan cada uno de los scripts que se utilizan.

Capítulo 3. Programación de los autómatas.

Como se menciona anteriormente se utilizan dos tipos de autómatas: una Raspberry Pi 3 y una Raspberry Pi Pico. La Raspberry Pi 3 es la encargada de hacer lo siguiente:

- Enviar el estado de la luminaria (apagado o encendido).
- Enviar la regulación (valor comprendido entre 0 y 100).
- Consultar el estado de la luminaria.
- Almacenar los datos en su memoria interna y trasladarlos a la base de datos MySQL.
- Enviar correo electrónico si se produce un fallo recurrente.

A continuación, se describen todos los scripts que ejecuta la Raspberry Pi 3. De aquí en adelante a la Raspberry Pi 3 se le hará referencia como 'Pi 3' y a la Raspberry Pi Pico como 'Pico'.

3.1. Programa '*Forzar.py*'.

Con este programa la Pi 3 envía a la Pico una secuencia de caracteres indicando la regulación y el identificativo de la luminaria, por ejemplo 'F1-100' (esto haría referencia a poner la Luminaria número 1 al 100% de su iluminación). Las Pico al recibir dicha secuencia reconocen el identificativo de la luminaria (F1). Si una Pico recibe un mensaje cuyo identificativo no corresponde al de su luminaria no hace nada. La segunda parte del mensaje, la posterior al guion, indica la regulación, en este caso '100'. Si se recibe un valor distinto a un número entre 0 y 100 la Pico no hace nada. A continuación, lo que hará la Pico de la luminaria 1 es activar la señal de salida llamada 'rele' (le asigna el valor 1) y a la señal de salida 'regulacion' le asigna el valor 100.

Cuando ya se ha modificado la regulación, se almacena el cambio en la MySQL interna de Pi 3, para que quede registrado dicho cambio.

```
#!/usr/bin/python
import MySQLdb
import serial
import sys
from time import sleep
import os, time
from datetime import datetime

#Conexion con la BBDD Mysql
db = MySQLdb.connect(host="localhost", user="***", passwd="***",
db="Monitorizacion")
#Fecha
Timestamp = datetime.now()
#Puerto Serie
antena = serial.Serial('/dev/serial0', 9600)

def Encender(valor):
    antena.write(valor)

def Escribir_MYSQL(Timestamp, Luminaria, Regulacion, Estado,
Calendario):
    cursor = db.cursor()
    consulta = "INSERT INTO Luminarias(Timestamp, Luminaria,
Regulacion, Estado, Calendario) VALUES ('%s','%s','%s','%s','%s')" %
(Timestamp, Luminaria, Regulacion, Estado, Calendario)
```

```
    try:
        cursor.execute(consulta)
        db.commit()

    except:
        db.rollback()
        sys.exit()

def main():
    try:
          Luminaria = sys.argv[1]
          Regulacion = sys.argv[2]
          Valor_Envio = "F" + Luminaria + "-" + Regulacion
          Encender(Valor_Envio)
          (Luminaria, Regulacion, Estado, Calendario) =
(int(Luminaria),int(Regulacion), 2, 1)
          Escribir_MYSQL(Timestamp, Luminaria, Regulacion, Estado,
Calendario)
          print(Timestamp,"L" + str(Luminaria), Regulacion, Estado,
Calendario)
    except:
      print(Timestamp, "Fallo")
```

3.2 Programa 'Consulta_Luminaria.py'.

Si con el programa anterior se modifica la regulación de la luminaria, surge la necesidad de crear un programa que permita conocer el estado actual de cada luminaria.

Con 'Consulta_Luminaria.py' desde Pi 3 se envía un mensaje a todas las Pico del tipo 'e-F2'. La Pico cuando recibe el mensaje contesta con uno del tipo '2-080', donde se indica el identificador de la luminaria y el porcentaje al que está regulada, en este caso, la luminaria 2 estaría al 80% de su iluminación total.

Una vez se ha producido la comunicación y la Pi 3 recibe el estado de cada luminaria, se almacenan la fecha, el identificativo de la luminaria, el estado y la regulación en la tabla MySQL interna de la Pi 3.

```
import MySQLdb
import signal
import resource
import math
import smtplib
import serial
from time import sleep
import os, time
from datetime import datetime
import sys
from contextlib import contextmanager
class TimeoutException(Exception): pass

#Calculo tiempo ejecuccion
start_time = time.time()

#Config Correo
sender_email = "alarmas@pavener.com"
Listado_Correos = ["igarcia@pavener.com"]
```

```
password = "***"

#Conexion con la BBDD Mysql
db = MySQLdb.connect(host="localhost", user="***", passwd="****",
db="Monitorizacion")
db2 = MySQLdb.connect(host="****", user="rasp", passwd="****",
db="Monitorizacion")

#Puerto Serie
antena = serial.Serial('/dev/serial0', 9600)

#Fecha
Timestamp = datetime.now()

#Esto es el temporizador
@contextmanager
def time_limit(seconds):
    def signal_handler(signum, frame):
        raise TimeoutException("Timed out!")
    signal.signal(signal.SIGALRM, signal_handler)
    signal.alarm(seconds)
    try:
        yield
    finally:
        signal.alarm(0)

#Consultamos el estado directamente a la Luminaria
def Consulta_Luminaria(Num_Luminaria):
  mensaje = "e-F" + str(Num_Luminaria)
  antena.write(mensaje)
  print('enviado: ',mensaje)
  time.sleep(3)
  rele = str(antena.read())
  porcentaje = str(antena.read())
  porcentaje = porcentaje + str(antena.read())
  porcentaje = porcentaje + str(antena.read())
  print(rele, porcentaje)
  return(rele,int(porcentaje))

#Exportamos los datos a la tabla mysql de la Pi
def Escribir_mysql(Timestamp, Luminaria, Regulacion, Estado,
Calendario):
    cursor = db.cursor()
    consulta = "INSERT INTO Luminarias(Timestamp, Luminaria,
Regulacion, Estado, Calendario) VALUES ('%s','%s','%s','%s','%s')" %
(Timestamp, Luminaria, Regulacion, Estado, Calendario)
    try:
        cursor.execute(consulta)
        db.commit()
    except:
        db.rollback()
        sys.exit()

#Exportamos los datos a la tabla SQL del Servidor
def Subir_MYSQL(Timestamp, Luminaria, Regulacion, Estado,
Calendario):
    cursor2 = db2.cursor()
    consulta2 = "INSERT INTO PAVENER_IOT_ESTADOS(Timestamp,
Luminaria, Regulacion, Estado, Calendario) VALUES
('%s','%s','%s','%s','%s')" % (Timestamp, Luminaria, Regulacion,
```

```
Estado, Calendario)
    try:
        cursor2.execute(consulta2)
        db2.commit()
    except:
        db2.rollback()
        sys.exit()

#Exportamos los datos a la tabla mysql de ERRORES
def Escribir_ERROR(Timestamp, Error_Luminaria):
    cursor3 = db.cursor()
    consulta3 = "INSERT INTO Luminarias_Errores(Timestamp,
Error_Luminaria) VALUES ('%s','%s')" % (Timestamp, Error_Luminaria)
    try:
        cursor3.execute(consulta3)
        db.commit()
    except:
        db.rollback()
        sys.exit()


# MAIN
def main():
  Timestamp = datetime.now()
  for i in range(2):
   j = i + 1
   try:
    with time_limit(5):
     (Estado, Regulacion) = Consulta_Luminaria(j)
     print(Timestamp, j, Regulacion, Estado, 1)
     Escribir_mysql(Timestamp, j, Regulacion, Estado, 1)
     time.sleep(2)
     Subir_MYSQL(Timestamp, j, Regulacion, Estado, 1)
   except TimeoutException as e:
    print("Tiempo agotado luminaria: ",j)
    Escribir_ERROR(Timestamp, j)
    print("-> %s seconds" % round((time.time() - start_time),2))

if __name__ == "__main__":
   main()
```

