Universidad de la República
Facultad de Ingeniería

Control Inteligente de Luminaria LED

Memoria de proyecto presentada a la Facultad de Ingeniería de la Universidad de la República por

Emilio Albarracín, Tomás Arrivillaga, Felipe Morán

en cumplimiento parcial de los requerimientos para la obtención del título de Ingeniero Electricista.

Tutor

Dr. Ing. Leonardo Steinfeld ..... Universidad de la República

Tribunal

Mag. Ing. Sebastián Fernández ..... Universidad de la República
Dr. Ing. Germán Fierro ..... Universidad de la República
Mag. Ing. Gabriel Gómez ..... Universidad de la República
Dr. Ing. Leonardo Steinfeld ..... Universidad de la República

Montevideo
lunes 21 septiembre, 2020

Control Inteligente de Luminaria LED, Emilio Albarracín, Tomás Arrivillaga, Felipe Morán.

Esta tesis fue preparada en LaTeX usando la clase iietesis (v1.1).
Contiene un total de 153 páginas.
Compilada el lunes 21 septiembre, 2020.
http://iie.fing.edu.uy/

Agradecimientos

A nuestros padres, hermanos y familia en general, por ser pilar fundamental de nuestro desarrollo académico y personal.

A Isabel, Stephanie y amigos, por su apoyo incondicional no solo durante el transcurso de este proyecto si no durante toda la carrera.

A Leonardo Steinfeld, por su labor como tutor, en particular su disponibilidad a contestar dudas en todo momento.

A Mauricio Gonzalez, Agustín Rodríguez y Javier Schandy por ayudarnos en diferentes aspectos y darnos acceso a la oficina de Focus para trabajar en el proyecto en época de pandemia.

A Fernando Silveira y Andrés Seré, por su disposición a ayudarnos.

Esta página ha sido intencionalmente dejada en blanco.

Resumen

En el presente documento se describe el trabajo realizado en el proyecto de fin de carrera Control Inteligente de Luminaria LED, donde se desarrolló un controlador de luminaria LED para alumbrado público. Los controladores de luminaria tienen como objetivo dotar de inteligencia a las luminarias de manera tal que sea posible comunicarse con ellas de forma inalámbrica para su monitoreo y control (encendido y apagado, dimerizado, etc.), permitiendo además que puedan encenderse o apagarse de forma autónoma.

El controlador desarrollado en el presente proyecto es capaz de encender y apagar la luminaria, y utiliza el protocolo 1-10V para manejar el dimerizado de la misma cuando está encendida. Además, el controlador utiliza la tecnología LoRaWAN para comunicarse de manera inalámbrica con un servidor central. Esta comunicación permite que el controlador reciba comandos desde el Servidor Central y responda a ellos cuando corresponda.

El controlador tiene cuatro modos de funcionamiento, según la fuente de información principal utilizada para la toma de decisiones. Estos modos son: Directo, en el que se controla la luminaria en base a instrucciones directas de encendido, dimerizado y apagado desde un servidor central; Plan Horario, en el que se controla la luminaria en base a un plan horario configurable con eventos que indican hora y nivel de luz que debe dar la luminaria a partir de dicha hora; Reloj Astronómico, en el que se controla la luminaria en base a la hora de amanecer y atardecer de cada día; y Fotocélula, en el que se controla la luminaria en base al nivel de luz ambiente. Los modos Plan Horario y Reloj Astronómico permiten la adición de la información del nivel de luz ambiente para encender la luminaria en las horas diurnas en que haya baja luminosidad. Además, los modos Plan Horario, Reloj Astronómico y Fotocélula permiten que el controlador comande la luminaria de forma autónoma, sin comunicarse con el Servidor Central.

El controlador consta de diferentes módulos hardware interconectados, cada uno con una función específica. Estos módulos se ubican en tres placas de circuitos impresos circulares, conectadas entre sí mediante cables flexibles, una encima de la otra. Cada placa cuenta con tres orificios, a través de los cuales se insertan tornillos que atraviesan las tres placas, dándole rigidez mecánica al conjunto. Este conjunto conectado con una base para su conexión a una luminaria compatible ANSI C136.41, y protegidos con una tapa, conforman el controlador.

Los objetivos particulares que se definieron para el proyecto se basan en la Licitación Pública 670/2017 de la IM, vigente al momento de publicación de este documento, la cual pretende cambiar las luminarias del alumbrado público a lumi-

narias LED dotadas de inteligencia por un controlador. En la misma se mencionan los requerimientos que debe cumplir dicho controlador. Si bien en el mercado existen controladores que cumplen con los requerimientos, su diseño de hardware y su software son propietarios. Entonces, resulta de interés diseñar y construir un controlador con funciones similares pero con diseño "open-source".

Glosario

- **ABP:** *Activation By Personalization* (del inglés), método de activación de un dispositivo end-point en una red LoRaWAN.
- **ADC:** *Analog-to-Digital Converter* (del inglés), conversor de señales analógicas a digital.
- **ASCII:** *American Standard Code for Information Interchange* (del inglés), estándar de codificacion de caracteres.
- **CCS:** *Code Composer Studio* (del inglés), entorno de desarrollo integrado de software embebido de Texas Instruments.
- **GPIO:** *General Purpose Input Output* (del inglés), entrada-salida de propósito general.
- **I2C:** *Inter-Integrated Circuit* (del inglés), bus serial de comunicación sincrónica, multi-maestro, multi-esclavo.
- **LSB:** *Least Significant Bit* (del inglés), bit más significativo.
- **MCU:** *Microcontroller Unit* (del inglés), microcontrolador.
- **MSB:** *Most Significant Bit* (del inglés), bit más significativo.
- **OTAA:** *Over The Air Activation* (del inglés), método de activación de un dispositivo end-point en una red LoRaWAN.
- **PCB:** *Printed Circuit Board* (del inglés), placa con circuito impreso.
- **PMIC:** *Power Monitoring Itegrated Circuit* (del inglés), circuito integrado de monitoreo de consumo.
- **PWM:** *Pulse Width Modulation* (del inglés), modulación por ancho de pulso.
- **RTC:** *Real Time Clock* (del inglés), Reloj de tiempo real.
- **TCP/IP:** *Transmission Control Protocol/Internet Protocol* (del inglés), conjunto de protocolos de comunicación utilizados en internet y otras redes de computadoras.

- **Transceiver:** *Transmitter-Receiver* (del inglés), dispositivo para enviar y recibir datos de forma inalámbrica.
- **TTN:** *The Things Network* (del inglés), plataforma que provee servidores de red y permite crear aplicaciones para redes LoRaWAN.
- **UART:** *Universal Asyncronous Receiver Transmitter* (del inglés), dispositivo de comunicación serial asíncrona.
- **WDT:** *Watch Dog Timer* (del inglés) perro guardian. Mecanismo de seguridad que provoca un reset del sistema en casa de que este se haya bloqueado.

Tabla de contenidos

Tabla de contenidos

Capítulo 1

Introducción

1.1. Contexto y motivación

Con el transcurso del tiempo los dispositivos destinados a la iluminación han mejorado su rendimiento, es decir, ha aumentado la relación entre flujo luminoso y potencia consumida. Actualmente las luminarias tipo LED son las líderes del mercado, dado su muy bajo consumo y su buen índice de reproducción cromática. El ahorro energético que se logra al utilizar luminarias LED en el alumbrado público es tan importante que muchas ciudades ya han hecho la transición a esta tecnología, cambiando las conocidas lámparas de vapor de sodio o de mercurio. Por otra parte, en la actualidad es posible comunicarse de forma inalámbrica con las luminarias para su monitoreo y control (encendido y apagado, dimerizado, etc.), debido a la aparición de nuevas tecnologías de comunicación. De esta manera se llega al concepto de alumbrado público inteligente, que entre otras cosas permite el acceso inalámbrico al control individual de cada una de ellas desde una computadora o un celular. A su vez, es posible ampliar la inteligencia de las luminarias de forma tal que puedan encenderse o apagarse de acuerdo a un horario configurado o según el nivel de luz ambiente.

Por otro lado, la infraestructura de comunicación que se generaría para tener acceso al control inalámbrico de las luminarias serviría para sumar sensores que mejoren la calidad de vida en la ciudad. Por ejemplo, un contador de tráfico, un sensor de nivel de basura en un contenedor o un sensor que reporte a tiempo real los espacios libres de estacionamiento en la calle. En este contexto surgen lo que se llaman ciudades inteligentes, las cuales utilizan el potencial de la tecnología para generar un desarrollo sustentable y una mejora en la calidad de vida de sus habitantes.

En el mundo existen ciudades como Wipperfürth (Alemania), Santiago de Chile o Calbuco (Chile) que ya utilizan luminarias inteligentes como parte del alumbrado público [1] [2]. La Intendencia de Montevideo (IM) pretende realizar la transición a luminarias LED en el alumbrado público, procurando, entre otras mejoras, una reducción de consumo similar a la que obtuvieron las ciudades previamente mencionadas. Tanto es así que en la actualidad existe una licitación expedida por la

IM para este fin (Licitación Pública 670/2017 [3]). En el Anexo 2 de la misma, la licitación expresa que se requiere un sistema digital de control para las luminarias, llamado Sistema de Gestión de Luminarias Inteligentes. En resumen, cada luminaria instalada deberá tener un dispositivo llamado *controlador* conectado a ella que se comunique de forma inalámbrica con una estación de trabajo remota de la IM. Esta comunicación servirá para que un usuario de la estación de trabajo remota envíe comandos de encendido, apagado o nivel de dimerización al controlador. A su vez, el controlador utilizará esta comunicación para transmitir información, como por ejemplo mediciones eléctricas, siempre que sea solicitado por la estación de trabajo remota. La licitación nombra a esta infraestructura de comunicación "red de comunicaciones", y explicita en el apartado 4 del Anexo 2 que debe estar implementada con alguna de las siguientes tecnologías: LoRaWAN sobre LoRa, Telensa UNB u IEEE 802.15.4 en cualquiera de sus versiones. Por otra parte, las luminarias a instalar deben tener un driver capaz de soportar el protocolo 1-10V o 0-10V. A través de alguno de estos dos protocolos el controlador deberá dimerizar la luminaria.

Lo anterior entra en el marco del desarrollo de Ciudad Inteligente de Montevideo. Actualmente, dentro de la dependencia Desarrollo Sostenible e Inteligente de la IM, existe una subdependencia llamada Tecnología para Ciudades Inteligentes [4]. Entre sus cometidos se encuentran:

> *" Articular diversas iniciativas internas y externas para la integración en una plataforma tecnológica de ciudades inteligentes."*
>
> *"Establecer pautas para la elaboración del Plan de Desarrollo Tecnológico, incluyendo objetivos, prioridades, asignación de recursos y oportunidad de realización, entre otros, en coordinación con la Direccion General del Departamento y la Gerencia de Tecnología de la Información."* [1]

Ambas citas muestran la importancia que la IM le está dando a transformar Montevideo en una Ciudad Inteligente.

El presente proyecto de fin de carrera tiene como objetivo el diseño de un controlador de luminaria inteligente. Los requerimientos que se describen en la Licitación Pública 670/2017 servirán de base para definir los objetivos particulares del actual proyecto. Sin embargo, no se pretende que el producto que se obtenga sea capaz de cumplir con todo lo que se especifica en dicho documento.

En el mercado existen controladores que cumplen con lo pedido en la licitación, como por ejemplo el producto Owlet IoT que ofrece la compañía Schréder [5]. Sin embargo, su diseño de hardware y su software es propietario, por lo cual es de interés diseñar y construir un controlador con funciones similares pero con diseño "open-source".

[1] https://montevideo.gub.uy/institucional/dependencias/tecnologia-para-ciudades-inteligentes

1.2. Objetivo general

El objetivo del proyecto es diseñar y construir un controlador de luminaria para alumbrado público. Este debe ser capaz de comandar la luminaria utilizando el protocolo 1-10V. A su vez, debe poder comunicarse de forma inalámbrica con una estación de trabajo remota (llamada Servidor Central de aquí en adelante) para recibir comandos de control y reportar información del estado de la luminaria. Esta comunicación debe estar implementada con la tecnología LoRaWAN sobre LoRa. Adicionalmente, se pretende diseñar el sistema de manera tal que sea posible agregar sensores en el futuro.

Este controlador tiene como fin permitir al Servidor Central configurar las luminarias del alumbrado público para que se ajusten a las necesidades específicas del momento del día, procurando maximizar el ahorro energético.

1.3. Alcance

Se abarcarán las siguientes funcionalidades:

- Comandar la luminaria para encenderla y apagarla. A su vez, utilizar el protocolo 1-10V para dimerizarla.
- Recibir las instrucciones de encendido, apagado y dimerizado de forma inalámbrica desde el Servidor Central y ejecutarlas. La tecnología de comunicación a utilizar es LoRaWAN sobre LoRa.
- Diseñar un reloj astronómico, el cual calcule la hora de amanecer y atardecer para una fecha y lugar geográfico dado. Esta información sirve para saber si a determinada hora es de día o de noche.
- Controlar la luminaria en función de la información generada por el reloj astronómico. Es decir, se pretende apagar la luminaria cuando el reloj astronómico indica que es de día y encenderla cuando es de noche.
- Controlar la luminaria a partir de instrucciones programadas de antemano, las cuales generen un plan horario que deberá ser utilizado por el controlador de forma autónoma, incluso si pierde comunicación. Estas instrucciones provienen del Servidor Central.
- Controlar la luminaria en función de la información recibida de un circuito fotosensible (o fotocélula). Es decir, se pretende apagar la luminaria cuando el circuito fotosensible indica que hay luz ambiente suficiente y encenderla cuando no la hay.
- Diseñar el controlador basándose en diferentes modos de funcionamiento, los cuales se diferencian en la fuente de información que se usa en la toma de decisiones: instrucciones directas desde el Servidor Central, plan horario, reloj astronómico o fotocélula.

- Diseñar un sistema de alimentación que satisfaga las necesidades energéticas de todos los componentes del controlador.
- Diseñar un sistema capaz de tomar medidas eléctricas (por ejemplo consumo de potencia) de la luminaria.
- Instalar mecánicamente el controlador en una caja de protección adecuada, cumpliendo con el grado de protección IP 66 [6].
- Sentar las bases para el acople de otros sensores al sistema.

Queda por fuera del alcance:

- Implementar el controlador en una placa única.
- Realizar una aplicación web o móvil para el control de la luminaria.
- Acoplar otros sensores al sistema.

1.4. Trabajos Relacionados

Durante el transcurso del presente proyecto, el equipo realizó otros trabajos en el contexto de las asignaturas Sistemas Embebidos para Tiempo Real y Redes de Sensores Inalámbricos, dictadas en la Facultad de Ingeniería de la Universidad de la República. En el caso de los trabajos finales de estas dos asignaturas, se eligieron propuestas relacionadas con las temáticas tratadas en el presente proyecto de fin de carrera. En esta sección se presentan superficialmente dichos trabajos. Se puede ver la documentación de ambos trabajos en el repositorio de git del proyecto [2].

1.4.1. Proyecto de Sistemas Embebidos para Tiempo Real

En el primer semestre del año 2019 el grupo realizó un trabajo titulado "Control de una Luminaria LED" en el marco de la asignatura *Sistemas Embebidos para Tiempo Real*. En este trabajo se desarrolló una primera versión del software del controlador. Esta versión logra el control de la luminaria basándose en diferentes modos de funcionamiento, según la fuente de información a partir de la cual se toman las decisiones. Estas son: instrucciones directas del usuario, plan horario, reloj astronómico y fotocélula. Sin embargo, el controlador obtenido de este trabajo lograba dimerizar la luminaria pero no podía apagarla.

Esta versión del controlador cuenta con una interfaz de comunicación capaz de recibir comandos desde una PC vía UART (del inglés Universal Asyncronous Receiver Transmitter). Dichos comandos son cadenas de caracteres ASCII (del inglés American Standard Code for Information Interchange) que son interpretadas por el controlador para actuar en consecuencia. Por ejemplo, si el usuario envía el comando "SET ON" el controlador enciende la luminaria al 100 % de su intensidad lumínica. A esta interfaz de comunicación se le adjudica el nombre *sisem_shell*.

[2] `https://gitlab.com/TomasArrivillaga/columint`

Esta primera versión del software fue desarrollada con el objetivo de servir como base a la versión final a utilizar para el proyecto de fin de carrera. A su vez, la interfaz sisem_shell se utilizó en etapas avanzadas del proyecto para realizar pruebas del funcionamiento del controlador. Esto es debido a la simplicidad que supone dicha comunicación en comparación con la comunicación inalámbrica vía LoRaWAN (la cual debe manejar la versión final del controlador).

1.4.2. Proyecto de Redes de Sensores Inalámbricos

En el segundo semestre del año 2019 el grupo realizó un trabajo titulado "Infraestructura de comunicación para red de sensores utilizando LoRaWAN" como proyecto de la asignatura *Redes de Sensores Inalámbricos*. En dicho trabajo, se creó un sistema "prueba de concepto" para evaluar la posibilidad de instalar sensores que usen la infraestructura de comunicación asociada al despliegue de las luminarias con controlador LoRaWAN en la ciudad. Como se mencionó anteriormente, la adición de estos sensores representa un avance en los lineamientos planteados por la dependencia de Desarrollo Sonstenible e Independiente de la IM en el marco del desarrollo de Ciudad Inteligente de Montevideo.

En este trabajo, se propone usar cada controlador de luminaria como raíz de una red IEEE 802.15.4/6lowpan. Los nodos de esta red son sensores de distintas magnitudes de interés, por ejemplo temperatura, sonido, luz o nivel de basura en un contenedor.

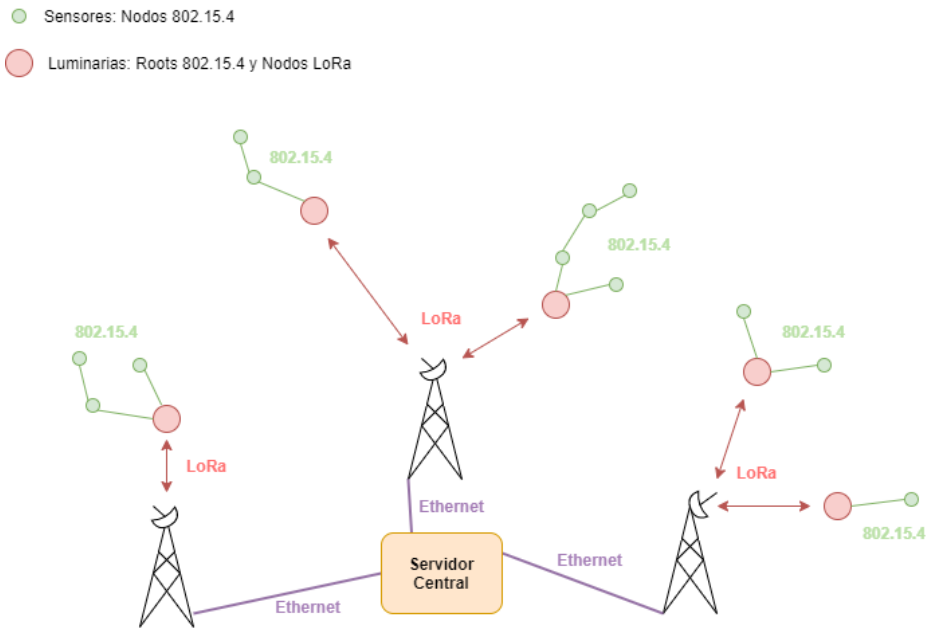


Figura 1.1: Diagrama de la red LoRaWAN con las subredes IEE 802.15.4.

En la figura 1.1 se muestra un esquema donde se pueden ver los controladores (en rojo), y los sensores (en verde). En este trabajo se logra establecer la comunicación Sensor-Controlador-Servidor Central pasando por las dos tecnologías de

comunicación mencionadas. En definitiva, los controladores de luminaria usan la red IEEE 802.15.4/6lowpan para recolectar los datos de los sensores, y luego usan la red LoRaWAN para hacer que los datos lleguen al Servidor Central. Al finalizar el trabajo, se logró realizar con éxito un camino de principio a fin desde la obtención de la magnitud sensada hasta su representación gráfica en el Servidor Central. No obstante, se encontró que la comunicación LoRaWAN que el grupo estableció presentaba una alta tasa de pérdida de paquetes. Este problema en la comunicación LoRaWAN se resolvió para el proyecto de fin de carrera.

1.5. Descripción de los capítulos

- **Capítulo 2: Tecnologías utilizadas en el proyecto**

  En este capítulo se describe la forma de controlar la luminaria a través de su receptáculo ANSI136.41 compatible con el protocolo 1-10V. A su vez, se describen conceptos básicos sobre la tecnología de comunicación inalámbrica LoRaWAN. Este capítulo no contiene información acerca del controlador diseñado en el presente proyecto. No obstante, se recomienda fuertemente la lectura de este capítulo para facilitar la comprensión de algunos aspectos de este proyecto.

- **Capítulo 3: Descripción de la solución**

  En este capítulo se describe brevemente el sistema completo y se da un primer acercamiento al controlador diseñado. El controlador está compuesto por varios módulos hardware, los cuales son descritos de forma somera en este capítulo para luego ser detallados en el capítulo 4. A su vez, este capítulo describe los modos de funcionamiento del controlador así como los comandos que este recibe, enviados desde el Servidor Central.

- **Capítulo 4: Hardware**

  En este capítulo se presentan los componentes utilizados para conformar los módulos hardware, así como la topología de los circuitos que contienen tales componentes. Adicionalmente, se explican las razones por las cuales estos componentes logran cumplir los objetivos planteados para cada módulo. Finalmente se desarrolla sobre la interconexión entre los módulos y las características físicas del controlador. En particular, se explican algunas decisiones de diseño y se muestra el producto final.

- **Capítulo 5: Software embebido**

  En este capítulo se describe el software del controlador. En particular, se listan los módulos y se describen de forma somera, se detalla el bucle principal y se especifica la implementación de los modos de funcionamiento. Además, se describen más en detalle algunos aspectos importantes del software. Estos son: la implementación del reloj astronómico, el manejo de eventos, la codificación de comandos, la comunicación LoRaWAN y la obtención de medidas eléctricas del circuito de medida de consumo.

- **Capítulo 6: Pruebas**

  En este capítulo se describen las pruebas realizadas tanto para los módulos de software como para los módulos hardware, en las diferentes etapas planteadas. Finalmente se adjuntan dos videos que muestran pruebas de funcionamiento del controlador.

- **Capítulo 7: Conclusiones**

  En este capítulo se realiza un análisis general del desarrollo del proyecto. A su vez, se evalúan los resultados obtenidos con respecto a los objetivos definidos en el alcance del proyecto (sección 1.3) y se realiza una conclusión final. Por último, se plantean mejoras al controlador y ampliaciones al proyecto que se podrían realizar en trabajos futuros.

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 2

Tecnologías utilizadas en el proyecto

2.1. Introducción

Las luminarias a controlar en este proyecto cuentan con un driver 1-10V y un receptáculo ANSI136.41, que hacen posible que el controlador se comunique con la luminaria para encenderla, apagarla o dimerizarla. Por otra parte, el controlador se comunica de forma inalámbrica con el Servidor Central para recibir los comandos de control. En este proyecto se utiliza la tecnología de comunicación inalámbrica LoRaWAN, la cual es idónea para redes del largo alcance y baja potencia.

En este capítulo se describe la forma de controlar la luminaria a través de su receptáculo ANSI136.41 compatible con el protocolo 1-10V. A su vez, se describen conceptos básicos sobre la tecnología de comunicación inalámbrica LoRaWAN. Este capítulo no contiene información acerca del controlador diseñado en el presente proyecto. No obstante, se recomienda fuertemente la lectura de este capítulo para facilitar la comprensión de algunos aspectos del proyecto.

2.2. Luminarias LED y sus drivers

Las luminarias LED para alumbrado público cuentan con un driver, el cual es el encargado de transformar la corriente de alterna a continua. Las luminarias a controlar en este proyecto son las Taurus-36 de la empresa Kanglight [7]. Estas luminarias contienen un driver compatible con el protocolo 1-10V, el cual se basa en la variación de la corriente continua que le llega a la luminaria con el objetivo de regular el flujo lumínico. Este driver necesita alimentación L (línea) y N (Neutro), más otras dos señales analógicas de entrada "+" y "-". La diferencia de tensión entre estas señales analógicas (que llamaremos Vdim) determinará la intensidad de luz de la luminaria. A modo de ejemplo, en la Fig. 2.1 se muestra la relación entre Vdim y la intensidad de luz de la luminaria para los drivers Xitanium de Phillips [8], presentes en las Taurus-36. Si se coloca como entrada un Vdim de entre 0 V y 1 V se mantendrá el nivel de luz mínimo en la luminaria, el cual corresponde al 10 % de intensidad de luz. Si se coloca un voltaje mayor o igual a 8 V se mantendrá el nivel de luz máximo en la luminaria (100 %). En los casos intermedios la relación

es lineal. Para poder usar 1-10V, las luminarias con drivers compatibles tienen receptáculos donde las señales "+" y "-" están accesibles. Es importante destacar que con este protocolo se puede manejar el nivel de luz mientras la luminaria está encendida, pero no se puede apagar la luminaria. Para lograr esto hace falta cortar la alimentación de la luminaria con un sistema auxiliar.

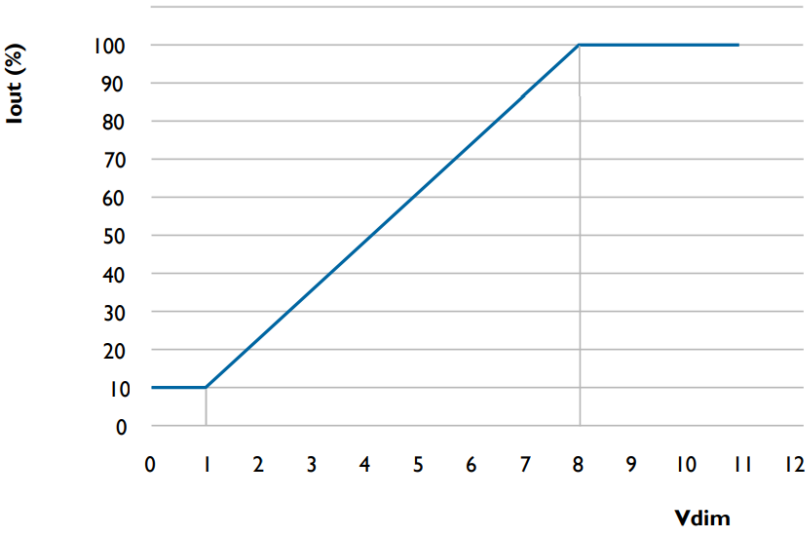


Figura 2.1: Relación entre voltaje y nivel de luz en el protocolo 1-10V.
Fuente: hoja de datos de drivers Xitanium (Phillips).

El Instituto Nacional Estadounidense de Estándares (ANSI, por sus siglas en inglés) en su estándar ANSI C136.41 describe métodos de control de nivel de luz para luminarias de alumbrado público. Estandariza, entre otras cosas, los receptáculos de estas luminarias. La Fig. 2.2 muestra un esquema de estos receptáculos estandarizados. Se puede apreciar que el receptáculo tiene 5 conexiones: a las ya nombradas "+", "-", L y N se le suma L1 (Carga). Cuando L1 está cortocircuitada a L, se alimenta el driver de la luminaria, lo cual a su vez enciende la luminaria. En cualquier otro caso, el driver no tiene alimentación, por lo que la luminaria se apaga. Por lo tanto, si al protocolo 1-10V se le suma el control de la señal L1 se tiene control total sobre la luminaria (encendido, apagado y dimerizado).

En la figura 2.3 se muestra una foto del receptáculo compatible ANSIC136.41 de la luminaria Taurus-36. En la foto se aprecian los 5 contactos del esquema de la figura 2.2. También se observan dos contactos extra cuyo uso previsto es comunicarse con sensores dentro de la luminaria en caso de tenerlos. Estos contactos extra no se utilizan en este proyecto. Por último, a modo de ejemplo, en la figura 2.4 se muestra a una foto de la luminaria Taurus-36, con un controlador conectado en su receptáculo.

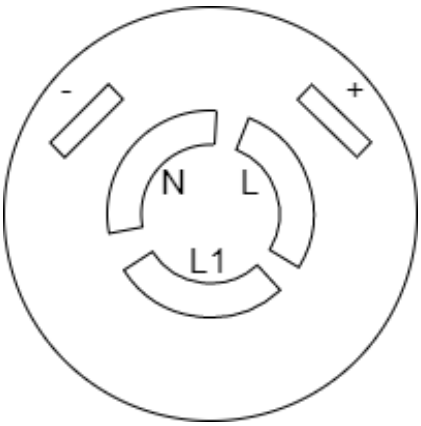


Figura 2.2: Esquema de receptáculo para luminarias viales según el estándar ANSI C136.41

Figura 2.3: Receptáculo ANSIC136.41 de la luminaria Taurus-36.

Figura 2.4: Luminaria Taurus-36 con un controlador conectado en su receptáculo.
Fuente: www.kanglight.com

2.3. Tecnología de comunicación LoRaWAN

Para obtener un alumbrado público inteligente es importante contar con una tecnología de comunicación que permita acceder al control inalámbrico de cada luminaria. En el presente proyecto se implementa una infraestructura de comunicación con la tecnología LoRaWAN. Esta tecnología es una de las opciones que se explicita en la Licitación Pública 670/2017 de la IM para la implementación de la "red de comunicaciones". Se eligió LoRaWAN sobre las otras opciones (Telensa UNB e IEEE 802.15.4) dado que fue la recomendada personalmente por funcionarios de la Unidad Técnica de Alumbrado Público de la IM. Para facilitar la comprensión de algunos aspectos de este proyecto, es importante conocer los conceptos básicos asociados a esta tecnología de comunicación. Entre estos se encuentran los distintos tipos de dispositivos de la red y las limitaciones que conlleva el uso de esta tecnología. Para información más detallada se recomienda consultar la página web de LoRa Alliance, una asociación dedicada a desarrollar y promover la tecnología LoRaWAN [9].

LoRa (Long Range) es una técnica de modulación que se destaca por permitir el desarrollo de redes de largo alcance y baja potencia (LPWAN, Low-Power Wide-Area Network), patentado por la empresa Semtech [10]. Sin embargo, la desventaja de esta técnica es el bajo ancho de banda que puede manejar, que se traduce en poca información que puede transmitir por mensaje. Por su parte, LoRaWAN es un protocolo de red para redes de largo alcance y baja potencia, ideado para trabajar con LoRa. Las frecuencias de comunicación que esta tecnología maneja pertenecen a bandas de uso libre.

La comunicación en un entorno LoRaWAN sobre LoRa es una red en forma de estrella. Se definen cuatro tipos de elementos en esta topología: los dispositivos end-point, los gateways, los servidores de red y las aplicaciones. En la figura 2.5 se muestra un diagrama de la topología. Los dispositivos end-point se comunican con los gateways de forma inalámbrica utilizando modulación LoRa, mientras que los gateways se comunican con los servidores de red a través de internet (TCP/IP, del inglés Transmission Control Protocol/Internet Protocol). Los servidores de red son enrutadores para los paquetes que se intercambian las aplicaciones y los gateways. A modo de ejemplo, un dispositivo end-point puede ser un sensor de temperatura que se comunique de forma inalámbrica vía LoRa y la aplicación puede ser un software que obtiene las medidas y realiza estadísticas sobre ellas. En este contexto, el gateway y el servidor de red son los elementos que permiten la comunicación entre el dispositivo end-point y la aplicación. Las comunicaciones del dispositivo end-point hacia la aplicación se llaman uplink, mientras que las de la aplicación hacia el dispositivo end-point se llaman downlink. Es relevante mencionar que en el presente proyecto se utiliza la banda de frecuencia de 915 MHz. Dicha banda se divide en 64 canales para uplinks y 8 canales para downlinks. En la sección inmediatamente posterior se desarrollará sobre cada uno de los tipos de elementos de la red LoRaWAN.

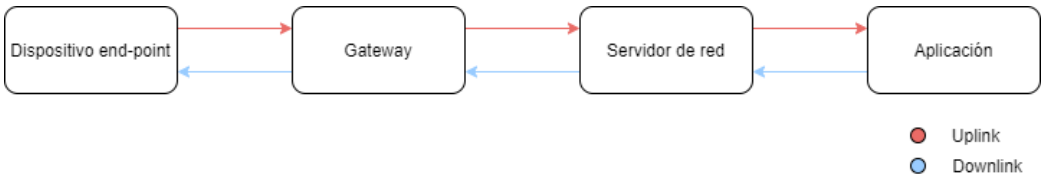


Figura 2.5: Diagrama de comunicación en la tecnología LoRaWAN sobre LoRa.

2.3.1. Tipos de elementos

Dispositivos end-point

Los dispositivos end-point envían y reciben información hacia y desde el gateway. Existen muchos tipos de dispositivos end-point, ya que el único requisito es que tenga un transceiver (del inglés transmitter-receiver) para comunicarse mediante LoRa con un gateway. Usualmente son sensores que reportan cierta magnitud o dispositivos que reciben comandos y actúan en base a ellos. Existen tres clases diferentes de dispositivos end-point, según la forma de comunicación entre éstos y el gateway. Éstas son clase A, clase B y clase C. En la figura 2.6 se pueden apreciar diagramas que representan los posibles intercambios de mensajes en estas las clases. A continuación se explica sobre cada clase.

- **Clase A**: la clase A es la clase más básica y la que implica el menor consumo de energía. Permite la comunicación uplink en cualquier momento de forma asincrónica. Luego de un uplink, se habilitan dos ventanas de tiempo para aceptar comunicaciones downlink. Estas son: Rx_1 que ocurre un tiempo configurable T_1 después del uplink y Rx_2 que ocurre un tiempo configurable T_2 después del uplink.

- **Clase B**: la clase B cumple con las características de la clase A y añade una característica sincrónica. Se sincroniza la red para permitir ventanas donde el dispositivo puede recibir mensajes de downlink sin la necesidad de iniciar un uplink. Es decir, además de las ventanas Rx_1 y Rx_2 que se habilitan luego de un uplink, también existen ventanas Rx_b que se repiten con período fijo T_b. Esto implica que no es necesario que los dispositivos end-point usen la red LoRaWAN para enviar uplinks "dummy" con el objetivo verificar si existe algún downlink pendiente para ellos. Los dispositivos clase B consumen más que los clase A dado que la radio se enciende periódicamente para mantenerse sincronizado con la red.

- **Clase C**: la clase C hereda las características de la clase A y permite que el dispositivo reciba downlinks en cualquier momento, sin necesidad de establecer una ventana de recepción y asegurando latencia mínima. Es la clase que implica el mayor consumo energético, dado que el transceiver del dispositivo end-point debe estar siempre encendido.

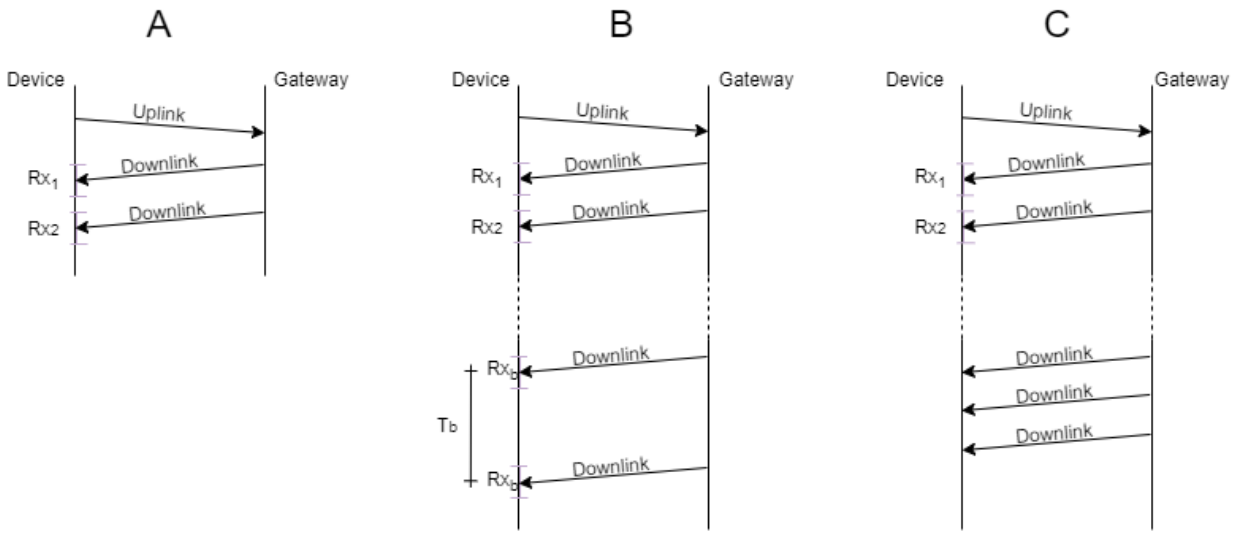


Figura 2.6: Diagrama de posibles intercambios de mensajes en LoRaWAN, según clases. Clase A en la izquierda, clase B en el medio y clase C en la derecha

Gateways

Los gateways son los encargados de recibir la información de los dispositivos end-point de forma inalámbrica utilizando la modulación LoRa y enviarla al servidor de red a través de internet. También hacen el camino inverso: reciben los mensajes del servidor de red a través de internet y los envían a los dispositivos end-point.

Servidores de Red

Los servidores de red establecen la comunicación entre las aplicaciones y los gateways. En caso de un mensaje de uplink recibido por un gateway, el servidor de red es quien envía este paquete a la aplicación correspondiente. En redes con más de un gateway, es posible que el uplink de un end-point llegue a varios gateways; en este caso el servidor de red elimina los mensajes duplicados recibidos por distintos gateways. Si una aplicación quisiera comunicarse con un dispositivo end-point, el servidor de red es quien se encarga de hacer llegar el mensaje de downlink al gateway correspondiente. El gateway correspondiente es aquel con el cual las comunicaciones del dispositivo end-point tienen mejor calidad de señal.

