INFOTEC CENTRO DE INVESTIGACIÓN E INNOVACIÓN EN TECNOLOGÍAS DE LA INFORMACIÓN Y COMUNICACIÓN

DIRECCIÓN ADJUNTA DE INNOVACIÓN Y CONOCIMIENTO
GERENCIA DE CAPITAL HUMANO
POSGRADOS

"DESARROLLO DE DISPOSITIVO IOT PARA MONITOREO AGRÍCOLA DE BAJO COSTE"

IMPLEMENTACIÓN DE UN PROYECTO LABORAL
Que para obtener el grado de MAESTRO EN SISTEMAS EMBEBIDOS

Presenta:

Cristian Jael Mejia Aguirre

Asesores:

Dr. Dagoberto Armenta Medina
Dr. Tania Aglae Ramírez del Real

Aguascalientes, marzo 2020.

AUTORIZACIÓN DE IMPRESIÓN Y NO ADEUDO EN BIBLIOTECA

MAESTRÍA EN SISTEMAS EMBEBIDOS

Ciudad de México, 15 de julio de 2020.
INFOTEC-DAIC-GCH-SE-0441/2020.

La Gerencia de Capital Humano / Gerencia de Investigación hacen constar que el trabajo de titulación intitulado

DESARROLLO DE DISPOSITIVO IOT PARA MONITOREO AGRÍCOLA DE BAJO COSTE

Desarrollado por el alumno **Cristian Jael Mejia Aguirre** y bajo la asesoría del **Dr. Dagoberto Armenta Medina** y la **Dra. Tania Aglae Ramírez del Real**; cumple con el formato de biblioteca. Por lo cual, se expide la presente autorización para impresión del proyecto terminal al que se ha hecho mención.

Asimismo se hace constar que no debe material de la biblioteca de INFOTEC.

Vo. Bo.

Mtra. Julieta Alcibar Hermosillo
Coordinadora de Biblioteca

Anexar a la presente autorización al inicio de la versión impresa del trabajo referido que ampara la misma.

C.p.p Servicios Escolares

Agradecimientos

Son muchas las personas que han formado parte de mi vida profesional a las que me encantaría agradecerles por su amistad, consejos, apoyo, ánimo y compañía en los momentos más difíciles de mi vida. Especialmente a mis padres Alberto y Mónica que se esfuerzan cada día para darnos lo mejor a mí y a mis hermanos. Les agradezco a todos, algunos están aquí conmigo, otros están en mis recuerdos y en mi corazón. Sin importar en donde estén quiero darles las gracias por formar parte de mí logro, por todo lo que me han brindado y por todas sus bendiciones.

Tabla de contenido

Índice de figuras

Índice de cuadros

Índice de gráficos

Siglas y abreviaturas

I^2C: Inter-Integrated Circuit

BNC: Bayonet Neill-Concelman

DC-DC: Direct Current-Direct Current

GPS: Global Positioning System

HASL: Hot Air Solder Leveling

IDE: Integrated Development Environment

IoT: Internet of Things

IP: Ingress Protection

IPC: Institute of Printed Circuits

LiPo: Lithium-ion Polymer

LoRa: Long Range

LPWAN: Low-Power Wide-Area Network

M2M: Machine-to-Machine

MCU: Microcontroller Unit

NTC: Negative Temperature Coefficient

OEM: Original Equipment Manufacturer

PCB: Printed Circuit Board

pH: Power of Hydrogen

RGB: Red-Green-Blue

RoHS: Restriction of Hazardous Substances

SPI: Serial Peripheral Interface

UART: Universal Asynchronous Receiver-Transmitter

USB-TTL: Universal Serial Bus-Transistor Transistor Logic

WSN: Wireless Sensor Network

Introducción

En el presente documento se detalla la construcción e implementación de un sistema embebido mediante el uso de tecnologías avanzadas de la información y comunicación en agricultura (agricultura inteligente). El desarrollo consiste en un sistema embebido que integra un módulo de GPS, además de contar con diferentes sensores para la monitorización de variables agrícolas como el índice de clorofila en plantas, así como el pH, y temperatura del suelo. De manera adicional, este dispositivo integra un panel solar con una batería mediante un sistema de control, lo cual proporciona un buen desempeño en la autonomía energética del dispositivo. El desempeño de este dispositivo se contrastó con instrumentos de medición empleados en laboratorio de suelos y a nivel campo, logrando un desempeño adecuado, para la monitorización de variables agrícolas que facilitan la toma de decisiones en el manejo de cultivos. Otro aspecto para resaltar es el diseño robusto y modular, desarrollado e implementado en la carcasa y de los componentes estructurales del sistema embebido.

Capítulo 1

Contexto tecnológico

Capítulo 1. Contexto tecnológico

Los sistemas embebidos son una combinación de hardware y software diseñados para realizar una función específica, ya sea como un sistema independiente o como parte de un sistema más grande [1] . La complejidad de un sistema embebido varía significativamente según la tarea a realizar. Opuesto a las cualidades de las computadoras de propósito general, que están construidas para cubrir un amplio rango de tareas. Sus principales características son el bajo costo, seguridad y reducción de consumo de energía [2]. Además, se construyen para efectuar necesidades dedicadas y en tiempo real. También, el costo por unidad es un aspecto importante ya que son elaborados en una gran cantidad.

Los sistemas embebidos están altamente relacionados con el paradigma del Internet de las Cosas (IoT, por sus siglas en inglés). El IoT es un concepto abstracto que refiere representar la conexión de cualquier cosa que exista al Internet. Asimismo, el IoT extiende las capacidades del Internet para permitir la comunicación de máquina a máquina (M2M, por sus siglas en inglés) [3]. La comunicación con los objetos presume resolver problemas y predecir eventos con el uso de Inteligencia Artificial y Big data [4]. El IoT se divide en sectores llamados dominios de aplicación, cada dominio se encarga de un conjunto específico de objetos relacionados entre sí [5]. Ejemplos de ello son: salud, industria, servicios, energía, transporte, agricultura, entre otros. Este proyecto abordará el dominio de la agricultura.

1.1 Agricultura inteligente

Se estima que para el 2050 la demanda de alimento se elevará en 70% y la agricultura deberá adaptarse al cambio climático y contribuir a mitigar sus efectos [6], [7]. Por otro lado, la tierra de uso agrícola estará disminuyendo debido a varias razones, como la industrialización, centros comerciales y edificios residenciales construidos [8]. Derivado de los retos anteriores, una necesidad imperante para lograr la sustentabilidad en la productividad y calidad de los alimentos es desarrollar tecnologías que permitan tener una administración precisa de los fertilizantes

nitrogenados y variables agrícolas mediante un manejo diferenciado por zonas dentro de los terrenos de cultivos. Además, las ventajas económicas que conlleva aplicar las dosis correctas de fertilizante en el lugar correcto también representan grandes ventajas ecológicas, ya que reduce los problemas de eutrofización de los mantos acuíferos asociado a la contaminación agrícola.

Una alternativa a los retos anteriores es la convergencia de la ciencia de datos, electrónica, tecnologías de la información y comunicación con la agricultura, derivando hoy en día en la llamada agricultura inteligente. La agricultura inteligente (*Smart Agriculture*) es un concepto de gestión agrícola que utiliza tecnologías modernas de información y comunicación, como el IoT, para aumentar la cantidad y la calidad de los productos [9]. También, presenta un potencial real para un incremento en la sostenibilidad y productividad agrícola, basada en un uso más eficiente y preciso en el manejo de recursos.

Al medir con precisión las variaciones dentro de un campo y adaptar la estrategia en consecuencia, los productores pueden aumentar en gran medida la efectividad de los recursos como pesticidas, fertilizantes, entre otros y usarlos de manera más selectiva. Esto se puede lograr con sistemas embebidos IoT de monitoreo. Igualmente, la información recolectada puede ser empleada para evaluar con mayor precisión la densidad óptima de siembra, estimar la cantidad adecuada de insumos necesarios, y predecir con más exactitud el rendimiento y la producción de los cultivos. En este documento, se discuten los pasos involucrados para el desarrollo de un sistema embebido IoT de monitoreo en el sector agrícola [9].

1.2 LPWAN y WSN

El desarrollo de sistemas embebidos IoT en la agricultura inteligente tiene retos tecnológicos interesantes. Anteriormente las arquitecturas de red no eran tan adecuadas para estos sistemas, ya que son costosas, de corto alcance y la infraestructura es compleja. Sin embargo, la creación de la arquitectura de red LPWAN demuestra más ventajas. Las redes de área amplia y de baja potencia (LPWAN, Low Power Wide Area Network) son un tipo de red para transporte

inalámbrico de datos que hoy en día se entiende como uno de los estándares básicos para la implementación del IoT [10]. Sus atributos tecnológicos son el largo alcance, la baja potencia, la baja tasa de transmisión y el bajo costo. El alcance geográfico operativo de la tecnología LPWAN varía desde unos pocos kilómetros en áreas urbanas, hasta más de 10 km en entornos rurales. Los transceptores LPWAN pueden funcionar con baterías pequeñas y económicas con una duración de años en comparación con semanas y meses. También, los protocolos simplificados de LPWAN reducen la complejidad en el diseño del hardware y reducen los costos del transceptor. Además, su largo alcance combinado con una topología en estrella reduce los costosos requisitos de infraestructura, el uso de bandas sin licencia disminuye los costos de red. La cantidad de datos transmitidos es regular, pero justa para aplicaciones IoT, ya que no permite manejar grandes volúmenes de datos por lo que elementos como fotos y videos están por completo descartados.

Las LPWAN son similares a las redes de sensores inalámbricas (WSN, Wireless Sensor Network) por la topología equivalente que comparten. Las WSN son un conjunto de dispositivos equipados con sensores que colaboran en una tarea en común [11]. Asimismo, estos sensores se utilizan para monitorizar condiciones físicas o ambientales, como temperatura, humedad, presión, etc.

1.3 LoRa

LoRa (*Long Range*) que significa largo alcance, es una técnica de modulación que permite la transferencia de información a largo alcance con una baja tasa de transferencia. La modulación LoRa ha sido patentada por Semtech y es altamente utilizada en arquitecturas de red LPWAN para aplicaciones IoT [12]. La red LoRa se implementa utilizando la topología de red en estrella. Además, la estructura de una arquitectura LoRa se puede separar en una parte de back-end, y una de front-end. La parte de back-end consiste en el servidor de red que almacena la información recibida de los sensores, la parte de front-end consta de los gateway y de los dispositivos finales. Los módulos Gateway actúan como un puente entre los

dispositivos finales y el servidor de red. La información entre el servidor de red y los módulos de Gateway se envía a través de la conexión IP. También, la seguridad de la red LoRa puede garantizarse en cada transmisión, ya que la señal se extiende de forma pseudoaleatoria que se presenta como un ruido, por lo tanto, la técnica de modulación proporciona seguridad básica para la red porque es difícil de interpretar la señal [13].

1.4 Definición del problema

Actualmente, existe la necesidad de los productores y de los proveedores de soluciones e insumos agrícolas, de monitorizar de una manera práctica y económica variables que sirvan para un adecuado diagnóstico de los suelos y cultivos en el campo, para el manejo diferenciado por zonas dentro del terreno cultivable. La diferenciación por zonas de acuerdo con las propiedades de los cultivo y suelos es facilitada por el sistema embebido, logrando un manejo más eficiente de los insumos agrícolas como el nitrógeno, ya que permite ubicar específicamente zonas deficientes en este macronutriente mediante el índice de clorofila, para la administración específica de fertilizante. Además del ahorro económico que conlleva aplicar las dosis correctas en el lugar correcto de los fertilizantes nitrogenados mediante el uso del sistema embebido, también representa grandes ventajas ecológicas, ya que reduce los problemas de eutrofización de los mantos acuíferos asociado a la contaminación agrícola [14]. También, la detección de características del suelo como el potencial de iones hidrogeno (pH) mediante la monitorización del sistema embebido favorece el diagnóstico para un adecuado tratamiento en la recuperación del suelo [15]. Normalmente, los niveles de pH alejados de la neutralidad son indicador de falta de disponibilidad de nutrientes, sin embargo, algunos cultivos son favorecidos por niveles de pH alejados y dependiendo en que extremo de la escala de pH se encuentren, es el tratamiento que se aplica [16].

1.5 Hipótesis

La convergencia de las tecnologías avanzadas de la información y comunicación permitirá desarrollar un sistema embebido de bajo costo y largo ciclo de vida útil en la monitorización de variables agrícolas con geolocalización y comunicación integrada para el diagnóstico de los suelos y cultivos.

1.6 Objetivos

En la presente subsección se establecen los objetivos, el general y los particulares.

1.6.1 Objetivo general

Diseñar y construir un sistema embebido IoT autónomo mediante energía solar para monitorización de variables agrícolas. El prototipo se caracteriza por ser de bajo costo manteniendo la ubicación de la medición de las variables dentro del terreno de cultivo, además su visualización de manera remota en tiempo real.

1.6.2 Objetivos específicos

- Diseñar arquitectura para el sistema embebido.
- Seleccionar sensores para monitorizar temperatura, humedad, acidez o alcalinidad (pH), iluminancia, índice de clorofila y posicionamiento.
- Diseñar e implementar de hardware para el sistema embebido.
- Desarrollar software embebido.
- Construir carcasa a prueba de agua y polvo para el dispositivo.
- Realizar verificación y validación del sistema embebido.

1.7 Justificación

Este proyecto contribuye en la disminución del impacto ambiental al favorecer una adecuada administración de los insumos agrícolas como los fertilizantes nitrogenados, pudiendo reducir los problemas de eutrofización asociado a la contaminación agrícola y cambio climático [17]. Por un lado, en el sector privado favorece la generación de servicios de alta tecnología en la industria agrícola. Por otro lado, en el sector público la disponibilidad del diseño del sistema embebido favorece la investigación y desarrollo tecnológico para la resolución de problemas agrícolas. Igualmente, contribuye económicamente a los agricultores favoreciendo el uso eficiente de los insumos agrícolas de alto costo como los fertilizantes nitrogenados por medio de la medición de variables en cultivos.

Capítulo 2

Desarrollo

Capítulo 2. Desarrollo

En este capítulo se aborda la construcción del sistema embebido, en particular, el diseño y ensamble de la tarjeta electrónica junto con la programación. Además, se presenta la comparación de costos contra otros dispositivos similares, empezando con la descripción del modelo de desarrollo a seguir.

2.1 Modelo de desarrollo

El modelo de desarrollo para el sistema embebido se divide en tres fases principales, la fase de análisis, implementación y administración. En la Fig. 1 se muestra el modelo de desarrollo.

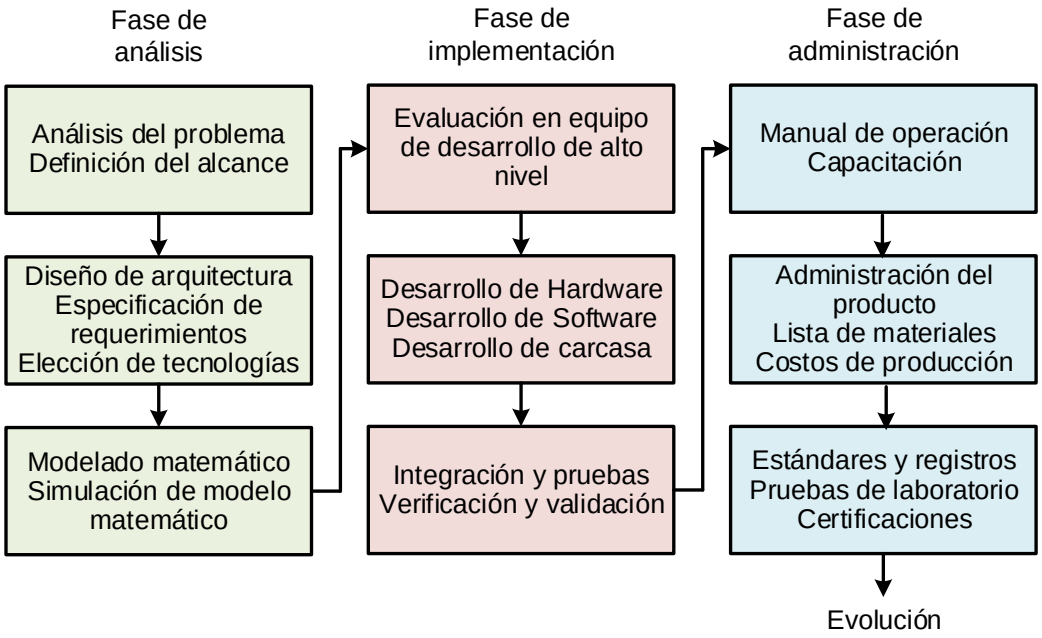


Figura 1: Modelo de desarrollo del sistema embebido. Fuente: Elaboración propia.

La fase de análisis se encarga de establecer el problema y la solución propuesta. Su objetivo principal es definir el alcance del proyecto, la especificación de los requerimientos y la tecnología a utilizar. La segunda fase tiene como objetivo implementar el sistema embebido. Pudiendo incluso realizar un prototipo en una tarjeta de desarrollo comercial para verificar algunas funciones de la plataforma.

Además, se debe construir una tarjeta electrónica dedicada, desarrollar el firmware y construir la carcasa. Luego se realizan pruebas de funcionamiento y validación de la integración del sistema completo. La fase de administración es la tercera fase de este modelo de desarrollo, en ella se realiza el manual de operación y se busca cumplir con los estándares o pruebas de laboratorio. El registro del producto también se realiza en esta fase. Por último, se tiene la fase de evolución en ella se hace retroalimentación para la siguiente versión del producto.

2.2 Diseño arquitectónico del sistema embebido

El diseño arquitectónico del sistema embebido se realiza para plasmar su estructura en el más alto nivel de diseño y define de manera abstracta los elementos que lo componen y la comunicación que existe entre ellos. El diseño arquitectónico del sistema embebido para monitorización agrícola se divide en 5 componentes principales. En la Fig. 2 se muestra el diseño arquitectónico. El componente principal es el controlador, se encarga de supeditar a los demás elementos e integra el *firmware* del dispositivo. El segundo componente es el de sensores, este reúne las variables físicas requeridas (temperatura, humedad, pH, iluminancia e índice de clorofila). El siguiente componente es el de comunicación, este delega la información que enviará y recibirá el sistema. Además, gestiona la identidad del dispositivo. En cuanto al componente de posicionamiento, este se encarga obtener y administrar la geolocalización del dispositivo. Por último, el componente gestor de energía monitoriza y dispone de manera eficiente la energía del dispositivo.

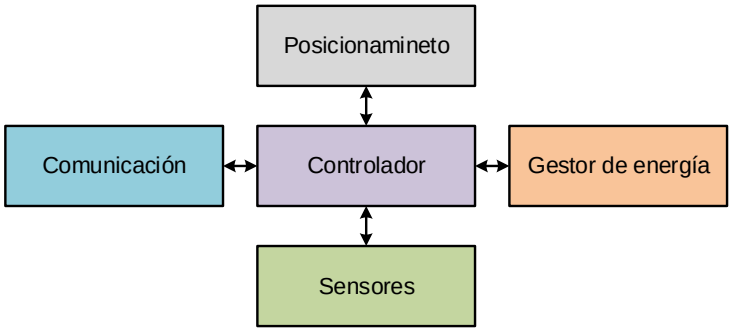


Figura 2: Diseño arquitectónico del sistema embebido. Fuente: Elaboración propia.

2.3 Especificación de requerimientos

Los requerimientos funcionales son declaraciones de los servicios que proveerá el sistema embebido, los requerimientos no funcionales son las limitaciones y certificaciones, se presentan en los Cuadros 1 y 2 respectivamente.

Número de requerimiento	Descripción
RF1	El dispositivo deberá sensar las magnitudes físicas: temperatura, humedad relativa, índice de clorofila, concentración de iones de hidrógeno (pH) e iluminancia.
RF2	El dispositivo integrará un GPS para su localización.
RF3	El dispositivo deberá ser autónomo por energía solar.
RF4	El dispositivo tendrá la capacidad de monitorizar el estado de la batería.
RF5	El dispositivo integrará gestor de energía de alta eficiencia.
RF6	El dispositivo enviará los datos adquiridos, localización y el estado de la energía utilizando el protocolo LoRa (Long Range).
RF7	La carcasa del dispositivo deberá ser de un material biodegradable.
RF8	La tasa de transmisión del dispositivo deberá ser configurable.

Cuadro 1: Requerimientos funcionales. Fuente: Elaboración propia.

Número de requerimiento	Descripción
RNF1	El tiempo para iniciar o reiniciar el dispositivo no podrá ser mayor a 2 segundos.
RNF2	Toda funcionalidad en el sistema debe de responder en menos en 1 segundo.
RNF3	El sistema debe ser capaz de operar adecuadamente las 24 horas del día, 7 días de la semana, 365 días al año.
RNF4	La temperatura de operación del sistema debe de ser de 0°C a 70°C.
RNF5	El sistema debe tener una disponibilidad del 99,99% de las veces en que un usuario intente accederlo.
RNF6	El dispositivo seguirá la normativa RoHS.
RNF7	La tarjeta electrónica deberá seguir los estándares de la IPC.
RNF8	La batería del dispositivo deberá seguir la norma UL.
RNF9	El dispositivo deberá cumplir con el grado de protección IP66.

Cuadro 2: Requerimientos no funcionales. Fuente: Elaboración propia.

2.4 Elección de tecnologías

Tomando en cuenta los requerimientos se debe elegir las tecnologías adecuadas para el desarrollo del proyecto. Se realiza la búsqueda de las tecnologías referente a los siguientes conceptos:

- Unidad de control
- Módulo GPS
- Sensores
- Gestor de energía
- Módulo de comunicación

2.4.1 Unidad de control

La unidad de control es la que dirige a los demás componentes y realiza las tareas del sistema embebido. Para este proyecto se elige un microcontrolador (MCU, por sus siglas en inglés) como unidad de control del dispositivo. La elección del microcontrolador se deduce porque es de bajo costo y bajo consumo de energía. Además, el desarrollo es rápido y para este proyecto no es necesario velocidad de procesamiento [18] [19].

2.4.2 Módulo GPS

Con el avance de la tecnología los módulos de Sistema de Posicionamiento Global (GPS, por sus siglas en inglés) han sido comercializados cada vez más, reduciendo su tamaño y aumentando su precisión [20]. Asimismo, logran tener bajo consumo de energía y un fácil manejo. Para el sistema embebido se requiere un módulo GPS con estas características haciendo énfasis al tamaño. Integrar la antena en el mismo modulo reduce el tamaño del dispositivo, ya que el espacio que requiere el módulo es menor.