3.3 Programa '*Traspaso.py*'.

Todas las acciones que se realizan entre la Pi 3 y las Pico quedan registradas en la tabla MySQL de la Raspberry Pi 3. A pesar de que estos dispositivos son fiables y duraderos, por seguridad, es necesario hacer un traspaso de todos los datos desde la Pi 3 al servidor de la empresa PAVENER.

Por seguridad en esta copia del código no aparecen ni dirección ip, ni usuario, ni contraseña ni la base de datos que se utilizan en el servidor.

```
import MySQLdb

#Arrays
Timestamp = []
Luminaria = []
Regulacion = []
Estado = []

#Conexion con la BBDD Mysql
db = MySQLdb.connect(host="localhost", user="pavener",
passwd="pavenerTI", db="Monitorizacion")
db2 = MySQLdb.connect(host="pavener.tusser.net", user="rasp",
passwd="PavenerTI2017", db="Monitorizacion")

#Exportamos los datos a la tabla mysql de la Pi
def Leer_mysql():
    cursor = db.cursor()
    cursor.execute("SELECT * FROM Luminarias")
    myresult = cursor.fetchall()
    for row in myresult:
        Timestamp.append(row[1])
        Luminaria.append(row[2])
        Regulacion.append(int(row[3]))
        Estado.append(row[4])

def Subir_MYSQL(Timestamp, Luminaria, Regulacion, Estado,
Calendario):
   mycursor2 = db2.cursor()
   sql2 = "INSERT INTO PAVENER_IOT_ESTADOS(Timestamp, Luminaria,
Regulacion, Estado, Calendario) VALUES (%s,'%s','%s','%s','%s')"
   val2 = (Timestamp, Luminaria, Regulacion, Estado, Calendario)
   mycursor2.execute(sql2, val2)
   db2.commit()

def BorrarLocal():
    mycursor = db.cursor()
    sql = "DELETE FROM Luminarias WHERE ID > 0"
    mycursor.execute(sql)
    db.commit()
    print(mycursor.rowcount, "record(s) deleted")

def main():
    Leer_mysql()
    for i in range(len(Timestamp)):
       Subir_MYSQL(Timestamp[i], Luminaria[i], Regulacion[i],
Estado[i], 1)
    BorrarLocal()


if __name__ == "__main__":
   main()
```

3.4 Programa ‘Mail_Luminarias.py’.

Al tratarse de un proyecto que está pensado para ser utilizado por parte de terceros, es necesario conocer de forma rápida si ocurre algún error.

Uno de los errores más recurrentes de este sistema es que alguna Pico no responde cuando se le consulta el estado de su luminaria. Si esto pasa una vez al día no es algo relevante, pues cada 5 minutos se hacen consultas para saber el estado de la luminaria. Sin embargo, si por algún motivo se pierde la conexión con la luminaria y no sabemos si está apagada o encendida y cuánto tiempo hace que esté así, sí es un problema.

Frente a esta posibilidad, este programa se encarga de verificar que en la tabla MySQL interna de la Pi 3 ha habido entradas durante los últimos 15 minutos. De no ser así significa que no ha habido ningún tipo de conexión desde entonces. Si esto sucede, el programa, a través del correo electrónico interno de la empresa PAVENER me notifica alertando sobre qué luminaria ha perdido la conexión.