Es importante destacar que los servidores de red no tienen acceso al contenido de los mensajes (payload) ya que este se encuentra encriptado, sino que solo tienen acceso a la información acerca de la procedencia y el destino del mensaje.

Aplicaciones

Las aplicaciones reciben los mensajes de uplink y son los generadores de mensajes de downlink. Estas pueden decodificar y acceder a la información de los paquetes, a diferencia de los servidores de red. Están programadas para actuar en

base a la información de los paquetes que recibe. Por ejemplo, puede mostrar los datos en una interfaz gráfica, enviarlos a una base de datos, etc.

2.3.2. Proceso de activación del dispositivo end-point

Para permitir la conexión de los dispositivos end-point con la aplicación se necesita primero que el dispositivo end-point se "active". Esto es que la aplicación reconozca al dispositivo end-point para que permita la comunicación. La activación se puede realizar de dos maneras: OTAA (del inglés Over The Air Activation) y ABP (del inglés Activation By Personalization). OTAA requiere tres parámetros: device EUI, application EUI y application key. El device EUI es una identificación del dispositivo end-point. El application EUI y el application key sirven como nombre y contraseña de la aplicación respectivamente. La aplicación debe tener configurado qué device EUI aceptar. Cuando el dispositivo end-point se quiere conectar por primera vez con la aplicación inicia el proceso de activación. Esta transacción la inicia el dispositivo end-point enviando un mensaje de Join Request. Si la aplicación acepta al dispositivo, se envía un mensaje downlink llamado Join Accept, que finaliza el proceso de activación de forma exitosa.

La otra forma para activar un dispositivo en una aplicación es ABP. La principal diferencia entre ABP y OTAA en cuanto al proceso de activación es que OTAA implica que el dispositivo end-point espere el Join Accept de la aplicación. El método ABP no tiene esta confirmación; el dispositivo end-point considera que su proceso de activación se realizó de forma exitosa y comienza a mandar uplinks. Otra diferencia importante es que si se usa ABP la dirección del dispositivo en la red está preconfigurada en el mismo, mientras que si se usa OTAA esta dirección es asignada de forma dinámica por el servidor de red.

En este proyecto se utiliza la plataforma The Things Network (TTN) [11], la cual permite que un usuario cree aplicaciones para su red LoRaWAN y ya tiene integrado un servidor de red. Además, TTN soporta múltiples tipos de gateways y posee mucha información para realizar la conexión entre el servidor de red y los gateways. TTN provee herramientas para cumplir múltiples objetivos como: configurar los gateways para aceptar las conexiones de diferentes dispositivos end-point, validar y decodificar los mensajes provenientes de los gateways, codificar los mensajes de downlink, entre otros. También posee lo que TTN llama integraciones, donde el servidor puede comunicarse con otras aplicaciones como Ubidots o MatLab, para realizar operaciones o mostrar los datos recibidos por el servidor [12] [13]. En este proyecto se utiliza TheThingsNetwork.

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 3

Descripción de la solución

3.1. Introducción

En vista de los requerimientos explicitados en el alcance de proyecto (sección 1.3), se desarrolló una solución que pretende cumplir con todos ellos. Esta solución refiere al diseño de un controlador de luminaria de alumbrado público como parte de un sistema más amplio. Este sistema está conformado por el Servidor Central, una infraestructura de comunicación inalámbrica LoRaWAN, el controlador y una luminaria.

En este capítulo se describe brevemente el sistema completo y se da un primer acercamiento al controlador diseñado. El controlador está compuesto por varios módulos hardware, los cuales son descritos de forma somera en este capítulo para luego ser detallados en el capítulo 4. A su vez, el presente capítulo describe los modos de funcionamiento del controlador así como los comandos que este recibe, enviados desde el Servidor Central.

3.2. Sistema completo

En la figura 3.1 se muestra un diagrama general del sistema completo. La luminaria se conecta a la red eléctrica de potencia 230 V y esta abastece al controlador a través del receptáculo ANSI C136.41. La comunicación entre el controlador y el driver de la luminaria se realiza utilizando el protocolo 1-10V. Dado que el controlador no puede manejar el encendido y apagado de la luminaria a través del protocolo 1-10V, es necesario que lo haga de forma independiente. Para encender la luminaria, el controlador la alimenta cortocircuitando Línea (L) y Carga (L1). Por otra parte, para apagarla el controlador abre este contacto, cortando la alimentación a la luminaria. Este funcionamiento del controlador se simboliza en el diagrama de la figura 3.1 como la señal ON/OFF, aunque dicha señal no es una salida del controlador ni una entrada de la luminaria a nivel de hardware.

Por otra parte, el controlador utiliza la tecnología LoRaWAN para comunicarse con el Servidor Central, tomando el rol de dispositivo end-point. Esta comunicación permite que el controlador reciba comandos desde el Servidor Central y a su vez

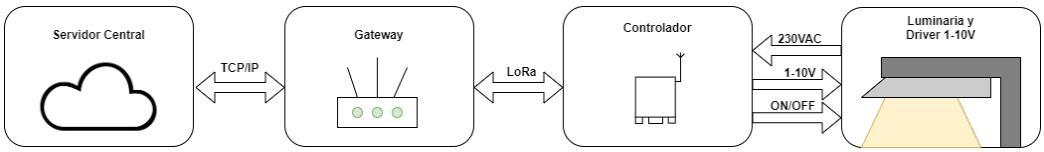


Figura 3.1: Diagrama de bloques del sistema completo.

que envíe mensajes de respuesta cuando corresponda. Como se explicó en la sección 2.3 el controlador se comunica de manera inalámbrica con el gateway, el cual recibe y transmite información vía TCP/IP desde y hacia el Servidor Central.

3.3. Controlador: módulos constitutivos

El controlador está conformado por varios módulos hardware que cumplen distintas funciones. Se puede ver en la figura 3.2 una representación esquemática de los módulos que conforman el controlador, así como las relaciones entre ellos.

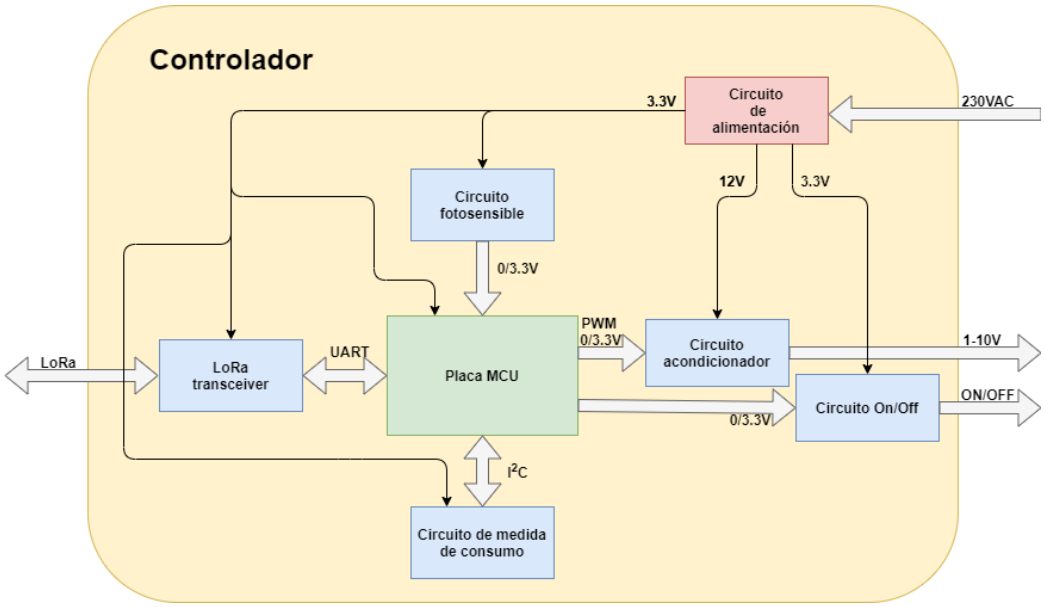


Figura 3.2: Esquema de módulos hardware del controlador.

A continuación se realiza un breve resumen de las funcionalidades que cumple cada módulo. En el capítulo 4 se desarrollará en detalle cada uno.

El circuito de alimentación es el encargado de proveer las tensiones requeridas por los restantes módulos para su funcionamiento. Las tensiones generadas son 3,3 V y 12 V de continua, las que se obtienen a partir de la red de 230VAC.

La placa MCU es la unidad central del controlador. Este módulo es una placa de desarrollo que posee un microcontrolador. Se comunica con el resto de los módulos a

través de diferentes protocolos. Además, es quien se encarga de tomar las decisiones sobre el estado de la luminaria para encenderla, apagarla o dimerizarla.

El circuito fotosensible (o fotocélula) cumple la función de generar una tensión continua entre 0 V y 3,3 V en función de la cantidad de luz que incide sobre él. El MCU es capaz de medir dicha tensión e inferir el nivel de luz ambiente.

El módulo LoRa transceiver es el que comunica al MCU con el Servidor Central. Este módulo se comunica con el MCU a través de un puerto serial UART y con el Servidor Central de manera inalámbrica con la tecnología LoRaWAN sobre LoRa. El transceiver recibe comandos desde el Servidor Central y se los reporta al MCU para su procesamiento. A su vez, es el encargado de enviar los mensajes desde el MCU al Servidor Central, cuando el MCU lo requiere.

El circuito de medida de consumo es el encargado de medir las magnitudes eléctricas del conjunto luminaria-controlador. Estas magnitudes son el voltaje en bornes del driver de la luminaria (entre L y N), corriente que consume, potencias (activa, reactiva y aparente) y factor de potencia. Estas medidas son reportadas al MCU cuando este las solicita. La comunicación entre el MCU y el circuito de medida de consumo se lleva a cabo usando el protocolo serial I^2C.

El circuito acondicionador es el responsable de dar la salida 1-10V hacia el driver de la luminaria. Esta salida es generada a partir de una señal PWM (del inglés Pulse Width Modulation) proveniente del MCU. El valor de tensión de la salida 1-10V depende directamente del ciclo de trabajo de la señal PWM. Este es el único módulo que necesita ser alimentado con una tensión de 12 V.

El circuito On/Off cumple la función de apagar o encender la luminaria cortando o no la alimentación de la misma, conmutando la continuidad entre “L” y “L1”. Realiza esta acción a partir de una señal lógica proveniente del MCU.

3.4. Modos de funcionamiento

El controlador, descrito en la sección anterior, controla la luminaria tomando en consideración diferentes fuentes de información. Estas son:

- Instrucciones directas del Servidor Central
- Plan horario
- Reloj astronómico
- Fotocélula

Se definieron cuatro modos de funcionamiento para el controlador: modo Directo, modo Plan Horario, modo Reloj Astronómico y modo Fotocélula. Cada uno de estos modos centran su operación en una de las fuentes de información mencionadas. Sin embargo, en los modos Plan Horario y Reloj Astronómico es posible configurar el controlador para que incluya información de la fotocélula. A continuación se describe cada modo de funcionamiento:

- Modo Directo (Modo 0):
  En este modo se comanda la luminaria únicamente mediante instrucciones directas del Servidor Central para apagado, dimerizado y encendido.
- Modo Plan Horario (Modo 1):
  En este modo el nivel de dimerización de la luminaria es impuesto por un plan horario previamente configurado desde el Servidor Central. El plan horario se define como una serie de acciones en el día, llamadas *eventos*, que indican a qué hora se debe cambiar el nivel de luz, y cuál es dicho nivel. Para adaptarse a las diferencias en la dinámica de la ciudad entre los días de semana y los fines de semana, el controlador cuenta con dos planes horarios independientes: plan de fin de semana y plan de entre semana. El controlador decide automáticamente el plan a usar según sea día de semana o no.

  Este modo permite incluir información proveniente de la fotocélula en las horas de sol, para mejorar la toma de decisiones en casos de baja luz durante el día (por ejemplo, un día tormentoso). Es importante destacar que en las horas de la noche la fotocélula no influye en la toma de decisiones, por más que se haya configurado el sistema para incluirla.
- Modo Reloj Astronómico (Modo 2):
  En este modo el nivel de dimerización de la luminaria depende de si es de día o de noche, determinado por la hora del amanecer y del atardecer que calcula el reloj astronómico. En particular, se enciende la luminaria al máximo nivel de dimerización 12 minutos antes del atardecer calculado y se apaga 12 minutos después del amanecer calculado. Este margen de seguridad de 12 minutos pretende compensar las imprecisiones de cálculo del reloj astronómico implementado (ver sección 6.2.1). Al igual que en el caso del modo Plan Horario, se da la posibilidad de tener en cuenta a la fotocélula en horas del día.

  El modo Reloj Astronómico funciona como un plan horario con dos eventos: uno que representa el amanecer y el otro el atardecer. El controlador actualiza este plan horario al iniciar un nuevo día, con las nuevas horas calculadas. Por otra parte, este plan no se puede configurar desde el Servidor Central.
- Modo Fotocélula (Modo 3):
  En este modo se comanda la luminaria según el nivel de luz que recibe la fotocélula. Cuando el controlador detecta que no hay suficiente luz ambiente enciende la luminaria, y cuando detecta que hay suficiente luz ambiente la apaga.

3.5. Lista de comandos

El controlador recibe comandos en cualquier modo de funcionamiento. Estos comandos llegan en primer lugar al transceiver LoRa de forma inalámbrica desde el Servidor Central, para luego ser enviados a través de la UART hacia la placa MCU.

A continuación se listan los distintos comandos que se pueden usar para controlar y monitorear la luminaria. Los comandos se presentan clasificados por modo, aunque todos los comandos son recibidos en todos los modos. Es decir, el criterio de clasificación es en cuál modo tiene efecto el comando. [1]

Comandos generales:

- SET MODE M: cambia el modo de funcionamiento al modo M. M puede valer 0, 1, 2, 3 representando el modo Directo, Plan Horario, Reloj Astronómico y Fotocélula respectivamente. En caso de pasar al modo Reloj Astronómico o Plan Horario, el nivel de dimerización a partir del cambio será el indicado por el evento previo más cercano del plan horario correspondiente. Si el plan horario correspondiente no tiene eventos, el controlador enciende la luminaria.

- GET MODE: el controlador responde con el modo actual, si se está usando la fotocélula o no y el nivel actual de dimerización.

- SET MOMENT DD/MM/YYYY HH:MM:SS : se modifica el momento actual (fecha y hora) al especificado en el comando.

- GET MOMENT : el controlador responde con la fecha y hora actual.

- USE PHC ON : habilita el uso de la fotocélula en los modos Reloj Astronómico y Plan Horario.

- USE PHC OFF : deshabilita el uso de la fotocélula en los modos Reloj Astronómico y Plan Horario.

- DIR EVENT DT: el controlador responde con todos los eventos cargados actualmente en el plan horario indicado por DT. DT puede ser WD (del inglés Weekday) para el plan de entre semana, WE (del inglés Weekend) para el plan de fin de semana o SE (del inglés Sun Events) para el plan del modo Reloj Astronómico. Los eventos están identificados con un "número identificador de evento" (ID). Para cada evento se detalla el ID, la hora y el nivel de luz.

- GET VOLT: el controlador responde con la tensión en bornes del driver de la luminaria (entre L y N).

- GET CURR: el controlador responde con la corriente consumida por el conjunto luminaria-controlador.

- GET ACTPWR: el controlador responde con la potencia activa consumida por el conjunto luminaria-controlador.

[1] Los comandos que recibe la placa MCU están codificados. Los detalles de la codificación en binario se explican en la sección 5.7.

- GET REAPWR: el controlador responde con potencia reactiva consumida por el conjunto luminaria-controlador.
- GET APPPWR: el controlador responde con la potencia aparente consumida por el conjunto luminaria-controlador.
- GET COSPHI: el controlador responde con el factor de potencia del conjunto luminaria-controlador.

Modo directo:

- SET ON: enciende la luminaria al nivel 100 % de dimerización. Es decir, la salida 1-10V será 10V.
- SET OFF: apaga la luminaria.
- SET DIM XX: fija la salida 1-10V del controlador al nivel ZZ, donde ZZ es el múltiplo de 5 más cercano a XX (el cual debe ser un número del 0 al 100). Esto quiere decir que si XX es igual a 23, la salida 1-10V del controlador será de nivel 25 %, es decir 2,5 V. Si XX es menor a 10 se apaga la luminaria (equivalente a SET OFF). Si XX es igual a 100, la salida 1-10V será 10 V (equivalente a SET ON).

Al recibir cualquiera de los comandos de modo Directo en un modo que no sea el modo Directo, el controlador efectúa el comando y automáticamente pasa al modo Directo.

Modo plan horario:

- ADD EVENT DT HH:MM XX: agrega un evento en el plan horario indicado por DT, donde DT es WD o WE, en la hora indicada por HH:MM y con nivel de dimerización XX.
- RES EVENT DT: elimina todos los eventos del plan horario correspondiente a WD o WE según sea DT.
- DEL EVENT DT ID: elimina el evento cuyo número identificador es ID del plan horario WD o WE, según sea DT.

Capítulo 4

Hardware

4.1. Introducción

El controlador posee diversos módulos hardware interconectados entre sí, cada uno con funciones determinadas, tal como se explicó en el capítulo 3. Para tener suficiente espacio para albergar a todos ellos el controlador cuenta con tres placas de circuitos impresos (PCBs, del inglés Printed Circuit Board). Los PCBs fueron diseñados utilizando el programa Eagle [14], y fueron fabricados por la empresa PCBWay [15].

En este capítulo se presentan los componentes utilizados para conformar los módulos hardware, así como la topología de los circuitos que contienen tales componentes. Adicionalmente, se explican las razones por las cuales estos componentes logran cumplir los objetivos planteados para cada módulo. Finalmente se desarrolla sobre la interconexión entre los módulos y las características físicas del controlador. En particular, se explican algunas decisiones de diseño y se muestra el producto final.

4.2. Módulos hardware

4.2.1. Circuito de alimentación

El circuito de alimentación es quien se encarga de proveer al resto de los módulos hardware con la corriente y voltaje necesario para su correcto funcionamiento. Entrega dos niveles de tensión continua: 12 V y 3,3 V. El circuito consta de dos fuentes de tensión conectadas en cascada: una que se alimenta de 230 VAC y su salida es de 12 VDC y una segunda fuente que se alimenta de 12 VDC y tiene una salida de 3,3 VDC. En la figura 4.1 se puede ver un esquema del circuito de alimentación. La fuente que entrega 12 VDC es la MPM-10-12 del fabricante MeanWell, mientras que la fuente que entrega 3,3 V es la SPBW06F-03 del mismo fabricante. Se pueden ver fotos de ellas en la figura 4.2.

La fuente MPM-10-12 puede entregar hasta 10 W y la fuente SPBW06F-03 puede entregar hasta 5 W. Estas potencias son suficientes para cumplir con las

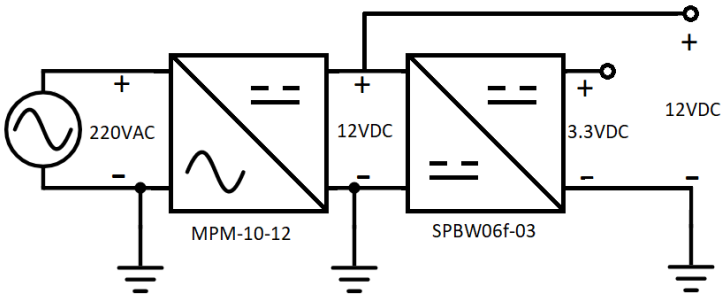
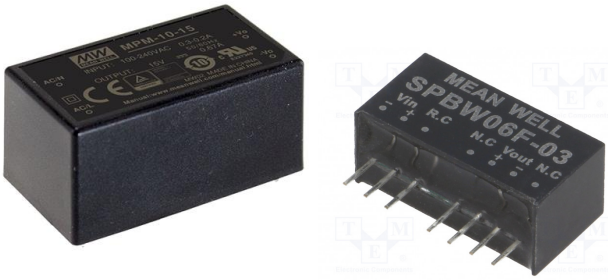


Figura 4.1: Diagrama del circuito de alimentación.



(a) MPM-10-12 (b) SPBW06F-03

Figura 4.2: Fuentes utilizadas para el circuito de alimentación.
Fuente: www.digikey.com

necesidades del controlador. En el apéndice A se muestra que los módulos a los que alimenta la fuente SPBW06F-03 no exceden los 1,13 W y los módulos a los que alimenta la fuente MPM-10-12 no exceden los 4,14 W. Además de cumplir con los requisitos energéticos del controlador, estas fuentes son idóneas por su tamaño relativamente pequeño.

4.2.2. Placa MCU

La placa MCU utilizada tiene el microcontrolador MSP432P401R del fabricante Texas Instruments (TI) [16]. Esta elección está motivada en la experiencia que el grupo tiene con los microcontroladores de TI. En particular, en la asignatura Sistemas Embebidos el grupo trabajó con la placa de desarrollo MSP-EXP430G2ET ya que es el que pone a disposición el grupo docente [17]. Esta placa contiene el microcontrolador MSP430G2553 [18]. Se comenzó el desarrollo del presente proyecto en la placa mencionada pero el microcontrolador no satisfizo las necesidades del proyecto. Esto es debido a que el cálculo del amanecer y atardecer para un día basándose en la posición geográfica (latitud y longitud) es computacionalmente costoso; se precisan numerosas multiplicaciones, divisiones y

funciones trigonométricas (ver sección 5.5). Cuando se intentó realizar estos cálculos se observó que la memoria del microcontrolador no era suficiente; la falta de multiplicador hardware del mismo hace que un programa que realice operaciones como las mencionadas sea muy grande. Por lo tanto, se optó por trabajar con la placa de desarrollo MSP-EXP432P401R ya que es la que fue proporcionada por el tutor del proyecto [19]. Esta placa posee el microcontrolador MSP432P401R, el cual tiene multiplicador hardware y cumple con los requerimientos de nuestro software.

La placa de desarrollo MSP-EXP432P401R brindó la posibilidad de desarrollar la primera versión del firmware. Sin embargo, es necesario que la placa a usar en el controlador sea del menor tamaño posible, así el ensamble posterior de todos los módulos hardware permite que el controlador entre en la caja de protección IP66 compatible con el conector ANSI C136.41. Por lo tanto, se decidió utilizar como placa MCU la placa de desarrollo MSP432 Clicker del fabricante MikroElektronika [20]. La misma se puede observar en la figura 4.3. La figura 4.4 muestra la diferencia en tamaño de las dos placas mencionadas.

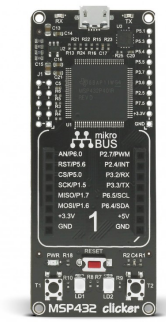


Figura 4.3: MSP432 Clicker de MikroElektronika.
Fuente: www.mikroe.com

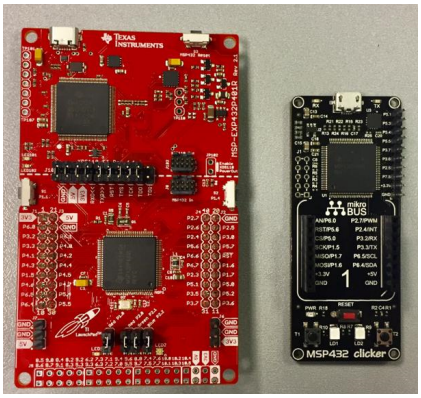


Figura 4.4: MSP-EXP432P401R (placa roja) y MSP432 Clicker (placa negra).

Además de ser más compacta, la placa MSP432 Clicker puede ser conectada fácilmente con otras placas del mismo fabricante mediante su conexión MikroBus. Esto es importante puesto que MikroElektronika ofrece un módulo LoRa transceiver con esta conexión llamado LoRa 2 Click [21]. La facilidad que supone conectar ambas placas y trabajar con ellas fue la razón principal que llevó al grupo a decidirse por ellas.

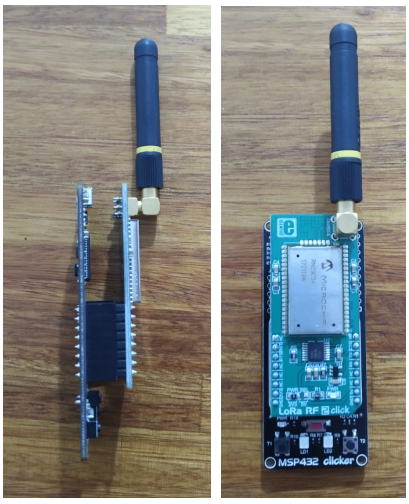
4.2.3. LoRa transceiver

Se utilizó como LoRa transceiver el Lora 2 Click de MikroElektronika (figura 4.5), como se explicó en la sección 4.2.2. La comunicación entre el MSP432 Clicker y el LoRa 2 Click es muy simple: se conecta el LoRa 2 Click en la conexión "MikroBus" que se encuentra en el MSP432 Clicker. En la figura 4.6 se puede ver la conexión de estas dos placas. Es importante destacar que la comunicación entre estas dos placas es vía UART.

Figura 4.5: Transceiver Lora 2 Click
Fuente: www.mikroe.com

El LoRa 2 Click es una placa que tiene como objetivo facilitar el desarrollo de aplicaciones que usen el chip RN2903 del fabricante Microchip [22]. El RN2903, que es el componente principal del Lora 2 Click, está conformado por un transceiver SX1276 del fabricante Semtech y un microcontrolador PIC18LF46K22 del fabricante Microchip [23] [24]. El microcontrolador del RN2903 cumple la función de brindar la posibilidad de comunicarse mediante comandos UART con el transceiver SX1276, facilitando su uso. Cuando se usa el RN2903 como parte de un dispositivo end-point permite que el dispositivo funcione en clase A o clase C[1]. Para el presente proyecto el RN2903 fue configurado para que el controlador sea un dispositivo end-point clase A.

[1]Por defecto el RN2903 funciona en clase A. Para poder utilizarlo en clase C, es necesario actualizar su firmware al menos a la versión 1.05.



(a) De costado (b) De frente

Figura 4.6: Fotos del acople de MSP432 Clicker y LoRa 2 Click

Los comandos UART que utiliza el RN2903 se dividen en tres grupos: *sys*, *mac* y *radio*. Los comandos *sys* están vinculados a las acciones sobre el sistema mismo. Por ejemplo, permiten consultar la versión del software del chip o reiniciarlo. Los comandos *mac* inciden sobre la red LoRaWAN a la que se conectó el dispositivo. Por ejemplo, permiten activarse en la red, definir ventanas de recepción (en caso de trabajar en clase A) o transmitir un mensaje. Los comandos *radio* son aquellos con los que se puede modificar los parámetros de funcionamiento del SX1276. Por ejemplo, permiten modificar la potencia de transmisión y la banda de frecuencia a utilizar, entre otros. Para más información sobre el uso del chip RN2903 y la sintaxis de los comandos, referirse a la documentación del integrado [22].

4.2.4. Circuito Fotosensible

El circuito fotosensible traduce el nivel de luz en el ambiente a un nivel de tensión de entre 0 y 3,3 V utilizando un fotorresistor (R_{PHC}). La figura 4.7 muestra un diagrama esquemático de este circuito. El mismo es un divisor de tensión, en el cual uno de sus resistores es R_{PHC}. El fotorresistor es un resistor que varía su valor de resistencia dependiendo de la luz que incide en él. Al ser un divisor de tensión, el valor de V_{out} dependerá de R_{PHC} como muestra la siguiente ecuación:

$$V_{out}(R_{PHC}) = 3{,}3V \times \left(\frac{R}{R + R_{PHC}}\right) \tag{4.1}$$

El fotorresistor usado es el NSL-4960 del fabricante Advanced Photonix [25] (ver figura 4.8). De la hoja de datos del componente se conoce que la resistencia del fotorresistor es de 500 Ω cuando inciden 100 ftc (pie candela), y es de $17k$ Ω cuando incide 1 ftc. Cuando está completamente oscuro la resistencia vale 1000 kΩ.

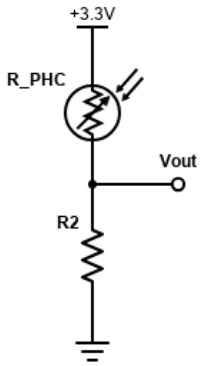


Figura 4.7: Esquemático del circuito fotosensible.

Figura 4.8: NSL-4960 del fabricante Advanced Photonix .
Fuente: www.digikey.com

Entonces, el valor resistivo de este componente disminuye al aumentar la luz incidente en él. Por lo anterior, la relación cualitativa entre V_{out} y R_{PHC} es como se muestra a continuación:

$$Luz \uparrow \longrightarrow R_{phc} \downarrow \longrightarrow V_{out} \uparrow \tag{4.2}$$

$$Luz \downarrow \longrightarrow R_{phc} \uparrow \longrightarrow V_{out} \downarrow \tag{4.3}$$

El valor de resistencia del resistor que acompaña al fotorresistor (R_2 en el diagrama) es 10 kΩ. Utilizando esta resistencia y la ecuación (4.1) se pueden obtener los valores de V_{out} en los escenarios proporcionados por la hoja de datos del componente NSL-4960. Estos valores se pueden apreciar en la tabla 4.1. Utilizando un ADC (del inglés Analog-to-Digital Converter) con una referencia de 3,3 V, se puede obtener este valor de tensión desde el MCU y a partir de ello determinar el nivel de luz ambiente y por ende la necesidad o no de encender la luminaria.

ftc	$R_{phc}[k\Omega]$	$V_{out}[V]$
∼0	1000	0.03
1	17	1.22
100	0.5	3.15

Tabla 4.1: Tabla de valores de V_{out} en funcion a la luz incidente, tomando los valores de R_{phc} de su documentación.

4.2.5. Circuito de acondicionamiento

El circuito de acondicionamiento, utilizado para controlar el dimerizado de la luminaria, toma como entrada una señal PWM de entre 0V y 3,3 V con ciclo de trabajo variable, y la transforma en un nivel de DC entre 0 V y 10 V. Este nivel será el utilizado como salida 1-10V del controlador. Para lograr lo anterior se utiliza un filtro pasabajos, el cual deja pasar el nivel de continua de la misma y filtra todos los armónicos de la señal PWM (incluyendo la frecuencia fundamental). El nivel de DC será 0 V cuando el ciclo de trabajo sea 0 (la señal es siempre cero), y será 3,3 V cuando el ciclo de trabajo sea 100 (la señal es siempre 3,3 V). En la figura 4.9 se representa gráficamente una señal PWM de periodo T, ancho de pulso L y amplitud A. En este caso, el ciclo de trabajo es L/T=D.

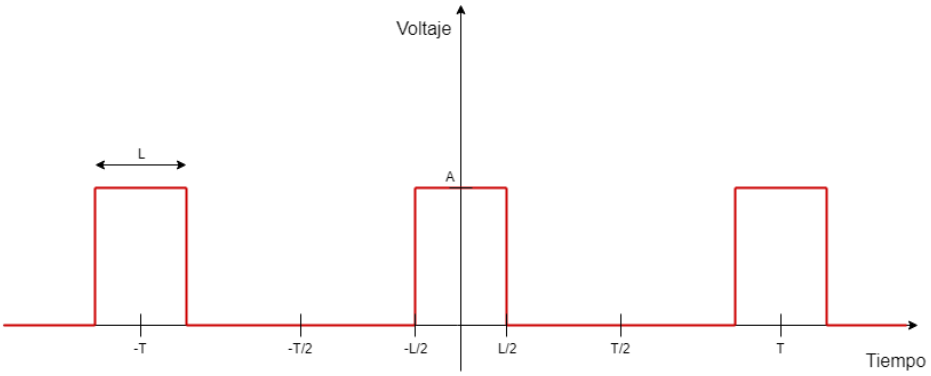


Figura 4.9: Señal PWM de periodo T, ancho de pulso L y amplitud A.

La relación entre el ciclo de trabajo de la onda y su componente continua es de proporcionalidad directa, dado que para una onda como la de la figura 4.9 sus coeficientes de Fourier son:

$$C_n = \frac{AL}{T} sinc(nL/T). \tag{4.4}$$

El valor de continua de la señal es (considerando que $sinc(0) = 1$):

$$C_0 = \frac{AL}{T} = AD. \tag{4.5}$$

La salida PWM del MCU tiene amplitud A = 3,3 V, por lo que la máxima tensión a la salida del pasabajos es 3,3 V. Para obtener una salida de 10 V para un

ciclo de trabajo de 100 % es necesaria una etapa de amplificación luego del filtrado, con factor de amplificación $\frac{10}{3,3} \approx 3,03$.

El circuito de acondicionamiento consta entonces de dos etapas:

Etapa de filtrado: esta etapa consta de un filtro pasabajos de primer orden, en particular un filtro RC. El criterio usado para definir la frecuencia de corte del filtro es que f_{-3dB} esté al menos dos décadas por debajo de la frecuencia fundamental del PWM. De esta forma todos los armónicos del PWM son atenuados al menos 40 dB. La frecuencia fundamental del PWM generado por el MCU es 1,638 kHz (ver sección 5.2). Por lo tanto, la restricción antes mencionada es que la máxima frecuencia de corte sea 16,4 Hz. Teniendo en cuenta que la frecuencia f_{-3dB} de un filtro RC es $\frac{1}{2\pi RC}$, los valores de capacidad y resistencia deben cumplir:

$$\frac{1}{2\pi RC} < 16{,}4Hz$$

Los valores utilizados son $R = 57{,}6$ kΩ y C = 10 μF resultando en una frecuencia de corte $f_{-3dB} = 0{,}276$ Hz. Para que la etapa de filtrado funcione como se espera, es necesario que la impedancia vista hacia la segunda etapa sea elevada.

Etapa de amplificación: Esta etapa consiste en un circuito amplificador simple basado en un amplificador operacional; en particular una configuración no inversora con factor de amplificación $G \approx 3$V/V.

$$G = \left(1 + \frac{R_2}{R_1}\right) = 3$$

Los valores de resistencia utilizados son $R_1 = 5$ kΩ y $R_2 = 10$ kΩ. Se muestra en la figura 4.10 un esquema del circuito de acondicionamiento. Al aplicar estas dos etapas, la relación entre ciclo de trabajo del PWM y V_{out} (salida 1-10V) es lineal. Esto quiere decir que, si se considera un operacional ideal, la salida 1-10V será en realidad un nivel de DC de entre 0 V y 10 V. Lo anterior no es un problema ya que según el protocolo 1-10V las tensiones entre 0 V y 1 V son equivalentes. El operacional a utilizar será alimentado con las señales 12 V y GND, por lo que es necesario que sea un operacional de suministro único (Single Supply) para su correcto funcionamiento. El operacional utilizado es el MC34071 del fabricante Motorola [26]. Cabe destacar que la impedancia de entrada de esta etapa es muy elevada (entrada de un operacional), lo cual cumple con el requisito impuesto por la primera etapa.

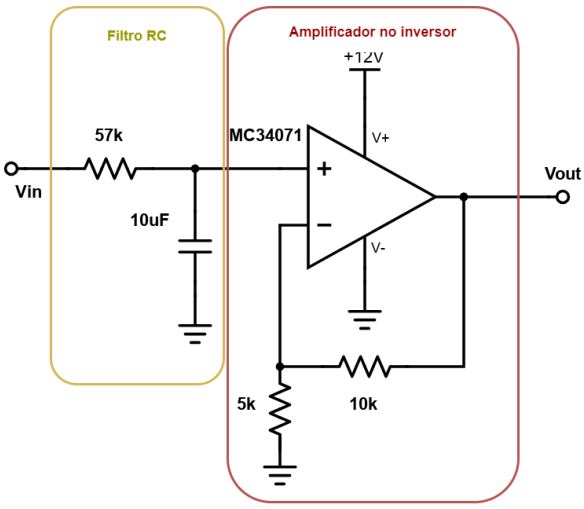


Figura 4.10: Esquemático del circuito de acondicionamiento.

4.2.6. Circuito On/Off

El circuito On/Off permite encender y apagar la luminaria, cortocircuitando o no la señales "L1" y "L" mediante el uso de un relé. La necesidad del circuito On/Off surge del hecho que el protocolo 1-10V permite dimerizar la luminaria pero no maneja el encendido y apagado de la misma, tal cual fuera explicado en la sección 2.2. Para esto el controlador cuenta con el circuito On/Off, cuyo esquemático se presenta en la figura 4.11.

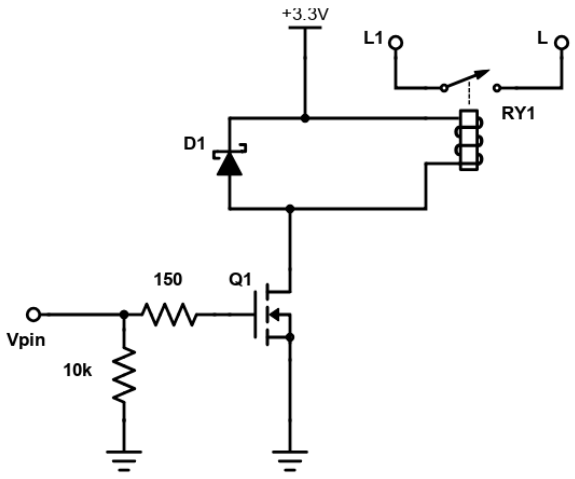


Figura 4.11: Esquemático del circuito On/Off.

El relé RY1 es el OZ-SS-103LM1,200 del fabricante TE Connectivity (figura 4.12). Este relé es adecuado para la aplicación, dado que puede manejar un voltaje de bobina de 3,3 V. Este requisito deriva de que el MCU, quien funciona con 3,3 V,

es quien controla este relé mediante un pin de salida.

