Escuela Superior Politécnica del Litoral

Facultad de Ingeniería en Electricidad y Computación

"Diseño e Implementación de un Sistema IoT para el Monitoreo de Huertas Solares"

Proyecto Integrador

Previo la obtención del Título de:

Ingeniero en electrónica y automatización

Presentado por:
Baquerizo Cornejo Jefferson Aquiles
Rodríguez Lema Carlos Armando

Guayaquil - Ecuador
Año: 2023

Dedicatoria

El presente proyecto lo dedico a mis padres Julia Lema Bonete y Carlos Rodríguez Balarezo que en todo momento estuvieron conmigo brindándome su amor, consejos y apoyo incondicional para completar con éxito mi carrera de formación profesional. A mis hermanos Luis, Angélica y Verónica que siempre fueron y serán mi motivo de superación familiar y personal.

Este logro fue posible y va dedicado a cada uno de ustedes.

Carlos A. Rodríguez Lema

Dedicatoria

A dios, a mis padres, docentes que me han acompañado con su tiempo, dedicación y enseñanzas en este largo camino de aprendizaje, también a esta noble institución la Escuela Superior Politécnica del Litoral; ya que sin ellos este logro no hubiera sido posible.

Le dedico este éxito y los que vendrán a futuro.

Jefferson A. Baquerizo Cornejo

Agradecimientos

Nuestro más sincero agradecimiento en primer lugar a Dios por darnos las fuerzas y el espíritu para completar este reto. A todos aquellos familiares y conocidos que nos han brindado su apoyo incondicional para superar con éxito los obstáculos presentados en cada etapa de este largo camino. A nuestros docentes que con gran esmero y dedicación nos han impartido su conocimiento para culminar con éxito este proceso de formación profesional.

Declaración Expresa

Nosotros Baquerizo Cornejo Jefferson Aquiles y Rodriguez Lema Carlos Armando acordamos y reconocemos que la titularidad total y exclusiva sobre los derechos patrimoniales de patente de invención, modelo de utilidad, diseño industrial, información no divulgada y cualquier otro derecho o tipo de Propiedad Intelectual que corresponda o pueda corresponder respecto de cualquier investigación, desarrollo tecnológico o invención realizada durante el desarrollo de su trabajo de titulación, incluyendo cualquier derecho de participación de beneficios o de valor sobre titularidad de derechos, pertenecerán de forma total, perpetua, exclusiva e indivisible a LA ESPOL, sin limitación de ningún tipo. Se deja además expresa constancia de que lo aquí establecido constituye un "previo acuerdo", así como de ser posible bajo la normativa vigente de transferencia o cesión a favor de la ESPOL de todo derecho o porcentaje de titularidad que pueda existir.

Sin perjuicio de lo anterior los alumnos firmantes de la presente declaración reciben en este acto una licencia de uso gratuita e intransferible de plazo indefinido para el uso no comercial de cualquier investigación, desarrollo tecnológico o invención realizada durante el desarrollo de su trabajo de titulación, sin perjuicio de lo cual deberán contar con una autorización previa expresa de la ESPOL para difundir públicamente el contenido de la investigación, desarrollo tecnológico o invención.

Así también autorizamos expresamente a que la ESPOL realice la comunicación pública de la obra o invento, por cualquier medio con el fin de promover la consulta, difusión y uso público de la producción intelectual.

Guayaquil, 26 de septiembre del 2023.

Carlos Armando
Rodríguez Lema

Firmado electrónicamente por:
JEFFERSON AQUILES
BAQUERIZO CORNEJO

Jefferson Aquiles
Baquerizo Cornejo

Evaluadores

Firmado electrónicamente por:
DENNYS DICK CORTEZ ALVAREZ

Dennys Dick Cortez Alvarez

Profesor de Materia

Firmado electrónicamente por:
WILTON EDIXON AGILA GALVEZ

Wilton Edixon Agila Gálvez

Tutor de proyecto

RESUMEN

El presente proyecto integrador consiste en el diseño e implementación de un Sistema IoT para el monitoreo de granjas solares que facilite al usuario la interpretación de la información recopilada por este sistema y que a través de la estimación de la producción diaria de energía ayude a tomar decisiones referentes a los mantenimientos preventivos y/o correctivos de la granja solar.

Este sistema consta de un dispositivo IoT cuya función es la de comunicarse con los sensores instalados en la granja solar y obtener la información referente a las condiciones meteorológicas del sitio en donde se encuentra ubicada esta granja y una aplicación desarrollada para procesar y observar los datos que el dispositivo obtiene a través de una comunicación Modbus RS-485 y esto a su vez se complementa con una base de datos en línea para que el acceso a esta información se realice desde el sitio en donde el usuario se encuentre.

ABSTRACT

This integrative project consists of the design and implementation of an IoT System for monitoring solar farms that facilitates the user's interpretation of the information collected by this system and that, through the estimation of daily energy production, helps make decisions regarding preventive and/or corrective maintenance of the solar farm.

This system consists of an IoT device whose function is to communicate with the sensors installed in the solar farm and obtain information regarding the meteorological conditions of the site where this farm is located an application developed to process and observe the data that the device through Modbus RS-485 communication and this in turn is complemented with an online database so that access to this information is obtained from wherever the user is.

Keywords: Modbus RS-485, IoT, Database.

CONTENIDO

Evaluadores 6
Resumen 7
Abstract 8
Contenido 9
AbreviAturas 12
Simbología 14
Índice de figuras 15
Índice de tablas 19
Capítulo 1 20
1.1 Introducción 21
1.2 Descripción del problema 21
1.3 Justificación del problema 22
1.4 Objetivos 22
1.4.1 Objetivo general 22
1.4.2 Objetivos específicos 22
1.5 Marco teórico 23
1.5.1 Energía 23
1.5.2 Energía fotovoltaica 24
1.5.3 Inversores de voltaje 25
1.5.4 Batería VRLA 26
1.5.5 Monitoreo de producción de energía eléctrica 28
1.5.6 Monitoreo de condiciones meteorológicas 29
1.5.7 Protocolo de comunicación Modbus 31
1.5.8 Tecnología IoT 32
1.5.9 Bases de datos 33
1.5.10 Servidores Web 35

1.5.11 Aplicaciones GUI 36
1.5.12 CAD electrónico 37
1.5.13 Coeficiente de rendimiento para una instalación fotovoltaica P. R. 38
Capítulo 2 42
2.1 Metodología. 43
2.2 Desarrollo de dispositivo de IoT 44
2.2.1 Diseño electrónico del dispositivo IoT 47
2.2.3 Diseño PCB del dispositivo electrónico y costes 65
2.2.4 Configuración y programación de la ESP32 67
2.3 Diseño de base de datos 72
2.3.1 Configuración de la base de datos 74
2.4 Diseño de la etapa de aplicación GUI 75
2.4.1 Aplicaciones disponibles 76
2.4.2 Autenticación de usuario 78
2.4.3 Menú principal 79
2.4.4 Conexión a Firebase. 80
2.4.5 Presentación de gráficos 81
2.4.6 Entradas de texto para el usuario 82
2.4.7 Configuración de un nuevo dispositivo 83
2.4.7 Widgets 84
2.4.8 Detalle de funciones incorporadas a la aplicación. 85
2.4.9 Producción de energía por hora 86
2.4.10 Producción de energía por día 87
2.4.11 Consultar datos del histórico 87
2.4.12 Calculo de pérdidas y determinación de P. R. 89
2.4.13 Producción real 91
Capítulo 3 93

3.1 Resultados y análisis 94
3.2 Funcionamiento del dispositivo IoT 94
3.2.1 Funcionamiento de la fuente redundante 94
3.2.2 Funcionamiento del módulo de comunicación 98
3.2.3 Funcionamiento del módulo acondicionador de señal 100
3.2.4 Funcionamiento del circuito identificador de la fuente principal 105
3.2.5 Funcionamiento de esclavo Modbus 108
3.3 Funcionamiento de la aplicación 110
3.3.1 Análisis de datos durante un intervalo de tiempo de 8 días 111
3.3.2 Análisis de datos durante 2 días de producción 111
3.3.4 Detalle de costo de elaboración de sistema IoT. 113
Capítulo 4 115
4.1 Conclusiones y recomendaciones 116
4.1.1 Conclusiones 116
4.1.2 Recomendaciones 117
Bibliografía 118
Anexos 120

ABREVIATURAS

ESPOL Escuela Superior Politécnica del Litoral

IoT Internet of Things (Internet de las cosas)

PR Perfomance Ratio (factor de rendimiento)

DC Direct Current (Corriente Directa)

AC Altern Current (Corriente Alterna)

PLC Programmable Logic Controller (Controlador lógico programable)

RTU Remote Terminal Unit (Unidad de terminal remoto)

ID Identification (Identificación)

CRC Cyclic redundancy checking (Comprobación de redundancia cíclica)

GPRS General Packet Radio Service (Servicio general de radio por paquetes)

SQL Structured Query Language (Modelo de datos relacional)

NoSQL No Structured Query Language (Modelo de datos no relacional)

HTTPS Hypertext Transfer Protocol Secure (Protocolo de transferencia de hipertexto seguro)

SSL/TLS Secure Sockets Layer/ Transport Layer Security (Capas de socket seguros/Capas de transporte seguro)

CAD Computer Aided Design (Diseño asistido por computadora)

PBC Printed Circuit Board (Placa de circuito impreso)

MQTT Message Queuing Telemetry Transport

UART Universal Asynchronous Receiver / Transmitter (Receptor/transmisor asíncrono universal)

USB Universal Serial Bus (Bus de serie universal)

TCP/IP Transmission Control Protocol/Internet Protocol (Protocolo de control de transmisión/protocolo de Internet)

ADC Analog Digital Converter (Convertidor analógico digital)

GPIO General Purpose Input/Output (Entrada/salida de propósito general)

TTL Transistor-Transistor Logic (Lógica Transistor-Transistor)

UPS Uninterruptible Power Supply (Sistema de alimentación ininterrumpida)

NTP Network Time Protocol (Protocolo de tiempo de red)

SIMBOLOGÍA

mA	Miliamprerio
Ah	Amperio hora
°C	Grados centígrados
mV	Milivoltio
V	Voltaje
h	Horas
kw/h	Kilovatio/hora
m^2	Metros cuadrados
w/m^2	Vatios/metro cuadrado
Ω	Ohms
$k\Omega$	Kilo ohms
w	Vatio
ms	Milisegundo

ÍNDICE DE FIGURAS

Ilustración 1. Funcionamiento de un panel fotovoltaico. 25
Ilustración 2. Ejemplo de uso de un inversor de corriente en una red domiciliaria. 26
Ilustración 3. Curva de descarga de una batería VRLA. 27
Ilustración 4. Representación de la curva de carga de una batería VRLA. 28
Ilustración 5. Comportamiento de producción de energía de una panta fotovoltaica durante un día de trabajo. 29
Ilustración 6. Representación del comportamiento de la nubosidad durante un periodo de tiempo. 30
Ilustración 7. Representación de la curva de radiación obtenida durante un día. Se aprecia el pico máximo de radiación alrededor de las 12:00 horas. 30
Ilustración 8. Representación de la utilización del protocolo Modbus con varios dispositivos en la industria. 32
Ilustración 9. Descripción de los alcances que brinda la tecnología IoT. 33
Ilustración 10. Diagrama de flujo utilizado para el almacenamiento y lectura de una base de datos. 34
Ilustración 11. Representación gráfica para entender las diferencias entre un Frontend y un Backend. 35
Ilustración 12. Ejemplo de un dashborad desarrollado para el escritorio de una PC. . 36
Ilustración 13. Ejemplo de una aplicación desarrollada para una interfaz móvil. 37
Ilustración 14. Etapas definidas para el desarrollo del sistema de monitoreo. 44
Ilustración 15. Diagrama de bloques que representa las etapas que conforman el dispositivo IoT. 48
Ilustración 16. Fotografía de la batería implementada como respaldo para desconexión de la fuente principal. 49
Ilustración 17. Circuito implementado para establecer la comunicación entre la ESP32 y los sensores de la estación de clima. 49
Ilustración 18. Módulo HW-221 implementado. 50
Ilustración 19. Módulo Rs485 implementado. 51
Ilustración 20. Detalle del circuito del módulo RS485. 51
Ilustración 21. Lógica interna del integrado MAX485. 52
Ilustración 22. Circuito implementado para adaptar y medir el voltaje de la batería. .. 53

Ilustración 23. Circuito implementado para detectar el funcionamiento de la fuente de alimentación principal. 55
Ilustración 24. Descripción de pines utilizados en la ESP32. 56
Ilustración 25. Circuito implementado para conmutar a la batería de respaldo cuando la fuente principal entra en fallo o no hay energía eléctrica. 57
Ilustración 26. Detalle del flujo de corriente cuando la fuente principal está en operación. 58
Ilustración 27. Detalle del flujo de corriente cuando la batería de respaldo entra en operación. 59
Ilustración 28. Circuito regulador de voltaje de 12V a 5V para alimentar la ESP32. ... 59
Ilustración 29. Flujo de corriente desde la etapa reguladora hacia los demás elementos del circuito. 62
Ilustración 30. Layout de la PCB del dispositivo. 66
Ilustración 31. Figura de la PCB del dispositivo. 66
Ilustración 32. Estructura de la partición de la memoria de la ESP32. 69
Ilustración 33. Descripción de la rutina del procesamiento de información. 71
Ilustración 34. Registro de usuarios para utilización de base de datos online. 74
Ilustración 35. Ven/tana de inicio de sesión de usuario. 78
Ilustración 36. Mensaje de error emitido al ingresar credenciales incorrectas. 78
Ilustración 37. Ventana de menú principal de la aplicación. 80
Ilustración 38. Gráfica proporcionada por la librearía matplotlib. 82
Ilustración 39. Ejemplo de las entradas de texto implementadas dentro de la aplicación. 83
Ilustración 40. Ventana de configuración para un nuevo dispositivo 84
Ilustración 41. Widgets utilizados para indicar el funcionamiento de la fuente principal o de la batería de respaldo. 85
Ilustración 42. Widgets utilizados para demostrar la conexión de la aplicación a internet. 85
Ilustración 43. Apartado para visualizar la producción teórica de energía por hora. ... 86
Ilustración 44. Apartado para visualizar la producción teórica de energía por día. 87
Ilustración 45. Apartado para consultar los datos de producción almacenados en el histórico. 88
Ilustración 46. Ventana para realizar consultas de la energía perdida durante intervalos de tiempo de paradas de planta. 89

Ilustración 47. Ventana para consultar el rendimiento de la granja solar. 90
Ilustración 48. Ejemplo de la gráfica obtenida al realizar las consultas para determinar el P. R. de la planta. 91
Ilustración 49. Apartado para ingresar datos de producción real de la granja solar. ... 92
Ilustración 50. Medición de la tensión de entrada proporcionada por la fuente de alimentación principal. 95
Ilustración 51. Medición de la corriente proporcionada por la fuente de alimentación principal. 96
Ilustración 52. Medición de la tensión de entrada proporcionada por la fuente de respaldo. 97
Ilustración 53. Medición de la corriente proporcionada por la batería de respaldo. 98
Ilustración 54. Implementación de un esclavo para probar el funcionamiento del módulo de comunicación en la PCB. 99
Ilustración 55. Visualización de la señal digital producida al transferir datos entre el esclavo implementado y la PCB. 99
Ilustración 56. Datos proporcionados por el esclavo implementado que fueron cargados durante las pruebas de funcionamiento del módulo de comunicación. 100
Ilustración 57. Pruebas de funcionamiento del circuito medidor de voltaje de la batería con un nivel de 12.09V. 101
Ilustración 58. Datos cargados a la base de datos indicando el nivel de carga al 23.6%. 101
Ilustración 59. Pruebas de funcionamiento del circuito medidor de voltaje de la batería con un nivel de 13.35V. 102
Ilustración 60. Datos cargados a la base de datos indicando el nivel de carga al 41.3%. 102
Ilustración 61. Pruebas de funcionamiento del circuito medidor de voltaje de la batería con un nivel de 14.00V. 103
Ilustración 62. Datos cargados a la base de datos indicando el nivel de carga al 63.9%. 103
Ilustración 63. Pruebas de funcionamiento del circuito medidor de voltaje de la batería con un nivel de 14.58V. 104
Ilustración 64. Datos cargados a la base de datos indicando el nivel de carga al 85.6%. 104

Ilustración 65. Circuito identificador de la fuente de alimentación principal emitiendo 3.00V (Fuente principal conectada). 105
Ilustración 66. Base de datos recibiendo el valor de “True” correspondiente a la presencia de la fuente de alimentación principal. 105
Ilustración 67. Aplicación mostrando el widget correspondiente al uso de la fuente de alimentación principal. 106
Ilustración 68. Circuito identificador de la fuente de alimentación principal emitiendo 0V (Fuente principal desconectada). 107
Ilustración 69. Base de datos recibiendo el valor de “False” correspondiente a la ausencia de la fuente de alimentación principal. 107
Ilustración 70. Aplicación mostrando el widget correspondiente al uso de la batería de respaldo. 108
Ilustración 71. Prueba adicional del módulo de comunicación RS485 con un pórtico serial conectado al computador. 109
Ilustración 72. Verificación a través del pórtico serial de los datos enviados entre el esclavo Modbus y la PCB. 109
Ilustración 73. Información subida a la base de datos provenientes del esclavo Modbus. 110
Ilustración 74. Promedios de producción teórico por día obtenidos durante 8 días de producción consecutivos. 111
Ilustración 75. Comportamiento de un día de producción con condiciones meteorológicas muy irregulares. 112
Ilustración 76. Comportamiento de un día de producción con condiciones meteorológicas muy regulares. 113
Ilustración 77. Pruebas de trasmisión del dispositivo instalado en planta. 132
Ilustración 78. Módulos fotovoltaicos instalados en la granja solar. 132
Ilustración 79. Acceso principal a los módulos fotovoltaicos. 133
Ilustración 80. Configuración y visualización de datos en línea cargados por el dispositivo. 133

ÍNDICE DE TABLAS

Tabla 1. Detalle de uso de pines de la ESP32. 55
Tabla 2. Consumo de corriente por cada elemento utilizado. 62
Tabla 3. Consumo de corriente total del dispositivo. 64
Tabla 4. Estimación de propagación de error en base a porcentaje de error de resistores..... 68
Tabla 5. Tipo de valores almacenado por variable de datos. 69
Tabla 6. Configuración de pines de entrada y de salida de la ESP32. 70
Tabla 7. Detalle de utilización de los módulos UART de la ESP32. 70
Tabla 8. Detalle de costos para la implementación del dispositivo. 113

CAPÍTULO 1

1.1 Introducción

En la actualidad, la demanda de energías renovables ha incrementado de forma acelerada debido a las ventajas ambientales que estas pueden presentar frente a otros tipos de producción energética que suelen ser más contaminantes. Por esto, es necesario desarrollar sistemas que permitan maximizar su rendimiento a través del monitoreo continuo de la energía que estas plantas producen y sistemas que permitan estimar los valores de producción en base a las condiciones atmosféricas características del sitio en donde estas plantas se encuentran ubicadas. Así mismo, es necesario facilitar al usuario la interpretación de estos datos ajustándolos a las necesidades que estos puedan presentar, ya sea con la utilización de gráficos referentes a los valores de producción organizados por intervalos de tiempo y dispositivos que recojan y guarden esta información de manera que el usuario pueda acceder a esta desde cualquier sitio dándole la facilidad de tomar acciones correctivas o preventivas basándose en la información recopilada y procesada por estos sistemas.

1.2 Descripción del problema

La empresa Brineforcorp S.A. dedicada a la generación de energía a partir de la implementación de paneles solares carece de un sistema que le permita estimar la producción de energía en base a las condiciones meteorológicas del sitio en donde esta se encuentra ubicada. Esto hace que llevar un control acerca de la estimación de la producción energética de la granja solar sea tediosa y el acceso a esta información se dificulte al no llevar un histórico detallado referente a los datos de producción teórica y real que se generan en la granja y que, a su vez, la determinación del factor de rendimiento de esta sea difícil de obtener.

1.3 Justificación del problema

El presente proyecto busca facilitar la forma en que el usuario obtiene, almacena y presenta información acerca de los valores teóricos y reales de producción diaria de energía, otorgando a la empresa Brineforcorp SA. un sistema que le permita realizar estas actividades basándose en datos meteorológicos obtenidos a través de sensores implementados en la granja solar de tal manera que, la interpretación de esta información sea fácil y rápida y ayude a tomar decisiones referentes al mantenimiento preventivo y/o correctivo de la planta.

1.4 Objetivos

1.4.1 Objetivo general

Implementar un sistema de monitoreo para una granja solar que permita al usuario determinar los valores de producción teórica de energía mediante el uso de tecnología IoT.

1.4.2 Objetivos específicos

- Diseñar un dispositivo IoT que recopile información acerca de las condiciones meteorológicas del sitio en donde se encuentra ubicada la granja solar a través de una comunicación Modbus RS-485.
- Diseñar una aplicación que procese y muestre los datos obtenidos de la granja solar además que de facilidad en la interpretación de estos datos que ofrezcan beneficios a la planta.
- Integrar una base de datos en línea que permita el almacenamiento de la información obtenida de la granja solar para que el usuario acceda a un historial de producción de energía desde cualquier dispositivo con la aplicación instalada.
- Utilizar los parámetros meteorológicos a través de los sensores instalados para que estime los valores de producción teórica diaria de energía y posteriormente calcule el P.R. de la planta.

1.5 Marco teórico

Para la resolución de este proyecto, es necesario empezar a revisar algunos conceptos que permitirán acelerar la resolución del problema definido con anterioridad. Así mismo, ayudarán a delimitar los alcances del proyecto en base a las necesidades del cliente elaborando de esta manera un sistema que cumpla con las expectativas del usuario final. Es necesario revisar la cantidad necesaria de conceptos para comprender cuál es el funcionamiento total del sistema implementado, así como comprender como podrían verse afectados cada uno de los parámetros que conforman el total del sistema ya instalado, así como del sistema a incorporarse como parte fundamental para el monitoreo continuo de la granja solar en mención.

Empecemos tratando los conceptos básicos de producción de energía fotovoltaica ya que, para este caso de estudio y como ha sido mencionado anteriormente, se busca mejorar y facilitar la forma en la que el usuario obtiene y visualizar los valores referentes a producción de energía de la planta.

1.5.1 Energía

Podemos definir a la energía como la capacidad para realizar un trabajo y qué se manifiesta de diversas formas en el universo transformándose de una a otra, pero manteniéndose constante en todo el tiempo cumpliendo así la ley física que afirma que la energía no se crea ni se destruye, solamente se transforma. De esta manera podemos evidenciar las formas más comunes en la que la energía se manifiesta en nuestro entorno y qué de alguna u otra forma puede ser aprovechada con la finalidad de colaborar con las actividades cotidianas que realiza el ser humano, como ejemplos tenemos:

Energía cinética: que es la energía que posee un cuerpo en movimiento y que depende únicamente de su masa y la velocidad con la que ese cuerpo se mueve en el espacio.

Energía potencial: que es la energía que un cuerpo adquiere según su ubicación en un campo gravitacional, por ello, depende de su masa, la altura a la que se encuentra ubicado y de la fuerza gravitacional

Energía térmica: es la energía relacionada a la temperatura de un cuerpo, cuánto más caliente esté un cuerpo mayor energía térmica contiene este.

Energía luminosa: es la energía que se manifiesta en el espacio en forma de ondas electromagnéticas y que comúnmente conocemos como la luz. Mientras más sea la luz que emite un objeto, mayor energía luminosa este posee.

Energía química: es la energía que se encuentra contenida en la estructura molecular de algunos elementos químicos y que puede ser liberada a través de procesos simples o complejos dependiendo de la composición de estos.

Energía nuclear: Es la energía que se libera a través de procesos como la fusión nuclear en donde los átomos se combinan para formar un átomo nuevo o la fisión nuclear en dónde los átomos se dividen para formar dos o más átomos.

1.5.2 Energía fotovoltaica

Consiste en la obtención de energía eléctrica proveniente de la radiación solar y se obtiene a partir de la excitación de un elemento semiconductor que conforma la estructura de los paneles fotovoltaicos y que, a su vez, esta excitación de los electrones del elemento semiconductor produce una diferencia de potencial DC que dependiendo de su configuración, ya sea en serie o en paralelo, se pueden configurar para alcanzar potenciales y/o corrientes mucho más altas dando la posibilidad al usuario de obtener valores de voltaje y corriente que se ajusten a sus requerimientos (Pep Puig, 2007, págs. 1-2).

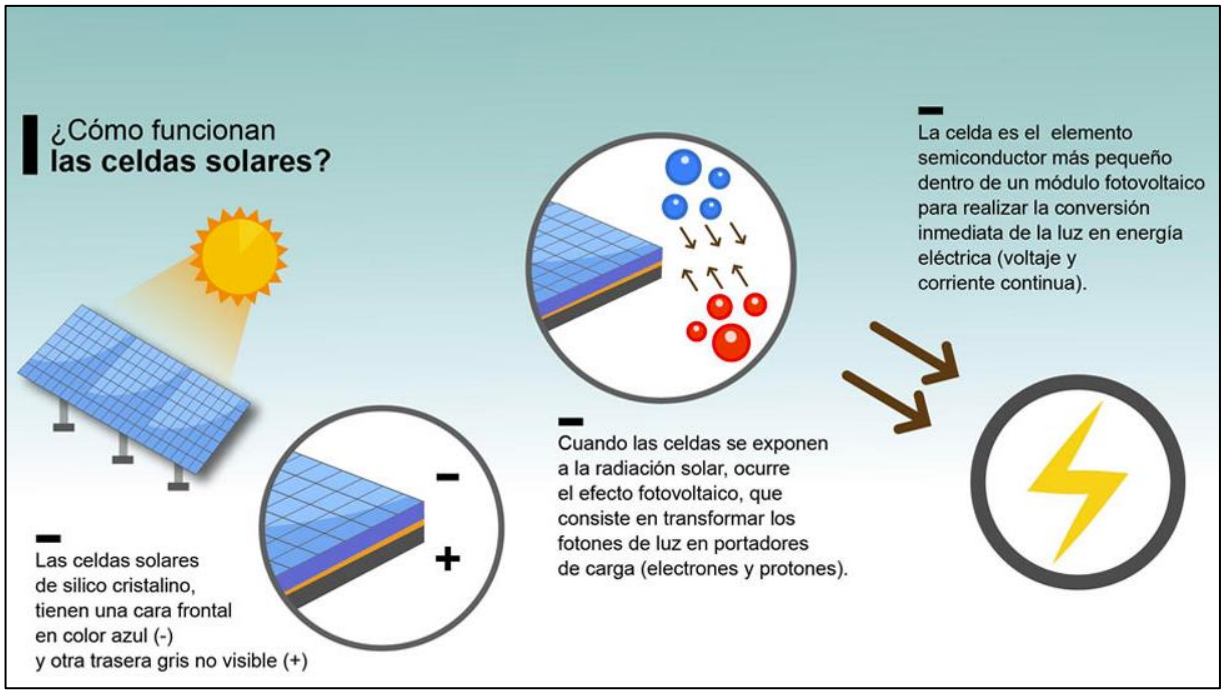


Ilustración 1. Funcionamiento de un panel fotovoltaico.

1.5.3 Inversores de voltaje

Básicamente, los inversores de corriente son equipos cuya función es la de convertir la corriente AC en DC o DC en AC. Para el caso en mención sabemos que los paneles fotovoltaicos utilizados en la granja solar proporcionan corriente continua DC. Sin embargo, para poder inyectar esa corriente a la red interconectada es necesario transformarla en AC y sincronizarla de tal manera que se incremente la capacidad de suplir la demanda actual en cuanto al consumo de energía se refiere y, sobre todo, esta energía pueda ser utilizada por la población con los mismos equipos que se cuentan sin necesidad de implementar otros sistemas para aprovechar esta energía (Remus Teodorescu, Marco Liserre, & Pedro Rodr´ıguez, 2011). Es necesario conocer cuál es la potencia a la cual operan nuestros paneles debido a que no todos los inversores tienen la misma capacidad de operación y este es un factor muy importante para tener en cuenta al momento de implementar un sistema de captación de energía solar.

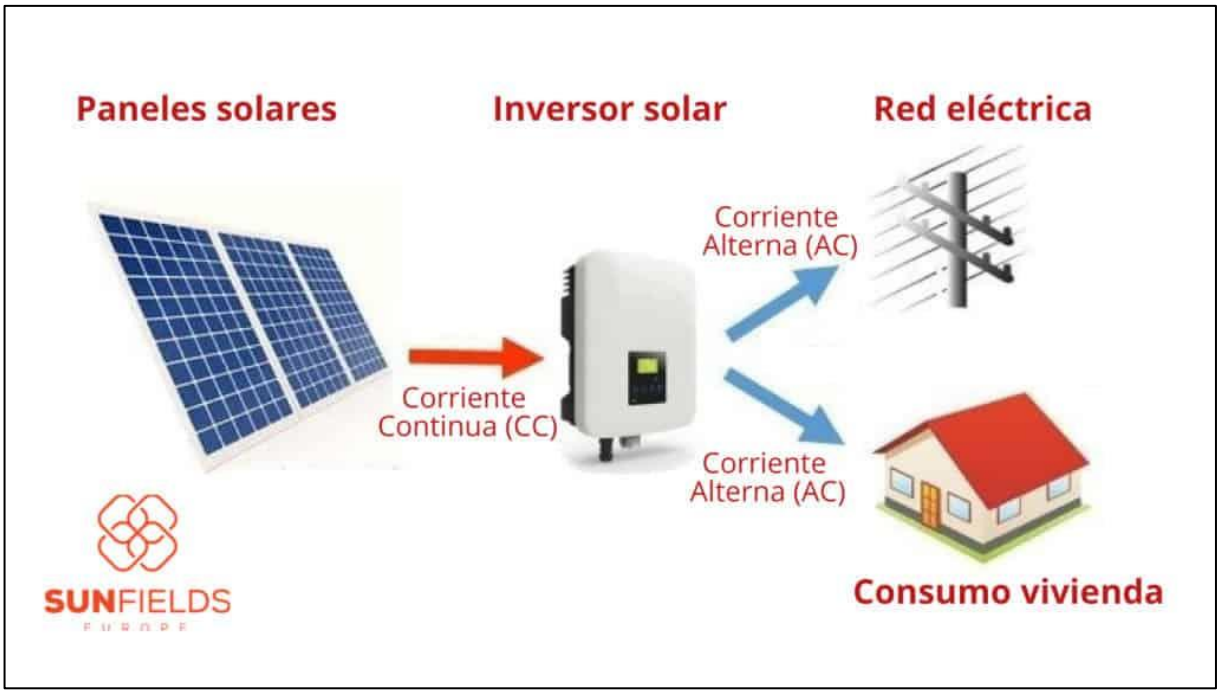


Ilustración 2. Ejemplo de uso de un inversor de corriente en una red domiciliaria.

1.5.4 Batería VRLA

La batería VRLA conocida también como batería de ácido-plomo regulada por válvula, es una batería recargable seca, esto la hace no inflamable y robusta para funcionar sin problemas en entornos hostiles. Además, tiene bajo costo de fabricación y es muy utilizada en tecnologías aplicadas a aparatos eléctricos portátiles de gran tamaño y de bajo mantenimiento.

Por otro lado, las consideraciones ambientales de batería son mantener nivel de humedad y temperatura controladas, siendo una temperatura recomendad de 25℃, además de evitar variaciones bruscas de la misma, aquello permite extender su tiempo de vida útil.