De esta manera resulta fácil y rápido saber si hay algún error de conexión con alguna de las luminarias.

```
#Enviar correo de alarma
def EnviarMail(user, pwd, recipient, subject, body):
    FROM = user
    TO = recipient if isinstance(recipient, list) else [recipient]
    SUBJECT = subject
    TEXT = body
    #Construir el mensaje
    message = """From: %s\nTo: %s\nSubject: %s\n\n%s """ % (FROM, ",
".join(TO), SUBJECT, TEXT)
    try:
        server = smtplib.SMTP("smtp.gmail.com", 587)
        server.ehlo()
        server.starttls()
        server.login(user, pwd)
        server.sendmail(FROM, TO, message)
        server.close()
        print('Correo enviado')
    except:
        print("Algo fue mal..")

# MAIN
def main():
  (error1,error2) = Consulta_Errores()
  if error1 >= 3:
    Asunto = "Alarma Luminarias"
    Cuerpo = "Conexion perdida 3 veces con la luminaria 1."
    EnviarMail(sender_email, password, Listado_Correos, Asunto,
Cuerpo)
  if error2 >= 3:
    Asunto = "Alarma Luminarias"
    Cuerpo = "Conexion perdida 3 veces con la luminaria 2."
    EnviarMail(sender_email, password, Listado_Correos, Asunto,
Cuerpo)

if __name__ == "__main__":
   main()
```

3.5 Programa main.py configurado en la Raspberry Pi Pico.

Como se comenta anteriormente en la Raspberry Pi Pico se configura un programa llamado 'main.py'. Este dispositivo funciona de tal manera que una vez se encienda, está ejecutando continuamente ese script.

```
from machine import UART, Pin, PWM
import time

uart0 = UART(0, baudrate=9600, tx=Pin(0), rx=Pin(1))
led_onboard = machine.Pin(25, machine.Pin.OUT)
led_pwm = PWM(Pin(16, Pin.OUT))
led_rele = Pin(17, Pin.OUT)

led_onboard.value(1)
led_rele.value(0)
led_pwm.freq(1000)
rx_rele = 2

def ComprobarEstado(rele, porcentaje):
    if rele == 0:
        respuesta1 = str(rele)
        uart0.write(respuesta1) #envia el 0
    elif rele == 1:
        respuesta1 = str(rele)
        uart0.write(respuesta1) #envia el 1
        respuesta2 = str(porcentaje)
        #print(respuesta2)
        uart0.write(respuesta2) # envia el porcentaje

def RegularLuminaria(luminaria, porcentaje,rx_rele):
    if porcentaje == '000':
        led_rele.value(0)
        rx_rele = 0
        regulacion = 0
        led_pwm.duty_u16(int(regulacion))
        #print("Luminaria "+str(luminaria)+" apagada")
        #print()
        return rx_rele, porcentaje
    elif porcentaje == '100':
        led_rele.value(1)
        rx_rele = 1
        regulacion = 65025
        led_pwm.duty_u16(int(regulacion))
        #print("Luminaria "+str(luminaria)+" con regulación
total.")
        #print()
        return rx_rele, porcentaje
    else:
        led_rele.value(1)
        rx_rele = 1
        regulacion = 65025*(int(porcentaje)/100)
        led_pwm.duty_u16(int(regulacion))
        #print("Luminaria "+str(luminaria)+" al
"+(porcentaje[1:3])+"%")
        #print()
        return rx_rele, porcentaje
```

```
def main():
    while uart0.any() >= 0:
        mensaje1 = str(uart0.read(11))
        mensaje2 = mensaje1[2:11]
        luminaria = mensaje2[0:2]
        if luminaria == 'F1':
            porcentaje = mensaje2[3:6]
            rele, porcentaje = RegularLuminaria(luminaria,
porcentaje, rx_rele)
        if mensaje2 == 'e-F1':
            estado = mensaje2[0:1]
            ComprobarEstado(rele, porcentaje)


if __name__ == "__main__":
    main()
```

3.6 Archivo crontab de Linux.

Crontab es un archivo donde se escribe una lista de comandos a ejecutar en un tiempo específico. Haciendo uso de esto resulta sencillo poder ejecutar los anteriores programas a las horas que deseemos.

Por otro lado, como ya se ha mencionado, el encargado de recibir la información enviada por la Pi 3 y ejecutar las acciones pertinentes a las luminarias es la Pico. Ésta no funciona con un sistema operativo como tal, sino que se le pueden programar una serie de instrucciones en lenguaje Micro Python o C a través de un software desde el ordenador, en este caso he utilizado el software Thonny.

Así pues, en la Raspberry Pi Pico programamos la siguiente serie de instrucciones y guardamos dicho archivo como 'main.py'. Por ese motivo, cuando se encienda, de forma automática ejecuta ese programa por defecto.

```
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/

#Forzamos la iluminacion de las luminarias
15 21 * * * /home/pi/Luminarias/Forzar.py 2 100
30 00 * * * /home/pi/Luminarias/Forzar.py 2 040
00 07 * * * /home/pi/Luminarias/Forzar.py 2 000

#Hacemos la consulta cada 5 minutos y comprobamos que no haya errores
*/3 * * * * python /home/pi/Luminarias/Consulta_Luminarias.py >> loglum.log
#16 * * * 1,2,3,4,5  pi /home/pi/Luminarias/Mail_Luminarias.py

#Traspaso de los datos de la Pi al servidor
* 6 * * * /home/pi/Luminarias/Traspaso.py
```

Figura 21. Captura de pantalla del archivo crontab de la Raspberry Pi 3.

Capítulo 4. Pruebas de funcionamiento.

Con todos los circuitos montados y conectados a sus luminarias era hora de empezar a realizar las pruebas de funcionamiento del sistema.

4.1. Pruebas iniciales.

En un principio solo utilizamos la Raspberry Pi 3 y una de las luminarias con su circuito y la Raspberry Pi Pico.

Las primeras pruebas que hicimos fueron en el mismo escritorio de oficina donde teníamos todo montado. Consistieron en saber cómo se comunicaban entre sí los dispositivos, cómo configuramos el módulo E32, qué enviamos o cómo se recibían los datos.

Se puede concluir que las primeras pruebas fueron un éxito al ver que desde la Pi 3 enviamos un 0 y la luminaria se apagaba y si enviamos un 1 la luminaria se encendía. Después comprobamos que si enviamos un entero entre 0 y 100 la luminaria regulaba su iluminación a ese porcentaje. Los mensajes a esta distancia no tenían apenas retardo.

4.2. Pruebas secundarias.

Una vez comprobada la conexión entre la Pi 3 y una luminaria, fue cuando añadimos la segunda luminaria. Ahora ya teníamos: la Pi 3 y las dos luminarias con sus respectivos circuitos y su Pico correspondiente. Comprobamos que después de añadir la segunda luminaria el sistema seguía funcionando de forma esperada.