2.4.3 Sensores

Actualmente, en el desarrollo de sistemas embebidos se utiliza el protocolo Circuito Inter-Integrado (I^2C, por sus siglas en inglés) para la comunicación con sensores y otros periféricos [21]. El I^2C es un bus de comunicación multipunto compuesto por un dispositivo maestro y dispositivos esclavos [22]. Asimismo, el bus I^2C cuenta con dos líneas de señal: una de reloj (*SCK*) y una de datos (*SDA*) [23]. Para esta aplicación el microcontrolador será el dispositivo maestro y los sensores los esclavos. La ventaja de utilizar este protocolo es que se pueden agregar diferentes tipos de sensores y de esta forma se obtiene un dispositivo modular.

2.4.4 Gestor de energía

El lanzamiento de nuevos productos con el concepto de IoT trajo consigo el desarrollo de nuevos circuitos integrados para la gestión de energía. El bajo consumo energético es un requerimiento obligatorio para el sistema embebido. Por esta razón, es necesario que el dispositivo tenga circuitos integrados de alta eficiencia. Este concepto involucra aspectos como la fuente de poder, la regulación de voltaje, el medio de carga y el monitoreo del sistema. Por una parte, como fuente de poder se tiene una batería recargable. La batería elegida para esta aplicación es la de polímero de iones de litio (LiPo, por sus siglas en inglés), ya que esta tecnología puede otorgar más corriente por hora que sus competencias y además su tamaño pequeño [24]. Por otra parte, la regulación de voltaje es crítica en esta aplicación. Se escoge el convertidor DC-DC como circuito de regulación porque a diferencia de los LDO y reguladores de aislamiento galvánico esta tecnología es de mayor eficiencia ya que la conversión de energía no se realiza por disipación de calor [25]. El siguiente aspecto importante es el medio de carga. Como se ha mencionado anteriormente el sistema embebido debe ser autónomo, es decir, debe tener la capacidad de generar energía por sus propios medios. Se opta por utilizar un panel fotovoltaico como medio de carga de la batería puesto que el ambiente de aplicación del sistema embebido está expuesto a la radiación solar directa [26]. El

último aspecto por considerar de la gestión de energía es la monitorización. Lo anterior se refiere específicamente a la vida de la batería. Para determinar la carga de la batería se debe medir su voltaje. Sin embargo, el comportamiento de la carga con respecto al voltaje es una función logarítmica, es decir, no es lineal [27]. Por esta razón, actualmente existen circuitos integrados que tienen internamente programado algoritmos para el cálculo de porcentaje de batería y para esta aplicación son predilectos.

2.4.5 Módulo de comunicación

Como se menciona en la introducción del documento, la tecnología de comunicación elegida es la de LoRa por sus características que se ajustan a los requerimientos del sistema embebido. Sin embargo, es preciso señalar que actualmente hay un solo fabricante de circuitos integrados y es la empresa Semtech. Por esta razón, se deduce que se usará el circuito integrado de Semtech [28].

2.5 Evaluación en equipo de desarrollo de alto nivel

Después de indagar sobre las tecnologías apropiadas para el desarrollo del sistema embebido. Se realiza la búsqueda de los componentes electrónicos requeridos. A causa de la investigación se encontró un equipo de desarrollo de alto nivel para aplicaciones IoT del fabricante Dragino. Se muestra en la Fig. 3.

Figura 3: Equipo de desarrollo de alto nivel Dragino. Fuente: Elaboración propia.

El kit de evaluación incluye dos tarjetas de desarrollo con comunicación LoRa, una ellas con GPS y aparte contiene un gateway LoRa. Conjuntamente, cuenta con una serie de sensores para hacer demostraciones de la tecnología. Como resultado de la experimentación del equipo de desarrollo Dragino se tiene la elección de algunos componentes electrónicos. Como es el caso del microcontrolador ATmega328P (Microchip Technology, AZ, USA), el módulo LoRa RFM95W (HOPERF, SZX, China) y el GPS L80 (Queltec, SZX, China). Las características principales del microcontrolador ATmega328P son el bajo consumo, la sencillez de su arquitectura y sobre todo su popularidad. Estas características también aplican para los demás componentes.

Luego, se hace la búsqueda de los sensores. Como consecuencia, se encuentra un fabricante que desarrolla sensores para el ámbito agrícola llamado Atlas Scientific. Esta empresa ofrece los sensores de pH e índice de clorofila para aplicaciones agrícolas. En la Fig. 4 se ilustra el sensor de clorofila EZO-RGB (Atlas Scientific, NY, USA).

Figura 4: Sensor de índice clorofila EZO-RGB de Atlas Scientific.
Fuente: Elaboración propia.

Por consiguiente, el sensor de pH se compone de dos elementos: la sonda de medición y el acondicionador de señal. El acondicionador de señal es un módulo electrónico pH Fabricante Original de Equipo (OEM, por sus siglas en inglés) que utiliza el protocolo I^2C y como se muestra en la Fig. 6. La sonda de pH es el transductor de este sensor y tiene un conector BNC (*Bayonet Neill-Concelman*), se ilustra en la Fig 5. Siendo así, se necesita una tarjeta electrónica para comprobar el funcionamiento de los dos elementos.

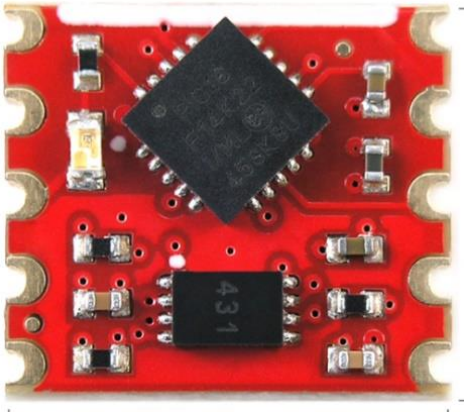


Figura 5: Acondicionador de señal pH OEM Atlas Scientific.
Fuente: Atlas Scientific.

Figura 6: Sonda pH Atlas Scientific.
Fuente: Elaboración propia.

Se fabrica la tarjeta electrónica de manera experimental para realizar pruebas como se observa en la Fig. 7.

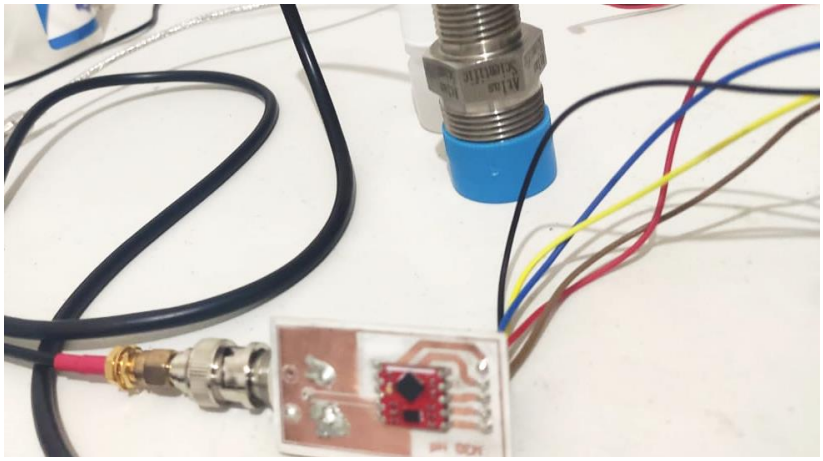


Figura 7: Tarjeta electrónica experimental para el sensor de pH.
Fuente: Elaboración propia.

Se visita el Centro Interdisciplinario de Investigación para el Desarrollo Integral Regional CIIDIR-IPN Unidad Sinaloa (Laboratorio de análisis vegetal), con el fin de evaluar el correcto funcionamiento los sensores. En la Fig. 8 se ilustra la medición de un suelo ácido.

Figura 8: Medición de suelo ácido con sensor Atlas Scientific.
Fuente: Elaboración propia.

De la misma manera se realiza el experimento con el pH-metro modelo HI9813-6 (Hanna Instruments, PD, Italia) ilustrado en la Fig. 8 y se comprueba su funcionamiento. Cabe señalar que, con anterioridad al experimento se hicieron pruebas con soluciones calibradas de diferentes niveles de pH.

Figura 9: pH-metro Hanna Instruments HI 9813-6.
Fuente: Elaboración propia.

Se elige el sensor SHT20 (Sensirion, Stäfa, Suiza) para la medición de temperatura y humedad. Sin embargo, el sensor es un circuito integrado, para realizar mediciones es necesario una tarjeta electrónica y una carcasa. El fabricante DFrobot comercia la solución la cual integra una cubierta a prueba de agua como se muestra en la Fig. 10.

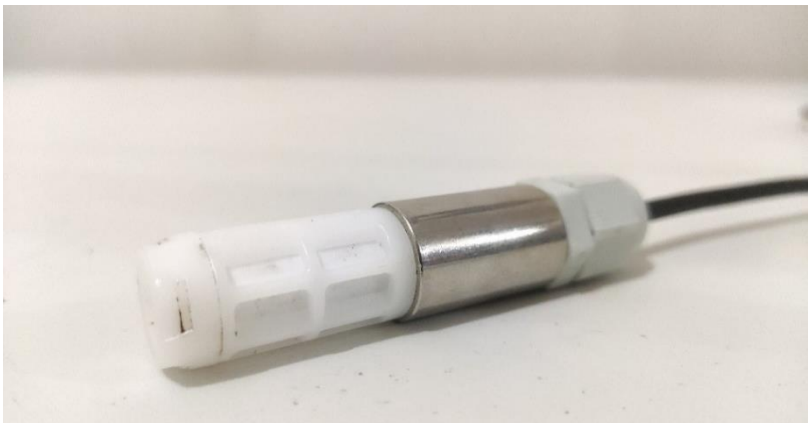


Figura 10: Sensor de temperatura y humedad SHT20 DFrobot.
Fuente: Elaboración propia.

Para la medición de iluminancia se elige el sensor TSL2561 (ams, Premstaetten, Austria). Es importante resaltar, que la iluminancia es la cantidad de flujo luminoso que incide sobre una superficie por unidad de área y su unidad es el lux [29] [30]. También, de la misma manera este sensor tiene el inconveniente de que es un circuito integrado, así que fue necesario fabricar una carcasa y comprar un módulo para su implementación. En la Fig. 11 se observa el sensor de iluminancia TSL2561.

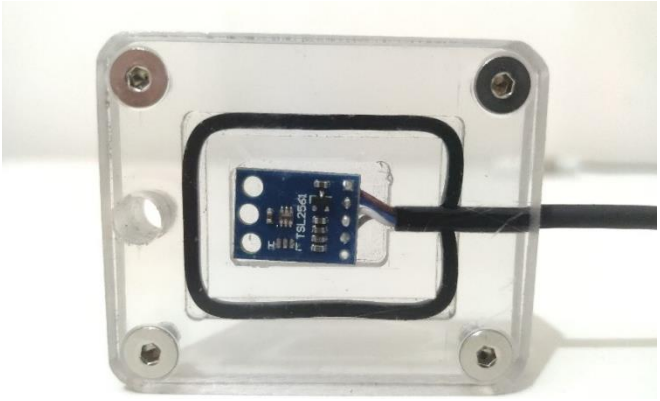


Figura 11: Sensor de iluminancia TSL2561. Fuente: Elaboración propia.

Una vez encontrado todos los sensores requeridos para el sistema embebido ahora se busca los circuitos integrados para la implementación del gestor de energía. Se escoge el circuito integrado MCP73871 (Microchip Technology, AZ, USA) como controlador de carga de batería. Cabe mencionar que este chip indica cuándo la batería está cargada o descarda y cuándo está cargado. En la Fig. 12 se muestra la tarjeta de desarrollo del fabricante Adafruit para realizar la experimentación.

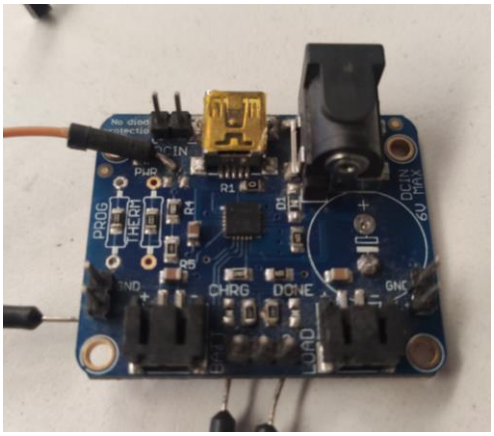


Figura 12: Tarjeta de experimentación MCP73871 de Adafruit. Fuente: Elaboración propia.

Para el monitoreo de la batería se elige el circuito integrado MAX17043G+U (Maxim Integrated, CA, USA) y se utiliza la tarjeta de desarrollo de la empresa Sparkfun para realizar la experimentación. Se muestra en la Fig. 13.

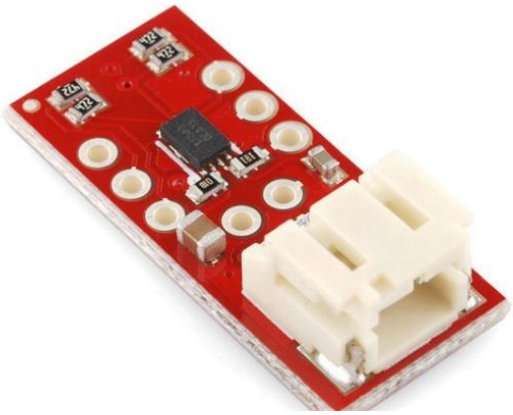


Figura 13: Tarjeta de experimentación MAX17043G+U de Sparkfun. Fuente: Sparkfun.

Todos los componentes del sistema funcionan con una diferencia de potencial de 3.3 V. Para mantener voltaje fijo se integra un convertidor DC-DC Buck-Boost TPS6303 (Texas Instruments, TX, USA). La tarjeta de experimentación para este caso se tuvo que fabricar porque la original del fabricante no es accesible y se muestra en la Fig. 14.

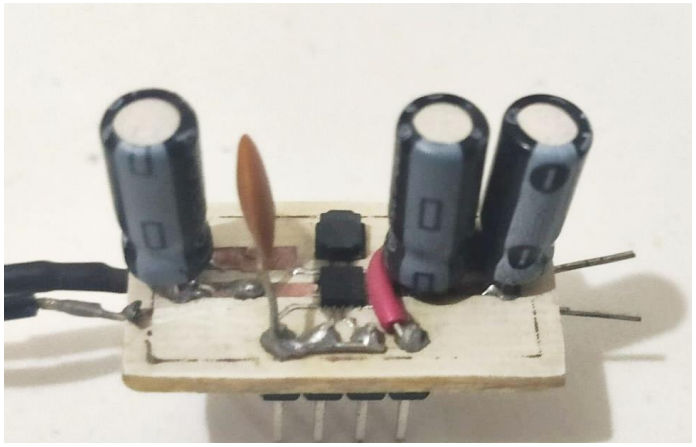


Figura 14: Tarjeta de experimentación Convertidor DC-DC TPS6303.
Fuente: Elaboración propia.

Por último, en la elección de la batería influye en su capacidad y tamaño. Para este caso se escoge la batería recargable tipo LiPo de 6000mAh (Sparkfun, CO, USA) porque tiene una alta capacidad, bajo peso y certificación UL. El modelo de la batería es el DTP605068-3P como se ilustra en la Fig. 15.

Figura 15: Batería tipo LiPo DTP605068-3P Sparkfun.
Fuente: Sparkfun.

2.6 Desarrollo de hardware

En esta sección se desarrolla el hardware del sistema embebido.

2.6.1 Diagrama a bloques de hardware

El hardware está basado en un microcontrolador principal. El microcontrolador se comunica con los demás componentes usando los periféricos integrados. La comunicación con el transceptor LoRa se realiza mediante el protocolo de Interfaz Periférica Serial (SPI, por sus siglas en inglés) y también con puertos de entrada y salida (I/O) de propósito general. Se utiliza el bus comunicación I^2C para los sensores y un puerto Receptor-Transmisor Asíncrono Universal (UART, por sus siglas en inglés) para el manejo del GPS. El gestor de energía se encarga de alimentar todos los componentes del sistema a una diferencia de potencial de 3.3 volts. Además, se encuentra conectado al mismo bus I^2C y también maneja puertos de entrada y salida de propósito general para enviar su estatus. En la Fig. 16 se muestra el diagrama a bloques de hardware utilizado en el sistema embebido.

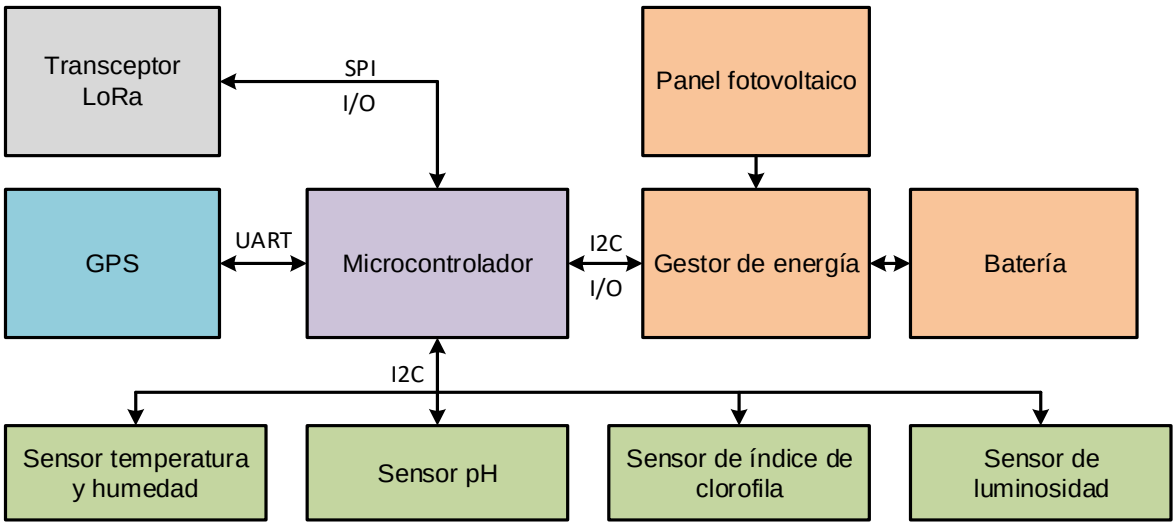


Figura 16: Diagrama a bloques de hardware. Fuente: Elaboración propia.

2.6.2 Diseño electrónico

Posteriormente de definir el diagrama a bloques de hardware se produce el diseño de la placa de circuito impreso (PCB, por sus siglas en inglés) con el software Circuit Maker. Este software es una plataforma en línea de diseño de hardware libre con herramientas de primera categoría. La tarjeta electrónica se compone de dos elementos principales: el diagrama esquemático y el diseño PCB. El diagrama esquemático contiene la conexión entre los componentes, es importante resaltar que para cada componente se desarrolla su librería de diseño siguiendo las especificaciones y recomendaciones de los fabricantes. Del mismo modo el diseño PCB comprende las conexiones a nivel físico entre los componentes, atribuidas en el diagrama esquemático. A continuación, se describe cada circuito del diagrama esquemático de la tarjeta por componente electrónico, el esquemático completo se encuentra en el Anexo I.

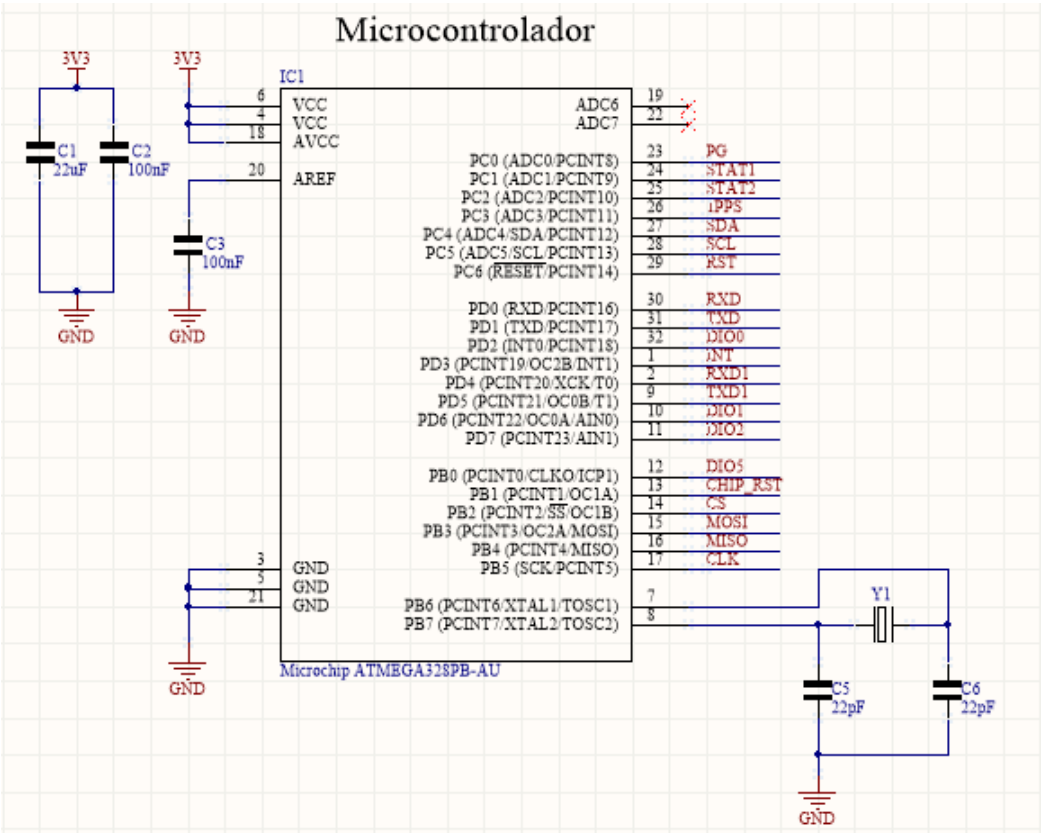


Figura 17: Sistema mínimo del microcontrolador ATmega328P. Fuente: Elaboración propia.