Proceso de carga y descarga de la batería VRLA

En un ciclo de trabajo las baterías inician con una tensión superior su valor nominal que en el mayor número de aplicaciones suelen ser 3V, 5V, 9V, 12V, 15V y 24V. Dependiendo de la carga las baterías VRLA tienen ciclo de trabajo de 5%, 10%, 20%; además, un parámetro importante

que se toma en cuenta es la carga de la batería que se mide en amperios hora (Ah) que equivale a 3600 Culombios. Donde mayor sea la carga de esta, más tiempo en necesario para la carga y descarga de batería. En la curva se puede identificar el proceso de descarga (WEG, 2019):

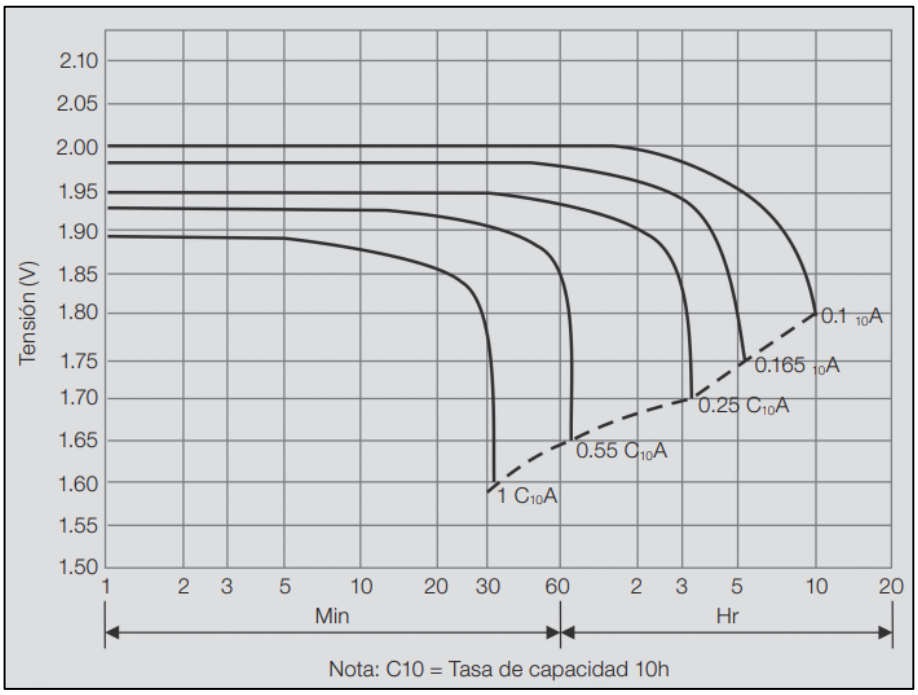


Ilustración 3. Curva de descarga de una batería VRLA.

En la *Ilustración 3* se puede observar el comportamiento de voltaje de celda en función del tiempo para distintas corrientes de consumo; además, identificamos distintos ciclos de trabajo. Cabe recalcar que los ciclos de trabajo suelen depender de la carga, pero en algunas aplicaciones para obtener mayor duración de descarga se usa el ciclo de carga mayor.

El proceso de carga de una batería VRLA consiste en suministrar baja corriente inicial a voltaje constante para luego incrementar la corriente suministrada a menor voltaje. Estos parámetros dependen de la capacidad de la batería y del ciclo de trabajo (WEG, 2019).

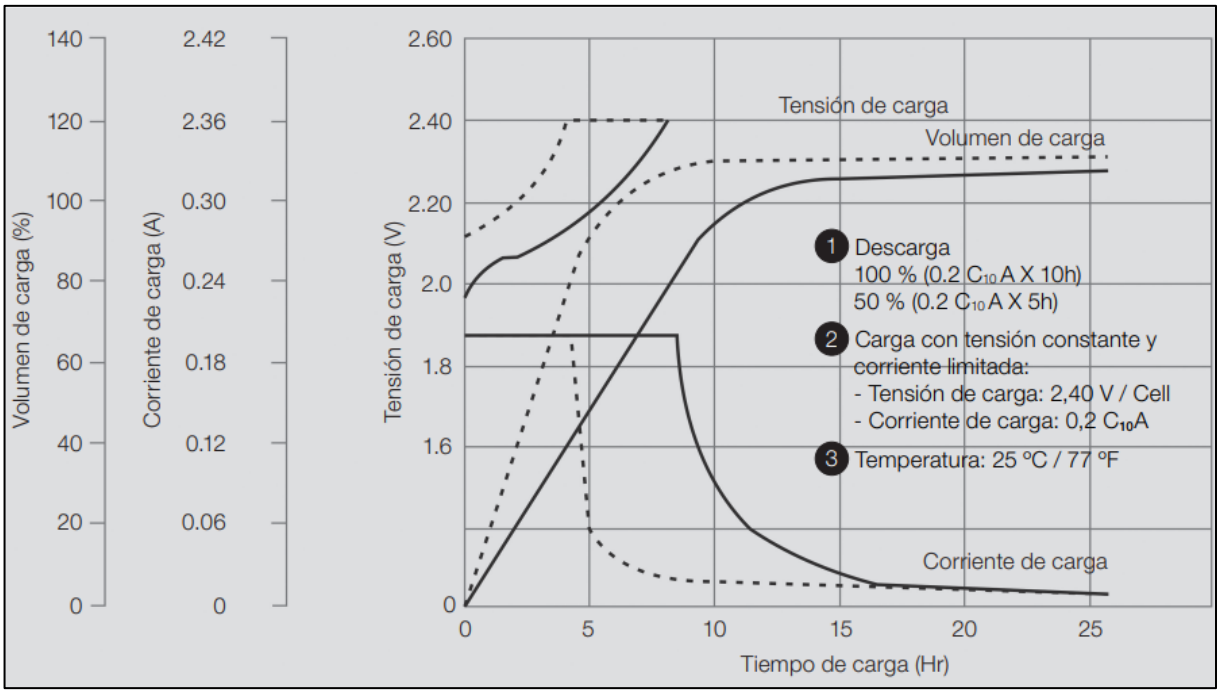


Ilustración 4. Representación de la curva de carga de una batería VRLA.

1.5.5 Monitoreo de producción de energía eléctrica

Consiste en el monitoreo de parámetros de producción energética que sirven como referencia de cuál es la potencia que se está inyectando a la red eléctrica nacional o de un área local. La mejor forma de monitorear este parámetro es en tiempo real, en donde se puede observar la curva de producción energética, que se espera que en el mejor de los casos y condiciones casi constantes la curva se represente como una campana en donde los picos de producción se den mientras el sol se ubique casi perpendicular al panel instalado y esto dependerá mucho de la localización geográfica de la granja solar (Kalogirou, 2009).

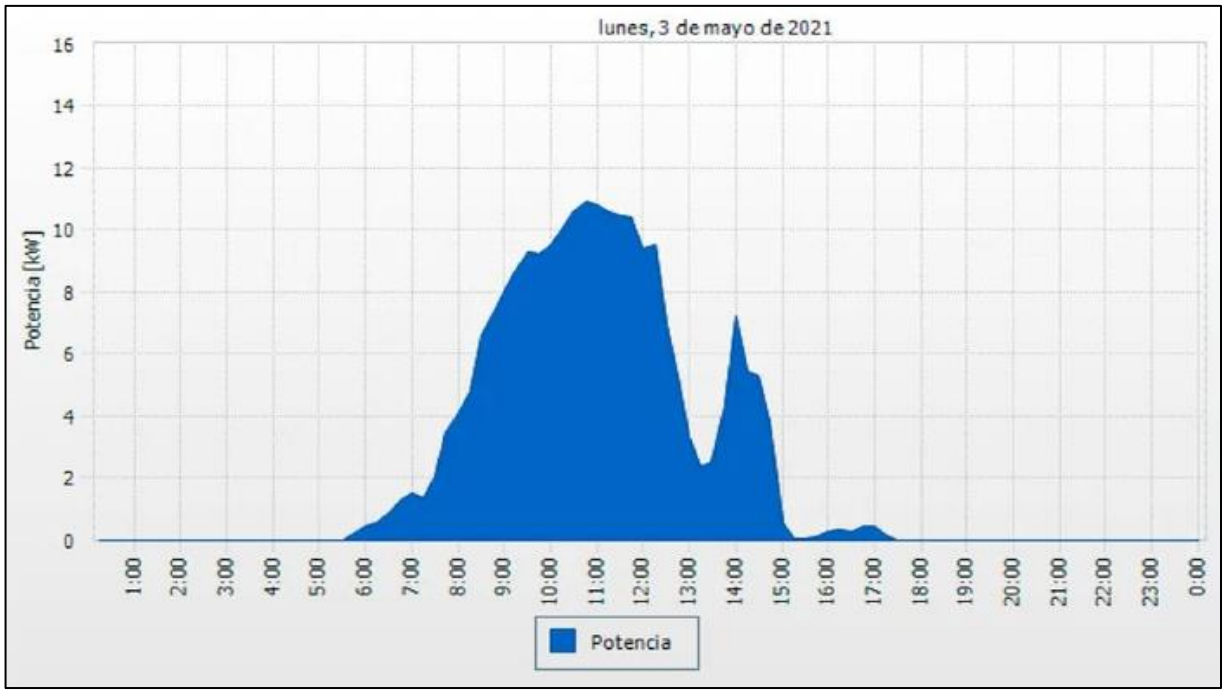


Ilustración 5. Comportamiento de producción de energía de una panta fotovoltaica durante un día de trabajo.

1.5.6 Monitoreo de condiciones meteorológicas

Para una granja solar es importante monitorear cuáles son las condiciones de clima del sitio en donde esta se encuentra ubicada, pues, sabemos que en condiciones ideales en donde el cielo se encuentra despejado la incidencia de la radiación solar es alta lo que a su vez se traduce como una mayor producción de energía por cada panel instalado. Sin embargo, existen factores como, la velocidad del viento, la temperatura de cada panel y la presencia de nubes en el sitio que pueden reducir la capacidad de generar energía a cada panel, dando como resultado una baja eficiencia de estos. Consiste en el muestreo y almacenamiento de parámetros como radiación solar, la temperatura, la velocidad del viento, en cual permite predecir la producción de energía y detectar ciertas anomalías.

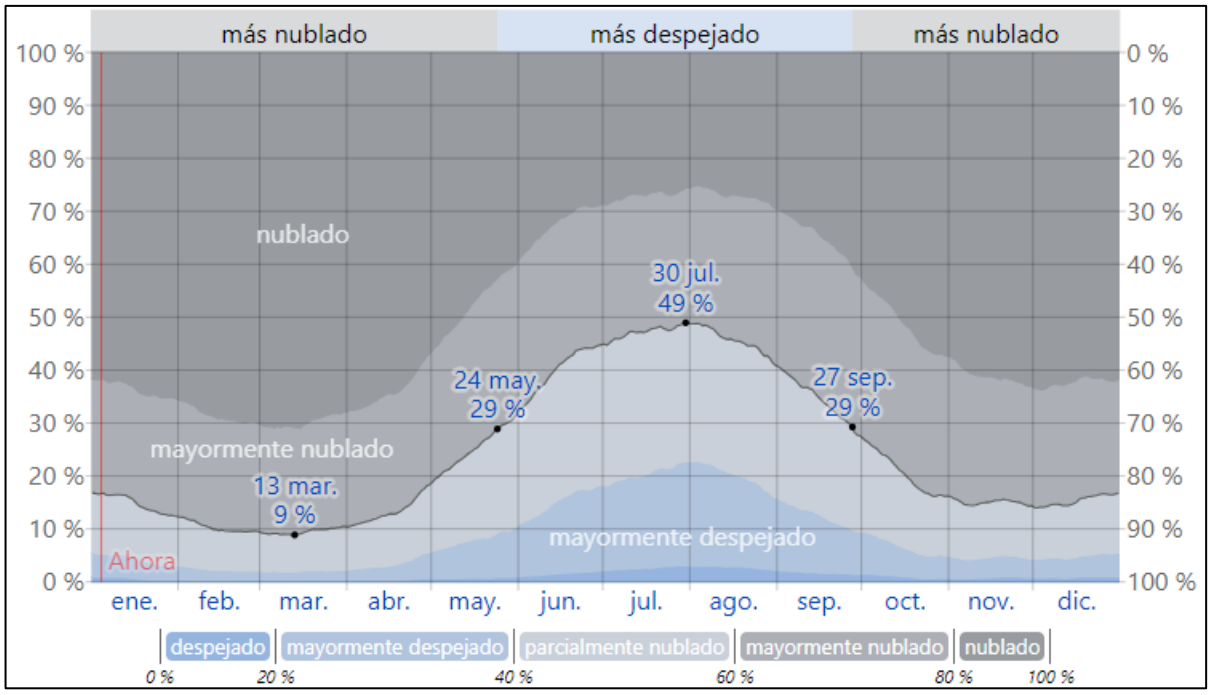


Ilustración 6. Representación del comportamiento de la nubosidad durante un periodo de tiempo.

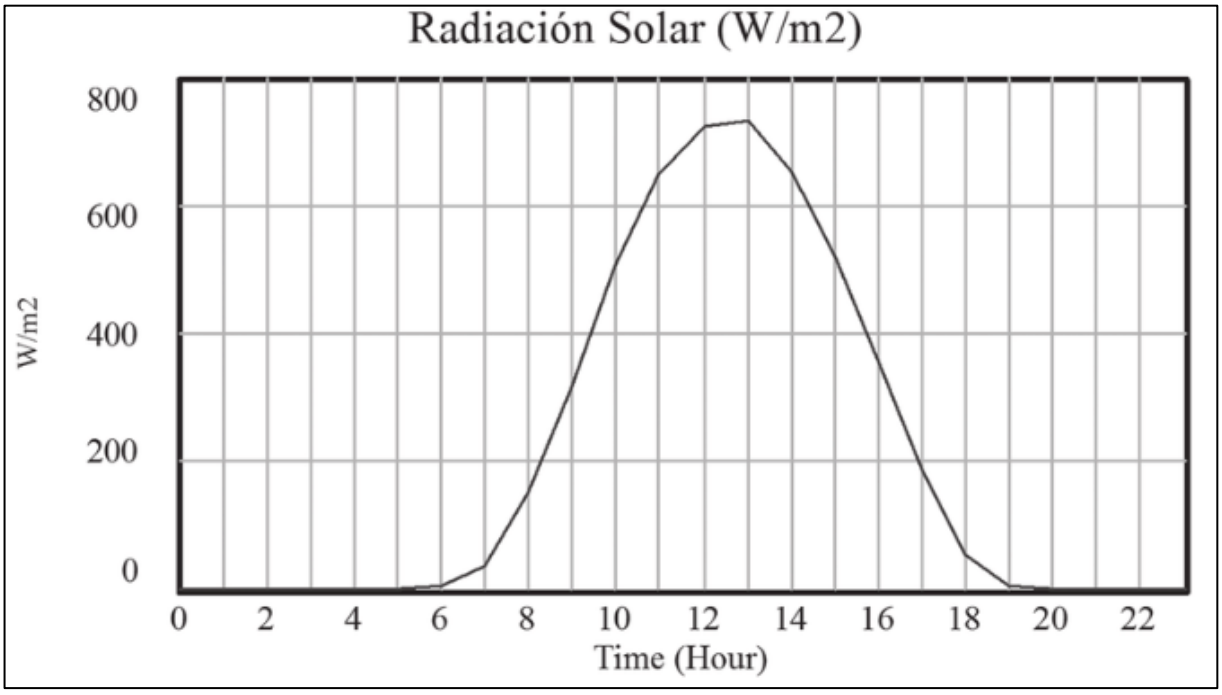


Ilustración 7. Representación de la curva de radiación obtenida durante un día. Se aprecia el pico máximo de radiación alrededor de las 12:00 horas.

En las *Ilustraciones 6 y 7* adjuntas, podemos observar un ejemplo de dos variables meteorológicas que son de suma importancia para la producción de energía a través de granjas solares, en este caso, vemos como el comportamiento de la curva de radiación solar, en condiciones estables, tiene su pico cuando el sol se ubica en lo más alto (Para una ciudad situada en la zona ecuatorial) y decae en los externos cuando el sol se oculta y cuando empieza a salir. En este caso, se espera que los picos de producción para un comportamiento de irradiación solar similar al gráfico adjunto sean alrededor de las 11:00 horas y las 13:00 horas. De igual manera, se observa la *Ilustración 6* en dónde en base la presencia de nubes se puede estimar los meses en donde la producción de energía puede ser baja y en los que la producción de energía pueda ser alta.

1.5.7 Protocolo de comunicación Modbus

Es un protocolo desarrollado por Modicon (Schneider Electric) en 1979, mismo que es utilizado en sistemas de control y automatización industrial para transmitir datos entre los dispositivos que conforman un sistema de control. En la actualidad es un estándar de comunicaciones de distintos dispositivos tales como PLC, sensores, actuadores, entre otros. Modbus RTU (Remote Terminal Unit), es una variación del protocolo original, este es usado en las interfaces de comunicación serial como RS-232 o RS-485; además, transmite los datos en formato binario a una velocidad que va desde los 1200 baudios hasta los 115,200 baudios. Por otro lado, los mensajes transmitidos tienen una estructura especifica en donde se componen del ID de esclavo, código de función (operación a realizar en el esclavo), datos y CRC (verifica la integridad del mensaje enviado).
De entre las ventajas más relevantes que tiene este protocolo de comunicación está la capacidad de conectarse en multipunto a diferencia de RS-232.

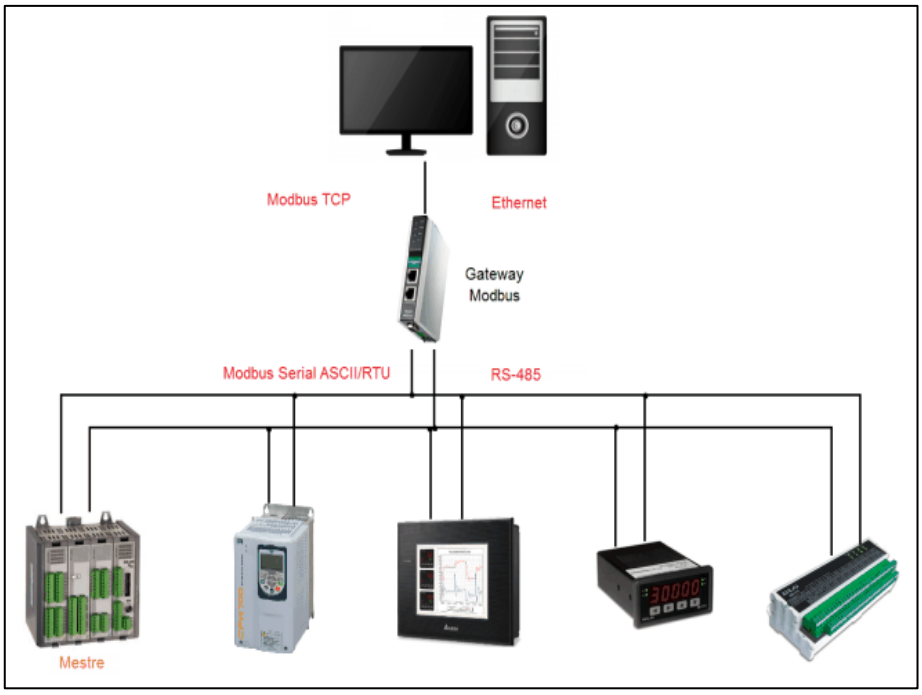


Ilustración 8. Representación de la utilización del protocolo Modbus con varios dispositivos en la industria.

1.5.8 Tecnología IoT

Es un sistema de interconexión de dispositivos conocidos como dispositivos de IoT que se comunican a través del internet entre sí; además, es una integración de hardware y software que otorgan al usuario la capacidad de muestrear, enviar, recibir, configurar o monitorear el comportamiento de un sistema a través de una conexión a internet. En sistemas complejos el intercambio de esta información entre dispositivos brinda al usuario la capacidad de controlar sistemas enteros conformados por actuadores que se accionan en tiempo real y emiten señales de respuesta que pueden contener información acerca de otras variables derivadas de la ejecución de una determinada acción. Por otro lado, los dispositivos de IoT de forma general suelen utilizar protocolos de comunicación inalámbrica como Wi-Fi, Bluetooth, Zigbee, GPRS, LoraWAN, entre otros; esto permite al usuario y al mismo sistema IoT prescindir de cableado que en ocasiones puede sufrir daños físicos sobre todo si estos están situados en lugares abrasivos o a la

intemperie. Así mismo, la ausencia de cableado abarata en cierta magnitud los costos de instalación de estos sistemas abarcando largas distancias con costos reducidos (López Garzón, W. & Cárdenas López, J., 2019).

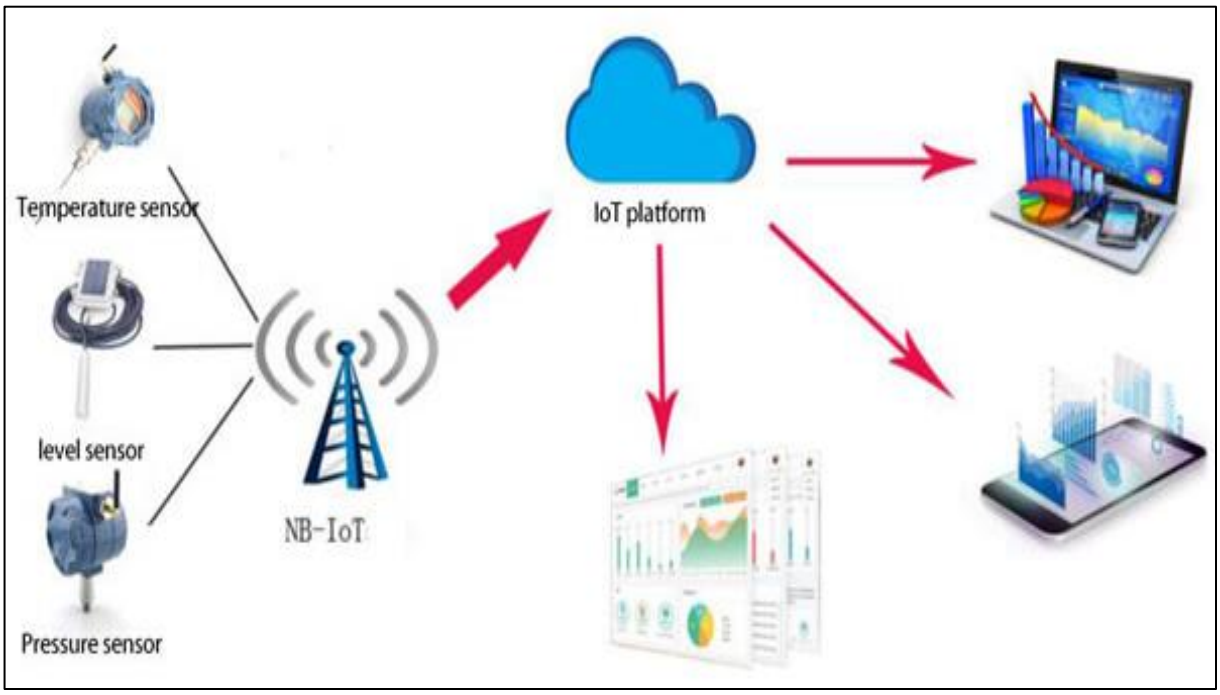


Ilustración 9. Descripción de los alcances que brinda la tecnología IoT.

1.5.9 Bases de datos

Las bases de datos son modelos y estructuras lógicas que se utilizan para almacenar, representar y organizar información proveniente de un proceso de producción o simplemente como un respaldo de información relevante para la optimización en el funcionamiento de una empresa (Rafael Camps Paré, y otros, 2005). La variedad de bases de datos existentes hace que el usuario pueda elegir la más adecuada para su sistema dependiendo de las necesidades de cada uno y del uso que se plantea dar a estás bases de datos, ya que en ocasiones existe el interés de almacenar solamente datos numéricos o strings de información y en otras ocasiones existe la necesidad de almacenar contenido multimedia. De entre las bases de datos más conocidas y utilizadas tenemos:

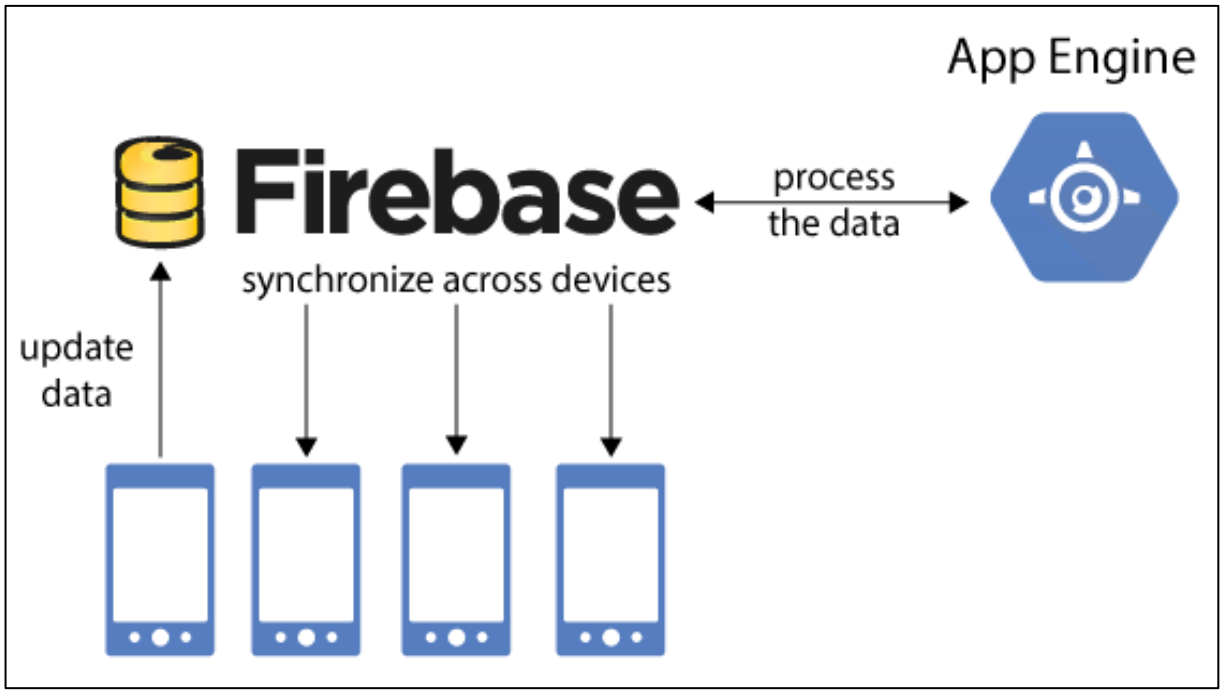


Ilustración 10. Diagrama de flujo utilizado para el almacenamiento y lectura de una base de datos.

Modelo de datos relacional (SQL): es un modelo donde los datos se organizan en tablas, donde cada tabla representa una entidad y se relaciona entre entidades a través de claves primarias o foráneas (Rafael Camps Paré, y otros, 2005).

Modelo de datos no relacional (NoSQL): es un modelo donde el almacenamiento de los datos es más flexible y tiene múltiples formas tales como, documentos, grafos, key-value, etc.

Modelo de datos jerárquico: es un modelo donde los datos se organizan en estructuras como árbol con nodos y subnodos. Estos son ampliamente utilizados en sistemas de archivos.

Modelo de datos de Red: este modelo organiza los datos en múltiples nodos, donde cada nodo está relacionado con otros nodos lo que es útil para representar datos interconectados.

Modelo de datos orientado a objetos: este modelo relaciona un objeto del mundo real a un objeto virtual en la que se almacena y se relacionan usando la lógica de la programación orientada a objetos con atributos y métodos (Rafael Camps Paré, y otros, 2005).

1.5.10 Servidores Web

Los servidores web son programas en donde se alojan distintos servicios que pueden ser solicitados por clientes tales como: páginas web, archivos, aplicaciones web, recursos, entre otros (Bonaventure, 2011, pág. 20). Además, los servidores web pueden ser programados por lenguajes como PHP, Python, Ruby, y JavaScript, donde su integración permite regular, administrar y priorizar las peticiones de distintos clientes (usuarios), esto se conoce como backend, y en función de las necesidades del servidor web se manejan distintos protocolos, donde el protocolo más utilizado es HTTPS (Hypertext Transfer Protocol Secure) donde son ejecutados en navegadores muy conocidos como Chrome o Microsoft Egde (Bonaventure, 2011, pág. 46). Los servidores web públicos tienen múltiples amenazas de manipulación no adecuada de los servicios e información personal; por lo que es necesario desarrollar múltiples métodos de seguridad como los certificados SSL/TLS capaces de habilitar conexiones seguras en la que se cifra la información transmitida. Algo muy utilizado en el protocolo HTTPS (Bonaventure, 2011, pág. 52).

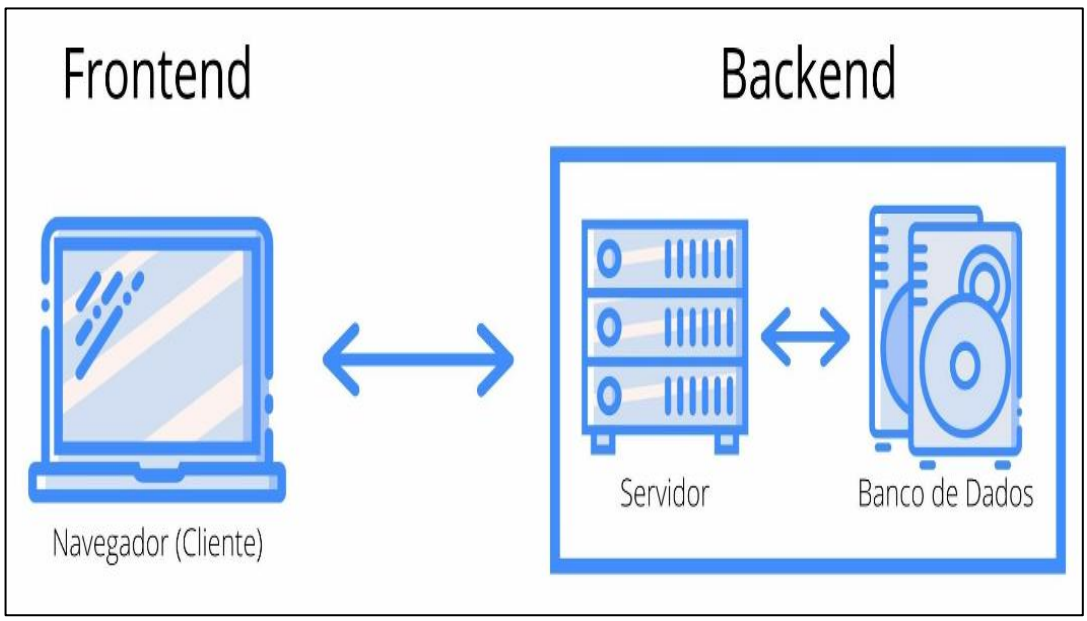


Ilustración 11. Representación gráfica para entender las diferencias entre un Frontend y un Backend.

1.5.11 Aplicaciones GUI

Son programas que permiten la interacción del usuario con el hardware de un computador a través de interfaces gráficas (Shneiderman, 2004). Estas interfaces gráficas pueden instalarse en sistemas operativos muy utilizados como Windows, Linux o MacOS; también pueden estar diseñadas para ejecutarse desde sistemas operativos móviles como Android o iOS. Sin embargo, existen aplicaciones que son desarrolladas en entornos web para facilitar el acceso a través de un servidor desde cualquier sitio sin necesidad de tener instalado su ejecutable.

El desarrollo de estas aplicaciones puede efectuarse utilizando distintos lenguajes de programación y esto dependerá específicamente de los alcances del proyecto, su complejidad y la experiencia con la que cuente el programador (Shneiderman, 2004). Como un ejemplo para el desarrollo de aplicaciones de escritorio tenemos: C++, Java, C#, Python, Swift, etc. Estos son algunos de los lenguajes para desarrollo de aplicaciones para escritorio más utilizados en la actualidad.

Ilustración 12. Ejemplo de un dashborad desarrollado para el escritorio de una PC.

Para el desarrollo de aplicaciones móviles se pueden utilizar java, kotlin, Swift, Objective-C, y Flutter. Estas SDK de sus siglas en inglés Software Development Kit, permiten al desarrollador diseñar programas para ser ejecutados en los sistemas operativos más comunes como Android y iOS.

Ilustración 13. Ejemplo de una aplicación desarrollada para una interfaz móvil.