Figura 4.12: Imagen del relé OZ-SS-103LM1,200.
Fuente: www.digikey.com

El relé presenta una resistencia de bobina de 17 Ω. A su vez, el relé opera cuando tiene entre los bornes de su bobina voltajes mayores a 2,25 V y corta con voltajes menores a 0,15 V. Por lo tanto, en caso de operación el relé requiere un mínimo de 132 mA ($\frac{2,25V}{17\Omega}$). El MCU utilizado puede entregar un máximo de 20 mA por pin por lo que no puede por sí solo satisfacer las necesidades de corriente del relé. Por lo tanto, el MCU no puede manejar directamente el relé. Es por este motivo que el circuito On/Off incluye un transistor MOSFET que es usado por el MCU como "llave". De este modo, el relé se alimenta de la fuente de 3,3 V a través del transistor y no de un pin del MCU. Cuando Vpin vale 3,3 V, la corriente circula entre source y drain energizando el bobinado del relé y consecuentemente cortocircuitando "L" y "L1". Cuando Vpin es 0 V, el transistor entra en corte, y la corriente entre source y drain es cero desenergizando la bobina. Cuando esto ocurre, se abre el contacto de "L" y "L1". El transistor elegido es el DMN65D8L-7 del fabricante Diodes Incorporated [27]. La elección de este MOSFET en particular está respaldada por las simulaciones que se encuentran en la sección 6.3.5. Estas muestran que la corriente circulará cuando Vpin sea 3,3 V y y el transistor cortará cuando Vpin sea 0 V.

Por otra parte, al cortar el paso de corriente por el bobinado del relé se produce un pico de tensión que puede danãr al transistor. Para evitar esto, se incluye un diodo Schottky (D1 en la figura 4.11). La función de este diodo es proporcionar un camino a la corriente cuando la tensión en bornes del bobinado empieza a elevarse, mitigando el potencial pico de tensión. El diodo Schottky elegido es el BAT54HT1 del fabricante ON Semiconductor [28].

Cuando el MCU está iniciando (tras un reset por ejemplo) la tensión en Vpin podría estar indeterminada, lo cual a su vez podría provocar aperturas y cierres indeseados en el relé. Como medida preventiva se incluye una resistencia de 10 kΩ a modo de "pull down", lo cual garantiza que la tensión en el gate se mantiene en cero durante el inicio del MCU.

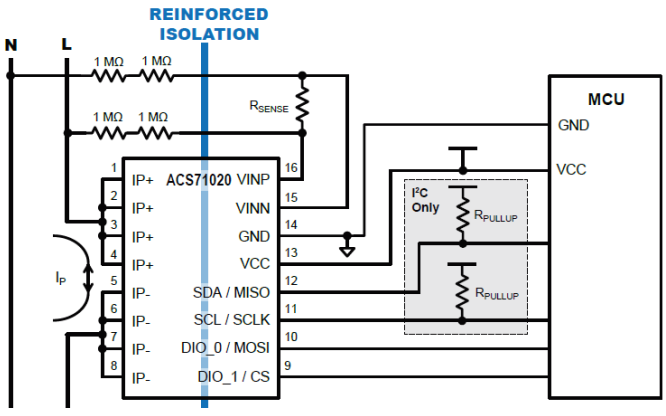
4.2.7. Circuito de medida de consumo

El circuito de medida de consumo está compuesto por un PMIC (del inglés Power Monitoring Integrated Circuit) y un circuito resistivo de step-down. El PMIC seleccionado es el ACS71020KMABTR-030B3-I2C (llamado ACS71020 de aquí en más) de la compañía Allegro Microsystems [29]. Se puede apreciar una foto del mismo en la figura 4.13(a). El MCU se comunica con el PMIC a través del protocolo I^2C. La figura 4.13(b) muestra un diagrama del circuito de medida de consumo y su conexión con el MCU. El diagrama 4.13(b) se tomó directamente de la documentación del chip, o sea que el circuito implementado es el que sugiere el fabricante en dicha documentación. El ACS71020 proporciona datos para cualquier dispositivo alimentado por una fuente de alterna, sobre las magnitudes eléctricas:

- Tensión RMS.
- Corriente RMS.
- Potencia activa.
- Potencia reactiva.
- Potencia aparente.
- Factor de potencia.

Para lograrlo, este chip toma medidas de tensión entre sus pines VINP y VINN (bornes de R_{SENSE}) y corriente que pasa entre IP+ e IP-. A partir del muestreo de tensión y corriente, el ACS71020 calcula las magnitudes eléctricas que reporta.

(a) ACS71020.
Fuente: www.digikey.com



(b) Esquemático del circuito de medida de consumo
Fuente: hoja de datos ACS71020

Figura 4.13: Foto del chip ACS71020 y diagrama del circuito de medida de consumo utilizado.

La función del circuito de step-down es convertir la tensión alterna 230 V (entre Línea y Neutro) a una tensión alterna que no exceda los 275 mV entre los pines VINP y VINN del circuito integrado. Esta restricción es indicada por

el fabricante en la documentación del chip y es importante para elegir un valor adecuado para R_{SENSE}. Por otra parte, existe otro documento de Allegro que es el manual de usuario del kit de prueba ASEK71020, donde se proporciona una tabla para seleccionar el valor de R_{SENSE} en función de la tensión que se desea medir (tabla 4.2). Dado que se desea medir tensiones en el orden de 230 V, se utiliza 2,2 kΩ como valor de R_{SENSE}.

Vrms (v)	Vpico (V)	R Max (Ω)	R_{SENSE} Recomendado (kΩ)
50	70,7	14194	13
100	141,4	7085	5,6
110	155,54	6440	5,1
120	169,68	5902	4,7
150	212,1	4720	3,9
220	311,08	3217	2,2
240	339,36	2949	1,8

Tabla 4.2: Tabla de los valores de R_{SENSE} recomendados por el fabricante.

El circuito integrado ACS71020 es capaz de medir voltaje (vrms), corriente (irms), potencia activa (pactive), potencia reactiva (pimag), potencia aparente (papparent) y factor de potencia (pfactor). Los valores correspondientes a estas medidas se encuentran en registros del chip que están disponibles para su lectura. Cada registro es de 32 bits y tiene una dirección interna asociada. En la figura 4.14 se ve un diagrama del contenido de los registros de interés para este proyecto. Es importante destacar que además de los registros para lectura existen registros de configuración que pueden ser escritos. Por ejemplo, escribiendo el registro 0x1F se puede modificar la dirección I^2C del ACS71020 (Slave Address).

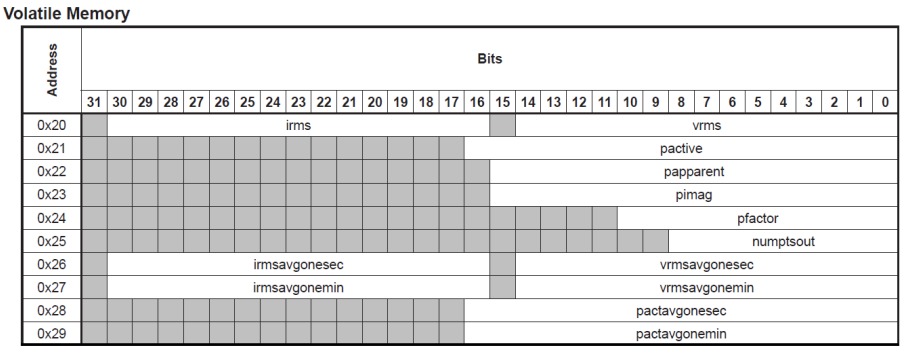


Figura 4.14: Registros internos del ACS71020 a ser leídos en el proyecto.
Fuente: hoja de datos ACS71020 (Página 28)

La figura 4.15 es un diagrama que muestra el intercambio de mensajes en el bus I^2C al realizar una escritura. El MCU comienza el proceso escribiendo en la dirección del ACS71020 (Slave Address), la dirección del registro del chip que se

desea escribir. Luego el MCU inicia otra escritura a la dirección del chip, quien recibirá los datos y los almacenará en el registro que corresponde.

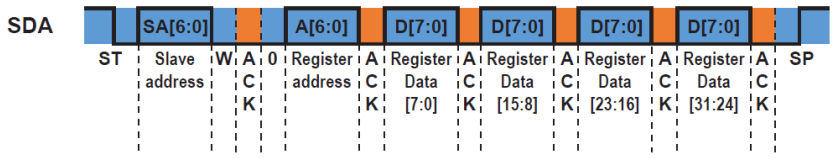


Figura 4.15: Diagrama de tiempos de una escritura a un registro del ACS71020.
Fuente: hoja de datos ACS71020

Luego de un reset el chip adquiere una slave address según el voltaje que haya en los pines DIO_0 y DIO_1 en ese momento[2]. Sin embargo, es posible adjudicar al chip una slave address fija escribiendo en el registro 0x1F. Para el presente proyecto, la dirección fue fijada en 0x45 usando este procedimiento.

Por otra parte, la figura 4.16 muestra el intercambio de mensajes en el bus I^2C al realizar una lectura. Al igual que en la escritura, el MCU comienza el proceso escribiendo en la dirección del ACS71020 (Slave Address), la dirección del registro del chip que se desea leer. Luego el MCU inicia una lectura a la dirección del chip, quien responderá con los datos almacenados en el registro interno que se desea leer.

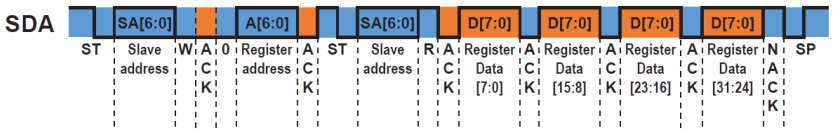


Figura 4.16: Diagrama de tiempos de una lectura a un registro del ACS71020.
Fuente: hoja de datos ACS71020

Los datos que se obtienen al realizar una lectura a un registro están representados en punto fijo, y en caso de que la magnitud pueda ser negativa es representada en punto fijo y complemento a dos. Cada magnitud presenta una cantidad diferente de bits fraccionales, por lo que luego de realizada una lectura es necesario operar con los datos obtenidos para tenerlos representados en punto flotante.

4.3. Interconexión entre módulos y características físicas del controlador

4.3.1. Diseño

Existen ciertos parámetros de diseño a respetar a la hora de diseñar los PCBs para el controlador. En primer lugar, como se explicó en la sección 2.2, los re-

[2]En la página 27 de la documentación del ACS71020 se explicita cuáles son estas direcciones en función del voltaje de los pines mencionados.

ceptáctulos de luminaria de alumbrado público están estandarizados en el documento ANSI C136.41. El diseño del controlador debe tomar en consideración este dato para la interfaz de comunicación con el driver de la luminaria. Por otra parte, la placa necesita poder trabajar con 230 VAC, ya que se utilizan los 230 VAC de la red eléctrica como entrada al circuito de alimentación. A su vez, el circuito On/Off debe cortar la alimentación a la luminaria si se desea apagarla. Por último, uno de los puntos del alcance del proyecto establece que el controlador debe cumplir con la norma de protección IP66. Por lo tanto, los componentes electrónicos deben estar dentro de un contenedor que los proteja frente al polvo y a chorros muy potentes de agua.

El controlador cuenta con una base con un conector compatible con el estándar ANSI C136.41. La base es 2213871-2 del fabricante TE Connectivity y se puede apreciar en la figura 4.17 (a). Adicionalmente, el controlador tiene una tapa de protección para sus componente electrónicos que también cumple con dicho estándar. Esta tapa es 1-2306130-1 de TE Connectivity (figura 4.17 (b)).

(a) Base 2213871-2 (b) Tapa 1-2306130-1

Figura 4.17: Tapa y base del controlador, de la compania TE Connectivity
Fuente: www.digikey.com

La figura 4.18 muestra una ilustración del controlador, sin tapa ni componentes electrónicos. Los PCBs que contienen los componentes electrónicos son circulares y tienen un diámetro ligeramente menor al de la base del controlador, de modo de maximizar el área disponible en cada PCB.

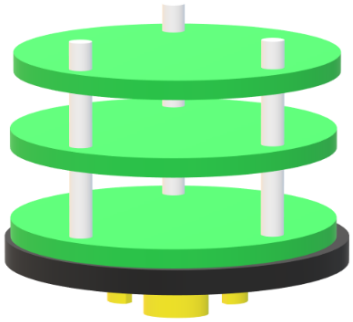


Figura 4.18: Esquema en tres dimensiones del controlador, sin tapa ni componentes electrónicos.

El controlador tiene tres PCBs distintos, que se conectan entre sí usando conectores tipo "flex" para compartir señales. Los módulos hardware fueron ubicados en los distintos PCBs como se muestra en la figura 4.19. Por otra parte, cada PCB posee 3 huecos de aproximadamente 4 mm formando los vértices de un triángulo isósceles. A través de estos huecos se pasan tornillos y cubriendo los tornillos se colocan separadores plásticos de 25 mm de altura.

A los efectos de esta documentación los PCBs se llamarán placa inferior, placa media y placa superior. A continuación se describe brevemente cada una de ellas. En el apéndice B se muestran imágenes de los esquemáticos y layouts de los tres PCBs desarrollados en Eagle. Cabe destacar que la placa inferior tal como se muestra en este capítulo presenta un error de diseño, el cual se descubrió en la etapa de pruebas. Esto generó que la misma deba ser corregida, por lo que se realizaron pequeñas modificaciones sobre el PCB (no se mandó a fabricar un nuevo PCB). Se desarrolla sobre este tema en la sección 6.4.6.

Placa inferior

La placa inferior posee las terminales en las cuales se conectan los contactos de la base ANSI C136.41. Además, contiene el circuito de medida de consumo y el circuito On/Off. En la figura 4.20 se muestran fotos de la placa inferior.

Placa media

La placa media contiene el circuito de alimentación y el circuito de acondicionamiento. En la figura 4.21 se muestran fotos de la placa media.

Placa superior

La placa superior contiene el circuito fotosensible, el MCU y el LoRa Transceiver. En la figura 4.22 se muestran fotos de la placa superior.

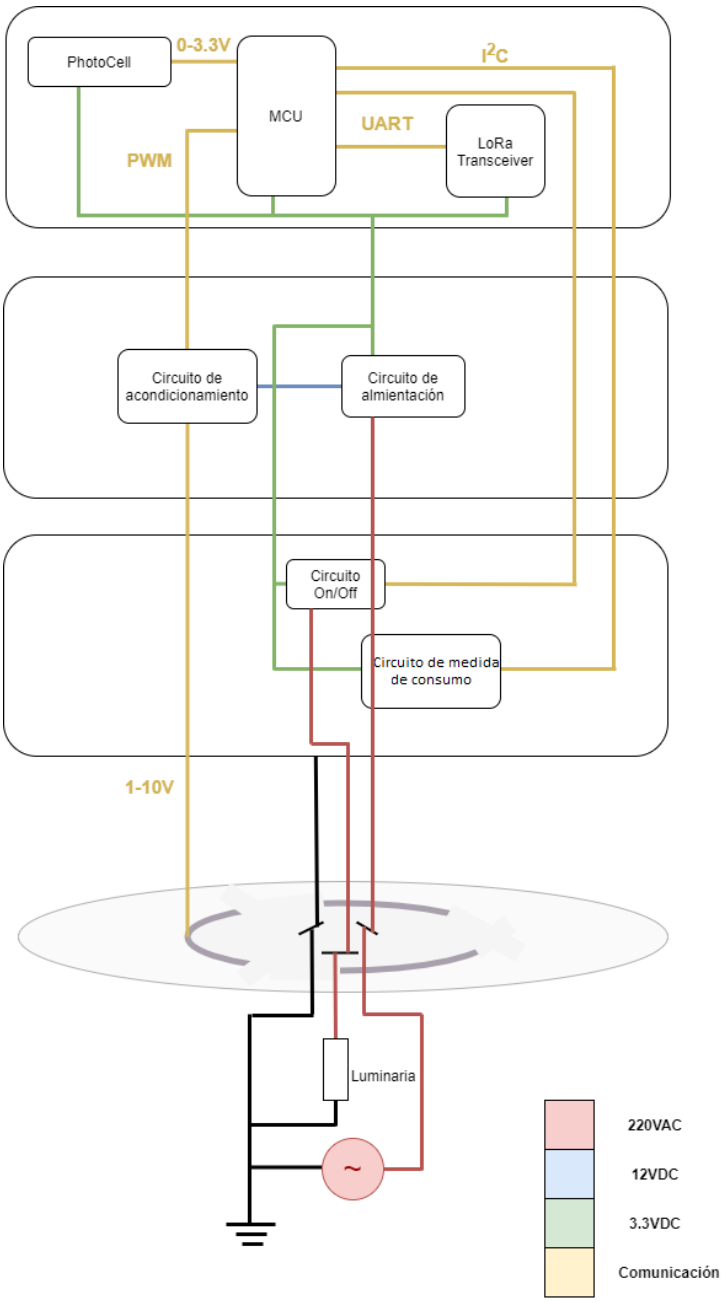


Figura 4.19: Representación de la distribución de módulos hardware.

4.3.2. Producto final

Se muestran en las figuras 4.23, 4.24, 4.25, 4.26, 4.27, 4.29 y 4.28 fotos del producto final del controlador.

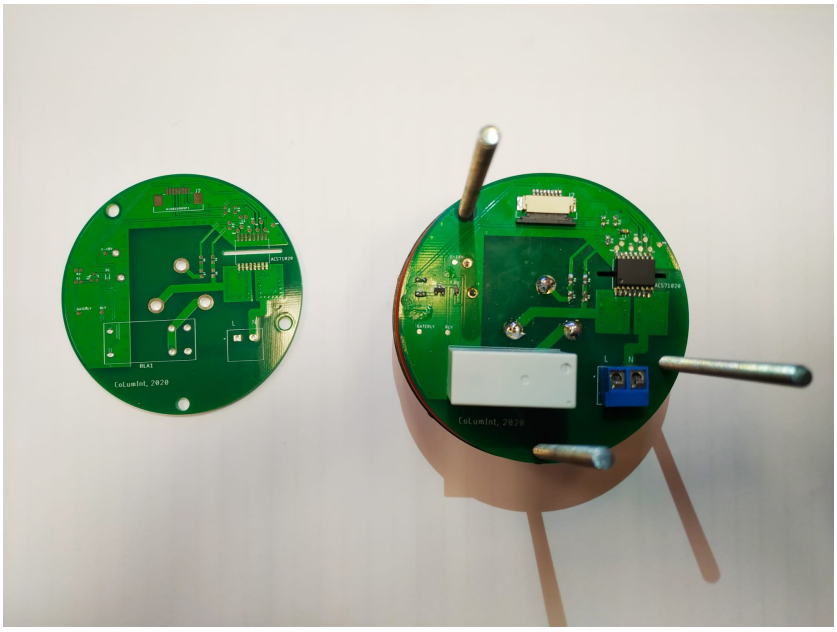


Figura 4.20: Fotos de placa inferior. La de la izquierda no tiene componentes soldados; la de la derecha tiene los componentes soldados y se encuentra acoplada a la base del controlador.

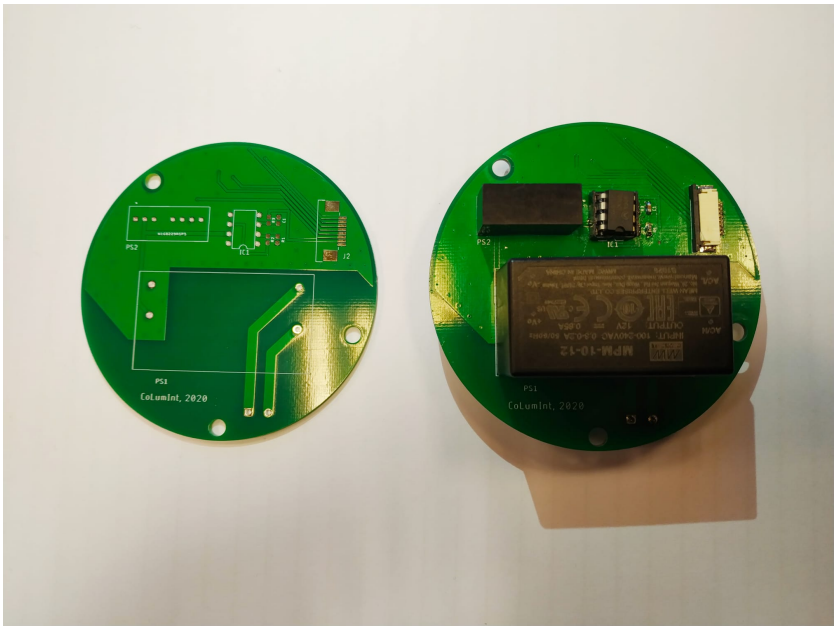


Figura 4.21: Fotos de placa media. La de la izquierda no tiene componentes soldados; la de la derecha tiene los componentes soldados.

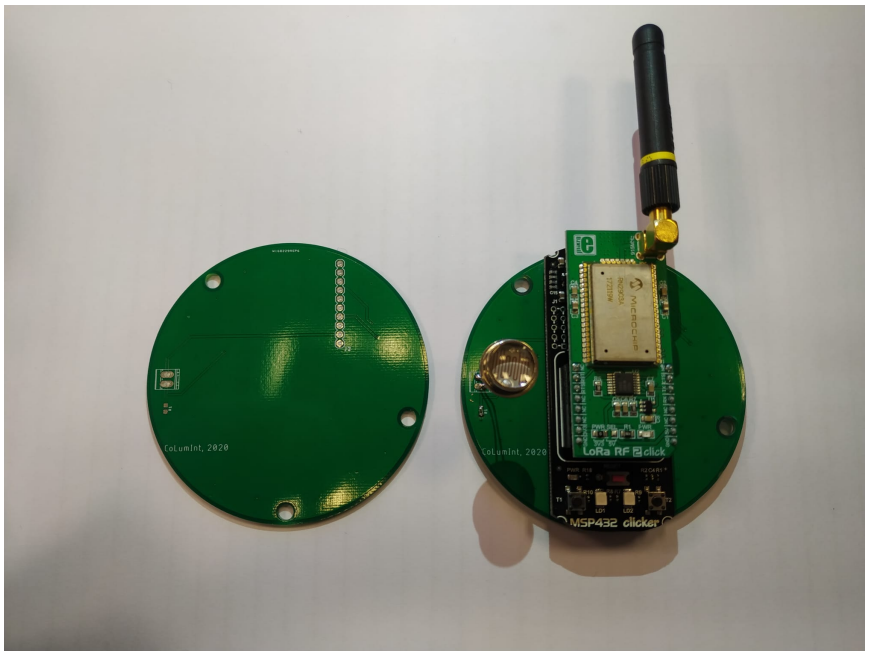


Figura 4.22: Fotos de placa superior. La de la izquierda no tiene componentes soldados; la de la derecha tiene los componentes soldados.

Figura 4.23: Foto del controlador con la tapa a su costado.

Figura 4.24: Foto del controlador sin tapa.

Figura 4.25: Foto del controlador de costado sin tapa, en la cual se aprecian los contactos de la base compatible ANSIC136.41.

Figura 4.26: Foto del controlador (con la tapa colocada).

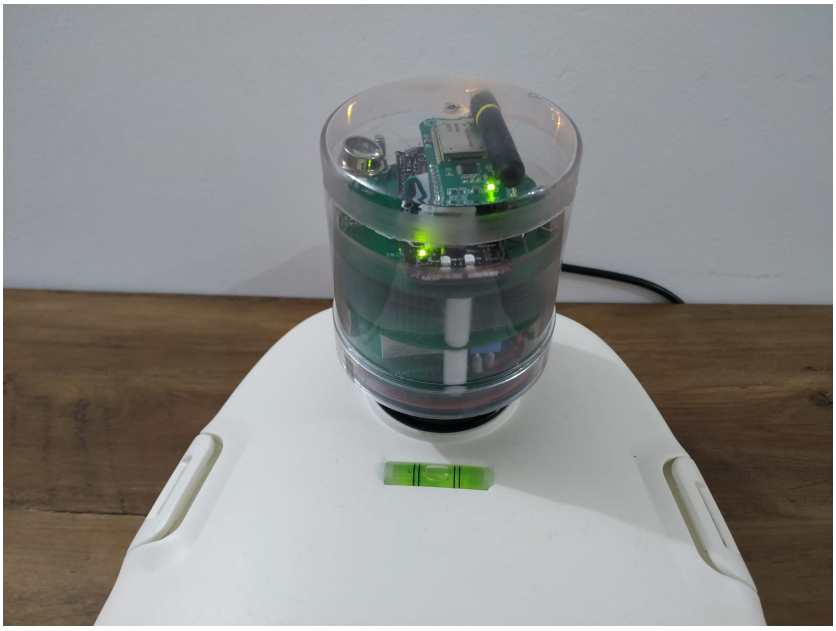


Figura 4.27: Foto del controlador conectado a la luminaria Taurus-36, de cerca.

Figura 4.28: Foto del controlador conectado a la luminaria Taurus-36, de costado.

Figura 4.29: Foto del controlador conectado a la luminaria Taurus-36, desde arriba.

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 5

Software embebido

5.1. Introducción

La unidad central del controlador es el módulo llamado placa MCU, el cual posee al microcontrolador MSP432P401R. El desarrollo del software embebido que permita al MCU cumplir con todos los objetivos explicitados en el alcance es una parte fundamental del proyecto. Este se realizó utilizando el entorno de desarrollo integrado CCS (Code Composer Studio) de Texas Instruments, programando en el lenguaje C [30]. La arquitectura de software utilizada es round robin con interrupciones. El software consta de diversos módulos, algunos de los cuales contienen funciones que levantan las banderas que se evalúan en el bucle principal.

En este capítulo se describe el software del controlador. En particular, se listan los módulos y se describen de forma somera, se detalla el bucle principal y se especifica la implementación de los modos de funcionamiento. Además, se describen más en detalle algunos aspectos importantes del software. Estos son: la implementación del reloj astronómico, el manejo de eventos, la codificación de comandos, la comunicación LoRaWAN y la obtención de medidas eléctricas del circuito de medida de consumo.

El software del proyecto con su correspondiente documentación Doxygen se encuentra en el repositorio de git del proyecto [1].

5.2. Módulos

La figura 5.1 muestra el diagrama de módulos de software del controlador, en el cual también se incluyen los periféricos del MCU utilizados y los módulos hardware con los que estos periféricos se comunican. Los módulos coloreados en azul refieren a software independiente de hardware, los coloreados en rojo son los módulos dependientes de hardware, los bloques en amarillo son periféricos del MCU y los bloques en verde representan los módulos hardware. Cada módulo en el diagrama hace uso de los módulos y bloques inmediatamente debajo de ellos.

[1] `https://gitlab.com/TomasArrivillaga/columint`

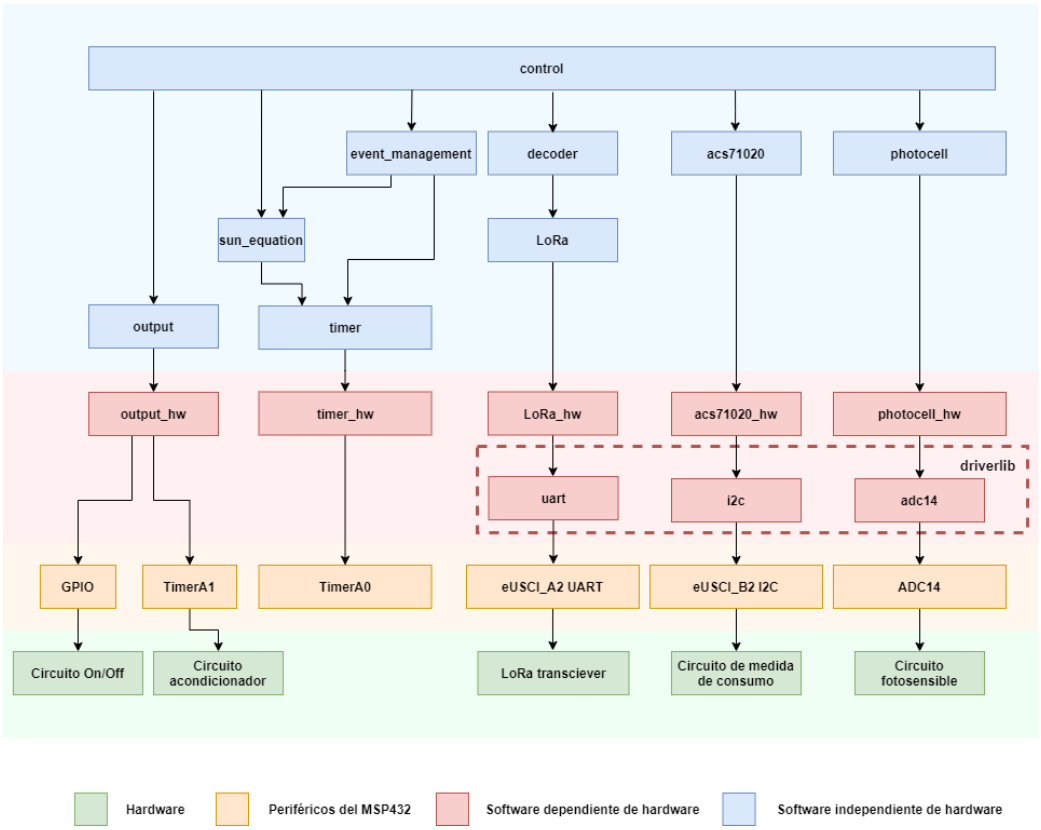


Figura 5.1: Diagrama de módulos de software del controlador.

A continuación se listan los módulos, con una breve descripción de su función.

- **output_hw:** el propósito del módulo output_hw es manejar las salidas del MCU que se conectan al circuito On/Off y al circuito acondicionador. Provee una función que se encarga de configurar el periférico "TimerA1" para generar el PWM de 1,638 kHz de salida hacia el circuito acondicionador. Esta función también configura un GPIO (del inglés General Purpose Input Output) como pin de salida hacia el circuito On/Off. Provee otra función que establece el ciclo de trabajo del PWM generado y fija la salida al circuito On/Off.
- **output:** el propósito del módulo output es el control de las salidas al circuito On/Off y circuito acondicionador. Para esto utiliza las funciones provistas por el módulo output_hw.
- **timer_hw:** el propósito del módulo timer_hw es configurar el periférico "TimerA0" del MCU para que interrumpa cada un segundo. Además, contiene la rutina de interrupción asociada a la misma, la cual llama a una función del módulo timer encargada de incrementar el momento actual (fecha y hora).
- **timer:** el propósito del módulo timer es el manejo del momento actual, a partir de la interrupción configurada por el módulo timer_hw. Provee una

función que incrementa el momento actual en un segundo, y levanta una bandera al comienzo de cada día (new_day), otra al comienzo de cada minuto (new_minute) y otra al comienzo de cada segundo (new_second). Además, este módulo incluye las funciones que operan con variables temporales (por ejemplo, contiene una función que calcula los minutos entre dos horas del día).

- **sun_equation:** el propósito del módulo sun_equation es obtener las horas de amanecer y atardecer de cada día. Provee funciones que realizan los cálculos para obtener dichas horas a partir de una fecha y una latitud dada. Necesita conocer la fecha actual, la cual obtiene del módulo timer.

- **event_management:** el propósito del módulo event_management es el manejo de eventos para los modos Plan Horario y Reloj Astronómico. Provee funciones que operan con los planes horarios, pudiendo eliminar o añadir eventos a los mismos. También provee funciones que operan con eventos, como por ejemplo una función capaz de comparar las horas de dos eventos. Necesita del módulo timer para operar con variables temporales. Para el manejo del plan horario del modo Reloj Astronómico necesita conocer las horas de amanecer y atardecer de la fecha actual, las cuales obtiene del módulo sun_equation.

- **driverlib:** el conjunto de módulos driverlib es provisto por Texas Instruments, que lo define como "un conjunto de API's completamente funcionales que se utilizan para configurar, controlar y manipular los periféricos de hardware de la plataforma MSP432" [31]. En particular se utilizan sus siguientes módulos:

  - uart: provee funciones para configurar una UART.
  - i2c: provee funciones para configurar una comunicación I$\hat{2}$C.
  - adc14: provee funciones para configurar un conversor analógico-digital, el cual convierte una tensión de entre 0 V y una tensión de referencia (en este caso 3,3 V) en un entero sin signo de 14 bits de forma lineal.

- **LoRa_hw:** el propósito del módulo LoRa_hw es configurar la comunicación UART con el LoRa transceiver, haciendo uso del módulo uart del driverlib aplicado al periférico eUSCI_A2 UART. Provee funciones que manejan la recepción y transmisión de mensajes entre el MCU y el LoRa transceiver. Estas funciones levantan una bandera cuando el MCU recibe un mensaje nuevo (lora_rx_complete) y una cuando el MCU está pronto para transmitir un nuevo mensaje (lora_tx_complete).

- **LoRa:** el propósito del módulo LoRa es controlar la comunicación con el LoRa transceiver. Para esto hace uso de las funciones provistas por el módulo LoRa_hw. Genera los mensajes a ser enviados al LoRa transceiver utilizando la sintáxis que se explica en la sección 5.8.1 y obtiene sus respuestas.

- **decoder:** el propósito del módulo decoder es decodificar los comandos recibidos desde el LoRa transceiver, los cuales obtiene haciendo uso del módulo LoRa, y codificar los mensajes a enviar por LoRa. Este módulo provee una única función, la cual decodifica los mensajes y se refiere al módulo control para disparar las acciones posteriores que generan dichos mensajes. También codifica las respuestas a los comandos que da el MCU para ser enviadas por LoRa (payload) y levanta una bandera que indica que hay un mensaje a enviar (lora_begin_tx).
- **photocell_hw:** el propósito del módulo photocell_hw es configurar el periférico ADC14 que se conecta a la salida del circuito fotosensible, haciendo uso de el módulo adc14 del driverlib. Provee una función que obtiene la digitalización de la tensión de salida del circuito fotosensible.
- **photocell:** el propósito del módulo photocell es determinar si hay suficiente luz ambiente o no a partir de las medidas que obtiene del módulo photocell_hw. Provee una función que promedia estas medidas en una ventana de un minuto y compara el resultado con umbrales de luz y oscuridad.
- **acs71020_hw:** el propósito del módulo acs71020_hw es configurar la comunicación I^2C con el circuito de medida de consumo, haciendo uso del módulo i2c del driverlib. Provee funciones que manejan la recepción y transmisión de mensajes entre el MCU y el circuito de medida de consumo.
- **acs71020:** el propósito del módulo acs71020 es controlar la comunicación del MCU con el circuito de medida de consumo, haciendo uso de las funciones provistas por el módulo acs71020_hw. Provee funciones que obtienen las lecturas en binario de los registros de interés del acs71020 y operan con ellas para tener las magnitudes representadas en punto flotante y en las unidades deseadas.
- **control:** el propósito del módulo control es tomar las decisiones de comando de la luminaria, teniendo en cuenta los mensajes recibidos por LoRa, las interrupciones y el modo en que se está trabajando. Provee una función llamada "take_action", la cual maneja los modos de funcionamiento.

5.3. Descripción del bucle principal

La arquitectura de software utilizada es round robin con interrupciones. Los módulos timer, LoRa_hw y decoder levantan banderas que indican que una acción debe ser tomada. El bucle principal consulta cíclicamente el estado de dichas banderas, y en caso de que alguna esté levantada, ejecuta las tareas asociadas a dicha bandera. Se puede apreciar un diagrama de la estructura del bucle principal en la figura 5.2.

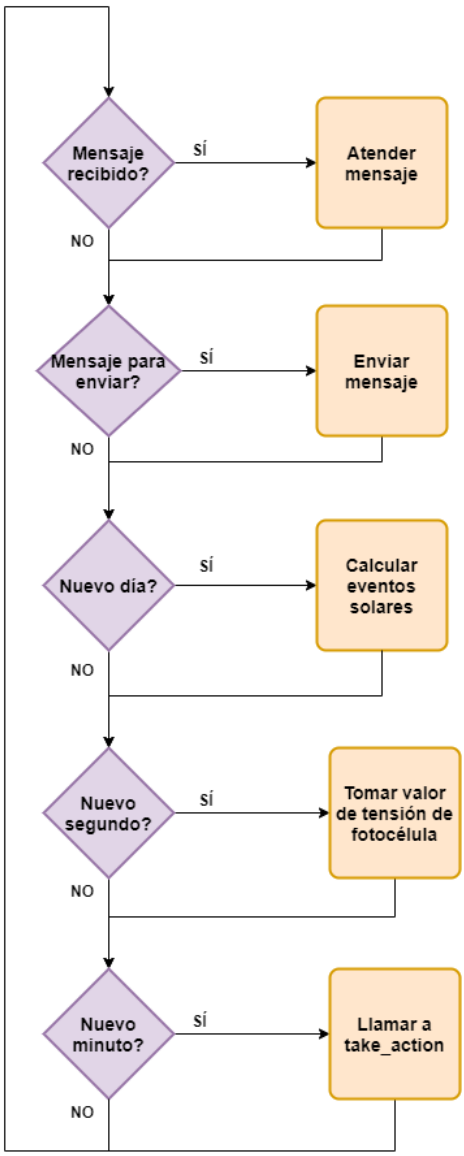


Figura 5.2: Esquema del bucle principal del Round Robin.

