



UNIVERSIDAD TECNOLÓGICA METROPOLITANA  
FACULTAD DE INGENIERÍA  
DEPARTAMENTO DE ELECTRICIDAD  
ESCUELA DE ELECTRÓNICA

DISEÑO E IMPLEMENTACIÓN DE UNA SOLUCIÓN IOT (INTERNET DE LAS COSAS) PARA MONITOREO Y ALARMA REMOTA DE FALLAS ELÉCTRICAS  
CON TECNOLOGÍA SIGFOX Y PLATAFORMA UBIOTS

TRABAJO DE TITULACIÓN PARA OPTAR AL TÍTULO DE INGENIERO EN  
ELECTRÓNICA

AUTOR:

ARRIAGADA MARTINEZ, VICTOR

PROFESOR GUÍA:

DURNEY WASAFF, HUGO

SANTIAGO – CHILE

2022

NOTA OBTENIDA:

---

Firma y timbre

autoridad responsable

## ÍNDICE DE CONTENIDOS

<b>1. INTRODUCCIÓN.....</b>	<b>12</b>
1.1 ANTECEDENTES .....	12
1.2 OBJETIVOS GENERALES Y ESPECÍFICOS .....	16
1.2.1 <i>Objetivo General.</i> .....	16
1.2.2 <i>Objetivos Específicos</i> .....	16
<b>2. ESTADO DEL ARTE Y LA TÉCNICA .....</b>	<b>17</b>
2.1 FUNDAMENTOS DEL IoT .....	17
2.1.1 <i>Cómo entender el IoT</i> .....	17
2.1.2 <i>Alcances y limitaciones</i> .....	22
2.1.2.1 Dispositivos de menor tamaño .....	22
2.1.2.2 Baja capacidad de procesamiento .....	23
2.1.2.3 Usos de redes con poca banda ancha .....	24
2.1.2.4 Uso de radiofrecuencia.....	25
2.1.3 <i>Concepto de una aplicación IoT</i> .....	26
2.1.4 <i>Principios del IoT</i> .....	26
2.1.4.1 Bajo costo .....	26
2.1.4.2 Bajo consumo energético.....	27
2.1.4.3 Alta seguridad .....	27
2.2 HARDWARE EN IoT .....	27
2.2.1 <i>Hardware programable o microcontrolador</i> .....	28
2.2.2 <i>Transductores (Sensores y/o actuadores)</i> .....	29
2.2.2.1 Transductor .....	29
2.2.2.2 Sensor.....	29
2.2.2.3 Actuadores .....	30
2.2.3 <i>Alimentación y respaldo de energía.</i> .....	30
2.3 REDES DE TELECOMUNICACIONES .....	31
2.3.1 <i>LTE-M y NB-IOT</i> .....	32
2.3.2 <i>LoRa y SigFox</i> .....	33
2.4 SOFTWARE EN LA NUBE.....	36
2.4.1 <i>Plataformas IoT</i> .....	37

2.4.2	<i>Plataforma IoT Ubidots</i> .....	37
2.4.2.1	Ubidots (Licenciado) .....	38
2.4.2.2	Ubidots STEM .....	38
<b>3.</b>	<b>DISEÑO E IMPLEMENTACIÓN DEL PROTOTIPO .....</b>	<b>40</b>
3.1	<b>DISEÑO DEL DISPOSITIVO IoT.....</b>	<b>40</b>
3.1.1	<i>Medidor de energía PZEM-004T</i> .....	42
3.1.2	<i>Cargador USB 220VAC/5VDC 2A</i> .....	43
3.1.3	<i>Módulo TP4056 + protección DW01A y FS8205A</i> .....	45
3.1.4	<i>Módulo DC/DC Step UP MT3608</i> .....	47
3.1.5	<i>Batería Li-Po 18650</i> .....	48
3.1.6	<i>Módulo LoPy4 de PyCom</i> .....	49
3.1.7	<i>Módulo DS3231</i> .....	50
3.1.8	<i>OLED SSD1306</i> .....	51
3.1.9	<i>Botón Pulsador</i> .....	52
3.1.10	<i>LED WS2812b</i> .....	53
3.1.11	<i>Módulo MicroSD</i> .....	54
3.1.12	<i>Antena 915 MHz + conector</i> .....	55
3.1.13	<i>Resistencia y divisor de tensión</i> .....	56
3.1.14	<i>Módulo USB/TTL CP2102</i> .....	57
3.1.15	<i>Implementación del dispositivo IoT</i> .....	58
3.1.16	<i>Metodología de montaje</i> .....	59
3.1.17	<i>Base de batería LiPo</i> .....	60
3.1.18	<i>Jumpers de ajustes y aislación</i> .....	61
3.1.19	<i>Puerto de comunicación UART</i> .....	61
3.1.20	<i>Leds RGB para visualización de estados</i> .....	62
3.1.21	<i>Botones pulsadores para control</i> .....	63
3.1.22	<i>Pantalla OLED para visualización de información</i> .....	63
3.1.23	<i>Reloj de tiempo real</i> .....	64
3.1.24	<i>Memoria MicroSD y archivos de depuración</i> .....	65
3.1.25	<i>Fusible de protección</i> .....	66
3.2	<b>DISEÑO Y PROGRAMACIÓN DEL FIRMWARE PARA EL DISPOSITIVO IoT .....</b>	<b>68</b>
3.2.1	<i>Preparación</i> .....	68
3.2.1.1	<i>Descarga e instalación de PyCom Firmware Update</i> .....	68
3.2.1.2	<i>Descarga e instalación de controlador CP2102</i> .....	70

3.2.1.3	<i>Descarga e instalación de editor de código Atom.....</i>	71
3.2.1.4	<i>Descarga e instalación del Plugin Pymakr para Atom.....</i>	73
3.2.1.5	<i>Cargar MicroPython PyCom en microcontrolador .....</i>	73
3.2.1.6	<i>Primer programa de prueba (Hola Mundo).....</i>	75
3.2.1.7	<i>Cargar programa a microcontrolador.....</i>	76
3.2.1.8	<i>Líneas de comandos MicroPython .....</i>	76
3.2.2	<i>Diagrama del programa .....</i>	78
3.2.2.1	<i>boot.py.....</i>	78
3.2.2.2	<i>main.py.....</i>	79
3.2.2.3	<i>Subrutina mensaje_sigfox_15_min.....</i>	80
3.2.2.4	<i>Subrutina todo .....</i>	81
3.2.2.5	<i>Subrutina clock.....</i>	84
3.2.2.6	<i>Subrutina voltaje_AC.....</i>	85
3.2.2.7	<i>Subrutina voltaje_batería.....</i>	86
3.2.2.8	<i>Subrutina botón .....</i>	87
3.2.2.9	<i>Subrutina estado_leds.....</i>	88
3.2.3	<i>Metodología de programación .....</i>	90
3.2.3.1	<i>Modo de testeo de módulos .....</i>	90
3.2.3.2	<i>Modo integración.....</i>	91
3.2.3.3	<i>Modo prueba en terreno y estrés.....</i>	92
<b>4.</b>	<b>RESULTADOS Y CONCLUSIONES .....</b>	<b>94</b>
<b>5.</b>	<b>BIBLIOGRAFÍA.....</b>	<b>105</b>

## ÍNDICE DE ILUSTRACIONES

<i>Ilustración 1 - Partes del IoT.....</i>	12
<i>Ilustración 2 - Partes de un dispositivo IoT .....</i>	13
<i>Ilustración 3 - Estructura típica de una aplicación IoT .....</i>	14
<i>Ilustración 4 - Plataforma IoT en la nube.....</i>	15
<i>Ilustración 5 – Ejemplo de una aplicación IoT en vehículos inteligentes.....</i>	18
<i>Ilustración 6 – “Dashboard” o tablero de control típico de una aplicación IoT .....</i>	19
<i>Ilustración 7 - Sensores en un smartphone .....</i>	20
<i>Ilustración 8 - Evolución de las comunicaciones móviles.....</i>	21
<i>Ilustración 9 - Sensor adhesivo para monitoreo.....</i>	23
<i>Ilustración 10 - Redes de comunicaciones IoT.....</i>	24
<i>Ilustración 11 - Ejemplo de una aplicación IoT en la agroindustria .....</i>	25
<i>Ilustración 12 - Estructura LoRaWAN [9] .....</i>	34
<i>Ilustración 13 - Estructura SigFox.....</i>	36
<i>Ilustración 14 – Estructura del dispositivo IoT .....</i>	40
<i>Ilustración 15 – Disposición de los módulos en placa reticulada .....</i>	41
<i>Ilustración 16 - Medidor de energía PZEM-004T.....</i>	42
<i>Ilustración 17 - cargador USB.....</i>	43
<i>Ilustración 18 - PCB del cargador USB .....</i>	43
<i>Ilustración 19 - PCB del cargador USB modificado.....</i>	44
<i>Ilustración 20 - Fuente de alimentación .....</i>	44
<i>Ilustración 21 - Módulo TP4056 .....</i>	45
<i>Ilustración 22 - Chip TP4056, DW01A y FS8205A.....</i>	46
<i>Ilustración 23 - Módulo MT3608.....</i>	47
<i>Ilustración 24 - Batería LiPo 18650.....</i>	48
<i>Ilustración 25 - Módulo LoPy4 de PyCom .....</i>	49
<i>Ilustración 26 - Estructura interna del módulo LoPy4 .....</i>	50
<i>Ilustración 27 - Módulo RTC DS3231 .....</i>	51
<i>Ilustración 28 - Pantalla OLED SSD1306 .....</i>	51
<i>Ilustración 29 - Botón Pulsador .....</i>	52

<i>Ilustración 30 - LED RGB WS2812b.....</i>	53
<i>Ilustración 31 - Módulo MicroSD.....</i>	54
<i>Ilustración 32 - Módulo MicroSD Modificado .....</i>	55
<i>Ilustración 33 - Antena 915 MHz.....</i>	55
<i>Ilustración 34 - Divisor de tensión .....</i>	56
<i>Ilustración 35 - Módulo USB/TTL CP2102 .....</i>	57
<i>Ilustración 36 - Tarjeta del prototipo.....</i>	58
<i>Ilustración 37 - Montaje del prototipo.....</i>	59
<i>Ilustración 38 - Fuente de alimentación dañada.....</i>	60
<i>Ilustración 39 - Base para batería LiPo 18650 .....</i>	60
<i>Ilustración 40 - Jumper del voltaje batería y jumper boot.....</i>	61
<i>Ilustración 41 - Conector TTL.....</i>	62
<i>Ilustración 42 - Leds RGB .....</i>	62
<i>Ilustración 43 - Botones de control.....</i>	63
<i>Ilustración 44 - Pantalla OLED.....</i>	64
<i>Ilustración 45 - Reloj de tiempo real.....</i>	64
<i>Ilustración 46 - Módulo microSD modificado.....</i>	65
<i>Ilustración 47 - Archivos DEBUG de la microSD .....</i>	66
<i>Ilustración 48 - Fusible de protección .....</i>	67
<i>Ilustración 49 - Pantalla de bienvenida Pycom .....</i>	68
<i>Ilustración 50 - Pantalla del sitio web para descarga de software .....</i>	69
<i>Ilustración 51 - Pantalla de instalación de software .....</i>	69
<i>Ilustración 52 - Pantalla de descarga de drivers LoPy .....</i>	70
<i>Ilustración 53 - Pantalla de instalación de drivers .....</i>	71
<i>Ilustración 54 - Pantalla de sitio web de descarga entorno de programación.....</i>	72
<i>Ilustración 55 - Pantalla de instalación de ATOM .....</i>	72
<i>Ilustración 56 - Pantalla de instalación plugin "Pymakr" .....</i>	73
<i>Ilustración 57 - Conexión del módulo LoPy4 vía USB.....</i>	74
<i>Ilustración 58 - Archivos de testeo .....</i>	90
<i>Ilustración 59 - Archivos principales.....</i>	92
<i>Ilustración 60 - Archivos de librerías.....</i>	93
<i>Ilustración 61 - Multímetro de medición, medición voltaje red eléctrica y batería .....</i>	102

## ÍNDICE DE DIAGRAMAS

<i>Diagrama 1 - Boot.py.....</i>	78
<i>Diagrama 2 - Main.py.....</i>	79
<i>Diagrama 3 - Subrutina: mensaje_sigfox_15_min.....</i>	81
<i>Diagrama 4 – Subrutina: todo (parte 1) .....</i>	82
<i>Diagrama 5 – Subrutina: todo (parte 2) .....</i>	83
<i>Diagrama 6 - Subrutina: clock.....</i>	84
<i>Diagrama 7 - Subrutina: voltaje_AC.....</i>	85
<i>Diagrama 8 - Subrutina: voltaje_batería.....</i>	86
<i>Diagrama 9 - Subrutina: botón .....</i>	87
<i>Diagrama 10 - Subrutina: estado_leds (Parte 1).....</i>	88
<i>Diagrama 11 - Subrutina: estado_leds (Parte 2).....</i>	89

## ÍNDICE DE CÓDIGOS

<i>Código 1 - &lt;Hola Mundo&gt;</i> .....	75
<i>Código 2 - Consola</i> .....	77

## ÍNDICE DE TABLAS

<i>Tabla 1 - Comparación de tecnologías NB-IOT vs LTE-M [8]</i> .....	33
<i>Tabla 2 - Planificación de fallas programables Periodo 1</i> .....	95
<i>Tabla 3 - Planificación de fallas programables Periodo 2</i> .....	96
<i>Tabla 4 - Resumen del error en medición de voltaje red eléctrica y batería</i> .....	102

## ÍNDICE DE GRÁFICOS

<i>Gráfico 1 - Evolución de dispositivos conectados a Internet según datos del IBSG de CISCO</i> .....	20
<i>Gráfico 2 - Mensajes perdidos y enviados por día, (periodo 21)</i> .....	98
<i>Gráfico 3 - Calidad de la señal en % por día, (periodo 1)</i> .....	98
<i>Gráfico 4 - Relación señal ruido SNR por día, (periodo 1)</i> .....	99
<i>Gráfico 5 - Fuerza de la señal RSSI por día, (periodo 1)</i> .....	99
<i>Gráfico 6 - Mensajes perdidos y enviados por día, (periodo 2)</i> .....	100
<i>Gráfico 7 - Calidad de la señal en % por día, (periodo 2)</i> .....	100
<i>Gráfico 8 - Relación señal ruido SNR por día, (periodo 2)</i> .....	101
<i>Gráfico 9 - Fuerza de la señal RSSI por día, (periodo 2)</i> .....	101

## **RESUMEN**

En el presente Trabajo de Título se explicarán los principios y fundamentos del internet de las cosas (IoT), cómo entenderlo, alcances y limitaciones que tiene, concepto de una aplicación IoT, y sus partes fundamentales como hardware (Módulo LoPy4), software (Plataforma Ubidots) y telecomunicaciones (Red SigFox).

Se pretende exponer el proceso diseño e implementación de un prototipo electrónico para una aplicación IoT, que monitorea el estado del voltaje en una red eléctrica monofásica. Se describen las partes electrónicas, módulos, características y funcionamiento de cada componente, para así poder capturar, procesar y tomar decisiones con los datos y eventos de la red eléctrica. Toda esta información se enviará a una plataforma IoT en la cual se puede procesar y analizar los datos y/o eventos recibidos para poder determinar acciones frente un corte o subidas y bajas de tensiones.

Se explicará a grandes rasgos la lógica del firmware que se utilizará en el dispositivo IoT para lograr monitorear la red, procesar datos de captura y envío así como un datalogger para tener de forma local la información capturada y así compararla con la recibida por la red SigFox.

Finalmente se describe un resumen de los resultados capturados, metodología de fallas programables (Dispositivo SONOFF), problemas reflejados durante el proceso, información de métricas en la red SigFox como por ejemplo calidad de señal, mensajes recibidos, relación señal ruido, error de la medición de la red y batería interna del dispositivo (con multímetro y calculado desde los sensores) y finalmente una breve conclusión del proyecto en conjunto con recomendaciones y mejoras que se podrían aplicar en el futuro.

## **ABSTRACT**

This work introduces, the principles and fundamentals of the Internet of Things (IoT), how to understand it, its scope and limitations, the concept of an IoT application, and its fundamental parts such as hardware (LoPy4 Module), software (Platform Ubidots) and telecommunications (SigFox Network).

Desing and fabrication process of an electronic prototype for an IoT application is presented, which monitors the state of the voltage in a single-phase electrical network. The electronic parts, modules, characteristics and operation of each component are described, which allow to capture, process and take decisions with the data and events of the electrical network. All this information will be sent to an IoT platform in which the data and/or events received can be processed and analyzed in order to determine actions in the event of blackout or power instability.

The logic of the firmware that will be used in the IoT device to monitor the network and process input/output data, will be explained, as well as the datalogger to have the captured information locally, and thus compare it with the one received by the SigFox.

Finally, a summary of the captured results is described, programmable fault methodology (SONOFF device), problems reflected during the process, information on metrics in the SigFox network such as signal quality, received messages, signal-to-noise ratio, measurement error of the network and internal battery of the device (with a multimeter and calculated from the sensors), and finally a brief conclusion of the project together with recommendations and improvements that could be addressed in the future.

## 1. INTRODUCCIÓN

### 1.1 Antecedentes

La automatización y domótica ya existen hace muchos años, pero la integración con las aplicaciones en la nube provoca que estas tecnologías se conviertan en lo que llamamos hoy en día IoT (Internet of Things) o en español Internet de las cosas [1]. Este término se usa desde que existen más dispositivos conectados a Internet que personas, alrededor del año 2009.

El IoT es la conexión de objetos a Internet (sensores, electrodomésticos, máquinas, sistemas de control, luminarias, vehículos, sistemas de riego, etc.), usando sensores para recopilar información, acceder a ella en línea, analizarla, procesarla, almacenarla en grandes bancos de datos y finalmente sintetizarla para poder tomar decisiones ya sea una persona o a través de un sistema con inteligencia computacional [2].

El IoT se divide en 3 partes fundamentales que se muestran en la Ilustración 1: Dispositivo o hardware, Telecomunicaciones, y la famosa Cloud o nube.

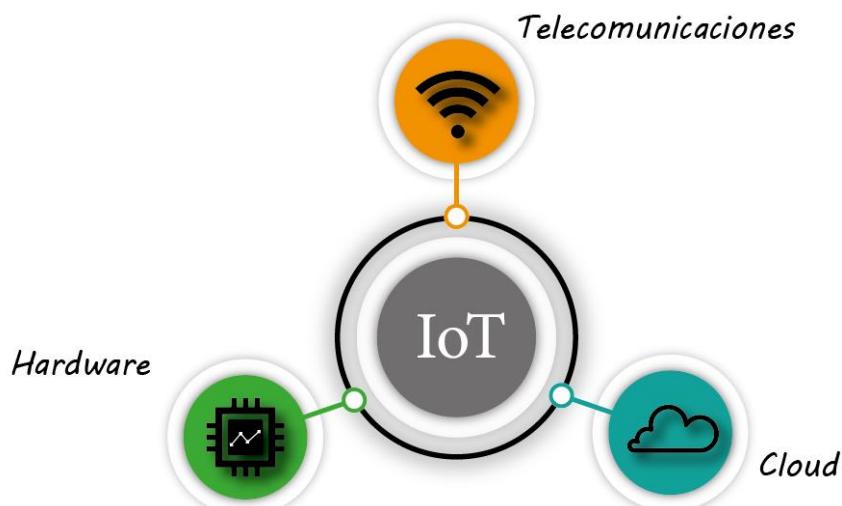
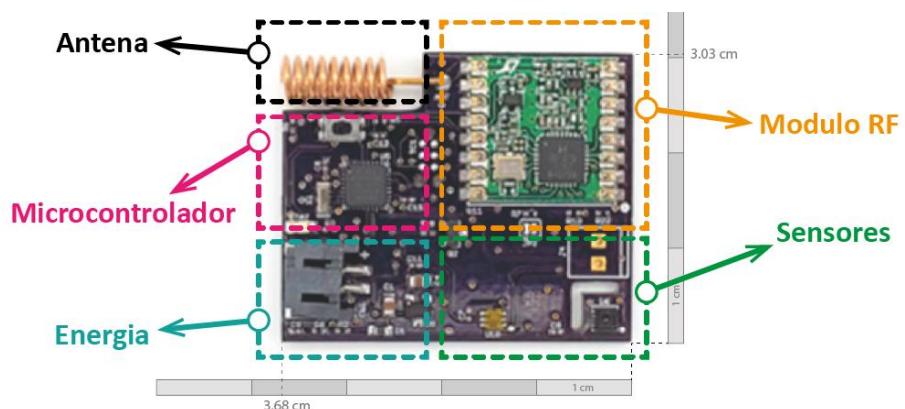


Ilustración 1 - Partes del IoT

Los dispositivos hardware del IoT son normalmente módulos que permiten la captación de datos con sensores y los envían por diversas redes de comunicaciones a través de puntos de acceso (gateway), los que a su vez envían la información a servidores distribuidos.

Tal como se muestra en la Ilustración 2, todos los dispositivos típicamente contienen un microprocesador, un módulo de radiofrecuencia u otro que permita enviar datos a un gateway o directamente a Internet, y uno o varios sensores más un sistema de baterías o alimentación alternativa (solar, eólica, etc.).



En telecomunicaciones existen muchas tecnologías que se usan en las soluciones IoT, entre las más conocidas tenemos redes de WiFi, Celular (2G, 3G, 4G, 5G), Bluetooth, BLE (Bluetooth de baja energía), ZigBee, LPWAN (Low Power Wide Area Networks), entre otros.

En particular, LPWAN corresponde a un tipo de redes más usadas en una aplicación IoT. Un ejemplo de esta se describe en la Ilustración 3. Este tipo de redes se caracteriza por un alto alcance, con un bajo consumo energético, es especial para tramas cortas de datos o mejor dicho para el IoT. Dentro de LPWAN destaca la tecnología Sigfox, que ya se ha comenzado a desplegar experimentalmente en algunas zonas de Chile, implementando progresivamente

toda una red de puntos con antenas base de cobertura amplia (varias cuadras) que dan acceso al sistema de servidores Sigfox a través de Internet.