1.5.12 CAD electrónico

Para el diseño electrónico de un circuito, se pueden utilizar varios programas que permiten simular estos circuitos y revisar que todo funcione correctamente antes de implementarlos en su totalidad. Por ejemplo, podemos simular un circuito regulador de voltaje utilizando los componentes adecuados y verificar que el resultado que este arroje como, voltajes, corrientes, etc. Sean adecuados y se comporten de la manera que nosotros lo hayamos planificado. De entre los programas disponibles para diseñar circuitos electrónicos tenemos:

Proteus: este software es muy utilizado en el ámbito académico por ser un software gratuito y la amplia disponibilidad de modelos de componentes electrónicos que se disponen en internet;

además, permite al usuario diseñar circuitos electrónicos, visualizarlos en 3D y realizar pruebas en tiempo real con el circuito implementado. Además de no solo simular los modelos elaborados, sino también tener la ventaja de ejecutar el código utilizado dentro de un microcontrolador que se haya utilizado en el modelo.

Eagle: a diferencia de Proteus, Eagle tiene limitaciones en su versión gratuita; sin embargo, este software en su versión de pago ofrece al usuario una gran variedad de funciones que facilitan el diseño e implementación de los circuitos electrónicos elaborados. A diferencia de Proteus, Eagle al formar parte de Autodesk funciona correctamente con otras herramientas que forman parte de esta plataforma.

KiCad: al igual que Proteus, permite diseñar circuitos electrónicos y visualizar sus PCB en 3D, sin embargo, la cantidad de elementos electrónicos disponibles dentro de esta plataforma no es tan amplia por lo que, diseñar e implementar circuitos electrónicos complejos utilizando esta herramienta no es tan recomendable. Sin embargo, es una plataforma perfecta para empezar en el mundo del diseño de circuitos electrónicos.

1.5.13 Coeficiente de rendimiento para una instalación fotovoltaica P. R.

El coeficiente de rendimiento P.R. por sus siglas en inglés (Performance Ratio) es una variable determinante al momento de evaluar la efectividad de una instalación fotovoltaica. Pues esta se encarga de medir la relación entre el rendimiento real de la planta fotovoltaica y el rendimiento teórico obtenido en base a los parámetros meteorológicos de esta. Esto ayuda a realizar comparaciones entre el rendimiento de múltiples plantas fotovoltaicas ubicadas en distintos lugares del mundo (SMA Solar Technology AG, pág. 2). El P. R. de una planta fotovoltaica se expresa en términos porcentuales y, al relacionar el rendimiento real con el rendimiento teórico de una instalación fotovoltaica esta permite al usuario determinar qué porcentaje de la energía disponible en un área delimitada está siendo aprovechada realmente para ser inyectada a una red

eléctrica habiendo ya descartado las perdidas energéticas causadas por pérdidas térmicas y por pérdidas ocasionadas por el cableado.

No está demás aclarar que mientras el P.R. de la instalación fotovoltaica se encuentre más cercano al 100% esto indicará que nuestra instalación fotovoltaica trabajará de forma más eficiente. Sin embargo, hasta la actualidad no es posible alcanzar un coeficiente de rendimiento igual al 100% debido a que siempre se generarán perdidas por los factores antes mencionados, pero es muy probable elevar la eficiencia de la instalación fotovoltaica utilizando materiales de calidad para la obtención y trasporte de la energía obtenida durante la operación de estas instalaciones.

El coeficiente de rendimiento nos brinda información acerca de la eficiencia energética y la efectividad de nuestra instalación fotovoltaica por lo que, el usuario puede planificar mantenimientos preventivos o mantenimientos correctivos basándose en este valor obtenido ya qué el coeficiente de rendimiento puede decaer dependiendo del estado físico de los paneles y los dispositivos que conforman el sistema de captación y transporte de energía hacia la red eléctrica (SMA Solar Technology AG, pág. 2).

El periodo óptimo para el cálculo del P.R. de una instalación fotovoltaica es de 1 año; sin embargo, podemos realizar comparaciones en tiempos mucho más cortos para evidenciar cómo se comporta la instalación fotovoltaica durante algunas condiciones climáticas que pueden representar un problema para la captación de energía en el sitio (SMA Solar Technology AG, pág. 3).

Para calcular el P.R. de una instalación fotovoltaica hacemos uso de la siguiente fórmula.

$$P.R. = \frac{Rendimiento\ real\ leído\ de\ la\ instalación\ fotovoltaica\ expresada\ en\ kw/h}{Rendimiento\ teórico\ de\ la\ instalación\ fotovoltaica\ expresada\ en\ kw/h}$$

En donde se detalla cómo se obtiene cada una de las variables a continuación:

El rendimiento real leído de la instalación fotovoltaica se lo obtiene de los medidores instalados.

El rendimiento teórico de la instalación fotovoltaica depende de múltiples factores, principalmente de valores físicos de la granja solar y son los siguientes:

Período de estudio: equivale al periodo de tiempo en que se quiere determinar el factor de rendimiento.

Superficie del generador de la instalación fotovoltaica (A): se refiere al área total de paneles instalados en la planta solar.

Rendimiento de los módulos fotovoltaicos (RM): Este valor es propio de los paneles implementados y viene especificado en las hojas de datos de cada uno.

Irradiación solar (Irr): hace referencia a los valores de radiación obtenidos por los sensores instalados en la planta en un determinado periodo de estudio y expresados en kwh/m^2.

Con estos valores obtenemos la siguiente fórmula:

$$Rendimiento\ teórico = Irr * A * RM$$

Para finalmente al reemplazar esta fórmula y obtener:

$$P.R. = \frac{Rendimiento\ real\ en\ kw/h}{Irr * A * RM} * 100$$

Que determinará cual es el porcentaje de energía aprovechada durante el periodo de tiempo estudiado.

Existen varios factores que pueden afectar el rendimiento de los paneles fotovoltaicos, tal es el caso de:

Temperatura de los módulos fotovoltaicos: es necesario mencionar que, el coeficiente de rendimiento de los paneles fotovoltaicos es calculados en condiciones estándar con irradiación de 1000 w/m^2 y a una temperatura de los módulos de 25 $°C$.

Sombra o suciedad de los módulos fotovoltaicos: que se puede dar en ambientes con demasiada vegetación o condiciones extremas de polvo en el lugar geográfico en donde se encuentran instalados.

Sombra o suciedad de la estación de medición: al igual que mantener los módulos fotovoltaicos limpios es importante mantener en las mismas condiciones a la estación de medición. De no hacerlo, las lecturas de radiación solar del sitio pueden ser muy bajas dando un P.R. muy elevado y viceversa.

Calidad de los equipos implementados: si bien es cierto que existen muchas marcas con equipos que ofrecen realizar el mismo trabajo con costos más bajos, no siempre son equipos que mantienen una eficiencia elevada, por lo que las pérdidas en dispositivos no confiables pueden ser mayores obteniendo factores de rendimiento muy bajos (SMA Solar Technology AG, pág. 9).

Diferencias entre las tecnologías implementadas en los módulos fotovoltaicos: puede darse en casos en donde se han implementado diferentes tecnologías para captación de energía. Por ejemplo, añadir paneles con mejor rendimiento a un conjunto de paneles con un bajo rendimiento.

Orientación de los módulos fotovoltaicos: hace referencia en cómo están ubicados los módulos fotovoltaicos en una estación solar.

Tiempo de vida de los paneles fotovoltaicos: este valor está sujeto a la tecnología que haya sido implementada en los paneles adquiridos y su durabilidad dependerá estrictamente de esto. En el caso de células monocristalinas y policristalinas pueden envejecer hasta un 20% en 20 años (SMA Solar Technology AG, pág. 9).

CAPÍTULO 2

2.1 Metodología.

Una vez teniendo en claro cuáles son las herramientas disponibles para desarrollar el sistema de monitoreo procedemos a dividir el proyecto para ser implementado en 3 etapas distintas cada una orientada a realizar un trabajo en específico, pero que se complementan para formar un sistema que funciona en conjunto. Si bien es cierto que se ha priorizado cada etapa del proyecto en base a la importancia que cada uno genera, el tiempo destinado al desarrollo de cada una de estas etapas no es proporcional al orden de ejecución, pues durante su implementación se han generados nuevos requerimientos que han sido integrados al sistema para elaborar un producto acorde a las necesidades del cliente final.

Desarrollo del dispositivo IoT.

Se le ha dado la prioridad a esta etapa debido a que es necesario entablar una comunicación entre los sensores instalados en la planta fotovoltaica y el dispositivo a utilizar durante el desarrollo del proyecto para de esta manera obtener los valores que se están midiendo con la estación de clima, mismos que son importantes para determinar el estimado teórico de producción de energía.

Implementación de una base de datos en línea.

Esta etapa busca las herramientas necesarias para almacenar los valores obtenidos con el dispositivo IoT y acceder a ellos desde cualquier sitio, por este motivo no se ha elegido trabajar con una base de datos local dado que el acceso a esta información debe realizarse en ocasiones de forma remota.

Desarrollo de una aplicación de escritorio.

Esta aplicación está destinada a procesar la información almacenada en la base de datos y mostrarla de tal manera que al usuario le sea fácil interpretar los datos que se han obtenido y en base a ellos pueda determinar o planificar mantenimientos preventivos y/o correctivos.

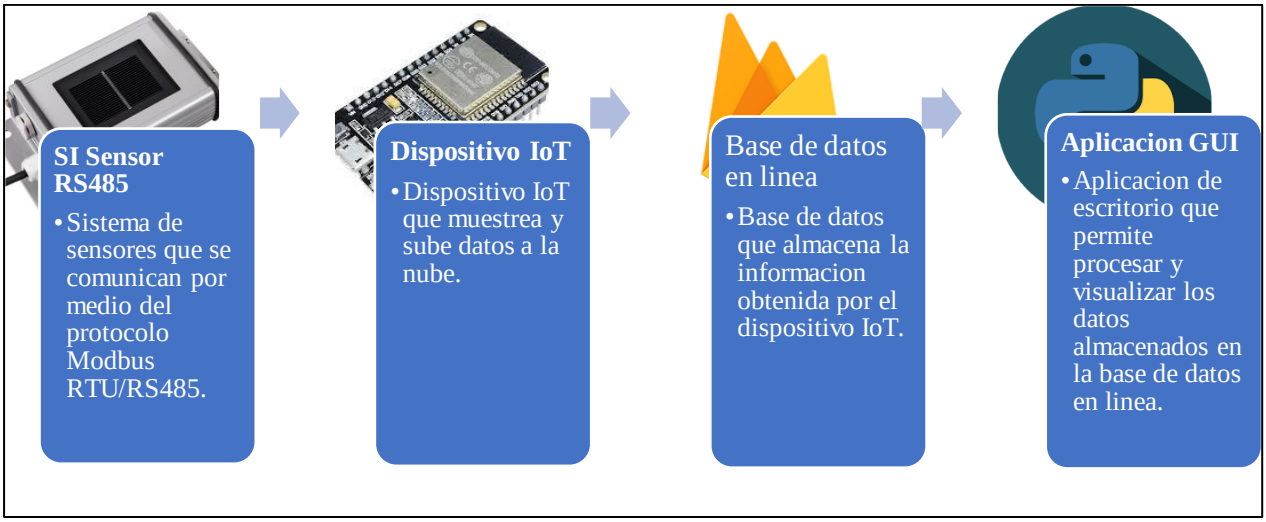


Ilustración 14. Etapas definidas para el desarrollo del sistema de monitoreo.

2.2 Desarrollo de dispositivo de IoT

Tal como se mencionó anteriormente, el desarrollo del proyecto comenzó con el diseño y la implementación del dispositivo IoT que permitiría comunicarse con los sensores de la estación de clima instalado en la granja solar y obtener los datos que estos sensores arrojaban utilizando un protocolo de comunicación RS-485/Modbus RTU. El trabajo asignado al dispositivo se enfoca básicamente en la recopilación de información establecida por un tiempo de muestreo y enviar esta información hacia la base de datos en línea. De esta forma se logra respaldar esta información para luego ser utilizada a conveniencia por parte del cliente final. De entre las opciones disponibles para el diseño e implementación de este dispositivo, se estudió las mencionadas a continuación:

- Integrar la comunicación Modbus RTU a un servidor MQTT. Esto consiste en programar un Brocker/ servidor MQTT, y por medio de un dispositivo Gateway integrar el bus RS-485 a TPC/IP.

  Ventajas:

  - Tiene soporte multiplataforma, es decir se puede acceder a los datos desde cualquier cliente.
  - Comunicación en tiempo real.
  - Escalabilidad, puede integrar varios dispositivos y configurar distintos protocolos.

  Desventajas:

  - Configuración adicional para la conversión de protocolos de Modbus RTU a MQTT.
  - Uso necesario de software o hardware compatible con MQTT.

- Necesita de plataformas adicionales para ser integrados en base de datos u otras aplicaciones.

- Utilizar un ordenador para a través del uso de un nodo de red (Node Red) para programar las peticiones a los dispositivos Modbus utilizando bloques de comunicación serial UART o bloques Modbus, y posteriormente cargar los datos a la nube en línea.

  Ventajas:
  - Node Red es compatible con distintos protocolos.
  - Node Red es flexible en adaptar datos de un protocolo y enviarlos por medio de otro protocolo.

  Desventajas:
  - El uso de equipos adicionales como un computador y módulos extras (Gateway o converters) que permita cambiar la interfaz RS-485 a USB o TPC/IP.
  - El diseño de los bloques Modbus RTU o el uso de las bibliotecas aumenta la complejidad al momento de adquirir los datos del conjunto de sensores.

- Integrar un diseño propio utilizando un microcontrolador, programando la interfaz que permita cargar los datos de los sensores directamente a la plataforma que se utilizaría como servidor o base de datos.

  Ventajas:
  - Tiene las flexibilidades en el uso del protocolo y en el procesamiento de datos.
  - Bajo costos de fabricación e implementación.
  - Es apropiado para proyectos en pequeña escala.
  - Se puede implementar de forma inalámbrica.

  Desventajas:
  - Se debe tener varios puntos en cuenta para tener eficacia en el sistema.
  - Se debe tener cuenta las limitaciones de hardware.

Luego de analizar cada opción disponible y presentada, se optó por desarrollar un diseño propio utilizando dispositivos disponibles en el mercado y considerando cuidadosamente los siguientes aspectos:

- Flexibilidad: el programar un microcontrolador permite adaptarlo a los requerimientos del usuario final en cuanto a información relevante se refiere; además, permite modificar su

funcionamiento en base a los resultados obtenidos con el fin de maximizar el aporte que la información recopilada pueda otorgar al usuario final y determinar mejores estados de operación de la planta fotovoltaica.

- Bajo coste: debido a que no se requiere de equipo adicional para entablar la comunicación con los sensores instalados en la granja solar se genera una reducción de la inversión para el desarrollo del dispositivo.
- Ubicación física: un punto importante considerado al momento de realizar la visita técnica a la planta fotovoltaica fue la ubicación de la estación de clima que contiene los sensores. Esto ayudo a determinar que la mejor forma de trabajar con el dispositivo sea de forma inalámbrica ya que la ubicación de estos sensores se encuentra a la intemperie junto a los paneles.
- Autonomía: si bien es cierto que el dispositivo está destinado a funcionar con un adaptador de corriente, es posible implementar un sistema con batería de respaldo para que el dispositivo siga recopilando información en caso de existir apagones eléctricos o fallas con la fuente principal del dispositivo.

Una vez definido cuáles eras las condiciones físicas a las que el dispositivo estaría sometido y revisar las características de funcionamiento que este debe contener, se empezó con el diseño electrónico de este. Como elemento principal se consideró utilizar un microcontrolador ESP32 dado a que este dispositivo se ajustaba a los aspectos mencionados anteriormente. Pues esta placa de desarrollo ofrece la capacidad de programarla en base a las necesidades que se requieren, la capacidad de conectarse a internet de forma inalámbrica, su costo es relativamente bajo a comparación de otras placas de desarrollo como Raspberry y a través de código se puede añadir algunas funciones destinadas otorgar al usuario final una mayor cantidad de información relevante para la toma de decisiones.

Nuevamente, en base a los requerimientos físicos de la placa se empezó con el diseño del hardware en donde se integran varias de las funciones principales del equipo para posteriormente iniciar con su programación.

2.2.1 Diseño electrónico del dispositivo IoT

Para garantizar que el dispositivo funcione correctamente y se ajuste a las necesidades del usuario se deben considerar varios aspectos para su diseño electrónico.

- El dispositivo debe establecer comunicación con los sensores mediante el protocolo RS-485.
- El dispositivo debe contar con una batería de respaldo para mantenerlo funcionando en caso de existir fallas con la red eléctrica o en la fuente de alimentación principal.
- El dispositivo debe tener detectar la interrupción de su fuente de alimentación primaria y notificar al usuario cuando esto ha sucedido.
- El dispositivo debe ser capaz de monitorear el estado del nivel de carga de la batería de respaldo.
- El dispositivo debe incorporar indicadores visuales que otorguen al usuario final información relevante acerca de su funcionamiento.

Todos estos requerimientos han sido considerados en el diseño electrónico de este dispositivo, mismos que se han separado por módulos en base al trabajo que realizará cada uno tal como se aprecia en la figura adjunta.

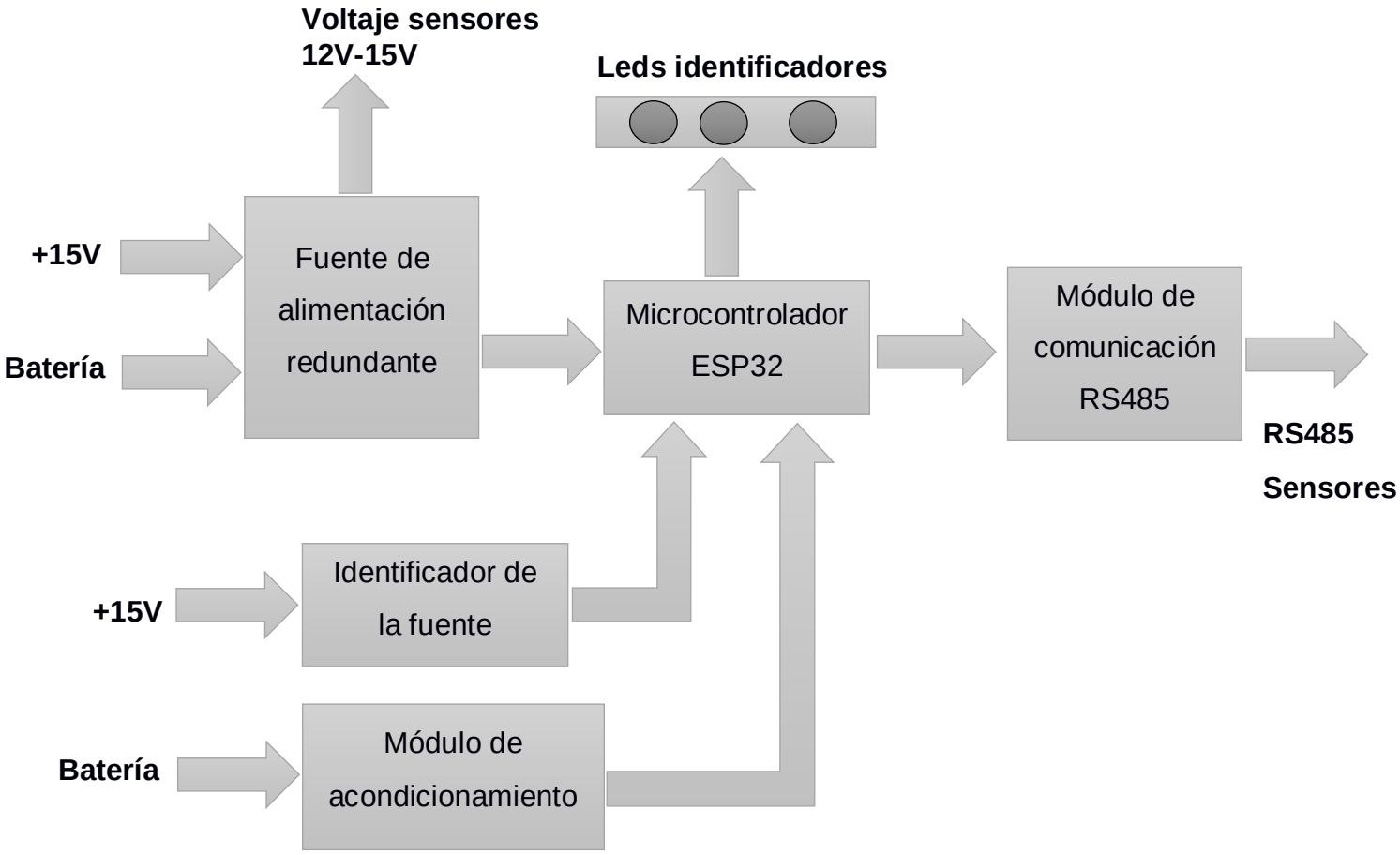


Ilustración 15. Diagrama de bloques que representa las etapas que conforman el dispositivo IoT.

2.2.1.1 Batería de respaldo VRLA del dispositivo

Comenzando con el diseño, se seleccionó una batería recargable VRLA con un rango de operación de 12V a 14.4V debido a ventajas como: una baja tasa de mantenimiento, su alto nivel de autonomía, su facilidad de recarga y su capacidad de mantener sus características aun permaneciendo a la intemperie. Esta batería es muy utilizada en dispositivos que funcionan como fuentes de respaldo como los UPS.

Ilustración 16. Fotografía de la batería implementada como respaldo para desconexión de la fuente principal.

2.2.1.2 Circuito de comunicación RS485

La implementación del sistema que permite al dispositivo utilizar el protocolo de comunicación RS-485 o Modbus serie se lo realizó en base el diagrama electrónico mostrado en la *Ilustración 17*.

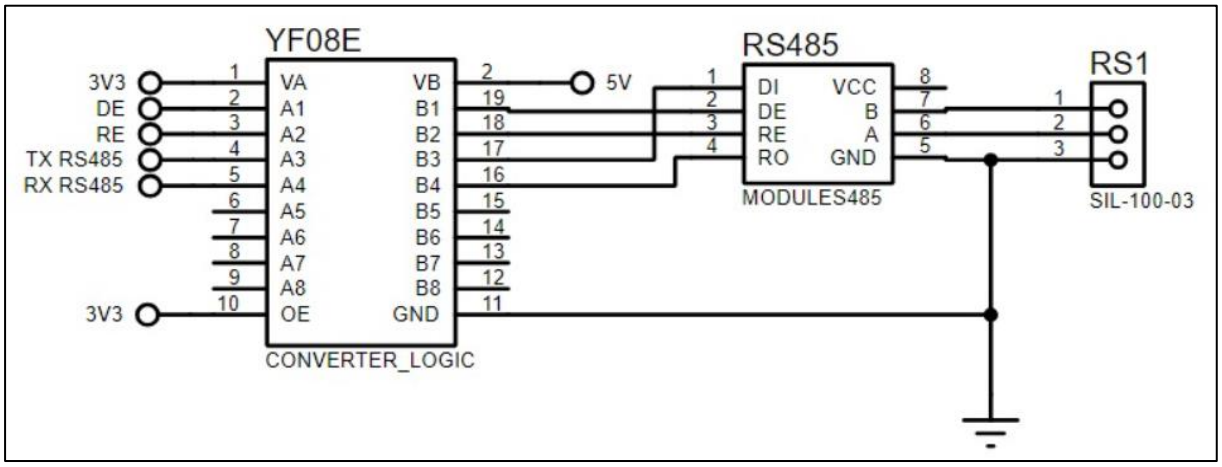


Ilustración 17. Circuito implementado para establecer la comunicación entre la ESP32 y los sensores de la estación de clima.

2.2.1.3 Módulo HW-221

El primer módulo consiste en un convertidor de nivel lógico bidireccional HW-221 que en su estructura implementa el circuito integrado YF08E. Este módulo es implementado debido a que el nivel de voltaje del pórtico serial con el que trabaja la ESP32 es de 3.3V y el nivel de voltaje de la señal con la que trabaja el MAX485 es de 5V.

La implementación de este módulo es proteger el pórtico serial de la ESP32 evitando que los 5V de MAX485 lleguen directamente a este.

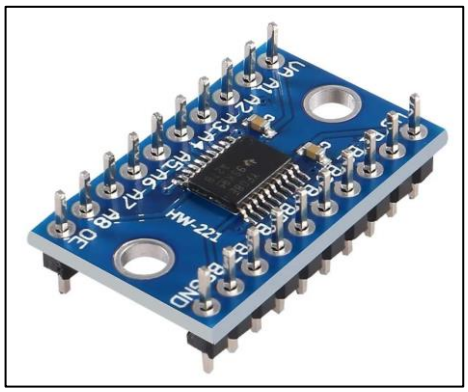


Ilustración 18. Módulo HW-221 implementado.

2.2.1.4 Módulo MAX485

El segundo módulo implementado es el MAX485 que permite convertir las señales Tx y Rx provenientes de la ESP32 en la señal diferenciada A+ y B- de la interfaz RS485. Además, integra dos pines de control RE y DE que se utilizan para habilitar los pines de recepción y transmisión en el MAX485. Es importante mencionar que el módulo MAX485 solo permite el modo de comunicación Half Duplex:

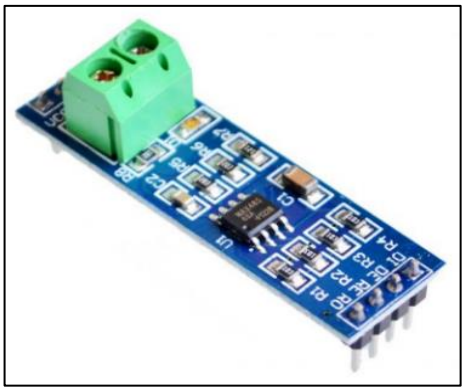


Ilustración 19. Módulo Rs485 implementado.

Este módulo utiliza el circuito integrado MAX485 implementado de tal manera que este no necesite configuraciones adicionales para su funcionamiento a excepción del convertidor lógico mencionado anteriormente.

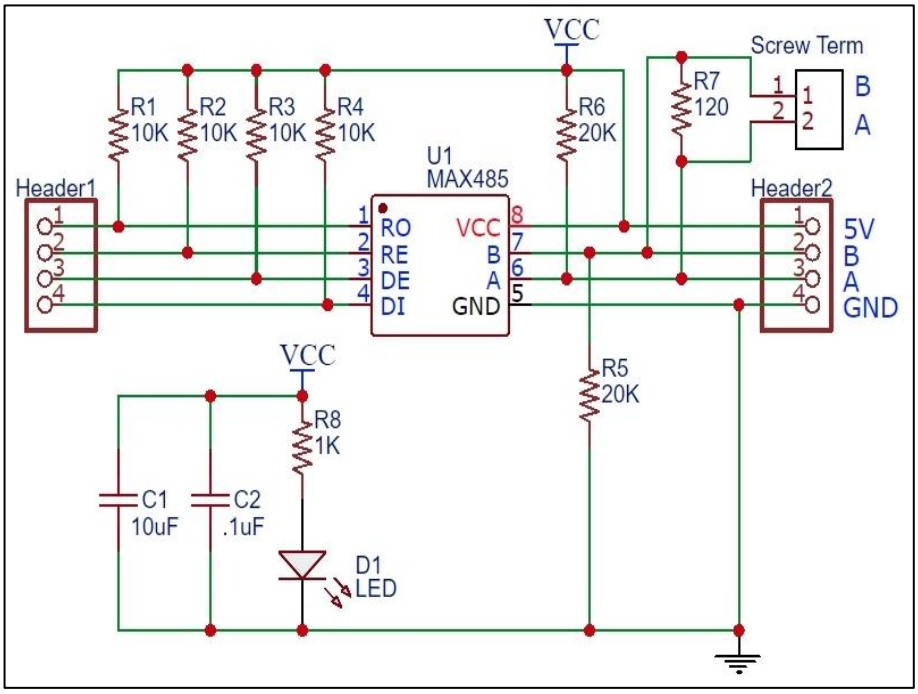


Ilustración 20. Detalle del circuito del módulo RS485.

El circuito integrado MAX485 está compuesto por dos buffers con topologías específicas que permiten convertir la señal de la interfaz TTL/UART a la señal diferenciada A y B y viceversa.

Además, se debe considerar los pines RE y DE que permiten el control de fuljo de los datos entre dispositivos; es decir, controlar cuando se envía y cuando se recibe datos.

Estos pines son controlados por los pines GPIO de la ESP32.

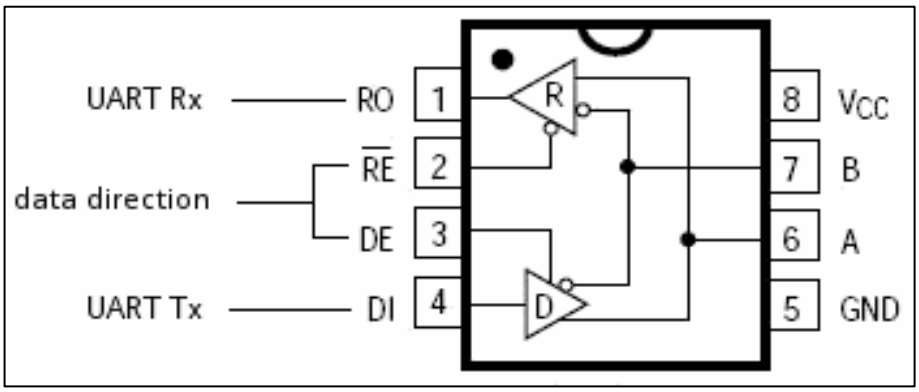


Ilustración 21. Lógica interna del integrado MAX485.

2.2.1.5 Circuito de medición de nivel de carga de la batería

Para medir el nivel de tensión de la batería, se debe considerar el rango de operación de esta que va desde los 12V a los 14.4V. Además, se debe considerar que el microcontrolador ESP32 en sus pines analógicos trabaja con voltajes de hasta 3.3V por este motivo para poder acondicionar los niveles de tensión a los rangos de operación adecuados se implementó un circuito de escalamiento de señal en donde se implementa un típico divisor de tensión utilizando el circuito integrado LM358 como seguidor de voltaje.

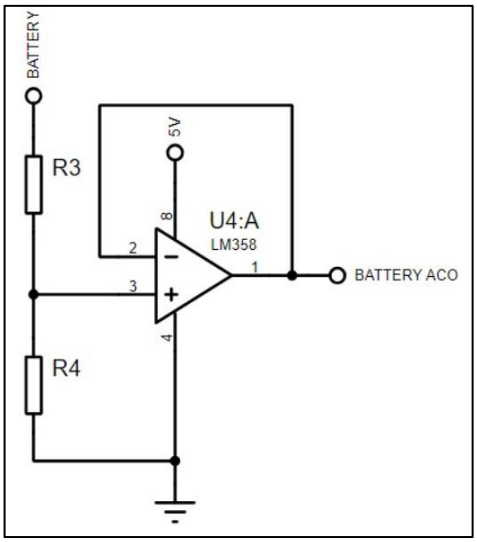


Ilustración 22. Circuito implementado para adaptar y medir el voltaje de la batería.

Se debe considerar que, la señal BATTERY tiene un rango de tensión que va desde los 0V a los 15V y que la señal BATTERY ACO debe tener un rango de tensión que va desde los 0V a los 3.3V. En base a estos valores procedemos a determinar la ganancia del circuito de la siguiente manera.

$$G = \frac{V_{in\_max} - V_{in\_min}}{V_{out\_max} - V_{out\_min}}$$

$$G = \frac{3.3V - 0V}{15V - 0V}$$

$$G = 0.22$$

Una vez determinada la ganancia del circuito, se procede a determinar los valores de las resistencias R3 y R4, considerando que estas deben tener un valor resistivo alto para disminuir el consumo de corriente en el circuito.

$$\frac{R_4}{R_4 + R_3} = G$$

$$\frac{22k\Omega}{22k\Omega + R_3} = 0.22$$

$$22k\Omega = 4.84k\Omega + 0.22R_3$$

$$R_3 = \frac{22k\Omega - 4.84k\Omega}{0.22}$$

$$R_3 = 78k\Omega$$

Para este caso en específico los valores de R4 y R3 se adaptaron de tal manera que los resistores utilizados tengan valores comerciales y evitar realizar configuraciones en serie o paralelo para determinar los valores exactos requeridos. Se estableció como referencia un resistor R_4 de $22k\Omega$ y se determinó que R_3 requería un valor de $78k\Omega$. Sin embargo, se lo aproximó a un valor comercial de 82 $k\Omega$.

$$R_3 = 82K\Omega$$

$$R_4 = 22K\Omega$$

Por lo tanto, la ganancia obtenida que utiliza estos valores de resistencia es de 0.21, misma que se aproxima mucho al valor necesitado.