A continuación se describen los elementos presentes en este diagrama:

- "Mensaje recibido?" corresponde a la consulta de la bandera lora_rx_complete, la cual se levanta en el módulo LoRa_hw cuando finaliza la recepción de un mensaje. "Atender mensaje" significa que el mensaje será decodificado por la función del módulo decoder, la cual llamará a funciones del módulo control para disparar las acciones posteriores.
- "Mensaje para enviar?" corresponde a la consulta del AND de las banderas lora_tx_complete y lora_begin_tx. La bandera lora_tx_complete se levanta en el módulo LoRa_hw, y significa que se terminó la última transmisión, por lo que se puede enviar otro mensaje. La bandera lora_begin_tx se levanta en

el módulo decoder, y significa que existe un mensaje para enviar. “Enviar mensaje” significa que se utilizarán funciones del módulo LoRa para enviar este mensaje por UART hacia el RN2903.

- “Nuevo día?” corresponde a la consulta de la bandera new_day, la cual se levanta en el módulo timer cuando inicia un nuevo día. “Calcular eventos solares” significa que se utilizarán funciones del módulo event_management, las cuales actualizarán el plan horario del modo Reloj Astronómico, con las horas de amanecer y atardecer calculadas en el módulo sun_equation.
- “Nuevo segundo?” corresponde a la consulta de la bandera new_second, la cual se levanta en el módulo timer cuando pasa un segundo. “Tomar valor de tensión de la fotocélula” significa que se utilizarán funciones del módulo photocell para obtener el valor de la digitalización de la tensión de salida del circuito fotosensible.
- “Nuevo minuto?” corresponde a la consulta de la bandera new_minute, la cual se levanta en el módulo timer cuando inicia un nuevo minuto. “Llamar a take_action” significa que se ejecutará la función take_action del módulo control. Esta función es la que maneja los modos de funcionamiento, consultando cuál es el modo en el que está el controlador y tomando las acciones necesarias a partir de esto. Dichas acciones son las que se describen en la sección 5.4.

5.4. Descripción de la implementación de los modos

La función take_action es la que maneja los modos de funcionamiento. Dentro de cada modo hay distintos procesos y tomas de decisiones. Se describe a continuación el funcionamiento de los distintos modos.

Modo Directo:
El modo Directo consiste únicamente en ejecutar los comandos que llegan desde el transceiver al MCU vía UART. En este estado no se realizan procesos ni se toman decisiones, más allá de la ejecución directa de los comandos.

Modos Reloj Astronómico y Plan Horario:
Los modos Reloj Astronómico y Plan Horario funcionan en base a eventos con horas de ocurrencia prefijadas y tienen la opción de tomar en cuenta la información proveniente de la fotocélula. En caso de activar dicha opción la información de la fotocélula es la de mayor jerarquía para la toma de decisiones en las horas de sol (específicamente entre el amanecer y el atardecer calculados por el reloj astronómico). En la noche la información de la fotocélula no es tenida en cuenta en ningún caso. Puede verse en la figura 5.3 el diagrama de flujo correspondiente a estos dos modos. Cabe destacar que el proceso que responde a la pregunta “¿Debo prender o apagar la luz según la fotocélula?” tiene su propio funcionamiento, el cual será desarrollado en la descripción del modo Fotocélula.

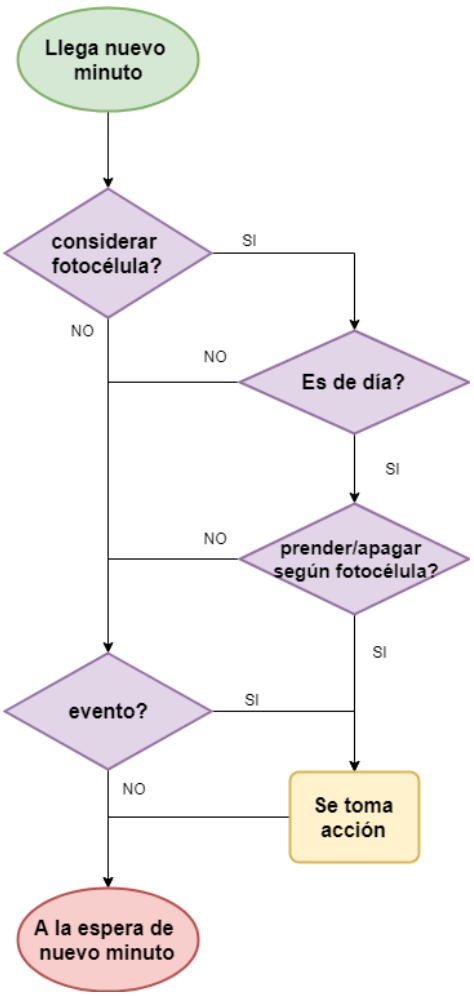


Figura 5.3: Diagrama de flujo de los modos Reloj Astronómico y Plan Horario.

Modo Fotocélula:

En este modo se controla la luminaria únicamente según el nivel del luz que incide sobre la fotocélula. Para saber si hay suficiente luz ambiente se toman los valores de tensión del circuito fotosensible cada un segundo. Al llegar un nuevo minuto, los sesenta valores son promediados y el promedio es comparado contra un umbral de luz y un umbral de oscuridad. Estos valores umbral están predefinidos. Si la tensión supera el umbral de luz significa que hay suficiente luz ambiente como para apagar la luminaria. Si por el contrario la tensión está por debajo del umbral de oscuridad, se interpreta que no hay suficiente luz en el ambiente y la luminaria debe estar prendida. Si el promedio es menor que el umbral de luz pero mayor al umbral de oscuridad la luminaria debe permanecer como estaba. Esta histéresis tiene como objetivo eliminar la posibilidad de que la luminaria sea encendida y apagada en minutos sucesivos, situación que podría suceder en caso de tener un único umbral de comparación. En la figura 5.4 se muestra un esquema representando los distintos niveles de tensión, según son intepretados por el MCU

en función de los umbrales.

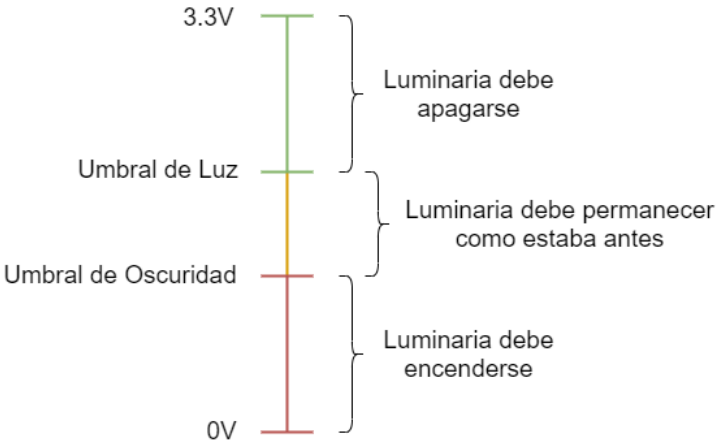


Figura 5.4: Niveles de luz interpretados por el MCU.

5.5. Implementación del reloj astronómico

El módulo sun_equation es quien provee las funciones que calculan la hora de amanecer y atardecer a partir de una fecha y una posición geográfica dada. Estos cálculos implementan la "Ecuación de amanecer", basándose en el artículo "Sunrise equation" [32]. La ecuación se muestra a continuación:

$$cos(\omega_0) = \frac{sin(a) - sin(\phi) \times sin(\delta)}{-cos(\phi) \times cos(\delta)}, \tag{5.1}$$

donde:

- ω_0 : es la hora ángulo del amanecer o atardecer dependiendo del signo que se toma en el resultado (al aplicar arcocoseno). Si se toma el valor positivo, corresponde al amanecer, y si es negativo al atardecer.
- ϕ : es la latitud de donde se quiere calcular el atardecer o amanecer.
- δ : es la declinación solar.
- a : es la altura del centro del disco solar.

La latitud ϕ se establece como un valor constante del programa, de valor $-34{,}9011$, que corresponde a la latitud de la ciudad de Montevideo. La declinación solar δ se calcula a partir de la fecha del momento actual del controlador. La altura del disco solar a se establece como un valor constante del programa, de valor $-0{,}83^\circ$, como se indica en el artículo. Utilizando estos parámetros en la ecuación 5.1 se despejan las horas ángulo del amanecer y atardecer. Luego se convierten esos valores en horas convencionales.

Esta implementación del reloj astronómico utiliza funciones trigonométricas del módulo "math" de la biblioteca estándar de C. Estas funciones operan con variables representadas en punto flotante. En consecuencia, la implementación del reloj astronómico es computacionalmente costosa por lo que es menester contar con una unidad de punto flotante hardware (FPU), ya que ahorra memoria y agiliza la velocidad de los cálculos. El MCU utilizado (MSP432P401R) posee una FPU.

5.6. Manejo de eventos

Para lograr hacer cambios programados en el estado de la luminaria en los modos Plan Horario y Reloj Astronómico se definió el evento. Como se mencionó antes, un evento está conformado por una hora en la cual se debe actuar sobre la luminaria, así como el nivel de luz que debe dar la luminaria a partir de dicha hora. Así definidos, los eventos son las piezas con las cuales se construye un plan horario. En esta sección se profundiza en las decisiones de diseño vinculadas al manejo de eventos, así como en los aspectos fundamentales de su implementación.

5.6.1. Consideraciones de diseño

El modo Reloj Astronómico y el modo Plan Horario difieren en muchos aspectos de su funcionamiento. Sin embargo, el modo Reloj Astronómico opera como si fuera un plan horario que siempre consta de dos eventos: uno para representar el amanecer y otro para representar el atardecer. La principal diferencia en el funcionamiento de estos dos modos está en la fuente de generación de eventos. Los eventos en el Plan Horario deben ser agregados por el usuario y pueden ser modificados por él en cualquier momento. En contraparte, los del Reloj Astronómico se generan automáticamente (a partir de los cálculos realizados por el módulo sun_equation) y no pueden ser modificados por el usuario (aunque sí consultados usando el comando DIR EVENT SE).

El controlador cuenta con tres listas de eventos: una para el modo Reloj Astronómico y dos para el modo Plan Horario. Una de las listas de eventos del Plan Horario corresponde a los días de fin de semana y otra a los días de entre semana. El comportamiento de la luminaria en los modos Plan Horario y Reloj Astronómico es regido completamente por los eventos de estas listas, a menos que se habilite la opción de tomar en cuenta a la fotocélula.

A continuación se listan decisiones de diseño relevantes:

- Luego de un reset, las dos listas del modo Plan Horario coinciden con la lista del modo Reloj Astronómico. Esto significa que las listas tendrán dos eventos: uno que representa el amanecer y otro el atardecer.
- Al pasar de un modo no basado en eventos (modos Directo y Fotocélula) a un modo con eventos (modos Plan Horario y Reloj Astronómico) es necesario actualizar la salida de la luminaria tras el cambio de modo. Concretamente, se pone la luminaria al nivel de luz indicado por el evento pasado más reciente perteneciente a la lista del modo con eventos. Si la lista está vacía, se

considera que hubo un error por parte del usuario y la luminaria se enciende por si dicho error se cometió en la noche.

- Si se habilita la opción de utilizar la fotocélula, durante las horas del día la luminaria se prenderá en caso de haber poca luz ambiente sin importar lo programado con eventos. Esta decisión de diseño busca que el sistema responda bien en situaciones en las que la luz diurna sea escasa, verbigracia, un día tormentoso. La información de la fotocélula no es tomada en cuenta durante la noche, lo cual evita que la luminaria se apague en la noche a causa de un factor externo como podría ser la luz de una fachada. La distinción entre día y noche se logra a partir de los eventos solares calculados por el módulo sun_equation.

- El evento de la lista del modo Reloj Astronómico que representa el amanecer tiene la hora del amanecer calculado por sun_equation, pero atrasada doce minutos. Por otro lado, el evento que representa el atardecer tiene la hora del atardecer calculada por sun_equation, pero adelantada doce minutos. Este cambio en las horas de los eventos solares surge a partir del análisis sobre la precisión de los cálculos realizados por el módulo sun_equation. En dicho análisis, presente en la sección 6.2.1, se estima que el error máximo en el cálculo de las horas amanecer y atardecer es de doce minutos. En consecuencia, atrasando la hora del amanecer y adelantando el atardecer en doce minutos se garantiza que, en modo Reloj Astronómico, la luminaria estará prendida durante la salida y puesta del sol.

- Cada vez que inicia un nuevo día, se actualizan los eventos de la lista correspondiente al Reloj Astronómico.

- Cada evento de una lista es identificado por un número identificador de evento (ID). Este número es asignado automáticamente por el controlador y permite al usuario referenciar eventos para borrarlos. Utilizando el comando DIR EVENT DT es posible acceder a la lista de eventos indicada por DT con sus respectivos números identificadores. Para borrar un evento de la lista, se utiliza el comando DEL EVENT DT ID, donde ID es el número identificador del evento que se quiere eliminar.

- Cuando se agrega un evento a una lista (utilizando el comando ADD EVENT) se hace un control de validez del mismo. Es decir, se verifica que la hora existe y que el número de dimerización sea entre 0 y 100.

En la figura 5.5 se muestra una situación ejemplo que pretende ilustrar algunas de las decisiones de diseño mencionadas anteriormente. En este ejemplo se pasa del modo Directo al modo Reloj Astronómico con uso de fotocélula. Por un lado, si la opción de tomar en cuenta la medida de la fotocélula estuviera desactivada, al cambiar de modo la luminaria debería mantenerse apagada ya que el evento pasado más cercano es el amanecer. No obstante, como dicha función está activada, la luminaria se prende. Esto es porque la fotocélula capta poca luz, y en este modo

la información de la fotocélula es de mayor jerarquía que la de los eventos en las horas del día.

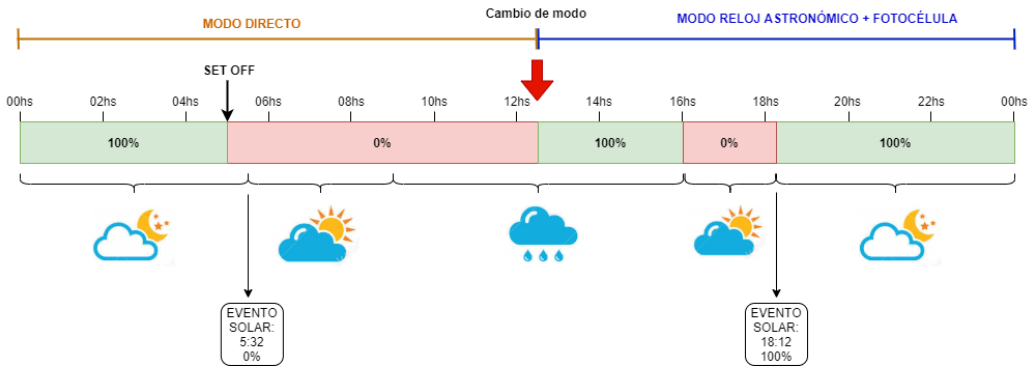


Figura 5.5: Ejemplo de la respuesta del controlador ante un cambio de modo. Los porcentajes en el centro de la barra representan los niveles de dimerización de la luminaria.

5.6.2. Aspectos de implementación

La implementación del manejo de eventos se realiza en el módulo event_management. En el mismo se define el evento, el cual es una estructura de dos campos: uno que contiene la hora del evento y el otro el nivel de dimerización. Se definen tres arreglos de estructuras evento, uno por cada lista de eventos concebida. Con el fin de distinguir los lugares del arreglo que tienen algún evento de aquellos que no, se definió el evento nulo. La única función de este evento es indicar que en ese lugar del arreglo de eventos no hay un evento. El evento nulo tiene el entero -1 en sus campos (hora y nivel de luz).

Los arreglos de eventos no están ordenados. Cuando se añade un evento, el mismo se agrega al lugar más bajo disponible del arreglo correspondiente (esto es, que en ese lugar del arreglo esté el evento nulo). Esto genera la necesidad de que cada evento tenga asociado un número identificador (ID) dentro de su arreglo, el cual es la posición del evento en el arreglo. En la figura 5.6 se muestra un ejemplo de la evolución de un arreglo (fin de semana) cuando el controlador recibe una secuencia de comandos determinada.

Para saber el ID del próximo evento en el día (evento futuro más cercano en el tiempo) se utilizan dos variables: en una se almacena el ID del próximo evento en el arreglo de eventos solares, y en otra se guarda el ID del próximo evento en el plan horario (puede corresponder al arreglo de día de semana o al de fin de semana, dependiendo del momento actual). Cada vez que llega un nuevo minuto se compara el tiempo actual con la hora del próximo evento en el arreglo correspondiente, con el fin de saber si el nivel de luz de la luminaria debe ser modificado. Al alcanzar la hora de este evento se debe recalcular el ID del próximo evento, para quedar a la espera del mismo.

Figura 5.6: Evolución del arreglo de eventos del fin de semana con una secuencia de comandos.

El ID del próximo evento debe calcularse en las siguientes situaciones:

- al alcanzar un nuevo evento (como se explicó recién).
- tras un reset.
- cada vez que se añade un evento al arreglo que rige el modo actual, ya que el evento añadido puede ser el más cercano.
- cada vez que se borra un evento del arreglo que rige el modo actual, ya que el evento borrado podía haber sido el más cercano.
- cada vez que se cambia de modo, ya que el próximo evento pudo haber cambiado desde la última vez que se estuvo en ese modo.

A continuación se describe un ejemplo para clarificar lo anteriormente expuesto: se considera el controlador en modo Plan Horario un día sábado a las 08:25 como

se muestra en la figura 5.7. En ese momento, llega un comando SET DIM 40, por lo que el controlador pasa a modo Directo, poniendo la luminaria al 40 % de su intensidad. No se ingresan nuevos comandos al controlador hasta las 16:00, por lo que permanece en modo directo y con la luminaria al 40 %. A dicha hora, se ingresa el comando SET MODE 2, lo cual lleva al controlador nuevamente al modo Plan Horario. Al cambiar de modo, volviendo al Plan Horario, se recalcula el ID del próximo evento. En este caso, el evento próximo cambió mientras el controlador se encontraba en modo directo, por lo que recalcular el ID del próximo evento es fundamental. Adicionalmente, según el último evento en el plan horario la luminaria debería estar al 70 % a las 16:00. Por lo tanto, al cambiar de modo se debe ajustar el nivel de luz de la luminaria, cambiando de 40 % a 70 %. En la figura 5.8 puede verse el estado de la luminaria tras el cambio de modo.

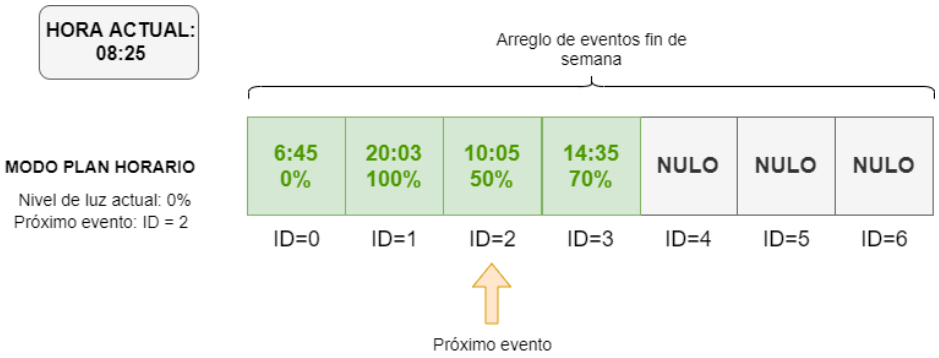


Figura 5.7: Estado de la luminaria y el arreglo de eventos a las 8:25.

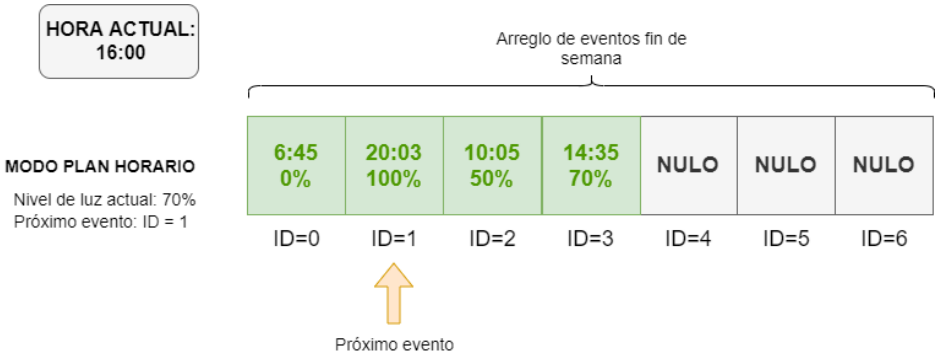


Figura 5.8: Estado de la luminaria y el arreglo de eventos a las 16:00.

5.7. Codificación de comandos

Como se mencionó previamente, los comandos llegan a la UART del MCU tras haber sido recibidos por el LoRa transceiver. En estas dos etapas (LoRa y UART) los comandos se encuentran codificados. Por ejemplo, el comando para encender la luminaria no llega al MCU como la cadena de caracteres "SET ON", si no que como el byte 0x00. Para entender por qué se usa la codificación y para justificar las decisiones tomadas en este ámbito, es necesario recordar una característica de la comunicación LoRa: su bajo ancho de banda. Por ser una tecnología de largo alcance y de bajo consumo, LoRa tiene un ancho de banda pequeño, limitando el tamaño de los paquetes a ser enviados. En consecuencia, el equipo concibió una codificación con el objetivo de transmitir comandos comprimidos vía LoRa. Las ventajas y limitaciones de la comunicación LoRa fueron desarrolladas en el capítulo 2.3.

Para lograr la codificación de los mensajes, el Servidor Central toma las cadenas de caracteres ASCII ingresadas por el usuario y las convierte en cadenas de bytes (mensaje codificado) para luego enviarlas por LoRa. De esta manera, el usuario podrá utilizar los comandos en su forma no codificada para mayor comodidad y comprensibilidad. Para codificar los comandos se utiliza el siguiente criterio: los 5 bits más significativos (MSB de aquí en más, del inglés Most Significant Bit) del byte 0, corresponden a la naturaleza del comando. Estos bits serán refieridos como *Identificador del comando* de aquí en más. A continuación se muestra la forma de codificar cada comando:

- SET ON: este comando se envía como un byte, conteniendo en sus 5 MSB el identificador de comando 0x00. Al recibir este comando, el MCU no envía respuesta.
- SET OFF: este comando se envía como un byte, conteniendo solo el identificador de comando 0x01. Al recibir este comando, el MCU no envía respuesta.
- SET DIM XX: se envían dos bytes. En el byte 0, se encuentra el identificador de comando 0x02. En el byte 1 se encuentra el nivel de dimerización XX, siendo 0x00 0 % y 0x14 100 % (hay 20 posibles niveles configurables). Al recibir este comando, el MCU no envía respuesta.
- ADD EVENT DT HH:MM XX: el envío consiste en tres bytes. El byte 0 tiene el identificador de comando 0x03 si el evento a agregar es para WE o 0x04 si es para WD. La hora se envía como minutos pasados desde la media noche, por ejemplo las 14:45hs se expresa como 885 minutos. Hay un total de 1440 minutos en un día, por lo que se necesitan once bits para enviar este número. Se utiliza el byte 1 y los 3 MSB del byte 2 para este número. Finalmente, se usan los 5 bits menos significativos (LSB, del inglés Least Significant Bit) para el nivel de luz XX siendo 0x00 0 % y 0x14 100 %. En la figura 5.9 se muestra la forma de un mensaje de este tipo. Al recibir este comando, el controlador no envía respuesta.

Figura 5.9: Diagrama de un mensaje ADD EVENT codificado.

- RES EVENT DT: se envía un solo byte con el identificador de comando 0x05 si DT es WE o 0x06 si es WD. Al recibir este comando, el MCU no envía respuesta.
- DIR EVENT DT: el envío consiste en un solo byte con el indentificador de comando 0x07 si DT es WE, 0x08 si es WD o 0x09 si es SE. El MCU responde paquetes de tres bytes (un paquete por cada elemento en la lista de eventos). Al igual que en el comando ADD EVENT, la hora de cada evento se transmite como minutos desde la medianoche, por lo que son necesarios 11 bits. El byte 0 y los 3 MSB del byte 1 de cada conjunto (cada evento) se utilizan para la cantidad de minutos. Los 5 LSB del byte 1 corresponden al identificador del evento (ID). En el byte 2 se envía el nivel de dimerización (0x00 para 0 % hasta 0x64 para 100 %). En la figura 5.10 se muestra un diagrama de un paquete de tres bytes, correspondientes a un evento en la lista. La respuesta completa tiene tamaño $3 \times n$ bytes, siendo n la cantidad de eventos registrados.

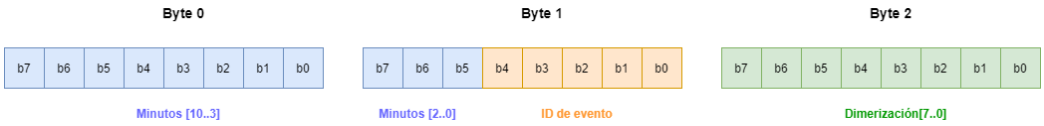


Figura 5.10: Diagrama del formato de cada evento, en respuesta a un DIR EVENT.

- DEL EVENT DT ID: se envían dos bytes. El byte 0 contiene el identificador de comando 0x0A si DT es WE o 0x0B si es WD. En el byte 1 se encuentra el ID del evento que se desea borrar. Al recibir este comando, el MCU no envía respuesta.
- SET MODE X: el envío consiste en un único byte. En los 5 MSB se encuentra el identificador de comando como es usual, en este caso siendo 0x0C. En los 2 LSB se encuentra el modo X. En la figura 5.11 se muestra gráficamente la composición del byte. Al recibir este comando, el MCU no envía respuesta.

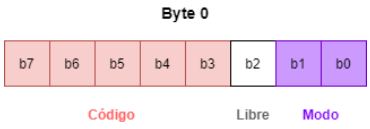


Figura 5.11: Diagrama del comando SET MODE codificado.

- SET MOMENT DD/MM/YYYY HH:MM:SS : se envían 5 bytes. El byte 0 tiene en los 5 MSB el indentificador de comando 0x0D. Con este comando se configura la hora, la cual se envía como los segundos transcurridos desde la medianoche. Un día completo tiene 86400 segundos, por lo que hacen falta 17 bits para representarlos. Estos 17 bits se ubican en el byte 1 y 2, además del MSB del byte 3. El año se representa como la cantidad de años transcurridos desde el 2000, es decir, para representar el año 2020 se envía un 0x14. Se decidió que el año se represente con 6 bits, admitiendo valores entre 2000 y 2063. Este dato ocupa los seis bits centrales del byte 3 (o sea byte3[6..1]). Para representar el día del año, se utiliza la cantidad de días transcurridos desde el 1 de enero (Por ejemplo el 2 de abril es 91, o 92 si el año es bisiesto). Como un año tiene un máximo de 366 días, se necesitan 9 bits para representar la cantidad de días. Estos 9 bits ocupan el LSB del byte 3 y todo el byte 4. En el diagrama de la figura 5.12 se esquematiza lo anteriormente expuesto.

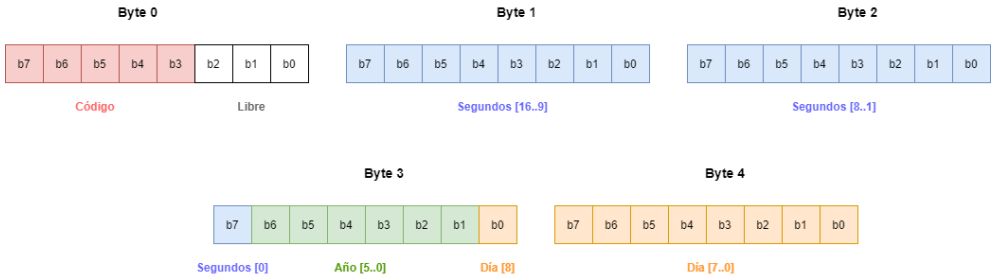


Figura 5.12: Diagrama de la organización de los bits en el comando SET MOMENT.

- USE PHC ON: el mensaje consta de un byte conteniendo el identificador de comando 0x0E. Al recibir este comando, el MCU no envía respuesta.
- USE PHC OFF: el mensaje consta de un byte conteniendo el identificador de comando 0x0F. Al recibir este comando, el MCU no envía respuesta.
- GET MOMENT: el mensaje consta de un byte conteniendo el identificador de comando 0x10. El MCU responde con la hora y fecha codificada en cuatro bytes exactamente como se ve en los bytes 1, 2, 3 y 4 de la figura 5.12.
- GET MODE: el mensaje consta de un byte conteniendo el identificador de comando 0x11. El MCU responde con un byte conteniendo el número del modo (del 0 al 3).
- GET VOLT: el mensaje consta de un byte conteniendo el identificador de comando 0x12. El controlador responde con dos bytes conteniendo la medida de voltaje en décimas de volt.
- GET CURR: el mensaje consta de un byte conteniendo el identificador de comando 0x13. El controlador responde con dos bytes conteniendo la medida de potencia activa en décimas de miliampere.

- GET ACTPWR: el mensaje consta de un byte conteniendo el identificador de comando 0x14. El controlador responde con dos bytes conteniendo la medida de potencia activa en décimas de watt.

- GET REAPWR: el mensaje consta de un byte conteniendo el identificador de comando 0x15. El controlador responde con dos bytes conteniendo la medida de potencia reactiva en décimas de volt-ampere reactivo.

- GET APPPWR: el mensaje consta de un byte conteniendo el identificador de comando 0x16. El controlador responde con dos bytes conteniendo la medida del módulo de la potencia aparente en décimas de volt-ampere.

- GET COSPHI: el mensaje consta de un byte conteniendo el identificador de comando 0x17. El controlador responde con dos bytes conteniendo el factor de potencia en centésimas.

Se presenta en la tabla 5.1 un resumen de los comandos, con los datos extras que se deben enviar y sus respuestas.

Id	Comando	Extras	Respuesta
00000	SET ON		
00001	SET OFF		
00010	SET DIM	XX	
00011	ADD EVENT WE	MMMM XX	
00100	ADD EVENT WD	MMMM XX	
00101	RES EVENT WE		
00110	RES EVENT WD		
00111	DIR EVENT WE		Eventos (2 bytes/evento)
01000	DIR EVENT WD		Eventos (2 bytes/evento)
01001	DIR EVENT SE		Eventos (4 bytes)
01010	DEL EVENT WE	ID	
01011	DEL EVENT WD	ID	
01100	SET MODE	X	
01101	SET MOMENT	DD/MM/YYYY HH:MM:SS	
01110	USE PHC ON		
01111	USE PHC OFF		
10000	GET MOMENT		SSSSS YY MM (4 bytes)
10001	GET MODE		X (1 bytes)
10010	GET VOLT		V (2 bytes)
10011	GET CURR		A (2 bytes)
10100	GET ACTPWR		P (2 bytes)
10101	GET REAPWR		Q (2 bytes)
10110	GET APPPWR		S (2 bytes)
10111	GET COSPHI		COSPHI (2 bytes)

Tabla 5.1: Comandos, sus respectivos identificadores y sus respuestas.

5.8. Comunicación LoRa

5.8.1. Comunicación UART para el manejo del LoRa transceiver

El módulo LoRa (software) maneja la comunicación UART con el LoRa transceiver. Como se explicó en la sección 4.2.3, el chip RN2903 recibe ciertos comandos UART que modifican su funcionamiento. El módulo LoRa (software) utiliza la UART para enviar algunos de estos comandos al RN2903 y recibir sus respuestas. Estos comandos son:

- **sys reset :** reinicia el dispositivo y devuelve su versión de firmware.
- **mac set 'param' 'value':** establece en el chip el valor 'value' en el parámetro 'param', donde 'param' puede ser DEVEUI, APPEUI o APPKEY.
- **mac join OTAA :** intenta activarse frente a una aplicación vía OTAA usando los valores establecidos con el comando mac set 'param' 'value'.
- **mac set ch status 'channel ID' 'status' :** habilita o deshabilita el uso del canal identificado con 'channel ID'. El parámetro 'channel ID' puede tomar valores del 0 al 71. El parámetro 'status' puede tomar el valor 'on' que habilita el uso de este canal y 'off' que lo deshabilita. Los canales del 0 al 63 son los canales disponibles para uplinks, mientras que los canales del 64 al 71 son los canales disponibles para downlinks.
- **mac tx 'type' 'portno' 'payload' :** este comando se utiliza para mandar uplinks. El parámetro 'type' indica si se quiere que la comunicación se haga con confirmación de la aplicación, indicando si el paquete se recibió o no. Toma el valor "cnf" para el primer caso o "uncnf" para el segundo. El parámetro 'portno' refiere a qué puerto se va a mandar la información. Este dato se usa para que la aplicación pueda diferenciar paquetes sin necesidad de utilizar bytes del payload. Toma valores del 1 al 223. El parámetro 'payload' son caracteres en ASCII que indican el mensaje que se quiere mandar. Es importante destacar que en este último parámetro es necesario escribir los bytes en ASCII; si se quiere mandar el byte 0xAB es necesario escribir ese byte como dos caracteres ASCII (AB).

El módulo LoRa (software) también recibe mensajes enviados por el chip. Estos pueden ser una confirmación de un comando ejecutado correcta o incorrectamente, o downlinks desde la aplicación. Con respecto a las confirmaciones, cuando se le manda un mensaje al RN2903 con un comando, el RN2903 responde con el resultado de haber ejecutado ese comando. Aunque la mayoría de las respuestas son simples como "ok", algunos comandos tienen resultados que proveen más información sobre el resultado. Por ejemplo, el comando **mac tx 'type' 'portno' 'payload'** devuelve como resultado: "ok", "invalid param", "not joined", "busy", entre otros. En cuanto a los downlinks, el chip RN2903 envía por UART **mac rx 'payload'** cuando recibe un mensaje desde la aplicación ('payload' es el mensaje que recibió desde la aplicación).

5.8.2. El controlador como dispositivo end-point clase A

Como se explicó en la sección 2.3, existen tres clases de dispositivo end-point: clase A, clase B y clase C. Entre estas, la clase C es la que mejor sirve a los objetivos del presente proyecto, dado que esta clase ofrece la posibilidad de enviar comandos al controlador en un momento cualquiera. Esto implica que el consumo del controlador sea mayor, pero en este proyecto el consumo no es una limitante dado que se cuenta con el suministro de la red eléctrica. Sin embargo, el funcionamiento de un dispositivo end-point (como el controlador) debe ajustarse a los requerimientos y limitaciones de la red de comunicación utilizada. Como será desarrollado en la sección 6.3.2, el proceso de prueba y validación del controlador como dispositivo end-point en una red LoRaWAN fue realizado utilizando la versión online TTN, la cual hasta la fecha de publicación del presente documento no soporta dispositivos clase C.

Por lo tanto, el controlador funciona como dispositivo end-point clase A. Para recibir comandos de control el controlador envía un uplink cada minuto con el payload 0xFF, lo cual genera una ventana de recepción para el controlador. Si desde el Servidor Central existe un downlink a enviar, este se envía en la ventana de recepción mencionada. Esta es la forma en la que el controlador recibe los comandos del Servidor Central.

5.8.3. Respuestas a comandos

Existen ciertos comandos del controlador que generan una respuesta del mismo. El controlador responde a cada tipo de comando en un puerto determinado. De esta manera, desde el Servidor Central se conoce qué comando generó el uplink recibido. En la tabla 5.2 se muestran las asociaciones entre tipo de comando y puerto.

Comando	Puerto de Respuesta
DIR EVENT WD	1-5
DIR EVENT WE	6-10
DIR EVENT SE	11
GET MODE	20
GET MOMENT	21
GET VOLT	22
GET CURR	23
GET ACTPWR	24
GET REACTPWR	25
GET APPPWR	26
GET PFACTOR	27

Tabla 5.2: Puertos utilizados para las respuestas a los distintos comandos.

5.9. Obtención de medidas eléctricas del circuito de medida de consumo

El módulo acs71020 maneja la comunicación I^2C con el circuito de medida de consumo y obtiene los datos de medidas eléctricas que este reporta. Utilizando el módulo i2c del driverlib, el MCU realiza las lecturas a los registros de interés del chip ACS71020, de la manera que se explica en la sección 4.2.7. Luego guarda los datos del registro leído en un arreglo de 4 bytes y opera con ellos para representar la magnitud correspondiente en punto flotante. A continuación se listan las medidas eléctricas que se obtienen y se explica la forma en la cual se interpretan los bytes leídos. Se recomienda al lector rever la figura 4.14, la cual muestra qué información tienen los registros de interés del ACS71020. Para facilitar la explicación se llamará B_3, B_2, B_1 y B_0 a los 4 bytes que se obtienen de leer estos registros, en orden de más significativo a menos significativo.

- Voltaje (vrms): para obtener el voltaje se lee el registro 0x20. Los 15 LSB de los 32 leídos corresponden al voltaje. Este es representado como un número en punto fijo sin signo de 15 bits, con 15 bits fraccionarios. Este número (entre 0 y 1) debe ser multiplicado por la tensión de fondo de escala para obtener el valor de tensión deseado (en volt). La tensión de fondo de escala es aquella tensión que debería haber entre "L" y "N" para que hayan 275mV en los pines VINN y VINP del ACS71020. Es importante notar que es necesario enmascarar el byte 1 (B_1) de manera de resetear el bit 15, pues este no forma parte de la lectura de tensión. Entonces, el dato se obtiene a partir de la lectura como se muestra a continuación:

$$V_{RMS} = \frac{(B_1 \& 7Fh) \times 2^8 + B_0}{2^{15}} \times V_{FE}, \tag{5.2}$$

  donde V_{FE} representa la tensión de fondo de escala, que vale 535 V para el presente proyecto.