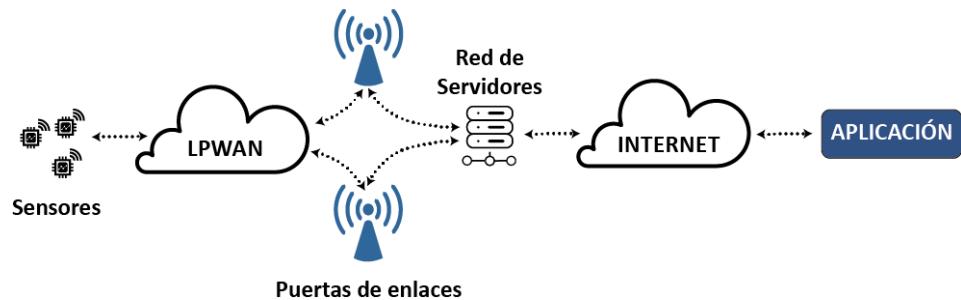


Ilustración 3 - Estructura típica de una aplicación IoT

Una plataforma IoT, como se muestra en la Ilustración 4, podría ser vista sencillamente como un servidor con una aplicación y servicios basados en recibir y enviar datos desde y hacia un entorno local que está bajo monitoreo y control a través de Internet. Cuando a lo anterior agregamos la capacidad de almacenamiento distribuido de datos podemos hablar de Cloud o nube en español. Existen muchos servicios de nube e incluso uno mismo puede crear de forma sencilla pero muy básica su propia nube en casa (similar a la tendencia denominada “edge computing”). Entre las más usadas, simples y estables tenemos a Ubidots ([www.ubidots.com](http://www.ubidots.com)). Ubidots es un servicio Cloud que nos permite conectar nuestros dispositivos con cualquier tecnología de telecomunicaciones a su servidor para enviarles métricas de cualquier tipo (sensores), también permite gestionarlas y analizarlas para crear eventos que nos informen ante cualquier cosa que pase con el dispositivo o con las métricas. Esta nube permite conexiones a través de protocolos como MQTT, HTML y TCP/UDP, a través de funciones como API REST.



Ilustración 4 - Plataforma IoT en la nube

## **1.2 Objetivos generales y específicos**

### **1.2.1 Objetivo General**

Diseñar e implementar una solución IoT (Internet de las cosas) para monitoreo y alarma remota, que permita responder oportunamente ante fallas relevantes de la provisión de energía eléctrica que pueden afectar gravemente procesos electrodependientes de un determinado entorno, para esto utilizando tecnología Sigfox y la plataforma Ubidots.

### **1.2.2 Objetivos Específicos**

- Diseñar e implementar, a nivel de hardware y firmware, un dispositivo de monitoreo eléctrico no intrusivo, que integre un grupo de diversos sensores de parámetros y estados, compatibilice el envío de datos a través de tecnología Sigfox y que cuente con un sistema autónomo de energía basado en baterías.
- Integrar el dispositivo de hardware desarrollado con la plataforma Ubidots, para efectos de lograr capacidad y robustez tanto en el almacenamiento de datos como en las comunicaciones desde y hacia el usuario final, aprovechando las capacidades de servicios web presentes en la nube.
- Documentar y sistematizar los resultados del funcionamiento de esta solución IoT, identificando sus potencialidades, eventuales problemas, tipos de fallas y recomendando posibles soluciones.

## **2. ESTADO DEL ARTE Y LA TÉCNICA**

### **2.1 Fundamentos del IoT**

#### **2.1.1 Cómo entender el IoT**

Los objetos forman parte de nuestra vida cotidiana y siempre están generando información, por ejemplo, un termómetro de mercurio, o un cepillo de dientes que se descolora al gastarse o simplemente saber si se cortó el suministro eléctrico en nuestros hogares, toda esa información estaba fuera de nuestro alcance, hasta ahora.

Imaginemos una ciudad inteligente en la que los sensores detecten fugas en las cañerías de agua o gas, vallas publicitarias que cambian sus anuncios según el perfil del consumidor, midan tráfico en las ciudades como se observa en la Ilustración 5, con vehículos inteligentes capaces de conectarse a las redes, comunicarse entre ellos, entregar información a los conductores, entre otras capacidades.



*Ilustración 5 – Ejemplo de una aplicación IoT en vehículos inteligentes*

A medida que avanza la tecnología, los costos y el tamaño de los sensores se reducen, permitiendo integrarlos en cualquier objeto. Esto implica que cualquier objeto puede ser una fuente de datos.

El IoT consiste en que las cosas u objetos tengan conexión a Internet en cualquier momento y lugar por medio de redes alámbricas o inalámbricas, conectando el mundo físico con el mundo digital.

Esto permite reunir, analizar y distribuir datos que podemos convertir en información y conocimiento,

Un ejemplo a esto es un dashboard como en la Ilustración 6, en él se muestra el monitoreo de variables como voltaje, estados de la red eléctrica, intensidad de señal inalámbrica y relación de señal a ruido.

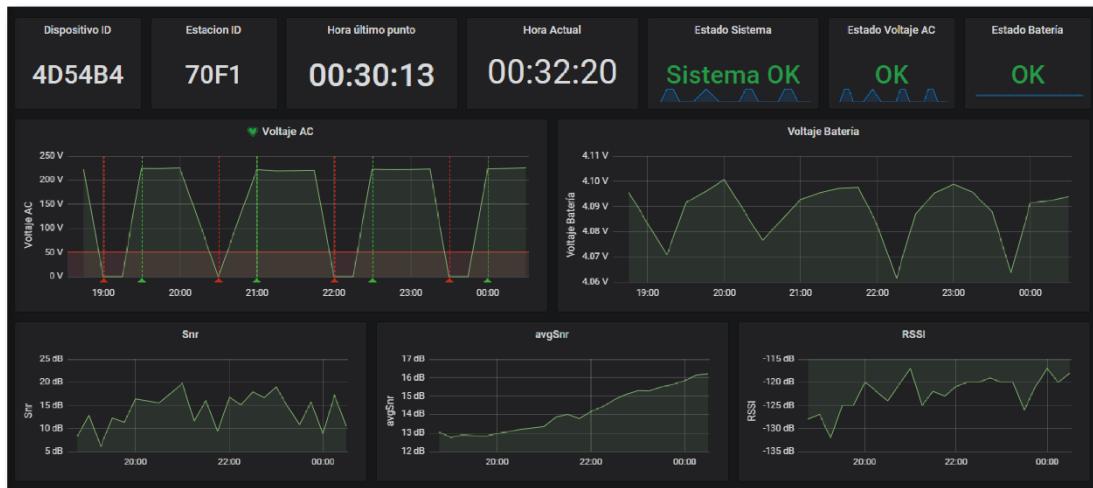


Ilustración 6 – “Dashboard” o tablero de control típico de una aplicación IoT

El tamaño, los costos y el consumo energético del hardware se han reducido drásticamente por lo que ahora es posible fabricar dispositivos electrónicos diminutos a un costo muy reducido.

El crecimiento explosivo de los smartphones y las tablet elevó la cantidad de dispositivos conectados a Internet.

Los smartphones y tablet contienen sensores que recopilan información y la envían a Internet (por ejemplo: temperatura, ubicación GPS, presión, pantalla táctil, sonido, videos, movimientos iniciales, etc.). En la Ilustración 7 se describen sensores típicos de un smartphone.

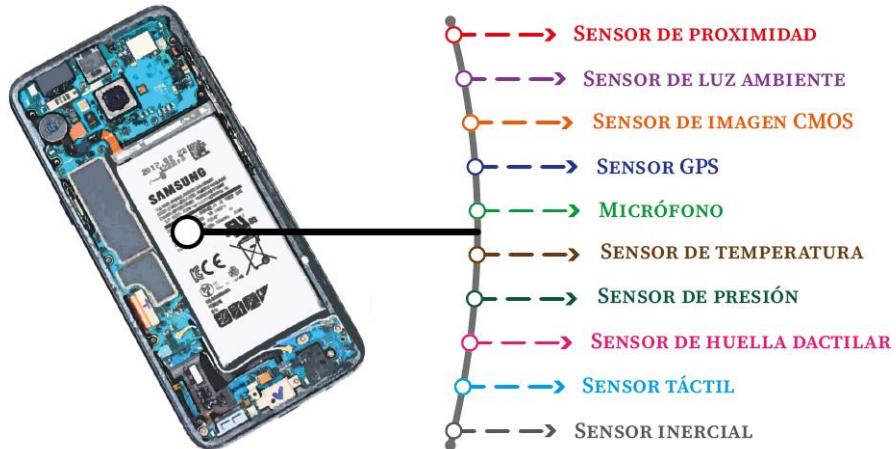


Ilustración 7 - Sensores en un smartphone

Según el IBSG (Internet Business Solutions Group) de Cisco, el IoT nace en algún punto del tiempo en el que se conectaron más cosas que personas a Internet [3]. El siguiente grafico muestra los dispositivos conectados a internet a lo largo del tiempo.

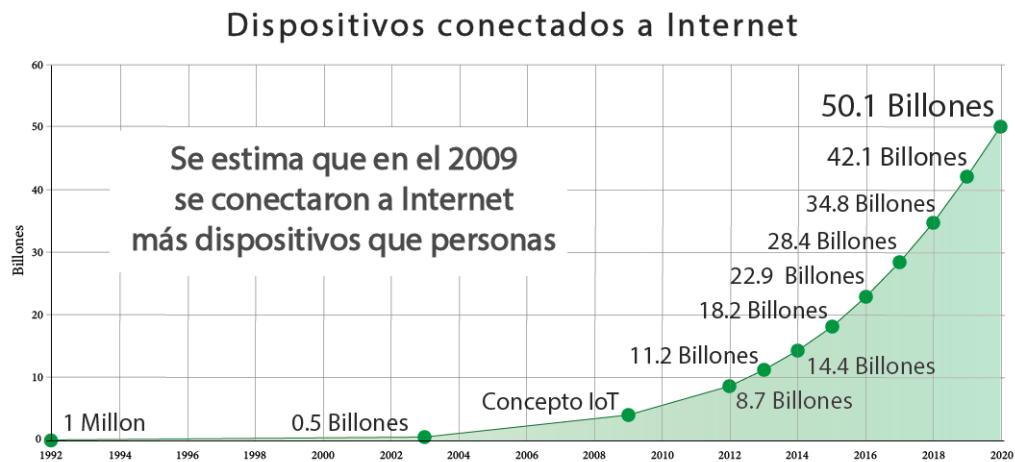


Gráfico 1 - Evolución de dispositivos conectados a Internet según datos del IBSG de CISCO

Con la evolución de las comunicaciones móviles, como se muestra en la Ilustración 8, ha aumentado considerablemente el número de smartphones conectados a Internet. Según datos de SUBTEL, a diciembre de 2020 en Chile los abonados móviles superaban los 25 millones [4].

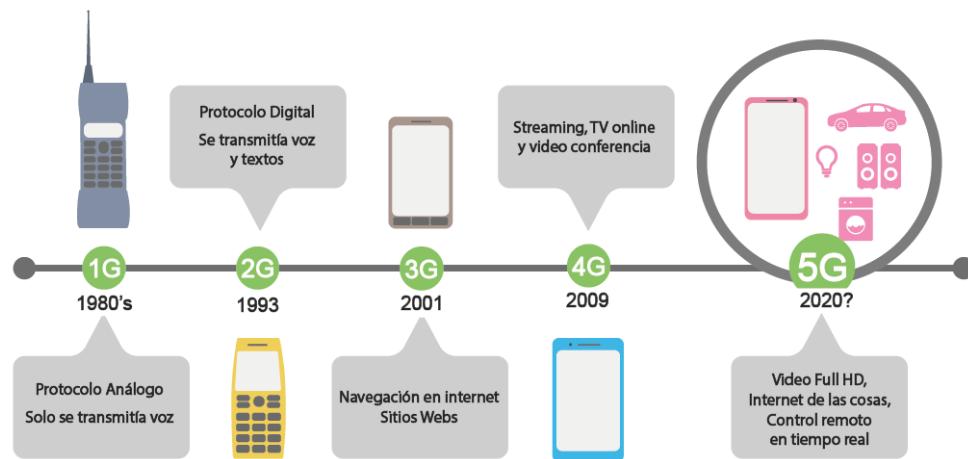


Ilustración 8 - Evolución de las comunicaciones móviles

Se estima que en algún punto del año 2009 nació el IoT.

En resumen, el concepto del IoT es tener dispositivos u objetos conectados a Internet y que son de bajo costo y bajo consumo energético, que nos brindan información importante para analizarla y tomar decisiones.

En el IoT existen aplicaciones que requieren gran ancho de banda para el envío de información, por ejemplo tenemos los vehículos autónomos, sistemas de tracking y monitoreo de procesos delicados, entre otros.

## **2.1.2 Alcances y limitaciones**

Al conectar los objetos a nuestro alrededor podremos obtener más información sobre lo que sucede en nuestro entorno, permitiendo tomar mejores decisiones tanto en nuestro día a día como en los procesos industriales. En general cualquier objeto es susceptible a ser conectado.

Los sensores abarcan todos los ámbitos, desde hogares hasta entornos de trabajo y espacios públicos.

Esto permitirá mejorar la calidad de vida por medio de aplicaciones orientadas tanto al consumidor como a la industria.

El IoT, al estar enfocado a conectar objetos, tiene características que lo hace atractivo para muchas aplicaciones, pero que también pueden verse como limitantes.

Algunas características son:

- Dispositivos de menor tamaño
- Baja capacidad de procesamiento
- Uso de redes con poca banda ancha.
- Uso de radiofrecuencia

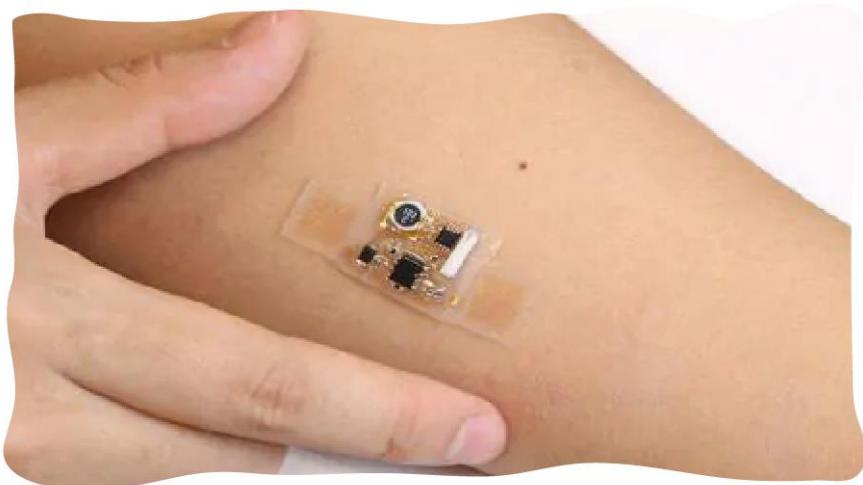
### **2.1.2.1 Dispositivos de menor tamaño**

Los sensores consumen energía y a medida que su tamaño disminuye, el consumo se convierte en un factor más limitante.

Con baterías más pequeñas, requieren consumo de energía más bajos para garantizar el mayor tiempo de funcionamiento de forma autónoma.

La reducción en tamaño, en algún momento, se enfrentará a la limitación física de los componentes basados en silicio.

En la Ilustración 9 se ve un ejemplo de sensores del futuro, el cual monitorea métricas vitales, y que se adhiere con adhesivo en la piel de las personas.



*Ilustración 9 - Sensor adhesivo para monitoreo*

#### **2.1.2.2 Baja capacidad de procesamiento**

La mayoría de las aplicaciones IoT no requieren que los dispositivos tengan gran capacidad de procesamiento.

La tarea principal de las aplicaciones IoT es recolectar y enviar la información hacia la nube. Una vez en la nube, la información puede ser sometida a análisis automatizado a través de algoritmos de inteligencia computacional residentes en los servidores. No obstante, también es posible dotar de cierta inteligencia a los dispositivos para que también puedan efectuar análisis y tomar decisiones por medio de la información recolectada, pero esto supone una carga de

procesamiento residente en el dispositivo con la consiguiente demanda de consumo de energía.

#### **2.1.2.3 Usos de redes con poca banda ancha**

Al conectar objetos que nos brindan información de su entorno, por lo general, no se necesita de un gran ancho de banda para que se comuniquen.

El IoT no está enfocado en enviar grandes cantidades de información (videos, audio, imágenes), lo importante es el envío de pequeños paquetes de datos.

Actualmente existen múltiples redes de comunicación inalámbrica orientadas al IoT (LoRa, NB-IoT, LTE-M y SigFox son las más conocidas) que solo permiten el envío de pequeños paquetes de datos, con costos relativamente bajos. En la Ilustración 10 se describen los logos de cada red mencionada anteriormente.



*Ilustración 10 - Redes de comunicaciones IoT*

En este trabajo, se decidió utilizar la red SigFox en conjunto con la plataforma Ubidots ([www.ubidots.com](http://www.ubidots.com)) para implementar el prototipo propuesto. En particular, cabe señalar que se decidió usar SigFox pues a través de una colaboración universidad-empresa, en algunas dependencias de la UTEM fueron

instalados puntos de acceso a la red SigFox que quedarían disponibles para desarrollos experimentales.

#### 2.1.2.4 Uso de radiofrecuencia

El uso de comunicación por radiofrecuencia es uno de los principales atractivos del IoT, pero también puede ser una de sus principales limitantes.

Dependiendo de la red de comunicación utilizada podemos tener:

- Menor o mayor requerimiento de ancho de banda.
- Menor o mayor radio de cobertura para la instalación de dispositivos.
- Límite de dispositivos conectados usando el canal simultáneamente.

En la Ilustración 11 podemos ver un ejemplo de una aplicación IoT, pero con la gran dificultad de estar alejada de una zona urbana. A raíz de esto, la comunicación inalámbrica de la red de sensores debe ser muy específica y cumpliendo con los requerimientos que se necesitan.



Ilustración 11 - Ejemplo de una aplicación IoT en la agroindustria

Es importante tener presente estas limitaciones al momento de seleccionar el método de comunicación adecuado, dependiendo de la solución IoT a implementar.

### **2.1.3 Concepto de una aplicación IoT**

Para entender el IoT desde un punto de vista más técnico, es necesario comprender los componentes principales que conforman una aplicación IoT, a saber:

- Hardware (dispositivo o sensores)
- Telecomunicaciones (tipo de medio, canal)
- Software en la nube (servidores, aplicaciones, etc.)

Estos tres componentes juegan un papel muy importante dentro de una solución IoT, porque definen cual será la aplicación final.

### **2.1.4 Principios del IoT**

Se puede mencionar tres principios en el IoT que permiten su expresión y aceptación en el terreno de las tecnologías de la información y la comunicación:

- Bajo costo
- Bajo consumo energético
- Alta seguridad

#### **2.1.4.1 Bajo costo**

Los avances tecnológicos en la miniaturización y la industrialización a gran escala han permitido la reducción de costos de fabricación de los componentes electrónicos necesarios. De esta forma los objetos se comunican de forma más cotidiana y masiva cuando generan la información.

#### **2.1.4.2 Bajo consumo energético**

Uno de los retos dentro del IoT son el consumo de energía de los millones de objetos conectados que estarán en nuestro entorno generando información.

Se han desarrollado en los últimos años componentes electrónicos (microcontroladores, sensores, etc.) dedicados al IoT con bajos consumos energéticos.

Se espera que los dispositivos IoT sean capaces de generar su propia energía, o capturarla del entorno (energy harvesting), lo que permitiría a estos dispositivos funcionar de forma autónoma durante más tiempo.

#### **2.1.4.3 Alta seguridad**

A medida que conectamos más dispositivos a Internet, surgen nuevas oportunidades para explotar potenciales vulnerabilidades de seguridad.

Uno de los aspectos más importantes es la seguridad y la confiabilidad de la información generada por millones de dispositivos de nuestro alrededor.

El desarrollo de aplicaciones del IoT debe garantizar que los dispositivos no expongan la información de sus usuarios ni de otras personas a potenciales daños.

## **2.2 Hardware en IoT**

El primer elemento es el hardware, que engloba a los dispositivos o sensores. Sus principales características están relacionadas con:

- Está orientado a capturar información tanto del entorno como del objeto en el que se encuentra integrado.

- Actúa como desencadenante de acciones o eventos.
- Aporta información sobre localización, lo que permite expandir el rango de aplicaciones.

Las características físicas del hardware (tamaño, peso, materialidad, estanqueidad, etc.) pueden limitar su uso en ciertos entornos (exteriores, entornos húmedos, fríos, explosivos, etc.) por lo que es muy importante considerar las características del hardware de acuerdo con su aplicación.

En un dispositivo IoT podemos encontrar los siguientes componentes:

- Hardware programable o microcontrolador
- Sensores y/o actuadores
- Alimentación y/o respaldo de energía

### **2.2.1 Hardware programable o microcontrolador**

Entre los módulos más usados y compatibles con la tecnología SigFox, existe el módulo LoPy4 de Pycom (<https://docs.pycom.io/index.html>). Este módulo es una tarjeta de desarrollo habilitada para lenguaje MicroPython y que es compatible con LoRa, SigFox, WiFi y Bluetooth).

LoPy4 contiene:

- Un ESP32, microcontrolador de doble núcleo de 32 bit fabricado por la empresa Espressif. Este microcontrolador contiene comunicación WIFI y Bluetooth.
- Un módulo Lora & SigFox SX1276 fabricado por la empresa Semtech.
- Memoria Flash de 8 MB, para cargar el Firmware.

## **2.2.2 Transductores (Sensores y/o actuadores)**

### **2.2.2.1 Transductor**

Un transductor se define como aquel dispositivo que es capaz de convertir una variable física en otra que tiene un dominio diferente [5].

### **2.2.2.2 Sensor**

Un sensor es un dispositivo o transductor de entrada, que provee una salida manipulable de la variable física medida.

Dentro de los sensores tenemos:

- Sensores piezorresistivo
- Sensores capacitivos
- Sensores piezoeléctricos
- Sensores ultrasónicos
- Sensores magnéticos
- Sensores termoeléctricos
- Sensores fotoeléctricos
- Sensores químicos

Con estos tipos de sensores podemos medir variables físicas y obtener en su salida variables útiles para procesar.

Clasificación de los sensores según la variable física a medir:

- Sensores de Posición, velocidad y aceleración.
- Sensores de nivel y proximidad.
- Sensores de humedad y temperatura.
- Sensores de fuerza y deformación.
- Sensores de flujo y presión

- Sensores de color, luz y visión.
- Sensores de gas y pH
- Sensores biométricos.
- Sensores de corriente.

Un sensor usado comúnmente es el PZEM-004t [6], el cual mide variables físicas de voltaje y corriente alternos, y en su salida provee valores digitales, utilizando un protocolo de comunicación compatible con microcontroladores. Este módulo contiene todo el circuito eléctrico y electrónico para proteger y aislar la alta potencia y evitar cortocircuitos y peligros de electrocución.

#### **2.2.2.3 Actuadores**

Un actuador es un dispositivo o transductor capaz de transformar energía en la activación de un proceso, con la finalidad de generar un efecto sobre algún elemento externo.

Hay tres tipos de actuadores utilizados en soluciones IoT:

- Hidráulico: utiliza la presión del líquido para realizar el movimiento mecánico, se emplean cuando se necesita potencia.
- Neumático: utiliza aire comprimido a alta presión para permitir el funcionamiento mecánico.
- Eléctrico: impulsado por un motor que convierte la energía eléctrica para el funcionamiento mecánico.