2.6.3 Microcontrolador ATmega328P

El microcontrolador principal es el ATmega328P, la elección de este componente se acredita porque en evaluación alto nivel se utilizó una tarjeta de desarrollo con este microcontrolador que tiene compatibilidad con el entorno de programación y además cuenta con características de bajo consumo y periféricos requeridos para la comunicación con los demás componentes. En la Fig. 17 se presenta el diagrama eléctrico de conexión del sistema mínimo para el microcontrolador utilizado en el esquemático de la tarjeta electrónica. Los componentes requeridos son capacitores y un cristal de cuarzo de 16 MHz [31].

2.6.4 Puerto de programación

Asimismo, el microcontrolador tiene precargado un gestor de arranque (*Bootloader*) que sirve para realizar la descarga del firmware y es ejecutado con el protocolo UART. Por lo tanto, es necesario un convertidor USB-TTL externo. La conexión del puerto de programación se muestra en la Fig. 18. También, en caso de que se necesite un reinicio (*reset*) el diseño cuenta con un puente físico indicado como J1.

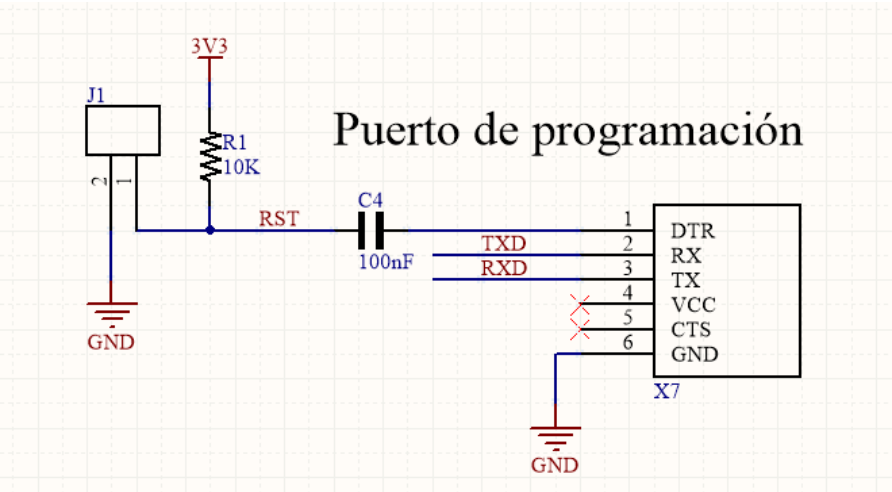


Figura 18: Puerto de programación del microcontrolador. Fuente: Elaboración propia.

2.6.5 Controlador de carga MCP73871T

La alimentación del microcontrolador, los módulos de comunicación y los sensores como se ha mencionado es de una diferencia de potencial de 3.3 volts, a fin de producir este voltaje se emplea un diseño de gestor de energía con 3 circuitos integrados principales: un controlador de carga, un convertidor DC-DC y un indicador de carga de batería. El controlador de carga es el MCP73871T también del fabricante Microchip y se encarga de administrar la carga de la batería usando el panel fotovoltaico. En la Fig. 19 se presenta la conexión del controlador de carga usada en el esquemático de la tarjeta. El controlador cuenta con señales discretas de salida que indican el estado actual de la carga, estas señales están conectadas al microcontrolador principal [32]. También tiene conexión para un sensor termistor de Coeficiente de Temperatura Negativo (NTC, por sus siglas en inglés) situado en la parte inferior de la placa para monitorear la temperatura de la batería y proteger el sistema de un calentamiento no previsto.

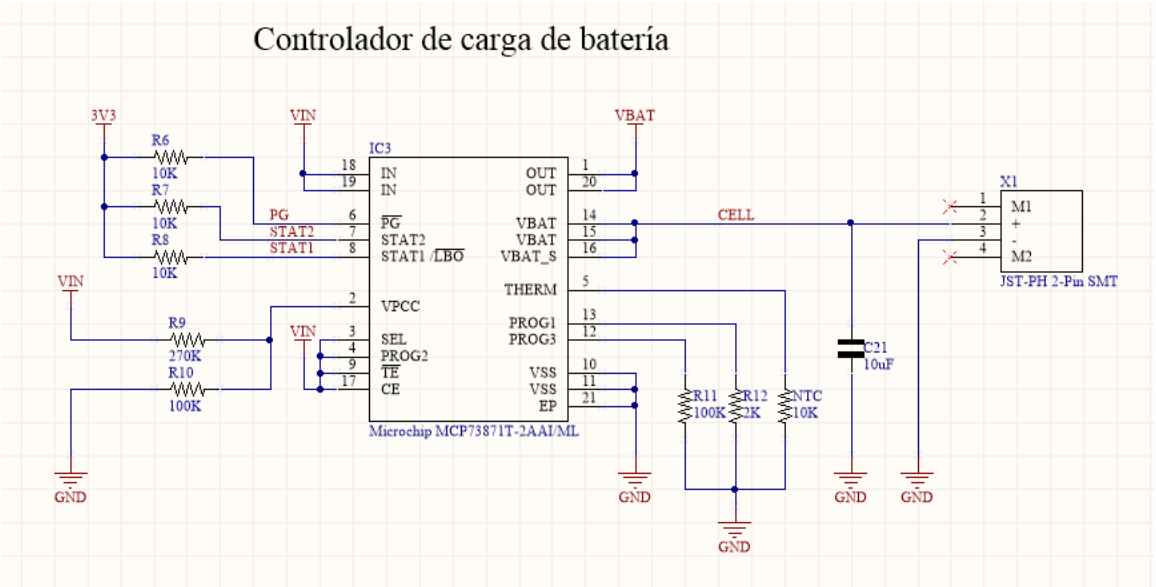


Figura 19: Conexión del controlador de carga. Fuente: Elaboración propia.

2.6.6 Panel fotovoltaico

El sistema embebido incorpora un panel fotovoltaico estándar modelo 2.0W-PANEL (Voltaic Systems, NY, US). Maneja un voltaje de salida de 6 volts con una potencia de 2 watts suficiente para cargar la batería y alimentar el sistema. Las características principales son el tamaño reducido y la resistencia a la humedad. El circuito de acondicionamiento para el panel fotovoltaico posee un diodo Schottky y un diodo normal en serie junto con un capacitor de alta capacidad en paralelo para retener la carga [33], como se muestra en la Fig. 20. La conexión física del panel fotovoltaico es con un bloque terminal de tornillo.

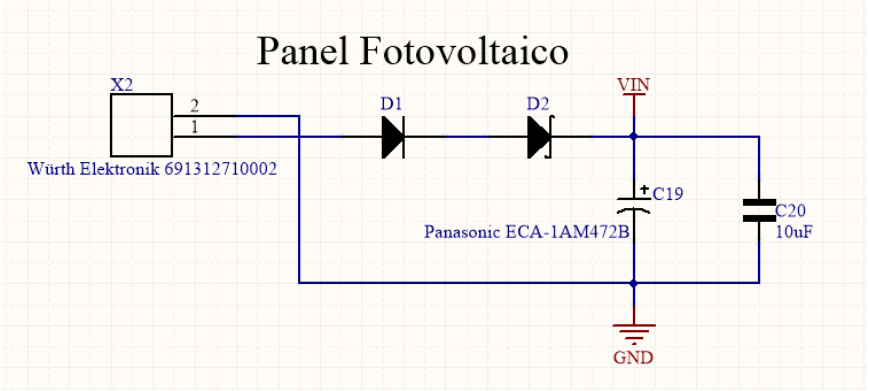


Figura 20: Circuito para panel fotovoltaico. Fuente: Elaboración propia.

2.6.7 Convertidor DC-DC TPS63030

El voltaje constante en el sistema embebido se emplea con un convertidor DC-DC Boost-Buck modelo TPS63030DSKT del fabricante Texas Instruments descrito en la Fig. 21. Su característica principal es la alta eficiencia [34]. La entrada de voltaje proviene del controlador de carga y puede variar desde 3 volts hasta 6 volts. Es necesario un inductor a fin de generar la alta frecuencia y capacitores en la salida para mantener el voltaje continuo.

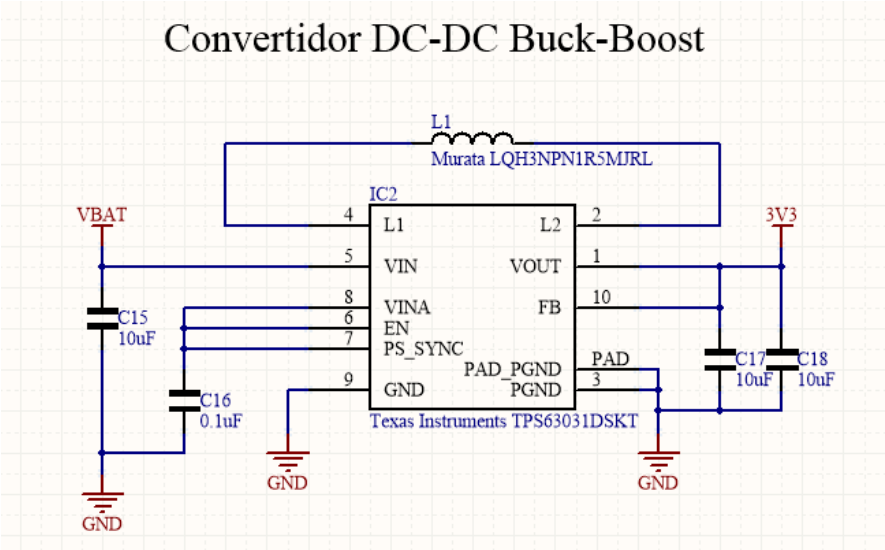


Figura 21: Diagrama eléctrico del convertidor DC-DC. Fuente: Elaboración propia.

2.6.8 Monitor de batería MAX17043

El tercer circuito integrado en la gestión de la energía es el indicador de carga de la batería, el modelo MAX17043G del fabricante Maxim Integrated. Este componente mide el voltaje de la batería y además internamente tiene un algoritmo complejo para calcular el porcentaje de carga de la batería. Los datos son transmitidos usando el bus I^2C del sistema embebido [35]. En la Fig. 22 se presenta el diagrama de conexión del indicador de carga de la batería.

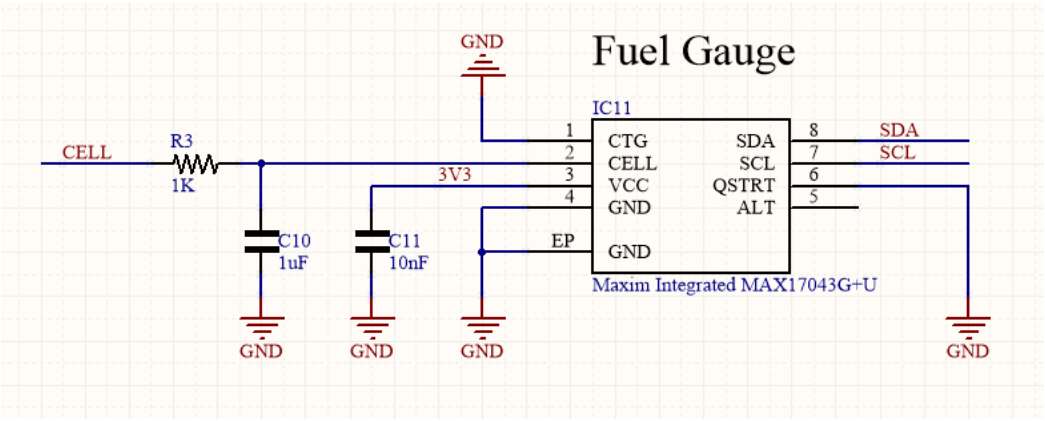


Figura 22: Diagrama de conexión del indicador de carga de la batería. Fuente: Elaboración propia.

2.6.9 Módulo embebido pH

En la medición de pH se usa un módulo embebido pH OEM del fabricante Atlas Scientific, este circuito acondiciona la señal de la sonda y la digitaliza. La comunicación se lleva a cabo usando el bus I^2C [36]. En la Fig. 23 presenta la configuración del módulo, cabe señalar que es el único sensor del sistema que su acondicionamiento se encuentra integrado en la tarjeta electrónica.

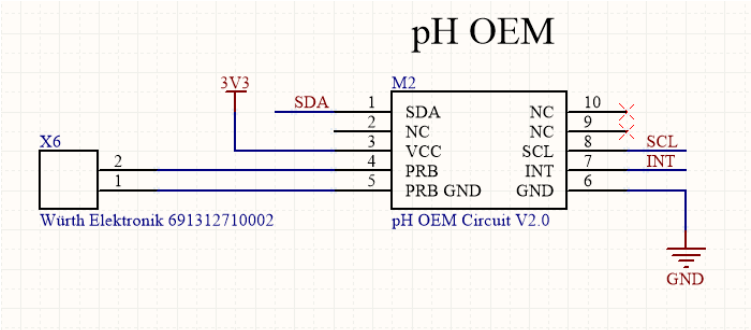


Figura 23: Conexión de módulo pH Atlas Scientific. Fuente: Elaboración propia.

2.6.10 Sensores externos

Los demás sensores son externos y se utilizan bloques terminales para su conexión, la Fig. 24 muestra el circuito de las resistencias de terminación del bus I^2C y la conexión de los sensores: índice de clorofila, iluminancia, temperatura y humedad [23]. El sensor para la determinación de índice de clorofila es el sensor de color RGB del fabricante Atlas Scientific con modelo EZO-RGB.

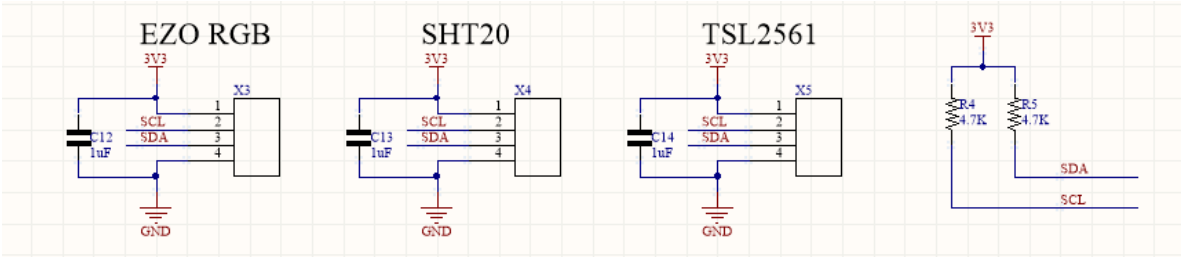


Figura 24: Diagrama de conexión de sensores externos. Fuente: Elaboración propia.

El sensor de iluminancia se compone de un circuito integrado TSL2561 actualmente del fabricante ams. El sensor de temperatura y humedad es un componente distribuido por la compañía DFrobots que integra el sensor SHT20 del fabricante Sensirion.

2.6.11 Transceptor LoRa RFM95W

La transmisión inalámbrica con el protocolo LoRa se emplea con un módulo RFM95W de la compañía HOPERF Microelectronics que integra el transceptor de 915 MHz SX1276 del fabricante Semtech. En la Fig. 25 describe el diagrama eléctrico del módulo y la conexión con el microcontrolador usando el protocolo SPI [37].

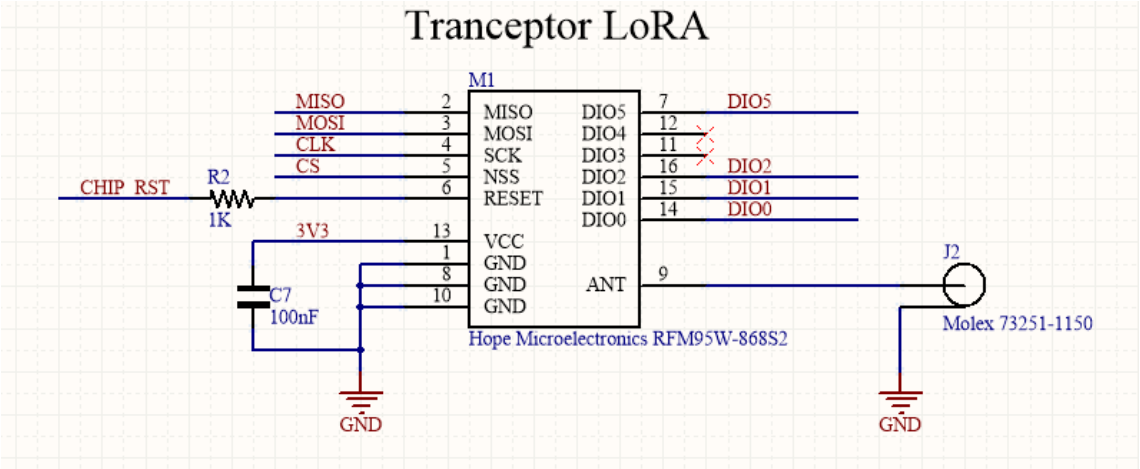


Figura 25: Conexión de módulo LoRA RFM95W.
Fuente: Elaboración propia.

2.6.12 GPS L80

Para el posicionamiento se integra un módulo GPS ultra compacto modelo L80-M39 del fabricante Quectel que cuenta con una antena de cerámica embebida que facilita su incorporación. La Fig. 26 muestra la conexión del módulo, usando el protocolo UART en la comunicación con el microcontrolador [38].

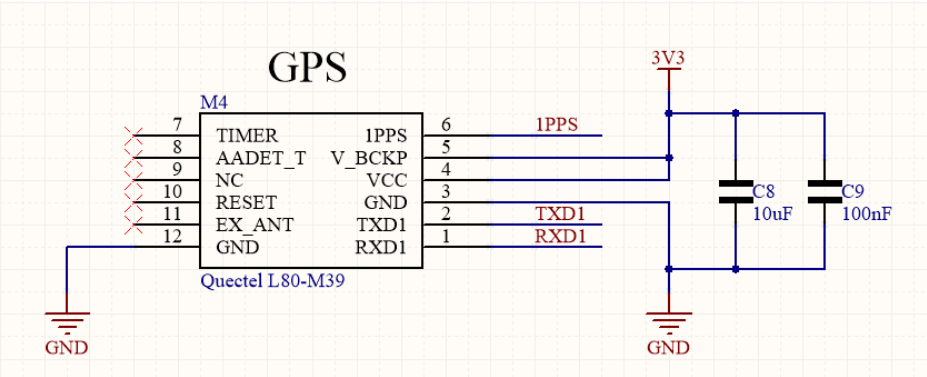


Figura 26: Conexión del GPS L80 Queltec.
Fuente: Elaboración propia.

2.6.13 Diseño PCB

Después de compilar el esquemático del circuito electrónico se realiza el diseño PCB a dos capas. La Fig. 27 muestra la capa superior en color naranja y la capa inferior en color azul. Conviene enfatizar que el diseño sigue la normatividad del estándar IPC (*Institute of Printed Circuits*).

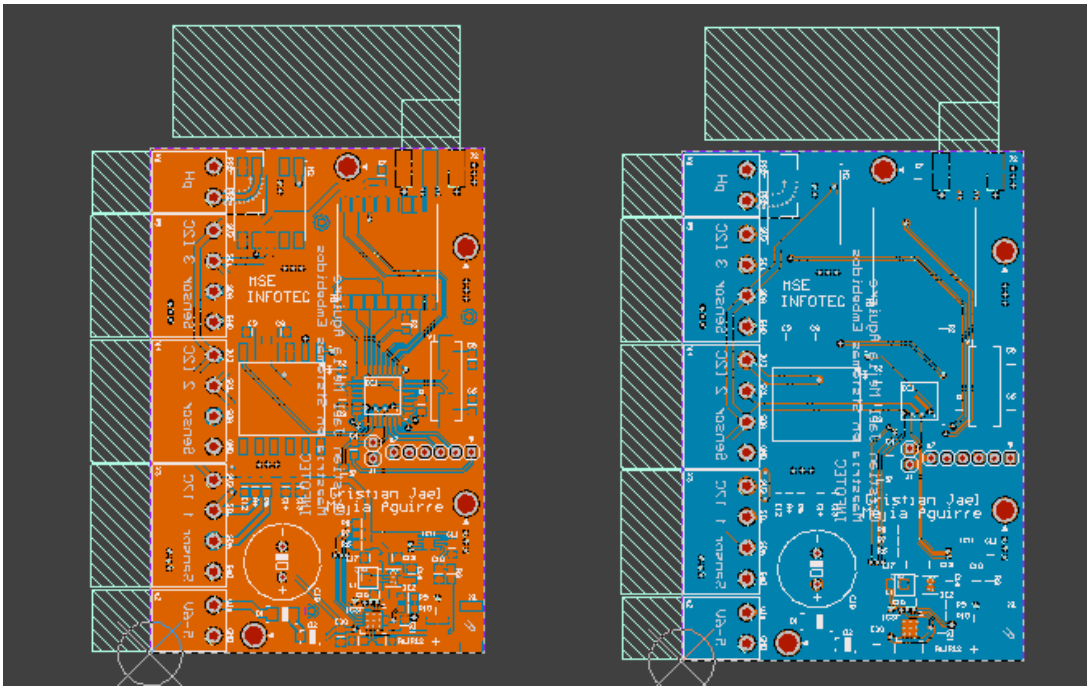


Figura 27: Diseño PCB en dos capas del sistema embebido.
Fuente: Elaboración propia.

El software cuenta con la herramienta de vista 3D con el fin de dimensionar los componentes y ver el modelo físico de la tarjeta electrónica. En las Figs. 28 y 29 se presenta el diseño 3D de la tarjeta electrónica del sistema embebido.

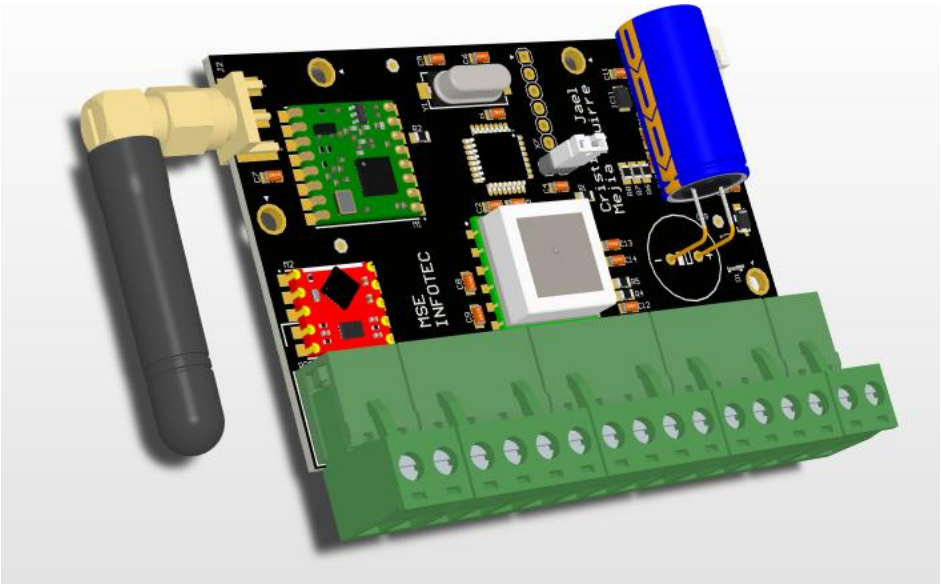


Figura 28: Diseño 3D de la PCB del sistema embebido.
Fuente: Elaboración propia.