- Corriente (irmsavgonesec): para obtener la corriente se lee el registro 0x26. Este registro contiene un promedio de las medidas del último segundo de la corriente o del voltaje, pero no de ambos simultáneamente. Para seleccionar cuál de las dos magnitudes se promedia, se debe escribir el bit 21 del registro 0x0b. Si ese bit está en 0 se promedia el voltaje y si está en 1 se promedia la corriente. En el presente proyecto, se utiliza el promedio de corriente y no el de tensión, ya que los valores de corriente a ser medidos son más pequeños respecto al fondo de escala que los de tensión, por lo que las medidas de corriente son más sensibles al ruido. Adicionalmente, se escribe en los 8 LSB del registro 0x0C la cantidad de medidas utilizadas para realizar el promedio. Una vez configurados estos dos registros, se obtiene la corriente promediada de los bits 30 al 16 del registro 0x26. Esta es representada como un número en punto fijo sin signo de 15 bits con 14 bits fraccionarios. Este número (entre 0 y 2) debe ser multiplicado por la corriente de fondo de escala para obtener el valor de corriente deseado (en amperes). La corriente de fondo de

escala es la máxima corriente que puede ser medida por el ACS71020, y para la versión del chip utilizada en el presente proyecto (ACS71020KMABTR-030B3-I2C) esta corriente es de 30A. Es importante notar que es necesario enmascarar el byte 3 (B_1) de manera de resetear el bit 31, pues este no forma parte de la lectura de tensión. Entonces, el dato se obtiene a partir de la lectura como se muestra a continuación:

$$I_{RMS}^{avg} = \frac{(B_3 \& 7Fh) \times 2^8 + B_2}{2^{14}} \times I_{FE}, \tag{5.3}$$

donde I_{FE} representa la corriente de fondo de escala (30 A).

- Potencia activa (pactavgonesec): para obtener la potencia activa se lee el registro 0x28. Este registro contiene un promedio de las medidas del último segundo de la potencia activa. Una vez más, es necesario escribir en los 8 LSB del registro 0x0C la cantidad de medidas utilizadas para realizar el promedio. Una vez configurado este registro, se obtiene la potencia activa promediada de los bits 16 al 0 del registro 0x28. Esta es representada como un número en punto fijo con signo de 17 bits con 15 bits fraccionarios en complemento a 2. Este número (entre -2 y 2) debe ser multiplicado por la potencia activa de fondo de escala para obtener el valor de potencia activa deseado (en watts). La potencia de fondo de escala se calcula como $P_{FE} = V_{FE} \times I_{FE}$, y toma el valor de 16500 W. Entonces, si el bit 16 es 0, la potencia será positiva y el dato se obtiene a partir de la lectura como se muestra a continuación:

$$P^{avg} = \frac{B_1 \times 2^8 + B_0}{2^{15}} \times P_{FE}. \tag{5.4}$$

  Si el bit 16 es 1 la potencia será negativa y el dato se obtiene de la siguiente manera:

$$P^{avg} = -\frac{[!(B_1 \times 2^8 + B_0)] + 1}{2^{15}} \times P_{FE}, \tag{5.5}$$

  donde el operador ! representa la negación de todos los bits.

- Potencia aparente (papparent): para obtener la potencia aparente se lee el registro 0x22. Los 16 LSB de los 32 leídos corresponden a la potencia aparente. Este es representado como un número en punto fijo sin signo de 16 bits, con 15 bits fraccionarios. Este número (entre 0 y 2) debe ser multiplicado por la potencia aparente de fondo de escala para obtener el valor de potencia deseado (en volt ampere). La potencia de fondo de escala se calcula como $S_{FE} = V_{FE} \times I_{FE}$, y toma el valor de 16500 VA. Entonces, el dato se obtiene a partir de la lectura como se muestra a continuación:

$$S = \frac{B_1 \times 2^8 + B_0}{2^{15}} \times S_{FE}, \tag{5.6}$$

- Potencia reactiva (pimag): para obtener la potencia reactiva se lee el registro 0x23. Los 16 LSB de los 32 leídos corresponden a la potencia reactiva. Este

es representado como un número en punto fijo sin signo de 16 bits, con 15 bits fraccionarios. Este número (entre 0 y 2) debe ser multiplicado por la potencia reactiva de fondo de escala para obtener el valor de potencia deseado (en volt ampere reactivo). La potencia de fondo de escala se calcula como $Q_{FE} = V_{FE} \times I_{FE}$, y toma el valor de 16500 VAr. Entonces, el dato se obtiene a partir de la lectura como se muestra a continuación:

$$Q = \frac{B_1 \times 2^8 + B_0}{2^{15}} \times Q_{FE}, \tag{5.7}$$

- Factor de potencia (pfactor): para obtener el factor de potencia se lee el registro 0x24. Los 11 LSB de los 32 leídos corresponden al factor de potencia. Este es representado como un número en punto fijo con signo de 11 bits, con 9 bits fraccionarios (resultando en un rango de entre -2 y 2). Es importante notar que es necesario enmascarar el byte 1 (B_1) de manera de resetear los 5 MSB bits, pues este no forma parte de la lectura de factor de potencia. Entonces, si el bit 10 es 0, el factor de potencia será positivo y el dato se obtiene a partir de la lectura como se muestra a continuación:

$$FP = \frac{(B_1 \& 07h) \times 2^8 + B_0}{2^9} \tag{5.8}$$

Si el bit 10 es 1 el factor de potencia será negativo y el dato se obtiene de la siguiente manera:

$$FP = \frac{![(B_1 \& 07h) \times 2^8 + B_0] + 1}{2^9} \tag{5.9}$$

donde el operador ! representa la negación de todos los bits.

Capítulo 6

Pruebas

6.1. Introducción

En el presente proyecto las pruebas fueron realizadas siguiendo una estrategia marcada: modularización y validación. Esta estrategia consiste en separar lo que se desea probar en módulos independientes, de manera de llevar a cabo evaluaciones individuales de dichos módulos. La idea detrás de este procedimiento de seccionamiento es estudiar y validar módulos de manera sencilla, antes de integrarlos al sistema completo. Dicho procedimiento es aplicado tanto para el hardware como para el software. En el caso particular del hardware los módulos fueron validados inicialmente fuera de los PCBs finales, para ser luego soldados y probados en dichos PCBs.

Para algunas pruebas se hizo uso de la interfaz de comunicación desarrollada para el trabajo de Sistemas Embebidos para Tiempo Real (mencionado en la sección 1.4) conocida como *sisem_shell*. Esta interfaz de comunicación hace que el controlador pueda recibir comandos en forma directa desde una PC vía UART. Dichos comandos son cadenas de caracteres ASCII que son interpretadas por el controlador para actuar en consecuencia. Estas cadenas de caracteres corresponden a los comandos sin codificar, por ejemplo "SET OFF".

En este capítulo se describen las pruebas realizadas tanto para los módulos de software como para los módulos hardware, en las diferentes etapas planteadas. Finalmente se adjuntan dos videos que muestran pruebas de funcionamiento del controlador.

6.2. Pruebas de software

Como se mencionó, las pruebas fueron realizadas en forma modular. A continuación se describen brevemente los procedimientos para llevar a cabo dichas pruebas en los principales módulos de software.

6.2.1. Reloj astronómico

Para verificar que los cálculos del módulo sun_equation son correctos, se compararon los resultados obtenidos con este módulo con los proporcionados por la NOAA (National Oceanographic and Atmospheric Administration) [33]. Para comparar los valores, se ejecutó el código de sun_equation que calcula el amanecer y atardecer para todos los días de un año (2021) y se halló la diferencia (error en minutos) entre ambas fuentes de datos para cada día. Se toman los errores a través de un año cualquiera (2021) como representantes estadísticos del error en un día arbitrario, de cualquier año. Todas las pruebas realizadas fueron hechas usando la latitud de la ciudad de Montevideo. Las gráficas de las figuras 6.1 y 6.2 muestran el valor absoluto del error en minutos (barras azules) y el promedio del error en el año (línea naranja). Los valores notables de error se presentan en la tabla 6.1. Las imprecisiones en los resultados son producto de algunas aproximaciones realizadas para simplificar el cálculo en el módulo sun_equation. La existencia de estos errores de cálculo genera la necesidad de tomar un margen de seguridad para encender y apagar las luminarias, lo cual es explicado en la sección 5.6.

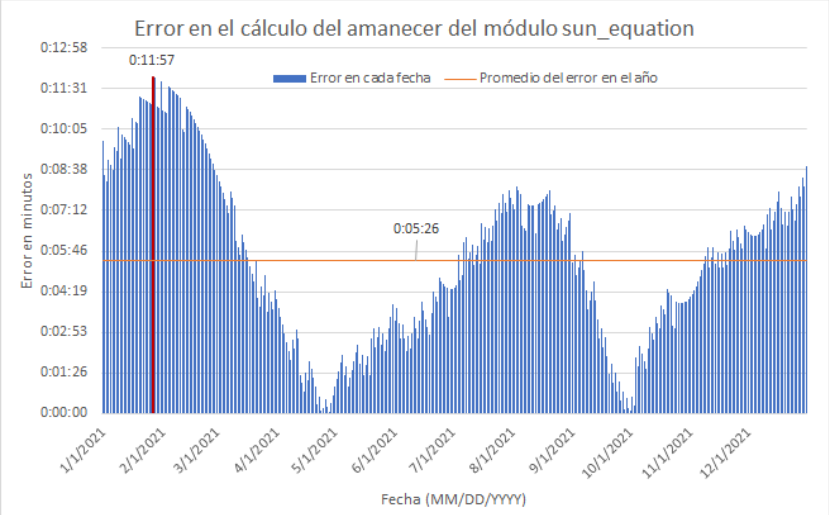


Figura 6.1: Gráfica que muestra el error en el cálculo del amanecer del módulo sun_equation en función de la fecha para el año 2021.

	Error máximo	Error promedio	σ del error
Amanecer	11:57	5:26	2:59
Atardecer	10:44	5:28	2:49

Tabla 6.1: Error máximo, promedio y desviación estándar en minutos:segundos del módulo sun_equation en el año 2021.

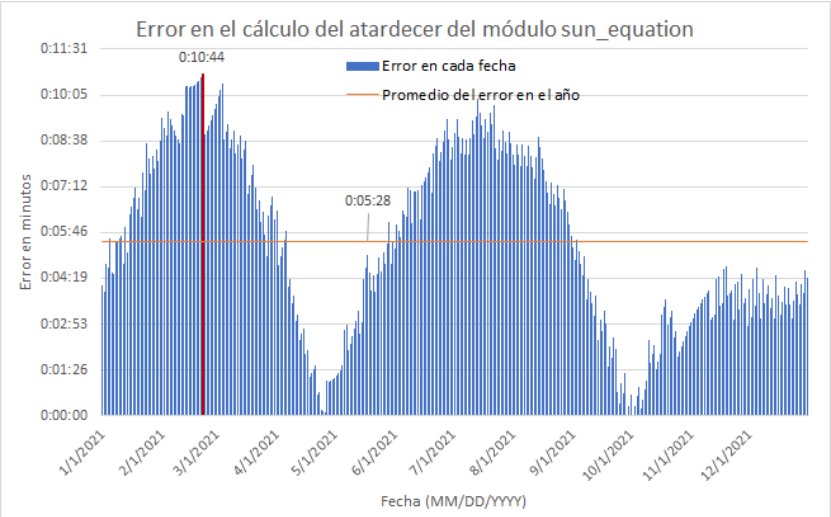


Figura 6.2: Gráfica que muestra el error en el cálculo del atardecer del módulo sun_equation en función de la fecha para el año 2021.

6.2.2. Manejo de eventos

Para comprobar el correcto funcionamiento del módulo de manejo de eventos (event_management) se realizaron pruebas sobre el arreglo de eventos del modo Reloj Astronómico y los arreglos de eventos del modo Plan Horario (arreglos de entre semana y de fin de semana).

En el caso del modo Reloj Astronómico, se verificó que al pasar a un nuevo día el arreglo correspondiente es actualizado con las horas de amanecer y atardecer calculadas por el módulo sun_equation.

En el caso del modo Plan Horario, se verificó que luego de un reset los arreglos adquieren los eventos de amanecer y atardecer calculados por el módulo sun_equation. A su vez se comprobó el correcto funcionamiento de los arreglos de Plan Horario, realizando operaciones sobre los mismos. Estas operaciones son: agregar un evento, borrar un evento, y borrar todos los eventos. Para realizar dichas operaciones se enviaron comandos por UART, utilizando el sisem_shell. Estos comandos son ADD EVENT, DEL EVENT y RES EVENT. Para ver el estado de los arreglos de eventos se utilizó el debugger del CCS, así como el comando DIR EVENT. Por último, se verificó que al estar en un día de semana el módulo se refería al arreglo de día de semana y lo mismo para el fin de semana. Se constató que los arreglos de eventos se comportaron de la forma esperada en todos los casos, por lo que el módulo se consideró validado.

6.2.3. Decoder

La verificación de funcionamiento del módulo decoder fue realizada usando CCS. Como se mencionó antes, la función principal del decoder es extraer la información de un comando codificado y proporcionársela a la función del módulo control que corresponda. Esta información puede ser una hora (en el caso de un

ADD EVENT), un modo (en el caso de un SET MODE) o un nivel de dimerización (en caso de SET DIM) entre otros. El módulo decoder tiene variables internas que almacenan esta información para ser pasadas como argumento a las funciones del módulo control. En vista de esto, se crearon manualmente cadenas de bytes representando comandos codificados y se enviaron por UART al MCU.

Se utilizó el debugger del CCS para corroborar que se había interpretado correctamente el comando, y que se había extraído la información de manera correcta en cada caso. La forma de corroborarlo fue consultar las variables internas que alojan dicha información (haciendo uso de breakpoints) y compararla con lo que se había deseado enviar. En la figura 6.3 se puede ver una representación gráfica de un ejemplo de este procedimiento.

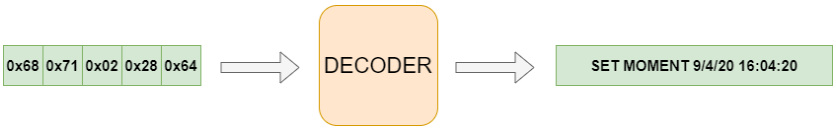


Figura 6.3: Ejemplo de prueba de la decodificación hecha por el módulo decoder.

Por otro lado, algunos de los comandos son consultas de variables del controlador, por lo que es necesario enviar una respuesta. Éstas también deben codificarse (como se explicó en la sección 5.7), y la tarea de codificarlas recae sobre el módulo decoder. Para probar el correcto funcionamiento de la codificación de respuestas, se enviaron comandos vía UART que esperan respuesta. Luego se compararon las respuestas obtenidas con las esperadas. En la figura 6.4 se muestra gráficamente el recorrido por los módulos de un comando y su respectiva respuesta. En verde se muestra el camino del comando y en rojo el de la respuesta a ese comando. Es importante aclarar que esta prueba se realizó con la etapa de decodificación ya validada. Tras verificar que las respuestas eran las esperadas en todos los casos probados, se tomó como validado el módulo decoder.

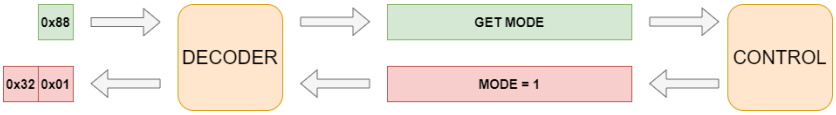


Figura 6.4: Ejemplo de prueba de la codificación de respuestas hecha por el módulo decoder.

6.2.4. Interpretación de lecturas del circuito de medida de consumo

Para comprobar el correcto funcionamiento de la interpretación de lecturas del circuito de medida de consumo se utilizó un Arduino Due oficiando como slave I^2C, cuya función es simular un ACS71020 [34]. Esta se programó con un código que configura al Arduino Due como slave I^2C con la dirección 0x45 (la misma que el PMIC del circuito de medida de consumo) y establece las respuestas que ésta debe dar al ser consultada por un master. El código simula el sistema de

direccionamiento de registros internos del ACS71020 explicado en la sección 4.2.7, en particular la figura 4.16. Para cada dirección de registro interno se programó una respuesta "hard-coded" generada manualmente siguiendo el formato de datos especificado en la hoja de datos del ACS71020. Por ejemplo, si el master inicia una lectura al registro de tensión, la placa auxiliar responderá con cuatro bytes. Esta palabra de cuatro bytes contiene en sus 15 LSB un número en punto fijo, sin signo, de 15 bits y con 15 bits fraccionarios, que representará una lectura de voltaje. El valor es elegido por el grupo, y pretende ser un valor cercano a lo que se medirá en la aplicación final.

Se realizó esta prueba leyendo todos los registros de interés, y se constató que el módulo de software acs71020_lib interpreta de forma correcta los datos que lee.

6.3. Pruebas modulares de hardware fuera del PCB

Como se mencionó anteriormente, los distintos módulos hardware fueron probados fuera de los PCBs finales antes de ser soldados a dichos PCBs. A continuación se desarrolla sobre las distintas pruebas realizadas en esta etapa para cada módulo.

6.3.1. Circuito de alimentación

El circuito de alimentación no se probó en esta etapa (fuera del PCB), debido a su sencillez. El circuito consiste de dos componentes electrónicos interconectados a partir de los cuales se obtienen los valores de tensión necesarios (figura 4.1).

6.3.2. LoRa transceiver

Las pruebas realizadas para el LoRa transceiver se dividieron en dos etapas. Primero se probó la comunicación entre la placa MCU y el LoRa transceiver mediante UART. Luego, se probó la comunicación entre el LoRa transceiver y una aplicación de prueba desarrollada en TTN.

Comunicación placa MCU - LoRa transceiver

Para validar la comunicación con el LoRa transceiver se programó en la placa MCU un "passthrough". Es decir, un programa que hace que la placa MCU envíe por UART hacia el LoRa transceiver los mensajes que recibe por otra UART desde una PC y viceversa. El objetivo de este programa es poder comunicarse directamente con el LoRa transceiver desde la PC.

Una vez configurado el programa anterior se probó enviar diferentes comandos al LoRa transceiver, los cuales generaron las respuestas esperadas. Por ejemplo, se envío el comando "sys reset", a partir del cual el Lora transceiver se reseteó y respondió de la siguiente manera: "RN2903 1.0.3 Aug 8 2017 15:11:09".

Comunicación LoRa transceiver - Aplicación TTN

Una vez confirmado el correcto funcionamiento de la comunicación entre el LoRa transceiver y la placa MCU, se construyó una red LoRaWAN para validar la comunicación entre el LoRa transceiver y una aplicación desarrollada en TTN. Como se explicó en la sección 2.3, una red LoRaWAN consta de cuatro tipos de elementos: dispositivos end-point, gateways, servidor de red y aplicaciones. El dispositivo end-point de la red construida es el controlador. Los demás dispositivos se describen a continuación.

El gateway utilizado es el Kona Micro Gateway, del fabricante Tektelic [35], del cual se presenta una foto en la figura 6.5. Se utilizó este gateway ya que fue el proporcionado por el tutor del proyecto.

Figura 6.5: Tektelic Kona Micro Gateway
www.tektelic.com

Como se mencionó en la sección 2.3.2, este proyecto utiliza The Things Network para tener acceso a un servidor de red y crear una aplicación [1]. Como paso previo es necesario registrar el gateway que se va a utilizar ante TTN. Una vez creada la aplicación, se procedió a habilitar la posibilidad de que se activen dispositivos end-point. Esta habilitación implica la definición de los parámetros necesarios para la activación vía OTAA del dispositivo end-point en la aplicación (DEV_EUI, APP_EUI y APP_KEY). Entonces, se establecieron estos parámetros en el LoRa transceiver utilizando el comando "mac set param value" y se procuró la activación del mismo mediante el comando "mac join OTAA". Desde la plataforma TTN se constató que se había recibido el Join Request, y que la activación se había realizado con éxito.

TTN cuenta con una consola que muestra el historial de los mensajes que son compartidos entre la aplicación y los dispositivos end-point. A su vez, la consola permite el envío de downlinks. Por otra parte, TTN posee un decodificador programable en lenguaje JavaScript, el cual genera salidas en formato JSON según

[1] Para crear la aplicación se siguieron los pasos que se explican en `https://www.thethingsnetwork.org/docs/`

los mensajes que recibe. Estas salidas son usadas para las integraciones con otras herramientas.

La integración que se utiliza para la visualización de datos en estas pruebas es la plataforma Ubidots. El propósito de esta plataforma es proveer una forma fácil de mostrar los datos de los mensajes de uplink que envían los dispositivos end-point. Ubidots ofrece múltiples "widgets" para mostrar los datos, como por ejemplo un termómetro pensado para mostrar temperaturas o una batería para mostrar niveles de batería en un dispositivo. A su vez, estos "widgets" pueden ser representaciones más complejas, como histogramas o gráficas que muestren la evolución de una variable en el tiempo. Un ejemplo de un "widget" de termómetro se puede apreciar en la figura 6.6. Más allá de estos "widgets" predefinidos, Ubidots también ofrece un HTML CANVAS que permite usar código de HTML, CSS y Javascript para que el usuario cree su propio "widget".

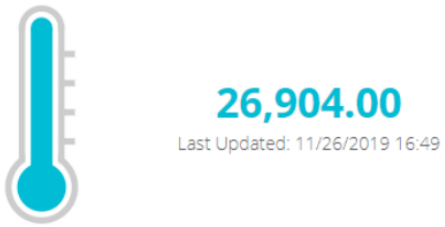


Figura 6.6: Widget termómetro de Ubidots
Fuente: www.ubidots.com

Para estas pruebas se crearon "widgets" utilizando el HTML CANVAS. En la figura 6.7 se puede apreciar un tablero que muestra los "widgets" creados. Cada "widget" se diseñó para mostrar la información de las respuestas a los downlinks que recibe el controlador. Estos downlinks son los comandos enviados al controlador desde la aplicación. El código que se utilizó para la creación de los "widgets" se puede apreciar en el apéndice C.

Para estas pruebas se utilizó la versión gratis para estudiantes que posee Ubidots. Esta versión tiene la limitación de que no permite realizar downlinks de manera libre, si no que cada downlink es desencadenado cuando se cumple cierta condición configurable en las salidas del decoder de TTN. Además, los mensajes a enviar por downlink deben estar predeterminados en Ubidots (hard-coded). Por ejemplo, se podría programar que la aplicación envíe el downlink correspondiente al comando SET ON cuando determinada salida del decoder de TTN sea mayor a un valor predefinido. Sin embargo, el usuario del controlador diseñado debería tener la posibilidad de enviar comandos en cualquier momento, sin estar restringido a la evolución de las salidas del decoder. Por otro lado, existen comandos como "SET MOMENT DD/MM/YYYY HH:MM:SS" que no pueden estar predeterminados en la aplicación, dada la enorme cantidad de parámetros posibles. Es por estos motivos que la versión gratuita de Ubidots no resulta apropiada para implementar el envío de downlinks al controlador. Por ende, el envío de downlinks se realiza utilizando la consola de TTN. Dicha consola toma cadenas de bytes ingresadas

1

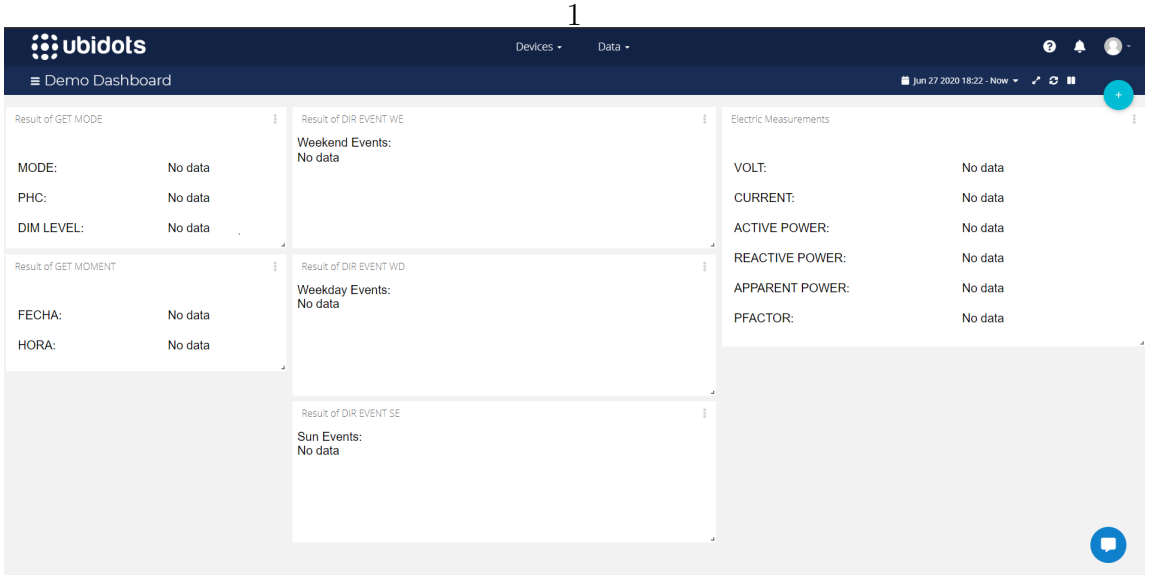


Figura 6.7: Tablero en Ubidots sin datos en sus widgets.

por el usuario y las envía como downlink en la siguiente ventana de recepción del dispositivo end-point. Cabe destacar que para utilizar esta consola se debe conocer el comando a enviar en su formato codificado. Es por esta razón que se creó un programa en Octave, llamado *encoder*, que toma como entrada el comando en su versión decodificada y devuelve su versión codificada, para ser ingresada en la consola de TTN. El mismo se encuentra en el repositorio de git del proyecto [2].

Con respecto a los uplinks, cuando el LoRa transceiver envía un mensaje a la aplicación TTN, el decoder de TTN recibe dicho mensaje, así como el número de puerto al cual fue enviado. Como se explicó en la sección 5.8.3, se programó el controlador para que envíe las respuestas a cada tipo de comando en un puerto determinado. Por ejemplo, las respuestas del controlador al comando "GET MODE" las envía al puerto 20, mientras que las respuestas a "GET MOMENT" las envía al puerto 21. De esta manera, desde la aplicación en TTN se conoce qué comando generó la respuesta recibida. Para estas pruebas, se programó el decoder de TTN para crear la variable "Datos", la cual es la concatenación del número de puerto (Port) y los bytes del mensaje (Bytes). Luego, se da dicha variable como salida hacia Ubidots. En la figura 6.8 se representa esquemáticamente con un ejemplo este funcionamiento del decoder de TTN.

Es importante destacar que TTN limita el tamaño de la variable Datos a un máximo de 8 bytes. Por lo tanto, las respuestas que envía el controlador que pueden superar este tamaño se dividen en más de un mensaje, cada uno enviado a un puerto específico. Estos comandos son DIR EVENT WD y DIR EVENT WE, los cuales generan una respuesta de tamaño 3 bytes por evento. Por ejemplo, si el arreglo correspondiente a WD tiene 3 eventos la respuesta del controlador necesita 9 bytes para enviar su información, lo cual supera el tamaño que puede

[2] `https://gitlab.com/TomasArrivillaga/columint`

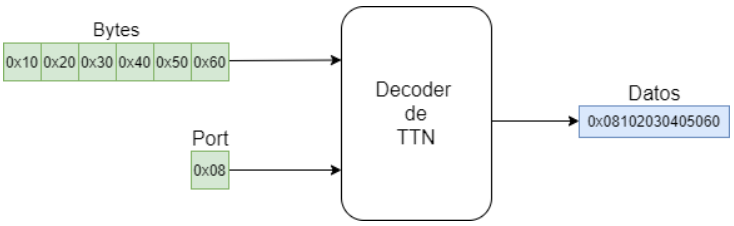


Figura 6.8: Diagrama del funcionamiento del decoder de TTN.

tener la variable Datos. Entonces, se divide el mensaje original en paquetes de 6 bytes o menos y en cada mensaje se aumenta el número de puerto. En este caso se enviarían los primeros dos eventos al puerto 1 y el tercer evento al puerto 2. Es por esta razón que se reservan 5 puertos para las respuestas al comando DIR EVENT WD y otros 5 para DIR EVENT WE. La cantidad de puertos asignados para estas pruebas hace que la cantidad máxima de eventos por arreglo sea 10 (en lugar de los 32 que permite almacenar el controlador).

Cada "widget" muestra la información de las respuestas a un tipo de comando. Por lo tanto, cada "widget" tiene asociado uno o varios puertos, siguiendo la información de la tabla 5.2. Estos "widget" leen continuamente la variable Datos, decodificando su contenido y desplegándolo en el caso que el byte más significativo de Datos coincida con el puerto asociado a dicho "widget".

Para poder enviar comandos al controlador en un momento cualquiera, es necesario que el controlador sea un dispositivo end-point clase C. No obstante, TTN no soporta dispositivos end-point clase C, razón por la cual se programó el controlador para funcionar como un dispositivo end-point clase A, que envía uplinks cada minuto (como se mencionó en la sección 5.8.2).

Una vez realizados los pasos anteriores, se procedió a probar la comunicación entre el LoRa transceiver y la aplicación TTN. Para esto se cargó un programa de prueba en la placa MCU que solo hace que el LoRa transceiver envíe uplinks cada minuto. Desde la consola de TTN se constató que el conjunto placa MCU-LoRa transceiver como dispositivo end-point se había activado de manera satisfactoria en la aplicación. Sin embargo, se notó que siete de cada ocho uplinks que se enviaban no llegaban a la aplicación. Luego de investigar para resolver el problema, se encontró que el gateway utilizado solo recibe mensajes en los primeros ocho canales de los sesenta y cuatro disponibles, mientras que el RN2903 está configurado por defecto para utilizar todos los canales disponibles. Por lo tanto, se configuró el RN2903 para que solo envíe uplinks en los primeros ocho canales, utilizando el comando "mac set ch status 'channel ID' 'status"'. De esta manera se solucionó el problema de la pérdida de paquetes. Con el nuevo ajuste, se constató desde la consola de TTN que todos los uplinks que se envían llegan a la aplicación. A su vez, se enviaron downlinks de prueba desde la consola de TTN y se verificó mediante el uso de breakpoints que el LoRa transceiver los recibía en la primera ventana de recepción disponible.

Para probar el uso de los puertos y la decodificación de uplinks desde Ubidots, se cargó la placa MCU con un programa de prueba que da respuestas fijas hard-

coded a los comandos que recibe. Se verificó desde Ubidots que la decodificación de uplinks se realiza de forma correcta. En la figura 6.9 se muestra el tablero de Ubidots generado en esta prueba, luego de enviar desde TTN al LoRa transceiver todos los comandos que generan respuestas.

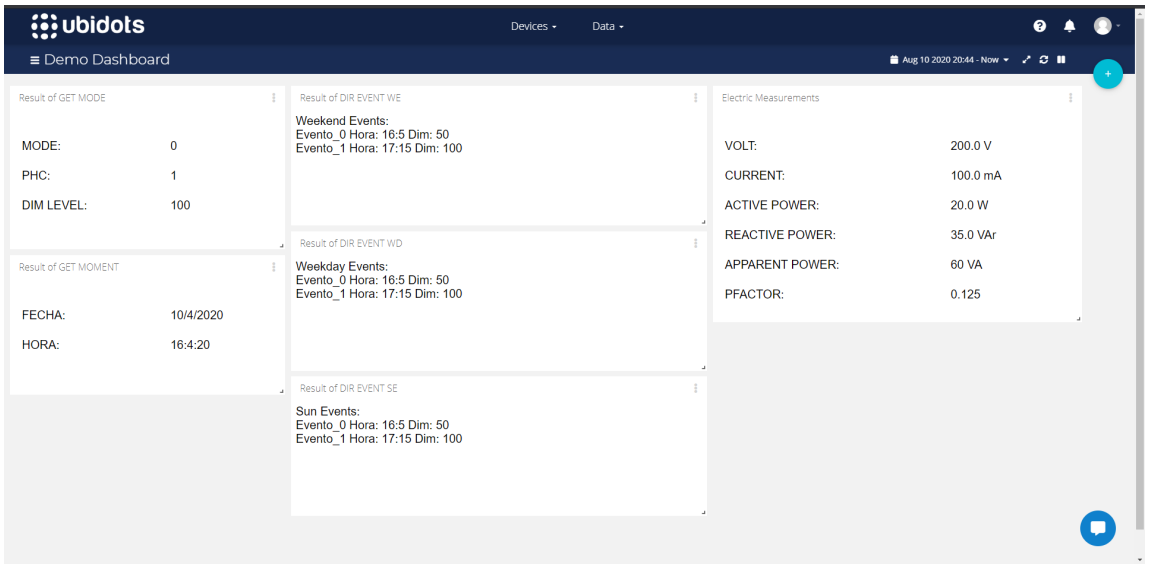


Figura 6.9: Tablero en Ubidots con datos en sus widgets.

6.3.3. Circuito fotosensible

Para probar el circuito fotosensible en esta etapa, se armó el circuito en un protoboard y se alimentó a través de un pin de 3,3 V de la placa MCU. A su vez, este fue alimentado por el puerto USB desde una PC. Para realizar la prueba se varió la luz recibida por el fotorresistor de la fotocélula, haciendo uso de una luz led en una habitación oscura. Se midió la tensión de salida de la fotocélula con un multímetro digital. Se definieron los valores de tensión que corresponden a umbral de luz y umbral de oscuridad (ver figura 5.4). Se obtuvo la tabla 6.2.

Umbral	Lectura ADC (14 bits)	Tensión
Oscuridad	7000	1.41V
Luz	9000	1.81V

Tabla 6.2: Umbrales de luz y oscuridad definidos.

6.3.4. Circuito de acondicionamiento

El circuito de acondicionamiento fue armado en un protoboard, con la entrada PWM conectada a la placa MCU y la salida en vacío. El amplificador operacional fue alimentado con una fuente de continua de 12 V y el MCU se alimentó a través del puerto USB de la placa MCU. El objetivo de esta prueba es medir la tensión

de salida al cambiar el ciclo de trabajo de la entrada. Para medir la entrada y la salida, se utilizó un osciloscopio digital de dos canales; uno de los canales se usó para medir la entrada y otro para la salida. Para medir ambos puntos se utilizaron puntas de 10 MΩ.

Se configuró el MCU para dar como salida un PWM de 1.64kHz por el puerto 2.5 con ciclo de trabajo configurable con comandos, haciendo uso del sisem_shell. En particular se utilizó el comando SET DIM para variar el ciclo de trabajo. Durante la prueba, se varió el ciclo de trabajo desde 0 hasta 1, obteniendo a la salida los valores esperados. En el apéndice D se muestran las imágenes obtenidas en osciloscopio para los valores de ciclo de trabajo 0, 25, 50, 75 y 100.

6.3.5. Circuito On/Off

El circuito On/Off no se probó de forma modular en esta etapa (fuera del PCB). Se utilizó el programa LTSpice para validar el circuito, diagramando el mismo como muestra la figura 6.10. Cabe destacar que el relé está representado como un resistor de 17 Ω, que es la resistencia del bobinado del relé (ver sección 4.2.6). En particular, se simuló el mismo en dos escenarios: cuando V_{pin} = 3,3 V y cuando V_{pin} = 0 V. Los resultados son los que muestra la gráfica de la figura 6.11, donde la señal en azul es V_{pin} y la señal en rojo es el voltaje en bornes del bobinado del relé (V_{RLY}). Como se explicó en la seccion 4.2.6, el relé cierra su contacto cuando la tensión en bornes su bobinado supera los 2,25 V. Además, el relé abre su contacto cuando la tensión en bornes de su bobinado no alcanza los 0,15 V. Observando la gráfica de la figura 6.11, se puede concluir que el relé abrirá su contacto cuando V_{pin} sea 0 V ($V_{RLY} < 0{,}15$ V), y que el relé cerrará su contacto cuando V_{pin} sea 3,3 V ($V_{RLY} > 2{,}25$ V).

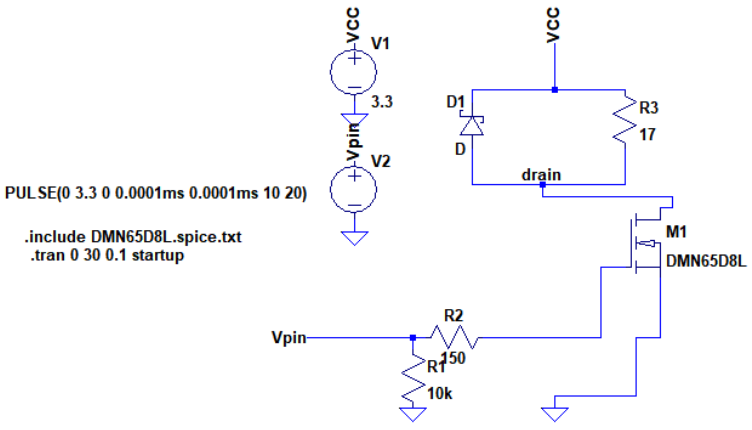


Figura 6.10: Esquemático del circuito simulado en Spice.

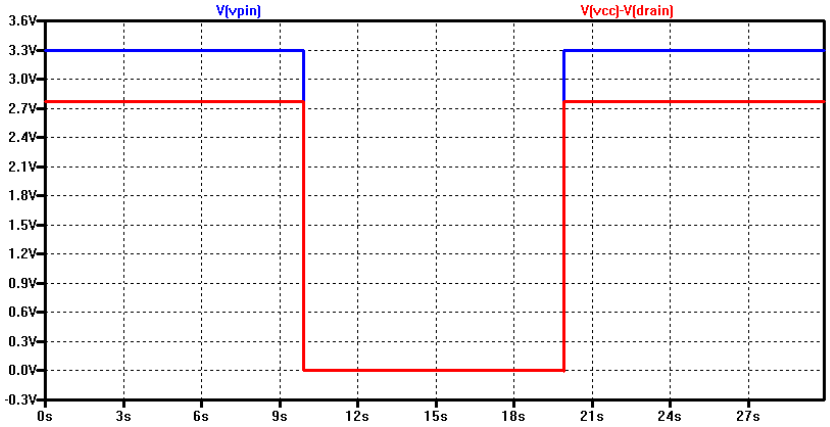


Figura 6.11: Señales obtenidas en la simulación Spice. V_{pin} en azul y V_{RLY} en rojo.