#### **2.2.3 Alimentación y respaldo de energía**

La energía es sumamente importante en soluciones IoT. El consumo de cada dispositivo es despreciable, pero cuando la variable cantidad de dispositivos y tiempo de uso entran en la ecuación, ya no es tan despreciable. Por ese motivo

es que hasta el día de hoy muchas empresas investigan nuevas formas de almacenar y generar energía para dispositivos IoT.

Las baterías Li-Ion y Li-Po son las más usadas para IoT, pues permiten almacenar gran cantidad de energía y su costo de fabricación se ha reducido considerablemente en los últimos años.

Una de las baterías más usadas y fácil de conseguir en el mercado son las Li-Ion 18650. Estas baterías recargables son de un mayor tamaño, pero muy parecidas a las populares AA.

La batería 18650 tiene un voltaje nominal de 3.7V continua y su capacidad varía entre 1000 y 4000 mAh, siendo las más comunes las de 2300 mAh.

Para cargar este tipo de baterías es necesario un controlador de carga y descarga, por ejemplo, el TP4056, ya que estas baterías son muy delicadas a su voltaje, no pueden superar los 4.3V para carga y no se pueden descargar por debajo de 2.5V. Si esto sucede se corre el riesgo de dañar la batería e incluso que reviente.

En condiciones normales, cuando la batería está 100% cargada, su voltaje es de 4.2V y cuando ya está descargada completamente llega a 3.0V.

El módulo TP4056 es un dispositivo que permite cargar una batería Li-Ion o LiPo de una celda con una fuente de 5V, y su corriente máxima de carga es de 1A. Existen módulos que agregan un circuito integrado (8205A) para proteger la batería de bajo voltaje.

### **2.3 Redes de Telecomunicaciones**

Son las responsables de dotar de vida a los dispositivos, al permitirle transmitir su información y generar datos valiosos.

En las aplicaciones IoT las redes más utilizadas hoy en día son las LPWAN (Low-Power Wide-Area Network). Estas redes tienen la característica de tener bajo ancho de banda, lo que permite su funcionamiento con un consumo energético mucho más bajo que las redes comunes que se utilizan para la transferencia de grandes cantidades de datos como imágenes y videos.

Las redes LPWAN en el rango de frecuencias licenciadas más conocidas son:

- LTE-M (Long Term Evolution for Machines)
- NB-IOT (Narrowband-IOT)

Y en el rango de frecuencias no licenciadas (ISM - Industrial, Scientific and Medical)

- LoRa
- Sigfox

### **2.3.1 LTE-M y NB-IOT**

Estas redes trabajan a una frecuencia de 700MHz y ambas utilizan parte de la banda 4G, permitiendo que el uso para dispositivos IoT sea mayor y a un bajo costo.

En la Tabla 1, se resumen las diferencias entre estas dos redes.

## COMPARACIÓN DE TECNOLOGÍAS NB-IOT vs LTE-M

	NB-IOT	LTE-M
Ancho de banda	180 KHz	1.4 Mhz
Velocidad Máxima de datos	< 100 Kbps	384 Kbps
Velocidad de bajada / subida	27.2 / 62.5 Kbps	Hasta 1 Mbps
Latencia	1.5 - 10 seg	50 - 100 ms
Duración de batería	* 10 años	* 10 años
Consumo de energía	Medio	Medio
Coste por módulo	5 - 10 dolares	10 - 15 dolares
Despliegue de frecuencia	Flexible	En banda LTE
Penetración en interiores	Excelente	Buena
Voz	No	Si, VoLTE

\* Según el uso

Tabla 1 - Comparación de tecnologías NB-IOT vs LTE-M [8]

### 2.3.2 LoRa y SigFox

LoRa es una tecnología inalámbrica de largo alcance principalmente patentada por Semtech, que opera en el espectro de frecuencias ISM.

Hoy en día es administrada por la “Alianza LoRa”, Cualquier fabricante de hardware puede construir módulos LoRa, pero tiene que obtener una certificación de cumplimiento de la alianza.

LoRaWAN es un protocolo abierto construido sobre LoRa. En la Ilustración 12 se puede apreciar la estructura de este protocolo.

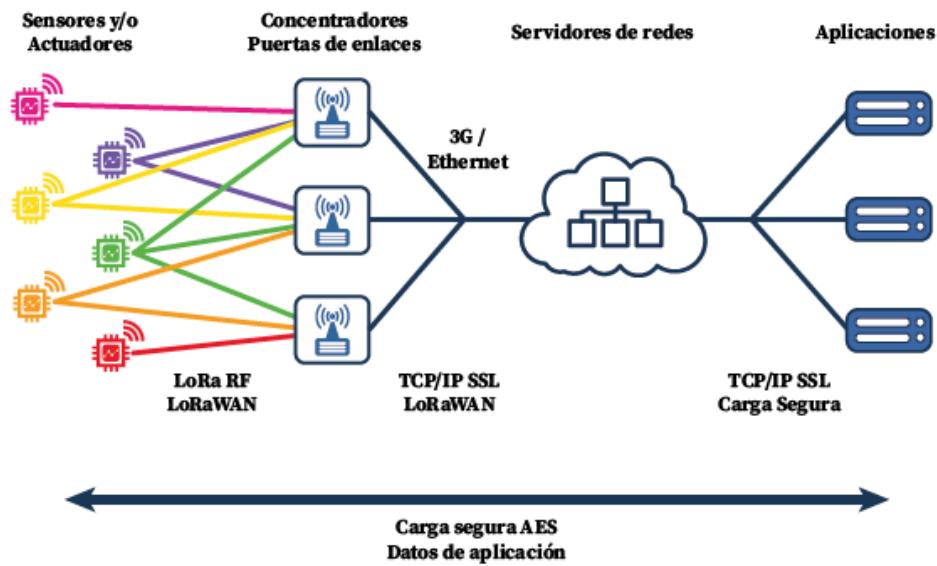


Ilustración 12 - Estructura LoRaWAN [9]

SigFox es una compañía francesa que se posiciona como un proveedor de red IoT y aspira a convertirse en el principal proveedor a nivel global.

En esta red los dispositivos envían sus datos a través de la red SigFox hacia una Backend Sigfox, y este maneja la transferencia de los mensajes haciendo una petición HTTP a un backend preconfigurado.

SigFox es una tecnología Ultra-Narrow Band que también funciona en el espectro de frecuencias ISM (en el caso de Chile es 915 MHz).

SigFox impone una serie de restricciones en los mensajes transferidos a través de la red, donde cada dispositivo puede enviar sólo 140 mensajes por día con un

límite de 7 mensajes cada hora. Cada mensaje puede tener hasta 12 bytes de longitud para carga útil (payload).

El protocolo de SigFox establece que los módulos radio pueden recibir sólo hasta 4 transmisiones entrantes por día. Esto, que viene siendo como una comunicación “downlink”, se puede efectuar inmediatamente después de un ciclo de transmisión de datos desde el módulo radio hacia el concentrador SigFox (comunicación uplink) y puede ser útil para enviarle cambios de configuración al módulo desde la red (por ejemplo, cambiar la periodicidad del envío de datos sensados).

SigFox gestiona completamente la comunicación entre el dispositivo IoT y el backend de la aplicación, lo que hace que la integración del módulo de radio sea un proceso bastante sencillo para los desarrolladores. Para interactuar con el módulo de radio se proporciona una sola API, no se requiere ninguna configuración.

En la Ilustración 13 se describe la estructura de la tecnología SigFox.

Para comenzar a integrar SigFox en nuestros proyectos, es necesario adquirir un módulo de radio compatible certificado y un plan de suscripción renovable anualmente.

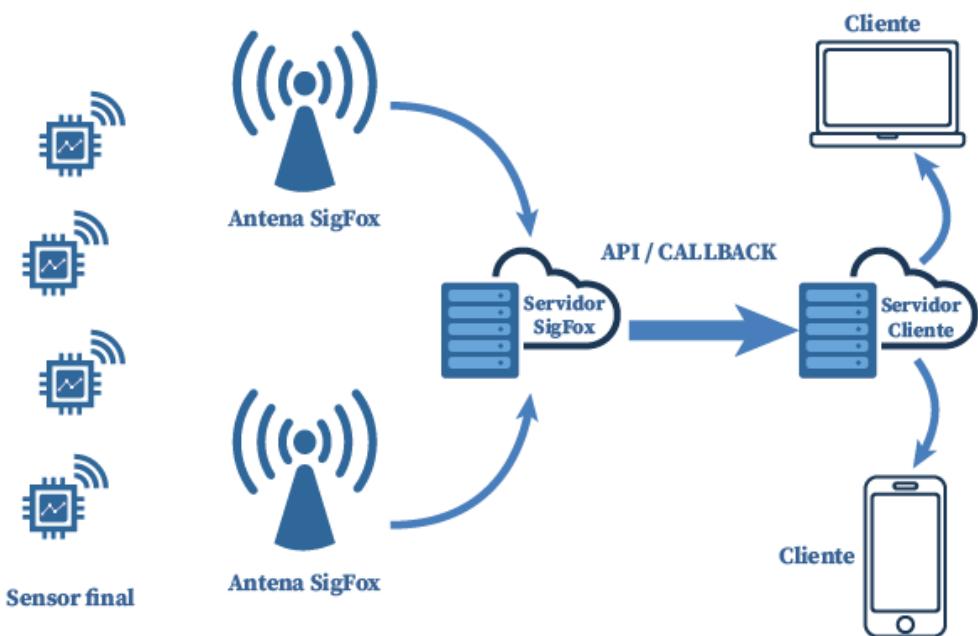


Ilustración 13 - Estructura SigFox

## 2.4 Software en la nube

Una vez que los datos recogidos por la red de sensores han sido subidos a la nube, estos son disponibilizados por SigFox hacia servidores donde existen aplicaciones software para efectuar análisis de datos, desplegar interfaces gráficas para la visualización de información basada en dichos datos y producir respuestas o acciones de control en la planta monitoreada, de acuerdo con la información recogida y las reglas de control que corresponda.

Por ejemplo, es posible que a partir de los datos básicos subidos a la nube se pueda detectar y clasificar patrones dinámicos de movimiento que siguen determinados objetos o personas y luego efectuar acciones o toma de decisiones.

A partir de esto, pueden surgir nuevos niveles de interacción entre softwares que permiten, a distintos niveles, recolectar, almacenar, organizar e integrar la

información con otras fuentes de datos y luego aplicar modelos de estimación o inferencia estadística de mayor nivel.

El verdadero valor agregado (y potencial de negocio) de cara al usuario final, viene de la mano con el software en la nube (Machine Learning, Inteligencia Artificial, etc.) porque permite el análisis de los datos obtenidos para generar información y conocimiento. Sumado a lo anterior, hay que destacar la gran relevancia que tendrá la interfaz con la que el usuario final interactuará para aprovechar la información y conocimiento generado a partir del sistema IoT.

#### **2.4.1 Plataformas IoT**

Existen plataformas para el IoT que son sistemas de software en la nube, poseen un completo conjunto de herramientas para conectar, procesar, almacenar y analizar datos tanto en el perímetro como en la nube. Entre ellas tenemos:

- ThingSpeak for IoT Projects
- Google Cloud IoT
- IBM Watson IoT
- Amazon AWS IoT Core
- Ubidots

#### **2.4.2 Plataforma IoT Ubidots**

Ubidots es una plataforma con 7 años de trayectoria, la cual permite de forma sencilla programar y configurar un dispositivo para que pueda enviar mediciones al servidor de Ubidots. Su dashboard permite agregar widgets fácilmente y con un diseño moderno.

Es compatible con protocolos HTTP, MQTT, TCP y UDP para recibir datos.

Existen dos modos de usar Ubidots:

- Ubidots Licenciado
- Ubidots STEM (versión educacional)

#### **2.4.2.1 Ubidots (Licenciado)**

Dependiendo de cuantos dispositivos se conectarán el precio mensual de la licencia para utilizar la plataforma varía desde los 49 a 1799 dólares.

Esta licencia permite tener las mismas opciones que Ubidots STEM más:

- Administración de aplicaciones
- Creador de aplicaciones IoT
- Gestión de dispositivos
- Gestión de usuarios
- Herramientas de análisis de Ubidots
- Crea tus propios puntos finales API
- Motor de eventos complejos

#### **2.4.2.2 Ubidots STEM**

Este modo es completamente gratuito con ciertas limitaciones. Es ideal para dar los primeros pasos con IoT. Dentro de las opciones que permite este modo tenemos:

- Servidor y recursos computacionales compartidos entre todos los usuarios de STEM. Velocidad y fiabilidad basadas en las solicitudes totales de la plataforma en cualquier momento.
- SOLO para uso no comercial (educación personal, investigación de IoT o proyectos de bricolaje).
- 1 Dashboard (solo estático).
- Hasta 3 dispositivos, cada dispositivo puede tener hasta 10 variables.
- 1 mes de retención de datos.

- 4.000 datos por día.
- Hasta 30 datos por minuto en todos sus dispositivos.
- Soporte de autoservicio con el Centro de Ayuda o la comunidad de Ubidots.
- 10 SMS gratis. Puede agregar saldo a su cuenta en cualquier momento para enviar más alertas, incluidos SMS internacionales y llamadas de voz.
- Máximo 10 widgets por tablero.
- API REST y soporte MQTT.
- SMS, llamadas de voz, correo electrónico, WebHook, Slack y Alertas Telegram.
- Cálculo de variables sintéticas (no se incluyen funciones especiales).

### 3. DISEÑO E IMPLEMENTACIÓN DEL PROTOTIPO

#### 3.1 Diseño del dispositivo IoT

En la Ilustración 14 se visualiza la estructura y conexión de cada módulo.

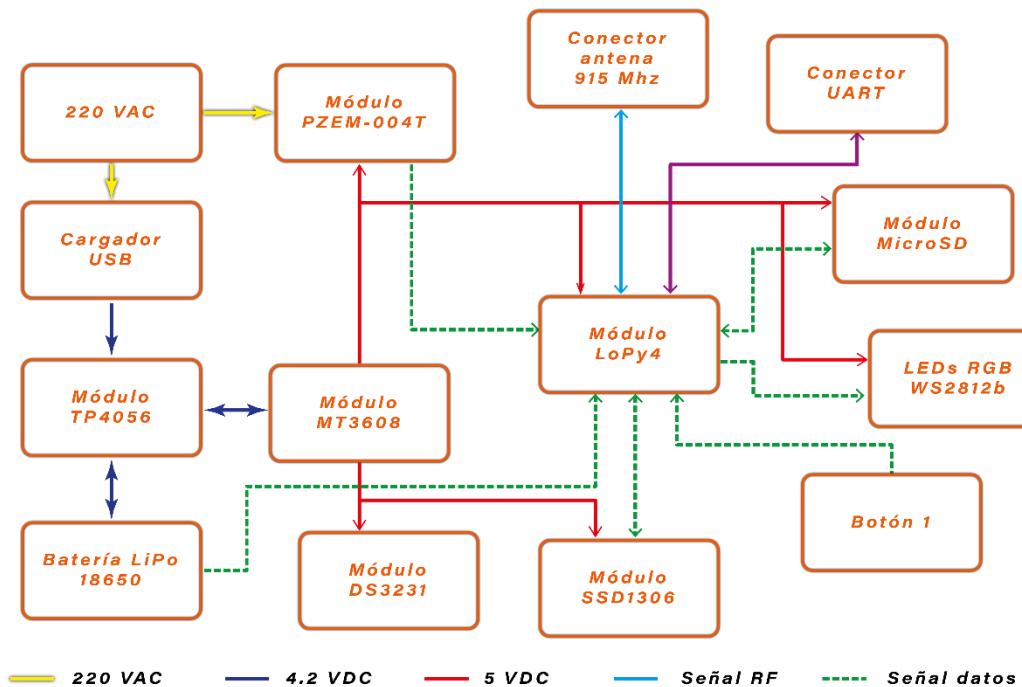


Ilustración 14 – Estructura del dispositivo IoT

El dispositivo es gobernado por el módulo LoPy4, a él se conectan los periféricos (Sensores y Actuadores) por comunicación I2C, SPI, Señales digitales y análogas y UART.

El módulo TP4056 se encarga de recibir 5V desde una fuente regulada como un cargador USB convencional.

Contiene una batería con un voltaje nominal de 4.2V que se mantiene cargada y es protegida por el módulo TP4056.

El módulo MT3608 se encarga de elevar de 4.2V a 5V.

El módulo PZEM-004T es el encargado de sensorizar la red eléctrica, se comunica con el microcontrolador con el protocolo UART.

El módulo MicroSD se encarga de almacenar los datos debugger en una memoria externa.

Los LEDs se encargan de informar al usuario a través de colores, el estado del sistema.

El botón 1 se utiliza para darle instrucciones al microcontrolador directamente del usuario.

El módulo SSD1306 se encarga de mostrar información relevante al usuario, como el voltaje actual, el estado de cada sistema, la hora y fecha entre otros.

El módulo DS3231 permite tener una marca de tiempo real para complementar el datalogger con hora y fecha actual.

El conector UART permite programar el microcontrolador desde un computador.

En la Ilustración 15 se muestra la disposición de cada módulo en la tarjeta reticulada.

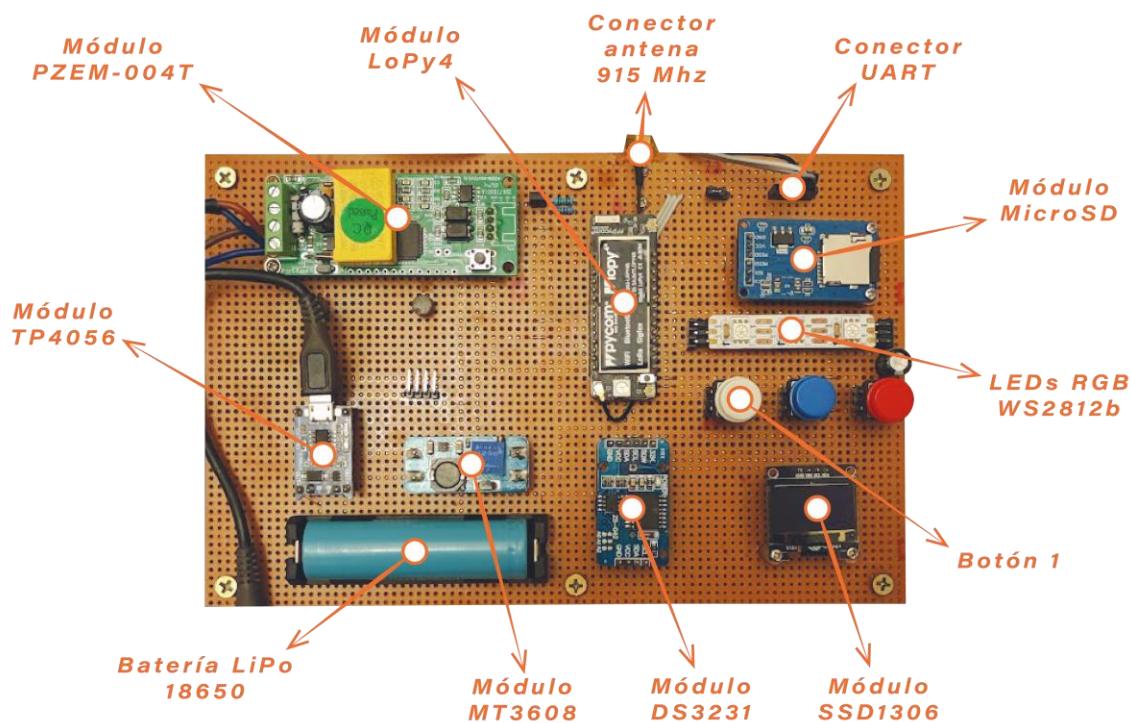


Ilustración 15 – Disposición de los módulos en placa reticulada

### 3.1.1 Medidor de energía PZEM-004T

La Ilustración 14 corresponde a un PZEM-004T, dispositivo capaz de medir la corriente y el voltaje de una red eléctrica monofásica. Tiene un puerto de comunicación TTL, lo que permite enviar las mediciones a cualquier dispositivo compatible como un microcontrolador para su posterior procesamiento. El puerto de comunicación está aislado por optoacopladores, lo que aumenta su seguridad en caso de corto circuitos o electrocuciones.

Para este prototipo solo usaremos la lectura de voltaje, ya que, al no existir carga en el sistema, prácticamente las lecturas de corrientes serían nulas.

Se tomó la decisión de utilizar este sensor por su sencillez de operación y su alta precisión a bajo costo.

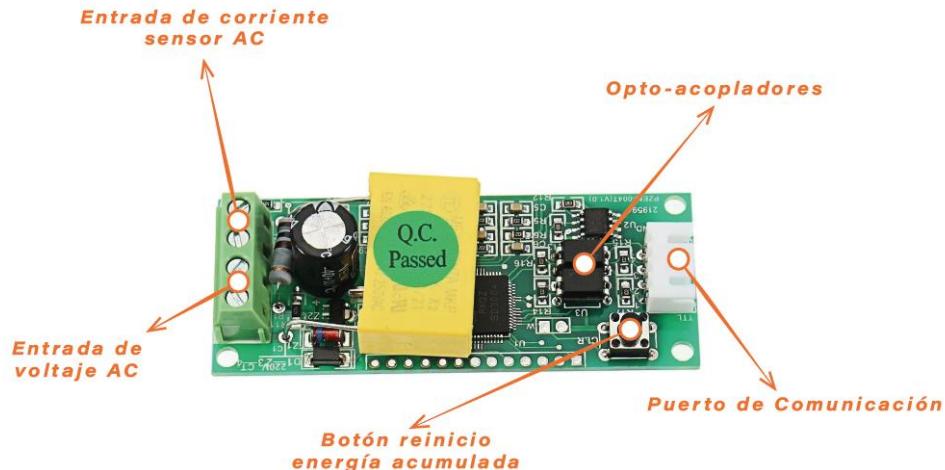


Ilustración 16 - Medidor de energía PZEM-004T

### 3.1.2 Cargador USB 220VAC/5VDC 2A

La Ilustración 15 corresponde a un simple cargador de celular con puerto USB, con este dispositivo se mantiene cargada la batería. Para simplificar las conexiones se desmontó de su caja plástica como se observa en la Ilustración 16 y 17, y que posteriormente se montó en una placa reticulada.