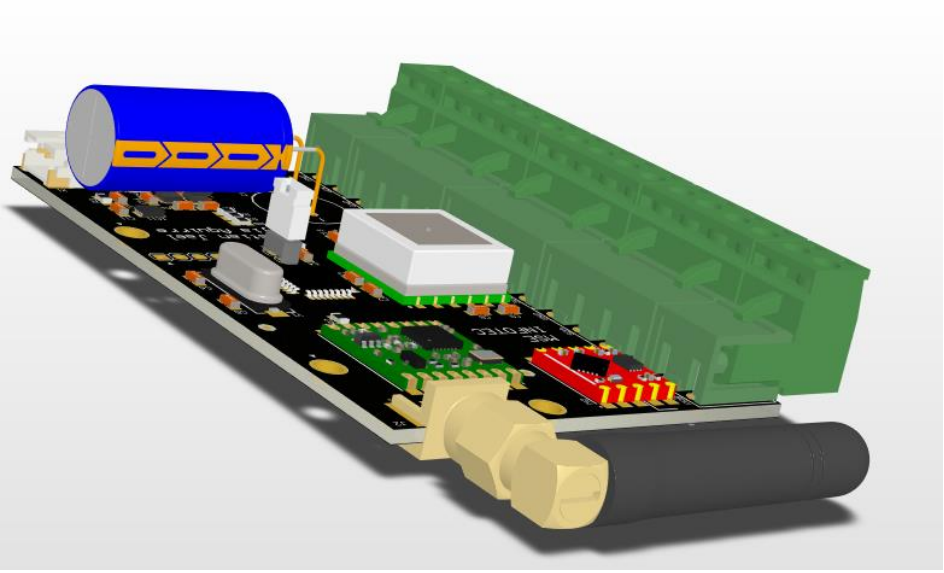


Figura 29: Diseño 3D de la PCB del sistema embebido vista posterior.
Fuente: Elaboración propia.

2.6.14 Prototipado

Con la finalidad de comprobar el circuito electrónico del sistema embebido se arma un prototipo funcional con programas de ejemplo de los fabricantes de esta manera se valida la conexión física de los componentes. En las Figs. 30 y 31 se muestran el prototipo del sistema embebido.

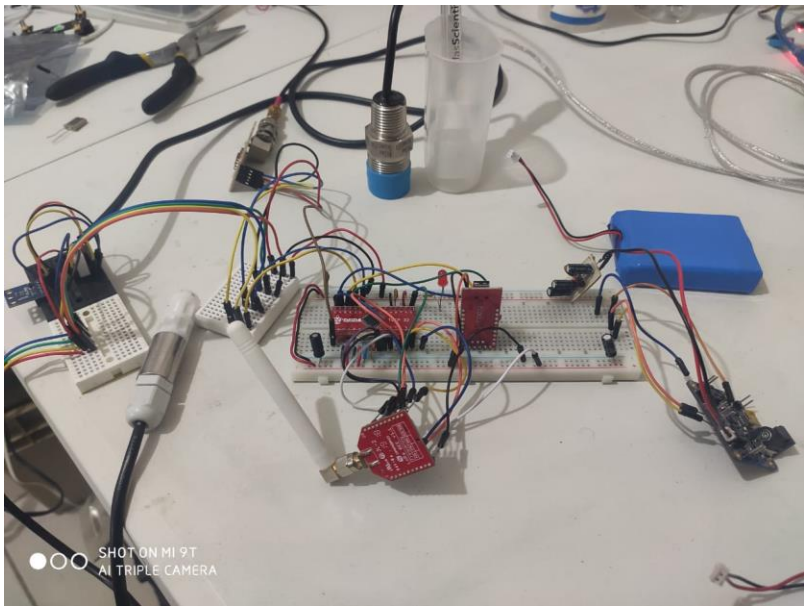


Figura 30: Prototipo funcional del control del sistema embebido. Fuente: Elaboración propia.

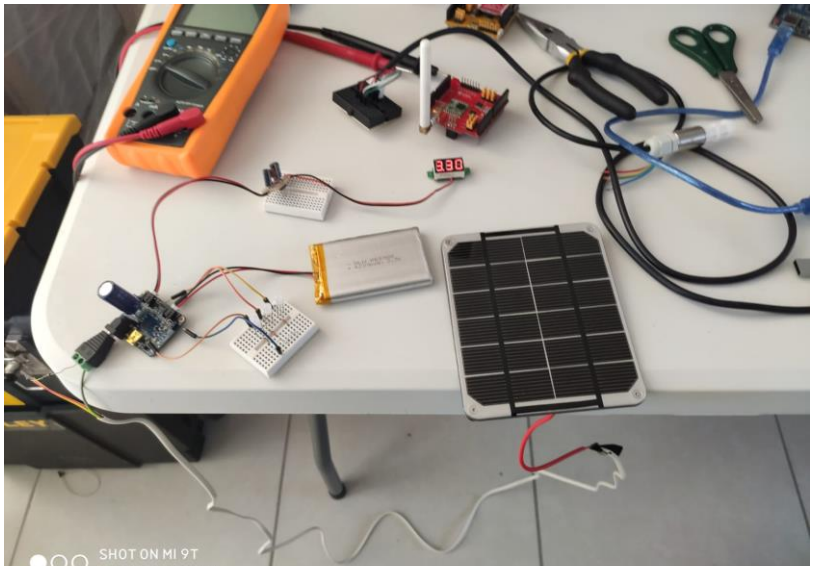


Figura 31: Prototipo funcional de la energía del sistema embebido. Fuente: Elaboración propia.

2.6.15 Fabricación y ensamble

Es necesario resaltar que todos los componentes del sistema embebido y la fabricación de la tarjeta electrónica cuentan con la certificación RoHS (Restriction of Hazardous Substances) que indica que no incluye materiales o residuos peligrosos, por ejemplo, plomo. Por la complejidad de la PCB se manda a fabricar con el proveedor externo Seeedstudio. Las especificaciones de la PCB son las siguientes:

- Material: FR-4 TG130
- Número de capas: 2
- Dimensiones: 55*83 mm
- Espesor: 1.6 mm
- Color: negro
- Acabado: HASL libre de plomo
- Mínimo de máscara para soldar: 0.4 mm
- Peso del cobre: 1 oz
- Mínimo de agujeros pasantes: 0.3 mm
- Ancho del trazo y espaciado: 6/6 mil
- Vías ciegas o enterradas: No

La Fig. 32 muestra la PCB sin el ensamble de los componentes.

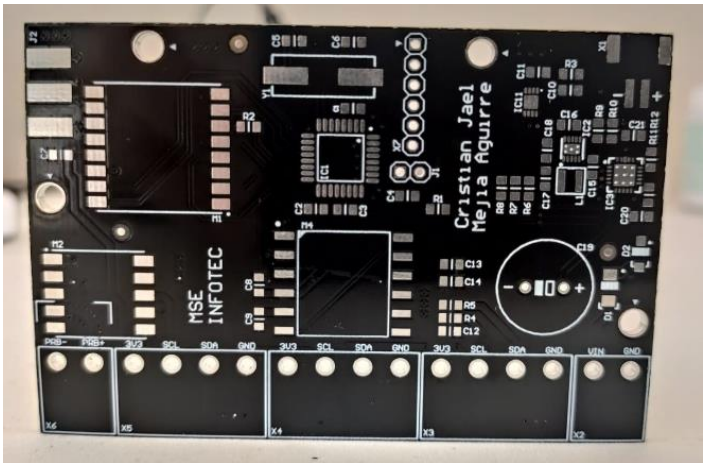


Figura 32: PCB del sistema embebido.
Fuente: Elaboración propia.

Enseguida se realiza el ensamble manual de la tarjeta electrónica con una estación de calor que incorpora un cautín fino y una pistola de aire caliente. En las Figs. 33 y 34 se observa la tarjeta electrónica.

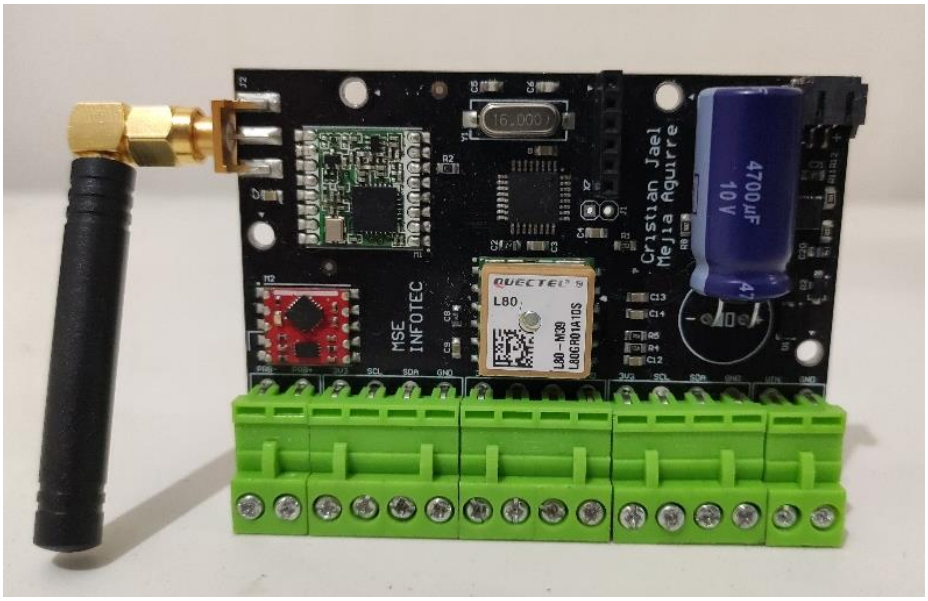


Figura 33: Tarjeta electrónica ensamblada. Fuente: Elaboración propia.

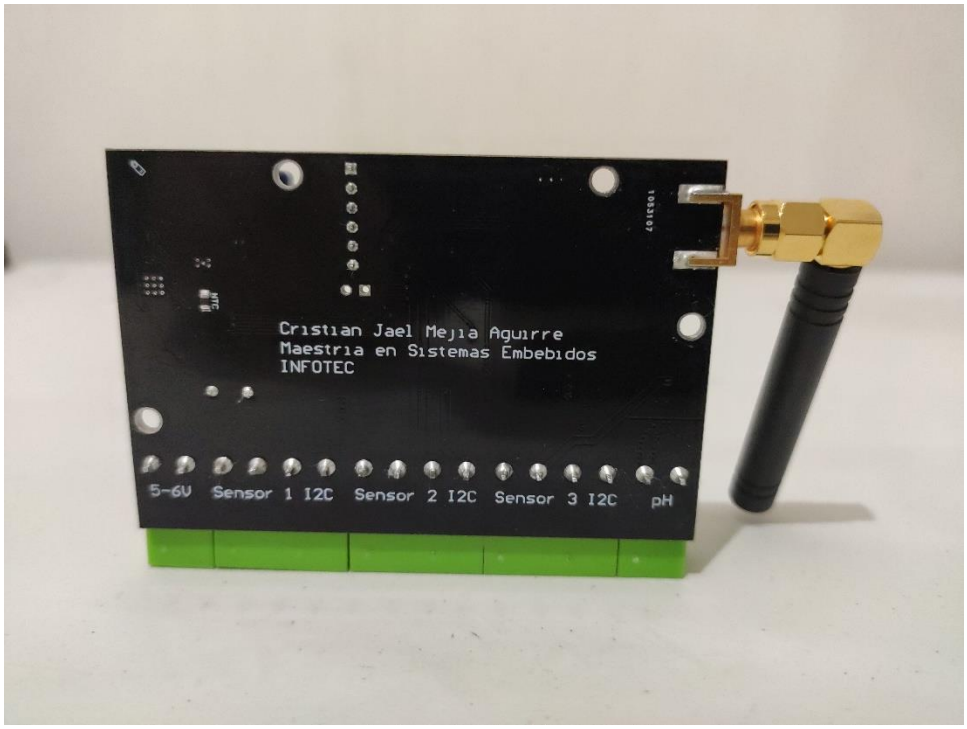


Figura 34: Tarjeta electrónica ensamblada vista posterior. Fuente: Elaboración propia.

Conviene destacar que el diseño de la tarjeta electrónica está disponible en la plataforma CircuitMaker [39] presentada en la Fig. 35.

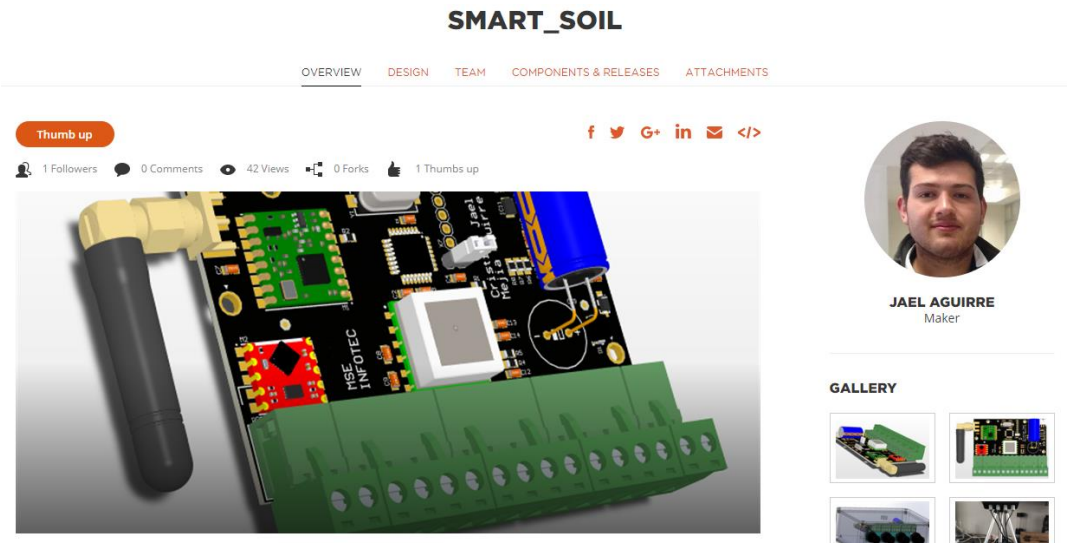


Figura 35: Diseño PCB disponible en la plataforma CircuitMaker. Fuente: Elaboración propia.

2.7 Software embebido

El microcontrolador ATmega328P es el único componente que se programa en el sistema embebido, este programa es llamado *firmware*. Por un lado, el *firmware* está alojado en la memoria de programa tipo *Flash* del microcontrolador. Es preciso señalar que la memoria de programa está seccionada en dos partes. En la primera parte se encuentra el gestor de arranque (*bootloader*) ya mencionado y en la segunda parte la sección de aplicación. Esta última es el código que se ejecuta. Por otro lado, el código se realiza en el Entorno de Desarrollo Integrado (IDE, por sus siglas en inglés) de la plataforma Arduino usando el lenguaje de programación C++. Se ha escogido este entorno dado que los fabricantes de los sensores aportaron las librerías para el manejo de estos. Además, es un software libre de alto nivel popular para el desarrollo de software embebido.

2.7.1 Diagrama a bloques de software

El software indica la estructura e interacción en alto nivel del sistema embebido. Este diseño es una estructura global del funcionamiento y componentes del código fuente. En la Fig. 36 se muestra el diagrama a bloques de software con 4 componentes principales: Configuración inicial, muestra de sensores, envío de datos y modo suspensión. El primer componente configura todos los elementos del sistema embebido antes de iniciar el ciclo repetitivo. El componente de muestra de sensores activa los sensores, realiza la medición y después desactiva los sensores. Luego se realiza la transferencia de datos con el protocolo LoRa. Por último, el sistema entra en un estado de suspensión predefinida por el usuario para el ahorro de energía, este ciclo se repite infinitamente.

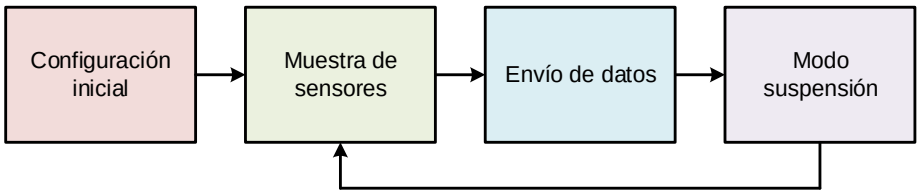


Figura 36: Diagrama a bloques de software. Fuente: Elaboración propia.

2.7.2 Diagrama de flujo

El diagrama de flujo del programa se muestra en la Fig. 37. En el inicio del programa se configura el microcontrolador y los sensores. Además, los sensores y el GPS entran en modo de suspensión. Al iniciar el ciclo repetitivo se activan los sensores para tomar una muestra de cada una de forma consecutiva. Después vuelven a modo de suspensión y se trasmite la información recabada. Una vez verificada la transmisión el microcontrolador entra en modo suspensión un tiempo determinado, por último, activarlo y volver a iniciar con el proceso.

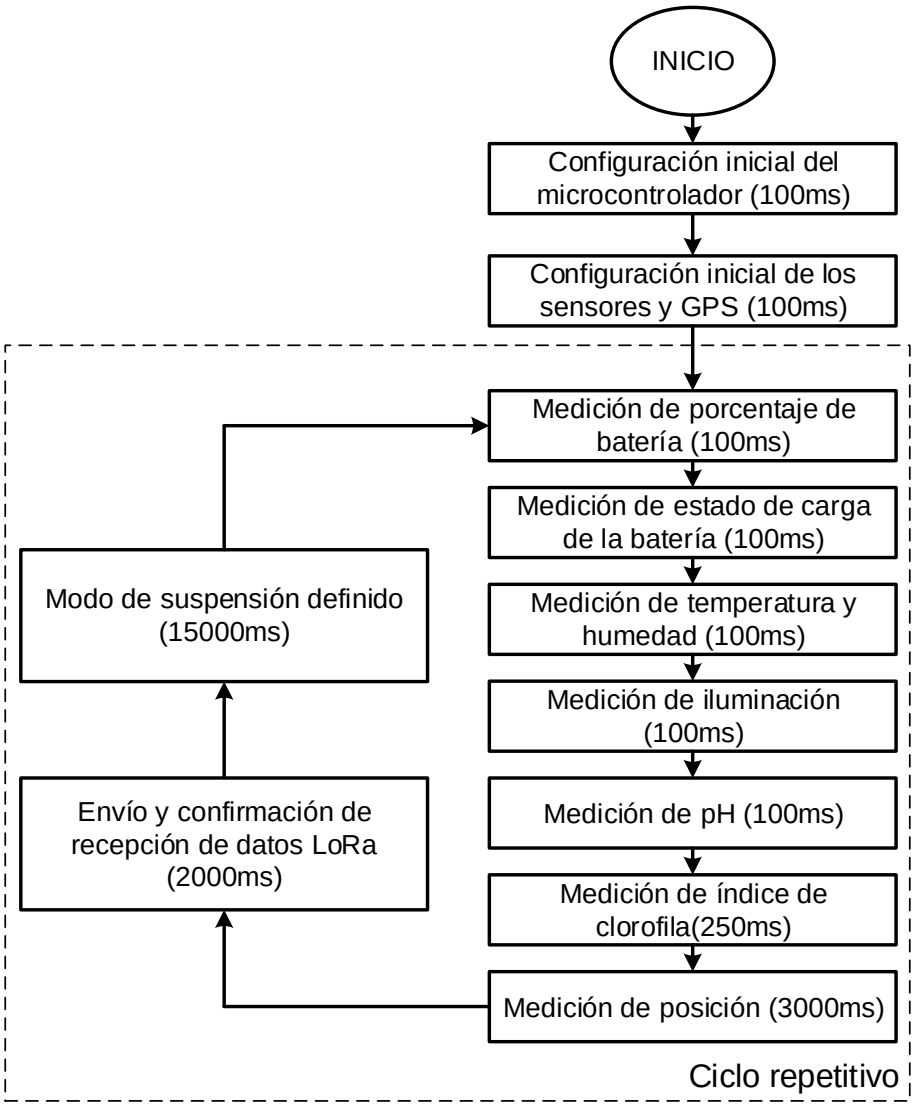


Figura 37: Diagrama del flujo del software embebido. Fuente: Elaboración propia.

2.7.3 Descripción del programa

El código de programación en el IDE Arduino se le llama *Sketch* y tiene una extensión ".ino". El primer paso antes de desarrollar el *Sketch* es incluir las bibliotecas que se necesitan, el IDE cuenta con una herramienta automática para incluirlas. En las siguientes líneas de código se muestra las bibliotecas usadas en el programa.

```
#include <MAX17043.h>                // Monitoreo de batería
#include <Wire.h>                    // Protocolo I2C
#include <DFRobot_SHT20.h>           // Sensor de temperatura y humedad
#include <LowPower.h>                // Modo sleep microcontrolador
#include <Adafruit_Sensor.h>         // Sensor de iluminancia
#include <Adafruit_TSL2561_U.h>      // Sensor de iluminancia
#include <SoftwareSerial.h>          // UART para GPS
#include <TinyGPS.h>                 // Manejo de GPS
#include <SPI.h>                     // SPI para LoRa
#include <RH_RF95.h>                 // Manejo de LoRa
```

En su defecto la estructura de programación tiene dos funciones principales obligatorias: *void setup()* y *void loop().* La primera función se ejecuta una sola vez y se encarga de inicializar y configurar las funciones en el *Sketch*. Posteriormente, en la función *void loop()* se encuentra el programa principal que se repite infinitamente. Por un lado, las siguientes líneas de código muestran la función *void setup(),* en ella se establecen otras funciones creadas para la configuración de los objetos en el *Sketch*.

```
void setup()
{
  Wire.begin();                      // Configuración de protocolo I2C
  MAX17043_CONFIGURACION();          // Configuración del monitor de batería
  MCP73871_CONFIGURACION();          // Configuración del controlador de carga
  SHT20_CONFIGURACION();             // Configuración del sensor de temp & humedad
  TSL2562_CONFIGURACION();           // Configuración del sensor de iluminancia
  ph_CONFIGURACION();                //Configuración del sensor de pH
  EZO_CONFIGURACION();               //Configuración del sensor de clorofila
  L80_CONFIGURACION();               //Configuración del GPS
  LORA_CONFIGURACION();              //Configuración del módulo LoRa
}
```

La función *Wire.begin()* establece el protocolo I^2C. La *MCP73871_CONFIGURACION()* configura las entradas digitales para medir el

estado de carga. La función *SHT20_CONFIGURACION()* inicializa el sensor SHT20 de temperatura y humedad. Igualmente, la función *TSL2562_CONFIGURACION()* inicializa el sensor de iluminancia. Después, la función *ph_CONFIGURACION()* configura el módulo pH OEM. Posteriormente, la función *EZO_CONFIGURACION()* establece la configuración inicial del sensor de índice de clorofila. A continuación, el GPS L80 se inicializa poniéndolo en modo suspensión. Por último, se confirma el módulo LoRa.