6.3.6. Circuito de medida de consumo

Para probar el módulo circuito de medida de consumo en esta etapa, se utilizó la placa de evaluación ASEK71020 que proporciona Allegro Microsystems [36]. En la figura 6.12 se muestra una foto de la misma. Este módulo fue elegido para realizar pruebas de manera simple, haciendo uso de los conectores de fácil acceso instalados en el ASEK71020. Esta placa de evaluación cuenta con el chip ACS71020, un regulador lineal de tensión, capacitores de desacople y resistores de pull-up para la comunicación I^2C. En la figura 6.13 se puede ver el diagrama esquemático de los componentes presentes en esta placa.

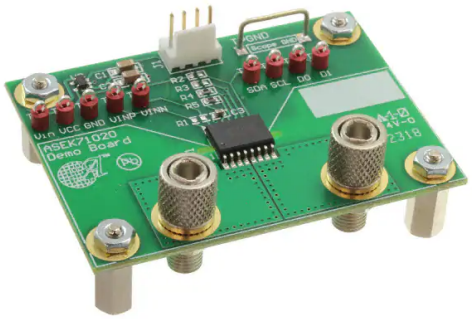


Figura 6.12: Foto del ASEK71020.
Fuente: digikey.com.

Los resistores R_2 y R_3 no vienen soldados en la placa ya que no son necesarios en esta versión del chip. El resistor R_1 tampoco viene soldado, puesto que el valor de resistencia a utilizar depende de la aplicación. Por otra parte, los resistores de pull-up (R_4 y R_5) vienen soldados en la placa, y tienen un valor de $10k\Omega$.

Para realizar las pruebas, se usó un circuito como el que se muestra en la

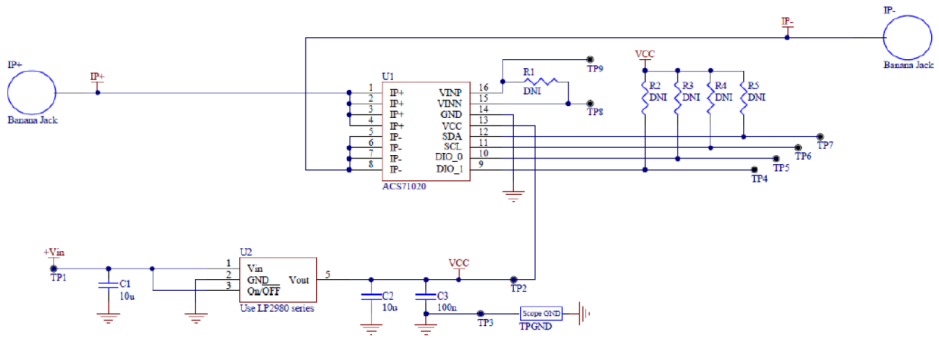


Figura 6.13: Esquemático del ASEK71020.
Fuente: Allegro Microsystems.

figura 6.14. Es importante aclarar que, una vez más, la topología del circuito es la recomendada por el fabricante en la hoja de datos.

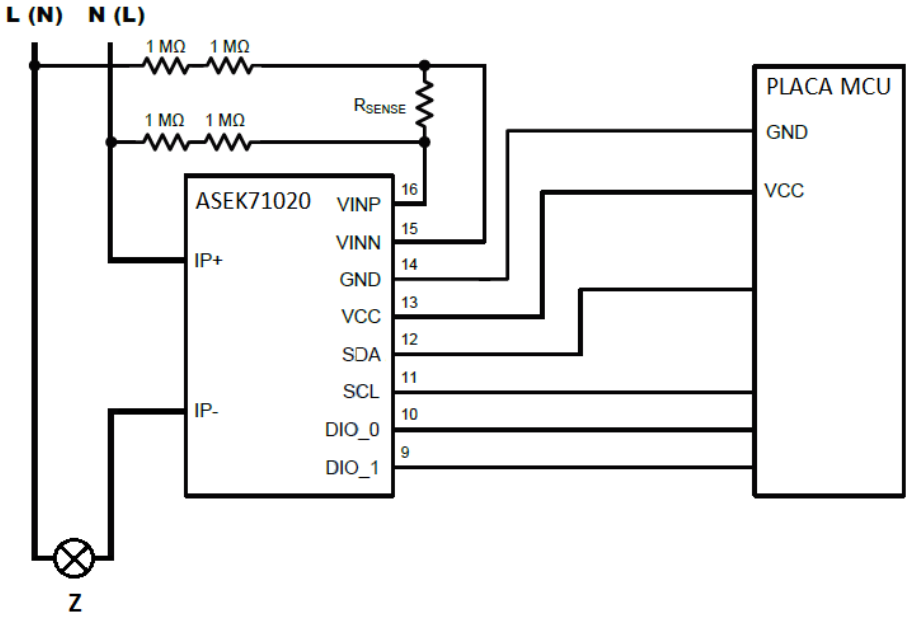


Figura 6.14: Esquemático del circuito utilizado para realizar las pruebas del ACS71020 con el ASEK71020.
Fuente: realización propia, basado en un esquema presente en la hoja de datos del ACS71020.

Se colocaron distintas cargas, las cuales fueron bombillas incandescentes de los siguientes valores de potencia nominal: 40W, 75W y 100W. Se tomaron cuatro lecturas de las magnitudes listadas a continuación, en caso de bombilla apagada y encendida:

- Tensión (V).

- Promedio de corriente (I).
- Promedio de potencia activa (P).
- Potencia aparente (S).
- Potencia reactiva (Q).
- Factor de potencia (FP).

Dichas lecturas se presentan en las tablas 6.3, 6.4 y 6.5. Adicionalmente, se midió la tensión y corriente consumida por las tres cargas, utilizando un multímetro digital. La medida de tensión fue 227 V en todos los los casos, mientras que las corrientes fueron 414 mA, 370 mA y 181 mA para las bombillas encendidas de 100 W, 75 W y 40 W respectivamente. En caso de bombillas apagadas, la medida de corriente fue 0 mA en todos los casos. Se toman estos valores como referencia para la evaluación de las medidas obtenidas del ACS71020. En cuanto a la potencia activa, se toma como valor de referencia el producto de la tensión y corriente de referencia, esto es 93,978 W, 83,990 W y 41,087 W.

Estado	V (V)	I (mA)	P (W)	S (VA)	Q (VAR)	FP
OFF	223.79	58.723	0	12.866	7.2720	-0.19775
	223.94	58.723	0	13.425	5.5939	0.18750
	224.37	58.723	0	10.628	6.7126	-0.21240
	224.78	58.723	0	12.866	6.1532	-0.19043
ON	225.28	414.72	90.621	105.16	10.628	-0.25000
	224.13	412.89	90.621	104.05	10.628	-0.25000
	223.66	412.89	91.180	105.72	9.5092	0.74854
	224.09	412.89	90.627	124.18	0	0.74853

Tabla 6.3: Medidas realizadas con la bombilla de potencia nominal 100 W. Valores de referencia en ON: 227 V, 414 mA, 93,978 W. En OFF: 227 V, 0 mA, 0 W.

Estado	V (V)	I (mA)	P (W)	S (VA)	Q (VAR)	FP
OFF	229.44	58.723	0	12.306	4.4751	-0.17725
	227.67	56.888	0	12.306	6.7126	-0.19727
	226.83	56.888	0	12.866	8.3908	0.18164
	225.90	55.053	0	11.188	8.9502	0.19970
ON	224.11	370.69	81.111	96.774	10.069	-0.25000
	221.57	370.69	81.111	95.096	10.069	-0.25000
	217.19	372.52	81.671	74.398	8.9502	-0.25000
	216.78	372.52	81.671	71.602	0	0.74853

Tabla 6.4: Medidas realizadas con la bombilla de potencia nominal 75 W. Valores de referencia en ON: 227 V, 370 mA, 83,990 W. En OFF: 227 V, 0 mA, 0 W.

Estado	V (V)	I (mA)	P (W)	S (VA)	Q (VAR)	FP
OFF	224.24	58.723	0	11.188	5.5939	0.18408
	224.00	58.723	0	12.307	7.2720	-0.21240
	223.92	56.888	0	12.307	8.9502	-0.20654
	224.00	56.888	0	12.307	7.2720	0.17822
ON	224.78	181.67	39.716	30.207	7.2720	-0.25000
	225.82	183.51	40.835	30.766	0	0.74756
	225.99	183.51	40.276	31.885	0	-0.25000
	225.97	183.51	40.835	29.088	0	-0.24561

Tabla 6.5: Medidas realizadas con la bombilla de potencia nominal 40 W. Valores de referencia en ON: 227 V, 181 mA, 41,087 W. En OFF: 227 V, 0 mA, 0 W.

Con respecto a la medida de tensión, se aprecia que la máxima desviación porcentual respecto al valor de referencia es de 4,5 %. Esta medida de tensión se obtiene leyendo el registro 0x20 del ACS71020. Como se explicó en la sección 5.9, el registro 0x26 contiene un promedio de las medidas del último segundo de la corriente o del voltaje, pero no de ambos simultáneamente. En el presente proyecto, se utiliza el promedio de corriente y no el de tensión. Si se utilizara el promedio de tensión, las medidas de tensión serían aún mejores. Con respecto a la corriente, cuando las bombillas están apagadas el chip mide cerca de 60 mA. Esto muestra que en caso de corrientes en el orden de las decenas de mA no se tienen buenos resultados utilizando el ACS71020. Sin embargo, en caso de bombillas encendidas, las máximas desviaciones porcentuales respecto a los valores de referencia son 0,27 %, 0,68 % y 1,39 % para las bombillas de 100 W, 75 W y 40 W respectivamente. Este resultado demuestra que el ACS71020 obtiene mejores resultados en la medida de corriente conforme aumenta la corriente a medir. Esto es porque cuando aumenta la corriente, mejora la relación entre la señal a medir y el ruido. En cuanto a la potencia activa, cuando las bombillas estaban apagadas, la medida obtenida fue de 0 W en todos los casos. Por lo tanto, esta medida es la adecuada para identificar si la carga está encendida o apagada. En caso de bombillas encendidas, las máximas desviaciónes porcentuales respecto a los valores de referencia son 3,57 %, 3,42 % y 3,34 % para las bombillas de 100 W, 75 W y 40 W respectivamente. El grupo esperaba en este caso resultados aún mejores, teniendo en cuenta que la medida de potencia activa se obtiene promediada.

Se puede ver en las tablas que las medidas de potencia aparente, reactiva y factor de potencia no son consistentes en ningún caso. Por ejemplo, en el caso de bombilla encendida de potencia nominal 100 W, el valor de potencia reactiva es a veces cercano a 10 VAR y a veces nula. La situación es similar para las otras bombillas. Por otra parte, para las bombillas de potencia nominal 75 W y 40 W, las medidas de potencia aparente son en algunos casos menores en módulo que la potencia activa. El grupo desconoce la causa de dichas inconsistencias.

6.4. Pruebas modulares de hardware en el PCB

Como se mencionó anteriormente, los distintos módulos hardware fueron probados en los PCBs finales tras ser validados fuera de dichos PCBs. A continuación se desarrolla sobre las distintas pruebas realizadas en esta etapa para cada módulo.

6.4.1. Circuito de alimentación

Para probar el circuito de alimentación se midieron las líneas de alimentación de la placa y los resultados fueron los correctos: 12 V y 3,3 V.

6.4.2. LoRa transceiver

Teniendo en cuenta que el funcionamiento del LoRa transceiver no es alterado por estar en el PCB, este módulo ya es validado por la prueba fuera del PCB correspondiente. Sin embargo, se realizaron pruebas similares a las ya realizadas (envío y recepción de mensajes) y se confirmó el buen funcionamiento.

6.4.3. Circuito fotosensible

Para realizar la prueba del circuito fotosensible se realizó un procedimiento igual al realizado en las pruebas fuera del PCB. Sin embargo, en este caso la fotocélula fue alimentada con el circuito de alimentación diseñado. Se expuso a la fotocélula a los mismos niveles de luminosidad usados para las pruebas fuera del PCB y se constató que las tensiones medidas son las mismas.

6.4.4. Circuito de acondicionamiento

La prueba del circuito de acondicionamiento fue realizada conectando el controlador a la red eléctrica, sin estar conectado a la luminaria. Es decir, se conectaron las terminales Línea y Neutro del controlador directamente a la red. Esto es para evitar cualquier daño al driver de la luminaria en caso de existir algún error en el diseño o el ensamblado. Se realizó el mismo procedimiento que en la prueba fuera del PCB correspondiente: se midió con osciloscopio en dos canales, uno en la salida PWM del MCU y otro en el puerto 1-10V del controlador. Se mandaron diversos comandos de dimerización al controlador utilizando el sisem_shell y en todos los casos el resultado fue el esperado. En particular, se dimerizó a nivel 0 %, 25 %, 50 %, 75 % y 100 % y las gráficas obtenidas fueron iguales a las de las figuras D.1, D.2, D.3, D.4 y D.5 (apéndice D). Por lo tanto, se validó este módulo.

6.4.5. Circuito On/Off

Para la realización de la prueba del circuito On/Off se tomó la misma precaución que en el caso del circuito de acondicionamiento: se conectó el controlador directamente a la red. Se utilizó un multímetro digital midiendo continuidad entre Línea y Carga. Esto es porque el circuito debe cortocircuitar Línea y Carga cuando

se desea prender la luminaria y debe abrir este contacto cuando se desea apagarla. Teniendo en cuenta que tras un reset el controlador indica que la luminaria debe estar apagada, inmediatamente después de alimentar al controlador no debe haber continuidad. Efectivamente se corroboró esto y se procedió a enviar el comando SET ON haciendo uso del sisem_shell. Luego, se comprobó que sí había continuidad entre Línea y Carga. Adicionalmente se enviaron comandos de dimerización entre 10 % y 100 % y como era esperado la continuidad siguió existiendo. Por último se enviaron comandos de dimerización entre 0 % y 9 % y se perdió esta continuidad. Por lo tanto se validó este módulo.

Una vez realizadas las pruebas de validación de los circuitos de alimentación, acondicionamiento y On/Off, se consideró seguro conectar el controlador a la luminaria. Entonces, se conectó el controlador a la luminaria y se enviaron diversos comandos usando el sisem_shell, obteniendo resultados satisfactorios. Se comprobó el manejo del encendido, apagado y dimerizado de la luminaria.

6.4.6. Circuito de medida de consumo

Para realizar las pruebas del módulo circuito de medida de consumo se conectó el controlador a la luminaria. Utilizando el sisem_shell se enviaron al controlador los comandos para obtener medidas eléctricas. Al utilizar el comando GET CURR, se observó que la corriente medida al encender o apagar la luminaria permanecía constante y alrededor de 28 mA. La causa de este comportamiento se evidenció al analizar el layout del PCB de la placa inferior (ver apéndice B): la rama del circuito sensada por el ACS71020 está en paralelo con la luminaria. Es decir, la corriente consumida por la luminaria no pasa por el sensor de corriente del ACS71020. En la figura 6.15 se aprecia un diagrama esquemático de la situación. La corriente que mide el ACS71020 es la que pasa por N2, la cual es la que consume solo el controlador. Por otra parte, en la figura 6.16 se muestra de forma esquemática una forma correcta de conectar el ACS71020 para que este consiga medir la corriente consumida por la luminaria y el controlador.

Tras tomar conciencia de este error de conexión, se modificó la placa inferior de manera tal de llegar a un circuito como el de la figura 6.16. Esta modificación se hizo sobre el PCB con el que ya se contaba, y no se mandó a fabricar una nueva versión.

Con la versión modificada del PCB, se enviaron al controlador los comandos para obtener medidas eléctricas utilizando el sisem_shell, con la luminaria dimerizada al nivel 0 (apagada), 25, 50, 75 y 100. Se obtuvo cuatro valores de cada magnitud y los resultados se muestran en la tabla 6.6. Adicionalmente, se midó tensión y corriente usando un multímetro digital. La tensión medida con el multímetro en todos los casos fue 227 V, mientras que las corrientes medidas fueron 4,3 mA, 62,8 mA, 108,8 mA, 161,5 mA y 171,8 mA para los niveles de dimerización 0, 25, 50, 75 y 100 respectivamente. Se toman estos valores como referencia para la evaluación de las medidas obtenidas del circuito de medida de consumo.

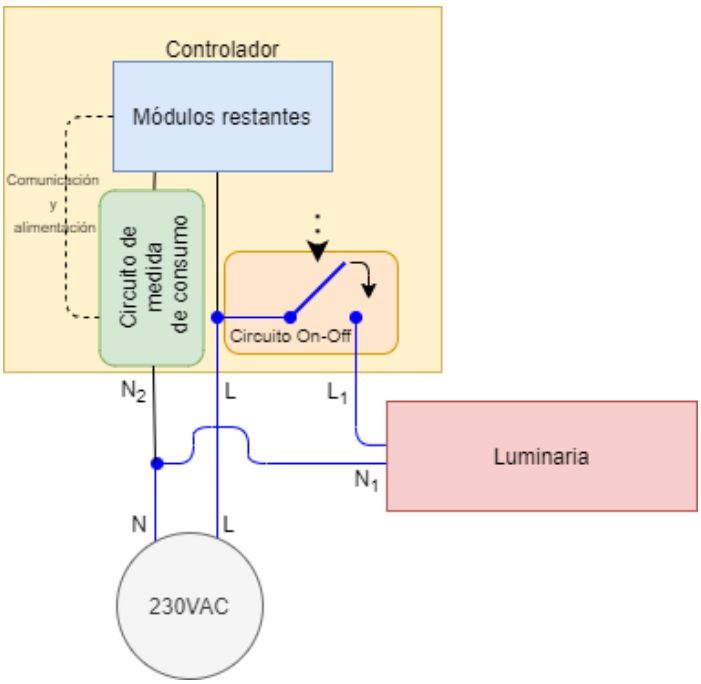


Figura 6.15: Diagrama que muestra la conexión del circuito de medida de consumo previo al ajuste realizado. Se muestra en azul el bucle de corriente consumida por la luminaria.

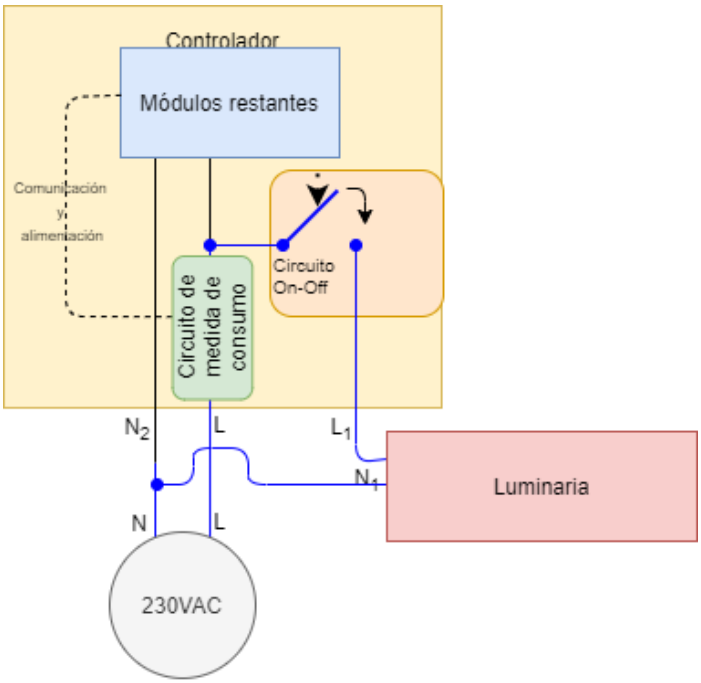


Figura 6.16: Diagrama que muestra la conexión del circuito de medida de consumo luego del ajuste realizado. Se muestra en azul el bucle de corriente consumida por la luminaria.

Estado	V (V)	I (mA)	P (W)	S (VA)	Q (VAR)	FP
DIM 100	227.39	172.50	34.776	38.695	21.552	0.21387
	231.06	172.50	32.817	39.185	22.531	0.21338
	237.80	172.50	33.307	41.146	18.123	0.21729
	211.78	172.50	33.307	38.695	20.082	0.22510
DIM 75	220.79	159.65	30.858	39.674	18.613	0.22559
	237.33	161.49	27.919	39.185	20.572	0.22656
	245.82	163.32	27.919	36.246	15.674	0.22556
	246.75	161.49	27.919	36.736	16.653	0.21875
DIM 50	207.42	110.11	19.592	19.592	16.164	0.20947
	226.52	110.11	19.102	19.592	15.184	0.19727
	227.48	110.11	19.592	20.572	16.653	0.20654
	226.70	110.11	19.102	30.858	14.694	0.21533
DIM 25	215.24	69.733	7.8369	84.737	12.245	0.12012
	211.24	67.898	6.8573	83.267	16.164	0.13135
	228.85	67.898	6.8573	65.144	16.653	0.17675
	229.07	67.898	6.8573	62.695	16.653	0.13086
DIM 0	230.50	27.526	0	5.8777	6.3675	-0.083496
	230.76	29.361	0	5.38788	4.8981	0.076660
	230.57	27.526	0	7.3471	6.8573	-0.049804
	231.52	29.361	0	7.3471	6.3675	0.083008

Tabla 6.6: Medidas realizadas con la luminaria. Valores de referencia: 227 V en todos los casos; 171,8 mA, 161,5 mA 108,8 mA, 62,8 mA y 4,3 mA para dimerizado 100, 75, 50, 25 y 0.

Con respecto a la medida de tensión, se aprecia que la máxima desviación porcentual respecto al valor de referencia es 8,70 %. Además, la variación en las medidas es notoria, obteniendo tensiones desde 207,42 V hasta 246,75 V. Sin embargo, el promedio de todas las medidas de tensión es 227,73 V, lo cual coincide con el valor de referencia. Por lo tanto, si se configurara el chip para promediar tensión, los valores obtenidos del registro 0x26 (tensión promediada) serían correctos. En cuanto a la corriente, si la luminaria está apagada se obtiene una medida de corriente de casi 30 mA, cuando el valor de referencia es 4,3 mA. Esto reafirma lo concluido en la sección 6.3.6: el chip ACS71020 no obtiene buenos resultados cuando la corriente está en el orden de las decenas de mA. Cuando la luminaria está encendida, las máximas desviaciones porcentuales respecto a los valores de referencia son 0,41 %, 1,1 %, 1,2 % y 11,0 % para los niveles de dimerización 100, 75, 50 y 25 respectivamente. Al igual que en las pruebas de la sección 6.3.6, la medida de corriente mejora conforme aumenta el valor de la misma. Con respecto a la potencia activa medida, esta aumenta conforme aumenta el nivel de dimerización, y es 0 cuando la luminaria está apagada, lo cual coincide con lo esperado. Además, la potencia nominal de la luminaria utilizada a nivel de dimerización 100 es 35 W, lo cual es cercano al valor medido en este caso. Cabe notar que no se tienen valores de referencia para los casos de menor nivel de dimerización (la carga

no es puramente resistiva por lo que P no es el producto de V e I).

Se puede ver en la tabla que las medidas de potencia aparente, reactiva y factor de potencia no son consistentes en ningún caso, tal como sucedió en las pruebas de la sección 6.3.6.

6.5. Pruebas finales

Se adjuntan dos videos en los cuales se realizan pruebas de funcionamiento del controlador. En el primero se utiliza el sisem_shell para enviar comandos y obtener respuestas del controlador, el cual se hace funcionar en los distintos modos [3]. En el segundo se prueba la comunicación LoRaWAN, enviando comandos al controlador desde TTN y observando sus respuestas en Ubidots [4].

[3] `https://youtu.be/123EwKuBNvw`
[4] `https://youtu.be/x6nTDueM2KA`

Capítulo 7

Conclusiones

7.1. Introducción

En este capítulo se realiza un análisis general del desarrollo del proyecto. A su vez, se evalúan los resultados obtenidos con respecto a los objetivos definidos en el alcance del proyecto (sección 1.3) y se realiza una conclusión final. Por último, se plantean mejoras al controlador y ampliaciones al proyecto que se podrían realizar en trabajos futuros.

7.2. Análisis general del desarrollo del proyecto

Se diseñó y construyó un controlador de luminaria para alumbrado público, capaz de comandar una luminaria utilizando el protocolo 1-10V. El control de la luminaria se basa en diferentes modos de funcionamiento, según la fuente de información utilizada para la toma de decisiones. Estas fuentes son: instrucciones directas desde un servidor central, plan horario, reloj astronómico y fotocélula. Además, el controlador se comunica de forma inalámbrica con el Servidor Central usando la tecnología LoRaWAN. Esta comunicación permite que el controlador reciba comandos de control y responda a ellos cuando corresponda. Por otra parte, el controlador toma medidas eléctricas de la luminaria, las cuales reporta al Servidor Central cuando se le solicita.

Se concibió el controlador como la interconexión de diversos módulos hardware, cada uno con una función específica. Luego de definir la función de cada módulo, se diseñaron los circuitos que los implementan. Este diseño se dividió en dos etapas: en primer lugar, se realizó el diseño conceptual de cada circuito. En segundo lugar, se dimensionaron y eligieron los componentes a utilizar. Estos circuitos fueron probados individualmente, con resultados satisfactorios. Para la interconexión de los módulos hardware se diseñaron tres PCBs, los cuales contienen los circuitos que implementan dichos módulos. El controlador está conformado por estos tres PCBs conectados a una base con conector compatible ANSI C136.41 y cubiertos por una tapa. Las pruebas individuales realizadas a los módulos hardware se repitieron tras ensamblar los PCBs, obteniendo nuevamente resultados satisfactorios.

Se utilizó el microcontrolador MSP432P401R como unidad central del controlador. Se desarrolló el software en el entorno de desarrollo integrado CCS, utilizando el lenguaje C. Al igual que en el hardware, el software consta de diversos módulos, cada uno con funciones específicas. Estos módulos se dividen en software dependiente e independiente de hardware. La arquitectura de software usada fue round robin con interrupciones. El software logra controlar con éxito los periféricos del MCU, los cuales se utilizaron para comunicarse con el resto de los módulos hardware. Además, se implementó exitosamente la lógica de control de la luminaria basándose en modos de funcionamiento según la fuente de información utilizada.

Se concibió el controlador como parte de un sistema más amplio, el cual consiste de un servidor central, una infraestructura de comunicación inalámbrica LoRaWAN, el controlador y una luminaria. Por lo tanto, para probar el funcionamiento del controlador se construyó una red LoRAWAN utilizando el servicio The Things Network. Una vez construida dicha red, se pudo probar el controlador como dispositivo end-point de la misma. Adicionalmente, se hizo una aplicación de prueba utilizando TTN y Ubidots que permite comunicarse con el controlador utilizando los comandos. Se consiguió que el controlador se comunique con el servidor de The Things Network de forma consistente y se observó la información de sus uplinks en Ubidots.

7.3. Evaluación de los resultados respecto al alcance del proyecto

A continuación, se listan los puntos del alcance, evaluando si se lograron o no en el producto final. En base a estos juicios se reflexionará acerca del éxito del proyecto.

- *Comandar la luminaria para encenderla y apagarla. A su vez, utilizar el protocolo 1-10V para dimerizarla.*

  El controlador logra controlar el encendido y apagado de la luminaria utilizando el circuito On/Off. También es capaz de utilizar el protocolo 1-10V para indicar a la luminaria en qué nivel de luz debe estar. Por lo tanto, este punto se realizó con éxito.

- *Recibir las instrucciones de encendido, apagado y dimerizado de forma inalámbrica desde un servidor central y ejecutarlas. La tecnología de comunicación a utilizar es LoRaWAN sobre LoRa.*

  El controlador forma parte de una red LoRaWAN como dispositivo end-point clase A. Recibe instrucciones a través de la red y actúa de manera acorde. El controlador también es capaz de responder a los comandos utilizando la red. Por lo tanto, este punto se realizó con éxito.

- *Diseñar un reloj astronómico, el cual calcule la hora de amanecer y atardecer para una fecha y lugar geográfico dado. Esta información sirve para saber si a determinada hora es de día o de noche.*

El cálculo de la hora de amanecer y atardecer por parte del reloj astronómico implementado se probó con un error máximo de 12 minutos para la ciudad de Montevideo. El grupo considera que este error es aceptable dada la aplicación en la que se utilizan los resultados. Se programó el controlador para sumarle 12 minutos a la hora del amanecer y restarle 12 minutos a la hora del atardecer, para estar cubiertos aún en el peor caso. Con este margen de seguridad se puede garantizar que la luminaria estará siempre encendida en horas de la noche cuando el controlador está en modo Reloj Astronómico. Por lo tanto, este punto se realizó con éxito.

- *Controlar la luminaria en función de la información generada por el reloj astronómico. Es decir, se pretende apagar la luminaria cuando el reloj astronómico indica que es de día y encenderla cuando es de noche.*

  Cada día se forma un plan horario con eventos que poseen las horas de amanecer (atrasada 12 minutos) y atardecer (adelantada 12 minutos) que calcula el reloj astronómico. El controlador apaga o enciende la luminaria en función de estos eventos cuando está en el modo Reloj Astronómico. Por lo tanto, este punto se realizó con éxito.

- *Controlar la luminaria a partir de instrucciones programadas de antemano, las cuales generen un plan horario que deberá ser utilizado por el controlador de forma autónoma, incluso si pierde comunicación. Estas instrucciones provienen de un servidor central.*

  El controlador posee dos planes horarios: uno para fin de semana y otro para entre semana. Estos planes horarios se configuran vía LoRaWAN utilizando comandos. Se controla la luminaria en función de estos planes horarios cuando el controlador está en modo Plan Horario. Por lo tanto, este punto se realizó con éxito.

- *Controlar la luminaria en función de la información recibida de un circuito fotosensible (o fotocélula). Es decir, se pretende apagar la luminaria cuando el circuito fotosensible indica que hay luz ambiente suficiente y encenderla cuando no la hay.*

  El controlador obtiene el promedio de las medidas de tensión de salida del circuito fotosensible del último minuto. Este promedio es comparado con umbrales de luz y oscuridad que se definieron. Cuando el controlador está en un modo que tiene en cuenta a la fotocélula, la enciende cuando el valor promediado es menor al umbral de oscuridad, y la apaga cuando es mayor al umbral de luz. Por lo tanto, este punto se realizó con éxito.

- *Diseñar el controlador basándose en diferentes modos de funcionamiento, los cuales se diferencian en la fuente de información que se usa en la toma de decisiones: instrucciones directas desde un servidor central, plan horario, reloj astronómico o fotocélula.*

  El controlador contiene cuatro modos de funcionamiento: Directo, Plan Horario, Reloj Astronómico y Fotocélula. La fuente de información principal de

estos modos son instrucciones directas de encendido, apagado y dimerizado del Servidor Central, un plan horario, el reloj astronómico y la fotocélula respectivamente. Por lo tanto, este punto se realizó con éxito.

- *Diseñar un sistema de alimentación que satisfaga las necesidades energéticas de todos los componentes del controlador.*

  El circuito de alimentación del controlador satisface las necesidades energéticas del mismo. Por lo tanto, este punto se realizó con éxito.

- *Diseñar un sistema capaz de tomar medidas eléctricas (por ejemplo consumo de potencia) de la luminaria.*

  El controlador es capaz de tomar medidas eléctricas de la luminaria. Estas son: voltaje, corriente, potencia activa, potencia reactiva, potencia aparente y factor de potencia. Sin embargo, las medidas de potencia reactiva, potencia aparente y factor de potencia difieren en gran medida de lo que se quiere medir. Por lo tanto, si bien los comandos de obtención de estas medidas están implementados se recomienda al usuario no utilizarlos. La medida de tensión no es consistente, por más que se está en el entorno de lo que se desea medir ($\pm 9\,\%$). No obstante, se observa que el promedio de las medidas coincide con lo que se desea medir, por lo que si se utilizara el promediado de tensión del chip disminuiría el error en el resultado leído. Por otra parte, la medida de corriente es precisa para valores de dimerización mayores a 50 %, situación en la cual se registró un error máximo del 1,2 % en la medida. A medida que el nivel de dimerización disminuye, la corriente consumida baja y el error en la medida comienza a ser mayor. Cuando la luminaria está apagada, el controlador mide una corriente de alrededor de 30 mA cuando el consumo real es en promedio 4,3 mA. Por lo tanto, el controlador solo es adecuado para medir corriente con precisión cuando la misma supera los 100 mA, que corresponde aproximadamente al 50 % de nivel de dimerización para la luminaria utilizada en este proyecto (35 W). En cuanto a la potencia activa, las medidas son consistentes y se encuentran en el rango esperado. A su vez, cuando la luminaria está apagada la medida reportada de potencia activa es cero y cuando está al 100 % es cercana al valor de potencia nominal de la luminaria.

  En conclusión, las medidas eléctricas que realiza el controlador no permiten conocer las magnitudes deseadas con gran precisión. No obstante, las medidas de tensión, corriente y potencia activa pueden ser útiles. Por ejemplo, la medida de potencia activa puede ser utilizada para inferir el nivel de dimerización de la luminaria o para detectar posibles fallas en la misma.

  Este punto del alcance es definitivamente un aspecto a mejorar del proyecto.

- *Instalar mecánicamente el controlador en una caja de protección adecuada, cumpliendo con el grado de protección IP 66.*

  El controlador tiene una tapa que protege los componentes electrónicos del mismo frente al polvo y a chorros potentes de agua. Por lo tanto, este punto

se realizó con éxito.

- *Sentar las bases para el acople de otros sensores al sistema.*

  El controlador diseñado se puede adaptar fácilmente para recibir datos desde otros sensores de manera inalámbrica. El trabajo realizado en el marco de la asignatura Redes de Sensores Inalámbricos (ver sección 1.4.2) puede considerarse una muestra de que la adaptación es sencilla. Por lo tanto, este punto se realizó con éxito.

En conclusión, el controlador cumple con la gran mayoría de los puntos del alcance. El único aspecto en el que no se lograron resultados satisfactorios fue el diseño de un sistema capaz de tomar medidas eléctricas de la luminaria.

7.4. Conclusión final

El controlador obtenido se podría utilizar para transformar al alumbrado público de una ciudad en un alumbrado público inteligente, ya que el mismo brinda la posibilidad de comandar las luminarias de forma inalámbrica. El controlador tiene diferentes modos de funcionamiento, algunos de los cuales le permiten trabajar de forma autónoma. En estos modos el controlador enciende o apaga la luminaria de acuerdo a un horario configurado, al nivel de luz ambiente o a la hora de amanecer y atardecer de cada día. Estos modos de funcionamiento dotan al controlador de una gran versatilidad.

Este proyecto puede considerarse un paso hacia el desarrollo de Montevideo como ciudad inteligente. La infraestructura de comunicación que se implementaría para la instalación de estos controladores permite la adición de sensores que mejoren la calidad de vida en la ciudad. Por ejemplo, un contador de tráfico, un sensor de nivel de basura en un contenedor o un sensor que reporte a tiempo real los espacios libres de estacionamiento en la calle.

Existen ciertos aspectos del controlador que pueden ser mejorados, por ejemplo, el circuito de medida de consumo del controlador. Por otro lado, el controlador forma parte de una red LoRaWAN como dispositivo end-point clase A. Esto hace que sea necesario que el controlador envíe uplinks periódicos sin otro objetivo que habilitar la recepción de comandos. Esta dinámica de comunicación implica que la red podría saturarse como consecuencia del exceso de mensajes, ya que podrían haber cientos de controladores enviando uplinks simultáneamente. Estos problemas se evitarían si el controlador formara parte de la red como dispositivo end-point clase C.

7.5. Trabajos futuros

El controlador logrado en este proyecto puede ser mejorado de diferentes maneras. En primer lugar, se podría lograr que el circuito de medida de consumo reporte medidas eléctricas más precisas. Por ejemplo, podría cambiarse el software de modo que reconfigure el ACS71020 para promediar la tensión o la corriente en

función de qué es lo que se desea medir. Es decir, si se desea medir tensión, el MCU configuraría el ACS71020 para que promedie tensión. Luego, se esperaría un tiempo predeterminado para que el chip acumule las muestras necesarias. Finalmente el MCU leería el registro 0x26, el cual tendría el promedio de las medidas de tensión del último segundo. De forma análoga se obtendría la medida de corriente. Sin embargo, esto no mejoraría las medidas de potencia aparente, reactiva y factor de potencia, las cuales no se logran obtener con precisión a través del circuito de medida de consumo utilizado. Tampoco cambiaría el hecho de que el circuito no mide de forma adecuada corrientes menores a 100 mA. Entonces, podría rediseñarse el circuito de medida de consumo del controlador, de manera que todas las medidas eléctricas sean correctas. Esto se podría lograr utilizando un chip similar al ACS71020 pero con aplicación a corrientes y potencias más bajas, o implementando un circuito de medida de consumo de diseño propio. Por ejemplo, se podría utilizar el ADC14 del MSP432 para que releve las señales de tensión y corriente, luego de haberlas acondicionado (llevarlas a niveles que puedan ser leídos por el MCU). Teniendo estas muestras de las señales de tensión y corriente, se podrían calcular las magnitudes eléctricas de tensión RMS, corriente RMS, potencia activa, reactiva, aparente y factor de potencia. Esta idea motivó al grupo a pensar un primer acercamiento a un diseño propio del circuito de medida de consumo, el cual se presenta en el apéndice E.

En segundo lugar, el controlador no hace uso de un WDT (del inglés Watch-Dog Timer) para reiniciar el sistema en caso de falla. Esta característica haría que el controlador sea más robusto ante fallas de funcionamiento, impidiéndole ejecutar bucles infinitos indeseados. Por otra parte, se podría utilizar el periférico RTC (del inglés Real Time Clock) del microcontrolador utilizado. Esto simplificaría el manejo de la fecha y hora del controlador, y además tiene la ventaja de recordar estos campos tras un reset.