Ilustración 17 - cargador USB

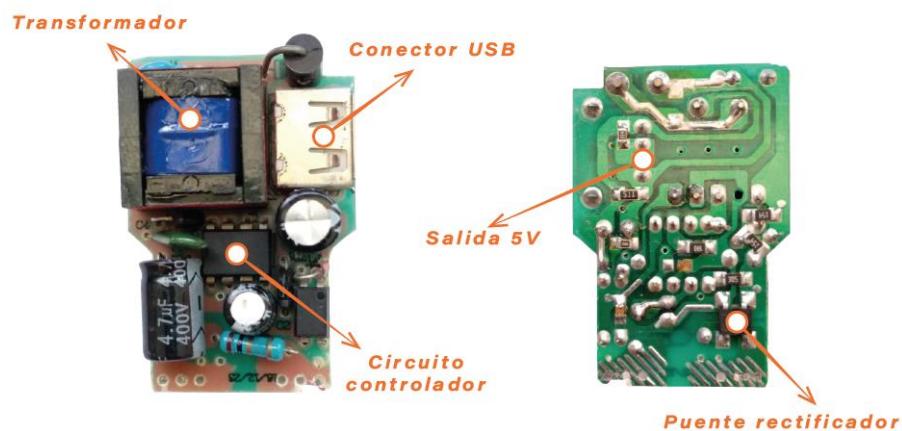


Ilustración 18 - PCB del cargador USB

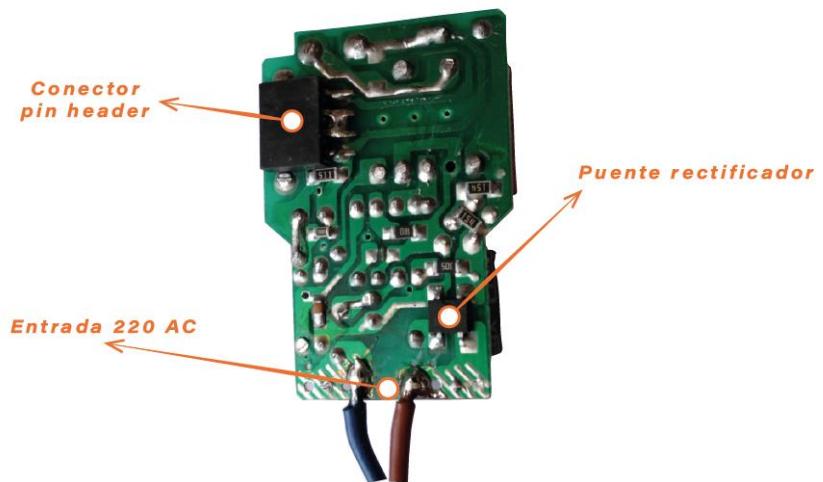


Ilustración 19 - PCB del cargador USB modificado

Para este prototipo se eligió un cargador sin descripción de fabricante (lo más probable chino) de bajo costo. Existen otros de mayor calidad en el mercado como lo representa la Ilustración 18.



Ilustración 20 - Fuente de alimentación

### 3.1.3 Módulo TP4056 + protección DW01A y FS8205A

La Ilustración 19 muestra un módulo que permite cargar una batería de Li-Po de una celda, monitoreando el voltaje para no sobrepasar los límites de este tipo de baterías. Funciona con una entrada de 5V y su salida es de 4.2V al momento de cargar la batería. Corta la energía si el voltaje llegó a su máximo para no calentar la batería y destruirla.

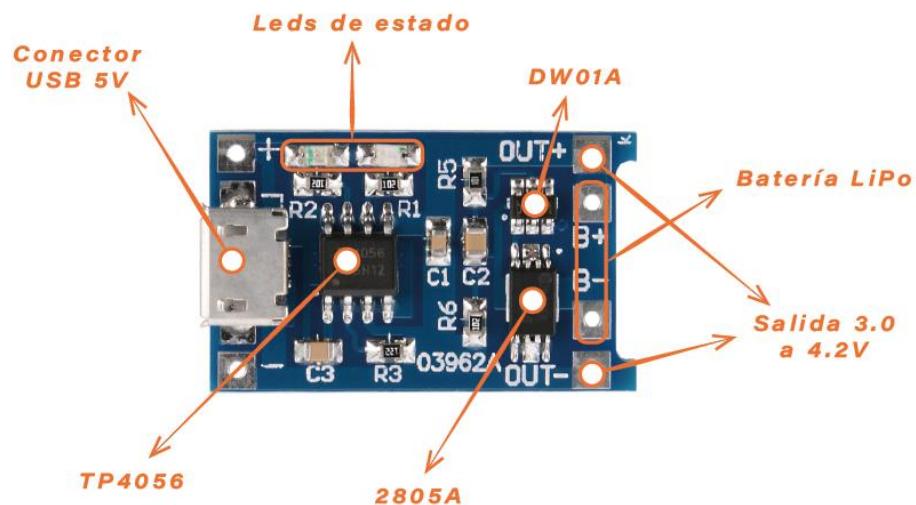


Ilustración 21 - Módulo TP4056

En la Ilustración 20 tenemos el circuito de protección DW01A para baterías Li-Po y en conjunto con el FS8205A que contiene dos MOSFET integrados, corta la salida de la batería si su voltaje disminuye por debajo de 3V, así protege la batería de descargas indeseadas que podrían destruirla.

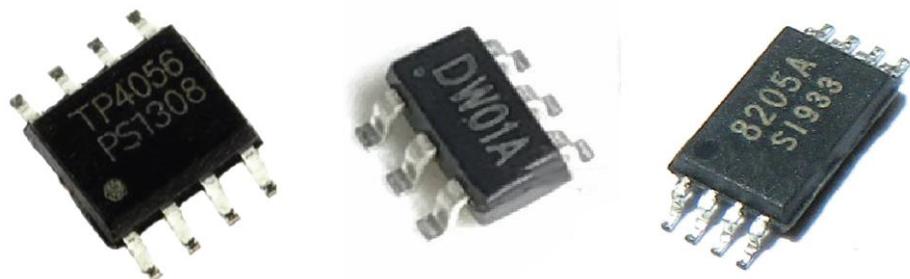


Ilustración 22 - Chip TP4056, DW01A y FS8205A

### 3.1.4 Módulo DC/DC Step UP MT3608

La Ilustración 21 muestra un módulo que convierte el voltaje DC a DC, permite aumentar el voltaje de una batería LiPo de una celda a un voltaje superior, en este caso a 5V.

Posee un potenciómetro que permite ajustar el voltaje de su salida a uno que se necesite. Se debe tener cuidado al conectarlo, ya que este potenciómetro no viene ajustado de fábrica y puede quemar algún componente si no se ajusta antes de integrarlo a su sistema.



Ilustración 23 - Módulo MT3608

### 3.1.5 Batería Li-Po 18650

La Ilustración 22 muestra una batería Li-Po 18650. Este tipo de batería se usan comúnmente en los notebooks, pero hoy en día el fácil acceso y bajo costo de estas, permiten usarlas en proyectos y prototipos electrónicos, sobre todo en el IoT.

La nomenclatura 18650 es debido a sus dimensiones, 18mm de diámetro y 650mm de largo. Existen algunas que varían su largo al incorporarles circuitos de protección en uno de sus extremos. No en todos los proyectos sirven aquellas que tienen ese circuito protector, ya que limita la corriente máxima de carga y descarga y no es suficiente para algunas aplicaciones.

Se pueden reciclar estas baterías de notebooks viejos y dados de baja, teniendo mucho cuidado de no perforarlas y no cortocircuitarlas por accidente, ya que podría provocar incendios.



Ilustración 24 - Batería LiPo 18650

### 3.1.6 Módulo LoPy4 de PyCom

La Ilustración 23 muestra un módulo o tarjeta de desarrollo que contiene un microcontrolador con comunicación inalámbrica embebida compatible con WiFi, Bluetooth, LoraWAN y SigFox. Este dispositivo se usará para que ejecute todas las instrucciones programadas de este proyecto. Aquí es donde se carga la programación o Firmware.

Tiene un procesador doble de 32 bit que permite cargarle un sistema de interprete basado en Python llamado MicroPython. Tiene una memoria de 4MB para su programación y posee puertos de comunicación más usados como UART, I2C, SPI entre otros. Su programación es a través del puerto UART Hardware, en el cual es necesario un convertidor USB/TTL para la conexión con un computador.

El fabricante entrega toda la información necesaria para su uso y proporciona un plugin Pymakr para los editores de código Atom o Visual Code Studio [10].

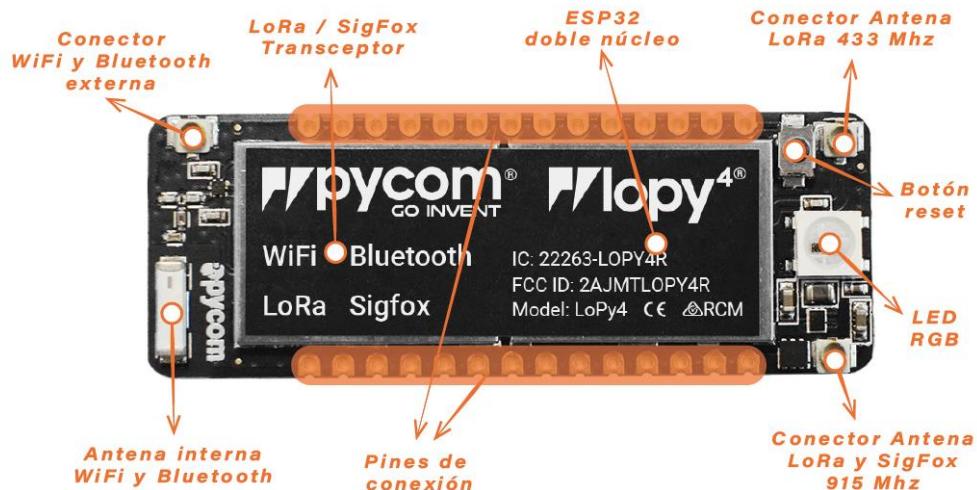


Ilustración 25 - Módulo LoPy4 de PyCom

En la Ilustración 24 se detalla la estructura interna del módulo.

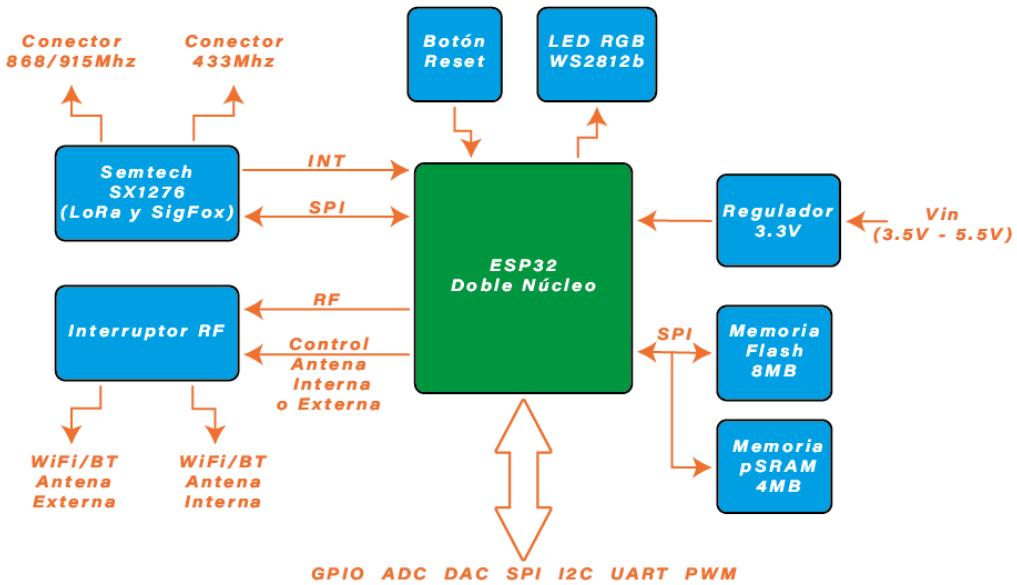


Ilustración 26 - Estructura interna del módulo LoPy4

### 3.1.7 Módulo DS3231

La Ilustración 25 muestra un módulo RTC (Real Time Clock) o reloj de tiempo real, usado para agregar marca de tiempo a cada evento que almacena la etapa debugger. Es muy preciso al tener compensación por temperatura, que permite obtener la fecha y hora actual, aunque no exista energía eléctrica en nuestro circuito. Esto es posible ya que posee una batería de respaldo CR2032 que mantiene al circuito funcionando. Su comunicación es por el protocolo I2C.



Ilustración 27 - Módulo RTC DS3231

### 3.1.8 OLED SSD1306

La Ilustración 26 muestra una pantalla OLED que permite dibujar gráficos y texto, se utiliza para mostrar información relevante al usuario como voltaje actual, hora y fecha, estado del sistema entre otra información. Tiene un tamaño es de 0.96 pulgadas y su resolución es de 128 x 64 pixeles, su comunicación es con el protocolo I2C y existen monocromáticas y RGB.

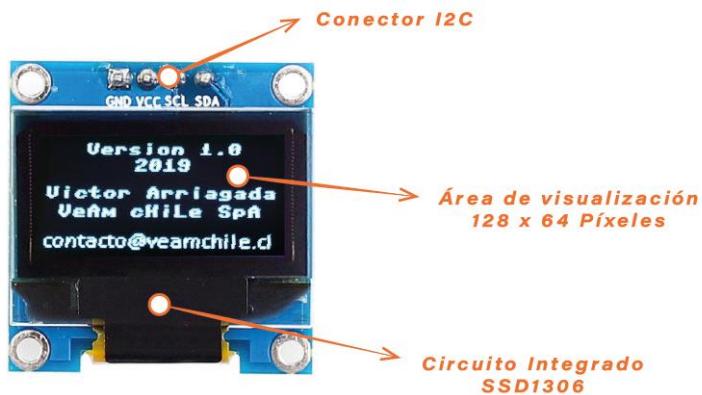


Ilustración 28 - Pantalla OLED SSD1306

### 3.1.9 Botón Pulsador

La Ilustración 27 describe un componente pasivo que sirve para enviar datos binarios a un microcontrolador, con este botón daremos instrucciones como por ejemplo cambiar la información mostrada en la pantalla OLED, o simplemente alternar el modo programación o funcionamiento normal del prototipo. Internamente contiene unas placas que se unen dejando pasar corriente al presionar el botón. Este tipo de pulsador es momentáneo, esto significa que, al soltar el botón, dejarán de estar en contactos las placas internas, cortando la corriente que circula a través de ella. Normalmente en electrónica digital, se usa en conjunto con una resistencia para evitar el estado flotante (Z). En este desarrollo la solución anti-rebote o anti-microcortes que se asoció a los pulsadores fue implementada a nivel de software.

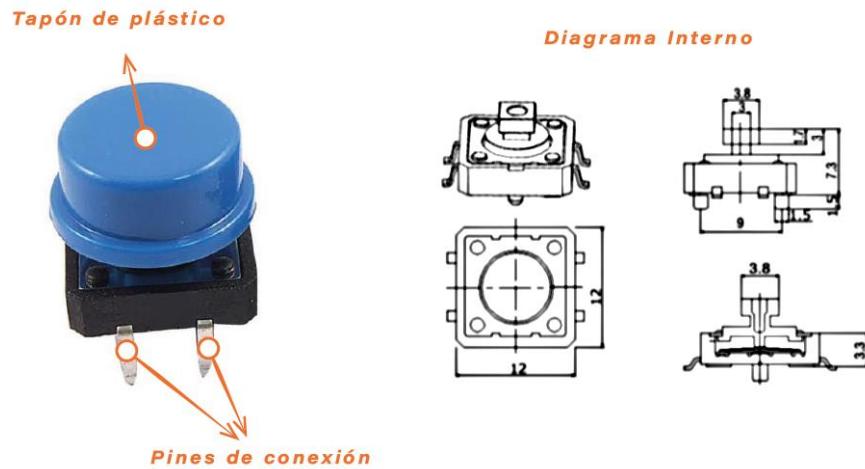


Ilustración 29 - Botón Pulsador

### 3.1.10 LED WS2812b

La Ilustración 28 muestra los Leds SMD RGB direccionables. Se usan para informar al usuario con colores diferentes el estado de la red eléctrica, estado del sistema y envío de datos de forma inalámbrica. A través de un solo pin se envían datos digitales con la información necesaria para encender o apagar los leds, cambiar colores, y brillo de estos. Al ser direccionables y estar conectados en serie, se pueden conectar varios leds a un solo pin de datos, esto quiere decir que se puede controlar cada led de forma única e independiente utilizando solo una salida del microcontrolador.

Cada led contiene un circuito digital que recibe los datos serial, los procesa y réplica al siguiente led y luego controla cada led interno (Rojo, Verde y Azul)

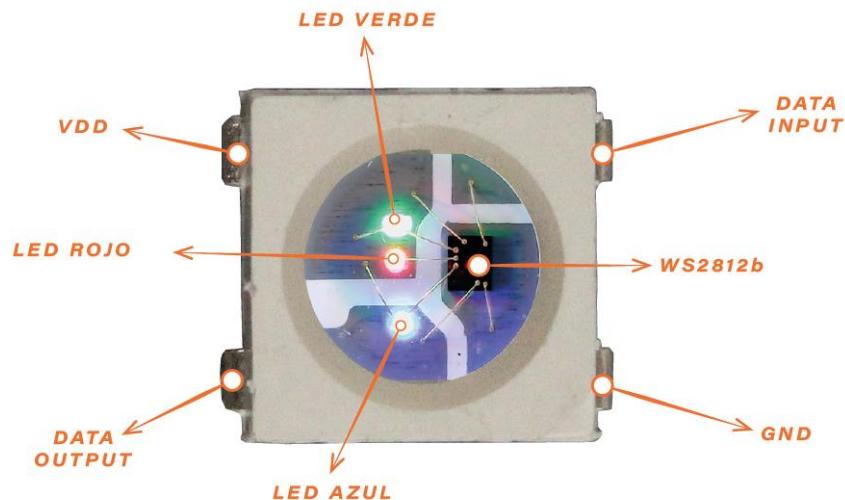


Ilustración 30 - LED RGB WS2812b

### 3.1.11 Módulo MicroSD

La Ilustración 29 describe un módulo que permite almacenar información en una memoria MicroSD, se usa como datalogger, almacenando cada evento que se presenta en el sistema. Su conexión es por el protocolo SPI y debe estar conectado a 3.3V igual que el estado Alto en los pines de comunicación del módulo LoPy4. Como se aprecia en la Ilustración 30, se debió realizar una modificación al módulo, eliminando el nivelador de tensión y conectando directamente las pistas según corresponda con cada pin de datos. Esta modificación se debe a que el microcontrolador utiliza niveles TTL 3.3V y el nivelador de tensión lo usa para compatibilizar niveles TTL de 5V.

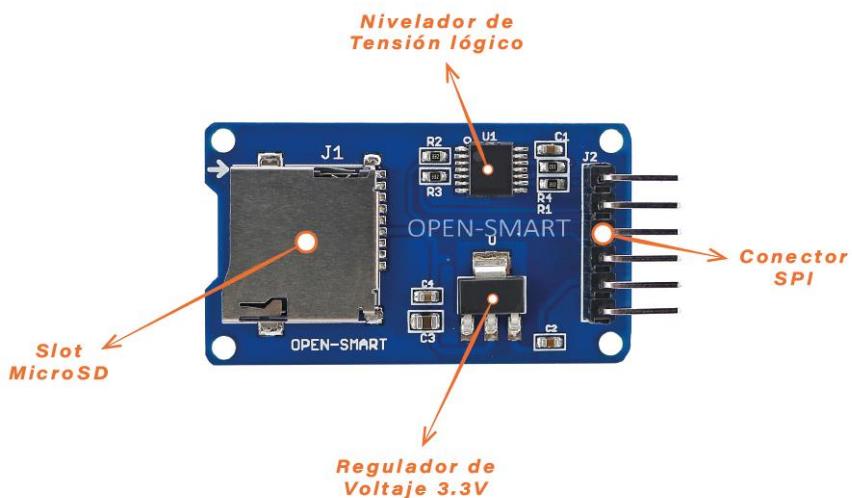


Ilustración 31 - Módulo MicroSD

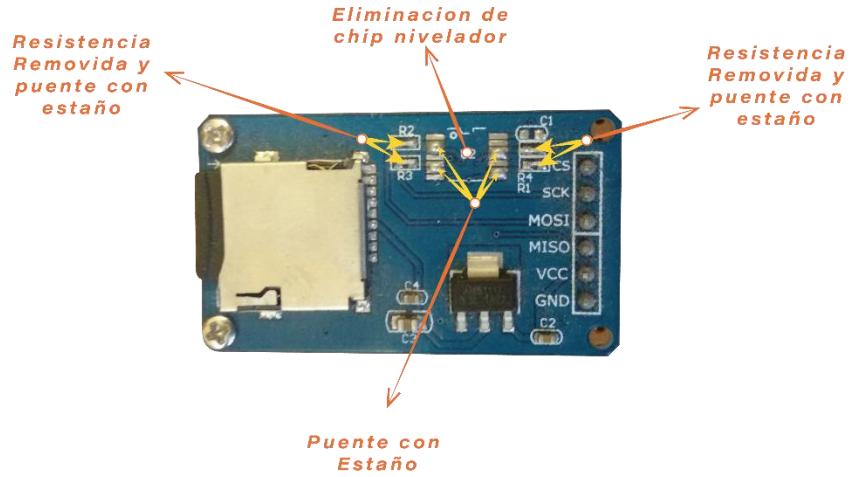


Ilustración 32 - Módulo MicroSD Modificado

### 3.1.12 Antena 915 MHz + conector

La Ilustración 31 describe una antena RF que es usada para que el módulo LoPy4 pueda comunicarse con las redes LoRaWAN o SigFox, se debe tener precaución de tener conectada la antena antes de utilizar la función inalámbrica del módulo, es posible dañar la etapa de salida RF si no se tiene precaucion.

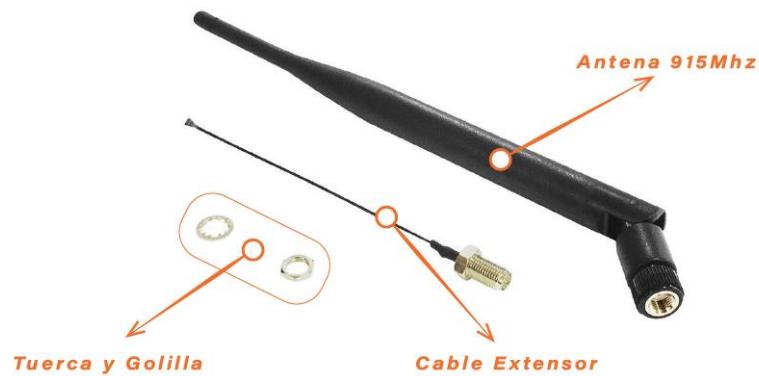


Ilustración 33 - Antena 915 MHz

### 3.1.13 Resistencia y divisor de tensión

Se utilizaron 2 resistencias para hacer un divisor de tensión que se utilizará en las lecturas de voltaje de la batería LiPo, estas lecturas se usaran para saber si la batería está cargada o descargada, alertando al usuario si sucede lo segundo. La entrada analógica del microcontrolador LoPy4 tolera un máximo de 1V. Este divisor realiza el trabajo de reducir el voltaje de entrada, adaptándolo en el rango requerido. En la Ilustración 32 se aprecia un jumper para desconectar el divisor de tensión y así poder calibrar la salida de voltaje de la batería.