Por otra lado, la función *void loop()* tiene las funciones de muestra para cada sensor y el GPS. Además, se encuentra la función de envío de datos por LoRa y el modo de suspensión “sleep” que pone hibernación el microcontrolador. En las siguientes líneas de código se muestra la función *void loop().*

```
void loop()
{
 MAX17043_MUESTRA();        // Monitoreo de batería (100ms)
 MCP73871_MUESTRA();        // Estado de carga (100ms)
 SHT20_MUESTRA();           // Medición de temperatura y humedad (100ms)
 TSL2561_MUESTRA();         // Medición de iluminancia (100ms)
 ph_MUESTRA();              // Medición de pH (100ms)
 EZO_MUESTRA();             // Medición de clorofila (250ms)
 L80_MUESTRA();             // Posición GPS (3000ms)
 Enviar_LoRa();             // Envío de datos LoRa (2000ms)
 sleep(15);                 // Esperar 15 segundos en suspensión (15000ms)
}
```

El *Sketch* completo con la descripción de todas las funciones se encuentra en el Anexo III. Además, está disponible en la plataforma web de Arduino Project Hub [40] como se muestra en la Fig. 38.

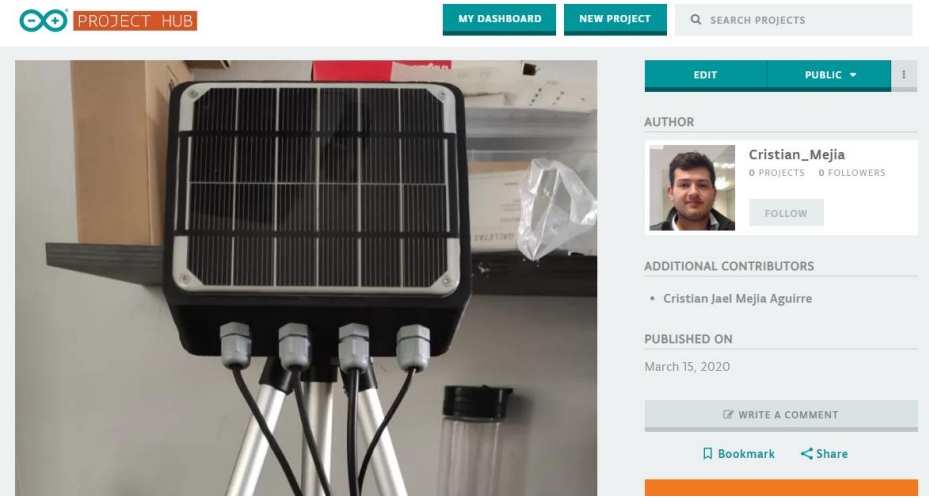


Figura 38: Sketch disponible en Project Hub. Fuente: Elaboración propia.

2.8 Configuración de Gateway y plataforma web

La implementación de la red inalámbrica de sensores utiliza un gateway modelo LG01-P del fabricante Dragino, el cual se muestra en la Fig. 39. El gateway se conecta a una red cableada *Ethernet* para enlazarlo a *Internet*. También, se configura para recibir datos del sistema embebido y publicar los datos en la plataforma web ThingSpeak.

Figura 39: Gateway LG01-P Dragino. Fuente: Elaboración propia.

ThingSpeak, es una plataforma IoT permite agregar, visualizar y analizar flujos de datos en vivo. Además, tiene la capacidad de ejecutar código *MATLAB*, puede realizar analítica y procesamiento en línea a medida que los datos ingresan [41]. Sin embargo, el alcance del proyecto se limita en la construcción del sistema embebido. Para fines de verificación y pruebas se realiza una demostración usando la plataforma, la Fig. 40 presenta los datos recibidos de la temperatura y humedad, cabe mencionar que el historial de los datos se puede exportar a formatos: JSON, XML y CSV.

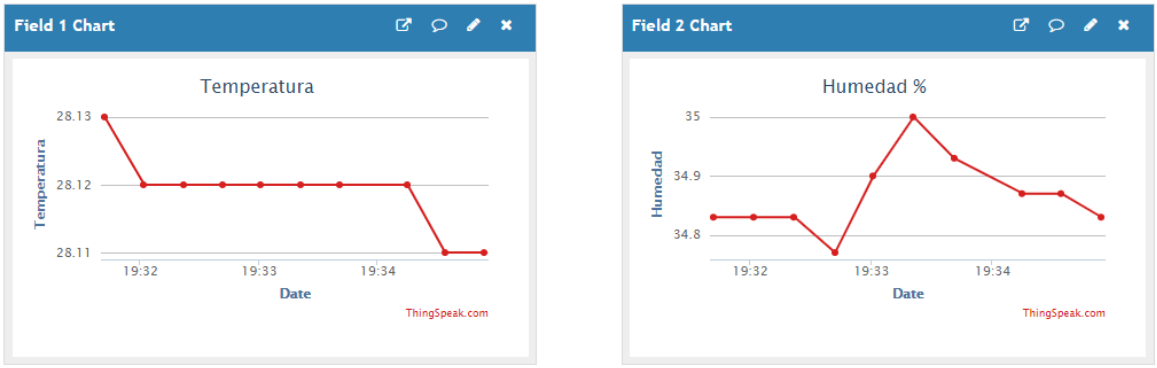


Figura 40: Datos de temperatura y humedad. Fuente: Elaboración propia.

La plataforma ThingSpeak soporta muchos dispositivos a la vez y son llamados canales. Cada canal tiene un límite de ocho campos (*fields*). La temperatura y humedad están en los campos 1 y 2 respectivamente como se presentó anteriormente. En los campos 3 y 4 se encuentran el pH y la iluminancia como se observa en la Fig. 41. El porcentaje de carga de la batería e índice de clorofila están los campos 5 y 6, en la Fig. 42 se muestra. En la Fig. 43 se muestra el GPS y el estado de la carga que pertenecen a los campos 7 y 8.

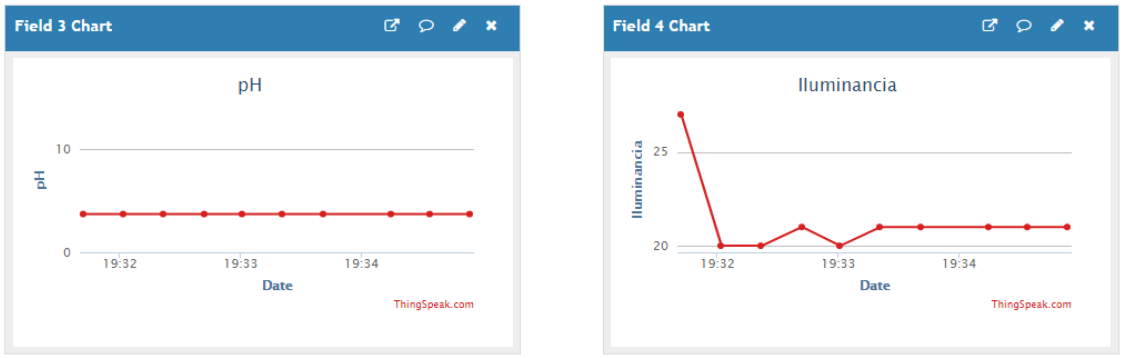


Figura 43: Datos de nivel de pH e iluminancia (lux). Fuente: Elaboración propia.

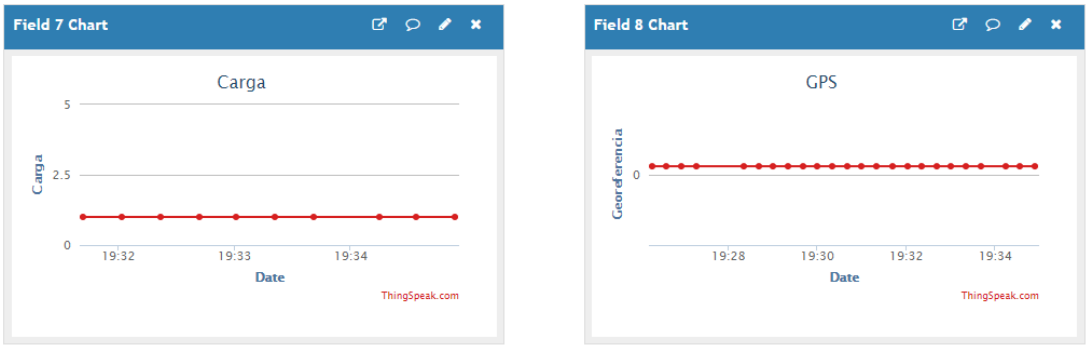


Figura 42: Datos del estado de la carga y el GPS (latitud y longitud). Fuente: Elaboración propia.

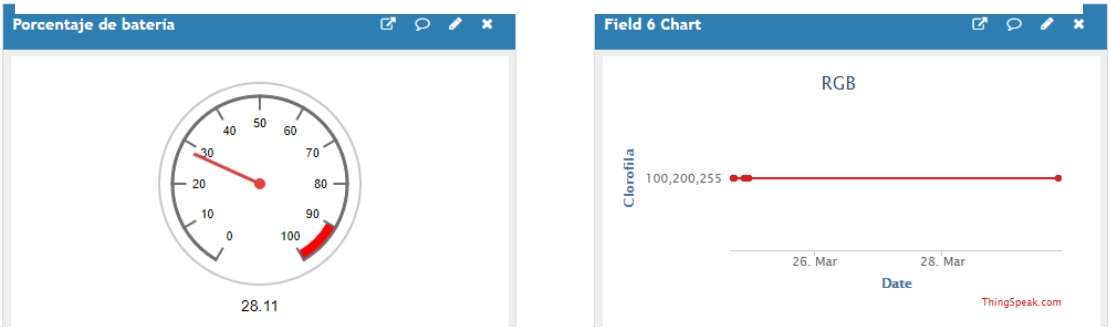


Figura 41: Datos de porcentaje de batería e índice clorofila (RGB-8bits). Fuente: Elaboración propia.

2.9 Diseño y construcción de la carcasa

A fin de cumplir con el requerimiento de grado de protección IP66 (*Ingress Protection*) [42], que refiere a la protección completa contra polvo y chorros de agua se diseña la carcasa usando el software SolidWorks. Para dimensionar la carcasa primero se exporta el diseño 3D de la tarjeta electrónica en formato *STEP* con el software CircuitMaker. En las Figs. 44, 45, 46 y 47 se ilustra el diseño 3D de la carcasa. Además, se incluye un conector glándula PG7 para cada cable de los sensores con el objetivo de mantener hermético el dispositivo. Las dimensiones de la carcasa son 148x122x60mm.

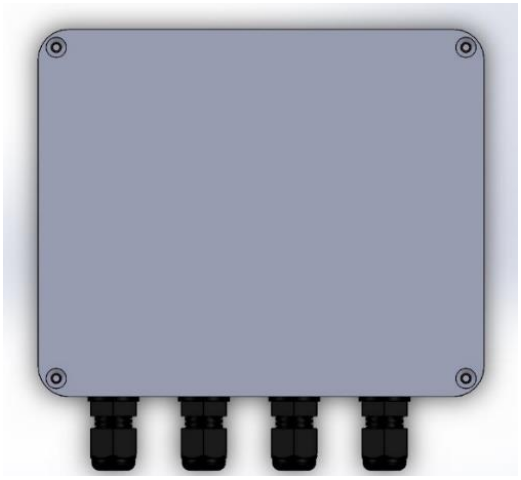


Figura 47: Diseño de carcasa 3D vista frontal. Fuente: Elaboración propia.

Figura 46: Diseño de carcasa 3D vista de posterior. Fuente: Elaboración propia.

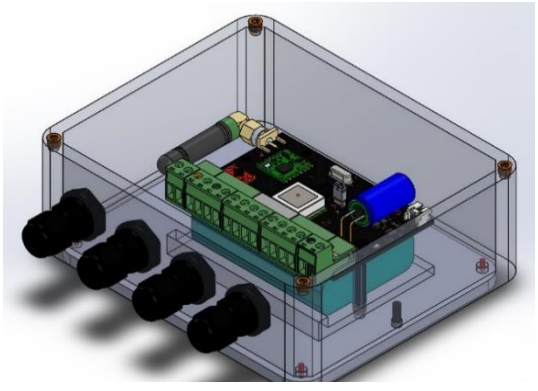


Figura 45: Diseño de carcasa en 3D vista de perfil. Fuente: Elaboración propia.

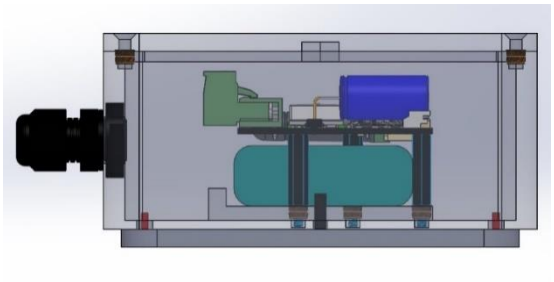


Figura 44: Diseño de carcasa en 3D vista lateral. Fuente: Elaboración propia.

El material usado en la carcasa es PLA, este polímero está constituido por ácido láctico que se obtiene del maíz y de la caña de azúcar, por este motivo es un material biodegradable en ciertas condiciones de temperatura. La construcción de la carcasa se lleva a cabo con una impresora 3D, la fijación de tornillos para la tapa y la tarjeta se hace mediante insertos de metal de 3mm. En la Fig. 48 se observa la carcasa impresa.

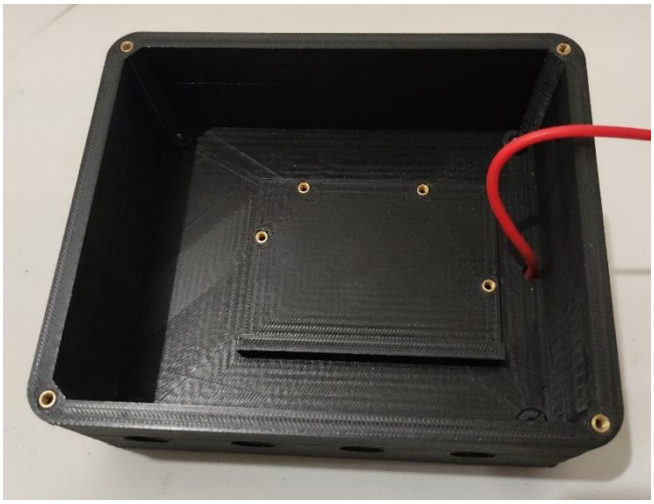


Figura 48: Carcasa impresa en PLA. Fuente: Elaboración propia.

La sujeción de la tarjeta electrónica con la carcasa se emplea con espaciadores (*standoff*) de nylon con el objetivo de que la batería quede sujeta en la parte inferior como se ilustra en la Fig. 49.

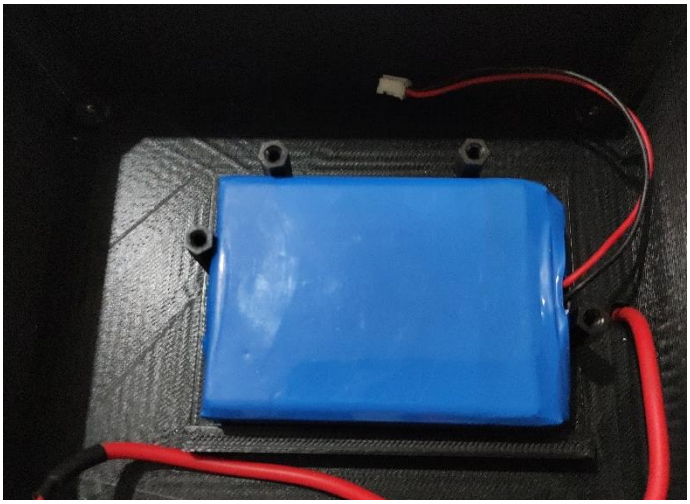


Figura 49: Espaciadores de nylon para la sujeción de la tarjeta.
Fuente: Elaboración propia.

Se realiza el montaje de la tarjeta electrónica, la conexión de los sensores como se ilustra en la Fig. 50.

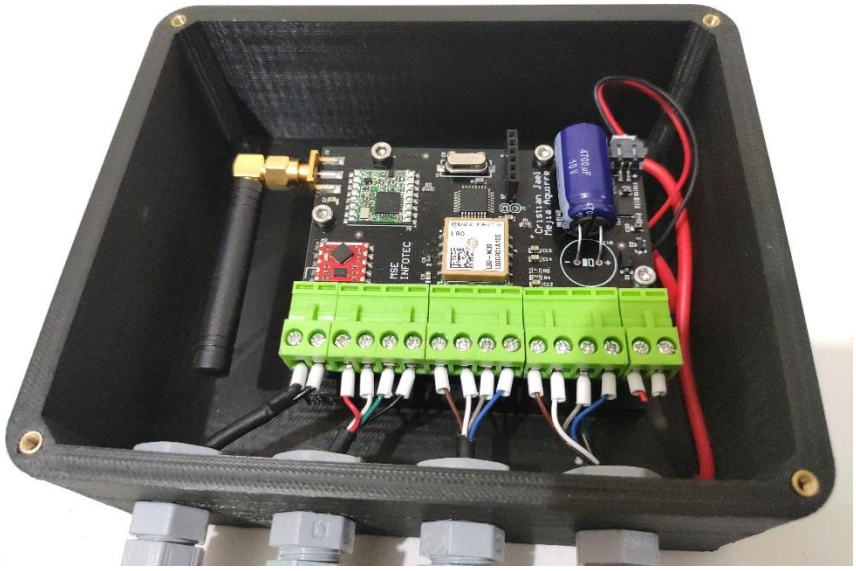


Figura 50: Tarjeta electrónica fija en la carcasa.
Fuente: Elaboración propia.

En la Fig. 51 se muestra el sistema embebido.

Figura 51: Imagen del sistema embebido. Fuente: Elaboración propia.

Además, la carcasa tiene la practicidad de colocar un tornillo en el medio de la tapa con el fin de soportar el dispositivo usando un trípode que maneja el estándar de cuerda de un cuarto de pulgada. Las Fig. 52 muestra el dispositivo soportado y acabado final.

Figura 52: Dispositivo soportado y acabado final. Fuente: Elaboración propia.

En la Fig. 53 se ilustra el sistema embebido realizando medición a una acelga.

Figura 53: Medición de variables físicas a acelga. Fuente: Elaboración propia.

2.10 Arquitectura de red

Las características principales de la red LoRa es el largo alcance, bajo consumo de energía, baja transferencia de datos, seguridad, fácil integración y bajo costo [12]. El sistema embebido tiene la capacidad de integrarse a una red inalámbrica de sensores (WSN) donde uno o más dispositivos finales (nodos) se conectan a uno o más *gateways* LoRA. Los *gateways* enlazan la información de la red a un servidor, como es el caso de ThingSpeak. Las aplicaciones se utilizan para visualizar la información del sistema. En la Fig. 54 se muestra la arquitectura de red del sistema embebido.

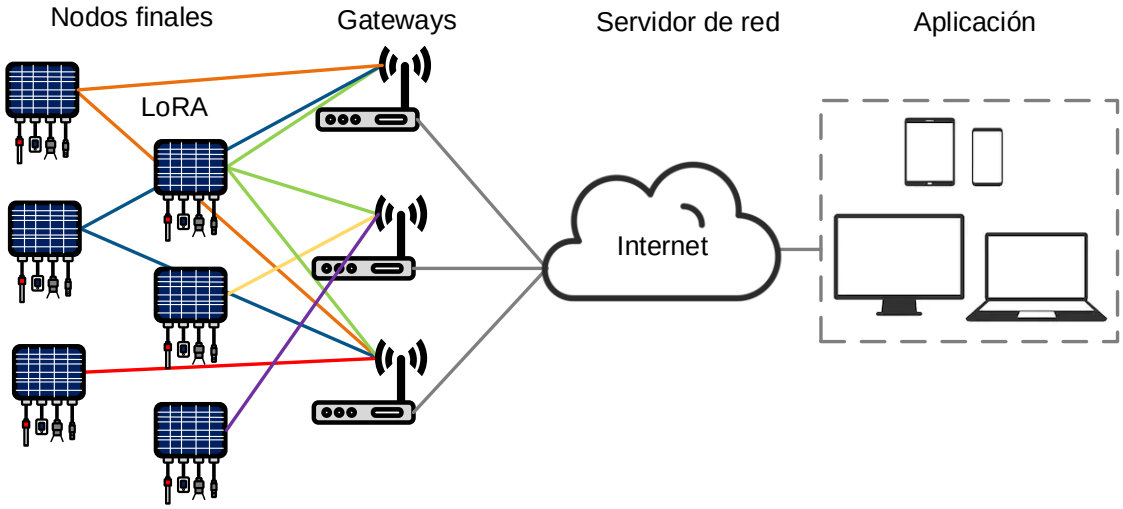


Figura 54: Arquitectura de red. Fuente: Elaboración propia.

2.11 Operación del sistema embebido

En esta sección se describe el manejo del sistema embebido, es decir, la forma en que se opera e instala. Aunque el dispositivo no tiene interfaz de usuario en él, se documenta el uso debido de los sensores.

2.11.1 Medición de pH

La medición de pH se realiza en el suelo y se muestra de forma detallada en la Fig. 55. Las instrucciones se siguen por recomendación del fabricante. El ciclo de vida de una sonda es de 10 años y cada 6 meses es necesario una recalibración usando soluciones definidas. La medición de pH se puede visualizar desde la plataforma IoT con unidades de 0-14 niveles.