Como teníamos todo montado encima de un escritorio, decidimos probar a llevar las luminarias a otra parte de las oficinas. Ahora las luminarias las teníamos colocadas en una planta superior a la planta donde estaba ubicada la Pi 3. De esta forma, entre Pi 3 y luminarias habrá en torno a 10 o 12 metros de distancia. Ya con este escenario, decidimos dejar funcionando el sistema durante unas semanas. Éste encendía las luminarias al 100% a las 21:30 horas, las regulaba al 80% a las 00:30 horas y las apagaba a las 7:30 horas de la mañana.

Después de dejar el sistema trabajando de forma autónoma durante esas semanas observamos su comportamiento. Consultando los datos almacenados en la base de datos del servidor pudimos ver que durante la mayor parte del tiempo el sistema trabajaba perfectamente, pero en ocasiones había fallos.

Un fallo recurrente durante esos días fue que, en ocasiones, cuando tenía que regular al 80%, en vez de enviar un ‘080’ enviaba un ‘030’. Esto supimos solucionarlo. Se debía a que, en la Pico, había un valor que no se asignaba bien a la variable ‘regulación’.

Otro fallo que ocurría, menos frecuente que el anterior, era la pérdida de conexión con una de las luminarias. En ocasiones, cuando se le consultaba el estado de la luminaria número 2, ésta no contestaba. Nos dimos cuenta porque había muchas menos entradas en la base de datos de la luminaria 2 en comparación con la luminaria 1. Una posible solución que pensamos fue en cambiar la placa de pruebas donde estaba montado el circuito regulador de la luminaria 2, y así lo hicimos.

También observamos que el tiempo que tardaba el mensaje desde que salía de la Pi 3 hasta que se almacenaba en la base de datos interna de la misma no superaba el segundo.

Después de observar el comportamiento durante algunas semanas más, observamos que la luminaria 2 seguía dando problemas, por lo que intercambiamos los circuitos entre las luminarias para observar si ese pudiera ser el problema. Efectivamente, después de unas cuantas muestras durante unas horas, nos dimos cuenta de que la luminaria 2 dejó de tener esos fallos y empezó a ser la luminaria 1 la que los tenía.

Caímos en la cuenta de que la solución podría pasar por mandar imprimir una placa pcb que tuviera esos circuitos, la Pico y el módulo E32 integrados. Así pues, diseñamos la placa pcb descrita en el apartado 2.2.4.6 de este proyecto.

4.3. Pruebas exteriores.

Hasta que llegaran las placas pcb fabricadas decidimos medir la distancia a la que podíamos operar con nuestro sistema. Un buen lugar para hacerlo es la cantera que tiene la empresa PAVASAL en Quart de Poblet, que está al lado de donde PAVENER tiene sus oficinas. Para estas pruebas solo tuvimos en cuenta la luminaria 2, ya que la 1 nos había dado los problemas comentados anteriormente.

Las luminarias, obviamente por su peso y necesidad de estar conectadas a la red, se quedaron fijas en las oficinas. No obstante, cogimos la Pi 3 con el módulo E32 y un ordenador portátil y estudiamos el comportamiento del sistema en diferentes puntos. En las siguientes imágenes se puede ver las diferentes distancias a las que probamos el sistema.

Figura 22. Captura de pantalla de la situación de la cantera de PAVASAL y las oficinas de PAVENER.

En primer lugar, probamos desde el aparcamiento de las oficinas. En este escenario entre los dos puntos existe el edificio de oficinas, no hay visión directa, por lo que nos imaginamos que podría alterar algo el sistema. Como se observa en la figura 22, entre los dos puntos hay aproximadamente 72 metros de distancia. Desde este punto pudimos observar cómo se regulaba bien la luminaria 1 y cómo pudimos consultar sin problema el estado de la luminaria. El retardo de la comunicación en ningún caso superó los 1.5 segundos.

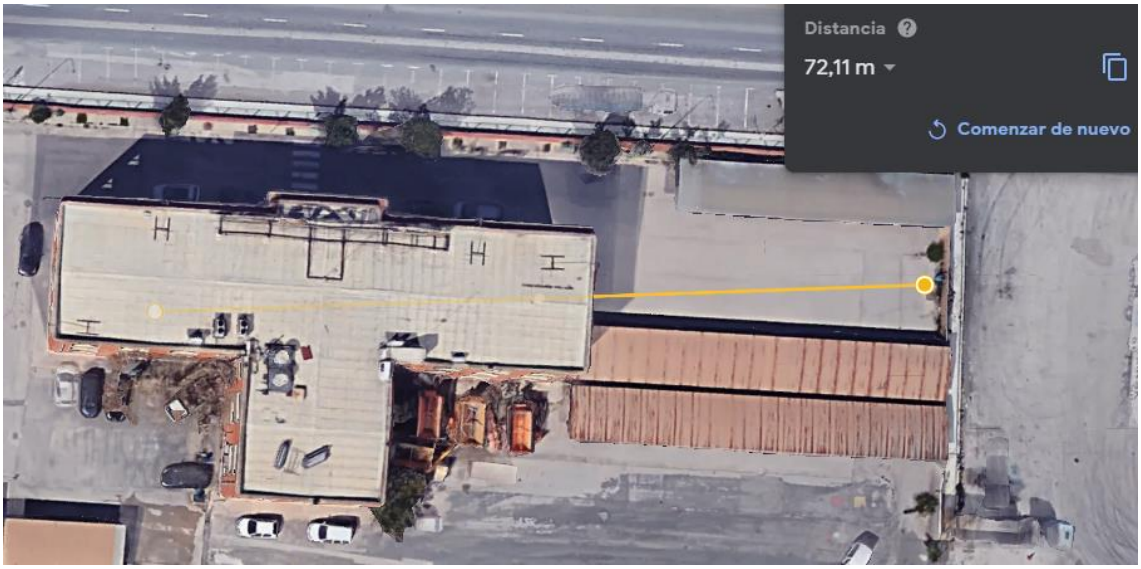

Figura 23. Vista aérea del primer punto de prueba entre las oficinas de PAVENER y el aparcamiento.