2.2.1.6 Circuito detector de fuente de alimentación principal

Para identificar si la fuente de alimentación principal se encuentra conectada al dispositivo, se utilizó un optoacoplador que al activarse envía una señal en alto al microcontrolador implementando una resistencia de pull down en el transistor del optoacoplador para obtener información acerca del estado de conexión o desconexión de la fuente de alimentación principal. Para lograr esto, se implementó el circuito mostrado a continuación y que utiliza 15V provenientes directamente de la fuente de alimentación principal a diferencia de la señal de 3.3V que proviene de la ESP32 que, en caso de existir desconexión de la fuente principal, esta se mantendrá energizada gracias a la batería de respaldo.

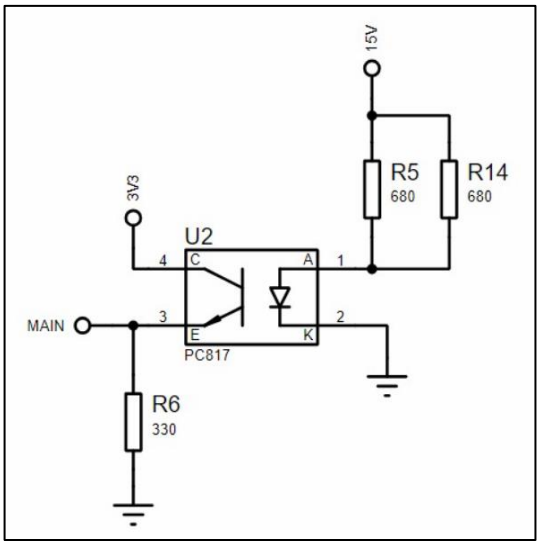


Ilustración 23. Circuito implementado para detectar el funcionamiento de la fuente de alimentación principal.

Para determinar los valores de las resistencias R5 y R14, se tuvo en cuenta la corriente mínima de operación del optoacoplador, considerando un valor de 50 mA. Por lo tanto, con los datos de la fuente de 15V y la corriente, se determina que la resistencia a implementar debe tener un valor de 300Ω a 1W. Sin embargo, debido a la ausencia de esta, se implementaron dos resistencias de 680Ω a 1W cada una y conectadas en paralelo.

2.2.1.7 CPU del dispositivo

La configuración de los pines de la ESP32 utilizados se detallan en la siguiente tabla:

Pines GPIO	Entrada/Salida	Acción
GPIO G35	Entrada	Identifica el estado de la fuente principal
GPIO G33	Salida	Indica el estado de no operación (Standby)
GPIO G25	Salida	Indica el estado de operación
GPIO G26	Salida	Indica que no está conectado a Wifi
GPIO G27	Salida	Indica que se encuentra conectado a Wifi
GPIO G14	Salida	Indica que no tiene permisos en la base de datos
GPIO G12	Salida	Indica que tiene permisos en la base de datos
GPIO G18	Salida	Controla el modo de trasmisión de datos RS485, DE
GPIO G5	Salida	Controla el modo de recepción de datos RS485, RE

Tabla 1. Detalle de uso de pines de la ESP32.

Además, se hizo uso del módulo ADC 1 en el pin G32, para leer el estado del nivel de carga de batería. Por otro lado, se hizo uso de dos módulos UART del microcontrolador correspondientes a UART0 y UART2. El primero permite la comunicación de la ESP32 por medio de un conector USB hacia el computador permitiendo así su programación y configuración de esta a través de comandos establecidos dentro del programa. El segundo es utilizado para enviar o recibir datos desde el módulo MAX485, y está enfocado en el uso de protocolo Modbus RTU.

La siguiente imagen muestra la ubicación de los pines con los indicadores visuales implementados en el dispositivo.

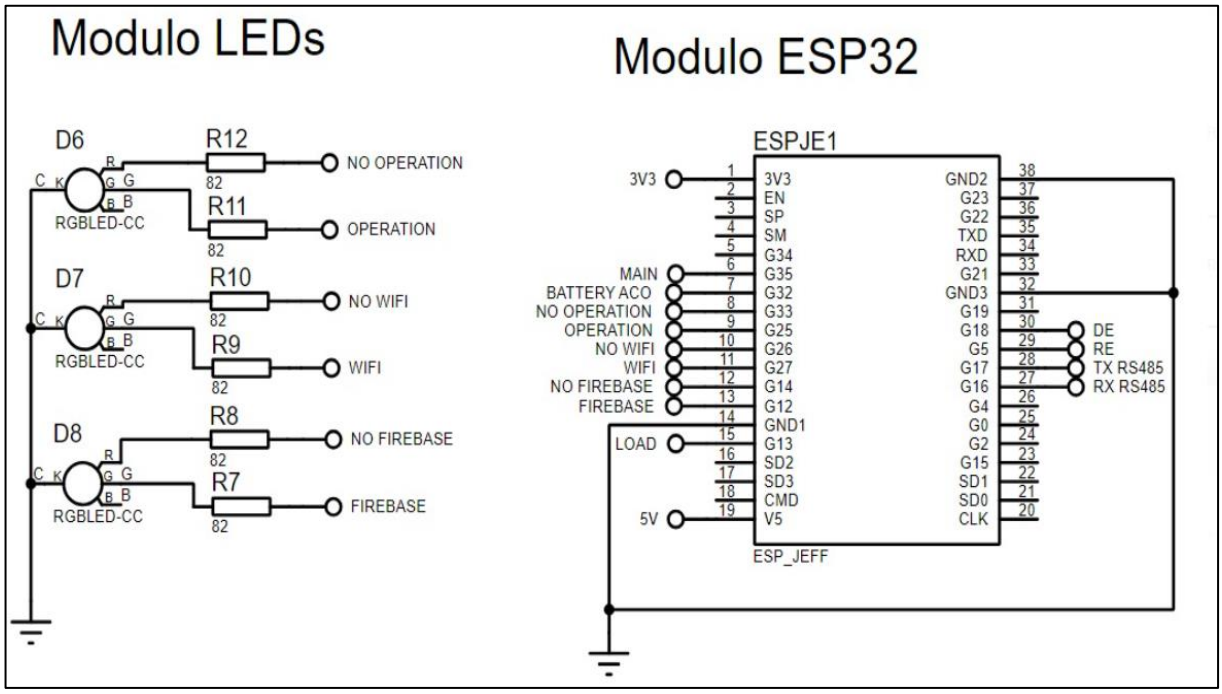


Ilustración 24. Descripción de pines utilizados en la ESP32.

2.2.1.8 Circuito de fuente redundante

En este apartado se detalla la implementación de una fuente redundante utilizada para conmutar de la fuente de alimentación principal hacia la batería de respaldo del dispositivo. Esto se lo

realiza para mantener funcionando al dispositivo en caso de existir desconexiones de la red eléctrica en la granja solar o en caso de que la fuente principal del dispositivo deje de funcionar. Se tuvo en cuenta que el rango de operación de los sensores de la estación de clima utiliza un voltaje de alimentación que va de 12V a 24V y la batería VRLA otorga un voltaje de operación de 12V a 14.4V; por lo tanto, se concluyó que los voltajes de operación del circuito de conmutación trabajarían en un rango de voltajes de 12V a 15V. El circuito de conmutación se presenta a continuación.

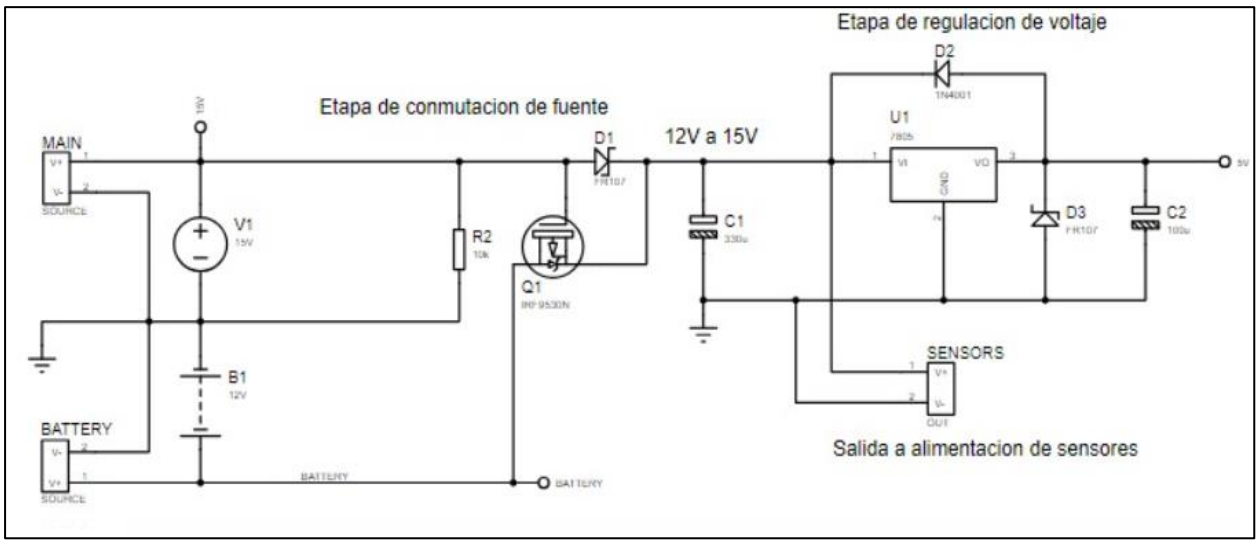


Ilustración 25. Circuito implementado para conmutar a la batería de respaldo cuando la fuente principal entra en fallo o no hay energía eléctrica.

Para la etapa de conmutación se utilizó un MOSFET canal P que conmuta desde la fuente de alimentación principal a la batería a través de un control automático en su compuerta. Su funcionamiento es el siguiente:

En el primer estado en donde el dispositivo mantiene su fuente de alimentación principal conectada, se polariza directamente al diodo D1 permitiendo el paso de la corriente a la etapa de regulación de voltaje y alimentación de sensores. En este caso, D1 tiene un voltaje de 0.6V y dado a su estado de conducción impide que el voltaje VGS del MOSFET llegue a su umbral de

operación. Como consecuencia de esto se impide la circulación de la corriente proveniente de la batería. Esta primera etapa la podemos apreciar en la siguiente imagen:

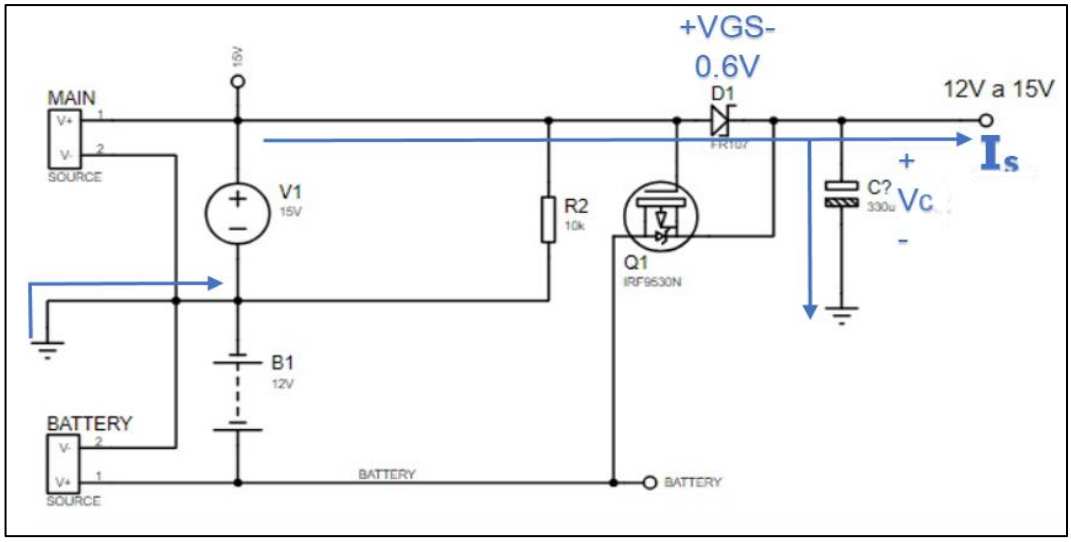


Ilustración 26. Detalle del flujo de corriente cuando la fuente principal está en operación.

En el segundo estado en donde la fuente de alimentación es desconectada la resistencia R2 de 10K al carecer de una circulación de corriente a través de ella, se define como una tierra virtual a 0V en el Gate del MOSFET. Por otro lado, el capacitor C1 tiene un voltaje almacenado lo que hace que el transistor tenga un voltaje VGS de -12V a -15V dependiendo del rango de operación en el que se encuentre, esto permite la conducción del transistor alimentando finalmente al circuito con la batería. Esto lo apreciamos en la siguiente imagen:

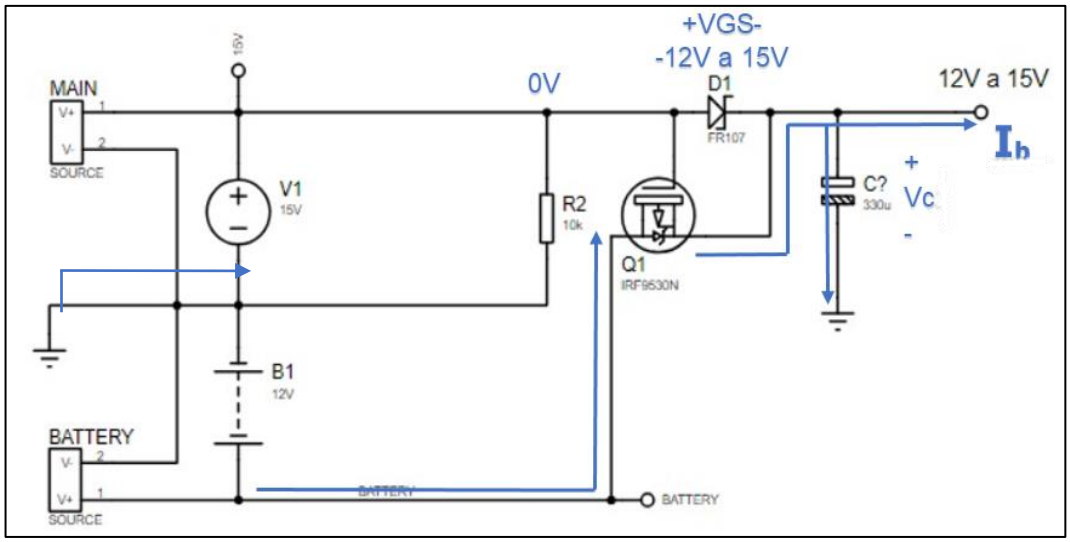


Ilustración 27. Detalle del flujo de corriente cuando la batería de respaldo entra en operación.

2.2.1.9 Circuito de regulación de voltaje

Para la etapa reguladora se hace uso del circuito integrado 7805 para bajar el voltaje de 15V a 5V que se utiliza para alimentar los componentes como: ESP32, MAX485, HW-221, circuito integrado LM358 y los indicadores visuales:

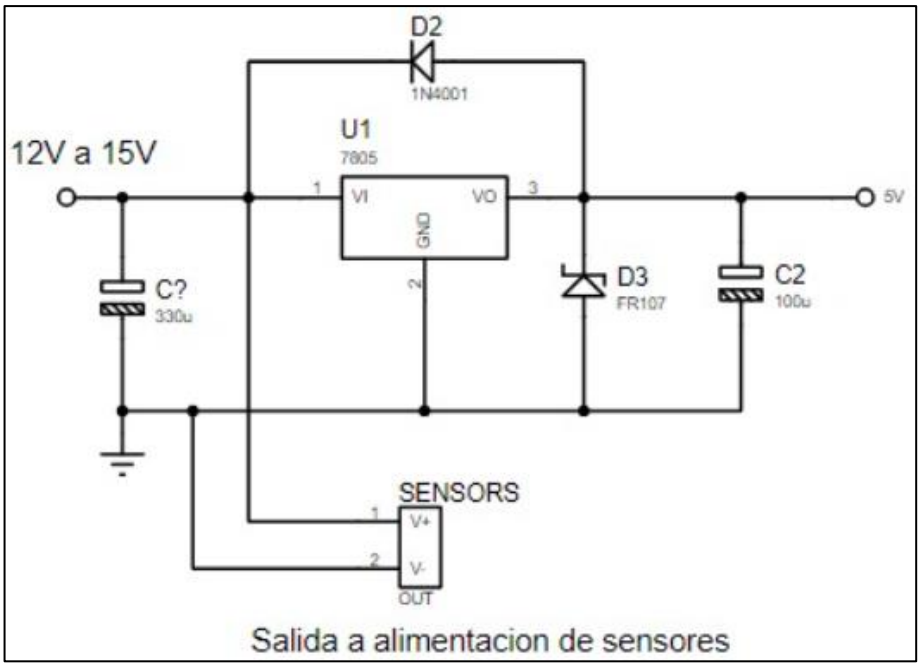


Ilustración 28. Circuito regulador de voltaje de 12V a 5V para alimentar la ESP32.

2.2.2 Estimación de consumos y respaldo

2.2.2.1 Estimación de consumo de potencia

A continuación, se estima el consumo de cada componente que es alimentado con 5V:

El consumo de los leds indicadores que corresponde a 3 led RGB, teniendo en cuenta que solo trabajan dos de sus componentes (led verde, y led rojo) pero no de forma simultánea, se calculó su consumo de la siguiente manera:

$$I_w = 3xI_{led}$$

$$I_w = 3x20\ mA$$

$$I_w = 60\ mA$$

El módulo de la ESP32 Tiene varios estados de trabajo, en este caso se tuvo en cuenta el estado de máximo rendimiento y uso del módulo Wifi (ver anexo 1) por los requerimientos de su programa, y se estableció un consumo de 180 mA.

Para el consumo del módulo MAX485 se tuvo en cuenta la información ofrecida por el datasheet (ver anexo 2) donde la corriente de estado de reposo es de 300 µA. Es necesario mencionar que el consumo del módulo en actividad depende de la frecuencia de solicitudes de datos, este caso 1 solicitud por minuto. En base a esto, se realizó la siguiente estimación:

Se calcula la corriente de la transmisión del mensaje identificando la carga más relevante correspondiente a la resistencia de 120 Ω en las terminales A+ y B-:

$$I_{op} = \frac{Vcc}{R_{485}}$$

$$I_{op} = \frac{5\ V}{120\ \Omega}$$

$$I_{op} = 42\ mA$$

El tiempo del mensaje se estima utilizando la frecuencia utilizada que en este caso es de 9600 baudios con el número de bytes que contiene el mensaje, como se hizo uso de 8 bytes para la solicitud de datos y teniendo en cuenta que se definió un rango de 10 solicitudes por minuto dentro del programa en caso de no existir respuesta inmediata del dispositivo, se calcula:

$$T_{msg} = \frac{N° \; de \; bytes \times N° \; de \; solicitudes}{baudrate}$$

$$T_{msg} = \frac{8 \times 10}{9600}$$

$$T_{msg} = 8.3 \; ms$$

Donde se obtiene la corriente promedio en un minuto de trabajo:

$$I_m = \frac{I_s \times T_s + I_{op} \times T_{msg}}{T_T}$$

Donde:

$I_s = corriente \; de \; reposo.$

$T_s = tiempo \; de \; reposo.$

$I_{op} = corriente \; de \; operacion$

$I_{op} = tiempo \; de \; operacion$

$T_T = Tiempo \; total$

$I_m = corriente \; promedio.$

$$I_T = \frac{300\mu A \; \times 59991.7ms + 42000\mu A \times 8.3ms}{60000ms}$$

$$I_T = 306 \; \mu A$$

Para el consumo del LM358, se utilizó la información dada en su datasheet, mima que estima un consumo promedio de 30 mA. (Ver anexo 3)

Para el consumo del módulo HW-221 se revisó los detalles técnicos del circuito integrado YF08E y su datasheet presento que puede consumir hasta 100mA. (Ver anexo 4).

Con toda la información definida anteriormente, se adjuntan los resultados en la siguiente tabla:

Componentes alimentados por la batería	**Corrientes de consumo promedio**
Leds indicadores (3 led RGB, solo trabaja un led a la vez)	60 mA
Circuito integrado LM358	30 mA
ESP32 (modulo alimentado a 5V)	180 mA
Módulo MAX485	0.3 mA
módulo HW-221	100 mA
Total	370.3 mA

Tabla 2. Consumo de corriente por cada elemento utilizado.

Con la potencia total se estimará la potencia de disipación máxima (W_d) del circuito integrado 7508.

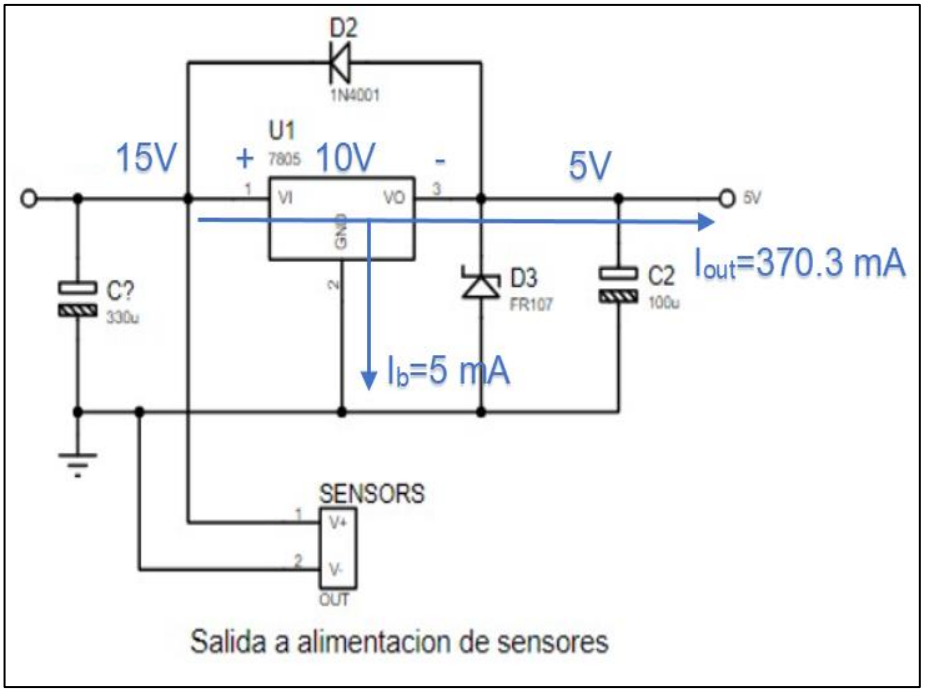


Ilustración 29. Flujo de corriente desde la etapa reguladora hacia los demás elementos del circuito.

$$W_d = (V_{in} - V_{out}) \times (I_b + I_{out})$$

$W_d = (15\ V - 5\ V) \times (5\ mA + 370.3\ mA)$

$W_d = 3.753\ W$

Dado que la potencia disipada es superior a 1.9 W y según su datasheet, se considera el uso de un disipador cuyo dimensionamiento es el siguiente:

Primero determinamos la resistencia térmica del disipador hacia el ambiente (R_{sa}):

$$W_d = \frac{T_j - T_a}{R_{jc} + R_{cs} + R_{sa}}$$

Donde:

$W_d = potencia\ de\ disipacion$

$T_j = temperatura\ de\ juntura\ (temperatura\ de\ operacion\ segun\ el\ datasheet)$

$T_a = temperatura\ ambiente$

$R_{jc} = resistencia\ termica\ de\ juntura - carcasa$

$R_{cs} = resistencia\ termica\ de\ carcasa\ disipdor$

$R_{sa} = resistencia\ termica\ de\ dispador\ al\ ambiente$

$$R_{sa} = \frac{T_j - T_a}{W_d} - R_{jc} - R_{cs}$$

$$R_{sa} = \frac{150 - 25}{3.753} - 4 - 2$$

$$R_{sa} = 27.3\ W/°C$$

Por lo tanto, se hizo uso del disipador DIS-202 de aluminio que tiene una resistencia térmica de 50W/°C.

2.2.2.2 Estimación de autonomía de batería de respaldo

Para el cálculo de la autonomía del dispositivo se debe considerar el usó una batería VRLA de 12V. El consumo del circuito se calculó de la siguiente manera:

Del apartado anterior se adjunta una corriente consumida por el 7805 correspondiente a 375.3 mA.

El consumo de los sensores de la estación meteorológica se estima en de 25 mA según su fabricante (ver anexo 5).

El consumo del circuito de acondicionamiento de señal de voltaje de batería se estima en base al consumo de la resistencia integrada al divisor de tensión.

$$I_R = \frac{V_{battery}}{R_3 + R_4}$$

$$I_R = \frac{15\ V}{82K\Omega + 22K\Omega}$$

$$I_R = 0.1\ mA$$

La tabla adjunta recopila el consumo de cada componente lo que a su vez se utiliza para determinar la corriente que debe suministrar la batería de respaldo.

Componentes alimentados por la batería	**Corrientes de consumo promedio**
Consumo del CI 7805	375.3 mA
Sensores estación meteorológica	25 mA
Circuito de acondicionamiento	0.1 mA
Total	395.4 mA

Tabla 3. Consumo de corriente total del dispositivo.

Para brindar una mayor seguridad en cuanto al funcionamiento del dispositivo en caso de desconexión de la fuente principal, se estableció que este funcione con su batería durante un intervalo de hasta 24 horas. Por ello, se realizó el siguiente cálculo para obtener la carga necesaria de la batería:

$$H = \frac{C_b}{I_T}$$

Donde:

$H = Autonomia\ de\ la\ bateria\ (H)$

$C_b = capacidad\ de\ bateria\ (AH)$

$I_T = corriente\ suministrada\ a\ la\ carga\ de\ la\ bateria\ (A)$

$$24 = \frac{C_b}{0.3954\ A}$$

$$C_b = 9.5\ Ah$$

Con el cálculo realizado, se determinó que la batería debe tener una capacidad de 9.5Ah. Sin embargo; al no contar con una batería con esa capacidad en específico, se implementó al sistema una batería con una capacidad de hasta 7.5Ah, la misma que teóricamente nos brinda una autonomía de hasta 19h de funcionamiento.

2.2.3 Diseño PCB del dispositivo electrónico y costes

Con cada uno de los circuitos detallados finalmente se procede a elaborar el circuito electrónico del dispositivo y el diseño de su PCB para proceder a implementarlo.

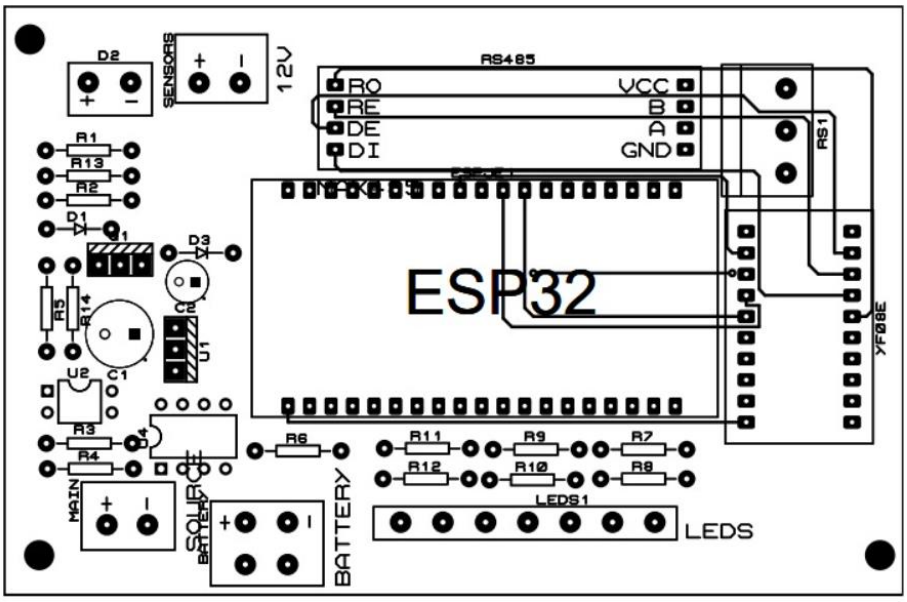


Ilustración 30. Layout de la PCB del dispositivo.

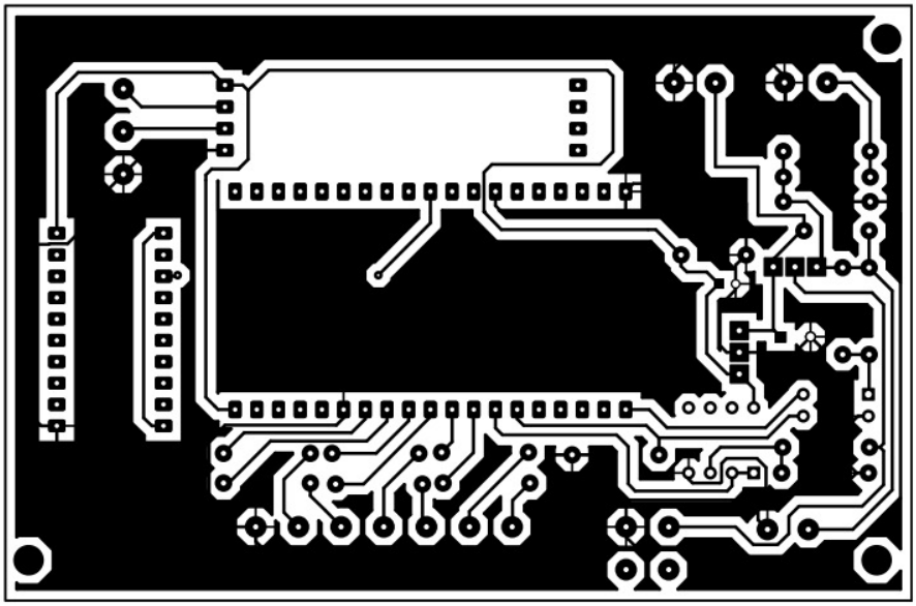


Ilustración 31. Figura de la PCB del dispositivo.

2.2.4 Configuración y programación de la ESP32

Para la programación del microcontrolador, se han considerado las funcionalidades necesarias para cumplir con los requerimientos del sistema de monitoreo. Esto incluye la configuración de la conexión a redes Wifi, la utilización de librerías para la carga de datos a una base de datos (Firebase), la configuración del reloj interno, el uso de la memoria de almacenamiento no volátil para cargar parámetros y datos importantes, la configuración de pines como entradas y salidas digitales para integrar algunas operaciones del dispositivo, así como la configuración de una entrada analógica para monitorear el estado de carga de la batería.

En el siguiente diagrama de flujo representamos la inicialización y ciclo principal de programa implementado en el dispositivo IoT:

- Para la configuración Wifi se integró un perfil de cliente de red donde el dispositivo escanea y se conecta a una red Wifi configurada por el usuario a través de proporcionar la información de las credenciales correspondientes a la red a la que se desea conectar. Además, se realiza una configuración adicional de sincronización de hora, misma que permite que el reloj interno de la ESP32 se sincronice con el reloj de un servidor NTP.
- El módulo ADC utilizado fue configurado con una resolución de 12 bits, por ende, los valores que llegan al pin G32 del dispositivo que van de 0 a 3.3V se escalan en un rango de datos que van de 0 a 4095. Por otro lado, es necesario determinar la propagación de error que se produce a causa del porcentaje de error que los resistores utilizados pueden tener.