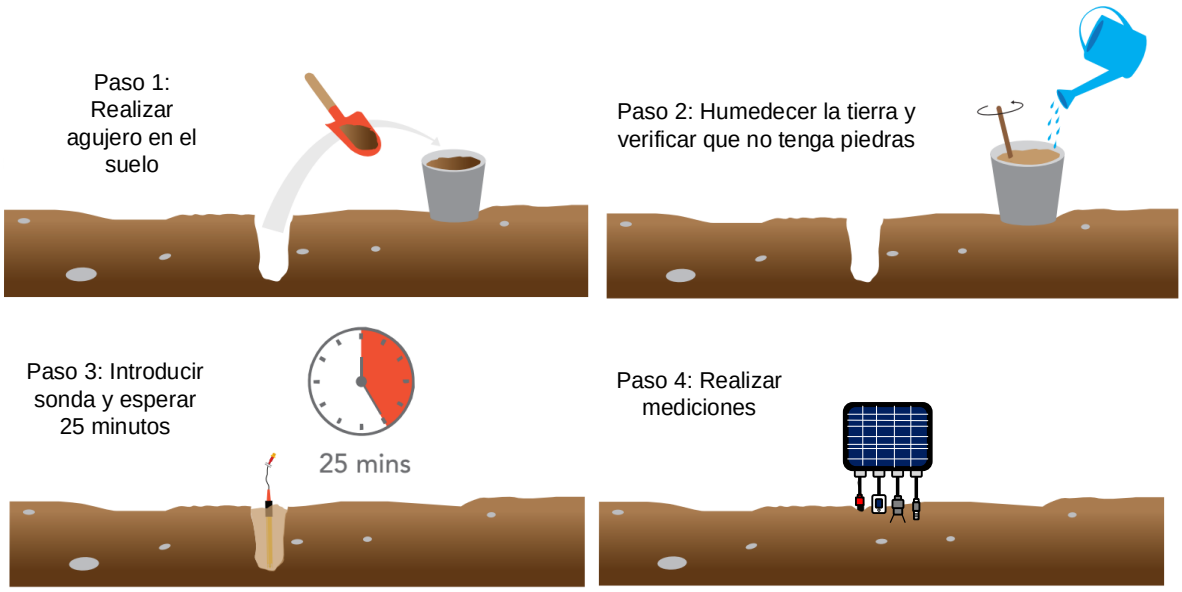


Figura 55: Medición de pH del suelo. Fuente: Elaboración propia.

2.11.2 Medición del índice de clorofila

El índice de clorofila se utiliza como método indirecto para determinar el estado nutricional de la planta y tomar decisiones en la administración de fertilizantes nitrogenados en zonas específicas del cultivo [43]. La medición del índice de clorofila se realiza en un rango de 2cm a 36cm apuntando al elemento a medir. En la Figura 56 se presenta la forma correcta de realizar la medición. La medición del índice de clorofila consiste en apuntar a la planta con la distancia correcta para obtener el valor, como se muestra en la Fig. 57. Igualmente, se puede visualizar la medición desde la plataforma IoT en unidades de RGB-8bits (0-255, 0-255, 0-255).

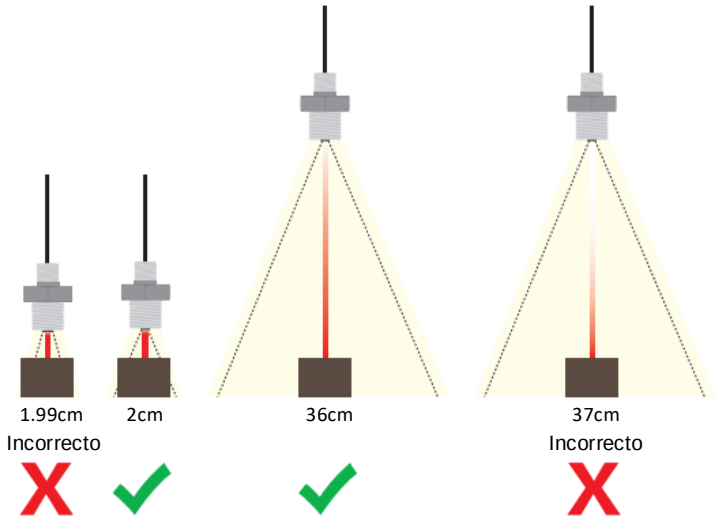


Figura 56: Rango de medición del sensor de clorofila. Fuente: Elaboración propia.

Figura 57: Medición de índice de clorofila. Fuente: Elaboración propia.

2.11.3 Medición de temperatura y humedad

La medición de temperatura y humedad de suelo es sencilla, la Fig. 58 describe los pasos a seguir para realizar una medición correcta. Es importante señalar que el sensor debe de estar seco antes de introducirlo al suelo. La manera más sencilla de secarlo es dejar el sensor expuesto a la radiación solar durante al menos dos horas. La medición se puede visualizar desde la plataforma IoT, en unidades de grados centígrados para la temperatura y en porcentaje de 0-100 para la humedad relativa.

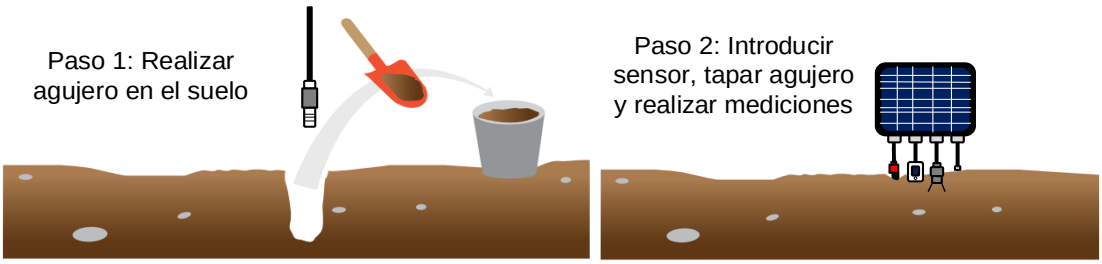


Figura 58: Medición de temperatura y humedad. Fuente: Elaboración propia.

2.11.4 Medición de iluminancia

La instalación del sensor de iluminancia es simple, solo se coloca en cualquier lugar despejado para que la radiación del sol lo excite. También tiene un orificio para colocar una rama y sumergirlo en la tierra como se ilustra en la Fig. 59. La unidad esperada en la plataforma es el lux.

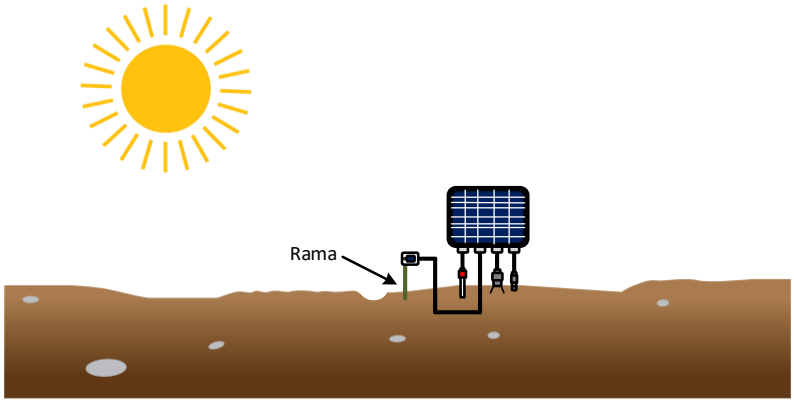


Figura 59: Medición de iluminancia. Fuente: Elaboración propia.

2.12 Análisis de costos

Los materiales para la construcción del sistema embebido suman un total de $322.10 dólares de acuerdo con una búsqueda de cotizaciones realizada en el año 2020, en el Anexo II se exhibe de manera detallada. Es importante destacar que el tipo de cambio fue seleccionado porque la mayoría de los componentes se compraron con proveedores extranjeros. Sin embargo, el costo de producción se puede reducir en aproximadamente 35% si se construyen cien o más unidades. La comparación de costos con otros dispositivos es complicada puesto que es un producto recientemente lanzado al mercado y existen variantes en las especificaciones y arquitecturas de cada uno. En el Cuadro 3, se muestra la comparación de costos contra otras soluciones de características similares desarrolladas en otros países. Se observa que el dispositivo se mantiene en un costo menor a su competencia, aunque el de dispositivo de la compañía Libelium tenga mejores características su costo es más elevado.

Dispositivo	Costo	Especificaciones
Smart QUBIX (Nederland)	3,080.00 €	Comunicación Zigbee - Alcance 75m Sensor CO2 Sensor de acidez o alcalinidad (pH) Sensor de conductividad Sensor de humedad de suelo Sensor de temperatura de suelo
Libelium Smart Agriculture (España)	4,100.00 €	Comunicación LoRa - Alcance 10 Km Sensor de humedad de suelo Sensor de radiación solar Sensor de temperatura de suelo Sensor de humedad de la hoja Sensor de presión atmosférica Sensor Anemómetro, veleta, pluviómetro GPS Panel solar
Smart Soil Plotchip	$322.10 USD	Comunicación LoRa - Alcance 10 Km Sensor de temperatura y humedad de suelo Sensor de acidez o alcalinidad (pH) Sensor de iluminancia Sensor de índice de clorofila GPS Panel solar

Cuadro 3: Comparación de costos. Fuente: Elaboración propia.

Capítulo 3

Resultados y discusión

Capítulo 3. Resultados y discusión

Este capítulo presenta los logros obtenidos en la experimentación e implementación del sistema embebido. Es importante destacar que se consiguió la integración satisfactoria del sistema embebido y que se cumplieron los requerimientos propuestos para el sistema.

3.1 Determinación analítica del pH de diferentes tipos de suelo

En el análisis de pH, se seleccionaron muestras de suelo previamente caracterizadas con el medidor de pH comercial marca Hanna Instruments modelo HI9813-6 utilizado en laboratorio de análisis de suelo. Las muestras provenían de 3 tipos de suelo y se encontraban dentro de los rangos ácido (3.4), neutro (7.1) y básico (9.2), de acuerdo con los valores arrojados por el medidor de pH de laboratorio. Estas muestras procedieron a analizarse por triplicado con el sensor de pH integrado al sistema embebido previa calibración con las soluciones amortiguadas. Es importante resaltar, que la temperatura infiere en la medición de pH, por este motivo el sistema embebido realiza la corrección automática de la compensación de temperatura con el mismo sensor de temperatura que integra, de esta forma obtener una medición más precisa.

El contraste de los resultados con el medidor de pH del dispositivo arrojó valores adecuados con respecto al medidor pH de laboratorio, con variación de +- 0.3 (suelo ácido), +- 0.2 (suelo neutro) y +-0.3 (suelo básico). En general de acuerdo con los expertos del laboratorio de análisis de suelo los resultados arrojados por el sistema embebido son adecuados para la toma de decisiones en el manejo agrícola. No obstante, las pruebas de medición son más funcionales que cuantitativas, ya que el fabricante del sensor afirma la precisión de la medición [36].

3.2 Determinación de índice de clorofila

El índice de clorofila se representa con 3 valores obtenidos de los colores RGB (*Red-Green-Blue*) a 8 bits de resolución, es decir, tenemos una escala de 0 a 255 para cada color. El método de prueba de manera general se realiza en las hojas o tallo de la planta.

Para generar la escala del índice de clorofila se seleccionaron los rangos de valores en RGB correspondientes a muestras de cultivo de frijol diferencialmente administrados con fertilizante nitrogenado. En este sentido las dosis administradas al cultivo se dividieron en 3 clases: mala, regular e ideal. La clase mala no tiene fertilización, la clase regular posee fertilización media y la clase ideal cuenta fertilización completa. En la Cuadro 4 se observa las mediciones realizadas en las diferentes clases.

Clase	Medición 1	Medición 2	Medición 3
Ideal	27,32,25 (R,G,B)	27,33,21 (R,G,B)	25,31,20 (R,G,B)
Regular	33,42,26 (R,G,B)	32,40,25 (R,G,B)	36,45,28 (R,G,B)
Mala	64,72,40 (R,G,B)	68,76,42 (R,G,B)	66,74,41 (R,G,B)

Cuadro 4: Medición de índice de clorofila en clases diferenciadas. Fuente: Elaboración propia.

En la Fig. 60 se ilustra las hojas de diferentes clases, la ideal, la regular y mala respectivamente.

Figura 60: Hojas de frijol de diferentes clases. Fuente: Elaboración propia.

De la tabla anterior se puede observar que de manera cualitativa los valores de RGB, (sobre todo el canal Green), que contribuyen en gran medida para detectar

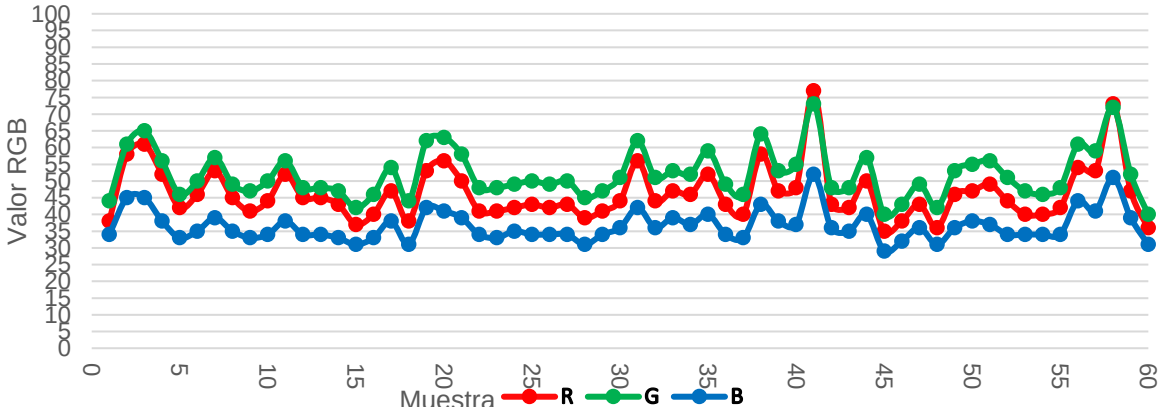


Gráfico 1: índice de clorofila sin fertilizante. Fuente: Elaboración propia.

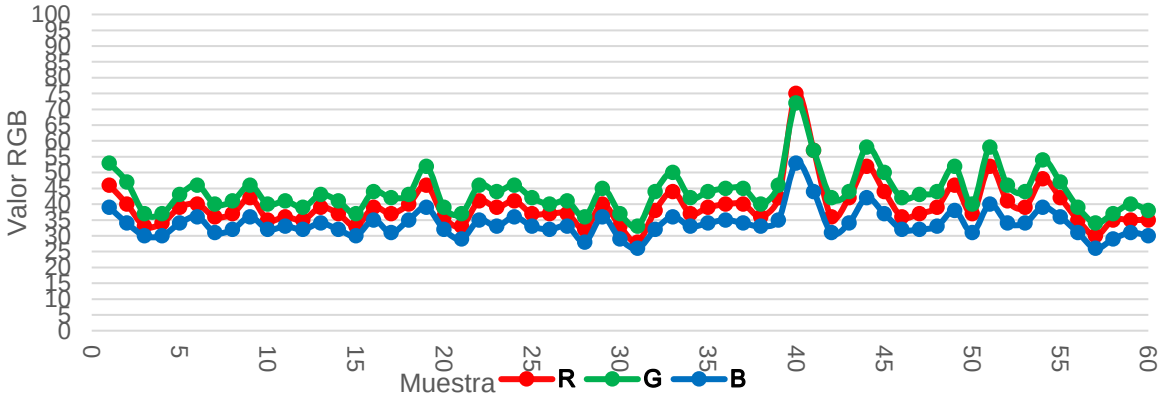


Gráfico 2: índice de clorofila medio fertilizante. Fuente: Elaboración propia.

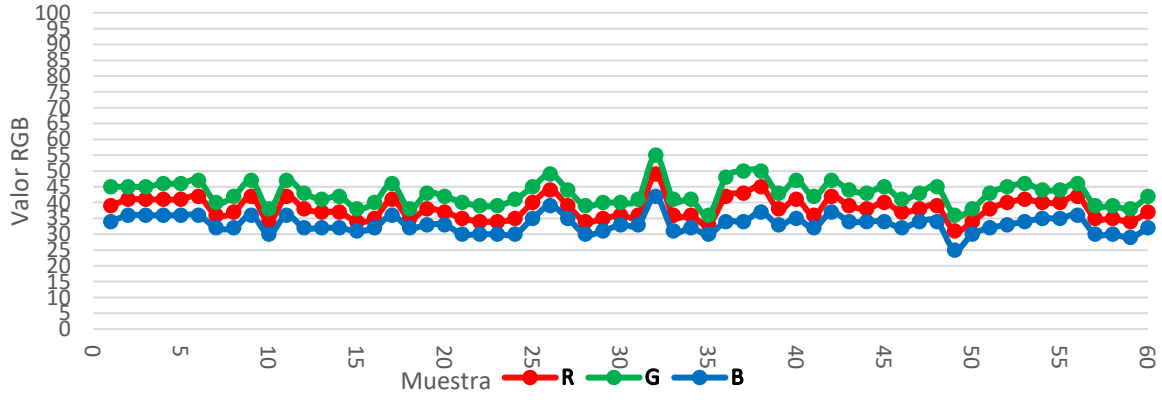


Gráfico 3: índice de clorofila con fertilizante. Fuente: Elaboración propia.

las deficiencias nutricionales en las plantas. Igualmente, de una manera más exhaustiva se realizó la medición de 50 hojas de maíz, divididas en dos clases nutricionales con 25 muestras para cada clase.

La clase 1 se asocia a deficiencia nutricional intermedia la cual sugiere que la planta debe administrarse con una dosis intermedia de fertilizante nitrogenado y la clase asociada a deficiencia nutricional alta, la cual debe ser suplementada con dosis altas de fertilizante nitrogenado. En las Gráficos 1, 2 y 3 se muestran las mediciones. Finalmente, para evaluar la capacidad de predicción de las dos clases asociadas a la deficiencia de nitrógeno, se utilizó los valores de índice de clorofila para cada clase y se entrenó un modelo de aprendizaje de máquina, específicamente mediante el algoritmo *Extreme Gradient Boosting* con el uso de la librería *xgboost* en el lenguaje de programación *R*. El modelo se entrenó con el 75 por ciento de las muestras y se evaluó con el 25 por ciento restante. Los resultados obtenidos para el modelo fueron un valor de 0.703 de precisión y un 0.2962963 de error. Los resultados anteriores muestran que el enfoque es útil en la detección de diferentes niveles deficiencia nutricionales, y tomando en cuenta que no se obtuvieron las medidas del índice de clorofila en la mejor etapa fisiológica de la planta (de acuerdo a los expertos en nutrición vegetal y quienes nos hicieron el favor de prestarnos sus experimentos de campo para realizar las mediciones), el enfoque es muy prometedor para mediciones futuras que contemplen las etapas de mayor importancia para la planta y con mayor número de muestras de entrenamiento. Cabe señalar que es importante para cada cultivo y cada material genético asociado, realizar las mediciones para obtener un conjunto de entrenamiento de buen tamaño que después nos será útil en la generación de un modelo predictivo, en el presente trabajo se obtuvieron pocas muestras de entrenamiento debido a que fue un muestreo de oportunidad, en experimentos facilitados por expertos en nutrición vegetal.

Finalmente, los resultados anteriores muestran que el sensor de RGB mediante la generación de una escala adecuada por variedad de planta puede

detectar de manera cualitativa el estado nutricional del cultivo. En la Fig. 61 se ilustra la medición de la hoja de maíz.

Figura 61: Medición de índice de clorofila en hoja de maíz.
Fuente: Elaboración propia.

3.3 Determinación de temperatura del suelo

Para la determinación de la temperatura del suelo se utilizó como contraste un termómetro de suelo comercial marca TFA a una profundidad de 5 y 10 cm por muestra, en un total de 3 muestras de acuerdo con criterios establecidos previamente. Los resultados del termómetro fueron evaluados contra las 3 muestras de temperatura suelo hechas por el sensor del sistema embebido. En el sistema embebido la temperatura se obtiene a través de semiconductores dándole ventaja en ahorro de energía a diferencia de otros sensores de temperatura. La cubierta

diseñada ayuda en la protección de agua y polvo. El promedio de las repeticiones para cada muestra y cada instrumento fueron contrastados obteniendo diferencias de hasta 1.7 grados Celsius entre dispositivo y el termómetro comercial. En general el desempeño del sensor de temperatura del sistema embebido se encuentra en un rango aceptable para las mediciones en campo. Igualmente, las pruebas de medición son más funcionales que cuantitativas, ya que el fabricante del sensor afirma la precisión de la medición [44]. En la Fig. 62 ilustra la medición de temperatura de suelo.

Figura 62: Medición de temperatura de suelo.
Fuente: Elaboración propia.

3.4 Determinación de la eficacia de comunicación y GPS

Para comprobar la eficacia de la comunicación LoRa se efectuó un conteo de muestra/transmisión de datos cada 10 segundos en un ambiente sin exposición al sol. Como resultado se obtuvo que más de 500 ciclos fueron exitosos y la carga de la batería solo bajo 1%. Con esto se determina una comunicación y gestión de energía aprobada. También se comprobó que la georreferenciación con el GPS es aceptable, mediante las coordenadas (latitud y longitud) ingresadas en la aplicación

de Google Maps como referencia. Sin embargo, los resultados conseguidos en campo abierto tuvieron mayor precisión.

3.5 Validación de datos recibidos

La validación de datos recibidos se comprobó tanto en el *gateway* como en la plataforma Thingspeak. Durante dos días consecutivos se enviaron datos, el primer día la plataforma recibió 1176 mensajes exitosos y el segundo día 1302. En el Gráfico 4 se muestra la gráfica de mensajes recibidos en la plataforma.