Como vimos que funcionó correctamente decidimos alejarnos unos metros más. En la figura 23 se puede observar que ahora, desde las luminarias hasta la Pi 3 hay 102 metros de distancia.

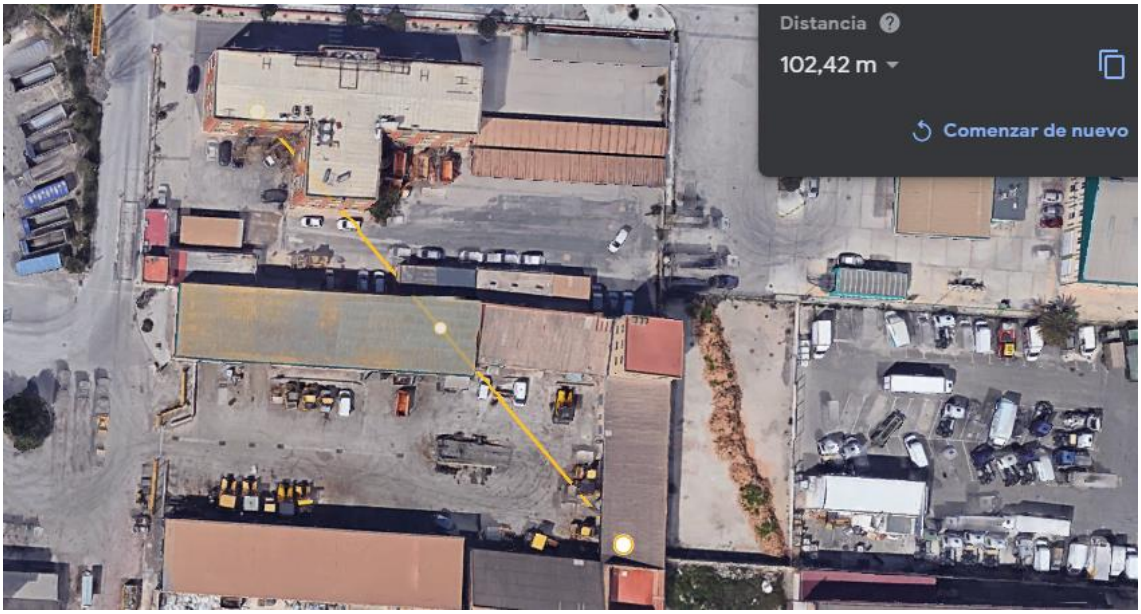

Figura 24. Vista aérea del segundo punto de prueba entre las oficinas de PAVENER y el taller de maquinaria de PAVASAL.

En este escenario observamos cómo el sistema seguía funcionando perfectamente. Desde allí modificamos y consultamos varias veces el estado de la luminaria 1 con éxito. He de decir que desde este punto tampoco se observó un retardo en la comunicación punto a punto superior a los 1.5 segundos.

El siguiente punto donde nos alejamos fue lo más lejos posible dentro del recinto, exactamente a unos 192 metros de distancia, como se puede observar en la figura 24.

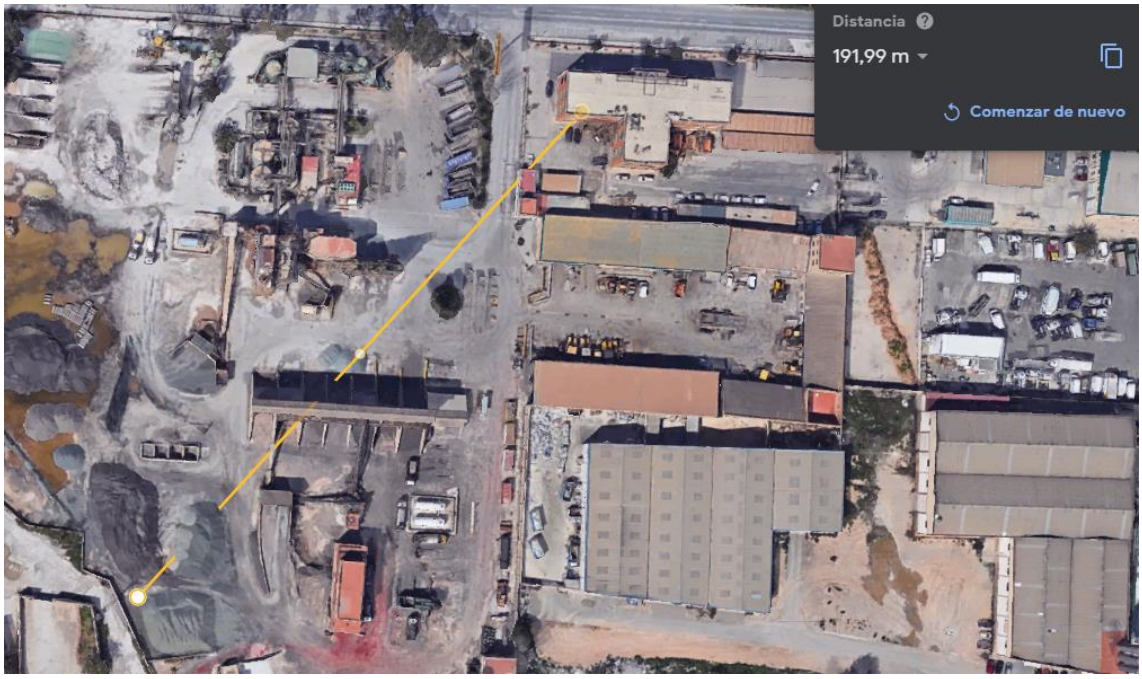

Figura 25. Vista aérea del tercer punto de prueba desde la cantera de PAVASAL y las oficinas de PAVENER.

Desde aquí pudimos observar un retardo superior al medido anteriormente. No obstante, éste nunca superó los 2 segundos.