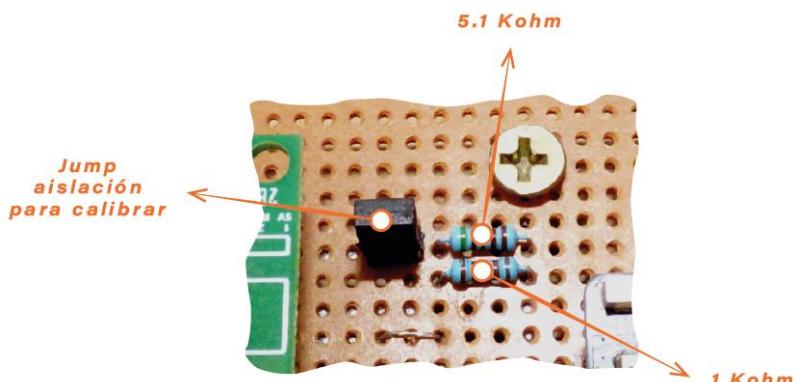


Ilustración 34 - Divisor de tensión

La calibración se realizó con el método de interpolación, como la batería se comporta de forma lineal entre 3 a 4.2 voltios, entonces podemos realizar un cálculo matemático lineal. Para esto se obtienen muestras discretas a diferentes voltajes de la batería, junto a esto calculamos con los datos RAW que nos entrega el conversor analógico a digital que contiene el microcontrolador y así podemos obtener una aproximación aceptable del voltaje de la batería. El error real del voltaje es de 1.20%, obtenido con un multímetro MY-64, pero como solo necesitamos cambios de estado (saber si la batería está cargada, media o descargada) el error es aceptable para esta aplicación.

### 3.1.14 Módulo USB/TTL CP2102

En la Ilustración 33 se muestra un módulo que permite la transferencia de información por el protocolo UART. Esto es necesario para controlar el microcontrolador y programar el Firmware básico MicroPython. Posteriormente se programan las instrucciones en lenguaje minimizado Python.

Este módulo es tolerante con niveles de 5V y 3.3V, posee una fuente de alimentación de 5V o 3.3V de baja corriente para sistemas que lo requieran.

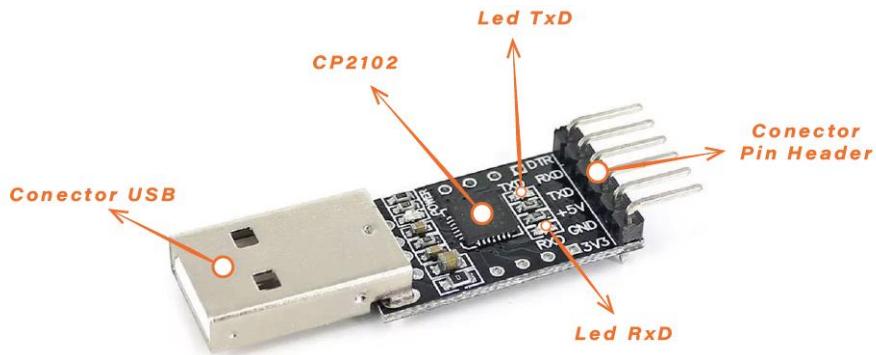


Ilustración 35 - Módulo USB/TTL CP2102

### 3.1.15 Implementación del dispositivo IoT

Todos estos componentes y módulos se montaron en una PCB reticulada puesta en una base de madera, la cual se aprecia en la Ilustración 34.

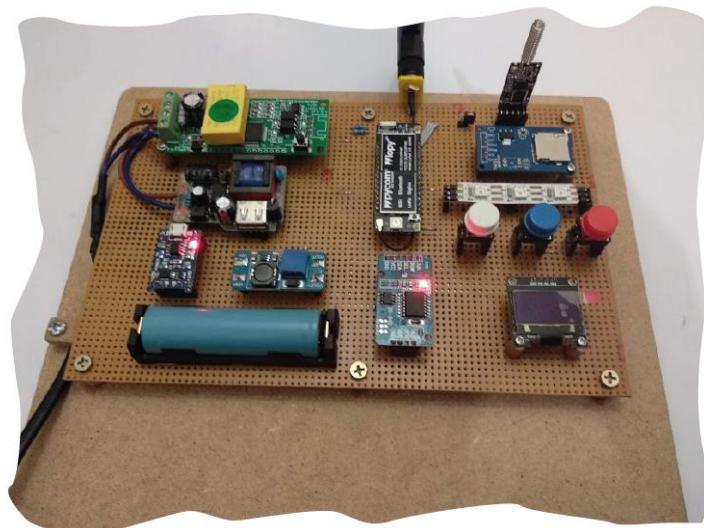
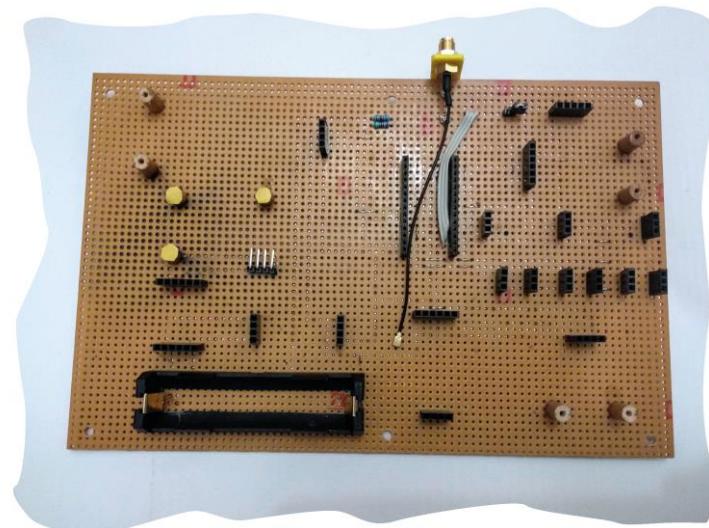


Ilustración 36 - Tarjeta del prototipo

### **3.1.16 Metodología de montaje**

Cada componente o módulo está sobre pines de conexión, facilitando la extracción de ellos. En la Ilustración 35 se observa la tarjeta sin ningún modulo.



*Ilustración 37 - Montaje del prototipo*

La metodología usada fue a raíz de posibles fallas de los módulos, casos en los cuales sería necesaria una rápida reparación o reemplazo de estos.

Gracias a esta metodología fue posible, en un corto plazo, reparar un cortocircuito que se provocó en el módulo “Cargador USB”. En la Ilustración 36 se detalla el componente electrónico que falló en el cargador.

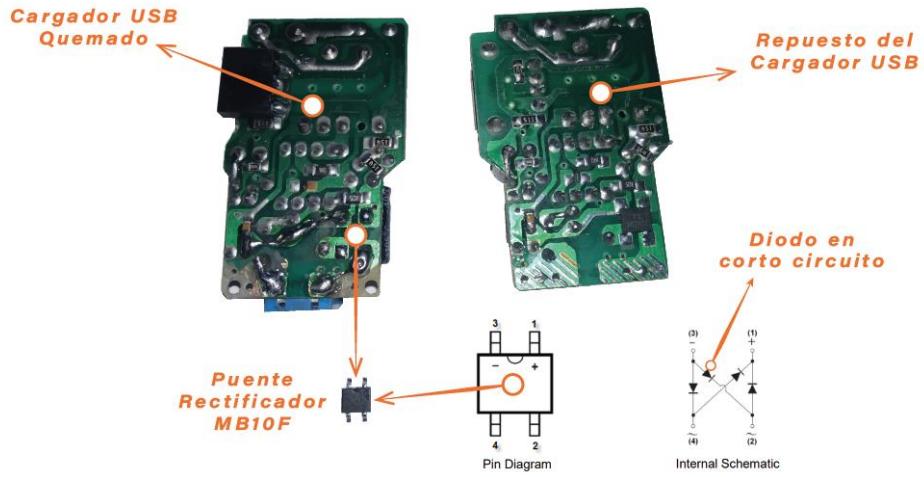


Ilustración 38 - Fuente de alimentación dañada

### 3.1.17 Base de batería LiPo

La batería está en una base de plástico, la cual se recicló de otro dispositivo. Esta base permite la extracción o reemplazo de otra batería sin ningún problema. En la Ilustración 37 se observa la forma de la base.



Ilustración 39 - Base para batería LiPo 18650

### 3.1.18 Jumpers de ajustes y aislación

En la Ilustración 38 se aprecia la instalación de un jumper para que el módulo LoPy4 se inicialice en modo BOOT. Esto permite cargarle en memoria el firmware de MicroPython de PyCom (Compilador Python para microcontroladores modificado para módulo LoPy4)

Se instaló otro jumper para aislar el voltaje de la batería y el divisor de tensión, para poder realizar la calibración correspondiente.

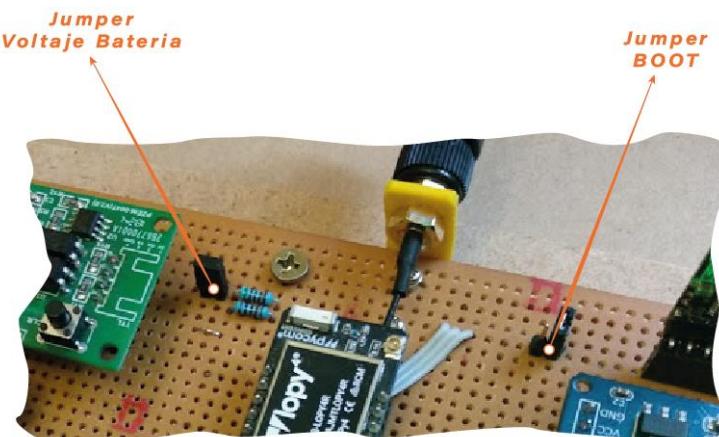


Ilustración 40 - Jumper del voltaje batería y jumper boot

### 3.1.19 Puerto de comunicación UART

En la Ilustración 39 se muestra la instalación de un puerto de comunicación con pines para la fácil conexión con el módulo USB/TTL. Esto permite programar el microcontrolador y cuando está en función o modo normal, envía datos de forma informativa como un debugger o depurador, el cual con cualquier software en el PC se puede almacenar en un archivo los eventos del microcontrolador.

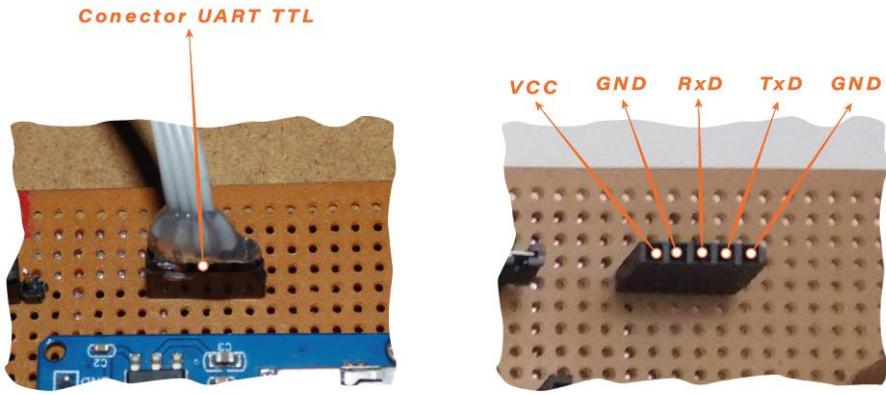


Ilustración 41 - Conector TTL

### 3.1.20 Leds RGB para visualización de estados

En la Ilustración 40 se aprecia la instalación de los leds RGB que se usan para mostrar el estado del voltaje de entrada, voltaje de batería, y estado de sistema, cambiando de color según el estado actual.

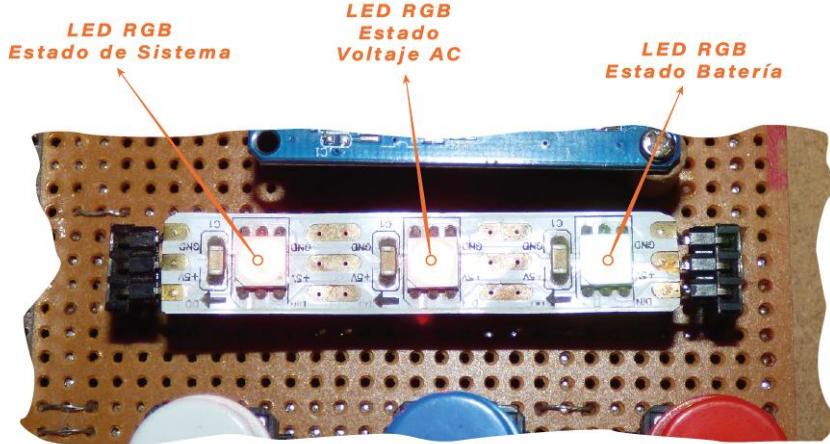


Ilustración 42 - Leds RGB

### 3.1.21 Botones pulsadores para control

En la Ilustración 41, los pulsadores se instalaron para futuras aplicaciones, pero para este proyecto solo se usará un botón, el que permite entrar en modo de programación (evita que se ejecute el programa principal), cambiar vistas de la pantalla OLED, y visualizar información del desarrollador del sistema.

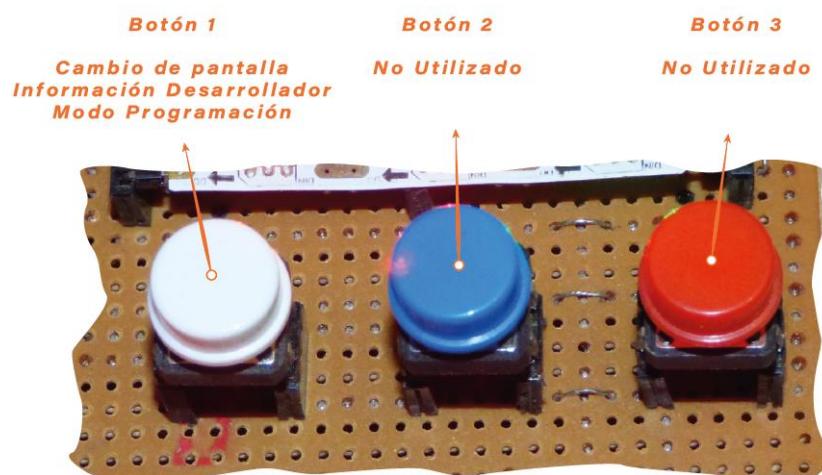


Ilustración 43 - Botones de control

### 3.1.22 Pantalla OLED para visualización de información

La Ilustración 42 muestra una pantalla OLED. Esta pantalla es usada para mostrar los valores de las variables, por ejemplo, el estado del sistema, voltaje actual de la red eléctrica y voltaje actual de la batería. También es posible mostrar información de contacto y nombre del proyecto en pantalla.

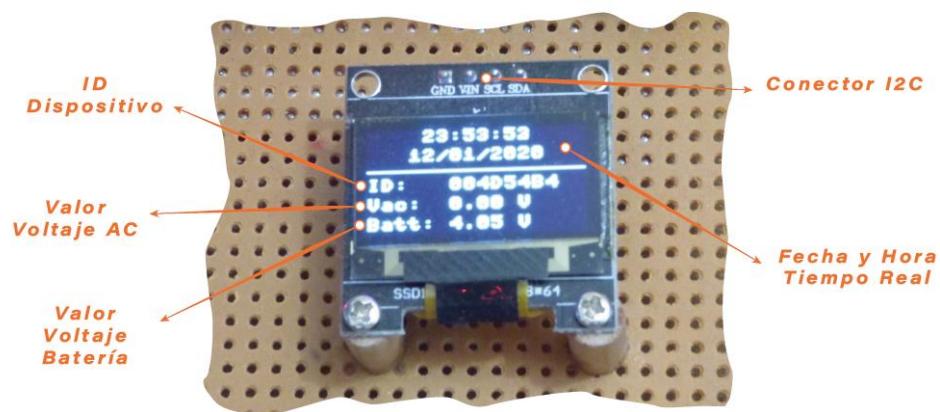


Ilustración 44 - Pantalla OLED

### 3.1.23 Reloj de tiempo real

La Ilustración 43 muestra un clock de tiempo real, que se usa para obtener la hora y fecha exacta en el sistema. Actualmente no se envía en el payload de SigFox la información de tiempo, solo se muestra en pantalla y en los archivos de la memoria MicroSD, ya que no alcanzaría dentro de las limitaciones de SigFox que son 12 bytes máximos en cada envío.

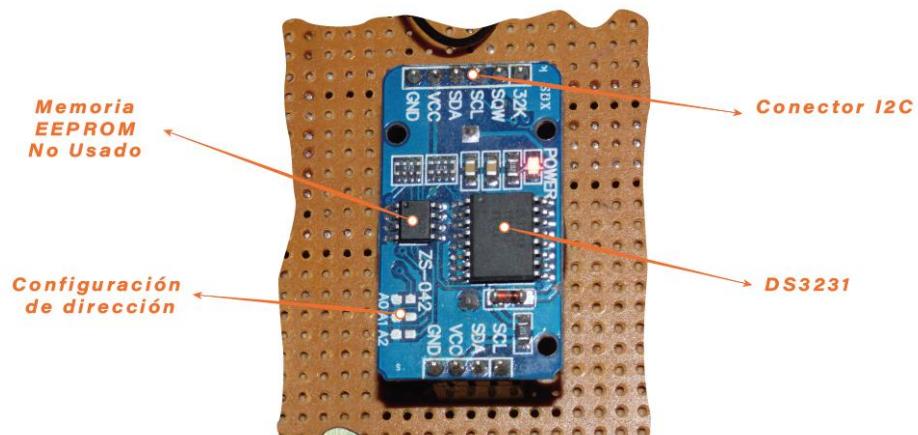


Ilustración 45 - Reloj de tiempo real

### 3.1.24 Memoria MicroSD y archivos de depuración

En la memoria MicroSD (Ilustración 44) se almacenan 3 tipos de datos en carpetas separadas. Como se aprecia en la Ilustración 45, las carpetas tienen los nombres eventos, métricas y payload; en el primero se almacena una marca de tiempo y los eventos importantes del sistema, como cambios de estados, alarmas, envíos de datos de forma inalámbrica, entre otros; el segundo almacena una marca de tiempo y las variables, para poder usarlos y tabularlos en Excel u otro programa y luego poder graficar los datos si fuese necesario; y el último guarda el payload en formato hexadecimal que envía al sistema SigFox, todo esto para realizar estudios y comparativas de los datos que debe enviar con los datos que llegan a la red SigFox.

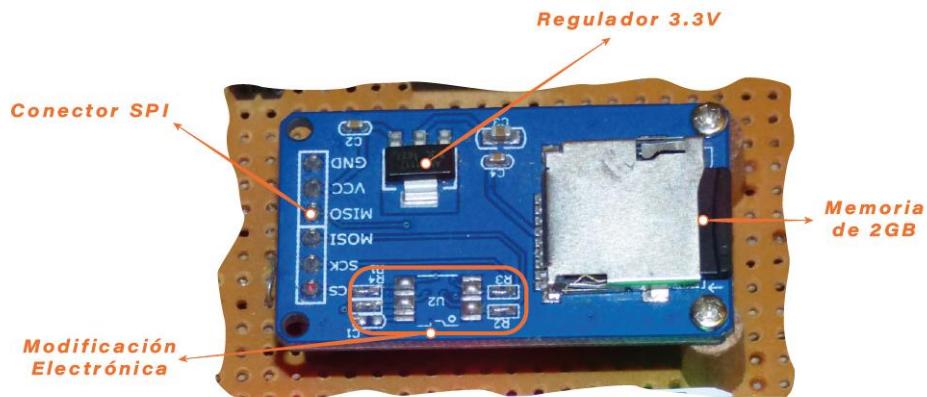


Ilustración 46 - Módulo microSD modificado

```

2019-12-31: Bloc de notas
Archivo Edición Formato Ver Ayuda
[00:06:38] [SigFox Send] [Payload HEX] 0000000002EB8240020200
[00:06:41] [SigFox] [Envio de metrica] [Voltaje_AC: 0.0 V] [V
[00:13:10] [Estado Volt AC] [OK] [Voltaje_AC: 222.6 V] [Volta
[00:13:10] [Estado Sistema] [OK] [Voltaje_AC: 222.6 V] [Volta
[00:1] Archivo Edición Formato Ver Ayuda
[00:1] [00:06:41] ac,0.0,bateria,4.086024
[00:1] [00:13:10] ac,222.6,bateria,4.105215
[00:1] [00:13:10] ac,222.6,bateria,4.105215
[00:3] [00:13:13] ac,222.6,bateria,4.106591
[00:1] [00:18:44] ac,223.2,bateria,4.098506
[00:30:47] ac,224.8,bateria,4.094787
[00:42:49] ac,221.3,bateria,4.086618

```

Ilustración 47 - Archivos DEBUG de la microSD

### 3.1.25 Fusible de protección

Después de analizar el cortocircuito provocado por el cargador USB, se determinó colocar un fusible de 500mA en la entrada de los 220AC, con el fin de evitar que se provoque un cortocircuito en la red en que estará conectado el sistema.

En los cargadores USB existe un circuito integrado que está conectado directamente a los 220VAC y dentro contiene un puente rectificador con 4 diodos rectificadores. Sin saber exactamente el motivo de la falla, uno de esos diodos se quemó y quedó en corto circuito, provocando un corte de energía involuntario provocado por el sistema prototipo. Al cambiar el cargador y comprobar que no contiene ningún tipo de protección, se incorporó un fusible en su entrada para evitar fallas involuntarias. El fusible se instaló directamente en el cable a modo de prototipo. En la Ilustración 46 podemos ver el fusible cubierto con termo retráctil.