Otra mejora a realizar podría ser modificar el software del controlador y el firmware del RN2903 para funcionar como dispositivo end-point clase C. La razón por la cual el controlador obtenido funciona como dispositivo clase A es que se utilizó The Things Network para implementar la red LoRaWAN. Este servicio, en el momento de publicación de este documento, no soporta dispositivos clase C en su versión online. Entonces, la transición a clase C del controlador implica que se utilice otro servicio para crear la red LoRaWAN. Desde el punto de vista del software del controlador, el único cambio a realizar sería la eliminación del envío periódico de uplinks.

A su vez, para transformar el controlador en un dispositivo confiable para su uso en la ciudad, es necesario realizar un análisis exhaustivo sobre las protecciones que debe tener frente a sobretensiones e interferencia electromagnética. De esta manera, podría garantizarse que el controlador no se dañe al estar sometido a sobretensiones, interferencia electromagnética u otros eventos indeseados.

Finalmente, se puede ampliar la utilidad del controlador trabajando en aspectos que quedaron por fuera del alcance del presente proyecto. Como primer ejemplo, se pueden agregar subredes de sensores inalámbricos que reporten sus magnitudes sensadas al controlador, para luego ser enviadas por este hacia el Servidor Central

vía LoRaWAN. Para lograrlo, se debería utilizar algunos de los pines libres de la placa MCU para configurar una UART y comunicarse con un transceiver capaz de recibir los datos de los sensores. También sería necesario asignar un puerto de comunicación LoRaWAN para cada tipo de dato obtenido de los sensores. En el trabajo presentado en la sección 1.4.2 se encuentra un ejemplo de una subred IEEE 802.15.4/6lowpan, cuyo nodo raíz se comunica vía UART con una placa con MCU y transceiver LoRa. El nodo raíz reporta vía UART los datos que recibe de los sensores, para que estos sean enviados a un servidor remoto vía LoRaWAN. Como segundo ejemplo de ampliación del proyecto, se puede desarrollar una aplicación web o móvil que sirva de interfaz gráfica entre el usuario y el controlador, facilitando el envío de comandos.

Esta página ha sido intencionalmente dejada en blanco.

Apéndice A

Cálculos para dimensionar las fuentes del circuito de alimentación

En este apéndice se realiza un análisis de la máxima potencia que pueden llegar a consumir los módulos hardware, con el objetivo de dimensionar las fuentes del circuito de alimentación. La fuente SPBW06f-03 es la que entrega 3,3 V y por lo tanto alimenta los siguientes módulos: placa MCU, LoRa transceiver, circuito fotosensible, circuito ON/OFF y PMC. La fuente MPM-10-12 es la que entrega 12 V; alimenta a la fuente SPBW06f-03 y al circuito acondicionador.

- Placa MCU: observando la hoja de datos del microcontrolador MSP432P401R se aprecia que el consumo del MCU es muy dependiente del programa que ejecute. Por lo tanto, se utilizó la herramienta Energy Trace que provee CCS para medir el consumo del MCU con el software del controlador. En la figura A.1 se muestra el resultado obtenido. La corriente que consume la placa MCU es siempre menor a 9,1 mA.

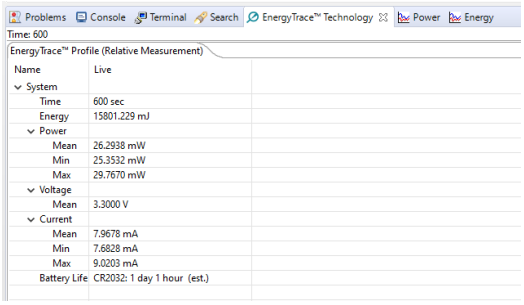


Figura A.1: Energy trace.

- LoRa transceiver: observando la tabla 2-5 de la hoja de datos del RN2903 se aprecia que en caso de transmisión el chip consume 124,4 mA. Por lo tanto, la corriente que consume este módulo es siempre menor a 124,4 mA.

- Circuito de medida de consumo: observando la tabla de la página 6 de la

hoja de datos del ACS71020 se deduce que la corriente que consume este chip está acotada superiormente por 14,0 mA.

- Circuito fotosensible: para hallar una cota superior en la corriente consumida por el circuito fotosensible, se toma el valor más pequeño de resistencia que puede presentar el fotorresistor (500 Ω). Entonces, la corriente que consume el circuito fotosensible está acotada superiormente por 0,3 mA ($\frac{3{,}3\ V}{10k\ \Omega+500\ \Omega}$).
- Circuito On/Off: para hallar una cota superior en la corriente consumida por el circuito On/Off se considera que la caída de tensión en la resistencia es la máxima posible: 3,3 V. En esta situación, el valor de la corriente consumida es 194,0 mA ($\frac{3{,}3\ V}{17\ \Omega}$), lo cual es una cota superior para el consumo real de este módulo.

La tabla A.1 resume los resultados anteriores.

Módulo	Cota superior de corriente (mA)
Placa MCU	9.1
LoRa transceiver	124.4
Circuito de medida de consumo	14.0
Circuito fotosensible	0.3
Circuito On/Off	194.0
Total	341.8

Tabla A.1: Cota superior de consumo en corriente de los módulos hardware alimentados a 3,3 V.

Como la fuente SPBW06F puede entregar hasta 1500 mA a 3,3 V (4,95 W) y en el peor caso se necesitaría 341,8 mA (1,13 W), esta fuente cumple con los requisitos.

Cuando la fuente SPBW06F entrega 1500 mA, la misma consume 310 mA, según su hoja de datos. Por lo tanto, la fuente MPM-10-12 tiene que poder entregar esa corriente y también la corriente requerida por el circuito de acondicionamiento.

Para calcular el consumo del circuito amplificador se tiene que tomar en cuenta el consumo pasivo del amplificador y la máxima corriente que puede entregar a su salida. En la página 4 de la hoja de datos del amplificador operacional MC34071 se menciona que en el peor caso el amplificador consume 2,8 mA pasivamente, y en la figura 13 se muestra una gráfica de la corriente de salida en cortocircuito en función a la temperatura. En esta gráfica se puede apreciar que el valor máximo de corriente son 32 mA.

Módulo	Cota superior de corriente (mA)
SPBW06F-03	310
Circuito acondicionador	34.8
Total	344.8

Tabla A.2: Cota superior de consumo en corriente de los módulos hardware alimentados a 12V.

Como la fuente MPM-10-12 puede entregar 850 mA a 12 V (10,2 W)y en el peor caso se necesitaría 344,8 mA (4,14 W), esta fuente cumple con los requisitos.

Esta página ha sido intencionalmente dejada en blanco.

Apéndice B

Esquemáticos y Layouts de PCBs

B.1. Placa inferior

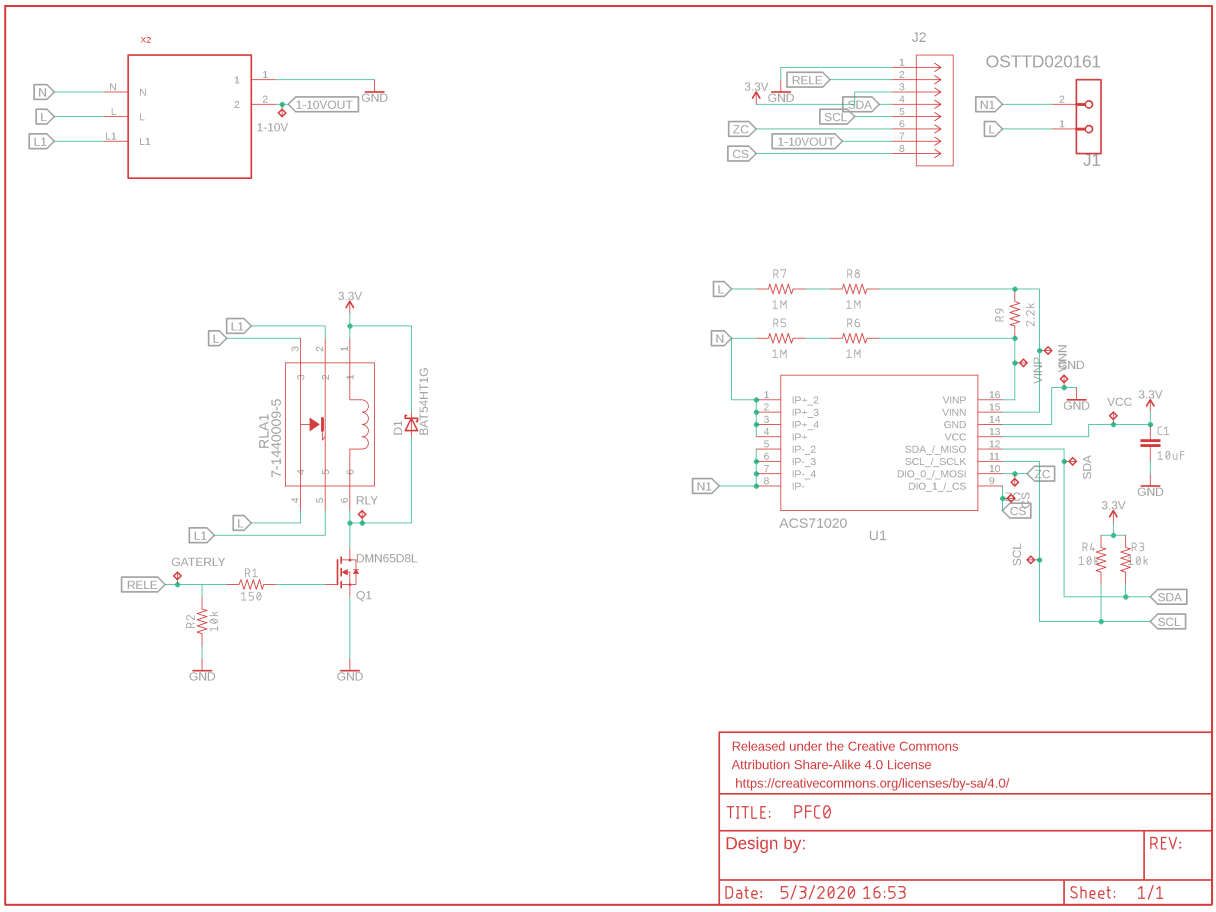


Figura B.1: Esquemático de placa inferior.

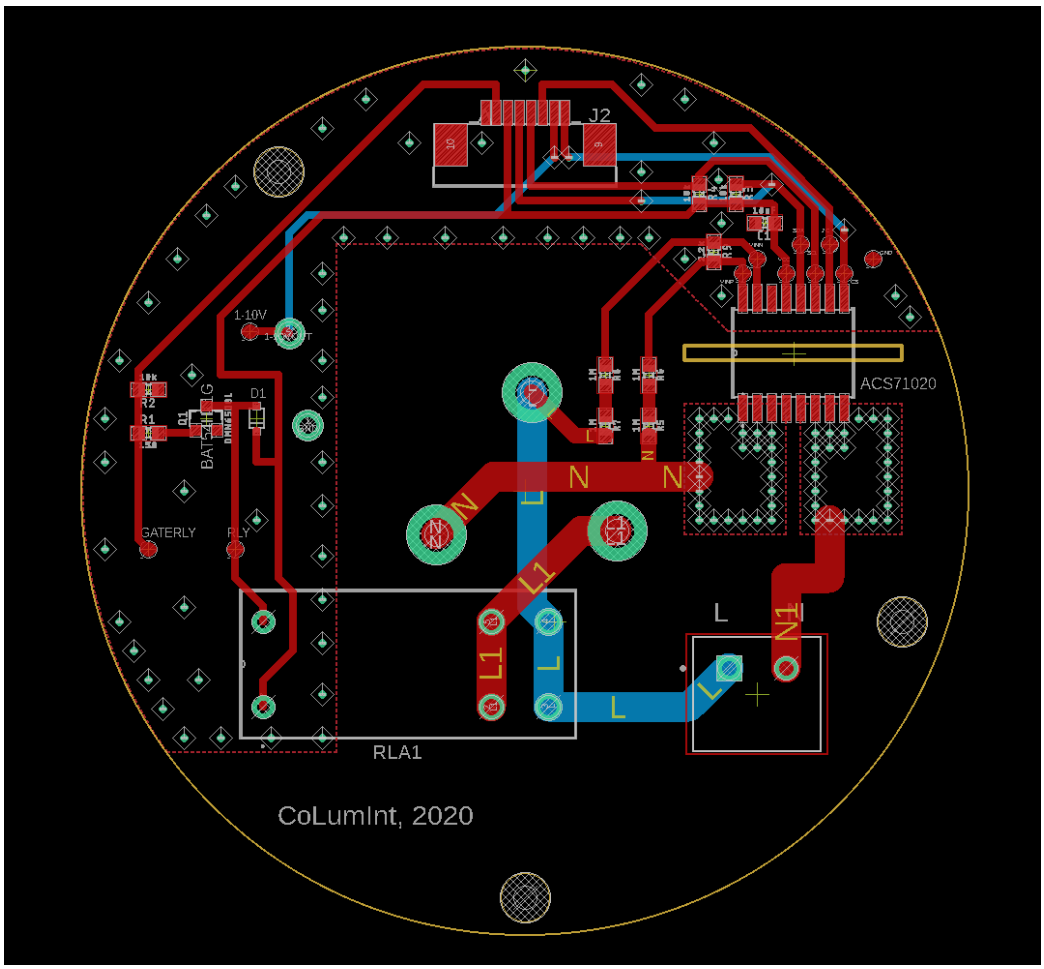
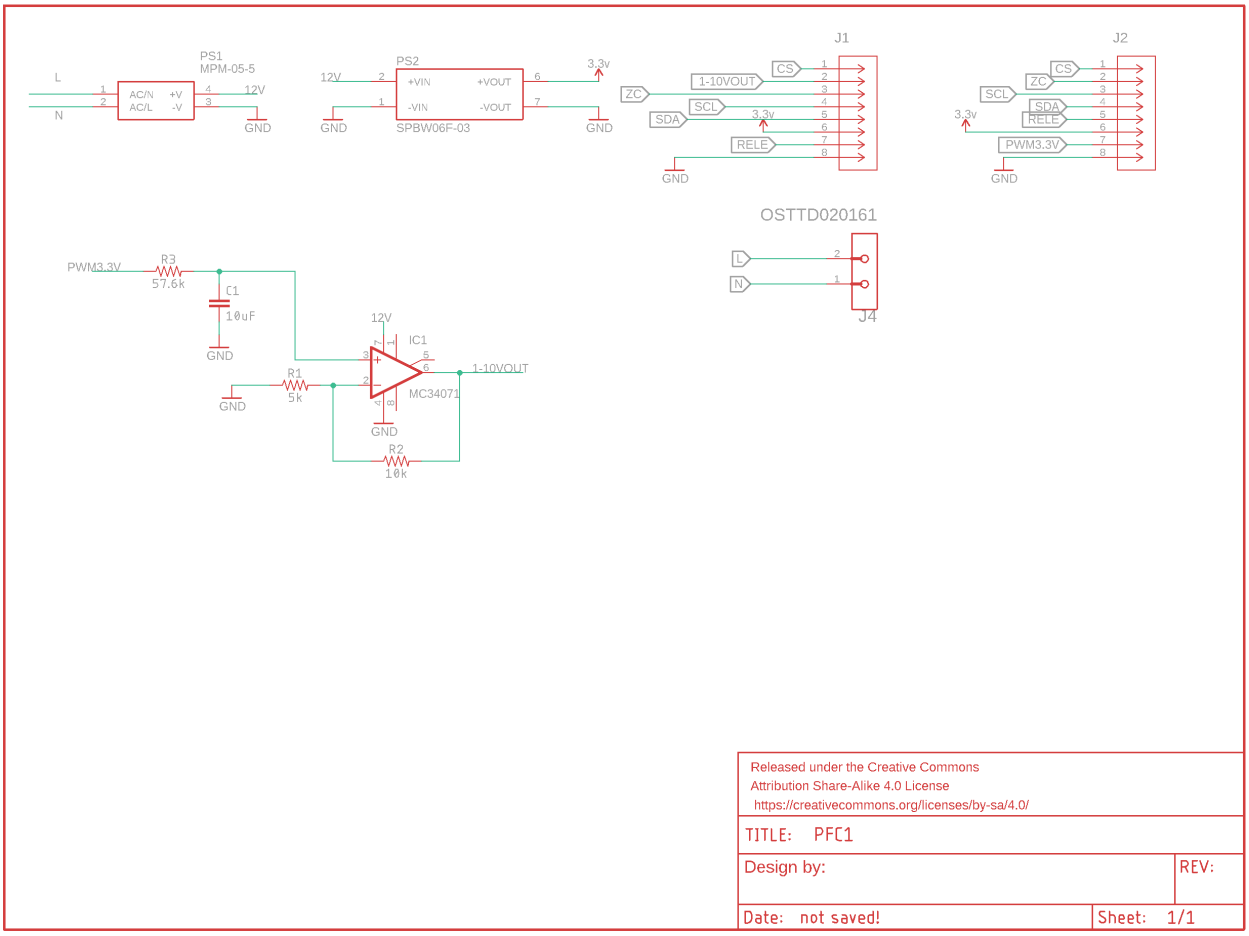


Figura B.2: Layout de placa inferior.

B.2. Placa media



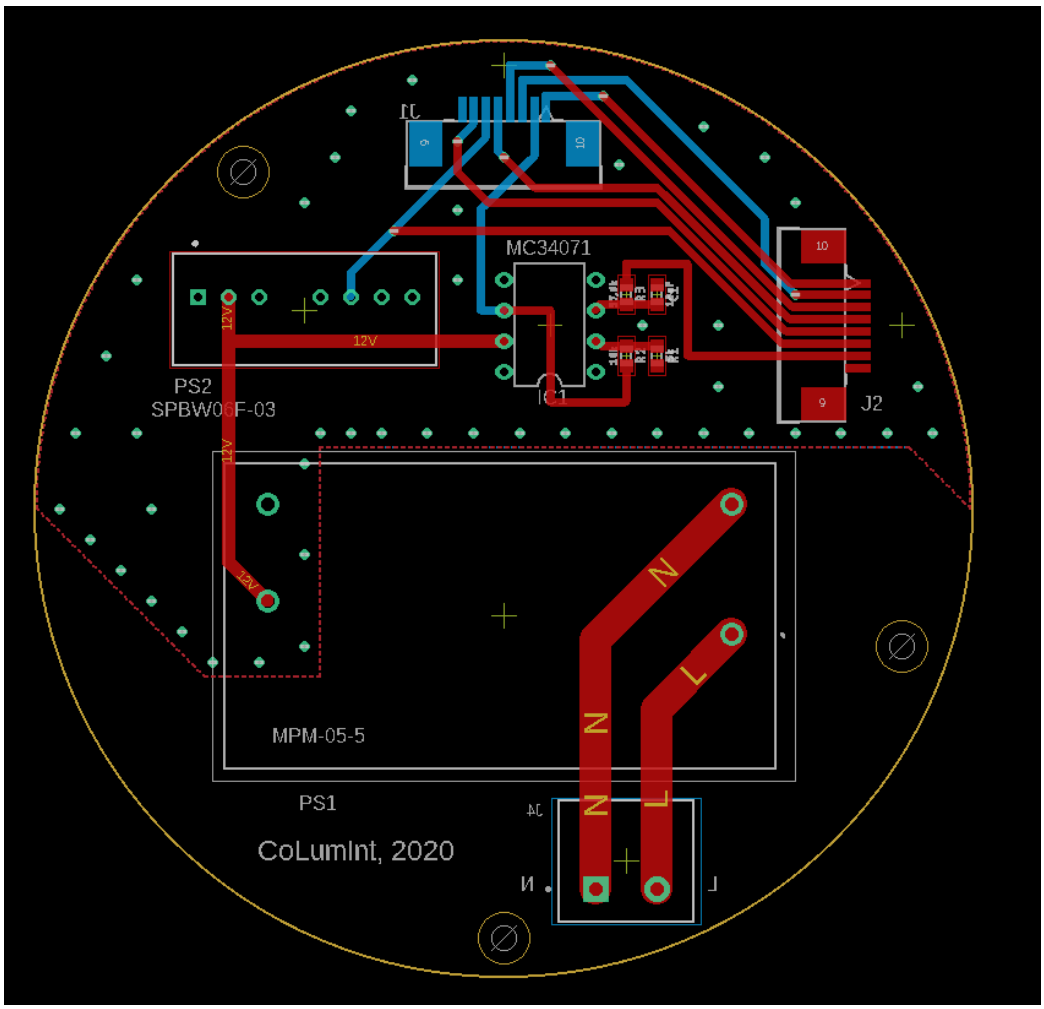


Figura B.3: Layout de placa media.

B.3. Placa superior

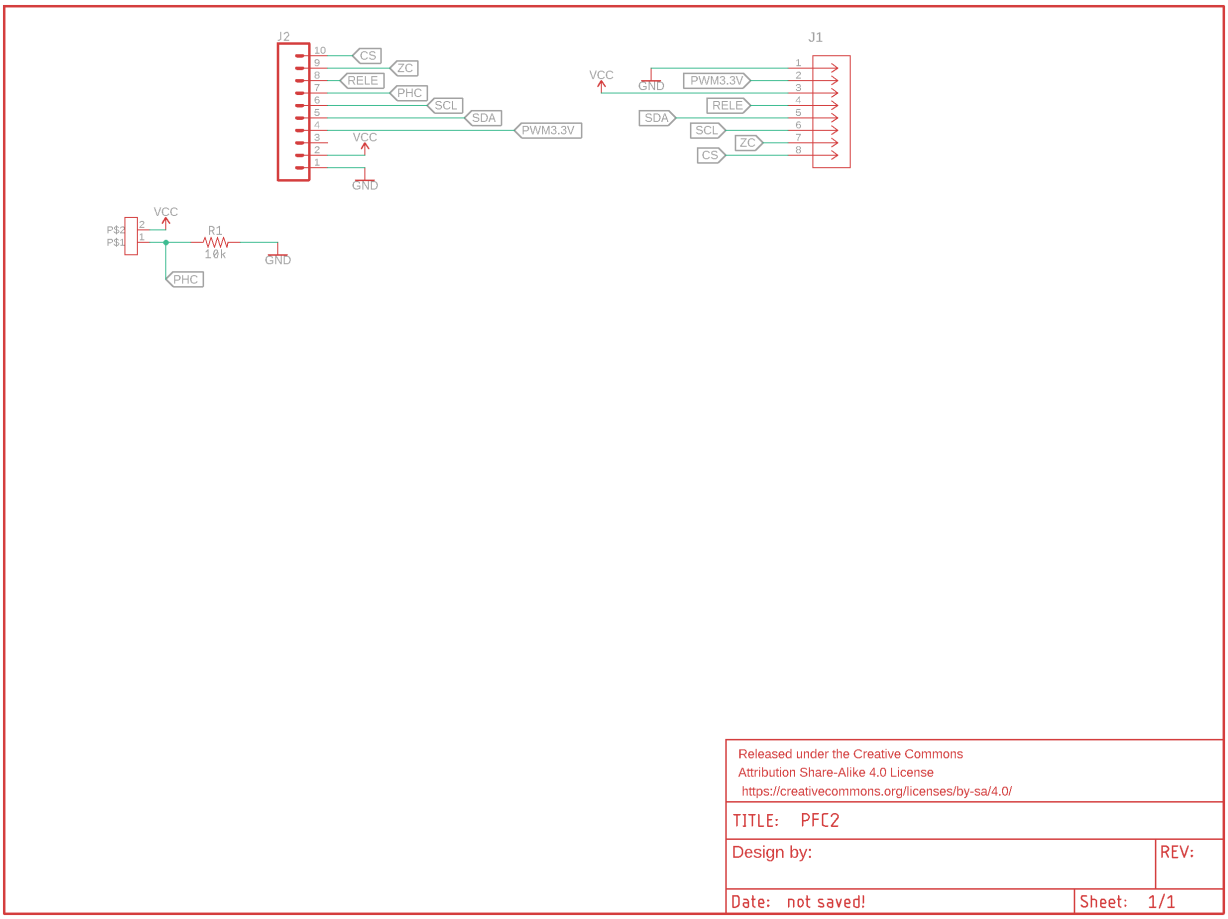


Figura B.4: Esquemático de placa superior.

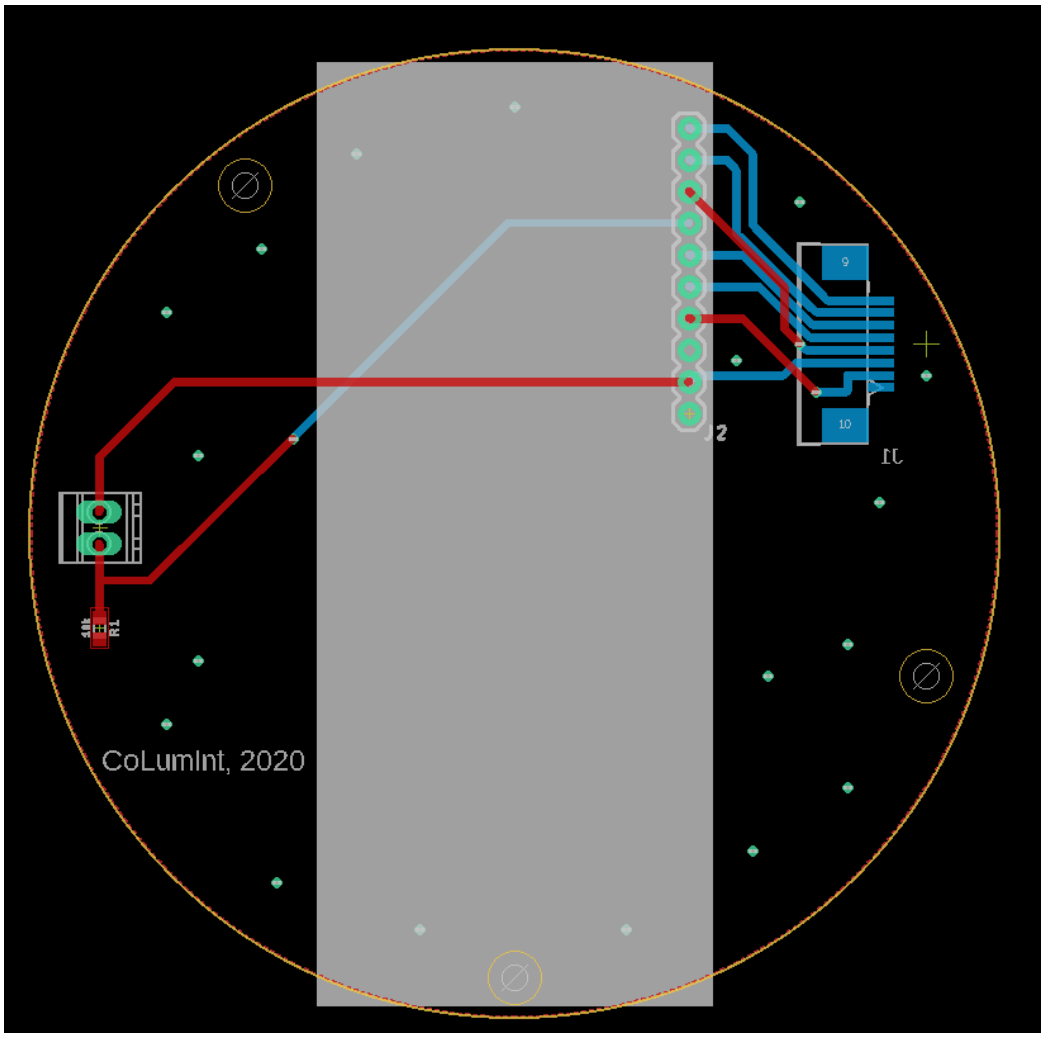


Figura B.5: Layout de placa superior.

Apéndice C

Codigo utilizado para Widgets de Ubidots

C.1. Widget : Result of GET MODE

C.1.1. HTML:

```
<div id="image_background">
  <p id="MODE_s">MODE:</p>
  <span id="MODE">No data</span>

  <p id="PHC_s">PHC:</p>
  <span id="PHC"> No data </span>

  <p id="DIM_LEVEL_s">DIM LEVEL:</p>
  <span id="DIM_LEVEL">No data</span>
</div>
```

C.1.2. CSS:

```
#image_background {
  font-family: "Arial";
  font-size: 20px;
  height: 500px;
  width: 500px;
}
#MODE, #PHC, #DIM_LEVEL {
  margin: 0;
  left: 270px;
  position: absolute;
}
```

```
 #MODE_s, #PHC_s, #DIM_LEVEL_s{
   margin: 0;
   left: 20px;
   position: absolute;
 }
#MODE, #MODE_s {
   top: 50px;
}
#PHC, #PHC_s {
   top: 100px;
}
#DIM_LEVEL, #DIM_LEVEL_s {
   top: 150px;
}
```

C.1.3. Javascript:

```
var $bg = $('#image_background');
var $text = $('#image_background_text');
var TOKEN  = 'BBFF-pNFBDWwdIZRS4wrWfHLQ8D4y6VGMOt';
var SUM_ID  = '5ec6e5c00ff4c37282a5e887';
var port;
var sum,suma;
var bytes=[];
var lastValue = null;

function byteDecoder(){
   var revbytes =[];
   var i=0;
   sum=suma;
      while(sum>256){
           revbytes[i] = sum%256;
           sum= sum - revbytes[i];
           sum= sum/256;
           i++;
      }
      bytes=revbytes.reverse();
      port=sum;
}
function changeText() {
   var dim_lvl = bytes[0];
   var mode = bytes[1] & 0x03;
   var phc = (bytes[1] & 0x04) >> 2;
   if (port == 20){
      document.getElementById('MODE').innerText = mode;
      document.getElementById('PHC').innerText = phc;
```

```
    document.getElementById('DIM_LEVEL').innerText = dim_lvl;
}

function getLastValue(variableId, token) {
  var url = 'https://industrial.api.ubidots.com/api/v1.6/variables/'
    + variableId + '/values';
  $.get(url, { token: token, page_size: 1 }, function (res) {
    if (lastValue === null || res.results[0].value !== lastValue) {
      lastValue = res.results[0].value;
      suma=lastValue
    }
  });
}

setInterval(function () {
  getLastValue(SUM_ID, TOKEN);
  byteDecoder();
  if (port == 20){
    changeText();
  }
}, 2000);
```

C.2. Widget : Result of DIR EVENT WD

C.2.1. HTML:

```
<div id="image_background">
    <span id="WE_s">Weekday Events:</span>
    <div id="data_canvas">
        <span id="WE_1">No data</span>
        <span id="WE_2"></span>
        <span id="WE_3"></span>
    </div>

</div>
```

C.2.2. CSS:

```
#image_background {
  font-family: "Arial";
  font-size: 20px;
  height: 512px;
  width: 512px;
  overflow-y: auto;
}
```

```
 #WE_s{
   margin: 0;
   left: 20px;
   top: 30px;
 }
#WE_1,#WE_2,#WE_3{
   margin: 0;
   left: 30px;
}
#WE_1{
   top: 50px;
}
#WE_2{
   top: 150px;
}
#WE_3{
   top: 250px;
}
```

C.2.3. Javascript:

```
var $bg = $('#image_background');
var $text = $('#image_background_text');
var TOKEN    = 'BBFF-pNFBDWwdIZRS4wrWfHLQ8D4y6VGMOt';
var SUM_ID   = '5ec6e5c00ff4c37282a5e887';
var port;
var sum,suma;
var bytes=[];
var lastValue = null;

function byteDecoder(){
  var i=0;
  sum=suma;
  var revbytes =[];
    while(sum>256){
        revbytes[i] = sum%256;
        sum= sum - revbytes[i];
        sum= sum/256;
        i++;
    }
    bytes=revbytes.reverse();
    port=sum;
}

function changeText() {
    var EventTime=[];
```

```
    var EventID=[];
    var EventDim=[];
    var eventQty;
    var j=0;
    if(suma < 4294967295){
        eventQty=1;
    }else {
        eventQty=2;
    }
    var k=0;
    while(k<eventQty){
      EventTime[k]=(bytes[3*k]<<3) | (bytes[3*k+1] >>5);
      EventID[k]=bytes[(3*k)+1] & 0x1f;
      EventDim[k]=bytes[(3*k)+2];
      k++;
    }
    var texto="";
    for(j=0;j<eventQty;j++){
    texto=texto + "␣Evento_" + EventID[j] +"␣Hora:␣" +
            Math.floor(EventTime[j]/60) +":" + EventTime[j]%60 +
            "␣Dim:␣" +  EventDim[j] +'\n';
    }
    if(port == 6){
        document.getElementById('WE_1').innerText =  texto ;
    }
    if(port == 7){
      document.getElementById('WE_2').innerText =   texto;
    }
    if(port == 8){
      document.getElementById('WE_3').innerText =   texto
    }
}

function getLastValue(variableId, token) {
  var url = 'https://industrial.api.ubidots.com/api/v1.6/variables/'
    + variableId + '/values';
  $.get(url, { token: token, page_size: 1 }, function (res) {
    if (lastValue === null || res.results[0].value !== lastValue) {
      lastValue = res.results[0].value;
      suma=lastValue;
    }
  });
}

setInterval(function () {
```

```
    getLastValue(SUM_ID, TOKEN);
    byteDecoder();
    if (port >= 6 && port<=10){
        changeText();
    }
},1000);
```

C.3. Widget : Result of GET MOMENT

C.3.1. HTML:

```
<div id="image_background">
  <p id="FECHA_s">FECHA:</p>
  <span id="FECHA">No data</span>
  <p id="HORA_s">HORA:</p>
  <span id="HORA"> No data </span>
</div>
```

C.3.2. CSS:

```
#image_background {
  font-family: "Arial";
  font-size: 20px;
  height: 500px;
  width: 500px;
}
#FECHA, #HORA {
  margin: 0;
  left: 270px;
  position: absolute;
}
 #FECHA_s, #HORA_s{
  margin: 0;
  left: 20px;
  position: absolute;
 }
#FECHA, #FECHA_s {
  top: 50px;
}
#HORA, #HORA_s {
  top: 100px;
}
```

C.3.3. Javascript:

```
var $bg = $('#image_background');
```

```
var $text = $('#image_background_text');
var TOKEN   = 'BBFF-pNFBDWwdIZRS4wrWfHLQ8D4y6VGMOt';
var SUM_ID  = '5ec6e5c00ff4c37282a5e887';
var port;
var sum,suma;
var bytes=[];
var lastValue = null;

function byteDecoder(){
  var revbytes =[];
  var i=0;
  sum=suma;
    while(sum>256){
        revbytes[i] = sum%256;
        sum= sum - revbytes[i];
        sum= sum/256;
        i++;
    }
    bytes=revbytes.reverse();
    port=sum;
}


function changeText() {
      var seconds  = (bytes[0]<<9)+(bytes[1]<<1)+((bytes[2] & 0x80)>>7)
      var year      = ((bytes[2] & 0x7e)>>1) + 2000;
      var day       = (((bytes[2] & 0x01)<<8)+ bytes[3]) + 1;
//calculo de hora a partir de segundos.
      var aux_seconds = seconds;
      var aux_minutes = Math.floor(seconds/60);
      var aux_hour    = Math.floor(aux_minutes/60);
      var seconds     = aux_seconds - (aux_minutes * 60);
      var minutes     = aux_minutes - (aux_hour*60);
      var hour        = aux_hour;

//calculo de dia a partir de cantidad de dias y anno
var daysPerMonth =[31,28,31,30,31,30,31,31,30,31,30,31];
var i=0;
  if (year %4 == 0){
    daysPerMonth[1]++;
  }
  var aux_day = day;
  while(daysPerMonth[i]<aux_day){
    aux_day=aux_day-daysPerMonth[i];
    i++;
```

```
  }
  day = aux_day;

  if (port == 21){
  document.getElementById('FECHA').innerText = day +'/' + (i+1) + '/' + year;
  document.getElementById('HORA').innerText = hour + ':'
        + minutes + ':' + seconds;
  }
}

function getLastValue(variableId, token) {
  var url = 'https://industrial.api.ubidots.com/api/v1.6/variables/'
        + variableId + '/values';
  $.get(url, { token: token, page_size: 1 }, function (res) {
    if (lastValue === null || res.results[0].value !== lastValue) {
      lastValue = res.results[0].value;
      suma=lastValue
    }
  });
}

setInterval(function () {
  getLastValue(SUM_ID, TOKEN);
  byteDecoder();
  if (port == 21){
    changeText();
  }
}, 1000);
```

C.4. Widget : Result of DIR EVENT WE

C.4.1. HTML:

```
<div id="image_background">
    <span id="WE_s">Weekend Events:</span>
    <div id="data_canvas">
        <span id="WE_1">No data</span>
        <span id="WE_2"></span>
        <span id="WE_3"></span>
    </div>

</div>
```

C.4.2. CSS:

```
#image_background {
```

```
  font-family: "Arial";
  font-size: 20px;
  height: 512px;
  width: 512px;
  overflow-y: auto;
}
 #WE_s{
  margin: 0;
  left: 20px;
  top: 30px;
 }
#WE_1,#WE_2,#WE_3{
  margin: 0;
  left: 30px;
}
#WE_1{
  top: 50px;
}
#WE_2{
  top: 150px;
}
#WE_3{
  top: 250px;
}
```