Cuando comenzamos a realizar los primeros envíos desde la Pi 3 observamos que en ocasiones no se obtenía respuesta por parte de la luminaria, quedando la Pi 3 en espera de esa respuesta. Esto nos dio la idea de poner un temporizador para que, si alguna luminaria no responde en un tiempo inferior a 5 segundos, se corte la comunicación y la Pi 3 deje de estar esperando una respuesta.

Me hubiera gustado probar el sistema en unas luminarias y en un cuadro eléctrico reales, pero es algo complejo. Aunque la empresa tenga contratados los mantenimientos de varias instalaciones de este tipo, es muy complejo llevar a un estudiante en prácticas a la calle de un municipio a probar un dispositivo en el alumbrado público.

Capítulo 5. Almacenamiento y análisis de datos.

Los servidores MySQL son una herramienta muy útil para el análisis de datos. En este proyecto se utiliza el software Workbench en el ordenador para visualizar tanto la base de datos de la Raspberry Pi 3 como la base de datos de PAVENER. La conexión para visualizar los datos de la Pi 3 se hace mediante SSH. En cambio, para visualizar los datos del servidor de la empresa, la conexión se realiza mediante una VPN. Para este proyecto trabajamos con dos tablas, una en la Pi 3 llamada 'Luminarias' y una en el servidor de la empresa llamada 'PAVENER_IOT_ESTADOS'.

Como se ha comentado en capítulos anteriores hacemos un traspaso de los datos desde la Raspberry Pi 3 hacia el servidor de la empresa, una copia de seguridad. Entonces, cada vez que se consulta el estado de una luminaria o que se le cambia la iluminación se queda registrado en la tabla MySQL. En ella se registra la fecha, la luminaria (1 o 2), el estado (0: apagada, 1: encendida ó 2: forzada), la regulación (que puede variar de 0 a 100) y el calendario donde almacenamos los datos.

Debido a un problema logístico, solo se ha podido experimentar con la 'Luminaria 2' desde el día 15 de junio del año 2021.

En la siguiente imagen se puede apreciar un ejemplo de la tabla del servidor.

ID	Timestamp	Luminaria	Estado	Regulacion	Calendario
29067	2021-07-26 21:57:02	2	1	100	1
29066	2021-07-26 21:54:01	2	1	100	1
29065	2021-07-26 21:51:02	2	1	100	1
29064	2021-07-26 21:48:02	2	1	100	1
29063	2021-07-26 21:45:02	2	1	100	1
29062	2021-07-26 21:42:01	2	1	100	1
29061	2021-07-26 21:39:02	2	1	100	1
29060	2021-07-26 21:36:01	2	1	100	1
29059	2021-07-26 21:33:02	2	1	100	1
29058	2021-07-26 21:30:01	2	1	100	1
29057	2021-07-26 21:28:01	2	2	100	1
29056	2021-07-26 21:27:01	2	0	0	1
29055	2021-07-26 21:24:03	2	0	0	1
29054	2021-07-26 21:21:01	2	0	0	1
29053	2021-07-26 21:18:01	2	0	0	1
29052	2021-07-26 21:15:02	2	0	0	1
29051	2021-07-26 21:12:01	2	0	0	1
29050	2021-07-26 21:09:01	2	0	0	1

Figura 26. Captura de pantalla de la tabla MySQL donde se muestra cómo se enciende la luminaria número 2 a las 21 horas.

Una vez los datos quedan almacenados en el servidor, observarlos y analizarlos directamente desde la tabla es tedioso. Para evitar tener que hacer eso, representamos gráficamente los valores de las columnas 'Fecha' y 'Regulacion', para cada luminaria. Estas gráficas se pueden observar en un *web server* creado con PHP. De esta manera resulta mucho más sencillo visualizar el comportamiento de las luminarias o identificar posibles errores.

En este caso, se tienen dos tipos de gráficas para cada luminaria. Una representa la regulación de la luminaria durante el día actual, con muestras cada 3 minutos (figura 10). Por otra parte, se tiene un gráfico que representa la regulación de cada luminaria de los 7 días anteriores a la fecha actual (figura 11).

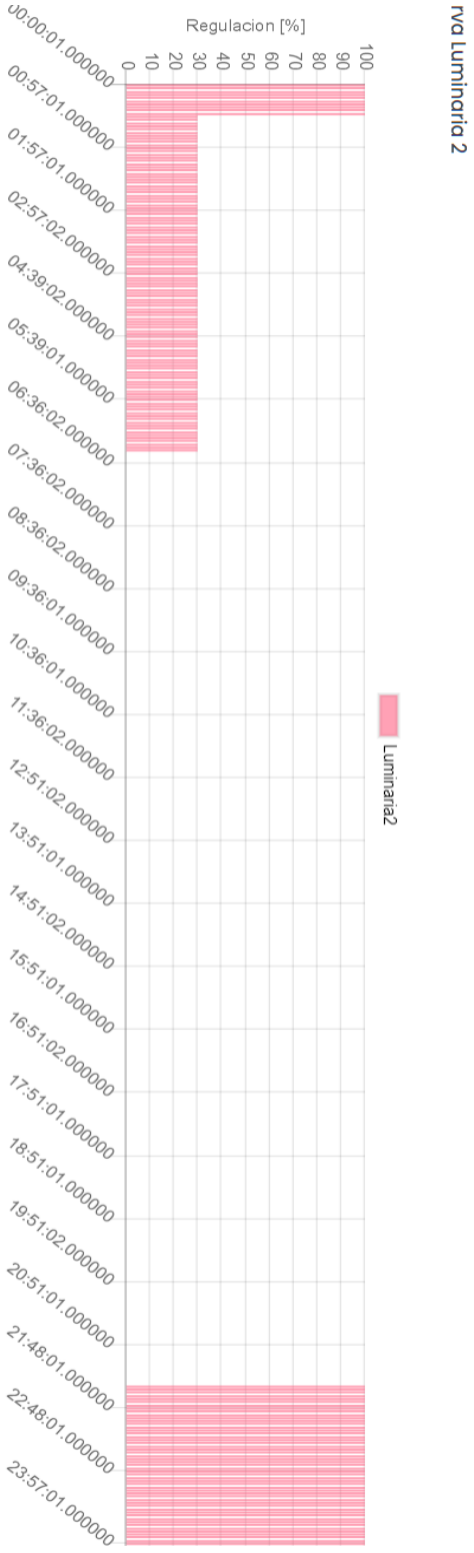

Figura 27. Captura de pantalla del gráfico que podemos observar en el web server donde se representa la regulación de la iluminación a lo largo del día.