  Los resistores adquiridos posen una tolerancia del 5%; por lo tanto, la ganancia del circuito implementado puede influir en la medición del nivel de la batería haciendo que el valor obtenido pueda ocasionar fallas de lectura o daño a la ESP32:

$$R_3 = [82000 \pm 4100\,]\,\Omega$$

$$R_4 = [22000 \pm 1100\,]\,\Omega$$

  Se define la ecuación de la ganancia de tensión del acondicionamiento:

$$V_{ADC} = GV_{battery}$$

$$V_{ADC} = \frac{R_4}{R_4 + R_3} V_{battery}$$

$$\delta V_{ADC} = \left[\frac{R_3}{(R_3 + R_4)^2}\delta R_4 + \frac{R_4}{(R_3 + R_4)^2}\delta R_3\right]$$

$$\delta V_{ADC} = \left[\frac{82000}{(82000 + 22000)^2}(1100) + \frac{22000}{(82000 + 22000)^2}(4100)\right]$$

$$\delta V_{ADC} = 0.0167\ V$$

Podemos identificar que se propaga un error de $\pm 0.0167\ V$ entre el voltaje de la batería y el voltaje del módulo ADC que se mide en el microcontrolador. Cabe recalcar que se despreció el error de medición del módulo ADC; por lo tanto, en la programación para definir los niveles de tensión de se debe tener en cuenta este error. Finalmente, para definir los valores máximos y mínimos realizamos lo siguiente:

Niveles de tensión de la batería	Niveles de tensión en el módulo ADC	Magnitudes definidas en bits
12 V	[2.5385±0.0167] V	3150±21
13 V	[2.7500±0.0167] V	3413±21
14 V	[2.9615±0.0167] V	3675±21
15 V	[3.1731±0.0167] V	3938±21

Tabla 4. Estimación de propagación de error en base a porcentaje de error de resistores.

Lo valores en bits nos permiten definir los niveles de tensión y tener en cuenta la propagación de error; por esto, se establecieron como limites más convenientes para su funcionamiento los presentados a continuación

3171 0%

3917 100%

- Para la configuración de las particiones de memoria se agregó una sección adicional para datos, donde se asignó el tipo de partición SPIFFS recomendado por el fabricante. Además,

se identificó el módulo NVS que será utilizado para el almacenamiento de parámetros de configuración del equipo.

```
# Name,   Type, SubType,  Offset,   Size,  Flags
ota_0,     app, ota_0,    0x10000,  0x1A0000,
ota_1,     app, ota_1,            ,  0x1A0000,
otadata,  data, ota,     0x350000,  0x2000,
nvs,      data, nvs,             ,  0x6000,
data,     data, spiffs,          ,  0xA8000,
```

Ilustración 32. Estructura de la partición de la memoria de la ESP32.

- La partición de memoria SPIFFS tiene como propósito almacenar un archivo definido como data.csv donde se almacena los datos adquiridos durante el día y que permite tener acceso a los datos durante la sincronización del dispositivo con la nube permitiendo al usuario evitar pérdidas de información durante fallas en la conexión a internet o en el microcontrolador.
- La partición de memoria nvs permite almacenar estructura de datos; sin embargo, esta fue adaptada para registrar variables que representan parámetros de configuración del dispositivo y que están definidas en la siguiente tabla:

Nombre de la variable	Tipo	Descripción
NAME_DEVICE	Cadenas de caracteres	Registra el nombre del dispositivo
SSID_WIFI	Cadenas de caracteres	Indica el nombre de la red Wifi que a conectar
PASS_WIFI	Cadenas de caracteres	Registra la contraseña de la red Wifi
USER_FIRE	Cadenas de caracteres	Registra el nombre de usuario asignado en la plataforma
PASS_FIRE	Cadenas de caracteres	Registra la contraseña de su usuario en la plataforma
DATE_ACTIVITY	Cadenas de caracteres	Registra la fecha de su actividad actual
HOUR_START	Numero entero	Registra la hora de inicio de su actividad
HOUR_STOP	Numero entero	Registra la hora de fin de su actividad
SAMPLING_TIME	Numero entero	Registra el tiempo de muestreo

Tabla 5. Tipo de valores almacenado por variable de datos.

- Los pines GPIO son configurados como entradas y salidas dependiendo de los requerimientos de los dispositivos tales como: leds identificadores, identificador de la fuente de voltaje y control digital de recepción y emisión de datos del módulo RS485.

Pines GPIO	Entrada/Salida	Descripción
GPIO 35	Entrada	Identifica el estado de la fuente principal
GPIO 33	Salida	Indica el estado de no operación (standby)
GPIO 25	Salida	Indica el estado de operación
GPIO 26	Salida	Indica que no está conectado a Wifi
GPIO 27	Salida	Indica que se encuentra conectado a Wifi
GPIO 14	Salida	Indica que no tiene permisos en la base de datos
GPIO 12	Salida	Indica que tiene permisos en la base de datos
GPIO 18	Salida	Controla el modo de trasmisión de datos RS485, DE
GPIO 5	Salida	Controla el modo de recepción de datos RS485, RE

Tabla 6. Configuración de pines de entrada y de salida de la ESP32.

- Los módulos UART fueron configurados de la siguiente manera:

Módulo UART	Actividad
UART0	Usa un protocolo propio que procesa comandos que permite la configuración del dispositivo.
UART2	Utilizado para enviar solicitudes bajo el protocolo Modbus RTU al conjunto de sensores para obtener sus datos.

Tabla 7. Detalle de utilización de los módulos UART de la ESP32.

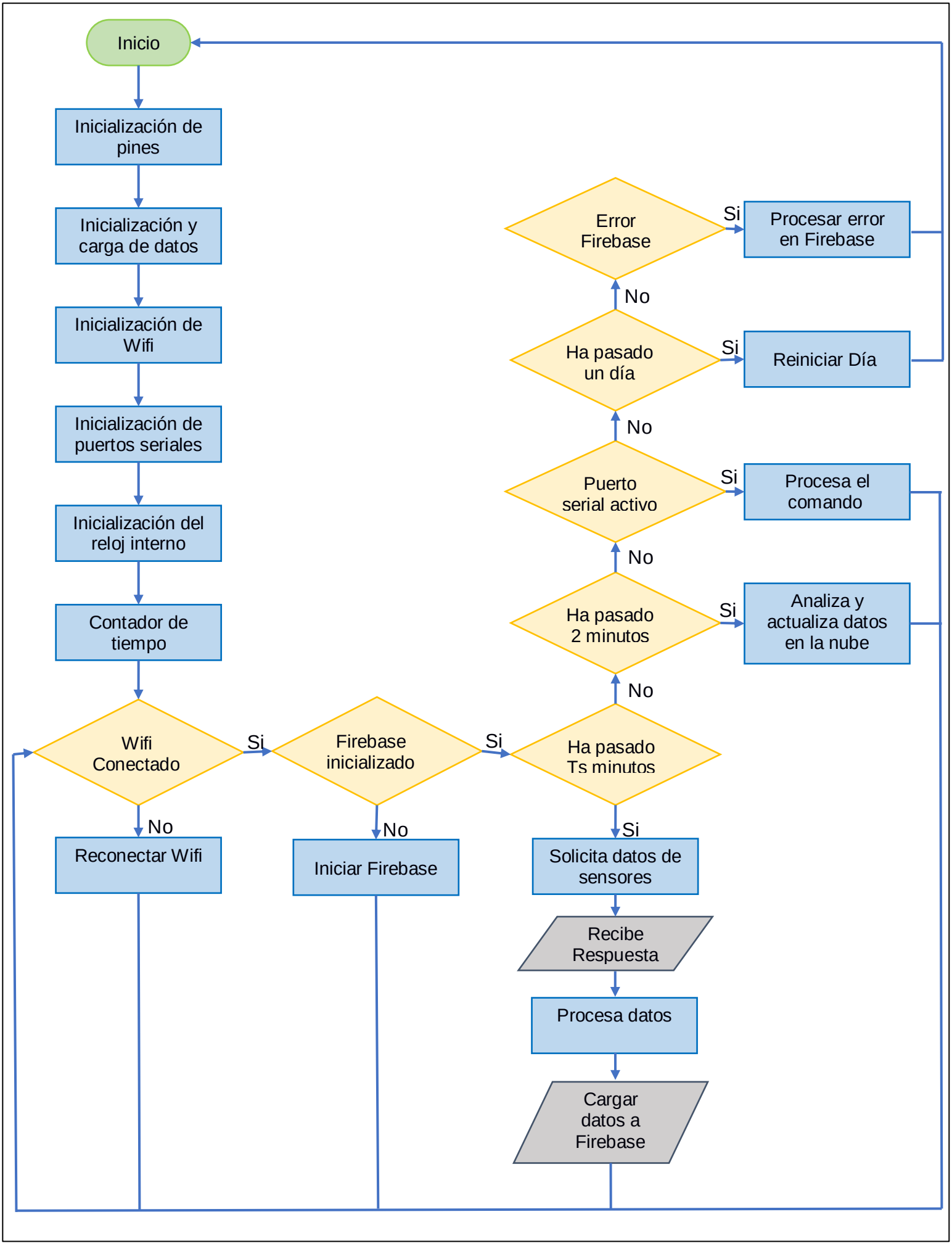


Ilustración 33. Descripción de la rutina del procesamiento de información.

Para este programa, es crucial tener en cuenta que se está muestreando una cantidad considerable de datos. Por lo tanto, en la función de “análisis y actualización de datos en la nube”, se ha considerado cuidadosamente los recursos del microcontrolador. En este caso, se ha definido un algoritmo que permite cargar y procesar datos cada 0.5 segundos. Además, se implementa la limpieza de los datos cargados en la memoria RAM mediante la función "Procesar Error Firebase", reiniciando Firebase. Esta estrategia asegura una utilización eficiente de los recursos del microcontrolador, proporcionando así una mayor autonomía.

2.3 Diseño de base de datos

En esta etapa, se analizó la base de datos más apropiada, teniendo en cuenta parámetros como escalabilidad, autentificación, documentación (información disponible para configurarla), accesibilidad y costes, cabe recalcar que solo se analizó base de datos no relacionales (noSQL) ya que son más apropiadas para almacenar datos de un sistema de monitoreo.

- MongoDB: Es una base de datos dedicada al almacenamiento de documentos, donde es muy utilizada en aplicaciones móviles y web, además utiliza documentos en formato BSON para almacenar datos.

Ventajas:

- Tiene servicio en línea por medio de la plataforma MongoDB Atlas de Amazon
- Tiene servicios de Autenticación y roles.
- Tiene diferentes servicios como: Data Base Real Time, Storage y migración de datos.
- La base de datos es escalable para aplicaciones muy grandes.

Desventajas:

- Lleva costes de suscripción para el uso de la plataforma en línea.
- Costos de almacenamiento de datos en la nube.
- Falta de información y documentación precisa para algunas aplicaciones.
- Tiene cierta complejidad en su configuración.

- Firebase: Es una plataforma dedicada al desarrollo de aplicaciones móviles y web medianas y pequeñas, en la cual tiene varias herramientas de desarrollo, tales como: Data Base Real Time, Storage, Autentificación, Hosting, etc:

Ventajas:

- Tiene los servicios integrados desde la base de datos hasta la plataforma en línea.
- Tiene servicios de Autenticación y roles.
- Tiene diferentes servicios como: Data Base Real Time y Storage.
- Tiene gratuidad en los servicios más importante.
- Sencillo de configurar e integrar algunas aplicaciones.

Desventajas:

- Su escalabilidad se limita hasta aplicaciones de mediana escala.
- Dificulta en la migración de datos.

- Prometheus: Es un software de código abierto especializado en la monitorización y almacenamiento de datos en serie temporales

Ventajas:

- Es altamente eficiente en la recolección de datos.
- Es optimo es su sistema de consultas.
- Almacena de forma eficiente datos históricos.
- La base de datos es escalable para aplicaciones muy grandes.

Desventajas:

- Falta de información y documentación precisa para algunas aplicaciones.
- Necesita software adicional para integrarlo en línea.
- Necesita software adicional para integrar servicio de autentificación y roles.

Teniendo en cuenta los puntos fuertes y débiles de cada base de datos presentadas, se concluyó que la base de datos Firebase es más apropiada para esta aplicación, dado a los siguientes puntos:

- Los servicios principales como Base de datos en tiempo real, autentificación, Storage y plataforma en línea son gratuitos.
- Su escalabilidad es suficiente para el sistema de monitoreo que se está implementado.
- Tiene información y documentaciones más completa para su configuración e integración.

2.3.1 Configuración de la base de datos

Para la configuración de la plataforma se hizo uso de tres servicios:

- Servicio de autentificación: se definió dos roles, el operador y el dispositivo IoT, donde cada uno tendrá funciones distintas en la base de datos, además permitirá acceder a la base de datos por medio usuario y contraseña para cada uno de los roles.

Ilustración 34. Registro de usuarios para utilización de base de datos online.

- Servicio de base de datos en tiempo real (RealTime Database): permitió almacenar los datos de los sensores, además permitió visualizar la hora y fecha de conexión del dispositivo, entre otros datos.

2.4 Diseño de la etapa de aplicación GUI

Para esta etapa se consideró varios aspectos que ayudan a determinar la mejor opción para el desarrollo de la aplicación de escritorio. Sin embargo, se da a conocer cuáles fueron las opciones estudiadas para efectuar de manera correcta y eficiente el diseño de esta aplicación.

A continuación, se citarán algunas de las funcionalidades que son requeridas por la aplicación para elaborar un sistema a la medida del usuario final:

- La aplicación debe tener la capacidad de conectarse a la base de datos de Firebase.
- La aplicación debe tener la capacidad de procesar la información de manera rápida debido a la cantidad de datos que se van a manejar durante la descarga de información.
- La aplicación debe tener la capacidad de elaborar gráficos estadísticos para visualizar el comportamiento de producción de energía agrupados por intervalos de tiempo.
- La aplicación debe contar con entradas de texto que permitan al usuario buscar y añadir información dentro de la base de datos.
- La aplicación debe tener la capacidad de buscar dispositivos a través de los puertos seriales del computador en donde esta se encuentra instalada para su posterior configuración.
- La aplicación debe contar con una pantalla de autenticación de usuarios para resguardar la información almacenada en la base de datos.
- La aplicación debe contar con un indicador visual que indique cuando el dispositivo se encuentra funcionando con el adaptador de corriente y cuando se encuentra funcionando con la batería de respaldo; además, indicar el nivel de carga de esta.
- La aplicación debe checar que el dispositivo tenga conexión a internet antes de intentar conectarse a la base de datos en línea, y a través de indicadores visuales informar al usuario cuando se ha establecido o perdido la conexión.

2.4.1 Aplicaciones disponibles

- Python: Es un lenguaje de programación de propósito general, este contiene múltiples librerías que ayudan a elaborar programas para procesamiento de datos y un apartado específicamente para el desarrollo de interfaces gráficas y que complementadas forman una herramienta sólida para el desarrollo de una interfaz gráfica lo suficientemente funcional para el propósito que le es asignado.

Ventajas:

- Amplia variedad de bibliotecas.
- Código abierto y gratuito.
- Fácil aprendizaje.
- Desarrollo de interfaces gráficas.
- Sintaxis clara.
- Compatible con múltiples sistemas operativos.

Desventajas:

- Problemas de GIL al utilizar hilos para ejecutar múltiples códigos a la vez.
- No permite el desarrollo de aplicaciones móviles.
- Los ejecutables generados en Python tienden a ocupar demasiada memoria.

- Qt Creator: Es un entorno de programación especializado en el desarrollo de aplicaciones de GUI basado principalmente en el lenguaje de programación C++ (Blanchette & Mark Summerfield, 2008).

Ventajas:

- Desarrollo de interfaces gráficas.
- Soporte para proyectos en C++.
- Permite la instalación de complementos para el desarrollo de aplicaciones.

Desventajas:

- Para principiantes puede ser complicado familiarizarse con su sintaxis.
- Para proyectos pequeños es una herramienta sobredimensionada.
- Interfaz visual limitada.
- Menos popular entre la comunidad de programadores.

- Flutter: Es un SDK especializado en el desarrollo de interfaces grafica multiplataforma que hace uso del lenguaje de programación Dart. Muy útil para crear interfaces gráficas interactivas para sistemas operativos Android, iOS y plataformas en línea.

Ventajas:

- Cuenta con una extensa biblioteca de widgets.
- Permite desarrollar plataformas web y aplicaciones de escritorio.
- Comunidad relativamente activa.
- Desarrollo multiplataforma.

Desventajas:

- Existe un menor control en cuanto a componentes nativos.
- El tamaño de los archivos ejecutables puede ser excesivo.
- Para principiantes puede ser complicado familiarizarse con su sintaxis.

Una vez revisado las ventajas y desventajas que se adjuntan a cada lenguaje de programación para elaborar interfaces gráficas se seleccionó como mejor opción el lenguaje de programación de Python, además de demostrar la flexibilidad que tiene este lenguaje para elaborar no solamente programas de procesamiento de datos sino también la capacidad de brindar al usuario la posibilidad de desarrollar interfaces gráficas e interactuar con datos en línea.

En base a los requerimientos listados, se detalla el procedimiento para desarrollar la aplicación de escritorio a utilizar para procesar y mostrar los datos obtenidos de la granja solar.

2.4.2 Autenticación de usuario

Para iniciar con el desarrollo de esta aplicación, se comenzó trabajando en el diseño de la pantalla de autenticación de usuario. Esta pantalla es necesaria para evitar que la información obtenida de la granja solar esté vulnerable al no contar con un método de protección ante el uso no autorizado de la aplicación desarrollada. Por esto, se empezó diseñando el estilo de esta pantalla en la que se dejan dos entradas de texto para poder ingresar el usuario y la contraseña respectivos. Además de contar con el botón de acceso que una vez presionado realiza la autenticación de los datos ingresados y, en caso de existir coincidencias, se muestra la pantalla principal. Caso contrario emitirá un mensaje de error indicando que las credenciales ingresadas son incorrectas.

Ilustración 35. Ven/tana de inicio de sesión de usuario.

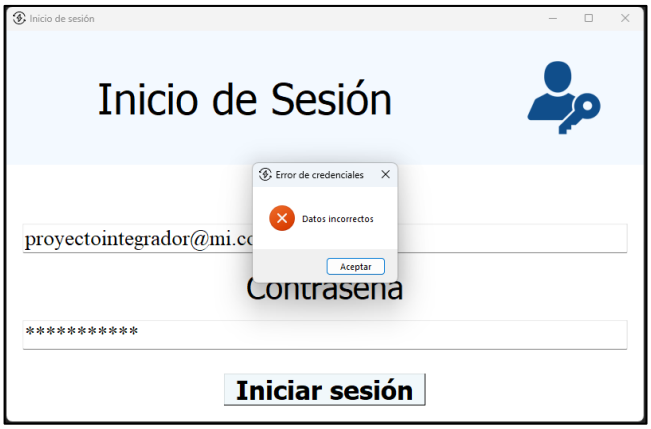


Ilustración 36. Mensaje de error emitido al ingresar credenciales incorrectas.

En este caso, por motivos de seguridad el programa solamente indica que las credenciales son incorrectas, más no indica si se ha encontrado coincidencias con alguna de las credenciales ingresadas, ya sea con el usuario o la contraseña.

2.4.3 Menú principal

En la pantalla de menú principal encontramos las funciones que están disponibles en la aplicación. El menú principal está dividido en 4 secciones que conforman toda la pantalla máster.
El encabezado del menú principal contiene información de la fecha actual y el nombre de la empresa para quien se desarrolló esta aplicación.
El cuerpo de esta aplicación está dividido en 2 secciones, la primera contiene la opción de determinar las pérdidas de energía cuando se ha producido un aparada de planta y la opción que nos permite estimar el P. R. diario de la planta.
La segunda sección contiene información acerca de la energía producida en intervalos definidos ya sea por hora, por día y el apartado que permite consultar los valores almacenados en el histórico.
Finalmente, la tercera sección contiene los widgets que funcionan como indicadores para determinar si el dispositivo está funcionando con el adaptador de corriente, la batería de respaldo o si el dispositivo se mantiene sin conexión a internet.

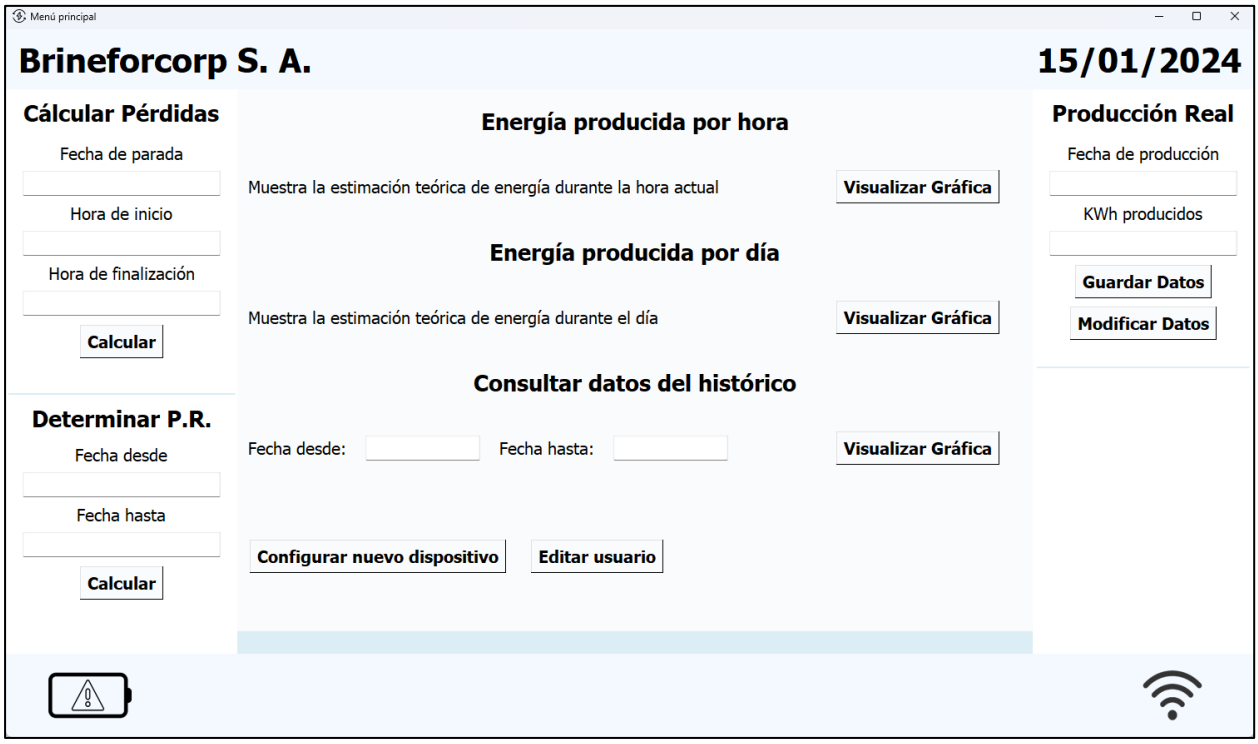


Ilustración 37. Ventana de menú principal de la aplicación.

2.4.4 Conexión a Firebase.

Para una mejor organización del código implementado se ha decidido seccionar por funciones cada apartado que procesa la información internamente. Para interactuar con la base de datos en línea necesitamos instalar las librerías de Firebase disponibles para Python, si esta no se instala o se instala de manera incorrecta se puede generar errores de funcionamiento durante la ejecución del código que permite acceder y obtener los datos de la base generada.

Para instalar la librería simplemente debemos ingresar la siguiente línea de código en el terminal de Python y presionar “Enter”.

- *Pip install firebase_admin*

Luego de instalar esta librería debemos llamarla de la siguiente forma:

- *from firebase import firebase*

Finalmente, procedemos a escribir el código que nos permitirá acceder a la base de datos en línea.

- *datos = firebase.FirebaseApplication('Dirección del archivo', None)*

En la sección de "Dirección del archivo" seleccionamos el link que se genera al crear una nueva carpeta para almacenar datos en línea.

- *datos = datos.get('Dirección carpeta', '')*

Luego creamos una variable que contendrá los datos bajados de una carpeta determinada, en la sección de "Dirección carpeta" ingresamos la dirección de la carpeta en la base de datos en línea de donde queremos extraer la información. Hecho esto ya tenemos los datos almacenados en la carpeta indicada. Es necesario recalcar que la información se descarga en forma de un diccionario en donde encontramos los "keys" que son el I.D. de la información y los "values" que contienen el valor numérico o string almacenado en la base de datos.

2.4.5 Presentación de gráficos

Los gráficos se generan utilizando librerías de matplotlib disponibles también en línea. Para instalarlas escribimos el siguiente comando en el terminal de Python.

- *pip install -U matplotlib*

Una vez finalizada la descarga procedemos a llamar a la librería para poder utilizarla dentro de nuestro código.

- *import matplotlib.pyplot as plt*

Esta librería nos permite realizar gráficos y editar como queremos que se observen estos modificando parámetros como el tipo de gráfico a realizar, los colores del gráfico y la ubicación de este dentro de una ventana. Para realizar el gráfico basta con enviar dos listas que representan la información que se representará en el eje "x" y en el eje "Y" de nuestra gráfica.

Es necesario mencionar que la lista que corresponde a la información ubicada en el eje "X" debe ser una lista con strings a diferencia de los valores contenidos en el eje "Y" que corresponden a datos enteros.

La siguiente línea de código muestra la forma en la que estas listas son utilizadas para realizar el gráfico correspondiente.

- *ax[0].bar(Datos eje X, Datos Eje Y, color = 'b')*

Finalmente, lo que observamos si no se han producido errores de escritura es una gráfica que en este caso se ha seleccionado la opción de mostrar barras que representan la estimación de energía producida durante los días mostrados en el gráfico.

Ilustración 38. Gráfica proporcionada por la librearía matplotlib.

2.4.6 Entradas de texto para el usuario

Las entradas de texto son útiles cuando se necesita ingresar información para realizar búsquedas o simplemente, de forma general, para ingresar información que puede ser relevante para ejecutar alguna acción dentro de nuestro código. Como un ejemplo sencillo, se presentan las entradas de texto disponibles en el apartado de "consultar datos del histórico" en donde el usuario tendrá la opción de ingresar un rango de fechas de las que queremos extraer la información y mostrarla en pantalla.

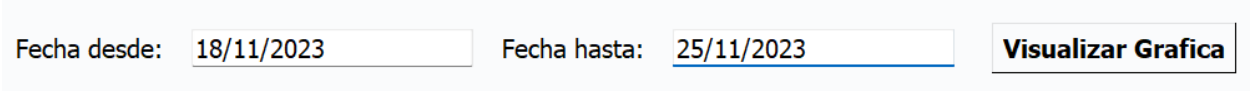


Ilustración 39. Ejemplo de las entradas de texto implementadas dentro de la aplicación.

En este caso en específico, esta información será tomada por el código y utilizada para realizar las búsquedas correspondientes al rango de días que han sido ingresados para realizar la búsqueda de información.

Para definir entradas de texto dentro de nuestro código podemos colocar las siguientes líneas de código.

- *self.NombreEntradaTexto = ttk.Entry(NombreFrame, Font=('FuenteTexto', 15))*
- *self.NombreEntradaTexto.pack(fill=tk.X, padx=20, pady=1)*

Este código generará una caja de texto con el formato de texto añadido. Tipo de letra: Tahoma y su tamaño a 15 puntos. Además de definir si queremos que esta caja de texto ocupe todo el ancho de la ventana y los valores de separación tanto por el lado del eje "X" y por el lado del eje "Y".

2.4.7 Configuración de un nuevo dispositivo

El apartado para configurar un nuevo dispositivo también se encuentra programado como una función por separado. Esta función fue diseñada para configurar un nuevo dispositivo en caso de que el dispositivo IoT principal hay sufrido daños físicos y este haya quedado inservible. En caso de llegar a suceder lo mencionado, el usuario final tendrá la opción de utilizar un dispositivo de respaldo para seguir muestreando datos de la granja solar. De esa manera se garantiza que la información perdida durante este tipo de percances sea mínima y no afecte a los valores finales obtenidos.

Dentro de esta función se ha implementado también un apartado para observar aquella información que por algún motivo no ha podido subirse a la base de datos y que se encuentra

almacenada en el dispositivo IoT hasta que la conexión a internet se haya reestablecido y los datos comiencen a subirse de forma ordenada garantizando que esos datos no se pierdan cuando haya sucedido algún problema con nuestra conexión a internet o simplemente el dispositivo se haya desconectado por algún error producido durante la operación de este. Esta información además de ser leída también se puede respaldar en nuestro ordenador como un archivo con formado CSV. Permitiendo al usuario respaldarla en caso de que la perdida de conexión a internet se haya extendido durante varias horas por algún motivo

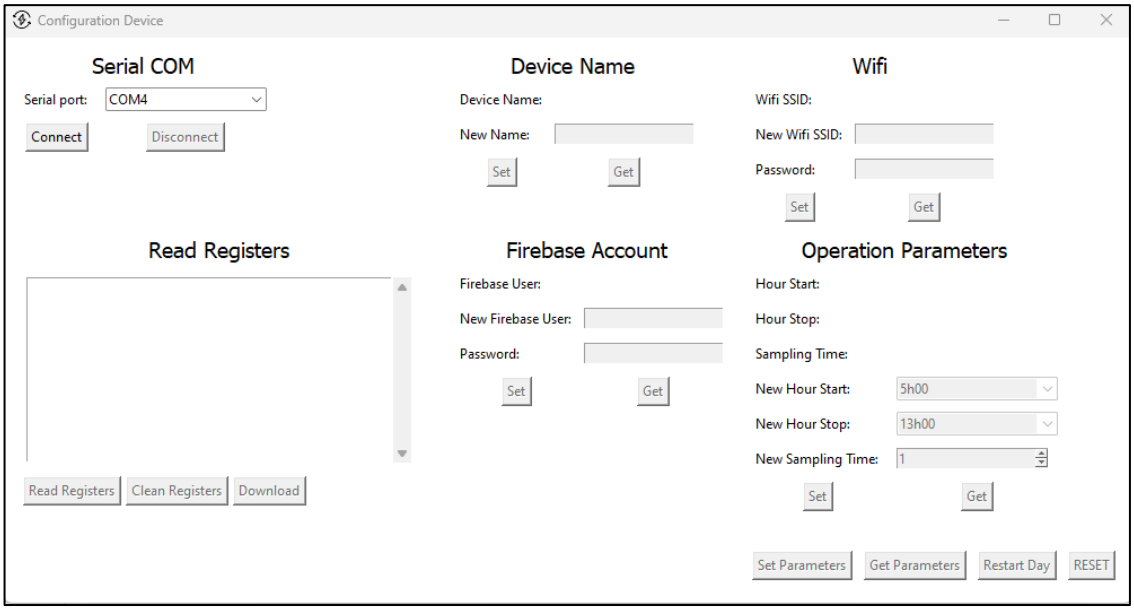


Ilustración 40. Ventana de configuración para un nuevo dispositivo.

2.4.7 Widgets

Finalmente, se han implementado widgets que tiene la función de indicar al usuario varios aspectos acerca del funcionamiento del dispositivo. Como una rutina en un bucle continuo el programa chequea si el dispositivo se encuentra operando con su fuente de alimentación principal proveniente de un adaptador de 15 V y, en caso de no ser así, chequea si el dispositivo se encuentra operando con la batería de respaldo y monitorea el estado de carga de la batería para

tomar las acciones pertinentes en caso de que la falla de la fuente de alimentación principal se esté extendiendo por algún daño mucho más grave.

Ilustración 41. Widgets utilizados para indicar el funcionamiento de la fuente principal o de la batería de respaldo.

Así mismo, el dispositivo cheque cada cierto tiempo si el ordenador se encuentra conectado a internet, de ser así, muestra ya sea el widget de adaptador o el widget de la batería de respaldo. En caso de no tener conexión, aparecerá un widget que indicará que no se tiene una conexión a internet.

Ilustración 42. Widgets utilizados para demostrar la conexión de la aplicación a internet.