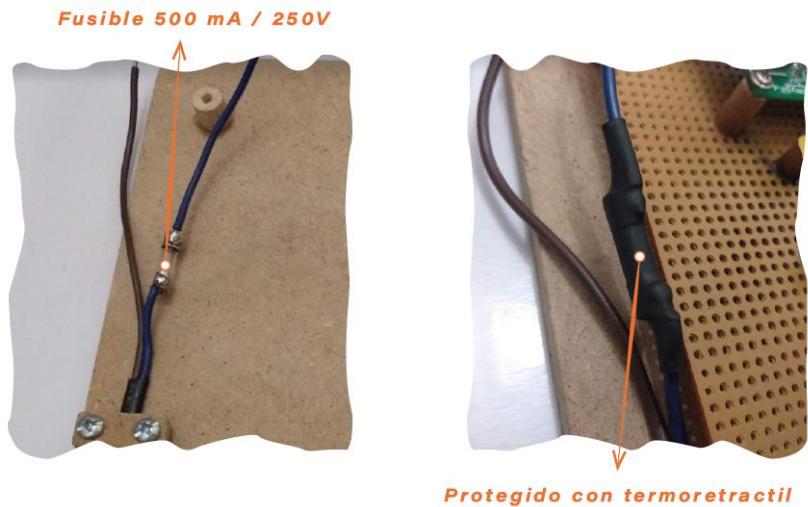


Ilustración 48 - Fusible de protección

## 3.2 Diseño y programación del firmware para el dispositivo IoT

### 3.2.1 Preparación

#### 3.2.1.1 Descarga e instalación de PyCom Firmware Update

Este software se necesita para poder cargar el Firmware MicroPython de PyCom al microcontrolador. Al finalizar este paso, estaremos en condiciones de poder usar el intérprete Python en nuestro microcontrolador.

La Ilustración 47 muestra la pantalla principal del software.

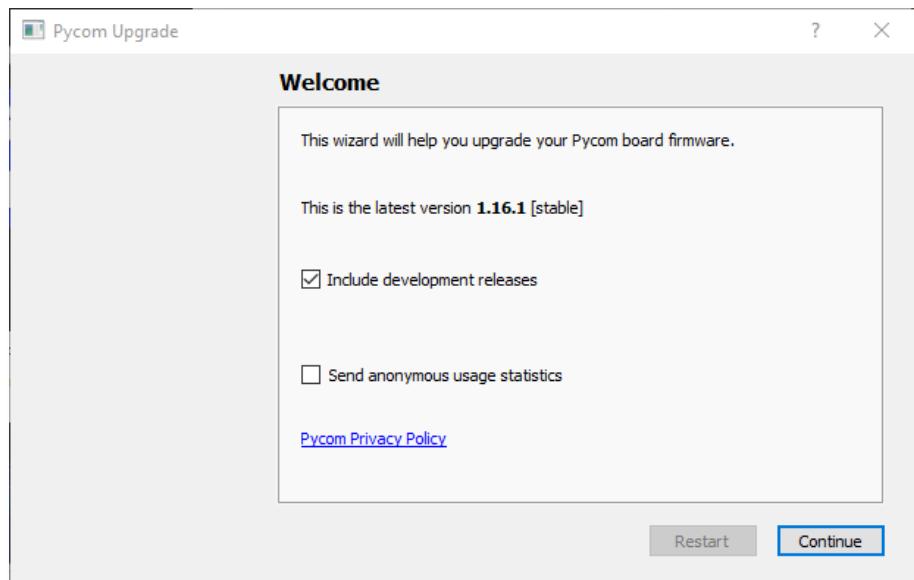


Ilustración 49 - Pantalla de bienvenida Pycom

Para descargar se ingresa al siguiente enlace:

<https://docs.pycom.io/gettingstarted/installation/firmwaretool/>

y se selecciona el sistema operativo que usan. En nuestro caso es Windows. La Ilustración 48 muestra el sitio web.

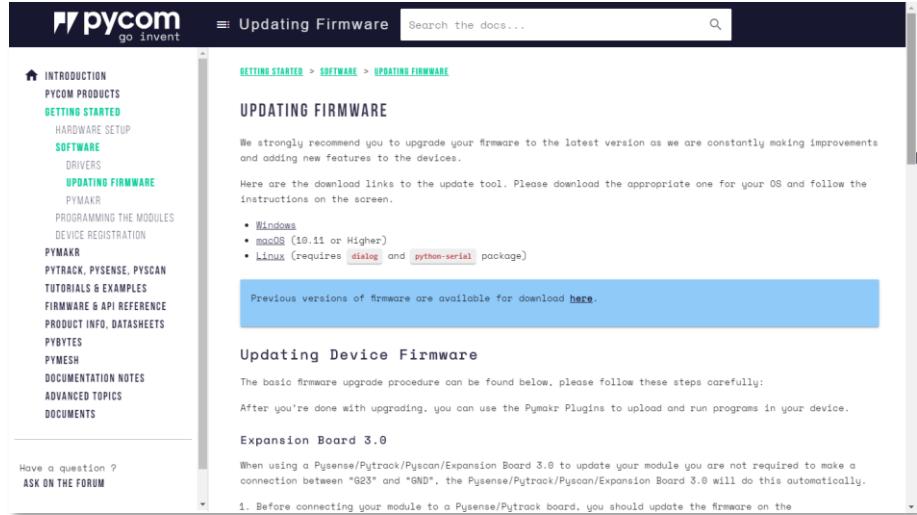


Ilustración 50 - Pantalla del sitio web para descarga de software

Una vez descargado procederemos a instalar el software. La Ilustración 49 muestra la pantalla de instalación.

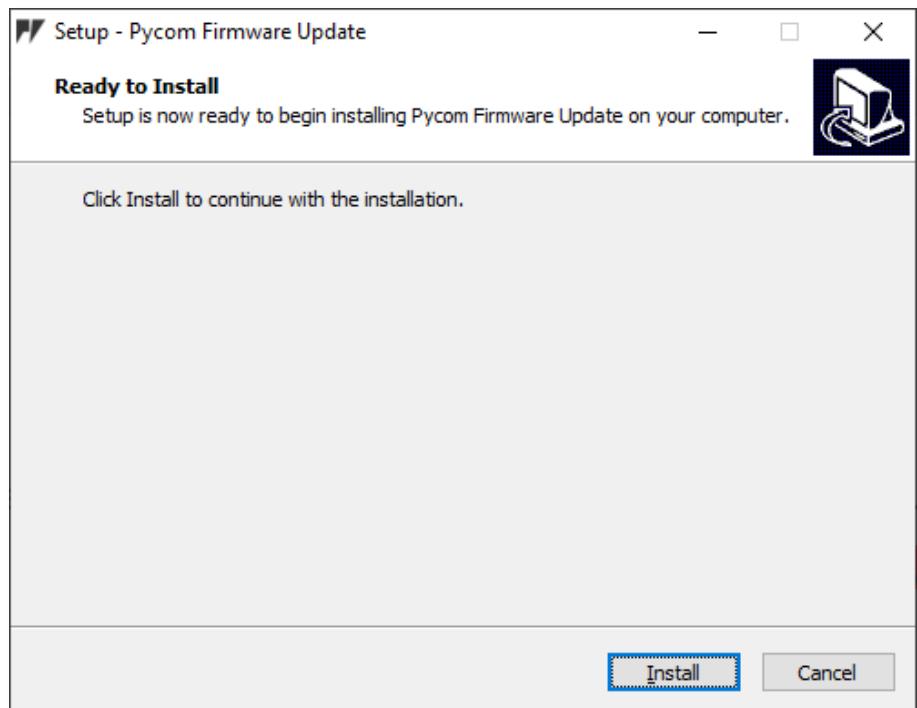


Ilustración 51 - Pantalla de instalación de software

### 3.2.1.2 Descarga e instalación de controlador CP2102

Para que nuestro computador reconozca el dispositivo USB/TTL que usaremos para la comunicación del hardware IoT, se necesita instalar estos controladores del siguiente enlace:

<https://www.silabs.com/products/development-tools/software/usb-to-uart-bridge-vcp-drivers>

Se debe seleccionar el sistema operativo y descargar como se aprecia en la Ilustración 50.

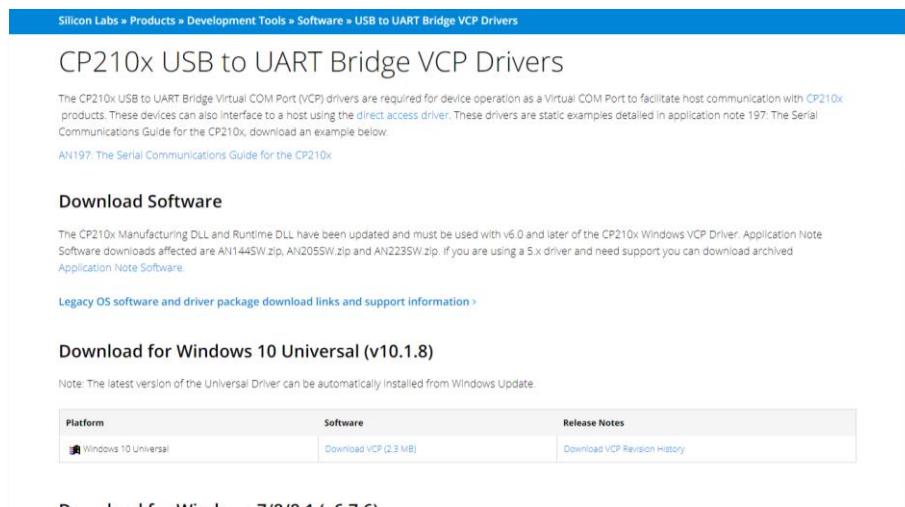


Ilustración 52 - Pantalla de descarga de drivers LoPy

Con algún software de compresión descomprimir el archivo descargado a una carpeta e instalar los drivers con el archivo de instalación “CP210xVCPIInstaller\_x86.exe” para computadores de 32bit o “CP210xVCPIInstaller\_x64.exe” para computadores de 64bit.

La Ilustración 51 muestra el asistente de instalación de los drivers.

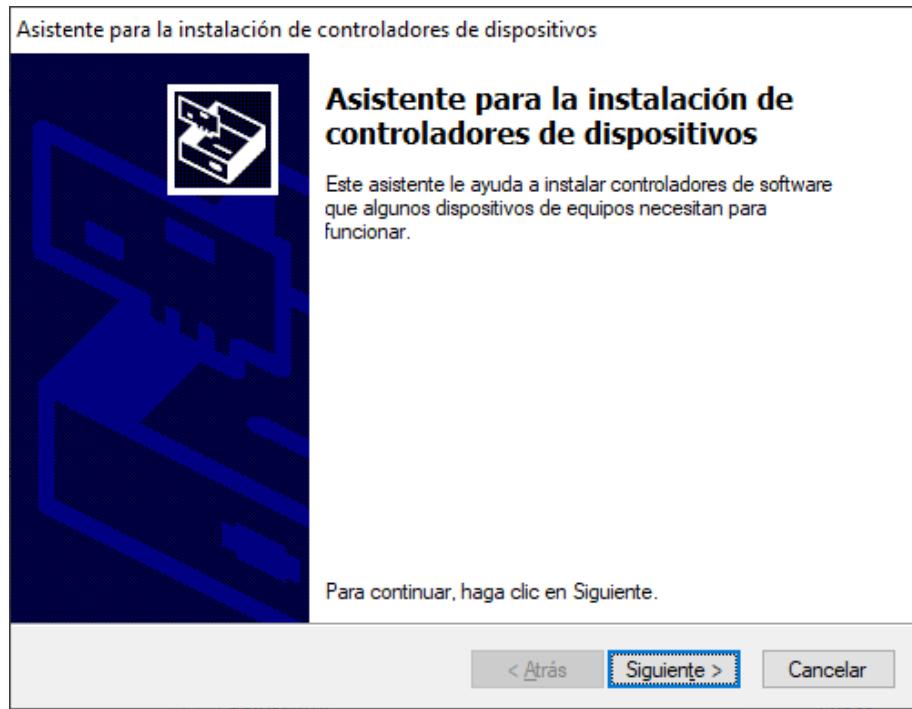


Ilustración 53 - Pantalla de instalación de drivers

Una vez finalizada la instalación, ya podremos colocar el dispositivo USB/TTL para que sea reconocido.

### 3.2.1.3 Descarga e instalación de editor de código Atom

Para crear nuestros programas necesitamos un editor de código. En este caso utilizaremos Atom. La Ilustración 52 muestra la página principal de descarga el cual responde al enlace: <https://atom.io/>

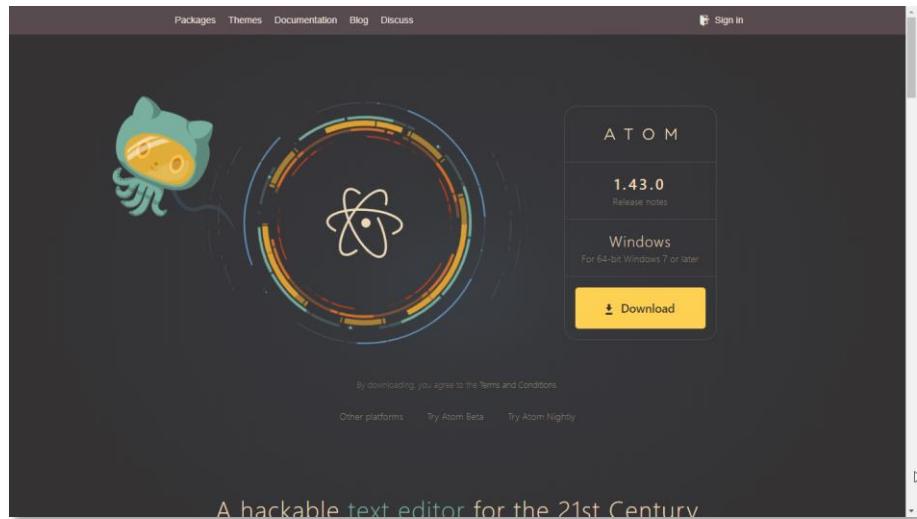


Ilustración 54 - Pantalla de sitio web de descarga entorno de programación

Posteriormente se debe presionar el botón Download.

Cuando ya esté descargado, procedemos a instalar el software, para lo cual aparecerá la pantalla de instalación de Atom como se aprecia en la Ilustración 53.

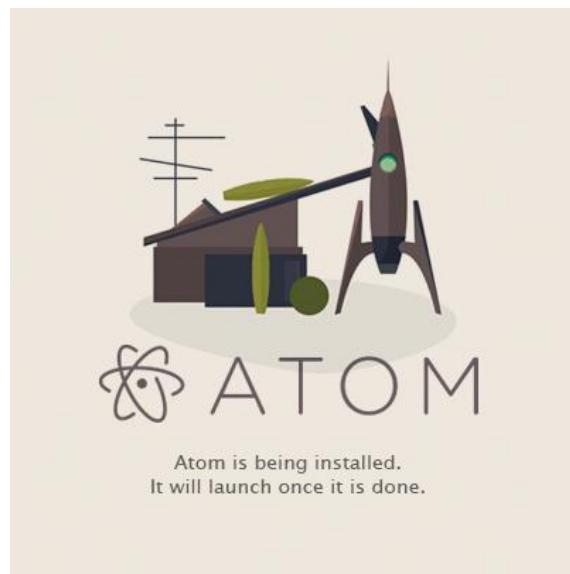


Ilustración 55 - Pantalla de instalación de ATOM

### 3.2.1.4 Descarga e instalación del Plugin Pymakr para Atom

Para poder programar en MicroPython descargamos el plugin Pymakr desde el gestor de instalación de Atom (Ilustración 54). Para esto iremos a “Settings” del menú “File” y seleccionamos la pestaña lateral izquierda “+ Install” en la barra de búsqueda, escribimos “Pymakr” y presionamos “Install”.

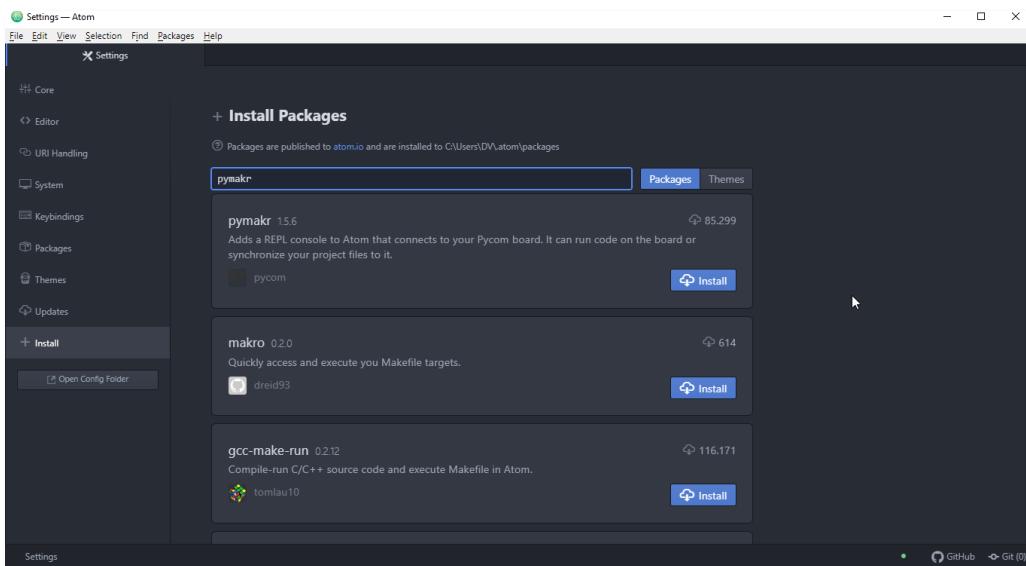


Ilustración 56 - Pantalla de instalación plugin "Pymakr"

### 3.2.1.5 Cargar MicroPython PyCom en microcontrolador

Para cargar el firmware de MicroPython al microcontrolador debemos conectar el dispositivo USB/TTL al módulo LoPy4 como se muestra en la Ilustración 55.

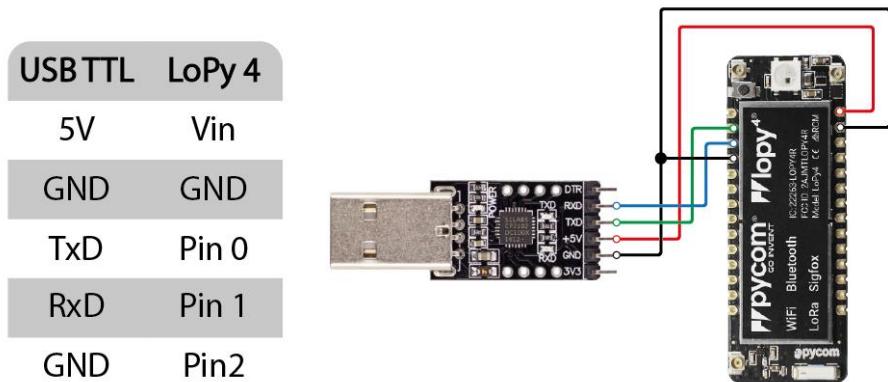


Ilustración 57 - Conexión del módulo LoPy4 vía USB

Posteriormente conectamos el USB/TTL a nuestro computador y si todos los drivers están correctos instalará el módulo y le asignará un puerto COM.

Ejecutamos el software PyCom Firmware Update

- Presionamos “Continue”
- “Continue” nuevamente
- Elegimos el puerto COM asignado a nuestro módulo
- Presionamos Continue
- En “Type” elegimos “pybytes-legacy” y la última versión, en este caso la “1.18.3”
- Activamos la casilla “Erase during update”
- Seleccionamos la región “AU915” y luego “Continue”

Y esperamos a que cargue el firmware.

Finalmente nos mostrará un resumen de los identificadores del dispositivo que necesitaremos más adelante para utilizar la red Sigfox.

Presionamos Done y nuestro dispositivo está listo para el primer programa.

### 3.2.1.6 Primer programa de prueba (Hola Mundo)

- Abrimos Atom.
- En la zona inferior presionamos “Connect”.
- Y en la línea de comandos escribimos “help()” y presionamos enter.

Si aparece el texto “Welcome to MicroPython!” seguido de información, quiere decir que está todo bien y podemos crear nuestro primer programa.

Seleccionaremos “Add Project folder” del menú “File” y crearemos nuestra carpeta “TEST” y presionamos “seleccionar carpeta”.

En el lado lateral izquierdo aparecerá una ventana con el nombre de la carpeta y sin nada dentro de ella. Ahora presionaremos el botón “Download” de la zona inferior y aparecerá una ventana, presionamos “Only new files” y comenzará a descargar los archivos que se encuentran en el dispositivo.

Ahora aparecerán dos archivos en la lista izquierda “boot.py” y “main.py”. Si los presionamos podemos apreciar que ambos archivos están vacíos, no tienen ninguna instrucción.

Para el primer programa, escribiremos en el archivo main.py el siguiente código:

```
1  from time import sleep
2  contador = 0
3  print("Inicio de mi primer programa...")
4  while True:
5      sleep(1)
6      print("Mi primer programa en MicroPython - Contador =",contador)
7      contador += 1
8      if contador == 5:
9          break
10     print("Fin de mi primer programa...")
11
```

Código 1 - <Hola Mundo>

Para probar este código se presiona el botón “run”. Esta instrucción ejecuta el código actual vía línea de comandos, lo cual quiere decir que no guarda el código en el dispositivo. Si se hace la prueba de reiniciar el módulo con el botón reset, se podrá apreciar que no se ejecuta el programa recién hecho.

### **3.2.1.7 Cargar programa a microcontrolador**

Para cargar los archivos lo primero que debemos hacer es guardar los cambios, para eso presionamos “Save” del menú “File” y posteriormente presionamos el botón “upload”.

Al terminar la carga el módulo se reiniciará y comenzará a ejecutar el programa. Si se reinicia con el botón reset, se puede apreciar que cada vez se ejecuta el programa cargado.

### **3.2.1.8 Líneas de comandos MicroPython**

Las líneas de comando son muy útiles para realizar programación sencilla. Es similar a la línea de comando del intérprete Python, pero las funciones cambian; por ejemplo, para encender el led RGB (a color rojo, después verde y finalmente azul) que está en el módulo LoPy4 debemos ingresar cada una de estas instrucciones en la línea de comandos seguido de Enter:

```
import pycom [Enter ↵]

pycom.heartbeat(False) [Enter ↵]

pycom.rgbled(0xFF0000) [Enter ↵]

pycom.rgbled(0x00FF00) [Enter ↵]

pycom.rgbled(0x0000FF) [Enter ↵]

Para apagar el led solo escribimos:

pycom.rgbled(0x000000) [Enter ↵]
```

Código 2 - Consola

### 3.2.2 Diagrama del programa

El diagrama que explica de forma simplificada las funciones que debe realizar el sistema se detalla a continuación.

#### 3.2.2.1 boot.py

Este archivo contiene las primeras instrucciones que se ejecutan al encender el microcontrolador. Es este archivo cargamos las librerías en memoria y ejecutamos comandos como iniciación de leds, desactivación del módulo wifi (no utilizado en este proyecto) y los pines IO para leer el estado del botón 1 (pulsador switch 1). Si al encender el sistema el botón 1 está presionado, se salta la ejecución del programa principal para poder utilizar el módulo en modo desarrollador (si existe un programa ejecutándose de forma cíclica, el entorno de creación de código no se puede comunicar vía serial para actualizar archivos).