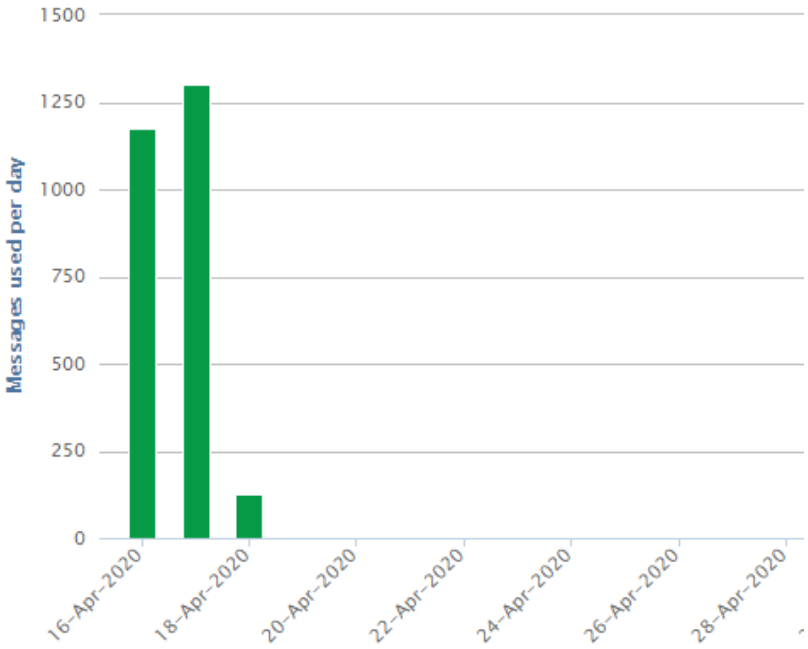


Gráfico 4: Datos recibidos en la plataforma ThinsSpeak. Fuente: Elaboración propia.

Conclusiones

Conclusiones

Como se mostró en el desarrollo del proyecto, fue factible la generación del sistema embebido por medio de la integración de plataformas libres logrando un funcionamiento adecuado de los diferentes módulos integrados con gran autonomía energética. Otro aspecto que resaltar del diseño es la utilización de materiales que cumplen con las normas oficiales del RoHS en el caso de las tarjetas electrónicas, y materiales biodegradables en el caso de la carcasa del dispositivo, que lo hacen atractivo para su producción y aplicación en el campo. Además, la comparación del desempeño de los sensores de pH, índice de clorofila y temperatura con estándares empleados por especialistas en el laboratorio arroja resultados satisfactorios que demuestran la utilidad del sistema embebido en la medición de variables agrícolas para un adecuado manejo de suelo y cultivos. Aunque actualmente este desarrollo tecnológico se centró en las mediciones de pH y temperatura del suelo e índice de clorofila que son las más demandadas en empresas de giro agrícola, también cuenta con otros sensores como el de humedad. Otra bondad del diseño modular del sistema embebido aquí descrito es que permite intercambiar sensores con algunas modificaciones, haciendo de este dispositivo un marco de referencia para nuevos desarrollos en la resolución de problemas agrícolas. Como trabajo futuro, se piensa evolucionar el sistema embebido para la producción en serie.

Referencias

[1] G. Berry, “Challenges and potential solutions for complex embedded systems,” in *2011 Proceedings of the Ninth ACM International Conference on Embedded Software (EMSOFT)*, Oct. 2011, p. 1, doi: 10.1145/2038642.2038644.

[2] M. Saadatmand, A. Cicchetti, and M. Sjödin, “A methodology for designing energy-aware secure embedded systems,” in *2011 6th IEEE International Symposium on Industrial and Embedded Systems*, 2011, pp. 87–90, doi: 10.1109/SIES.2011.5953687.

[3] K. Chopra, K. Gupta, and A. Lambora, “Future Internet: The Internet of Things-A Literature Review,” in *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*, Feb. 2019, pp. 135–139, doi: 10.1109/COMITCon.2019.8862269.

[4] G. Carvalho and J. Bernardino, “The Internet of Things and big data: Future trends,” in *2017 12th Iberian Conference on Information Systems and Technologies (CISTI)*, 2017, pp. 1–4, doi: 10.23919/CISTI.2017.7975972.

[5] K. L. Ang and J. K. P. Seng, “Application Specific Internet of Things (ASIoTs): Taxonomy, Applications, Use Case and Future Directions,” *IEEE Access*, vol. 7, pp. 56577–56590, 2019, doi: 10.1109/ACCESS.2019.2907793.

[6] H. C. J. Godfray *et al.*, “Food security: The challenge of feeding 9 billion people,” *Science*, vol. 327, no. 5967. pp. 812–818, Feb. 2010, doi: 10.1126/science.1185383.

[7] L. Lipper *et al.*, “Climate-smart agriculture for food security,” *Nat. Clim. Chang.*, vol. 4, no. 12, pp. 1068–1072, Dec. 2014, doi: 10.1038/nclimate2437.

[8] A. Gulati and S. Thakur, “Smart Irrigation Using Internet of Things,” in *2018 8th International Conference on Cloud Computing, Data Science Engineering (Confluence)*, 2018, pp. 819–823, doi: 10.1109/CONFLUENCE.2018.8442928.

[9] R. Dagar, S. Som, and S. K. Khatri, “Smart Farming – IoT in Agriculture,” in *2018 International Conference on Inventive Research in Computing Applications (ICIRCA)*, 2018, pp. 1052–1056, doi: 10.1109/ICIRCA.2018.8597264.

[10] N. Naik, “LPWAN Technologies for IoT Systems: Choice Between Ultra Narrow Band and Spread Spectrum,” in *2018 IEEE International Systems Engineering Symposium (ISSE)*, Oct. 2018, pp. 1–8, doi: 10.1109/SysEng.2018.8544414.

[11] Đ. Banđur, B. Jakšić, M. Banđur, and S. Jović, “An analysis of energy efficiency in Wireless Sensor Networks (WSNs) applied in smart agriculture,” *Comput. Electron. Agric.*, vol. 156, p. 500—507, 2019, doi: 10.1016/j.compag.2018.12.016.

[12] A. Zourmand, A. L. Kun Hing, C. Wai Hung, and M. AbdulRehman, “Internet of Things (IoT) using LoRa technology,” in *2019 IEEE International Conference on Automatic Control and Intelligent Systems (I2CACIS)*, 2019, pp. 324–330, doi: 10.1109/I2CACIS.2019.8825008.

[13] A. Lavric and V. Popa, “Internet of Things and LoRa™ Low-Power Wide-Area

Networks: A survey," in *2017 International Symposium on Signals, Circuits and Systems (ISSCS)*, 2017, pp. 1–5, doi: 10.1109/ISSCS.2017.8034915.
[14] P. M. Glibert, J. Harrison, C. Heil, and S. Seitzinger, "Escalating worldwide use of urea - A global change contributing to coastal eutrophication," *Biogeochemistry*, vol. 77, no. 3. Springer, pp. 441–463, Feb. 2006, doi: 10.1007/s10533-005-3070-5.
[15] D. L. Sparks, A. L. Page, P. A. Helmke, R. H. Loeppert, and G. W. Thomas, "Soil pH and Soil Acidity," John Wiley & Sons, Ltd, 1996, pp. 475–490.
[16] J.-H. Chen and S. A. Barber, "Soil pH and Phosphorus and Potassium Uptake by Maize Evaluated with an Uptake Model," *Soil Sci. Soc. Am. J.*, vol. 54, no. 4, pp. 1032–1036, 1990, doi: 10.2136/sssaj1990.03615995005400040017x.
[17] E. Jeppesen *et al.*, "Interaction of Climate Change and Eutrophication," in *Climate Change Impacts on Freshwater Ecosystems*, John Wiley & Sons, Ltd, 2010, pp. 119–151.
[18] P. Shrestha, B. Subedi, M. Girard, C. Parikh, and N. Kandalaft, "Wireless Communication Between FPGA and Microcontroller," in *2020 10th Annual Computing and Communication Workshop and Conference (CCWC)*, 2020, pp. 402–405, doi: 10.1109/CCWC47524.2020.9031196.
[19] element14, "Comparing an FPGA to a Microcontroller, Microprocessor or an ASIC," 2018. https://www.element14.com/community/groups/fpga-group/blog/2018/02/22/comparing-an-fpga-to-a-microcontroller-microprocessor-or-an-asic (accessed Jan. 28, 2020).
[20] S. Jun-yong, "Design and implementation of embedded GPS system," in *2012 IEEE International Conference on Computer Science and Automation Engineering (CSAE)*, May 2012, vol. 1, pp. 311–314, doi: 10.1109/CSAE.2012.6272604.
[21] B. V Vishakh, M. K. Khwaja, and C. M. Vidhyapathi, "Comprehensive automated device for hotel management using I2C protocol," in *2015 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)*, 2015, pp. 1–4, doi: 10.1109/ICCIC.2015.7435683.
[22] C. Liu, Q. Meng, T. Liao, X. Bao, and C. Xu, "A Flexible Hardware Architecture for Slave Device of I2C Bus," in *2019 International Conference on Electronic Engineering and Informatics (EEI)*, Nov. 2019, pp. 309–313, doi: 10.1109/EEI48997.2019.00074.
[23] "I2C-bus.org," 2006. https://www.i2c-bus.org/ (accessed Jan. 28, 2020).
[24] P. Hockicko, J. Kúdelčík, F. Muñoz, and L. Muñoz-Senovilla, "Electrical properties of LiPO3 glasses," in *2014 ELEKTRO*, May 2014, pp. 654–657, doi: 10.1109/ELEKTRO.2014.6848981.
[25] V. Shirmohammadli, A. Saberkari, H. Martínez-García, and E. Alarcón-Cot, "LDO-assisted vs. linear-assisted DC/DC converters: A comprehensive study and comparison," in *2016 IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, 2016, pp. 105–108, doi: 10.1109/ICECS.2016.7841143.
[26] S. Puengsungwan and K. Jirasereeamornkul, "Internet of Things (IoTs) based hydroponic lettuce farming with solar panels," in *2019 International Conference on Power, Energy and Innovations (ICPEI)*, 2019, pp. 86–89, doi: 10.1109/ICPEI47862.2019.8944986.

[27] B. Balasingam, G. V Avvari, B. Pattipati, K. Pattipati, and Y. Bar-Shalom, “Robust battery fuel gauge algorithm development, part 1: Online parameter estimation,” in *2014 International Conference on Renewable Energy Research and Application (ICRERA)*, Oct. 2014, pp. 98–103, doi: 10.1109/ICRERA.2014.7016538.

[28] M. Aref and A. Sikora, “Free space range measurements with Semtech Lora™ technology,” in *2014 2nd International Symposium on Wireless Systems within the Conferences on Intelligent Data Acquisition and Advanced Computing Systems*, 2014, pp. 19–23, doi: 10.1109/IDAACS-SWS.2014.6954616.

[29] M. Honda, T. Sakurai, and M. Takamiya, “Wireless temperature and illuminance sensor nodes with energy harvesting from insulating cover of power cords for building energy management system,” in *2015 IEEE PES Asia-Pacific Power and Energy Engineering Conference (APPEEC)*, Nov. 2015, pp. 1–5, doi: 10.1109/APPEEC.2015.7381080.

[30] M. Miki, Y. Kasahara, T. Hiroyasu, and M. Yoshimi, “Construction of illuminance distribution measurement system and evaluation of illuminance convergence in Intelligent Lighting System,” in *SENSORS, 2010 IEEE*, Nov. 2010, pp. 2431–2434, doi: 10.1109/ICSENS.2010.5690785.

[31] Microchip Technology, “ATmega328P,” 2015. https://www.microchip.com/wwwproducts/en/ATmega328p (accessed Jan. 28, 2020).

[32] Microchip Technology, “USB/AC Battery Charger with Power Path Management,” 2019. https://www.microchip.com/wwwproducts/en/en536670 (accessed Jan. 28, 2020).

[33] P. Guerriero, M. Coppola, I. Spina, I. Matacena, and S. Daliento, “A voltage divider strategy for reducing the hot spot temperature in partially shaded solar panels,” in *2017 6th International Conference on Clean Electrical Power (ICCEP)*, 2017, pp. 53–57, doi: 10.1109/ICCEP.2017.8004791.

[34] T. Instruments, “TPS63030,” 2020. https://www.ti.com/product/TPS63030 (accessed Jan. 28, 2020).

[35] M. Integrated, “MAX17043,” 2018. https://www.maximintegrated.com/en/products/power/battery-management/MAX17043.html (accessed Jan. 28, 2020).

[36] Atlas-Scientific, “pH OEM,” 2018. https://www.atlas-scientific.com/circuits/ph-oem-circuit/ (accessed Jan. 28, 2020).

[37] Anand N, G. Joseph, S. S. Oommen, and R. Dhanabal, “Design and implementation of a high speed Serial Peripheral Interface,” in *2014 International Conference on Advances in Electrical Engineering (ICAEE)*, 2014, pp. 1–3, doi: 10.1109/ICAEE.2014.6838431.

[38] Quectel, “GPS L80,” 2020. https://www.quectel.com/product/l80.htm (accessed Jan. 28, 2020).

[39] C. Mejia-Aguire and D. Armenta-Medina, “Smart Soil PCB,” 2020. https://workspace.circuitmaker.com/Projects/Details/Jael-Aguirre/SMARTSOIL (accessed Jan. 28, 2020).

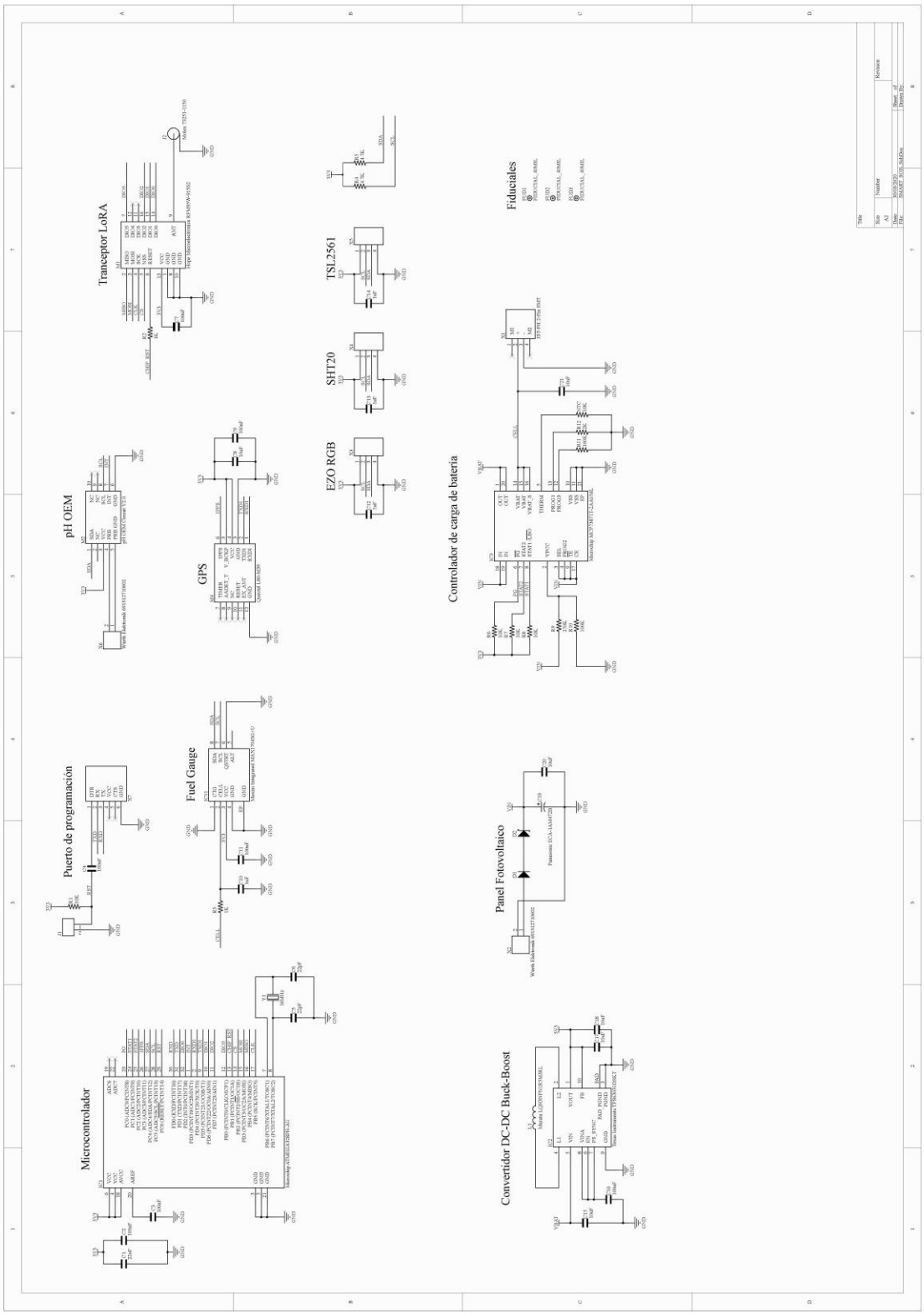
[40] C. Mejia-Aguire and D. Armenta-Medina, “Project Hub Smart Soil,” 2020. https://create.arduino.cc/projecthub/Cristian_Mejia/smart-soil-bd5ce3 (accessed Jan. 28, 2020).

[41] C. Mejia-Aguire and D. Armenta-Medina, “Thingspeak chanel Smart Soil,” 2020. https://thingspeak.com/channels/965199 (accessed Jan. 28, 2020).
[42] ANSI, “Degrees of Protection Provided by Enclosures (IP Code),” 2004. https://www.nema.org/Standards/ComplimentaryDocuments/ANSI-IEC-60529.pdf (accessed Jan. 28, 2020).
[43] D. Gaviria-Palacio, J. J. Guáqueta-Restrepo, D. M. Pineda-Tobón, and J. C. Pérez, “Fast estimation of chlorophyll content on plant leaves using the light sensor of a smartphone,” *DYNA*, vol. 84, no. 203, pp. 233–239, Dec. 2017, doi: 10.15446/dyna.v84n203.64316.
[44] Sensirion, “Digital Humidity Sensor SHT1x (RH/T),” *Sensirion*, 2018. https://www.sensirion.com/products/humidity-sensors/digital-humidity-sensors-for-accurate-measurements/ (accessed Jan. 28, 2020).

Anexos

Anexos

Anexo I: Esquemático de la tarjeta electrónica



Anexo II: Materiales y costos del sistema embebido

Costos					
#	**Componente**	**SKU**	**Ctd.**	**Precio unitario (USD)**	**Precio Ext. (USD)**
1	Microcontrolador 8 bits ATMEGA328P Microchip	ATMEGA328PB-AU	1	$1.74	$1.74
2	Modulo LoRa RFM95W Hope Microelectronics	RFM95W-915S2	1	$10.29	$10.29
3	GPS L80 Quectel	L80-M39	1	$7.481	$7.481
4	Controlador de carga para batería BCMC MCP73871 Microchip	MCP73871T-2CCI/ML	1	$2.33	$2.33
5	Convertidor DC-DC Buck-Boost TPS6303 Texas Instruments	TPS63031DSKR	1	$2.43	$2.43
6	Fuel Gauge con Model Gauge MAX17043 Maxim Integrated	MAX17043G+U	1	$3.18	$3.18
7	Modulo pH OEM Circuit V2.0 Atlas Scientific	pH-OEM	1	$40.00	$40.00
8	Consumer Grade pH Probe Atlas Scientific	ENV-30-pH	1	$39.99	$39.99
9	EZO-RGB Embedded Color Sensor Atlas Scientific	EZO-RGB	1	$49.99	$49.99
10	Panel solar Medium 6V 2W Voltaic System	2.0W-PANEL	1	$29.00	$29.00
11	Modulo TSL2561 Digital Lux/Light Sensor Adafluit	Adafruit-3611	1	$7.95	$7.95
12	Sensor SHT20 (Waterproof Probe) DFRobot	SEN0227	1	$22.50	$22.50
13	Polymer Li-ion Rechargeable Battery 3.7V/6000mAh Sparkfun	DTP605068-3P	1	$29.95	$29.95
14	Conector SMA EDGE MOUNT MOLEX	538-73251-1150	1	$4.54	$4.54
15	Antena Right-angle Quad-Band - 2dBi SMA Plug Adafruit	Adafruit-1858	1	$4.95	$4.95
16	Conector JST-PH 2-pin SMT Adafruit	Adafruit-1769	1	$0.75	$0.75
17	Insertos M3 4mm Adafruit	Adafruit-4255	8	$0.119	$0.952
18	Espaciadores de nylon M3 20mm	Adafruit-3299	4	$0.8	$3.2
19	Conector THT 4 P 5.0 mm 20A Wurth Elektronik	691312710004	3	$1.08	$3.24
20	Conector THT 2 P 5.0 mm 20A Wurth Elektronik	691312710002	2	$0.533	$1.066

21	Conector Plug 2 P 5.0 mm 20A Wurth Elektronik	691352710002	2	$1.61	$3.22
22	Conector Plug 4 P 5.0 mm 20A Wurth Elektronik	691352710004	3	$3.22	$9.66
23	Inductor 1212 1.5uH 1100mA Murata Electronics	LQH3NPN1R5MJRL	1	$0.546	$0.546
24	Diodo Schottky 1A 20V ON Semiconductor	MBR120VLSFT1G	1	$0.559	$0.559
25	Diodo 1N4001 1A 30V AMP	AMP60-20469	1	$0.494	$0.494
26	Cristal 16MHz 18Pf ECS	520-160-18-5PX-T	1	$0.351	$0.351
27	Capacitor electrolítico de aluminio radial Panasonic	ECA-1AM472B	1	$1.12	$1.12
28	Capacitor de tantalio 22uF 0805 Vishay	T58W9226M6R3C0500	1	$0.598	$0.598
29	Capacitor cerámico MLCC 0.1uF 0805 KEMET	C0805C104M5RAC	7	$0.016	$0.112
30	Capacitor cerámico MLCC 22pF 0805 KEMET	C0805C220K5HACTU	2	$0.13	$0.26
31	Capacitor de tantalio 10uF 0805 Vishay	TMCP1A106MTRF	6	$0.442	$2.652
32	Capacitores cerámicos MLCC 1uF 0805 Wurth Elektronik	885012207078	4	$0.13	$0.52
33	Termistor NTC 0805 10Kohm TDK	NTCG203NH103JTDS	1	$0.338	$0.338
34	Resistor 4.7Kohms 1% Vishay	CRCW08054K70FKEAC	2	$0.13	$0.26
35	Resistor 10Kohms 1% Vishay	CRCW080510K0FKEAC	4	$0.13	$0.52
36	Resistor 1Kohms 1% Vishay	CRCW08051K00FKEAC	2	$0.13	$0.26
37	Resistor 270K 5% Bourns	CR0805-JW-274ELF	1	$0.13	$0.13
38	Resistor 100K 1% TE Connectivity	CRGCQ0805F100K	2	$0.13	$0.26
39	Resistor 2Kohms 1% Vishay	CRCW08052K00FKEAC	1	$0.13	$0.13
40	Glándula eléctrica PG7 Phoenix Contact	1424485	4	$0.832	$3.328
41	Tornillo M3 6mm INOX	-	8	$0.04	$0.32
42	Tuerca 1/4" INOX	-	1	$0.04	$0.04
43	Placa de circuito impreso PCB	SeeedStudio	1	$3.5	$3.5
44	Filamento PLA 1.75mm negro	Ventronic	1	$19.63	$19.63
45	Trípode aluminio 100cm	-	1	$7.76	$7.76
				TOTAL:	**$322.10**