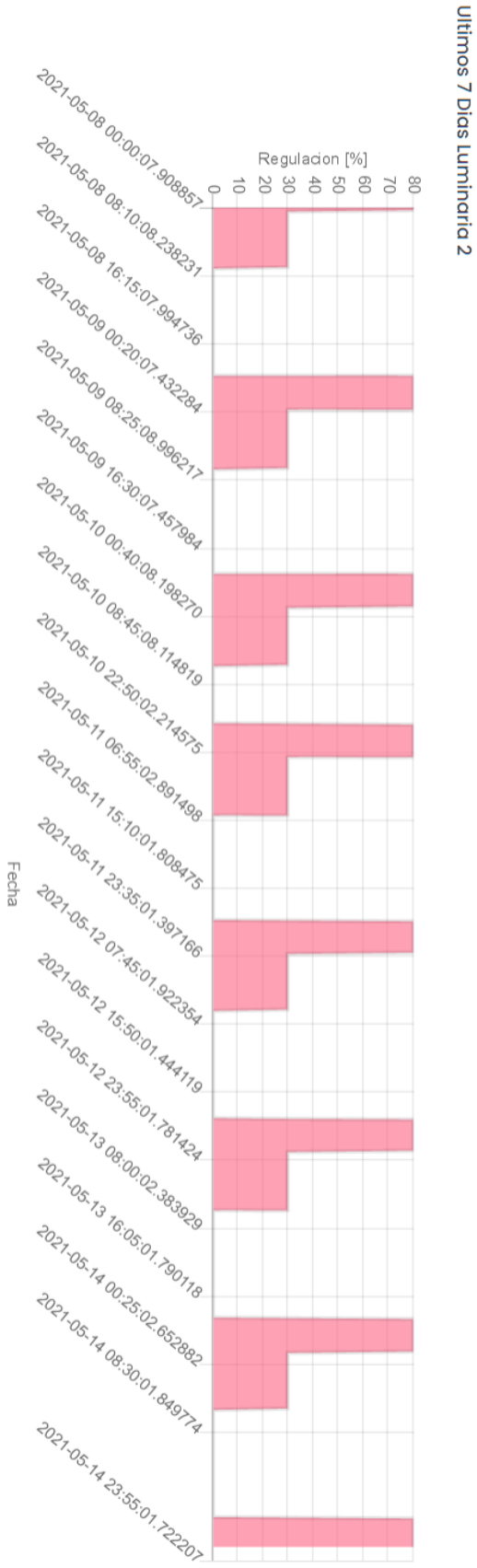

Figura 28. Captura de pantalla del gráfico que podemos observar en el web server donde se representa la regulación de la iluminación a lo largo de la última semana.

Capítulo 6. Análisis económico del proyecto.

Como se menciona al inicio del documento, uno de los objetivos del proyecto es llevar a cabo este sistema de la manera más económica posible.

Existen fabricantes como Schneider o Siemens que venden módulos de comunicaciones LoRa. Su configuración e implementación no son muy complicadas y son módulos con una alta fiabilidad y robustez. Sin embargo, el precio de estos módulos suele estar en torno a los 200 euros, lo que supera nuestro objetivo de implementar este sistema con un presupuesto inferior a los 100 euros.

En la siguiente tabla se puede ver una comparativa económica entre utilizar un módulo ya fabricado y utilizar este sistema para una misma aplicación.

Módulos fabricados	Sistema punto a punto basado en LoRaWAN
Siemens Gateway IoT 2040 (243.20€)	Raspberry Pi 3 (44.89€)
STM32L0 Transceptor Node (39,5€)	Raspberry Pi Pico (4,50€)
Circuito impreso diseño propio (5€)	E32-433T30D (7.36€)
	Circuito impreso diseño propio (5€)

Tabla 2. Comparativa económica entre realizar el proyecto con módulos ya fabricados para esta aplicación y módulos más generales.

Los precios de cada producto representados en la tabla han sido obtenidos de las páginas web de cada fabricante, a excepción del precio de los circuitos impresos, el cual depende de cuantos circuitos quieras imprimir.

Si para este proyecto se hubieran utilizado los módulos fabricados quedaría un presupuesto tal que así:

- Gateway del sistema 'Siemens Gateway IoT 2040' (1 unidad).
- Nodo del sistema 'STM32L0 Transceptor Node' (2 unidades).
- Circuito impreso (2 unidades).

De este modo el coste total del proyecto hubiera sido de 332.2 euros. No obstante, el presupuesto del proyecto ha sido el siguiente:

- Raspberry Pi 3 (1 unidad).
- Raspberry Pi Pico (2 unidades).
- Transceptor 'E32-433T30D' (2 unidades).
- Circuito impreso (2 unidades).

Con todo esto, el coste final del proyecto es de 78.61 euros. Lo que encaja perfectamente con nuestro objetivo inicial de realizar el sistema por un coste inferior a 100 euros.

Capítulo 7. Líneas futuras.

Hoy en día LoRa está bastante desarrollado y su aplicación es variada. Este proyecto tiene la parte de la conexión punto a punto entre dos dispositivos y la parte del diseño e implementación de un circuito que sea capaz de regular la iluminación de una luminaria.

Podemos decir que las aplicaciones que puede tener el circuito regulador de iluminación son ajustadas. Ahora bien, la parte de la conexión punto a punto puede tener múltiples aplicaciones. Hoy en día estando tan en auge las Smart Cities, es necesario buscar formas alternativas a la comunicación entre dispositivos.

Una aplicación alternativa de este sistema, con pequeñas modificaciones, podría ser la iluminación de luminarias en un puente o en un paseo. Hace un mes y medio asistí a una reunión entre los equipos de investigación y desarrollo de las empresas PAVENER y PAVAPARK.