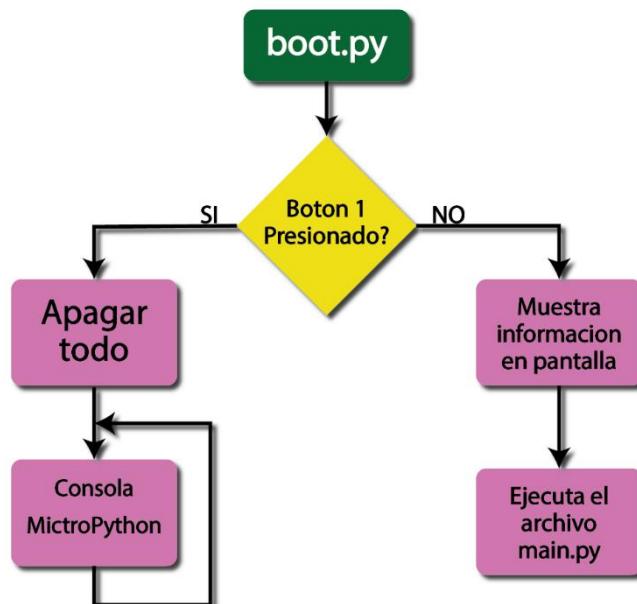


Diagrama 1 - Boot.py

### 3.2.2.2 main.py

Este archivo contiene las instrucciones más importantes del sistema, ejecuta subrutinas cada cierto tiempo de forma independiente, las cuales actualizan variables globales y realizan acciones de acuerdo con los cambios de dichas variables.

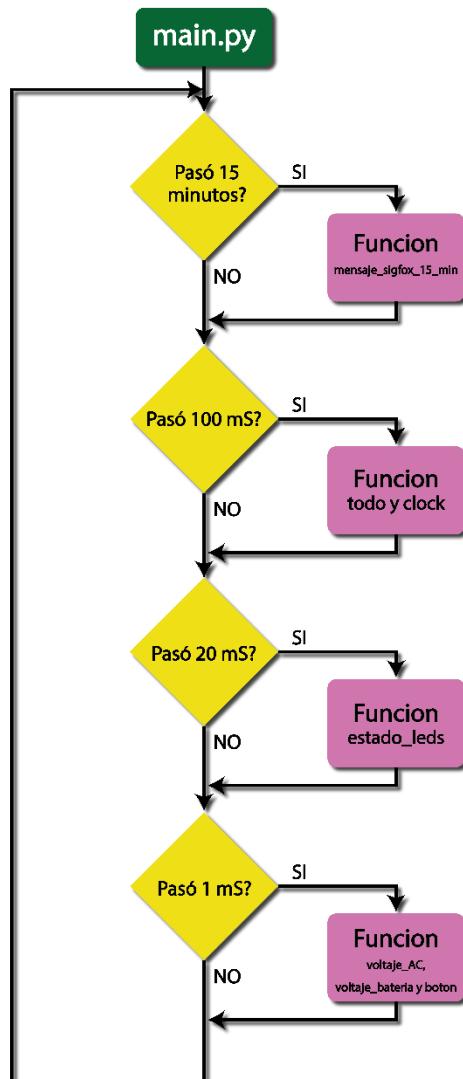


Diagrama 2 - Main.py

### **3.2.2.3 Subrutina mensaje\_sigfox\_15\_min**

Es una subrutina que evalúa si han pasado 15 minutos desde el último mensaje enviado, de ser así, nuevamente envía datos a SigFox con información actualizada de los estados (sistema, batería y voltaje AC), también envía la información a la memoria microSD y al puerto serial. Dentro del código tiene una rutina que se llama contador global, si este contador es modificado por otra subrutina, el conteo de los 15 minutos empieza de 0. Esto ayuda a no enviar datos con un período inferior a 15 minutos, pues al hacerlo estaremos violando los protocolos de SigFox.

Enviar datos a la microSD y al puerto serial es optativo. En este proyecto se utiliza para realizar un debug al código y así poder determinar fallas y solucionarlas con mayor facilidad. También podemos comparar los envíos realizados a SigFox desde el módulo con la información recibida en los servidores de SigFox y con esta información podemos determinar la estabilidad de la plataforma y las razones de fallas. Recordemos que no podemos tener retroalimentación de parte de los servidores SigFox, lo que provoca que el sistema esté completamente ciego al enviar datos a SigFox.

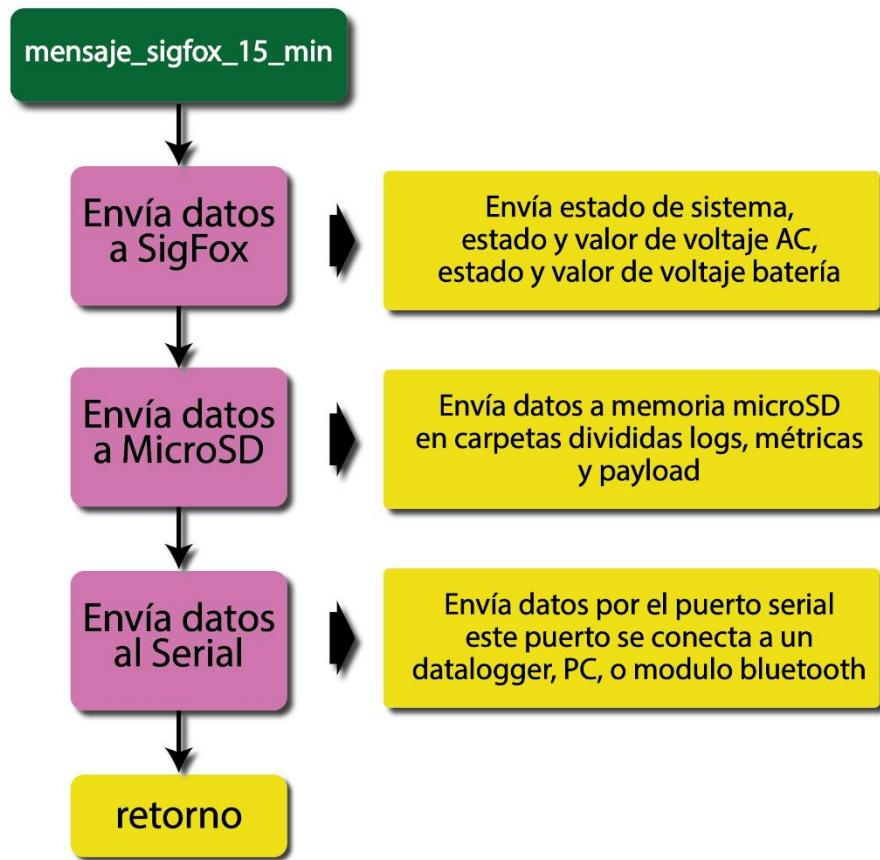


Diagrama 3 - Subrutina: *mensaje\_sigfox\_15\_min*

### 3.2.2.4 Subrutina todo

Esta subrutina evalúa las variables globales:

- **vol\_ac\_global**: Variable que almacena el voltaje actual de la energía alterna que ingresa por el sensor de energía PZEM-004T.
- **volt\_bat\_global**: Variable que almacena el voltaje de la batería que ingresa al conversor análogo digital del microcontrolador, pasando por el divisor de tensión anteriormente mencionado.

- **pantalla\_dato**: Variable que contiene la posición de la pantalla que se debe mostrar en el módulo OLED. Al cambiar el dato de esta variable, cambia la información mostrada en pantalla.
- **contador**: Variable que se utiliza para determinar si el sistema está en estado alarma, pendiente o sin problemas. También se aprovecha para los posibles falsos positivos.
- **estado\_sistema**: Variable utilizada globalmente para determinar el envío de información a los módulos microSD, serial y SigFox. Si es 0 el sistema está sin problemas.

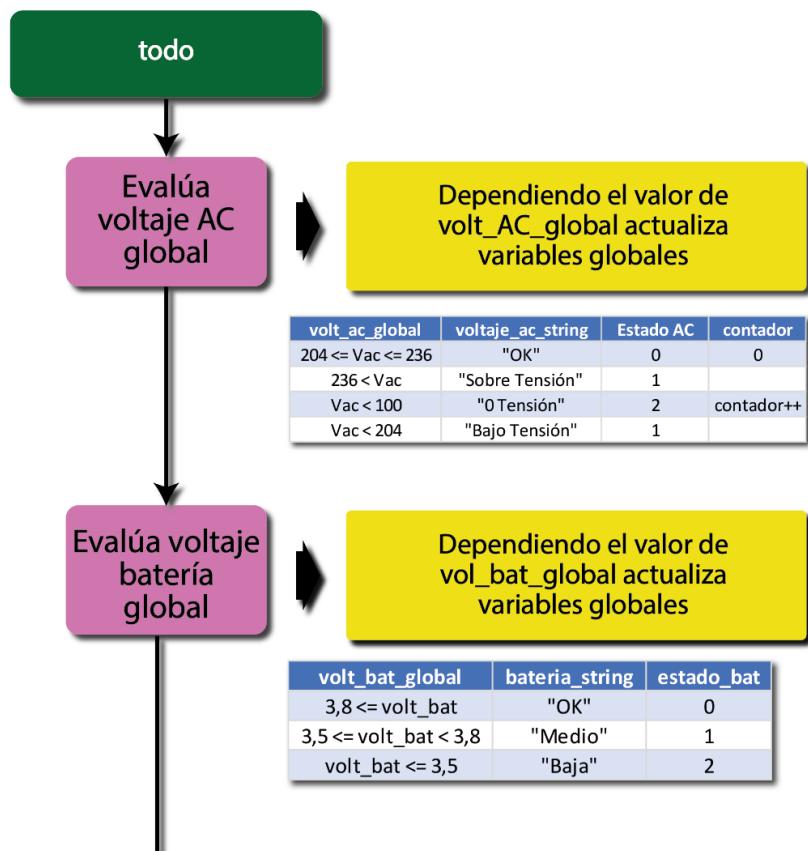


Diagrama 4 – Subrutina: todo (parte 1)

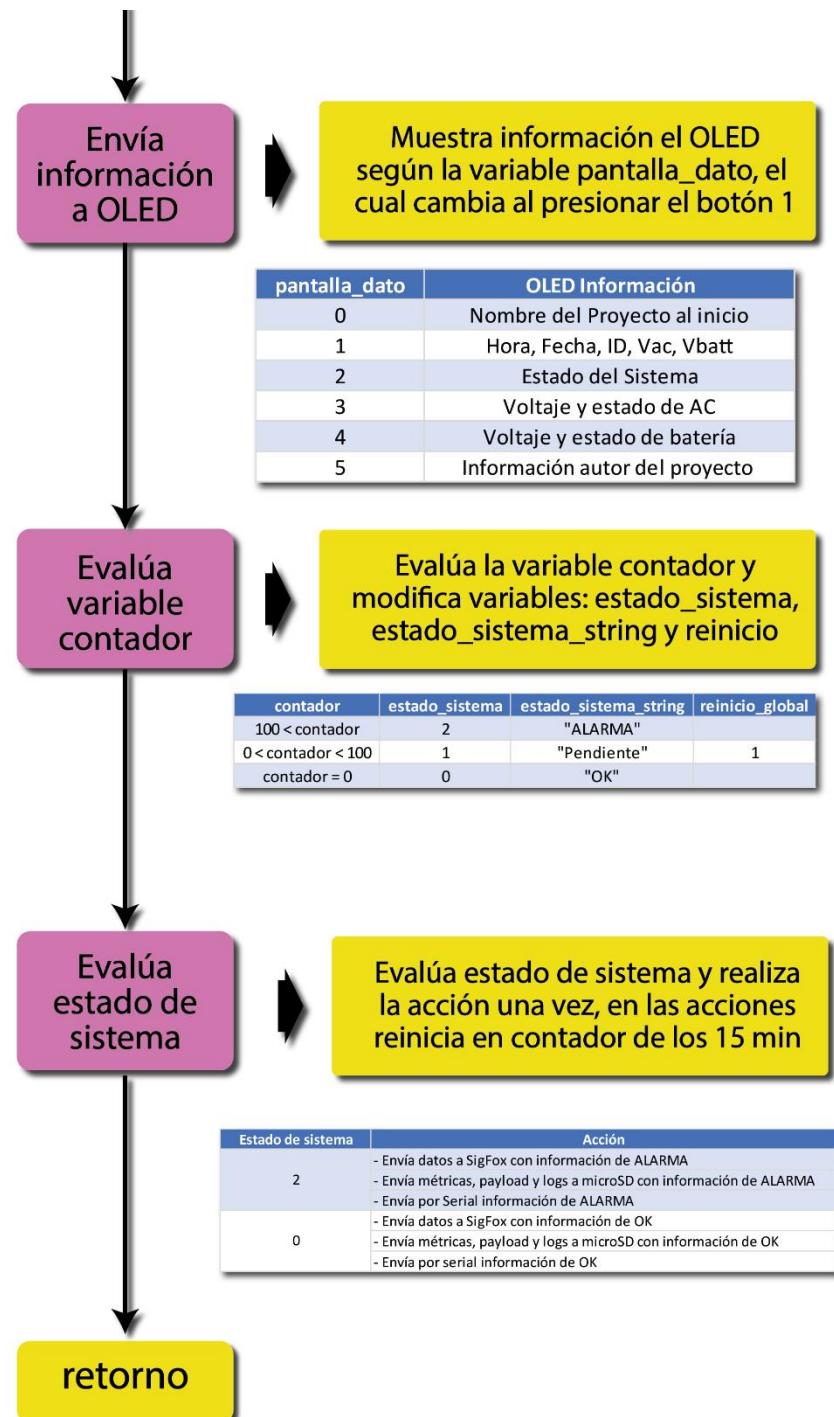


Diagrama 5 – Subrutina: todo (parte 2)

### 3.2.2.5 Subrutina clock

Subrutina que se encarga de actualizar la hora y fecha actual que entrega el módulo RTC (Real Time Clock) DS3231. Esta información se solicita por el protocolo I2C del microcontrolador.



Diagrama 6 - Subrutina: `clock`

### 3.2.2.6 Subrutina voltaje\_AC

Esta subrutina se encarga de solicitar el voltaje actual al sensor PZEM-004T vía serial. Para validar los datos, contiene una rutina de validación checksum, con la cual evitar que lleguen valores erróneos al microcontrolador.

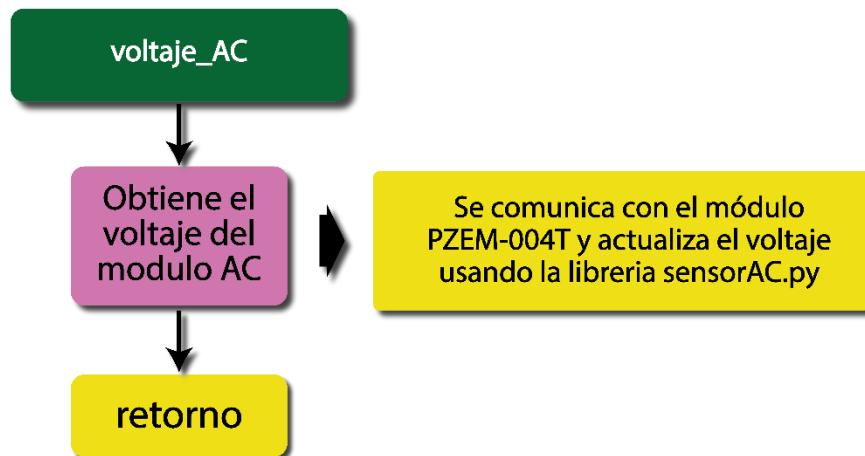


Diagrama 7 - Subrutina: `voltaje_AC`

### **3.2.2.7 Subrutina voltaje\_batería**

Esta subrutina se encarga de obtener el voltaje calculado al recibir datos del conversor análogo digital. Realiza la conversión de datos a voltaje con una fórmula matemática lineal (obtenida con datos reales y realizando una interpolación de estas).

También contiene una rutina para devolver el dato RAW de la lectura análoga, con fines de calibración.



*Diagrama 8 - Subrutina: voltaje\_batería*

### 3.2.2.8 Subrutina botón

Esta subrutina realiza un monitoreo de pulsadores switch. Es posible obtener actualización de cambios de estados de tres pulsadores, pero en este proyecto solo se utilizará 1. La subrutina puede discriminar pulsaciones simples, dobles y de tiempo prolongado; esta última contiene tres acciones más; cuando se inicia, durante y fin de la pulsación prolongada; lo que hace posible determinar diferentes acciones con un solo botón.



boton1	Acción
click	Cambia variable global pantalla_dato
doble click	Asigna un 1 a pantalla_dato
Comienzo de pulsación larga	Asigna un 5 a pantalla_dato
Fin de pulsación larga	Devuelve valor a pantalla_dato antes de comienzo de pulsación larga

Diagrama 9 - Subrutina: botón

### 3.2.2.9 Subrutina estado\_leds

Esta subrutina evalúa variables globales y cambia los colores de los leds de estados (WS2812b) dependiendo de cada variable. Para el sistema se asigna al LED 1, para el voltaje AC al LED 2 y finalmente para el voltaje de la batería, el LED 3.

En cada variable se pueden obtener 3 estados (0, 1, 2), siendo el de menor valor el estado de OK. Cada vez que realiza una acción, envía datos a la memoria microSD y al puerto serial con la información correspondiente.

Todos los datos enviados tienen una marca de tiempo, así es relativamente fácil saber cuándo cambió de estado.

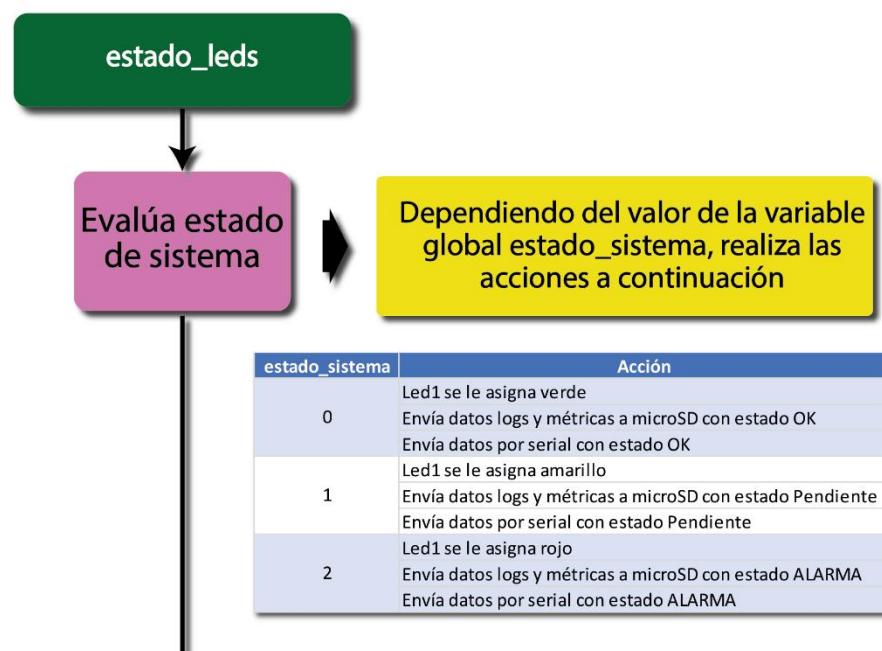


Diagrama 10 - Subrutina: `estado_leds` (Parte 1)

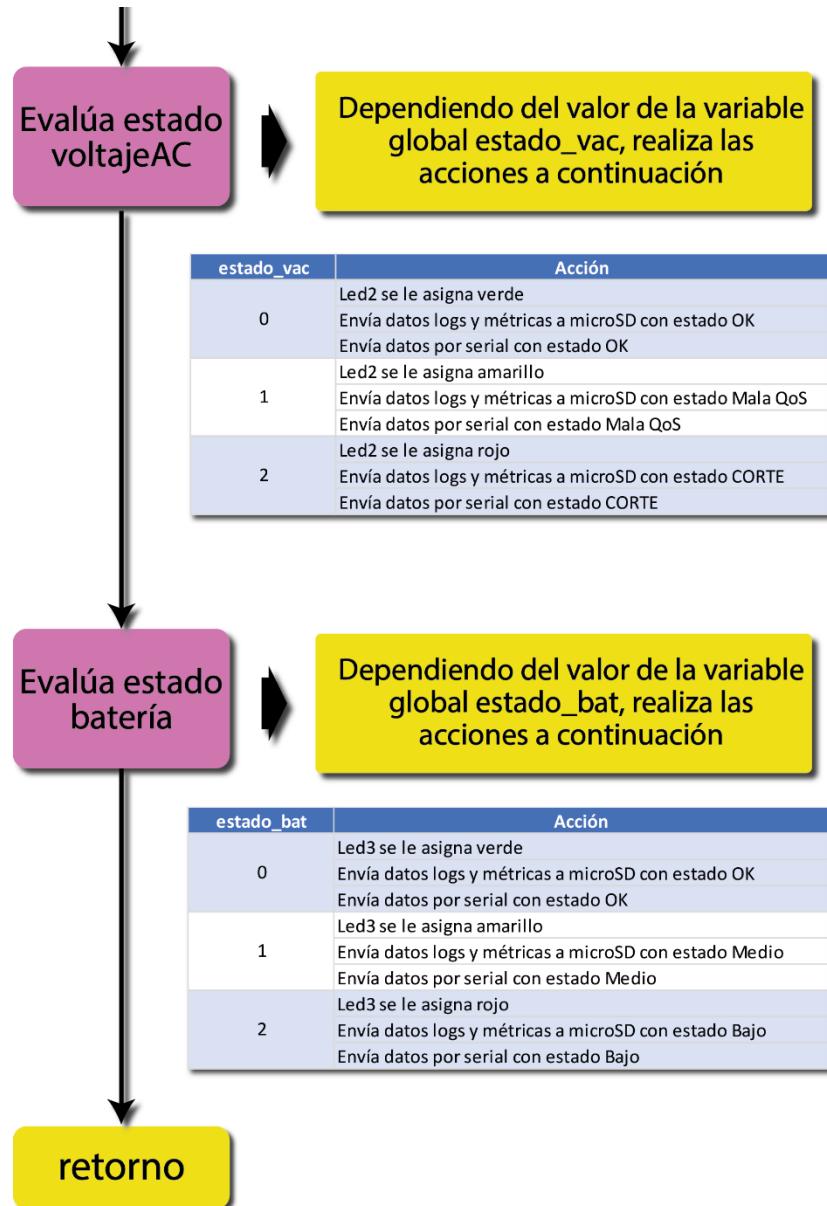


Diagrama 11 - Subrutina: `estado_leds` (Parte 2)

### 3.2.3 Metodología de programación

La metodología usada es probar el funcionamiento de cada módulo por separado. Esto asegura el correcto funcionamiento de cada módulo y evita fallas absurdas al programar el sistema con todos los módulos integrados. En pocas palabras se ahorra tiempo.