C.4.3. Javascript:

```
var $bg = $('#image_background');
var $text = $('#image_background_text');
var TOKEN   = 'BBFF-pNFBDWwdIZRS4wrWfHLQ8D4y6VGMOt';
var SUM_ID  = '5ec6e5c00ff4c37282a5e887';
var port;
var sum,suma;
var bytes=[];
var lastValue = null;

function byteDecoder(){
  var i=0;
  sum=suma;
  var revbytes =[];
    while(sum>256){
        revbytes[i] = sum%256;
        sum= sum - revbytes[i];
        sum= sum/256;
        i++;
    }
```

```
    bytes=revbytes.reverse();
    port=sum;
}

function changeText() {
    var EventTime=[];
    var EventID=[];
    var EventDim=[];
    var eventQty;
    var j=0;

    if(suma < 4294967295){
        eventQty=1;
    }else {
        eventQty=2;
    }
    var k=0;
    while(k<eventQty){
      EventTime[k]=(bytes[3*k]<<3) | (bytes[3*k+1] >>5);
      EventID[k]=bytes[(3*k)+1] & 0x1f;
      EventDim[k]=bytes[(3*k)+2];
      k++;
    }
    var texto="";
    for(j=0;j<eventQty;j++){
    texto=texto + "␣Evento_" + EventID[j] +"␣Hora:␣" +
    Math.floor(EventTime[j]/60) +":" +
    EventTime[j]%60 + "␣Dim:␣" +  EventDim[j] +'\n';
    }
    if(port == 1){
        document.getElementById('WE_1').innerText =  texto ;
        //document.getElementById('WE_2').innerText =   "";
        //document.getElementById('WE_3').innerText =   "";
    }
    if(port == 2){
      document.getElementById('WE_2').innerText =   texto;
    }
    if(port == 3){
      document.getElementById('WE_3').innerText =   texto
    }
}

function getLastValue(variableId, token) {
  var url = 'https://industrial.api.ubidots.com/api/v1.6/variables/'
  + variableId + '/values';
```

```
  $.get(url, { token: token, page_size: 1 }, function (res) {
    if (lastValue === null || res.results[0].value !== lastValue) {
      lastValue = res.results[0].value;
      suma=lastValue;
    }
  });
}

setInterval(function () {
  getLastValue(SUM_ID, TOKEN);
  byteDecoder();
  if (port >= 1 && port<=5){
      changeText();
  }
},1000);
```

C.5. Widget : Result of DIR EVENT SE

C.5.1. HTML:

```
<div id="image_background">
    <span id="WE_s">Sun Events:</span>
    <div id="data_canvas">
        <span id="WE_1">No data</span>
        <span id="WE_2"></span>
        <span id="WE_3"></span>
    </div>

</div>
```

C.5.2. CSS:

```
#image_background {
  font-family: "Arial";
  font-size: 20px;
  height: 512px;
  width: 512px;
  overflow-y: auto;
}

 #WE_s{
  margin: 0;
  left: 20px;
  top: 30px;
 }
#WE_1,#WE_2,#WE_3{
```

```
  margin: 0;
  left: 30px;
}
#WE_1{
  top: 50px;
}
#WE_2{
  top: 150px;
}
#WE_3{
  top: 250px;
}
```

C.5.3. Javascript:

```
var $bg = $('#image_background');
var $text = $('#image_background_text');
var TOKEN  = 'BBFF-pNFBDWwdIZRS4wrWfHLQ8D4y6VGMOt';
var SUM_ID  = '5ec6e5c00ff4c37282a5e887';
var port;
var sum,suma;
var revbytes =[];
var bytes=[];
var lastValue = null;

function byteDecoder(){
  var i=0;
  sum=suma;
    while(sum>256){
        revbytes[i] = sum%256;
        sum= sum - revbytes[i];
        sum= sum/256;
        i++;
    }
    bytes=revbytes.reverse();
    port=sum;
}

function changeText() {
    var EventTime=[];
    var EventID=[];
    var EventDim=[];
    var eventQty;
    var j=0;

    if(bytes[4] === null){
```

```
        eventQty=1;
    }else if(bytes[6] === null){
        eventQty=2;
    }else{
        eventQty=2;
    }

    var k=0;
    while(k<eventQty){
      EventTime[k]=(bytes[3*k]<<3) | (bytes[3*k+1] >>5);
      EventID[k]=bytes[(3*k)+1] & 0x1f;
      EventDim[k]=bytes[(3*k)+2];
      k++;
    }

    var texto="";
    for(j=0;j<k;j++){
    texto=texto + "␣Evento_" + EventID[j] +"␣Hora:␣"
    + Math.floor(EventTime[j]/60) +":" + EventTime[j]%60
    + "␣Dim:␣" +  EventDim[j] +'\n';
    }

    if(port == 11){
        document.getElementById('WE_1').innerText =  texto ;
        document.getElementById('WE_2').innerText =  "";
        document.getElementById('WE_3').innerText =  "";
    }
    if(port == 12){
      document.getElementById('WE_2').innerText =  texto;
    }
    if(port == 13){
      document.getElementById('WE_3').innerText =  texto
    }

}

function getLastValue(variableId, token) {
  var url = 'https://industrial.api.ubidots.com/api/v1.6/variables/'
  + variableId + '/values';
  $.get(url, { token: token, page_size: 1 }, function (res) {
    if (lastValue === null || res.results[0].value !== lastValue) {
      lastValue = res.results[0].value;
      suma=lastValue;
    }
  });
```

```
}

setInterval(function () {
  getLastValue(SUM_ID, TOKEN);
  byteDecoder();
  if (port >= 11 && port<=15){
      changeText();
  }
}, 2000);
```

C.6. Widget : Electric Measures

C.6.1. HTML:

```
<div id="image_background">
  <p id="VOLT_s">VOLT:</p>
  <span id="VOLT">No Data</span>
  <p id="CURR_s">CURRENT:</p>
  <span id="CURR">No Data </span>
  <p id="ACT_POW_s">ACTIVE POWER:</p>
  <span id="ACT_POW">No Data</span>
  <p id="REAP_POW_s">REACTIVE POWER:</p>
  <span id="REAP_POW">No Data</span>
  <p id="APP_POW_s">APPARENT POWER:</p>
  <span id="APP_POW">No Data</span>
  <p id="PFACTOR_s">PFACTOR:</p>
  <span id="PFACTOR">No Data</span>
</div>
```

C.6.2. CSS:

```
#image_background {
  font-family: "Arial";
  font-size: 20px;
  height: 500px;
  width: 500px;
}
#VOLT, #CURR, #ACT_POW, #REAP_POW, #APP_POW, #PFACTOR{
  margin: 0;
  left: 400px;
  position: absolute;
}
#VOLT_s, #CURR_s, #ACT_POW_s, #REAP_POW_s, #APP_POW_s, #PFACTOR_s{
  margin: 0;
  left: 20px;
  position: absolute;
```

```
 }
#VOLT, #VOLT_s {
  top: 50px;
}
#CURR, #CURR_s {
  top: 100px;
}
#ACT_POW, #ACT_POW_s {
  top: 150px;
}
#REAP_POW, #REAP_POW_s {
  top: 200px;
}
#APP_POW, #APP_POW_s {
  top: 250px;
}
#PFACTOR, #PFACTOR_s {
  top: 300px;
}
```

C.6.3. Javascript:

```
var $bg = $('#image_background');
var $text = $('#image_background_text');
var TOKEN   = 'BBFF-pNFBDWwdIZRS4wrWfHLQ8D4y6VGMOt';
var SUM_ID  = '5ec6e5c00ff4c37282a5e887';
var port;
var sum,suma;
var bytes=[];
var lastValue = null;

function byteDecoder(){
  var revbytes =[];
  var i=0;
  sum=suma;
    while(sum>256){
        revbytes[i] = sum%256;
        sum= sum - revbytes[i];
        sum= sum/256;
        i++;
    }
    bytes=revbytes.reverse();
    port=sum;
}

function changeText() {
```

```
  switch (port){
    case 22:
    var volt=((bytes[0]<<8)+(bytes[1]))/10;
      document.getElementById('VOLT').innerText = volt + " V";
    break;
    case 23:
    var curr=((bytes[0]<<8)+(bytes[1]))/10;
    document.getElementById('CURR').innerText = curr + " mA";
    break;
    case 24:
    var actpwr=((bytes[0]<<8)+(bytes[1]))/10;
    document.getElementById('ACT_POW').innerText = actpwr + " W";
    break;
    case 25:
    var reapwr=((bytes[0]<<8)+(bytes[1]))/10;
    document.getElementById('REAP_POW').innerText = reapwr + " VAr";
    break;
    case 26:
    var apppwr=((bytes[0]<<8)+(bytes[1]))/10;
    document.getElementById('APP_POW').innerText = apppwr + " VA";
    break;
    case 27:
    var pfactor=((bytes[0]<<8)+(bytes[1]))/100;
    document.getElementById('PFACTOR').innerText = pfactor;
    break;
  }
}

function getLastValue(variableId, token) {
  var url = 'https://industrial.api.ubidots.com/api/v1.6/variables/'
    + variableId + '/values';
  $.get(url, { token: token, page_size: 1 }, function (res) {
    if (lastValue === null || res.results[0].value !== lastValue) {
      lastValue = res.results[0].value;
      suma=lastValue
    }
  });
}

setInterval(function () {
  getLastValue(SUM_ID, TOKEN);
  byteDecoder();
  if (port >= 22 && port <=27){
    changeText();
  }
```

```
}, 2000);
```

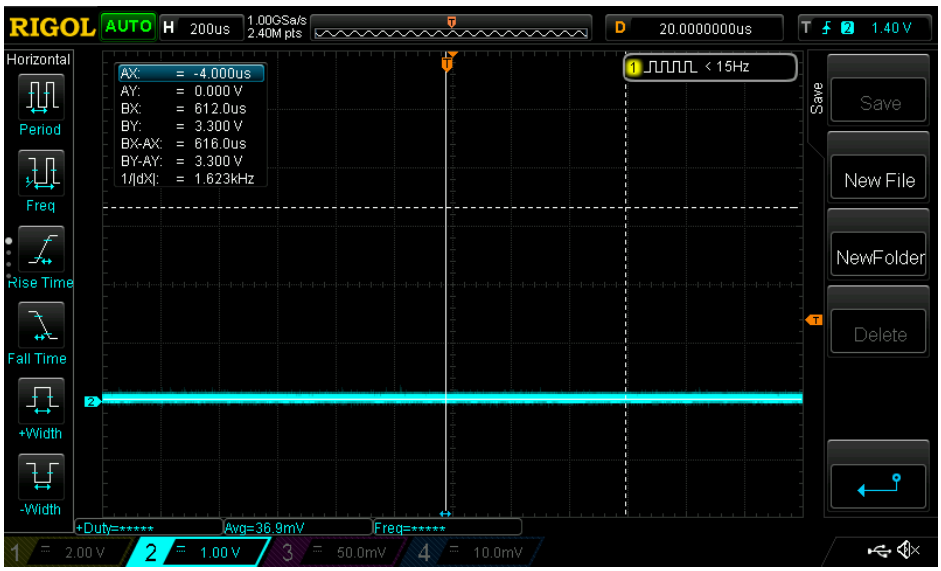
Esta página ha sido intencionalmente dejada en blanco.

Apéndice D

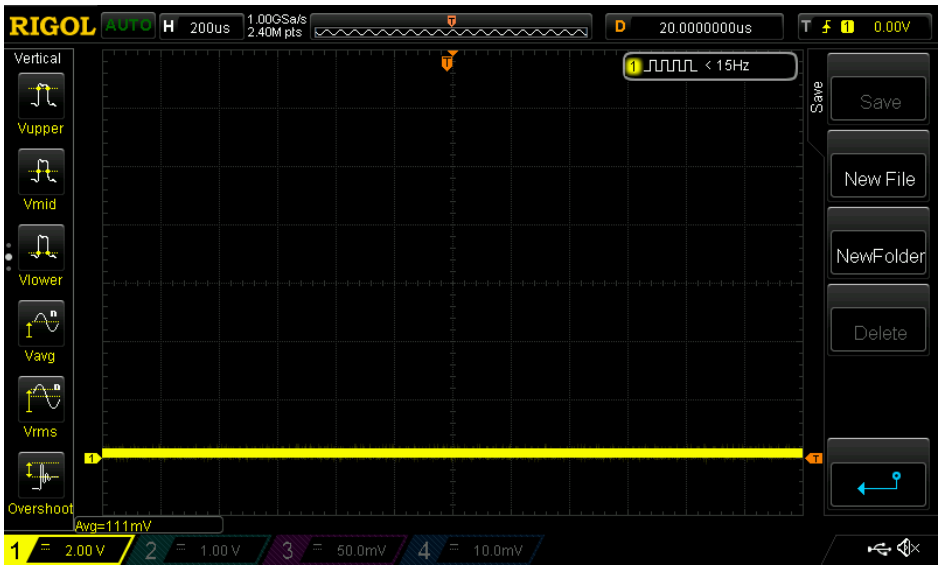
Imágenes del osciloscopio obtenidas en las pruebas de la sección 6.3.4

En las figuras D.1, D.2, D.3, D.4 y D.5 se muestran las imágenes obtenidas en osciloscopio para los valores de ciclo de trabajo 0, 25, 50, 75 y 100 respectivamente, en las pruebas de la sección 6.3.4.

Apéndice D. Imágenes del osciloscopio obtenidas en las pruebas de la sección 6.3.4

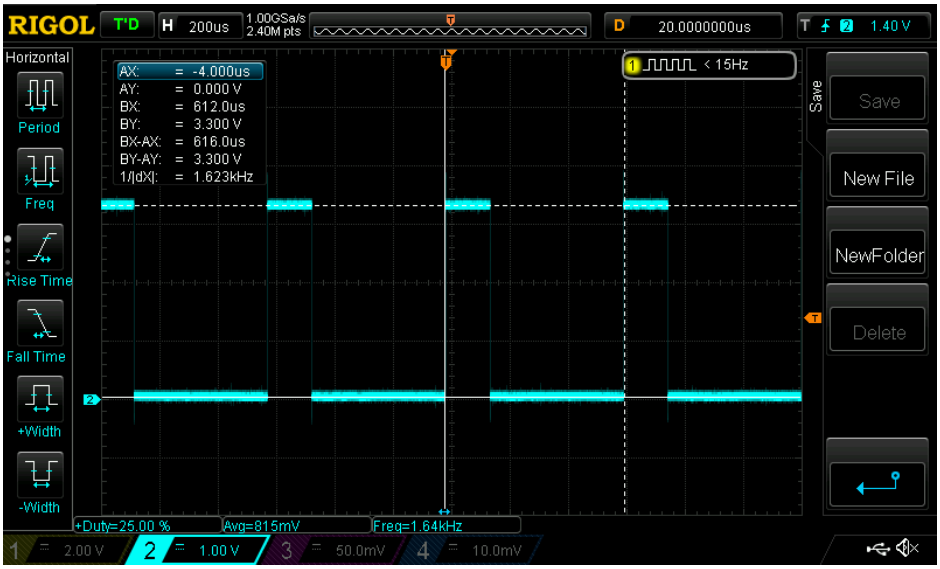


(a) PWM

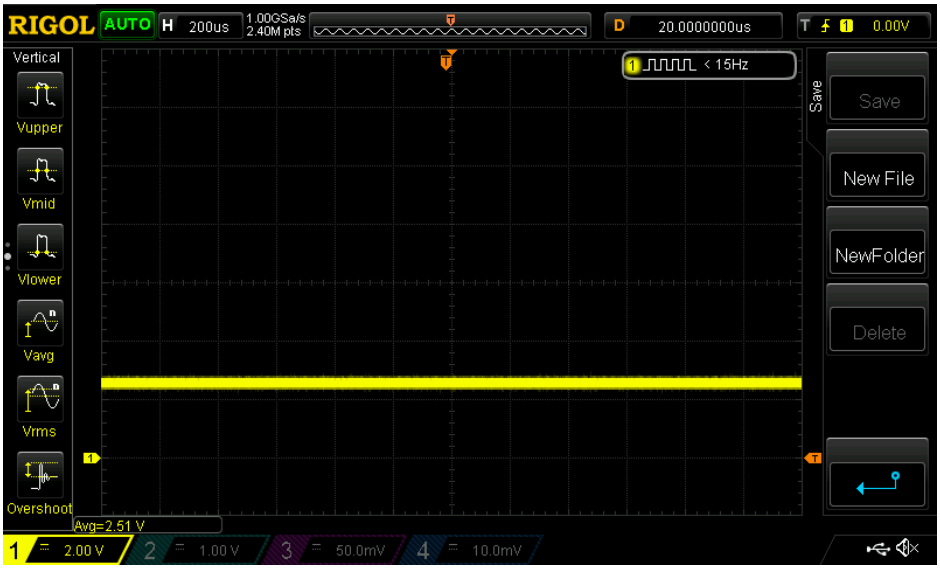


(b) Señal 0-10V

Figura D.1: PWM y 0-10V: en la figura (a) se aprecia el PWM con duty cycle 0 % y en la figura (b) se aprecia la salida 0-10V acorde. Esta salida es de valor 111 mV en promedio, como se aprecia en la parte inferior izquierda de la imagen.



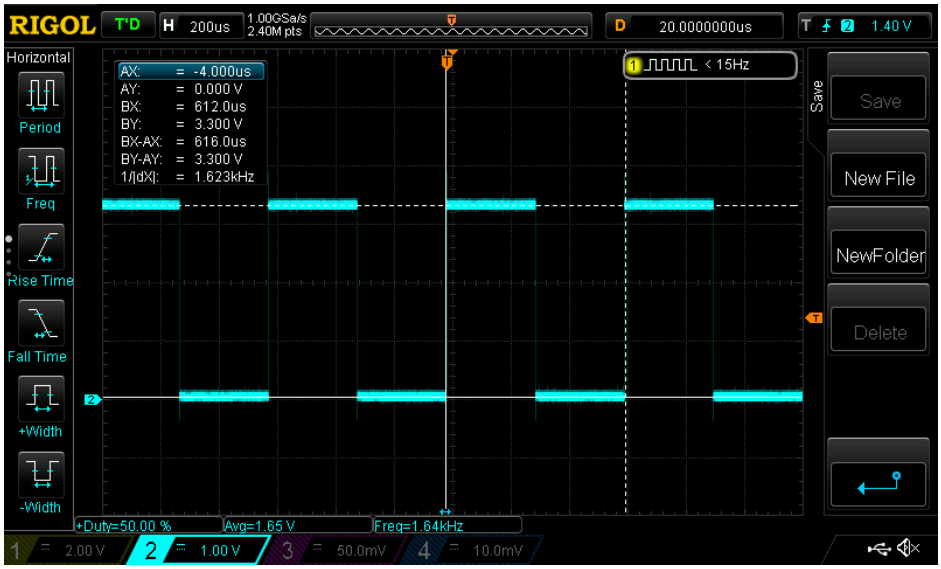
(a) PWM



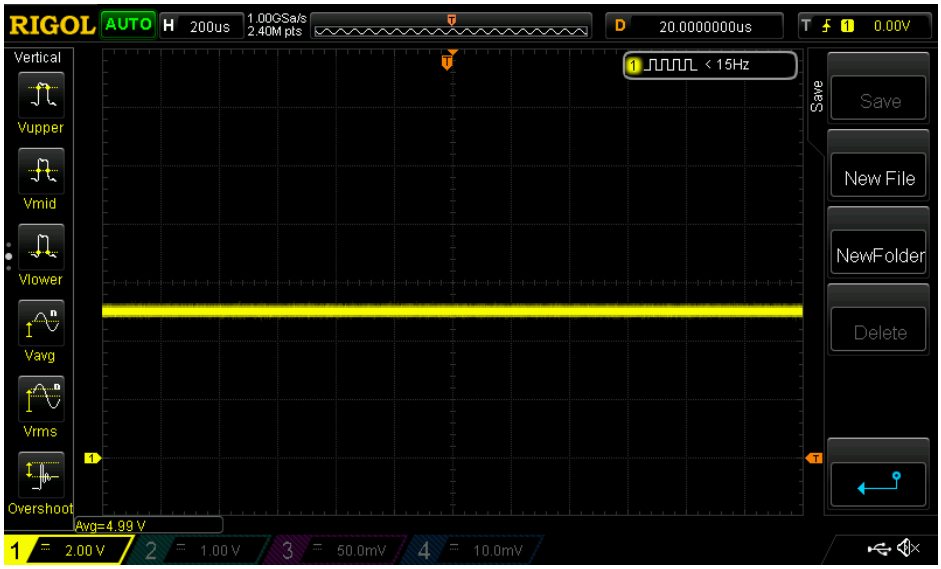
(b) Señal 0-10V

Figura D.2: PWM y 0-10V: en la figura (a) se aprecia el PWM con duty cycle 25 % y en la figura (b) se aprecia la salida 0-10V acorde. Esta salida es de valor 2,51 V en promedio, como se aprecia en la parte inferior izquierda de la imagen.

Apéndice D. Imágenes del osciloscopio obtenidas en las pruebas de la sección 6.3.4

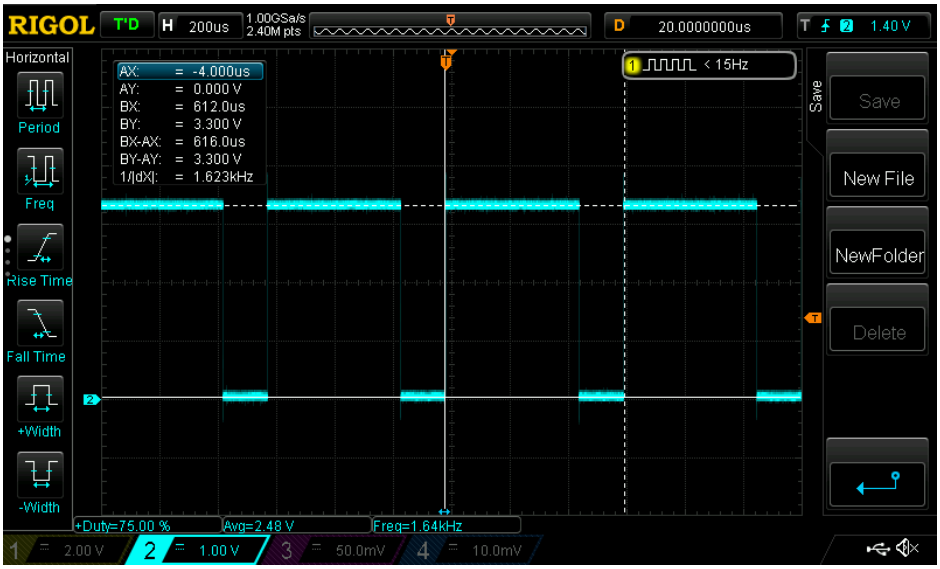


(a) PWM

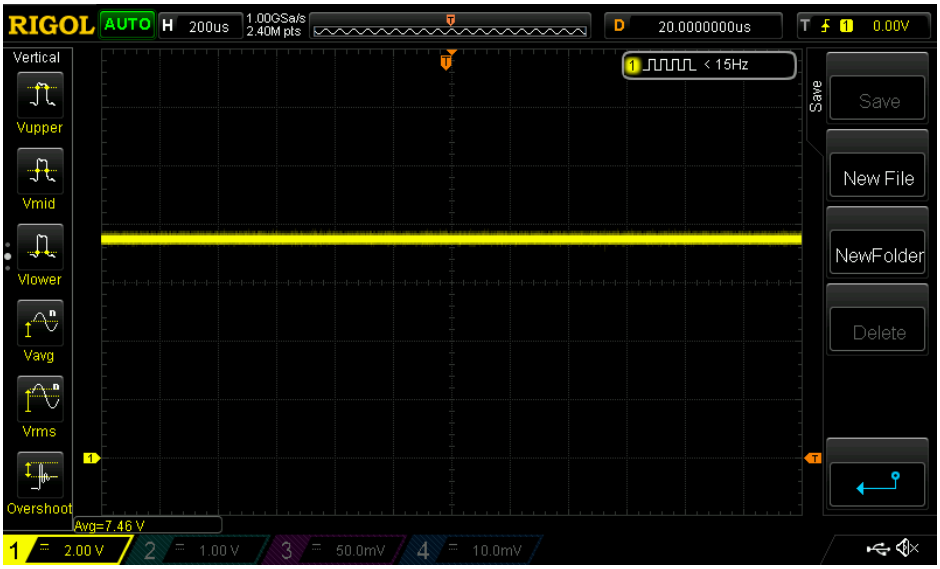


(b) Señal 0-10V

Figura D.3: PWM y 0-10V: en la figura (a) se aprecia el PWM con duty cycle 50 % y en la figura (b) se aprecia la salida 0-10V acorde. Esta salida es de valor $4{,}99$ V en promedio, como se aprecia en la parte inferior izquierda de la imagen.

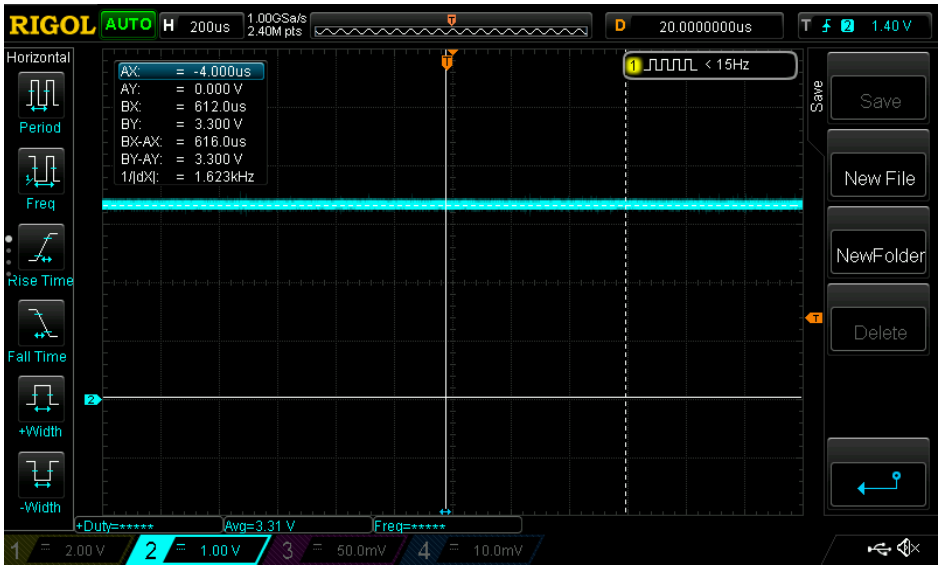


(a) PWM

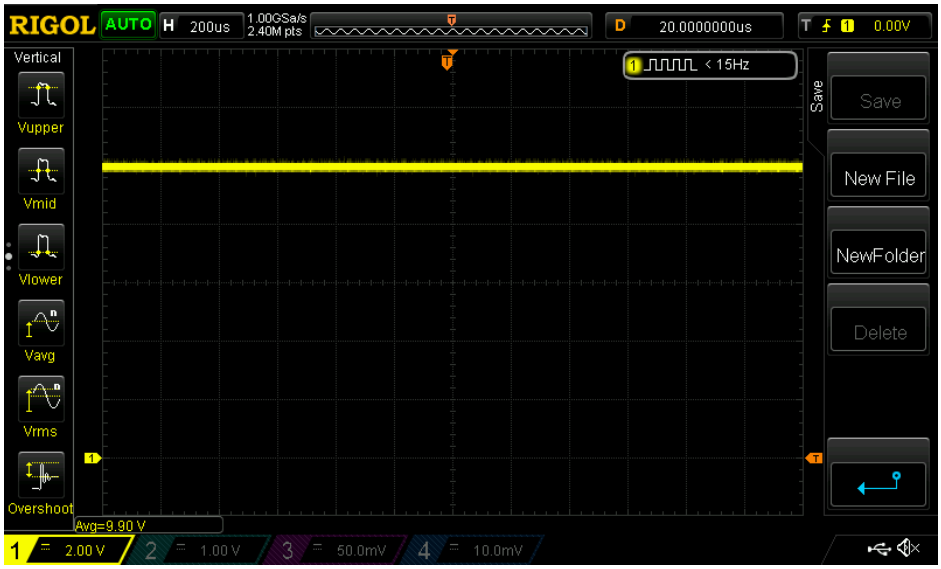


(b) Señal 0-10V

Figura D.4: PWM y 0-10V: en la figura (a) se aprecia el PWM con duty cycle 75 % y en la figura (b) se aprecia la salida 0-10V acorde. Esta salida es de valor $7{,}46$ V en promedio, como se aprecia en la parte inferior izquierda de la imagen.



(a) PWM



(b) Señal 0-10V

Figura D.5: PWM y 0-10V: en la figura (a) se aprecia el PWM con duty cycle 100 % y en la figura (b) se aprecia la salida 0-10V acorde. Esta salida es de valor 9,90 V en promedio, como se aprecia en la parte inferior izquierda de la imagen.

Apéndice E

Propuesta de diseño propio de circuito de medida de consumo

En la figura E.1 se muestra un prototipo de posible circuito de medida de consumo, el cual no incluye el ACS71020. El MCU utilizaría el ADC14 para tomar medidas en los puntos PinV y PinI, de manera de poder relevar las señales de tensión y corriente. La frecuencia de muestreo debería ser mucho mayor a 100 Hz, ya que la señal a muestrear es de 50 Hz. Se definen las magnitudes de fondo de escala V_{FE} e I_{FE}, que son la máxima tensión y corriente que el circuito sería capaz de medir. Se coloca una resistencia en serie al controlador y la luminaria ($R1$) para calcular la corriente que la atraviesa midiendo la tensión en sus bornes. Esta resistencia debe ser pequeña, de manera de no alterar la tensión de alimentación de la luminaria. El amplificador diferencial $U1$ debe transformar la sinusoidal de entrada, en otra sinusoidal cuya amplitud pico a pico sea 3,3 V cuando pasa I_{FE} por $R1$ y con un offset de 1,65 V. Este offset hace que la sinusoidal de salida esté centrada en la mitad del intervalo de tensión que se puede medir con el ADC. Las resistencias $R3$ y $R4$ forman un divisor resistivo, y la tensión que se desea medir es proporcional a la tensión en bornes de $R3$. El amplificador diferencial $U2$ debe ser tal que la tensión en $PinV$ sea una sinusoidal centrada en 1,65 V de amplitud pico a pico 3,3 V cuando la tensión V sea V_{FE}.

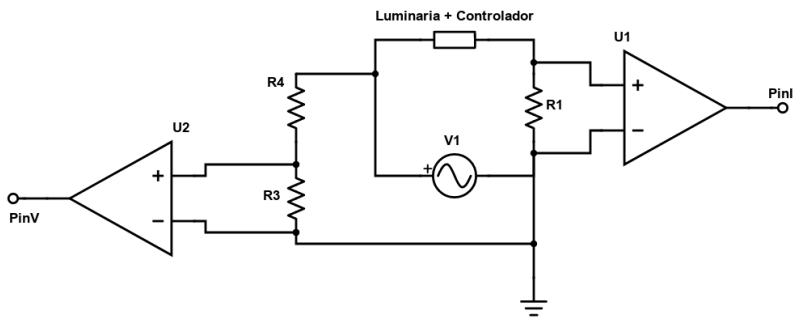


Figura E.1: Esquemático de propuesta de diseño propio del circuito de medida de consumo.

Teniendo las muestras de las señales de tensión y corriente, el MCU podría calcular la tensión RMS (V_{RMS}), la corriente RMS (I_{RMS}), potencia activa (P), reactiva (Q), aparente (S) y factor de potencia (PF). Estas podrian calcularse como se muestra a continuación, donde V_n son las muestras de tensión, I_n las de corriente, y N la cantidad de muestras en un ciclo de la señal:

$$V_{RMS} = \sqrt{\frac{\sum_{n=0}^{N-1} V_n^2}{N}}$$

$$I_{RMS} = \sqrt{\frac{\sum_{n=0}^{N-1} I_n^2}{N}}$$

$$|S| = V_{RMS} \times I_{RMS}$$

$$P = \frac{\sum_{n=0}^{N-1} V_n \times I_n}{N}$$

$$Q = \sqrt{S^2 - P^2}$$

$$|PF| = \frac{P}{|S|}$$

Referencias

[1] Claudia Rieling. Smart city lighting in Wipperfürth. https://www.ledinside.com/news/2017/1/smart_city_lighting_in_wipperfurth, 2017. [Online; accedido el 15-Agosto-2020].

[2] Revista Electricidad. Las luminarias inteligentes comienzan a brillar en Chile. http://www.revistaei.cl/2018/12/21/las-luminarias-inteligentes-comienzan-brillar-chile/, 2018. [Online; accedido el 15-Agosto-2020].

[3] Intendencia de Montevideo. Licitación pública 670/2017. 2017.

[4] Intendencia de Montevideo. Tecnología para Ciudades Inteligentes. https://montevideo.gub.uy/institucional/dependencias/tecnologia-para-ciudades-inteligentes, 2020. [Online; accedido el 15-Agosto-2020].

[5] Schreder. Owlet IoT. https://www.schreder.com/products/owlet-control-solutions, 2020. [Online; accedido el 15-Agosto-2020].

[6] ANSI Approval Date. ANSI/IEC 60529-2004. 2004.

[7] Kanglight. Taurus-36. https://www.kanglight.com/product/taurus-36/, 2020. [Online; accedido el 15-Agosto-2020].

[8] Phillips. Xitanium Led Drivers. https://www.lighting.philips.co.uk/oem-emea/products/house-of-brands/xitanium-led-drivers, 2020. [Online; accedido el 15-Agosto-2020].

[9] LoRa Alliance. https://lora-alliance.org/, 2020. [Online; accedido el 15-Agosto-2020].

[10] Semtech. https://www.semtech.com/, 2020. [Online; accedido el 15-Agosto-2020].

[11] TheThingsNetwork. https://www.thethingsnetwork.org/, 2020. [Online; accedido el 15-Agosto-2020].

[12] Ubidots. https://ubidots.com/, 2020. [Online; accedido el 15-Agosto-2020].

[13] Mathworks. Matlab. https://www.mathworks.com/products/matlab.html, 2020. [Online; accedido el 15-Agosto-2020].

[14] AutoDesk. Eagle. https://www.autodesk.com/products/eagle/overview, 2020. [Online; accedido el 15-Agosto-2020].

[15] PCBWay. https://pcbway.com, 2020. [Online; accedido el 15-Agosto-2020].

[16] Texas Instruments. MSP432P401R. https://www.ti.com/product/MSP432P401R, 2020. [Online; accedido el 15-Agosto-2020].

[17] Texas Instruments. MSP-EXP430G2ET. https://www.ti.com/tool/MSP-EXP430G2ET, 2020. [Online; accedido el 15-Agosto-2020].

[18] Texas Instruments. MSP430G2553. https://www.ti.com/product/MSP430G2553, 2020. [Online; accedido el 15-Agosto-2020].

[19] Texas Instruments. MSP-EXP432P401R. https://www.ti.com/tool/MSP-EXP432P401R, 2020. [Online; accedido el 15-Agosto-2020].

[20] MikroElectronica. MSP432 Clicker. https://www.mikroe.com/clicker-msp432, 2020. [Online; accedido el 15-Agosto-2020].

[21] MikroElectronica. Lora 2 Click. https://www.mikroe.com/lora-2-click, 2020. [Online; accedido el 15-Agosto-2020].

[22] MikroElectronica. RN2903. https://www.microchip.com/wwwproducts/en/RN2903, 2020. [Online; accedido el 15-Agosto-2020].

[23] Semtech. SX1276. https://www.semtech.com/products/wireless-rf/lora-transceivers/sx1276, 2020. [Online; accedido el 15-Agosto-2020].

[24] Microchip. PIC18F46K22. https://www.microchip.com/wwwproducts/en/PIC18F46K22, 2020. [Online; accedido el 15-Agosto-2020].

[25] Digikey. NSL-4960-ND. https://www.digikey.com/product-detail/es/advanced-photonix/NSL-4960/NSL-4960-ND/5039799, 2020. [Online; accedido el 15-Agosto-2020].

[26] On Semiconductors. MC94071-D Single Supply 3.0 V to 44 V Operational Amplifiers. 2014.

[27] Diodes Incorporated. DMN65D8L N-Channel Enhancement Mode Mosfet. 2016.

[28] On Semiconductors. BAT54H Schottky Barrier Diodes. 2014.

[29] Allegro Microsystems. ACS71020: Single Phase, Isolated, Power Monitoring IC with Voltage Zero Crossing and Overcurrent Detection. https://www.allegromicro.com/en/products/sense/current-sensor-ics/zero-to-fifty-amp-integrated-conductor-sensor-ics/acs71020, 2019. [Online; accedido el 15-Agosto-2020].

[30] Texas Instruments. Code Composer Studio (CCS) Integrated Development Environment (IDE). https://www.ti.com/tool/CCSTUDIO, 2020. [Online; accedido el 15-Agosto-2020].

[31] Texas Instruments. Texas Instruments MSP432 Driver Library. https://software-dl.ti.com/msp430/msp430_public_sw/mcu/msp430/MSP432_Driver_Library/latest/exports/driverlib/msp432_driverlib_3_21_00_05/doc/MSP432P4xx/html/driverlib_html/index.html, 2016.

[32] Wikipedia contributors. Sunrise equation — Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/Sunrise_equation, 2020. [Online; accedido el 15-Agosto-2020].

[33] NOAA. https://www.noaa.gov, 2020. [Online; accedido el 15-Agosto-2020].

[34] Arduino. Arduino Due. https://store.arduino.cc/usa/due, 2020. [Online; accedido el 15-Agosto-2020].

[35] Tektelic. KONA Micro IoT, Gateway Enterprise LoRaWAN® Gateway for Mission Critical Deployments. 2020.

[36] Allegro Microsystems. User Manual ASEK71020 Evaluation Board . 2019.

Esta página ha sido intencionalmente dejada en blanco.

Índice de tablas

Esta página ha sido intencionalmente dejada en blanco.

Índice de figuras

Esta es la última página.
Compilado el lunes 21 septiembre, 2020.
http://iie.fing.edu.uy/