Durante esta reunión ambos equipos comentamos proyectos en marcha e ideas para llevarlos a cabo. Cuando explicamos el proyecto de la conexión punto a punto para regular luminarias, al equipo de PAVAPARK les pareció una posible solución para uno de sus proyectos. Su proyecto, según nos explicaron, consiste en la instalación de unas luminarias a lo largo de un puente que permanezcan reguladas a la mínima iluminación excepto cuando pase un peatón, un ciclista o un coche.

De esta forma, según entrara un coche por un extremo del puente se irán encendiendo las lámparas en serie hasta que el coche saliera del puente, quedando de nuevo las luminarias a su paso reguladas al mínimo.

Conclusiones.

Hasta ahora no conocía la tecnología LoRa. Después de estar unos cuantos meses trabajando con LoRa he entendido que es una muy buena solución para muchas cosas. Tiene muchas aplicaciones en lugares con poca cobertura, donde no puedes comunicar dos dispositivos a través de internet. La única desventaja que tiene es que permite transmitir una cantidad de datos pequeña. Sin embargo, tiene un alcance razonablemente bueno y es barato de implementar.

Estudiando el proceso del proyecto he podido llegar a la conclusión de que ha merecido la pena investigar una forma alternativa de comunicarse mediante LoRa. Hasta la fecha nunca he tenido que investigar y desarrollar algo. Me parece un trabajo importante y a la vez difícil, siempre andar trabajando con la técnica del 'prueba y error'. Para llegar a buen puerto es muy importante saber cuál es el objetivo final y de qué medios dispones, tanto materiales como a nivel de conocimientos.

Pues si al principio, como todo inicio, había más dudas que certezas, ahora puedo afirmar que conocemos el modo de desarrollar un sistema de control punto a multipunto mediante LoRa de una manera más económica que otro sistema fabricado en el mercado.

En cuanto al grado se refiere, me he dado cuenta de que nuestro conocimiento como ingenieros de telecomunicaciones debe abarcar un espectro bastante amplio. Durante el transcurso del proyecto he comprendido que para ser ingeniero se debe saber tanto programar, como implementar un circuito, manejarte con dispositivos muy distintos o entender cómo funcionan las comunicaciones inalámbricas. Hay que estar siempre dispuesto a aprender sobre otras ramas de la ingeniería y no ceñirse a lo que a uno le gusta o se encuentra cómodo.

Bibliografía.

[1]. LoRa Alliance. What is LoRaWAN Specification [Internet]. Disponible en: https://lora-alliance.org/about-lorawan

[2]. Sabas, (2 de octubre de 2017). *Medium.com*. Haciendo IoT con LoRa: Capítulo 2.- Tipos y clases de nodos. Disponible en: https://medium.com/beelan/haciendo-iot-con-lora-capitulo-2-tipos-y-clases-de-nodos-3856aba0e5be

[3]. Sabas, (9 de octubre de 2017). *Medium.com*. Haciendo IoT con LoRa: Capítulo 3.- Tipos de Gateway. Disponible en: https://medium.com/beelan/haciendo-iot-con-lora-capitulo-3-tipos-de-gateways-756afdf0487d

[4] Raspberry Pi 3. Disponible en: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

[5] Raspberry Pi Pico. Disponible en: https://www.raspberrypi.org/documentation/microcontrollers/rp2040.html#welcome-to-rp2040

[6] Precio Gateway LoRa. Disponible en: https://es.rs-online.com/web/p/gateways-iot/1244038/?cm_mmc=ES-PLA-DS3A-_-google-_-PLA_ES_ES_Automatizaci%C3%B3n_y_Control_de_Procesos_Whoop-_-(ES:Whoop!)+Gateways+IoT+(2)-_-1244038&matchtype=&pla-303922359082&gclid=Cj0KCQjw4eaJBhDMARIsANhrQADwPo_zO7YB9UjBMCT6gsmtieVLuVoNoCMlXVhRcAny_ZlAzUGIimkaAsjTEALw_wcB&gclsrc=aw.ds

[7] Precio del nodo LoRa. Disponible en: https://www.digikey.es/product-detail/es/stmicro/B-L072Z-LRWAN1/497-17068-ND/6616000

[8] Datos del módulo E32. Disponible en: https://www.ebyte.com/en/product-view-news.aspx?id=108

[9] Precio del módulo E32. Disponible en: https://es.aliexpress.com/item/1005001839836390.html?src=google%2Chttps%3A%2F%2Fes.aliexpress.com%2Fitem%2F1005001839836390.html%3F_randl_currency%3DEUR&src=google&src=google&albch=shopping&acnt=439-079-4345&slnk=&plac=&mtctp=&albbt=Google_7_shopping&gclsrc=aw.ds&albagn=888888&ds_e_adid=475827849334&ds_e_matchtype=&ds_e_device=c&ds_e_network=u&ds_e_product_group_id=855473407372&ds_e_product_id=es1005001839836390&ds_e_product_merchant_id=106575158&ds_e_product_country=ES&ds_e_product_language=es&ds_e_product_channel=online&ds_e_product_store_id=&ds_url_v=2&ds_dest_url=https%3A%2F%2Fs.click.aliexpress.com%2Fdeep_link.htm%3Faff_short_key%3DUneMJZVf&albcp=11489913537&albag=114956049489&isSmbAutoCall=false&needSmbHouyi=false&gclid=Cj0KCQjw4eaJBhDMARIsANhrQADwFnrJc9RF2bI7yk7CEiLw4g4mwZlvp4P5JZTROgd_Rz5x6x9nF1AaAug4EALw_wcB&aff_fcid=14b49db4a86c47dba99db85aaca28ed0-1631196016509-07173-UneMJZVf&aff_fsk=UneMJZVf&aff_platform=aaf&sk=UneMJZVf&aff_trace_key=14b49db4a86c47dba99db85aaca28ed0-1631196016509-07173-UneMJZVf&terminal_id=442b931768af47b08f7b41747c864661