2.4.8 Detalle de funciones incorporadas a la aplicación.

En base a las necesidades del usuario se estableció un conjunto de funciones destinadas a facilitar la interpretación de la información obtenida de la granja solar. Cada una de estas funciones se ajusta a los requerimientos de visualización de información en base a estimaciones obtenidas en base a las condiciones meteorológicas del sitio.

En el menú principal de la aplicación contenemos 4 apartados destinados a mostrar información que se adaptan a las sugerencias del cliente. Cada uno de estos apartados permite interpretar los

datos de producción de la granja solar de forma sencilla, pero proporcionando información importante acerca del funcionamiento de la granja solar. En adelante, se detallará con mayor precisión que información muestra cada sección y como se interpreta.

2.4.9 Producción de energía por hora

En esta sección el usuario puede observar el comportamiento de la curva de producción de energía estimada durante la última hora en curso. Esto permite que el usuario observe el comportamiento de la curva durante promedios generados en intervalos muy cortos de tiempo, dado la posibilidad de que, en caso de notar anomalías en el comportamiento de las curvas, acudir al sitio a revisar que no existan fallas o condiciones que esté ocasionando un comportamiento irregular de los módulos fotovoltaicos. Esta función internamente obtiene la información cargada durante la última hora en curso y la promedia en grupos de 5 datos por barra mostrada, es decir, en la gráfica obtendremos un máximo de 12 barras que representan los promedios de producción total en una hora de muestreo de información.

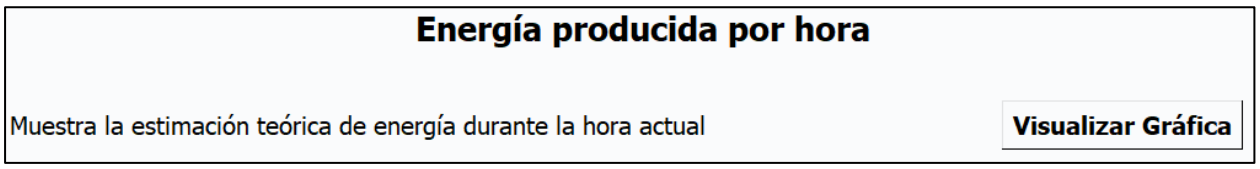


Ilustración 43. Apartado para visualizar la producción teórica de energía por hora.

El objetivo de observar el comportamiento de la curva durante estos cortos lapsos es de relacionarlo en conjunto con el nivel de radiación del sitio, pues existirán ocasiones en donde por pocos minutos la radiación solar sea muy alta produciendo picos de energía captada, así como en ocasiones tendremos caídas de estos valores produciendo bajadas considerables. De esta manera podemos observar si la producción real de energía se ajusta a estas condiciones y determinar si los módulos fotovoltaicos funcionan correctamente.

2.4.10 Producción de energía por día

Esta segunda sección del programa permite al usuario observar la estimación de producción de energía, pero agrupada en esta ocasión por lapsos de tiempo más extendidos. Esta funciona interna del programa obtiene la información guardada en el sistema desde la hora de inicio de operaciones en la planta hasta la hora en la que se realiza la consulta d estos valores, agrupando los datos por cada hora en la que fueron obtenidos y promediándolos para presentar en el gráfico una barra por cada hora de producción, dando un total de 12 barras referentes a la estimación de energía producida diariamente.

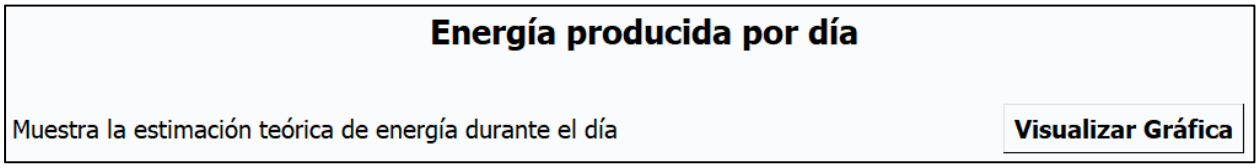


Ilustración 44. Apartado para visualizar la producción teórica de energía por día.

El objetivo de visualizar en este apartado una gráfica seccionada por horas es de que en condiciones normales podamos observar los picos de producción alrededor de las 12:00 h que es en donde la radiación solar se encuentra en sus mayores valores debido a la ubicación geográfica en la que se sitúa la granja solar, hay que tener en cuenta que este comportamiento no siempre será observado puesto que lo descrito anteriormente se da en condiciones atmosféricas constantes; sin embargo, en un día normal, podemos obtener altos niveles de radiación en distintos horarios y no necesariamente alrededor de la hora especificada. Todo dependerá de las condicione atmosféricas a las que se encuentre sometida la granja solar.

2.4.11 Consultar datos del histórico

En la sección de histórico, el usuario puede acceder a la información almacenada en la base de datos y que corresponde a datos obtenidos durante los días pasados. En esta sección el usuario

tiene dos entradas de texto en donde podrá ingresar la fecha desde donde se quiere consultar, hasta la fecha que indica hasta que día se quiere consultar esta información. El programa internamente realizará un barrido durante el intervalo de tiempo ingresado y promediará la información por cada día de producción, mostrando una barra por cada día consultado siempre y cuando la consulta sea para un intervalo de tiempo que se encuentra dentro de un mismo mes. Sin embargo, también puede ingresar un rango de fechas que abarquen diferentes meses de producción en donde el programa procesará los datos de tal manera que elabore promedios mensuales, dándonos como resultado una gráfica en dónde cada barra representará un mes de datos almacenados.

En caso de que se encuentre un día sin datos almacenados o que se tengan problemas para acceder a la información referente a un día de producción, la gráfica representará ese día sin la barra omitiendo el error que se puede generar por alguna de las dos posibilidades presentadas.

Ilustración 45. Apartado para consultar los datos de producción almacenados en el histórico.

Actualmente, la planta fotovoltaica lleva un registro a mano de la energía producida; sin embargo, no lleva un registro del estimado de producción que le permita observar el comportamiento de su planta generadora sobre todo con el detalle en cuanto a las horas pico de producción como se puede observar en estas gráficas. Con la base de datos en línea se puede saber que días o que épocas son buenas para la generación de energía en base a los datos almacenados y revisar qué días tuvieron bajas de producción que pueden darse por muchos factores que en la mayoría de los casos pueden corregirse a través de la planificación de mantenimientos ya sean preventivos o correctivos.

Es importante recordar que las condiciones de clima son muy cambiantes, por lo tanto, la producción de energía también se puede ver afectado por esto.

2.4.12 Calculo de pérdidas y determinación de P. R.

Es muy importante poder conocer las pérdidas que se pueden generar por paradas de planta ocasionadas ya sea por la ejecución de mantenimientos preventivos o correctivos o por fallas durante la operación de los equipos. En esta sección se brinda al usuario la capacidad de poder estimar las pérdidas ocasionadas por paradas de planta, en base a las condiciones meteorológicas del sitio en donde se encuentra ubicada la granja solar. De esa manera y en base a los datos obtenidos se pueden realizar estudios acerca de los horarios en donde la producción de energía es baja y reprogramar las actividades de parada para esos horarios, maximizando de esa forma el rendimiento de la granja solar.

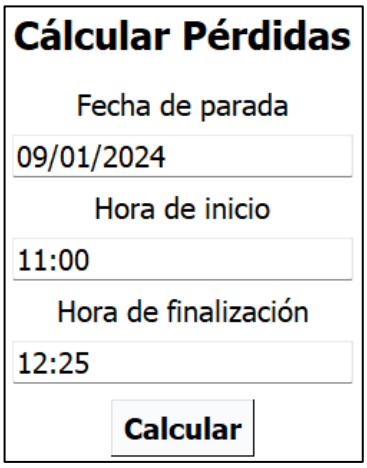


Ilustración 46. Ventana para realizar consultas de la energía perdida durante intervalos de tiempo de paradas de planta.

Para lograr este objetivo, se otorga al usuario un apartado en donde deberá de ingresar información acerca de la fecha en la que hubo una interrupción de producción en conjunto con el rango de horas que la planta estuvo sin producir para estimar la energía que no pudo ser captada durante ese intervalo de tiempo. Esta es el motivo por el que se decidió añadir una batería de respaldo al dispositivo IoT, para que, en caso de desconexión o apagones en el sitio, el dispositivo pueda seguir captando información y guardándola para poder estimar los valores de producción

perdidos durante las actividades ejecutadas; de esa manera aumentar la eficiencia de los trabajos realizados para incurrir en menor cantidad con las paradas por motivos de mantenimiento.

De la misma manera, contamos con un apartado que brinda al usuario la capacidad de conocer el factor de rendimiento de su planta fotovoltaica durante un día de producción. Esta es una de las funciones más importantes del programa, puesto que en base a esta gráfica se pueden comparar los datos teóricos, así como los datos reales de producción de energía en la granja solar la granja solar. En esta sección, el usuario

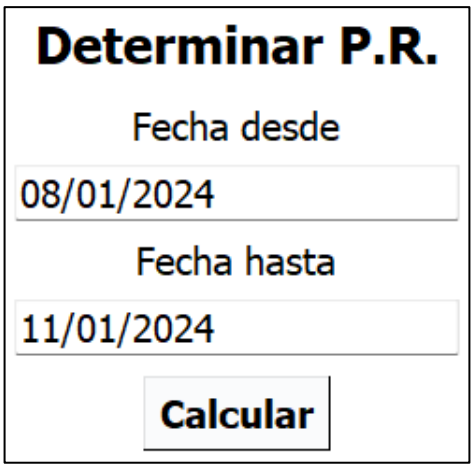


Ilustración 47. Ventana para consultar el rendimiento de la granja solar.

En el apartado el usuario deberá ingresar un intervalo de tiempo de hasta 7 días para determinar el P. R. de su instalación. El programa se encargará de buscar esa información y mostrará un conjunto de 2 barras consecutivas que representarán la estimación teórica en azul y el valor producido real en color verde. Estas barras, en condiciones normales, deberán ser similares en cuanto a su tamaño, puesto que se pretende que la granja solar opere bajo condiciones físicas favorables. De la misma manera, en caso de que las gráficas muestren una variación muy notoria en cuanto a sus tamaños, el usuario deberá revisar los parámetros de programación en el código o en su defecto, deberá planificar una parada de planta para realizar un mantenimiento para determinar el estado de los paneles y encontrar el motivo por el que el rendimiento de su granja solar sea demasiado inestable.

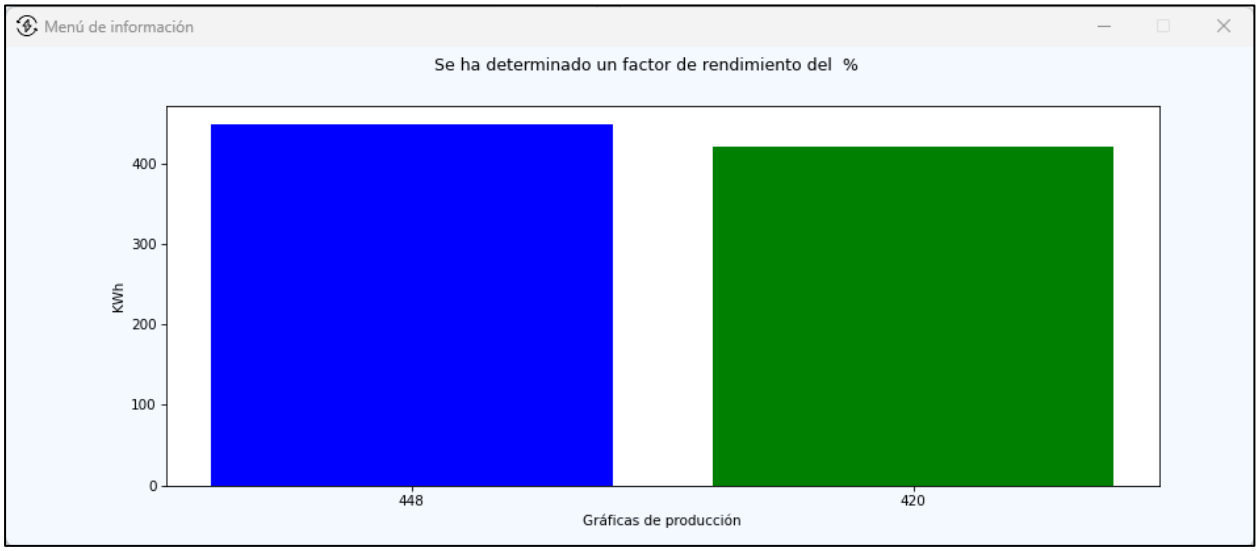


Ilustración 48. Ejemplo de la gráfica obtenida al realizar las consultas para determinar el P. R. de la planta.

2.4.13 Producción real

Finalmente, este apartado permite al usuario crear un respaldo de la información acerca del producido real de energía de la granja solar. Este dato deberá ser ingresado diariamente por parte del usuario, pues, esto ayudará a respaldar la información en la nube, así como permitir al programa procesar esta información para poder determinar las gráficas para comparar los valores de producción con los valores estimados. El usuario cuenta con 2 apartados que deberá llenar para poder almacenar un registro. Así mismo, el usuario tiene la facilidad de modificar un registro a su conveniencia.

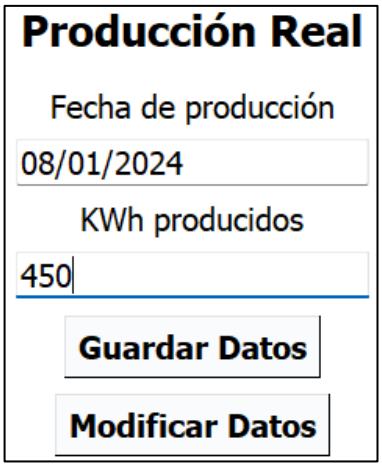


Ilustración 49. Apartado para ingresar datos de producción real de la granja solar.

Como se indicó anteriormente, el objetivo principal de esta función es facilitar al usuario la accesibilidad y la utilización de esta información para realizar otras funciones dentro del programa como la indicada en la sección de “Determinar P.R.”.

CAPÍTULO 3

3.1 Resultados y análisis

Se diseñó e implementó un sistema IoT que consta de dos elementos principales, un dispositivo encargado de recopilar y subir la información proveniente de la granja solar y una aplicación capaz de procesar y presentar gráficos que representan las estimaciones de producción teórica de energía y qué se desarrolló en base a las necesidades del cliente.

El dispositivo desarrollado cumple con las condiciones de funcionamiento planteadas en el capítulo anterior dando al usuario los datos necesarios para poder estimar los valores de producción de energía de la granja solar.

3.2 Funcionamiento del dispositivo IoT

3.2.1 Funcionamiento de la fuente redundante

Para probar el funcionamiento de este circuito se conectó a la vez tanto el adaptador de corriente, así como su batería de respaldo; sin embargo, gracias al circuito diseñado el dispositivo se mantiene funcionando con la energía proporcionada por el adaptador. Durante estas pruebas se realizaron las mediciones de corriente y voltaje consumidos por el dispositivo en cada condición de operación.

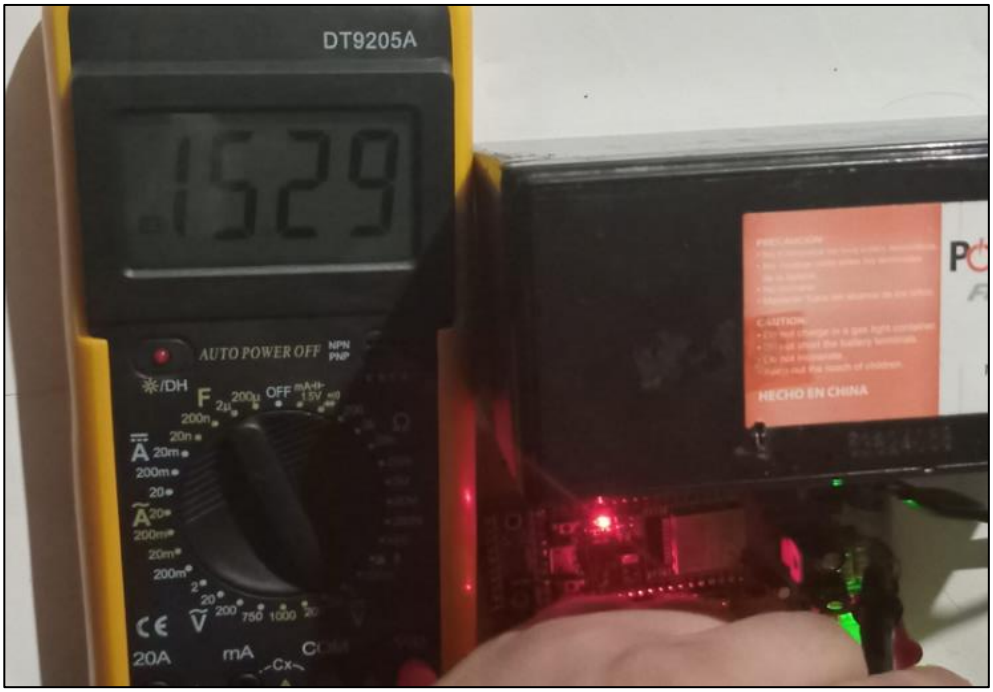


Ilustración 50. Medición de la tensión de entrada proporcionada por la fuente de alimentación principal.

En la *Ilustración 50* adjunta se aprecia la medición de la tensión de entrada del dispositivo, al permanecer conectada la fuente de alimentación principal, el led en color verde se mantiene encendido. Al medir la corriente de consumo de la ESP32 obtenemos un valor similar al especificado la *Tabla 2* correspondiente al capítulo 2 de este informe y su valor lo podemos apreciar en la *Ilustración 51*.

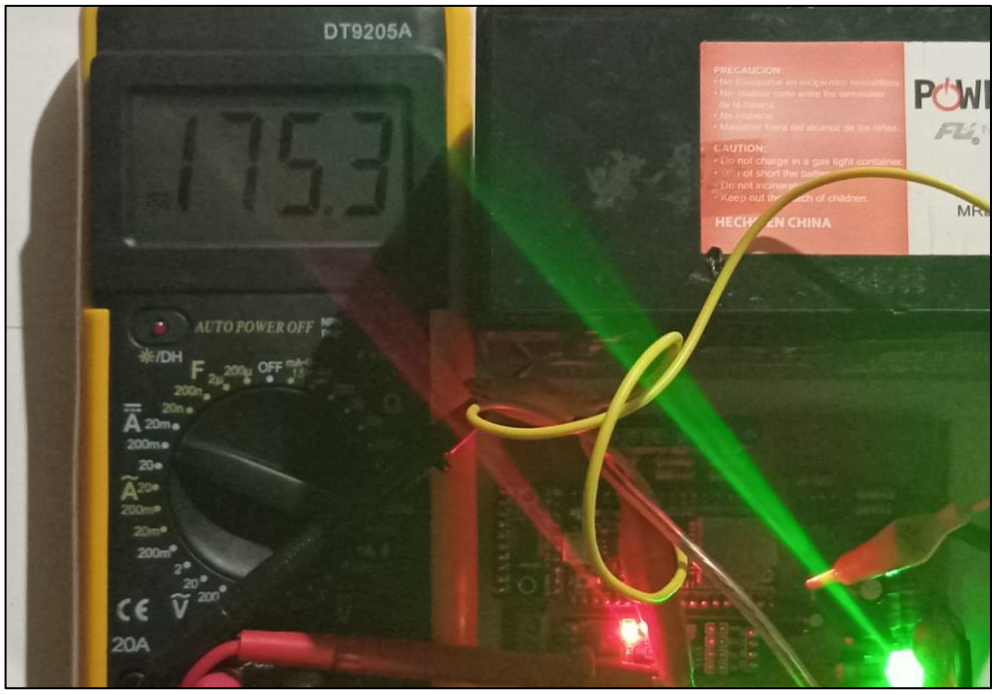


Ilustración 51. Medición de la corriente proporcionada por la fuente de alimentación principal.

Para observar el funcionamiento de la batería de respaldo, se procedió a desconectar el adaptador de corriente y se procedió a medir el voltaje de la batería, mismo que podemos apreciar en la *Ilustración 52*.

Ilustración 52. Medición de la tensión de entrada proporcionada por la fuente de respaldo.

En este caso, bajo las mismas condiciones de operación, se observa que el consumo de corriente disminuye (ver *Ilustración 53*), y eso se debe a que el led indicador ya no se enciende debido a que la fuente de alimentación se ha desconectado. Sin embargo, este consumo no es el nominal ya que para las pruebas de funcionamiento no se conectaron los sensores de la estación meteorológica.

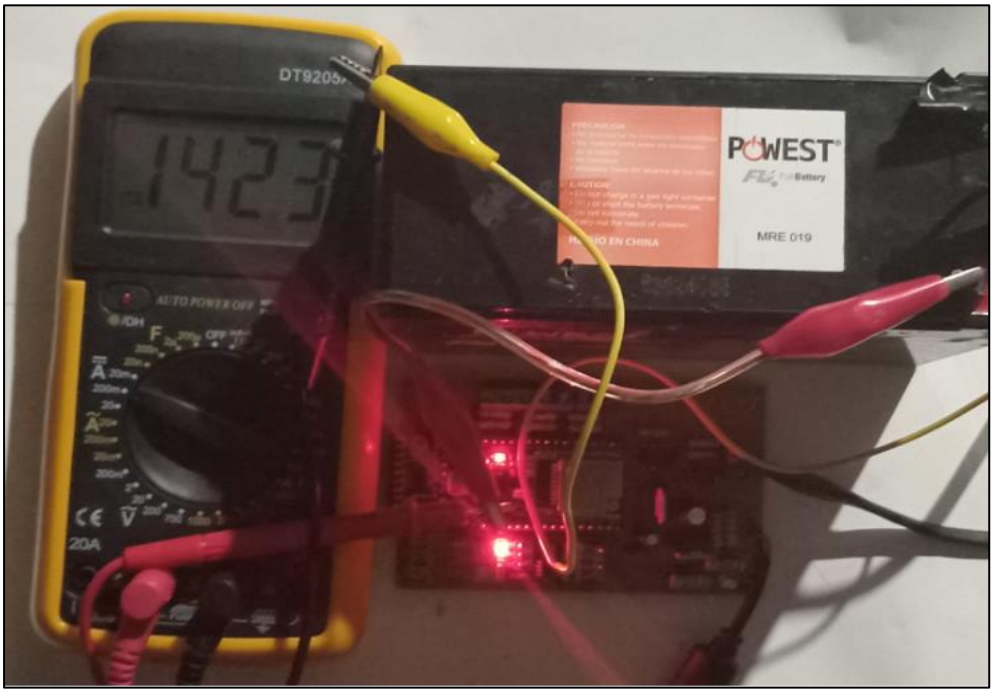


Ilustración 53. Medición de la corriente proporcionada por la batería de respaldo.

3.2.2 Funcionamiento del módulo de comunicación

Para realizar las pruebas del módulo de comunicación se programó una ESP32 adicional (ver *Ilustración 54*) como un esclavo, simulando los sensores instalados en la planta fotovoltaica. Este ejercicio se realizó para poder visualizar la señal pulsante correspondiente a la transmisión de datos tal como se aprecia en la *Ilustración 55*.

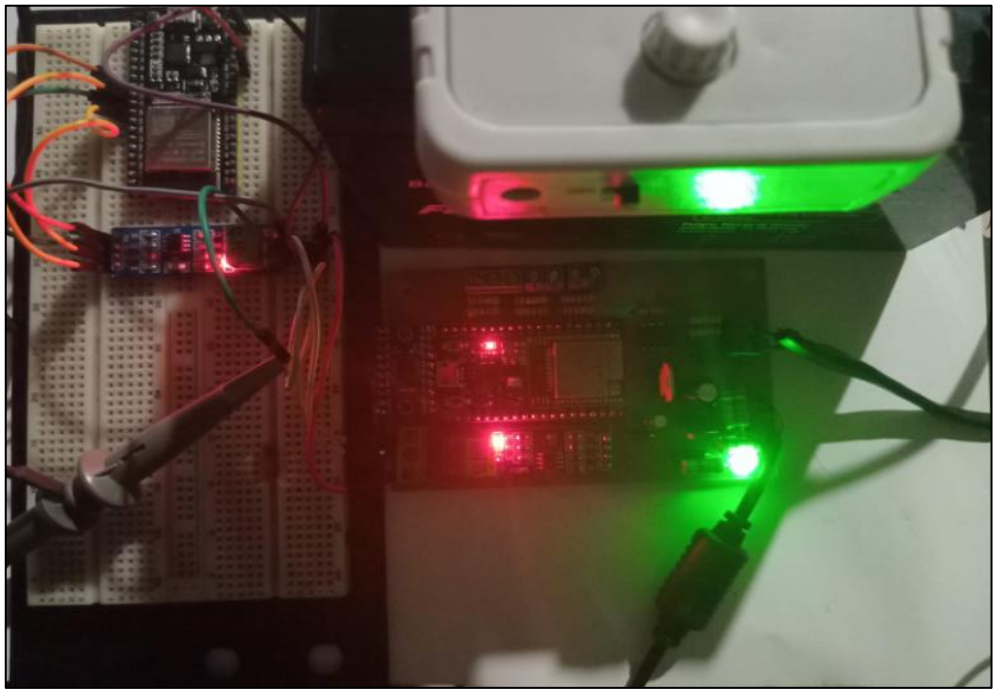


Ilustración 54. Implementación de un esclavo para probar el funcionamiento del módulo de comunicación en la PCB.

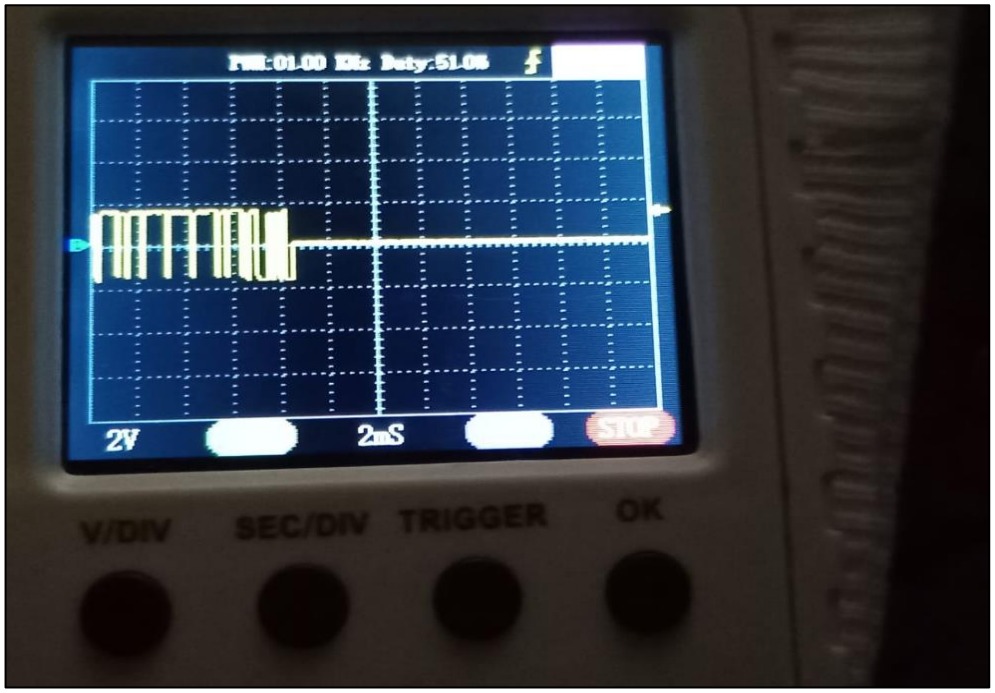


Ilustración 55. Visualización de la señal digital producida al transferir datos entre el esclavo implementado y la PCB.

Los datos proporcionados por la ESP32 operando como esclavo fueron subidos a la base de datos a un usuario de pruebas y los valores se observan en la *Ilustración 56*.

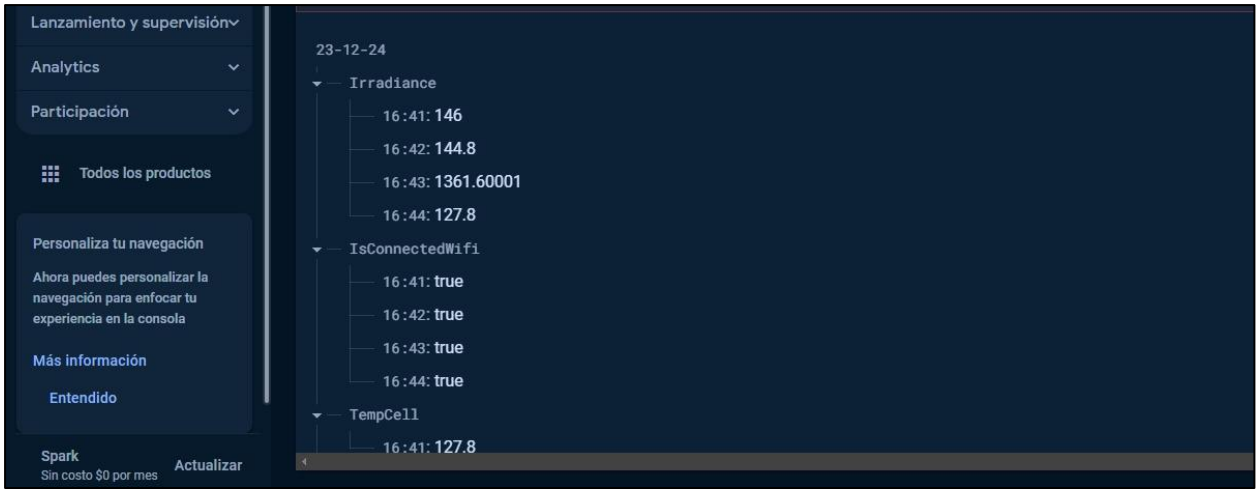


Ilustración 56. Datos proporcionados por el esclavo implementado que fueron cargados durante las pruebas de funcionamiento del módulo de comunicación.

3.2.3 Funcionamiento del módulo acondicionador de señal

Para verificar el funcionamiento de este circuito se midió el voltaje proporcionado por la batería de respaldo con 4 niveles distintos de tensión. Estos valores obtenidos son cargados a la base de datos como valores porcentuales correspondientes al nivel de carga de la batería implementada. Las imágenes adjuntas representan los niveles de tensión medidos y los valores porcentuales cargados a la base de datos durante su funcionamiento.

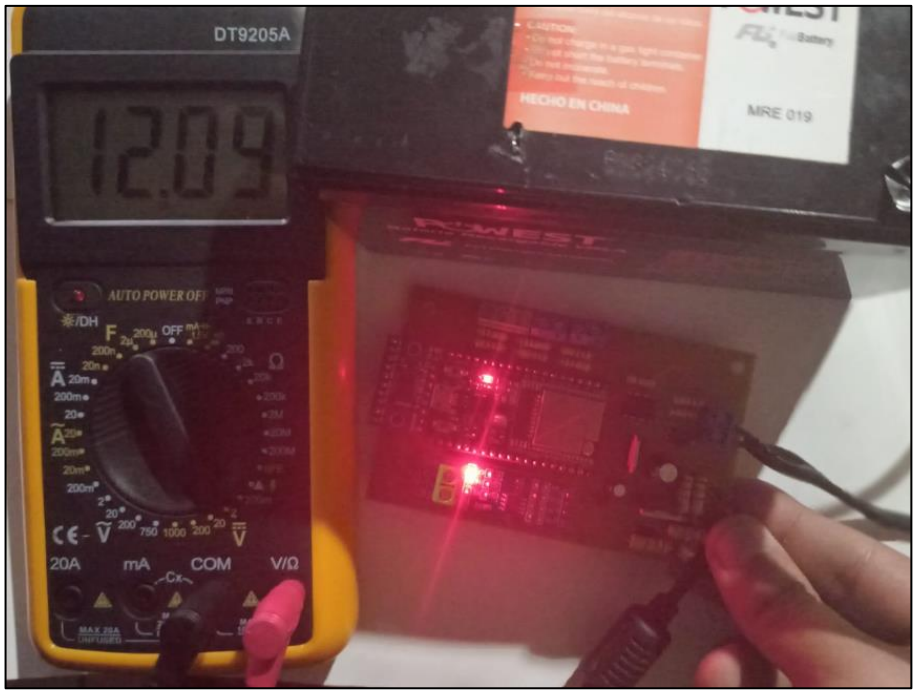


Ilustración 57. Pruebas de funcionamiento del circuito medidor de voltaje de la batería con un nivel de 12.09V.