Se utilizaron rutinas de testeos de los módulos por separado, las cuales quedaron guardadas en caso de alguna falla.

#### 3.2.3.1 Modo de testeo de módulos

En el modo de testeo de módulos, la estructura de los archivos fue de la manera que se muestra en la Ilustración 56, las cuales dependen de las mismas librerías que el programa principal.



Ilustración 58 - Archivos de testeo

Durante el proceso de diseño es común hacer modificaciones, por ejemplo, utilizar un puerto de comunicación que comparte protocolo con otro módulo, esto

puede generar problemas a la hora de integrarlos, de modo que para minimizar las posibles razones de las fallas, se prueban los módulos por separado, el RTC, la pantalla OLED, los leds, los botones, entre otros.

Una vez seguros de que todos los módulos están funcionando con las librerías creadas y descargadas, viene el modo de integración.

### **3.2.3.2 Modo integración**

En este modo se integran los de mayor prioridad, es decir, aquellos a los cuales no se le puede hacer modificaciones de manera sencilla. En el caso de nuestro proyecto, los de mayor prioridad son:

- Módulo RTC
- Sensor AC
- Módulo SigFox

Una vez integrados y funcionando correctamente se procede con los de prioridad media, estos son:

- Módulo OLED
- Módulo MicroSD
- Módulo ADC

Y finalmente los de menor prioridad

- Pulsadores Switch
- Leds

Finalmente, cuando todo ya esté funcionando se procede al modo prueba en terreno y estrés.

### 3.2.3.3 Modo prueba en terreno y estrés

En este modo se deja en funcionamiento el dispositivo, generando artificialmente fallas en la línea eléctrica. Esto puede ser de forma manual (conectando y desconectando la toma de corriente) o programando un dispositivo Timer para que lo realice de forma automática. Estas pruebas deben ser durante un tiempo prolongado, para observar posibles fallas durante la prueba.

Luego se compara datos teóricamente enviados (avalados por el log que se guarda en la MicroSD) con los datos reales en el backend de SigFox.

Para esto se reducen los archivos a 2, boot.py y main.py, manteniendo las mismas librerías que están en la carpeta “libs” (Ilustración 57).



Ilustración 59 - Archivos principales

Los siguientes archivos (Ilustración 58) conforman la carpeta libs, carpeta donde incluye las librerías usadas a lo largo del proyecto. Los archivos ds3231.py, botones.py, ssd1306.py y ws2812.py son librerías descargadas de Internet para las funciones básicas de cada módulo conectado al microcontrolador, botones para pulsadores switch; ds3231 para el módulo RTC; ssd1306 para la pantalla OLED y finalmente ws2812 para los Leds RGB direccionables.

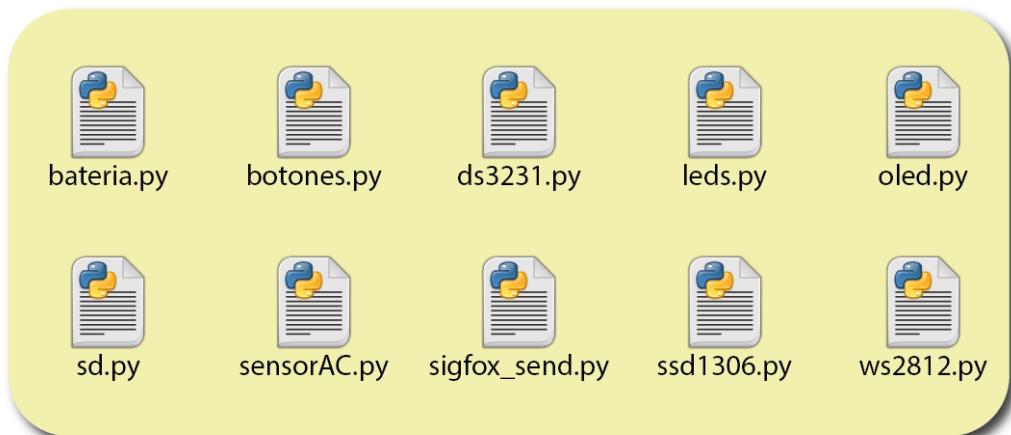


Ilustración 60 - Archivos de librerías

El tiempo y las veces que se deben realizar las pruebas depende exclusivamente de la cantidad de fallas encontradas durante el proceso. Idealmente después de cada periodo se debe realizar una optimización y corrección al código, si una falla desaparece, entonces se procede con la siguiente falla.

Teóricamente, cuando no existan fallas se puede decir que el dispositivo está terminado, pero en la práctica no es así. Solo resta diseñar y crear dispositivos lo más robustos posible para minimizar considerablemente las posibles fallas.

#### **4. RESULTADOS Y CONCLUSIONES**

Una vez puesto en marcha el prototipo, se necesitaba realizar fallas para testear la estabilidad. Para esto se implementó un dispositivo relé programable llamado SONOFF. Con este módulo, se pudo programar un corte eléctrico a una hora determinada y una reanudación del suministro eléctrico pasado los 5 minutos. Como solo se permite un máximo de 16 eventos, se programó 8 ciclos de corte y reposición del suministro. Esto permitió planificar de forma más sencilla los cortes programados y la elección del horario fue de forma aleatoria.

Se realizaron 2 pruebas del prototipo, en febrero 2020 desde el día 10 al día 29 (período 1) y en marzo 2020 desde el día 09 al 31 (período 2).

Las planificaciones de fallas programadas y sus resultados se resumen en las siguientes tablas.

## Fallas programadas en Febrero (5 minutos de duración)

	00:00	01:00	02:00	03:00	04:00	05:00	06:00	07:00	08:00	09:00	10:00	11:00	12:00	13:00	14:00	15:00	16:00	17:00	18:00	19:00	20:00	21:00	22:00	23:00	
10-02-2020	OK	OK	OK		OK										OK										
11-02-2020				OK			OK			OK											OK	OK			
12-02-2020	OK								OK												OK	OK		OK	
13-02-2020			OK		OK	OK	OK															OK			
14-02-2020				OK	OK			OK									X			OK	OK				
15-02-2020	OK				OK		OK								OK		OK								
16-02-2020					OK	OK							OK	OK										OK	
17-02-2020							OK	OK			OK	OK	OK												
18-02-2020					OK	OK																OK	OK	OK	
19-02-2020	OK					OK	OK							OK	OK										
20-02-2020	OK							OK			OK			OK		OK									
21-02-2020					OK			OK	OK				OK								OK				
22-02-2020						OK	OK		OK					OK								OK			
23-02-2020					OK					OK		OK			OK								OK		
24-02-2020																									
25-02-2020																									
26-02-2020																									
27-02-2020																									
28-02-2020																									
29-02-2020																									

OK Falla provocada y registrada exitosamente en plataforma IoT.

X Falla del prototipo no especificado, no registrado en plataforma IoT.

Tabla 2 - Planificación de fallas programables Periodo 1

En la Tabla 2 se puede observar que, en cada falla generada, el prototipo reaccionó y envió los datos al servidor SigFox de forma exitosa, excepto desde el día 24, cuando ocurrió una falla de la fuente de poder que cargar la batería. Este cargador demostró un bajo rendimiento al utilizarlo por largos períodos de tiempo.

A raíz de esta falla también se comprobó que el prototipo es capaz de funcionar por alrededor de 6 horas sin fuente de alimentación, con una sola batería Li-Po.

Al analizar los datos capturados en la memoria microSD se encontró una falla que no se reflejaba en la plataforma IoT. El día 14 de febrero alrededor de las 13 horas se generó una falla no especificada del prototipo, esto quiere decir que no

se sabe la razón de la falla, solo que el módulo quedó con la comunicación bloqueada por alrededor de 4 horas sin intervención humana y se restableció sin razón aparente.

### Fallas programadas en Marzo (5 minutos de duración)

	00:00	01:00	02:00	03:00	04:00	05:00	06:00	07:00	08:00	09:00	10:00	11:00	12:00	13:00	14:00	15:00	16:00	17:00	18:00	19:00	20:00	21:00	22:00	23:00	
09-03-2020			OK	OK							OK								OK	OK					
10-03-2020				OK							OK				F				OK					OK	
11-03-2020	OK								OK				OK			OK			OK						
12-03-2020		OK	OK			OK	OK												OK						
13-03-2020									OK	F		OK												OK	OK
14-03-2020			OK	OK				OK													OK	OK			
15-03-2020					OK			OK										OK		OK				OK	
16-03-2020		OK		OK	OK														F	OK					
17-03-2020		OK	OK								F													OK	OK
18-03-2020	OK						OK	OK	OK				OK												
19-03-2020	OK							OK					OK		OK									OK	
20-03-2020				OK		OK							OK				N	N							
21-03-2020	N											OK							OK					OK	OK
22-03-2020						OK	OK						OK	OK										OK	
23-03-2020		OK					OK		OK			OK												OK	
24-03-2020									OK	OK	OK	OK						OK							
25-03-2020				OK	OK										OK					OK	F				
26-03-2020		OK	OK	OK							OK	OK									OK	F			
27-03-2020	OK										F			OK	OK				OK						
28-03-2020											OK				OK	OK					OK	OK			
29-03-2020									F	F					OK		OK		OK						
30-03-2020											OK													X	
31-03-2020						OK			OK	OK	OK							OK						OK	

- OK Falla provocada y registrada exitosamente en plataforma IoT.
- F Falla provocada exitosamente pero no registrado en plataforma IoT.
- N Falla humana al no programar el dispositivo SONOFF
- X Falla del prototipo no especificado, no registrado en plataforma IoT.

Tabla 3 - Planificación de fallas programables Periodo 2

En el período 2 (marzo) se observa que existieron fallas, entre ellas se tiene 8 fallas registradas en la cual el prototipo sí detectó el corte de suministro pero no logró enviar los datos por SigFox. También se tienen 3 fallas registradas en las cuales por falla humana no se programó el relé SONOFF de forma correcta. Esto quiere decir que no se le atribuye la falla ni al prototipo ni a algún componente

electrónico, ni a la comunicación con SigFox. Y tenemos 2 fallas más graves en los días 14 y 30, donde la falla es exclusivamente del sensor de voltaje. Después de una reposición del suministro eléctrico, el sensor no era capaz de inicializarse como corresponde, lo cual se debe a que su sistema al ser opto acoplado, funciona con la alimentación exclusiva de la red eléctrica y no con la alimentación del prototipo. En futuras pruebas se contempla probar el mismo sensor pero la versión más actualizada.

El backend de SigFox tiene un recurso informático que permite descargar los datos más relevantes de las estadísticas del sistema. Con estos datos podemos determinar la cantidad de mensajes por día que el prototipo ha enviado, la calidad de la señal de conexión, inferir a través del número de secuencia cuántos mensajes se han perdido y cuándo se supera el máximo permitido de mensajes diarios. Los siguientes gráficos resumen dichos valores separados en los períodos de prueba.

## Período 1 (febrero)



Gráfico 2 - Mensajes perdidos y enviados por día, (periodo 21).

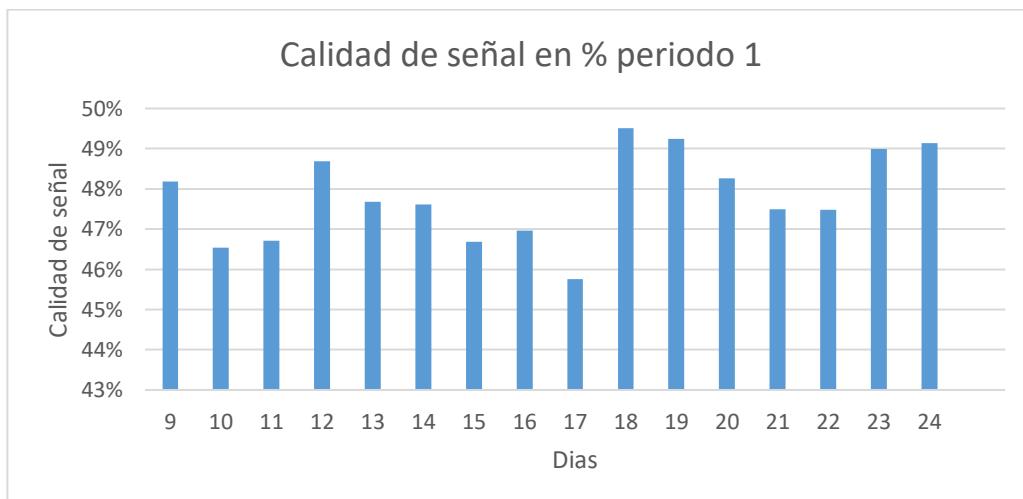


Gráfico 3 - Calidad de la señal en % por día, (periodo 1).



Gráfico 4 - Relación señal ruido SNR por día, (periodo 1).



Gráfico 5 - Fuerza de la señal RSSI por día, (periodo 1).

## Periodo 2 (marzo)

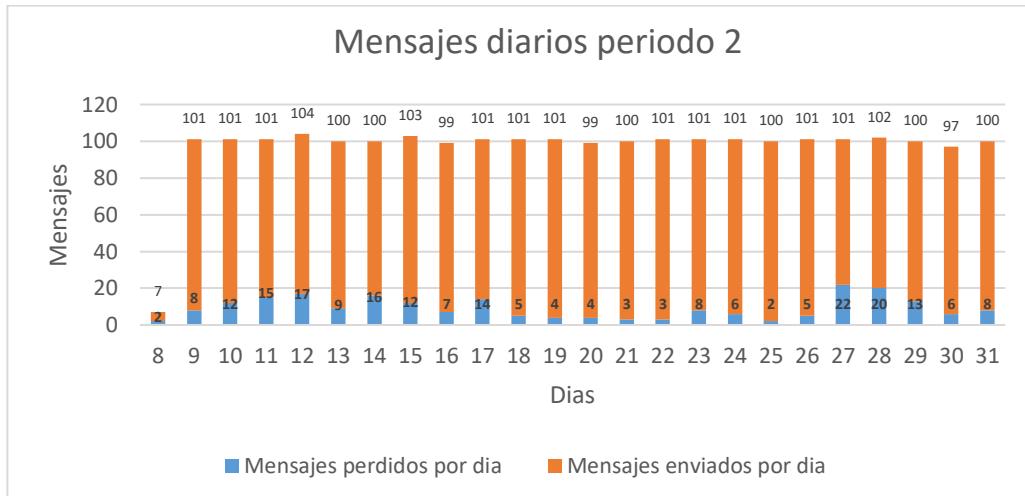


Gráfico 6 - Mensajes perdidos y enviados por día, (periodo 2).

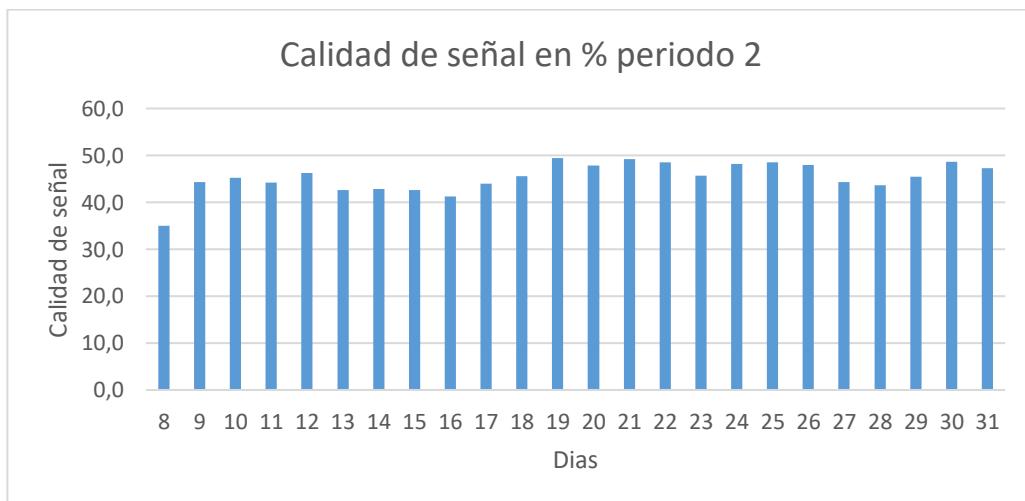


Gráfico 7 - Calidad de la señal en % por día, (periodo 2).



Gráfico 8 - Relación señal ruido SNR por día, (periodo 2)



Gráfico 9 - Fuerza de la señal RSSI por día, (periodo 2).

## Voltaje de batería y red eléctrica

Las mediciones de voltaje de la red eléctrica vienen ya procesadas desde el módulo PZEM-004T, que tiene una precisión según el fabricante de  $\pm 0.5\%$ . Considerando que el multímetro utilizado (Ilustración 59) para comparar la medición tiene una precisión de  $\pm 1.2\%$  en voltaje AC con escala de 750Vac y  $\pm 0.8\%$  en voltaje DC con escala de 20Vdc. Ahora se puede determinar los errores de las mediciones tanto de la red eléctrica como de la batería.



Ilustración 61 - Multímetro de medición, medición voltaje red eléctrica y batería

La siguiente tabla resume los valores obtenidos (Ilustración 59) y sus errores respectivos.

	Voltaje AC - Red eléctrica	Voltaje DC - batería LiPo	
	PZEM-004T $\pm 0,5\%$	Multímetro $\pm 1,2\%$	Divisor de voltaje
Medición	223,2 V	222 V	4,12 V
Error %	0,54 %		1,20 %

Tabla 4 - Resumen del error en medición de voltaje red eléctrica y batería

En conclusión, SigFox simplifica mucho el envío de datos a Internet, ya que no es necesario implementar la red de antenas, repetidores y enlaces que se necesita para una solución IoT. A pesar de sus restricciones de cantidad de mensajes diarios, se puede realizar una gran variedad de soluciones usando esta tecnología. Para otras aplicaciones que necesitan de una estructura más sólida, existen alternativas, aunque más costosas, o si se prefiere usar tecnología más accesible pero con menos robustez. Por eso es muy relevante realizar una evaluación y planificación detallada para aplicar una solución IoT.

Las recomendaciones que podemos dar para la utilización de SigFox son las siguientes:

- Para métricas de monitoreo de variables cuya variación se registra lentamente en períodos de tiempo que no requieren una periodicidad muy alta de muestreo, como temperatura, humedad o luz, SigFox es perfecto y no presenta problemas.
- Para alertas de emergencia como por ejemplo un botón de pánico, la recomendación es enviar una ráfaga de datos (unas 3 o 4 veces) para asegurarse de que llegará la información al servidor, ya que en las pruebas realizadas se registraron fallas de envío de datos hasta 4 veces seguidas, perdiendo una ventana de métricas de 1 hora.
- Para control domótico no es recomendable SigFox, ya que al tener límites de mensajes diarios de subida y de bajada, el factor tiempo real no es posible. En realidad no está diseñado para esto. Se recomienda un servidor MQTT a través de comunicación por WiFi o Red Celular.
- En la actualidad el acceso a SigFox en Chile es sólo para desarrolladores de soluciones IoT. En un futuro cercano ya se estará en condiciones de poder ofrecer servicio a todos, pero con relación a los módulos que soportan SigFox, Chile aún no ha conseguido romper barreras para que los dispositivos realmente sean de bajo costo como promocionan en

Europa, mencionando que un dispositivo puede costar unos 5 o 10 dólares pero que en la realidad acá en Chile rodean los 90 dólares.

- Y por supuesto se puede integrar tecnologías diferentes, desde lo básico que es WiFi pasando por GPRS, 3G, LTE e incluso tecnologías muy parecidas a SigFox como lo es LoRaWAN, logrando así una aplicación IoT muy robusta y confiable.

Mejoras que se podrían realizar al prototipo a futuro:

- Integrar una alarma sonora con buzzer para evitar que se descargue la batería completamente en caso de falla del cargador.
- Configurar en plataforma IoT una alerta adicional que notifica en caso de que el voltaje baje cierto umbral.
- Integrar una entrada analógica para monitorear el voltaje de la salida del cargador, así el sistema se asegura de que está cargando la batería bien.
- Para descartar fallas de comunicación se puede utilizar un analizador de espectro al momento de enviar señales RF para asegurarse si es problema de hardware o de comunicación.

## 5. BIBLIOGRAFÍA

- [1] J. Novillo-Vicuña, «www.3ciencias.com,» [En línea].  
Available: <https://www.3ciencias.com/wp-content/uploads/2018/10/ARDUINO-Y-EL-INTERNET-DE-LAS-COSAS.pdf>.  
[Último acceso: 27 de noviembre 2022].
- [2] F. P. Digital, «Cómo emprender en Internet de las Cosas: Conceptos prácticos,» [En línea].  
Available: <http://dg6223fhe15c2.cloudfront.net/PD/wp-content/uploads/2018/02/Como-emprender-en-Internet-de-las-Cosas.pdf>.  
[Último acceso: 27 de noviembre 2022].
- [3] <https://www.cisco.com>, «<https://www.cisco.com>,» [En línea].  
Available:  
[https://www.cisco.com/c/dam/global/es\\_es/assets/executives/pdf/Internet\\_of\\_Things\\_IoT\\_IBSG\\_0411FINAL.pdf](https://www.cisco.com/c/dam/global/es_es/assets/executives/pdf/Internet_of_Things_IoT_IBSG_0411FINAL.pdf).  
[Último acceso: 27 de noviembre 2022].
- [4] SUBTEL, «Estudios y Estadísticas, Abonados Móviles,» [En línea].  
Available: <https://www.subtel.gob.cl/estudios-y-estadisticas/telefonia/>.  
[Último acceso: 27 de noviembre 2022].
- [5] L. G. C. Ramírez, *Sensores y Actuadores*, Renacimiento: GRUPO EDITORIAL PATRIA, S.A., 2015.

- [6] InnovatorsGuru, «<https://innovatorsguru.com>,» [En línea]. Available: <https://innovatorsguru.com/ac-digital-multifunction-meter-using-pzem-004t/>. [Último acceso: 27 de noviembre 2022].
- [8] A. Systems, «Diferencias entre NB-IoT y LTE-M,» [En línea]. Available: <https://accent-systems.com/es/blog/diferencias-nb-iot-lte-m/?v=5bc574a47246> [Último acceso: 27 de noviembre 2022].
- [9] <https://www.catsensors.com>, «<https://www.catsensors.com>,» [En línea]. Available: <https://www.catsensors.com/es/lorawan/tecnologia-lora-y-lorawan>. [Último acceso: 27 de noviembre 2022].
- [10] Pycom, «<https://docs.pycom.io>,» [En línea]. Available: <https://docs.pycom.io/index.html>. [Último acceso: 27 de noviembre 2022].