Anexo III: Código fuente del microcontrolador

```
//////////////////////////////////////////////////////////////////////////////////
// Company: PLOTCHIP
// Engineer: Cristian Jael Mejia Aguirre
// Project Name: Smart Soil device
// Description: ^.^
// Version: 1.4
// Create Date: 20.11.2019
//////////////////////////////////////////////////////////////////////////////////

#include <MAX17043.h>
#include <Wire.h>
#include <DFRobot_SHT20.h>
#include <LowPower.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_TSL2561_U.h>
#include <SoftwareSerial.h>
#include <TinyGPS.h>
#include <SPI.h>
#include <RH_RF95.h>


#define i2c_id 0x65                 //default I2C address
#define one_byte_read 0x01          //used in a function to read data from the device
#define two_byte_read 0x02          //used in a function to read data from the device
#define four_byte_read 0x04         //used in a function to read data from the device
#define direccion_EZO 100

TinyGPS L80;
SoftwareSerial GPS(5, 4);
MAX17043 bateria;
DFRobot_SHT20 SHT20;
Adafruit_TSL2561_Unified tsl = Adafruit_TSL2561_Unified(TSL2561_ADDR_FLOAT, 12345);
RH_RF95 rf95;


//Lora
char node_id[3] = {1, 1, 1}; //LoRa End Node ID
float frequency = 868.0;
unsigned int count = 1;

//EZO RGB
byte code = 0;
char RGB_data[32];
byte in_char = 0;

//pH OEM
const byte active_hibernate_register = 0x06;        //register to read / write
const byte active_mode = 0x01;                      //take readings
const byte hibernate_mode = 0x00;                   //stop taking readings
const byte led_control_register = 0x05;             //register to read / write
const byte led_control_on = 0x01;                   //led on
const byte led_control_off = 0x00;                  //led off

byte bus_address = i2c_id;          //holds the I2C address.
```

```
union sensor_mem_handler              //declare the use of a union data type
{
  byte i2c_data[4];                   //define a 4 byte array in the union
  long answ;              //define an long in the union
};
union sensor_mem_handler move_data;     //declare that we will refer to the union as "move_data"


//MCP73871
const int PG_pin = A0; //Power Good Status
const int STAT1_pin =  A1; //Charge Status Output 1
const int STAT2_pin =  A2; //Charge Status Output 2
int PG = 0;
int STAT1 =  0;
int STAT2 =  0;

// Variables para enviar LoRa
unsigned int lux_e;
float temperatura_e;
float humedad_e;
float ph_e;
int bateria_e;
byte RED = 0x2D;
byte GREEN = 0x57;
byte BLUE = 0x2C;
int carga = 0;
float latitud;
float longitud;


////////////////////////////////////////////////////////////////////////////////
// MAIN PROGRAM
////////////////////////////////////////////////////////////////////////////////

void setup() {
  Serial.begin(9600);
  Wire.begin();
  MAX17043_CONFIGURACION();
  delay(100);
  MCP73871_CONFIGURACION();
  delay(100);
  SHT20_CONFIGURACION();
  delay(100);
  TSL2562_CONFIGURACION();
  delay(100);
  ph_CONFIGURACION();
  delay(100);
  EZO_CONFIGURACION();
  delay(100);
  L80_CONFIGURACION();
  delay(100);
  LORA_CONFIGURACION();
  Serial.println("");
  delay(1000);
}
```

```
void loop() {
  MAX17043_MUESTRA();
  MCP73871_MUESTRA();
  SHT20_MUESTRA();
  TSL2561_MUESTRA();
  ph_MUESTRA();
  EZO_MUESTRA();
  L80_MUESTRA();
  Serial.println("");
  Enviar_LoRa();
  Serial.println("");
  sleep(15); //15 segundos
}//LOOP



////////////////////////////////////////////////////////////////////////////////
// MCP73871 MICROCHIP
////////////////////////////////////////////////////////////////////////////////

void MCP73871_CONFIGURACION() {
  pinMode(PG_pin, INPUT);
  pinMode(STAT1_pin, INPUT);
  pinMode(STAT2_pin, INPUT);
  Serial.println("MCP73871");
}//MCP73871_CONFIGURACION

void MCP73871_MUESTRA() {
  PG = digitalRead(PG_pin);
  STAT1 = digitalRead(STAT1_pin);
  STAT2 = digitalRead(STAT2_pin);
  //  if (PG == LOW){Serial.println("PG");}
  //  if (STAT2 == LOW){Serial.println("STAT2");}
  //  if (STAT1 == LOW){Serial.println("STAT1");}
  if (STAT1 == HIGH && STAT2 == HIGH && PG == HIGH) {
    carga = 1;
    Serial.println("Estado de la bateria: No se presenta carga");
  }
  if (STAT1 == LOW && STAT2 == HIGH && PG == HIGH) {
    carga = 2;
    Serial.println("Low battery output");
  }
  if (STAT1 == LOW && STAT2 == HIGH && PG == LOW) {
    carga = 3;
    Serial.println("Constant Voltage");
  }
  if (STAT1 == HIGH && STAT2 == LOW && PG == LOW) {
    carga = 4;
    Serial.println("Chaerge complete");
  }
  if (STAT1 == LOW && STAT2 == LOW && PG == LOW) {
    carga = 5;
    Serial.println("Timer fault");
  }
```

```
  //Serial.println(" ");
}//MCP73871_MUESTRA


////////////////////////////////////////////////////////////////////////////////
// MAX17043 LIPO FUEL GAUGE
////////////////////////////////////////////////////////////////////////////////

void MAX17043_CONFIGURACION() {
  bateria.reset();
  bateria.quickStart();
  Serial.println("MAX17043");
}//MAX17043_CONFIGURACION()

void MAX17043_MUESTRA() {
  float voltaje = bateria.getVCell();
  Serial.print("Voltaje de la bateria: ");
  Serial.print(voltaje, 4);
  Serial.println("V");
  //float porcentaje = bateria.getSoC();
  bateria_e = (int)bateria.getSoC();
  Serial.print("Porcentaje de la bateria: ");
  //Serial.print(porcentaje);
  Serial.print(bateria_e);
  Serial.println("%");
  //Serial.println("");
}//MAX17043_MUESTRA

////////////////////////////////////////////////////////////////////////////////
// SHT20 SENSIRION HUMEDITY AND TEMPERATURE
////////////////////////////////////////////////////////////////////////////////

void SHT20_CONFIGURACION() {
  SHT20.initSHT20();// Init SHT20 Sensor
  //SHT20.checkSHT20(); // Check SHT20 Sensor
  Serial.println("SHT20");
}//SHT_CONFIGURACION

void SHT20_MUESTRA() {
  //float humedad = SHT20.readHumidity();
  humedad_e = SHT20.readHumidity();
  Serial.print("Humedad: ");
  Serial.print(humedad_e, 2);
  Serial.println("%");
  temperatura_e = SHT20.readTemperature();
  Serial.print("Temperatura: ");
  Serial.print(temperatura_e);
  Serial.println("°C");
  //Serial.println();
}//SHT_MUESTRA()


////////////////////////////////////////////////////////////////////////////////
// TSL2562 AMS LIGHT SENSOR
////////////////////////////////////////////////////////////////////////////////
```

```
void TSL2562_CONFIGURACION() {
  sensor_t sensor;
  tsl.getSensor(&sensor);
  tsl.enableAutoRange(true);
  tsl.setIntegrationTime(TSL2561_INTEGRATIONTIME_13MS);      /* fast but low resolution */

  if (!tsl.begin())  {
    Serial.print("Ooops, no TSL2561 detected ... Check your wiring or I2C ADDR!");
    while (1);
  }
  Serial.println("TSL2561");

}//TSL2562_CONFIGURACION

void TSL2561_MUESTRA() {
  sensors_event_t event;
  tsl.getEvent(&event);
  if (event.light) {
    lux_e = event.light;
    Serial.print("Iluminancia: ");
    Serial.print(lux_e);
    Serial.println(" lux");
  }
  else {
    Serial.println("TSL2561 overload");
  }
} //TSL2561_MUESTRA


////////////////////////////////////////////////////////////////////////////////
// pH OEM ATLAS SCIENTIFIC
////////////////////////////////////////////////////////////////////////////////

void ph_MUESTRA() {
  on_ph();
  delay(250);
  reading_ph();
  delay(250);
  off_ph();
}

void ph_CONFIGURACION() {
  Serial.println("pH OEM");
  led_off_ph();
}

void led_on_ph() {
  i2c_write_byte(led_control_register, led_control_on);       //write the led on command
  i2c_read(led_control_register, one_byte_read);             //read from the led control register to confirm
it is set correctly
  //if (move_data.i2c_data[0] == 1)Serial.println("*LED ON");
}

void led_off_ph() {
  i2c_write_byte(led_control_register, led_control_off);      //write the led off command
```

```
  i2c_read(led_control_register, one_byte_read);            //read from the led control register to confirm
it is set correctly
  //if (move_data.i2c_data[0] == 0)Serial.println("*LED off");
}

void on_ph() {
  i2c_write_byte(active_hibernate_register, active_mode);    //write the active mode enable command
  i2c_read(active_hibernate_register, one_byte_read);         //read from the active / hibernate control
register to confirm it is set correctly
  //if (move_data.i2c_data[0] == 1)Serial.println("active");
}

void off_ph() {
  i2c_write_byte(active_hibernate_register, hibernate_mode);    //write the active mode disable
command
  i2c_read(active_hibernate_register, one_byte_read);      //read from the active / hibernate control
register to confirm it is set correctly
  //if (move_data.i2c_data[0] == 0)Serial.println("hibernate");
}

void reading_ph() {
  const byte pH_register = 0x16;                //register to read
  float pH = 0;                       //used to hold the new pH value

  i2c_read(pH_register, four_byte_read);             //I2C_read(OEM register, number of bytes to read)
  pH = move_data.answ;                      //move the 4 bytes read into a float
  //pH /= 1000;                         //divide by 10 to get the decimal point
  ph_e = pH/1000;
  Serial.print("pH: ");
  Serial.println(ph_e);
}

void i2c_read(byte reg, byte number_of_bytes_to_read) {                 //used to read 1,2,and 4 bytes:
i2c_read(starting register,number of bytes to read)
  byte i;                                   //counter
  Wire.beginTransmission(bus_address);                        //call the device by its ID number
  Wire.write(reg);                                  //transmit the register that we will start from
  Wire.endTransmission();                                 //end the I2C data transmission
  Wire.requestFrom(bus_address, (byte)number_of_bytes_to_read);                 //call the device and
request to read X bytes
  for (i = number_of_bytes_to_read; i > 0; i--) {
    move_data.i2c_data[i - 1] = Wire.read();  //with this code we read multiple bytes in reverse
  }
  Wire.endTransmission();                                 //end the I2C data transmission
}

void i2c_write_byte(byte reg, byte data) {                        //used to write a single byte to a register:
i2c_write_byte(register to write to, byte data)

  Wire.beginTransmission(bus_address);                        //call the device by its ID number
  Wire.write(reg);                                  //transmit the register that we will start from
  Wire.write(data);                                  //write the byte to be written to the register
  Wire.endTransmission();                                 //end the I2C data transmission
}
```

```
////////////////////////////////////////////////////////////////////////////////
// EZO RGB ATLAS SCIENTIFIC
////////////////////////////////////////////////////////////////////////////////

void EZO_CONFIGURACION() {
  Serial.println("EZO RGB");
  //EZO_BRILLO_ON();
  //delay(3500);
  EZO_BRILLO_SET();
  //delay(500);
  EZO_SLEEP();
}

void EZO_MUESTRA() {
  byte i = 0;
  char leer[2] = "r";
  //Serial.println("MUESTRA");
  Wire.beginTransmission(direccion_EZO);
  Wire.write(leer);
  Wire.endTransmission();
  delay(250);
  Wire.requestFrom(direccion_EZO, 32, 1);
  code = Wire.read();
  //Serial.println(code);

  while (Wire.available()) {              //are there bytes to receive.
    in_char = Wire.read();                //receive a byte.
    RGB_data[i] = in_char;                  //load this byte into our array.
    i += 1;                             //incur the counter for the array element.
    if (in_char == 0) {                     //if we see that we have been sent a null command.
      i = 0;                            //reset the counter i to 0.
      break;                              //exit the while loop.
    }
  }
  Serial.print("Clorofila: ");
  Serial.println(RGB_data);
  EZO_SLEEP();
}

void EZO_SLEEP() {
  char sleep_1[6] = "Sleep";
  Wire.beginTransmission(direccion_EZO);
  Wire.write(sleep_1);
  Wire.endTransmission();
}

void EZO_BRILLO_ON() {
  char brillo_on[5] = "L,20";
  Wire.beginTransmission(direccion_EZO);
  Wire.write(brillo_on);
  Wire.endTransmission();
}

void EZO_BRILLO_SET() {
  char brillo[7] = "L,20,T";
  Wire.beginTransmission(direccion_EZO);
```

```
  Wire.write(brillo);
  Wire.endTransmission();
}

////////////////////////////////////////////////////////////////////////////////
// GPS L80 QUELTEC
////////////////////////////////////////////////////////////////////////////////

void L80_CONFIGURACION() {
  GPS.begin(9600);
  Serial.println("L80");
  sleep_L80();
}

void sleep_L80() {
  if (GPS.available()) {
    GPS.println("$PMTK161,0*28");
  }
}

void active_L80() {
  GPS.println("");
}

void L80_MUESTRA() {
  float flat, flon;
  float lat, lon;
  unsigned long age;
  bool newData = false;
  active_L80();
  delay(500);

  for (unsigned long start = millis(); millis() - start < 2500;)
  {
    while (GPS.available())
    {
      char c = GPS.read();
      //Serial.write(c); // uncomment this line if you want to see the GPS data flowing
      if (L80.encode(c)) // Did a new valid sentence come in?
        newData = true;
    }
  }

  if (newData)
  {
    L80.f_get_position(&flat, &flon, &age);
    Serial.print("LAT: ");
    Serial.print(flat == TinyGPS::GPS_INVALID_F_ANGLE ? 0.0 : flat, 6);
    //lat = flat == TinyGPS::GPS_INVALID_F_ANGLE ? 0.0 : flat;
    Serial.print(" LON: ");
    Serial.println(flon == TinyGPS::GPS_INVALID_F_ANGLE ? 0.0 : flon, 6);
    //lon = flon == TinyGPS::GPS_INVALID_F_ANGLE ? 0.0 : flon;
  } else {
    Serial.print("LAT: ");
    Serial.print(lat,6);
    Serial.print(" LON: ");
```

```
    Serial.print(lon,6);
    Serial.println(" *");
  }
  sleep_L80();
}

////////////////////////////////////////////////////////////////////////////////
// RFM95W HOPERF Low Power Long Range Transceiver Module
////////////////////////////////////////////////////////////////////////////////

void LORA_CONFIGURACION(){
    if (!rf95.init())
        Serial.println("Init failed Lora");
    // Setup ISM frequency
    rf95.setFrequency(frequency);
    // Setup Power,dBm
    rf95.setTxPower(13);
    rf95.setSyncWord(0x34);

    Serial.print("LoRa End Node ID: ");
    for(int i = 0;i < 3; i++)
    {
        Serial.print(node_id[i],HEX);
    }
    Serial.println();
}

void Enviar_LoRa(){

    int temperatura_integer = (int)temperatura_e;
    int temperatura_x100 = temperatura_e*100;
    int temperatura_decimal = temperatura_x100%100;

    int humedad_integer = (int)humedad_e;
    unsigned  int humedad_x100 = humedad_e*100;
    int humedad_decimal = humedad_x100%100;

    int ph_integer = (int)ph_e;
    int ph_x100 = ph_e*100;
    int ph_decimal = ph_x100%100;

    latitud = -179.123456;
    longitud = 120.123456;

//    Serial.println(latitud,6);

    unsigned long  latitud_x;
    byte latitud_signo;
    if(latitud < 0){latitud_signo = 1;}else {latitud_signo = 0;}
    int latitud_integer = abs((int)latitud);
    if(latitud_signo == 1){latitud_x = latitud*-1000000;}
    if(latitud_signo == 0){latitud_x = latitud*1000000;}
    unsigned long latitud_decimal = latitud_x%1000000;
```

```
//    Serial.print("############    ");
    Serial.print("COUNT=");
    Serial.println(count);
//    Serial.println("    ############");
    count++;

    unsigned char data[50] = {0};
    int dataLength = 23; // Payload Length

    //Node ID:
    data[0] = node_id[0];
    data[1] = node_id[1];
    data[2] = node_id[2];

    //Humedad:
    data[3] = humedad_integer;//Get Humidity Integer Part
    data[4] = humedad_decimal;//Get Humidity Decimal Part

     //Temperatura:
    data[5] = temperatura_integer;//Get Temperature Integer Part
    data[6] = temperatura_decimal;//Get Temperature Decimal Part

    //pH:
    data[7] = ph_integer;//Get pH Integer Part
    data[8] = ph_decimal;//Get pH Decimal Part

    //lux:
    data[9] = lux_e;//Get lux low part
    data[10] = lux_e >> 8;//Get lux high part

    //Bateria:
    data[11] = bateria_e;//Get % battery

    //Clorofila:
    data[12] = RED;//Get R Clorfila
    data[13] = GREEN;//Get G Clorfila
    data[14] = BLUE;//Get B Clorfila

    //Carga:
    data[15] = carga;//Get carga

    //Latitud:
    data[16] = latitud_signo;
    data[17] = latitud_integer;//Get latitud integer low part
    data[18] = latitud_integer >> 8;//Get integer high part
    data[19] = latitud_decimal;
    data[20] = latitud_decimal >> 8;
    data[21] = latitud_decimal >> 16;
    data[22] = latitud_decimal >> 24;


    //Longitud:
//    data[23] = longitud;
//   data[24] = longitud >> 8;
//    data[25] = longitud >> 16;
```

```
//    data[26] = longitud >> 24;



    //CALCULA EL CRC
    uint16_t crcData = CRC16((unsigned char*)data,dataLength);//get CRC DATA
    //Serial.print("CRC: ");
    //Serial.println(crcData,HEX);

    //MUESTRA DATOS EN HEX SIN CRC
//    Serial.print("Data to be sent(without CRC): ");
      int i;
//    for(i = 0;i < dataLength; i++)
//    {
//        Serial.print(data[i],HEX);
//        Serial.print(" ");
//    }
//    Serial.println();


    //COPIA DATOS PARA EL ENVIO
    unsigned char sendBuf[50]={0};
    for(i = 0;i < dataLength;i++)
    {
       sendBuf[i] = data[i] ;
    }

    //AGREGA EL CRC
    sendBuf[dataLength] = (unsigned char)crcData; // Add CRC to LoRa Data
    sendBuf[dataLength+1] = (unsigned char)(crcData>>8); // Add CRC to LoRa Data

    //MUESTRA DATOS EN HEX CON CRC
//    Serial.print("Data to be sent(with CRC):    ");
//    for(i = 0;i < (dataLength +2); i++)
//    {
//        Serial.print(sendBuf[i],HEX);
//        Serial.print(" ");
//    }
//    Serial.println();


    //ENVIAR DATOS
    Serial.print("Enviando datos: ");//
    rf95.send(sendBuf, dataLength+2);//Send LoRa Data

    uint8_t buf[RH_RF95_MAX_MESSAGE_LEN];//Reply data array
    uint8_t len = sizeof(buf);//reply data length

    if (rf95.waitAvailableTimeout(3000))// Check If there is reply in 3 seconds.
    {
       // Should be a reply message for us now
       if (rf95.recv(buf, &len))//check if reply message is correct
      {
          if(buf[0] == node_id[0] && buf[1] == node_id[2] && buf[2] == node_id[2] ) // Check if reply
message has the our node ID
         {
```

```
            pinMode(4, OUTPUT);
            digitalWrite(4, HIGH);
            Serial.print("Got Reply from Gateway: ");//print reply
            Serial.println((char*)buf);

            delay(400);
            digitalWrite(4, LOW);
            //Serial.print("RSSI: ");  // print RSSI
            //Serial.println(rf95.lastRssi(), DEC);
          }
        }
        else
        {
          Serial.println("recv failed");//
          rf95.send(sendBuf, strlen((char*)sendBuf));//resend if no reply
        }
    }
    else
    {
        Serial.println("No reply, is LoRa gateway running?");//No signal reply
        rf95.send(sendBuf, strlen((char*)sendBuf));//resend data
    }
}

uint16_t calcByte(uint16_t crc, uint8_t b)
{
    uint32_t i;
    crc = crc ^ (uint32_t)b << 8;

    for ( i = 0; i < 8; i++)
    {
        if ((crc & 0x8000) == 0x8000)
            crc = crc << 1 ^ 0x1021;
        else
            crc = crc << 1;
    }
    return crc & 0xffff;
}

uint16_t CRC16(uint8_t *pBuffer,uint32_t length)
{
    uint16_t wCRC16=0;
    uint32_t i;
    if (( pBuffer==0 )||( length==0 ))
    {
      return 0;
    }
    for ( i = 0; i < length; i++)
    {
      wCRC16 = calcByte(wCRC16, pBuffer[i]);
    }
    return wCRC16;
}

////////////////////////////////////////////////////////////////////////////////////
// SLEEP MODE ATMEGA328P
```

```
////////////////////////////////////////////////////////////////////////////////

void sleep(int sec) {
  while (sec >= 8) {
    LowPower.powerDown(SLEEP_8S, ADC_OFF, BOD_OFF);
    sec -= 8;
  }
  if (sec >= 4) {
    LowPower.powerDown(SLEEP_4S, ADC_OFF, BOD_OFF);
    sec -= 4;
  }
  if (sec >= 2) {
    LowPower.powerDown(SLEEP_2S, ADC_OFF, BOD_OFF);
    sec -= 2;
  }
  if (sec >= 1) {
    LowPower.powerDown(SLEEP_1S, ADC_OFF, BOD_OFF);
    sec -= 1;
  }
}
```