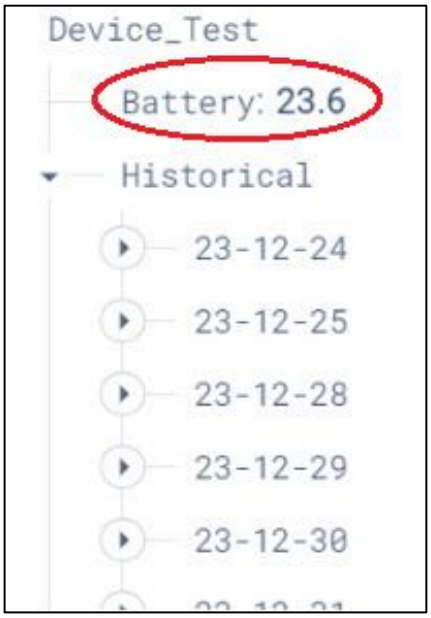


Ilustración 58. Datos cargados a la base de datos indicando el nivel de carga al 23.6%.

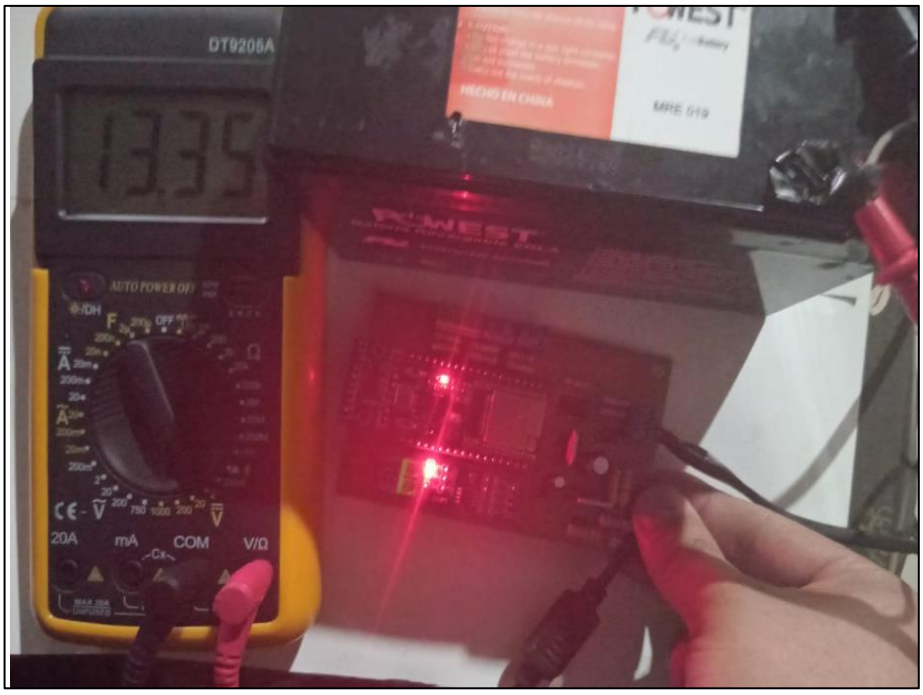


Ilustración 59. Pruebas de funcionamiento del circuito medidor de voltaje de la batería con un nivel de 13.35V.

Ilustración 60. Datos cargados a la base de datos indicando el nivel de carga al 41.3%.

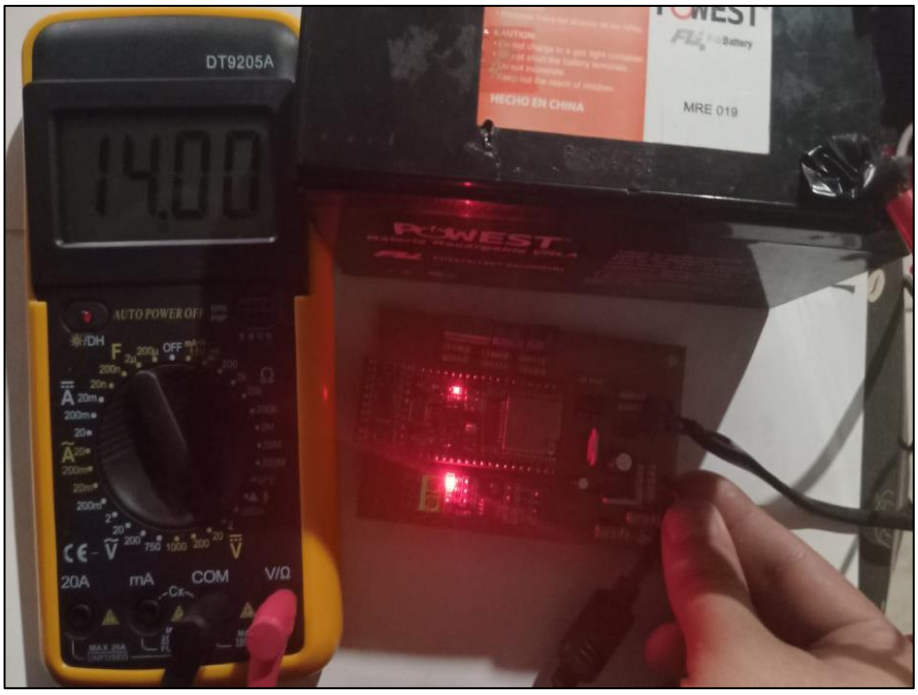


Ilustración 61. Pruebas de funcionamiento del circuito medidor de voltaje de la batería con un nivel de 14.00V.

Ilustración 62. Datos cargados a la base de datos indicando el nivel de carga al 63.9%.

Ilustración 63. Pruebas de funcionamiento del circuito medidor de voltaje de la batería con un nivel de 14.58V.

Ilustración 64. Datos cargados a la base de datos indicando el nivel de carga al 85.6%.

3.2.4 Funcionamiento del circuito identificador de la fuente principal

Para probar el funcionamiento de este circuito, se procedió a medir el voltaje en el pin de alimentación de 15V que, a su vez, hace que el transistor conduzca y el pin de entrada "MAIN" se mantenga en 3.0V aproximadamente (ver *Ilustración 65*), haciendo que la ESP32 interprete este valor como un 1 lógico y envíe esta información a la base de datos (Ver *Ilustración 66*) indicando que la fuente principal está alimentando al dispositivo.

Ilustración 65. Circuito identificador de la fuente de alimentación principal emitiendo 3.00V (Fuente principal conectada).

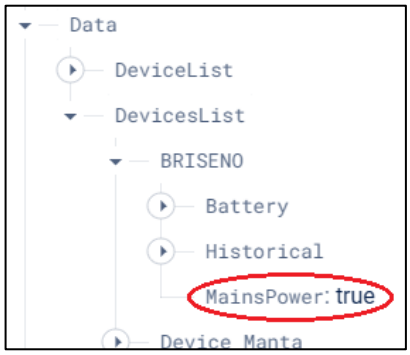


Ilustración 66. Base de datos recibiendo el valor de "True" correspondiente a la presencia de la fuente de alimentación principal.

La aplicación al leer esta información muestra un widget que permite saber el usuario que el dispositivo está funcionando con la batería principal sin necesidad de ir a ver el dispositivo al sitio (ver *Ilustración 67*).

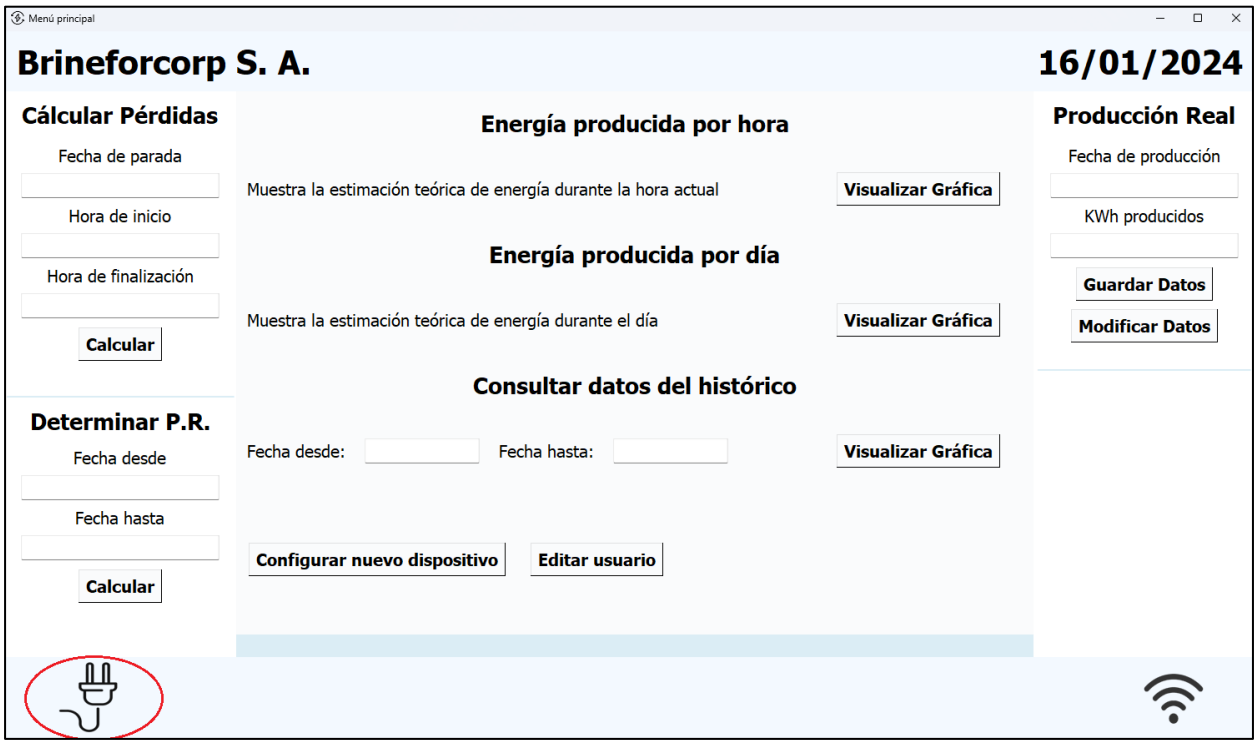


Ilustración 67. Aplicación mostrando el widget correspondiente al uso de la fuente de alimentación principal.

Sin embargo, al desconectar la fuente principal, evidenciamos como el transistor deja de conducir; por lo tanto, la señal de 3.3V cae a 0V aproximadamente (ver *Ilustración 68*) lo que hace que la ESP32 interprete este valor como un 0 lógico y envíe esta información a la base datos (Ver *Ilustración 69*).

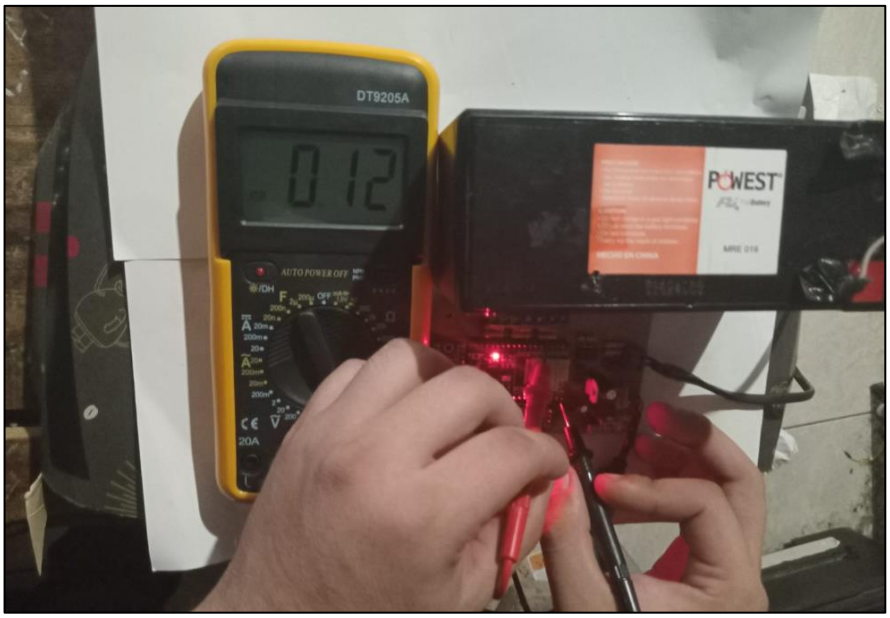


Ilustración 68. Circuito identificador de la fuente de alimentación principal emitiendo 0V (Fuente principal desconectada).

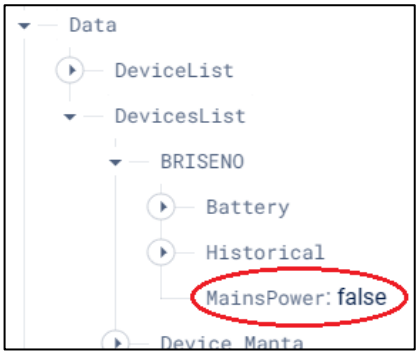


Ilustración 69. Base de datos recibiendo el valor de “False” correspondiente a la ausencia de la fuente de alimentación principal.

La aplicación al leer esta información muestra un widget que permite informar al usuario que el dispositivo ahora se encuentra funcionando con la batería de respaldo y gracias a la implementación del circuito acondicionador permite saber el nivel de carga que tiene la batería (ver *Ilustración 70*).

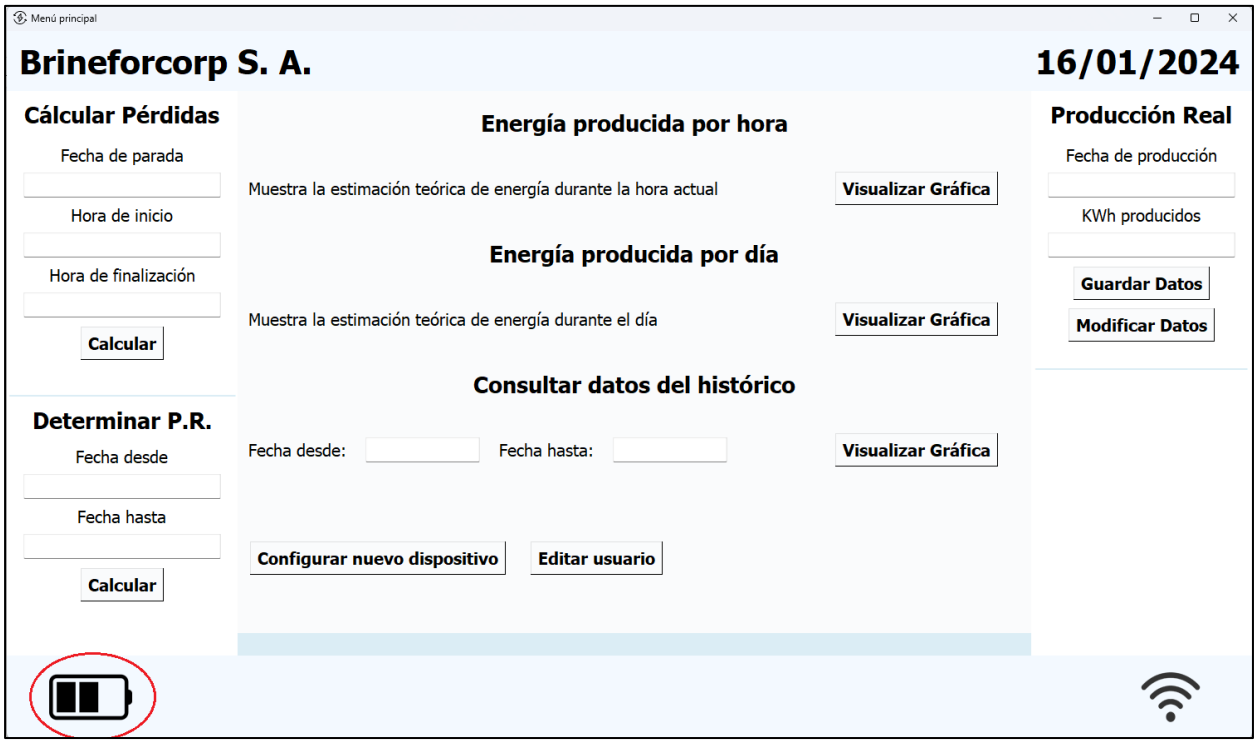


Ilustración 70. Aplicación mostrando el widget correspondiente al uso de la batería de respaldo.

3.2.5 Funcionamiento de esclavo Modbus

Esta última prueba al dispositivo IoT se la realiza con una aplicación que simula nuevamente a los sensores instalados en la granja solar. Es software permite enviar información a la ESP a través de un módulo de comunicación serial para corroborar el funcionamiento de la programación de datos en la ESP y fue la primera prueba que se realizó al iniciar con el diseño del dispositivo, esto debido a que fue necesario solucionar el primer problema correspondiente a la obtención de datos otorgados por los sensores de la estación de clima para poder continuar con el desarrollo de las demás partes del circuito que complementan su funcionalidad.

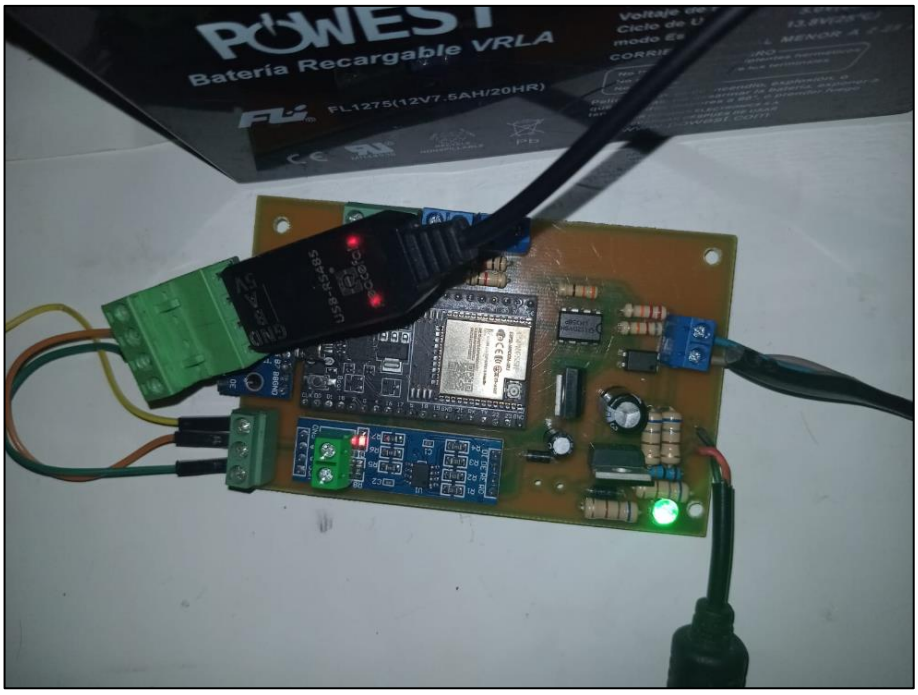


Ilustración 71. Prueba adicional del módulo de comunicación RS485 con un pórtico serial conectado al computador.

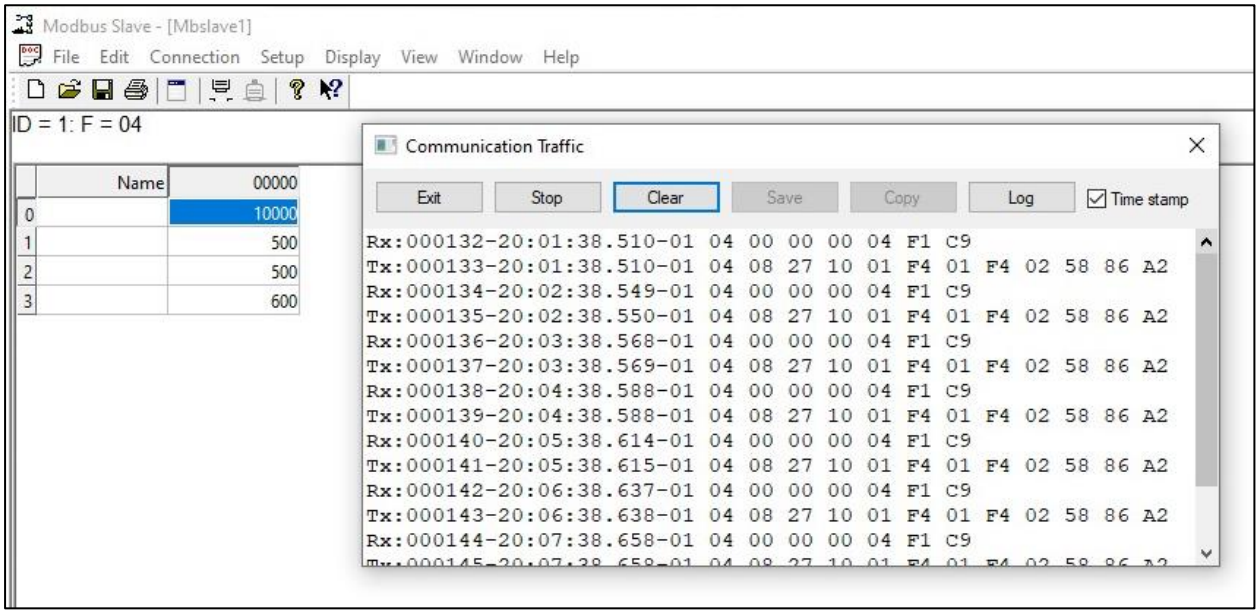


Ilustración 72. Verificación a través del pórtico serial de los datos enviados entre el esclavo Modbus y la PCB.

Ilustración 73. Información subida a la base de datos provenientes del esclavo Modbus.

3.3 Funcionamiento de la aplicación

Con el dispositivo implementado y funcionando se procedió a instalarlo en la granja solar para evidenciar su correcto funcionamiento y visualizar los datos que este iba cargando al sistema. Esto permitió comenzar a recopilar los datos que se muestran en pantalla y graficar las estimaciones de producción durante 8 días de muestreo consecutivo que más adelante serán comparados con los datos de producción real para determinar el factor de rendimiento de la granja solar.

3.3.1 Análisis de datos durante un intervalo de tiempo de 8 días

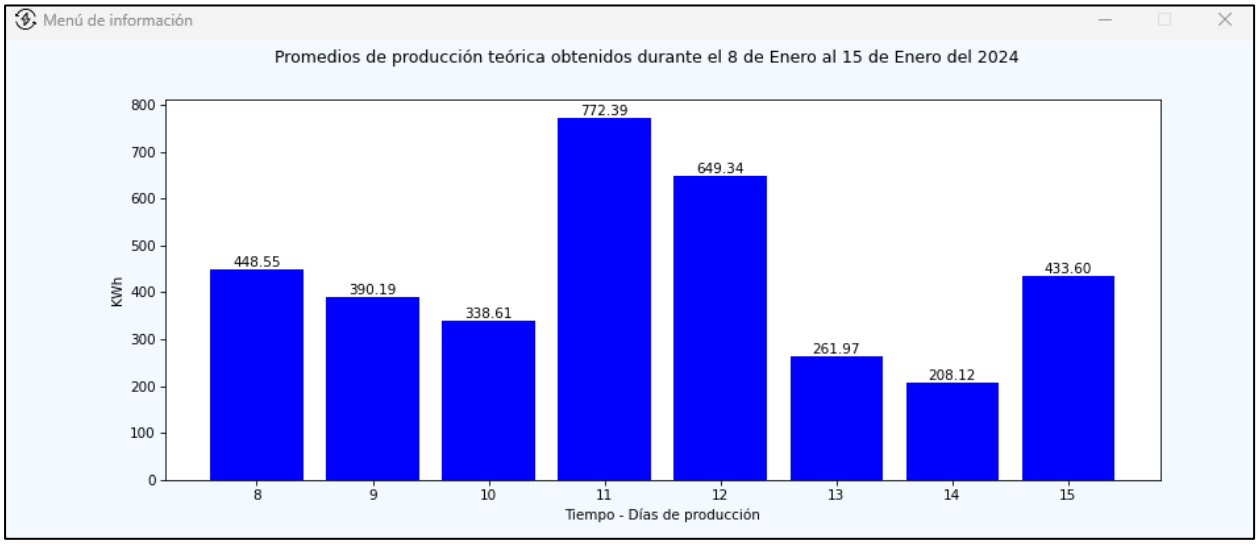


Ilustración 74. Promedios de producción teórico por día obtenidos durante 8 días de producción consecutivos.

En la gráfica podemos apreciar cómo los valores de producción teórica durante los primeros 3 días de recopilación de información son bajos a comparación del cuarto, quinto y octavo día. Los bajos niveles de producción teórica están ligados directamente con los niveles de radiación en el sitio en donde se encuentra ubicada la granja solar. Sin embargo, para el 13 de enero ya se planificó un mantenimiento debido a que la acumulación de polvo en los paneles fue demasiada pudiendo minimizar el rendimiento de estos.

3.3.2 Análisis de datos durante 2 días de producción

La *Ilustración 75* adjunta representa un día inestable en cuanto a las condiciones climáticas del sitio. En esta gráfica se evidenció como se produjeron cambios muy bruscos de producción de energía pasando de niveles muy altos a niveles muy bajos y viceversa, pero asemejándose a la curva de radiación presentada en la *Ilustración 7* adjunta en el capítulo 1

indicando que los picos de producción de energía alcanzados durante un día se deben ubicar alrededor de las 12:00h.

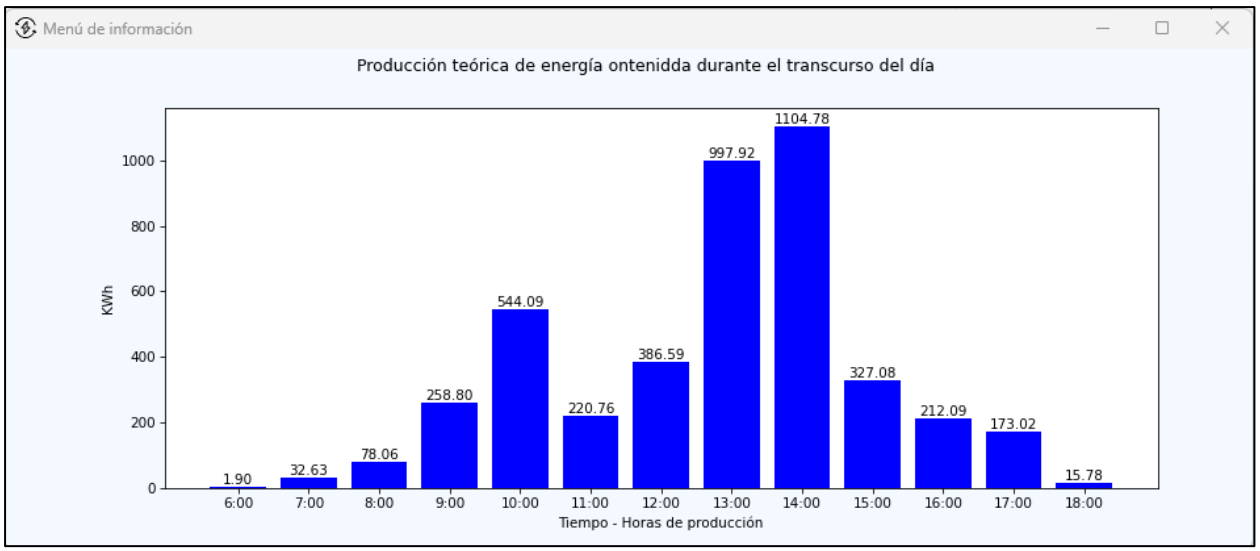


Ilustración 75. Comportamiento de un día de producción con condiciones meteorológicas muy irregulares.

La *Ilustración 76* adjunta representa un día muy estable en cuanto a las condiciones climáticas del sitio en donde se ubica la granja solar. Es esta gráfica se evidenció como la curva de producción teórica de energía tiene variaciones muy leves entre cada hora de producción y su forma se asemeja mucho a la curva de radiación presentada en la *Ilustración 7* adjunta en el capítulo 1.

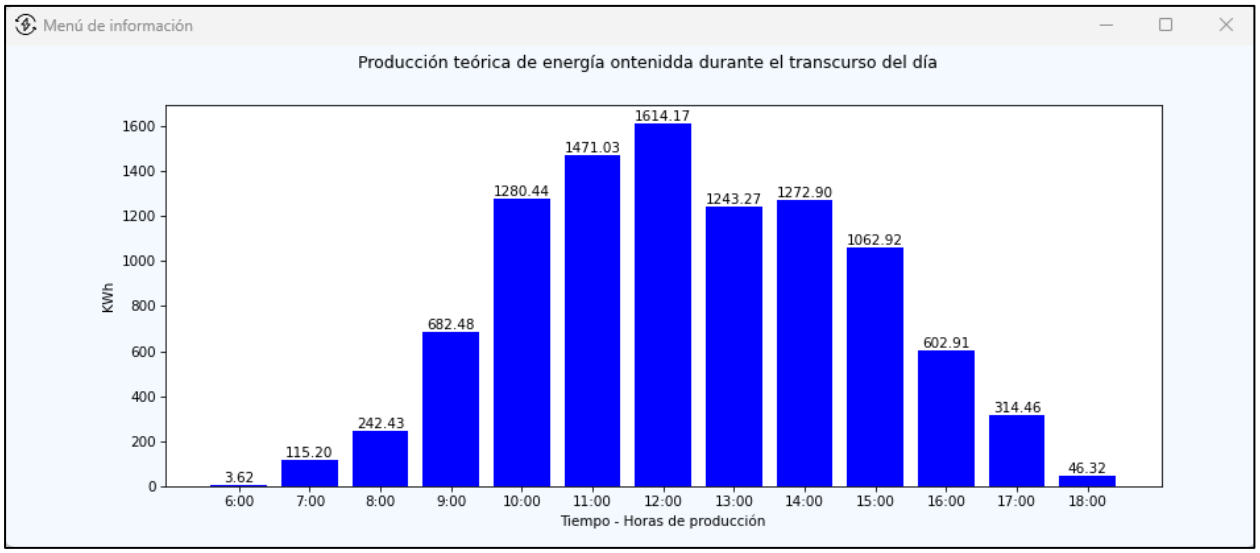


Ilustración 76. Comportamiento de un día de producción con condiciones meteorológicas muy regulares.

3.3.4 Detalle de costo de elaboración de sistema IoT.

N°	Elemento	Cantidad	Precio unitario	Precio total
1	Diseño electrónico	1	$ 800	$ 800
2	Programación ESP32	1	$ 1000	$ 1000
3	Desarrollo aplicación	1	$ 2000	$ 2000
			TOTAL	$ 3800

Tabla 8. Detalle de costos por desarrollo

N°	Elemento	Cantidad	Precio unitario	Precio total
1	Microcontrolador ESP32	1	$ 18,00	$ 18,00
2	Módulo MAX485	1	$ 1.75	$ 1,75
3	Resistencia de 10K 1/2W	1	$ 0,10	$ 0,10
4	Resistencia de 330 Ω 1/2W	1	$ 0,10	$ 0,10
5	Resistencia de 82 KΩ 1/2W	1	$ 0,10	$ 0,10
6	Resistencia de 22 KΩ 1/2W	1	$ 0,10	$ 0,10
7	Resistencia de 82 Ω 1/2W	6	$ 0,10	$ 0,60
8	Resistencia de 680 Ω 1W	4	$ 0.15	$ 0.60
9	Capacitor de 100uF 16V	1	$ 0,15	$ 0,15
10	Capacitor de 330uF 16V	1	$ 0,20	$ 0,20
11	Regulador LM7805	1	$ 0,75	$ 0,75
12	Optoacoplador PC817B	1	$ 0,50	$ 0,50
13	Circuito integrado LM358P	1	$ 0,75	$ 0,75
14	Transistor MOSFET canal P IRF9530N	1	$ 1.79	$ 1.79
15	Diodo fr107 100V 1A	2	$ 0.10	$ 0.20
16	Led RGB 5 mm	3	$ 0,25	$ 0,75
17	Led de color verde	1	$ 0,10	$ 0,10
18	Modulo HW-221	1	$ 2,50	$ 2,50
19	Adaptador de voltaje 15V 2A	1	$ 10,75	$ 10,75
20	Placa 105,9 x 62,5 mm con antisoldier	1	$ 15,65	$ 15,65
21	Batería VRLA 7.5AH	1	$ 20,00	$ 20,00
22	Bornera 2 pines 15mm	5	$ 0,10	$ 0,50
23	Bornea 3 pines 15mm	2	$ 0,30	$ 0,60
24	Gabinete cerrado	1	$ 30,00	$ 30,00
			TOTAL	$ 103,95

Tabla 9 . Detalle de costos para la implementación del dispositivo.

.

CAPÍTULO 4

4.1 Conclusiones y recomendaciones

4.1.1 Conclusiones

- Se pudo implementar un sistema que permitió al usuario obtener la información proveniente de una estación meteorológica utilizando la interfaz de comunicación RS-485. Esta información fue utilizada para poder visualizar los valores de producción teórica de energía eléctrica de la granja solar.
- Se pudo diseñar una aplicación donde el usuario puede observar a través de gráficas el comportamiento de su granja solar durante las horas y los días de producción. Además de tener la capacidad de almacenar información acerca de los datos reales de producción de energía y realizar las respectivas comparaciones para determinar el factor de rendimiento de su granja solar según su necesidades.
- La implementación de la base de datos para el almacenamiento de información es un nexo clave entre el dispositivo IoT y la aplicación, pues gracias a esta se pudo almacenar la información de manera sencilla permitiendo al usuario acceder a ella no solo localmente sino desde cualquier sitio siempre y cuando tenga la aplicación instalada y las credenciales de acceso al menú principal.
- Los datos de radiación solar en conjunto con información adicional como la cantidad de paneles y su rendimiento permitieron determinar la producción teórica de energía eléctrica y esta información a su vez permitió obtener el factor de rendimiento de la granja solar durante un día de producción.

4.1.2 Recomendaciones

- Para una mejor interpretación de la información se plantea la idea de implementar al sistema de monitoreo un aparatado en donde el usuario se comunique con los inversores de la granja solar para obtener las lecturas y almacenarlas en la base de datos de forma automática evitando tipearlas en la aplicación. Ofreciendo no solamente el producido final de su granja solar, sino también brindando detalles de producción durante cada hora de trabajo del sistema.
- Si bien es cierto que el sistema aporta información valiosa acerca del funcionamiento de la granja solar, se plantea el uso de un sistema adicional en donde se monitoreen la producción de 2 paneles a la vez pero en condiciones físicas distintas; es decir, un panel operando en condiciones normales y otro panel operando con un sistema automático de limpieza que garantice que el panel esté limpio y libre de polvo en todo momento para comparar la eficiencia de uno con el otro con la finalidad de definir si los intervalos de tiempo establecidos para la limpieza de los módulos fotovoltaicos son los adecuados o si acortar los intervalos de limpieza de estos se ven justificados con mayores niveles de producción de energía.

BIBLIOGRAFÍA

Banyeres, L. J. (2012). *Generacio de energia fotovoltaica.* Barcelona: Marcombo SA.

Barrero, F. (2004). *Sistemas de energia electerica.* Madrid : Parainfo SA.

Blanchette, J., & Mark Summerfield. (2008). *C++ GUI Programming with Qt 4.* Prentice Hall. Obtenido de https://tfetimes.com/wp-content/uploads/2015/09/c-gui-programming-with-qt-4-2ndedition.pdf

Bonaventure, O. (2011). *Computer Networking : Principles, Protocols and Practice.* The Saylor Foundation.

Bonilla Toribio, A. (s.f.). *MONITORIZACIÓN DEL ESTADO Y.* Obtenido de MONITORIZACIÓN DEL ESTADO Y: https://uvadoc.uva.es/bitstream/handle/10324/60693/TFG-I-2582.pdf?sequence=1&isAllowed=y

Energias Renovables. (2014). *Energias Renovables.* Obtenido de Energias Renovables: https://www.energiasrenovablesinfo.com/solar/tipos-paneles-fotovoltaicos/

espressif. (2024). *espressif*. Obtenido de espressif: https://docs.espressif.com/projects/esp-at/en/latest/esp32/AT_Command_Examples/TCP-IP_AT_Examples.html

Kalogirou, S. A. (2009). *Solar_Energy_Engineering-Processes_and_Systems.* Obtenido de Solar_Energy_Engineering-Processes_and_Systems: https://library.uniteddiversity.coop/Energy/Solar/Solar_Energy_Engineering-Processes_and_Systems.pdf

Llamuca Landa, Alex Bolívar, & Caisaguano Moreano, Alex Xavier. (2016). *https://docs.espressif.com/projects/esp-at/en/latest/esp32/AT_Command_Examples/TCP-IP_AT_Examples.html.* Obtenido de https://docs.espressif.com/projects/esp-at/en/latest/esp32/AT_Command_Examples/TCP-IP_AT_Examples.html: http://dspace.espoch.edu.ec/bitstream/123456789/6127/1/108T0182.pdf

López Garzón, W., & Cárdenas López, J. (2019). *Tecnología internet of things (IoT) y el big data.* Obtenido de https://doi.org/10.52948/mare.v1i1.183

Luis Rodriguez Ojeda. (2016). *Espol.* Obtenido de Espol: https://www.fcnm.espol.edu.ec/sites/fcnm.espol.edu.ec/files/PYTHON_PROGRAMACION_V2_3.pdf

MEJIA, A. E., TORRES, C., & RICARDO A. HINCAPIE ISAZA. (2010). *CONEXIÓN DE UN SISTEMA FOTOVOLTAICO A LA RED ELÉCTRICA.* (S. e. Technica, Ed.) Obtenido de CONEXIÓN DE UN SISTEMA FOTOVOLTAICO A LA RED ELÉCTRICA: https://dialnet.unirioja.es/descarga/articulo/4548810.pdf

Pep Puig, M. J. (2007). *fenercom.* Obtenido de fenercom: http://www.enginyeria-classea.cat/pdf-formativos/Cuaderno_FOTOVOLTAICA.pdf

Rafael Camps Paré, Luis Alberto Casillas Santillán, Dolors Costal Costa, Marc Gibert Ginestà, Carme Martín Escofet, & Oscar Pérez Mora. (2005). *Bases de datos.* Universitat Oberta de Catalunya. Obtenido de https://www.uoc.edu/pdf/masters/oficiales/img/913.pdf

Remus Teodorescu, Marco Liserre, & Pedro Rodr´ıguez. (2011). *Grid Converters for photovoltac and wind power systems.* John Wiley & Sons, Ltd.

Salazar, J. (s.f.). *Redes Inalambricas.*

SDK, F. A. (2023). *Firebase Admin Python SDK*. Obtenido de Firebase Admin Python SDK: https://firebase.google.com/docs/reference/admin/python

Shneiderman, B. (2004). *DESIGNING THE USER INTERFACE.* Pearson.

SMA Solar Technology AG. (s.f.). *Coeficiente de rendimiento.* Obtenido de Coeficiente de rendimiento: https://files.sma.de/downloads/Perfratio-TI-es-11.pdf

WEG. (2014). *WEG.* Obtenido de WEG: https://static.weg.net/medias/downloadcenter/hc8/hc6/WEG-srw01-manual-de-la-comunicacion-modbus-rtu-10000521680-6.0x-manual-espanol.pdf

WEG. (2019). *WEG.* Obtenido de WEG: https://static.weg.net/medias/h35/hc9/WEG-baterias-vrla-manual-del-usuario-10007225687-es.pdf

ANEXOS

Anexo 1. Tabla de consumo de la ESP32 por cada modo de operación

Modo ESP32	Consumo
“Deepsleep”	7 µA
“Lightsleep”	1 mA
Normal (240 MHz)	50 mA
Reloj del procesador reducido (3 MHz)	3,8 mA
Funcionamiento WiFi	80-180 mA

Fuente: https://www.radioshuttle.de/es/medias-es/informaciones-tecnicas/esp32-alimentado- por-bateria/#:~:text=Consumo%20de%20energ%C3%ADa%20del%20m%C3%B3dulo%20ESP32%20WROOM&text=Por%20lo%20general%2C%20las%20tarjetas,de%20dos%20mil%20veces%20m%C3%A1s!

Anexo 2. Tabla de características de los circuitos integrados MAX para comunicación RS485

PART NUMBER	HALF/FULL DUPLEX	DATA RATE (Mbps)	SLEW-RATE LIMITED	LOW-POWER SHUTDOWN	RECEIVER/ DRIVER ENABLE	QUIESCENT CURRENT (µA)	NUMBER OF TRANSMITTERS ON BUS	PIN COUNT
MAX481	Half	2.5	No	Yes	Yes	300	32	8
MAX483	Half	0.25	Yes	Yes	Yes	120	32	8
MAX485	Half	2.5	No	No	Yes	300	32	8
MAX487	Half	0.25	Yes	Yes	Yes	120	128	8
MAX488	Full	0.25	Yes	No	No	120	32	8
MAX489	Full	0.25	Yes	No	Yes	120	32	14
MAX490	Full	2.5	No	No	No	300	32	8
MAX491	Full	2.5	No	No	Yes	300	32	14
MAX1487	Half	2.5	No	No	Yes	230	128	8

Fuente: https://html.alldatasheet.com/html-pdf/73463/MAXIM/MAX485/125/1/MAX485.html

Anexo 3. Tabla de características eléctricas del circuito integrado LM358

ELECTRICAL CHARACTERISTICS (V_{CC} = 5.0 V, V_{EE} = Gnd, T_A = 25°C, unless otherwise noted.)

Characteristic	Symbol	LM258			LM358			LM2904			LM2904V			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
Input Offset Voltage V_{CC} = 5.0 V to 30 V (26 V for LM2904, V), V_{IC} = 0 V to V_{CC} −1.7 V, $V_O \approx$ 1.4 V, R_S = 0 Ω	V_{IO}													mV
T_A = 25°C		–	2.0	5.0	–	2.0	7.0	–	2.0	7.0	–	–	–	
T_A = T_{high} (Note 1)		–	–	7.0	–	–	9.0	–	–	10	–	–	13	
T_A = T_{low} (Note 1)		–	–	2.0	–	–	9.0	–	–	10	–	–	10	
Average Temperature Coefficient of Input Offset Voltage T_A = T_{high} to T_{low} (Note 1)	$\Delta V_{IO}/\Delta T$	–	7.0	–	–	7.0	–	–	7.0	–	–	7.0	–	∝V/°C
Input Offset Current	I_{IO}	–	3.0	30	–	5.0	50	–	5.0	50	–	5.0	50	nA
T_A = T_{high} to T_{low} (Note 1)		–	–	100	–	–	150	–	45	200	–	45	200	
Input Bias Current	I_{IB}	–	–45	–150	–	–45	–250	–	–45	–250	–	–45	–250	
T_A = T_{high} to T_{low} (Note 1)		–	–50	–300	–	–50	–500	–	–50	–500	–	–50	–500	
Average Temperature Coefficient of Input Offset Current T_A = T_{high} to T_{low} (Note 1)	$\Delta I_{IO}/\Delta T$	–	10	–	–	10	–	–	10	–	–	10	–	pA/°C
Input Common Mode Voltage Range (Note 2), V_{CC} = 30 V (26 V for LM2904, V)	V_{ICR}	0	–	28.3	0	–	28.3	0	–	24.3	0	–	24.3	V
V_{CC} = 30 V (26 V for LM2904, V), T_A = T_{high} to T_{low}		0	–	28	0	–	28	0	–	24	0	–	24	

Fuente: https://pdf1.alldatasheet.com/datasheet-pdf/view/3067/MOTOROLA/LM358.html

Anexo 4. Tabla de máximo valores de parámetros eléctricos del circuito integrado yf08e

6 Specifications

6.1 Absolute Maximum Ratings

over operating free-air temperature range (unless otherwise noted)(1)

		MIN	MAX	UNIT
Supply voltage, V_{CCA}		–0.5	4.6	V
Supply voltage, V_{CCB}		–0.5	6.5	V
Input voltage, V_I (2)	A port	–0.5	4.6	V
	B port	–0.5	6.5	
Voltage applied to any output in the high-impedance or power-off state, V_O (2)	A port	–0.5	4.6	V
	B port	–0.5	6.5	
Voltage applied to any output in the high or low state, V_O (2) (3)	A port	–0.5	V_{CCA} + 0.5	V
	B port	–0.5	V_{CCB} + 0.5	V
Input clamp current, I_{IK}	V_I < 0		–50	mA
Output clamp current, I_{OK}	V_O < 0		–50	mA
Continuous output current, I_O		–50	50	mA
Continuous current through V_{CCA}, V_{CCB}, or GND		–100	100	mA
Junction temperature, T_J			150	°C
Storage temperature, T_{stg}		–65	150	°C

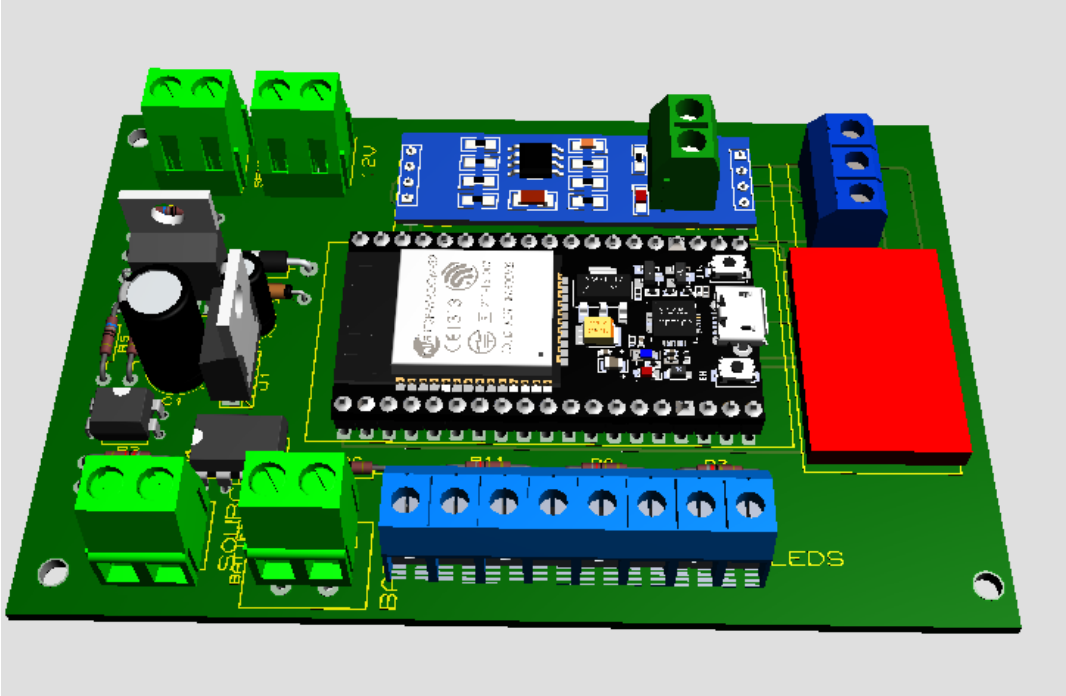
Fuente:

https://www.ti.com/lit/ds/symlink/txs0108e.pdf?ts=1705560130174&ref_url=https%253A%252F%252Fwww.google.com%252F

Anexo 5. Consumo eléctrico de sensores obtenido de la documentación ofrecida por el fabricante INGENIEURBURO

Si-V-10TC-T Irradiance, Cell Temperature	24 V_{DC} (12 to 28 V_{DC}) typic <1 mA	Yes	0 to 10 V for 0 to 1,500 W/m²	0 to 10 V for -40 to +90°C
Si-I-420TC Irradiance	24 V_{DC} (12 to 28 V_{DC}) typic 5 to 23 mA	Yes	4 to 20 mA for 0 to 1,500 W/m²	./.
Si-I-420TC-T Irradiance, Cell Temperature	24 V_{DC} (12 to 28 V_{DC}) typic 10 to 46 mA	Yes	4 to 20 mA for 0 to 1,500 W/m²	4 to 20 mA for -40 to +90°C
Si-RS485TC-T-MB Irradiance, Cell Temperature	24 V_{DC} (12 to 28 V_{DC}) typic 25 mA	Yes	MODBUS 0 to 1,500 W/m²	MODBUS -40 to +90°C
Si-RS485TC-2T-MB Irradiance, Cell Temperature, Ambient Temperature (sensor firmly connected with 3 m cable)	24 V_{DC} (12 to 28 V_{DC}) typic 25 mA	Yes	MODBUS 0 to 1,500 W/m²	MODBUS -40 to +90°C
Si-RS485TC-T-Tm-MB Irradiance, Cell Temperature, Module Temperature (sensor firmly connected with 3 m cable)	24 V_{DC} (12 to 28 V_{DC}) typic 25 mA	Yes	MODBUS 0 to 1,500 W/m²	MODBUS -40 to +90°C
Si-RS485TC-2T-v-MB Irradiance, Cell Temperature Accessories: External Temperature, Wind Speed	24 V_{DC} (12 bis 28 V_{DC}) typic 25 mA	Yes	MODBUS 0 to 1,500 W/m²	MODBUS -40 to +90°C

Anexo 6. Diseño 3D de la placa desarrollada para el dispositivo IoT



Anexo 7. función que se utilizó para enviar el comando a los sensores usan el protocolo modbus RTU en el programa de la ESP32

```
void modbus_sendData(uint8_t id, uint8_t fnc, uint16_t startReg, uint16_t noReg) {
  request[0] = id;
  request[1] = fnc;

  request[2] = (uint8_t)(startReg >> 8);
  request[3] = (uint8_t)(startReg & 0xFF);

  request[4] = (uint8_t)(noReg >> 8);
  request[5] = (uint8_t)(noReg & 0xFF);

  uint16_t crc = calculateCRC(request, 6);
  request[7] = (uint8_t)(crc >> 8);
  request[6] = (uint8_t)(crc & 0xFF);

  digitalWrite(RE, HIGH);
  digitalWrite(DE, HIGH);

  portRS485.write(request, sizeof(request));
  portRS485.flush();

  digitalWrite(RE, LOW);
  digitalWrite(DE, LOW);
}
```

Anexo 8. función para cargar un dato de tipo flotante en la base de datos firebase en el programa de la ESP32

```
bool loadDataFloatFirebase(String route, float data) {
  char routeDB[route.length() + 1];

  // Converts a string to a char array
  route.toCharArray(routeDB, sizeof(routeDB));

  // Send the float to the Firebase Database
  if(!Firebase.setFloat(fbdo, F(routeDB), data)){
    Serial.println(fbdo.errorReason());
    numErrorsFirebase++;
    return false;

  } else {
    num_iteration_request++;
    return true;

  }
}
```

Nota: se usó la misma estructura de la función para subir datos de tipo booleano, String, etc

Anexo 9. función que permite leer una variable String almacenada en la partición de memoria NVS del programa de la ESP32.

```
void nvsReadString(const char* add, char* buf) {
  NvsMemory.begin(NVS_MEMORY, true);
  String buf_str = NvsMemory.getString(add, "None");
  NvsMemory.end();

  int buf_str_lgth = buf_str.length();

  buf_str.toCharArray(buf, buf_str_lgth + 1);
}
```

Anexo 10. función que permite almacenar una variable String en la partición de memoria NVS del programa de la ESP32.

```
void nvsWriteString(const char* add, const String value) {
  int value_str_lgth = value.length();
  char buf[value_str_lgth];

  value.toCharArray(buf, value_str_lgth + 1);

  NvsMemory.begin(NVS_MEMORY, false);
  NvsMemory.putString(add, buf);
  NvsMemory.end();
}
```

Anexo 11. Función que permite almacenar una trama de data de las variables censadas en la partición SPIFFS de memoria de la ESP32

```
void spiffsWriteData(){
  File file = SPIFFS.open("/data_day.csv", "a");
  if (!file) {
    Serial.println("Error opening file for writing");
    return;
  }
  file.print(getHour());
  file.print(",");

  file.print(thereIsMainsPower);
  file.print(",");

  file.print(isConnectedWifi);
  file.print(",");

  file.print(responseReceived);
  file.print(",");

  if(responseReceived){
    file.print(variablesSampling[0]);
    file.print(",");

    file.print(variablesSampling[1]);
    file.print(",");

    file.print(variablesSampling[2]);
    file.print(",");

    file.println(variablesSampling[3]);

  } else{
    file.print("None");
    file.print(",");

    file.print("None");
    file.print(",");

    file.print("None");
    file.print(",");

    file.println("None");
  }
  file.close();
}
```

Anexo 12. Función que permite leer los datos de las variables censadas en la partición de memoria SPIFFS de la ESP32

```
void spiffsReadData(){
  File file = SPIFFS.open("/data_day.csv", "r");  // "r" indica abrir para lectura

  if (!file) {
    Serial.println("Error opening file for reading");
    return;
  }

  // Imprimir el contenido del archivo CSV en la consola serial
  while (file.available()) {
    Serial.write(file.read());
  }

  file.close();
}
```

Anexo 13. Función que permite limpiar la partición de memoria SPIFFS de la ESP32

```
void spiffsCleanData() {
  File file = SPIFFS.open("/data_day.csv", "w");  // Abre el archivo en modo escritura

  if (!file) {
    Serial.println("Error al abrir el archivo para escritura");
    return;
  }

  file.close();  // Cierra el archivo para finalizar la operación de limpieza
}
```

Anexo 14. Función de sincronización de reloj interno de la ESP32 con un servidor NTP en el internet

```
// Configuration and initialization of module NTP
void ntpInit() {
  // set notification call-back function
  //sntp_set_time_sync_notification_cb(timeavailable);

  sntp_servermode_dhcp(1);

  configTime(gmtOffset_sec, daylightOffset_sec, ntpServer1, ntpServer2);
  configTzTime(time_zone, ntpServer1, ntpServer2);

  printLocalTime();
}
```

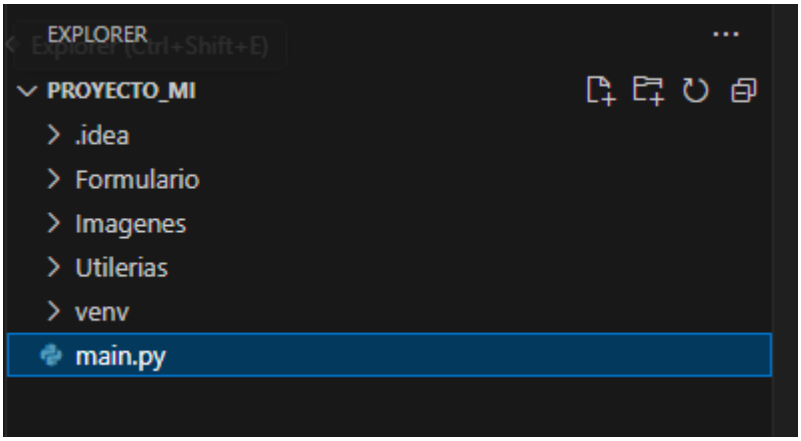
Anexo 15. función de configuración e inicialización de base de datos Firebase en la ESP32

```
void firebaseInit() {
  // Indicate the Firebase version
  //Serial.printf("Firebase Client v%s\n\n", FIREBASE_CLIENT_VERSION);

  // Assign the api key (required)
  configFire.api_key = API_KEY;

  // Assign the user sign in credentials
  auth.user.email = userEmail;
  auth.user.password = userPass;

  // Assign the RTDB URL (required)
  configFire.database_url = DATABASE_URL;

  // Assign the callback function for the long running token generation task
  configFire.token_status_callback = tokenStatusCallback;  // see addons/TokenHelper.h

  fbdo.setBSSLBufferSize(4096 /* Rx buffer size in bytes from 512 - 16384 /, 1024 / Tx buffer size in byt

  // Limit the size of response payload to be collected in FirebaseData
  fbdo.setResponseSize(2048);

  Firebase.begin(&configFire, &auth);

  // Comment or pass false value when WiFi reconnection will control by your code or third party library
  Firebase.reconnectWiFi(true);

  Firebase.setDoubleDigits(5);
}
```

Anexo 16. función de muestro y carga de datos de los sensores en la ESP32

```
// Sampling the sensors data
if (isLocalTimeSync() && currentMillisSampling - previousMillisSampling >= intervalSampling*samplingTime) {

  thereIsMainsPower = digitalRead(PIN_MAINS_POWER);
  battery = analogRead(pinBattery);

  //Serial.printf("Battery: %d \n", battery);

  requestDataSensors();

  if (WiFi.status() == WL_CONNECTED){
    isConnectedWifi = true;
  } else{
    isConnectedWifi = false;
  }

  spiffsWriteData();

  if (WiFi.status() == WL_CONNECTED && Firebase.ready()){
    if(loadDataFirebaseRTDB()){
      //Serial.println("Datos Cargados a firebase");
    }
  }

  previousMillisSampling = currentMillisSampling;
}

unsigned long currentMillisSync = millis();
```

Anexo 17. Estructura del proyecto de la Aplicación GUI



Anexo 18. Codigo que perimte configurar un marco que contiene la etiqueta de texto

```
##########################################################################################################################
# Frame_From_Top - Definimos el espacio para el titulo de inicio de sesion
frame_form_top = tk.Frame(frame_form, height=100, bd=0, relief=tk.SOLID, bg='black')
frame_form_top.pack(side="top", fill=tk.X)
title = tk.Label(frame_form_top, text="Inicio de Sesión", font=('tahoma', 40), fg='#000000', bg='#F3F9FF', pady=50)
title.pack(expand=tk.YES, fill=tk.BOTH)
```

Anexo 19. Codigo que permite configurar un marco que contiene un campo para ingresar texto

```
##########################################################################################################################
# Usuario - Comenzamos a definir las etiquetas y los campos de llenado para la inforamcion de usuario
etiqueta_usuario = tk.Label(frame_form_fill, text="Usuario", font=('tahoma', 30), fg='#000000', bg='#FFFFFF', anchor="n")
etiqueta_usuario.pack(fill=tk.X, padx=20, pady=5)
self.usuario = ttk.Entry(frame_form_fill, font=('Times', 20))
self.usuario.pack(fill=tk.X, padx=20, pady=10)
```

Anexo 20. Codigo que inicialeza la ventana con la configuración definida en etapas anteriores

```
self.ventana.mainloop()
```

Anexo 21. Código que permite solicitar datos en la plataforma de Firebase

```
##############NECESITO VALIDAR LA INFORMACION PARA QUE LA INFORMACION SE MUESTRE POR MES
datos = firebase.FirebaseApplication('https://materia-integradora-3f5c9-default-rtdb.firebaseio.com/', None)
```

Anexo 22. Código que permite calcular el factor de rendimiento promedio según las necesidades del usuario

```
##########################################################################################################
# Seleccion de rango de horas útiles para llevar el registro
ValPromediar = []  # Creo una lista para almacenar los valores que se encuentran en el rango de hora de
for i in range(len(horaRad)):
    hora = horaRad[i]
    if int(hora[:2]) >= 6 and int(hora[:2]) < 19:
        b = ValorRadInt[i]
        RadiacionTotal250 = (b * AreaSolar250) / 1000
        RadiacionTotal265 = (b * AreaSolar265) / 1000
        RendimientoNominalFot = (RadiacionTotal250 * (RendPan250)) + (RadiacionTotal265 * (RenPan265))
        ValPromediar.append(RendimientoNominalFot)
    else:
        pass
PromediosDiasTe.append(statistics.mean(ValPromediar))
EtiquetValorTe.append(str(int(statistics.mean(ValPromediar))))
```

Anexo 23. Función que configura e inicializa la ventana que contiene la gráfica de barras de la variable medida de radiación.

```
elif len(PromediosMeses) > 0:
    self.ventana = tk.Tk()
    self.ventana.title('Menú de información')
    self.ventana.iconbitmap('Imagenes/Panel.ico')
    self.ventana.geometry('800x500')
    self.ventana.config(bg='#F3F9FF')
    self.ventana.resizable(width=0, height=0)
    utl.centrar_ventana(self.ventana, 950, 380)  # Medidas de ventana de inicio de sesión

    logo = utl.leer_imagen("./Imagenes/Panel.ico", (100, 100))

    frame = tk.Frame(self.ventana, bg="blue")
    frame.grid(column=0, row=0, sticky="nsew")

    fig, ax = plt.subplots(1, 1, dpi=75, figsize=(13, 5), sharey=True, facecolor="#F3F9FF")
    fig.suptitle("Promedios mensuales de producción teórica obtenidos durante el " + AnoG)
    plt.xlabel('Tiempo - Días de producción')
    plt.ylabel('KWh', rotation=90)

    # plt.show()
    barras =ax.bar(NombreMeses, PromediosMeses, color='b')

    #ax[1].bar(Valor, PromediosMeses, color='b')

    canvas = FigureCanvasTkAgg(fig, master=frame)
    canvas.draw()
    canvas.get_tk_widget().grid(column=0, row=0, rowspan=3)

    ###################################################
    for barra in barras:
        altura = barra.get_height()
        ax.text(barra.get_x() + barra.get_width() / 2, altura,
                f'{altura:.2f}', ha='center', va='bottom')
    ###################################################

    self.ventana.mainloop()
```

Anexo 24. Función para iniciar la comunicación serial con el dispositivo IoT en la aplicación GUI

```
# OnClick event of connect serial port button
def connect_com():
    global ser, isConnected, port_selected
    try:
        # Intentar abrir el puerto serie
        ser = serial.Serial(port_selected,9600)
        print(f"Puerto serie {port_selected} inicializado correctamente.")
        isConnected = True
        update_ui_on_connect()

    except serial.SerialException as e:
        # Capturar y manejar excepciones en caso de error
        print(f"No se pudo inicializar el puerto serie {port_selected}. Error: {e}")
        show_warning_com_no_available()
```

Anexo 25. Función definida para leer datos del puerto serial en la aplicación GUI

```
# Function for reading the serial port
def serial_reader():
    global isConnected, list_ports_currentely
    while True:
        if isConnected:
            try:
                response = ser.readline().decode().strip()
                if response:
                    handle_response(response)
            except Exception as e:
                print(f"Error al leer el puerto serial: {e}")
                show_warning_error_connection()
                update_ui_on_disconnect()
                isConnected = False

        if not are_arrays_equal(list_ports_currentely,get_ports_com()):
            list_ports_currentely = get_ports_com()

            spinner_ser['values'] = list_ports_currentely

            if list_ports_currentely:
                spinner_ser.set(list_ports_currentely[0])
```

Anexo 26. Función para enviar un comando especifico al dispositivo IoT

```
# Send command by serial communication
def write_com(command):
    global ser
    try:
        if ser and ser.is_open:
            ser.write(command.encode())  # Envía el comando como bytes
            print(f"Comando enviado correctamente: {command}")
        else:
            print("No hay un puerto serie activo para enviar el comando.")
            show_warning_no_serial()
    except Exception as e:
        print(f"Error al enviar el comando. Error: {e}")
```

Anexo 27. Función para finalizar la comunicación serial con el dispositivo IoT en la aplicación GUI

```python
# OnClick event of disconnect serial port button
def disconnect_com():
    global port_selected, ser, isConnected

    if ser:
        try:
            ser.close()
            print(f"Puerto serie {port_selected} desconectado correctamente.")
            isConnected = False
            update_ui_on_disconnect()

        except Exception as e:
            print(f"Error al desconectar el puerto serie {port_selected}. Error: {e}")
    else:
        print("No hay un puerto serie activo para desconectar.")
```

Ilustración 77. Pruebas de trasmisión del dispositivo instalado en planta.

Ilustración 78. Módulos fotovoltaicos instalados en la granja solar.

Ilustración 79. Acceso principal a los módulos fotovoltaicos.

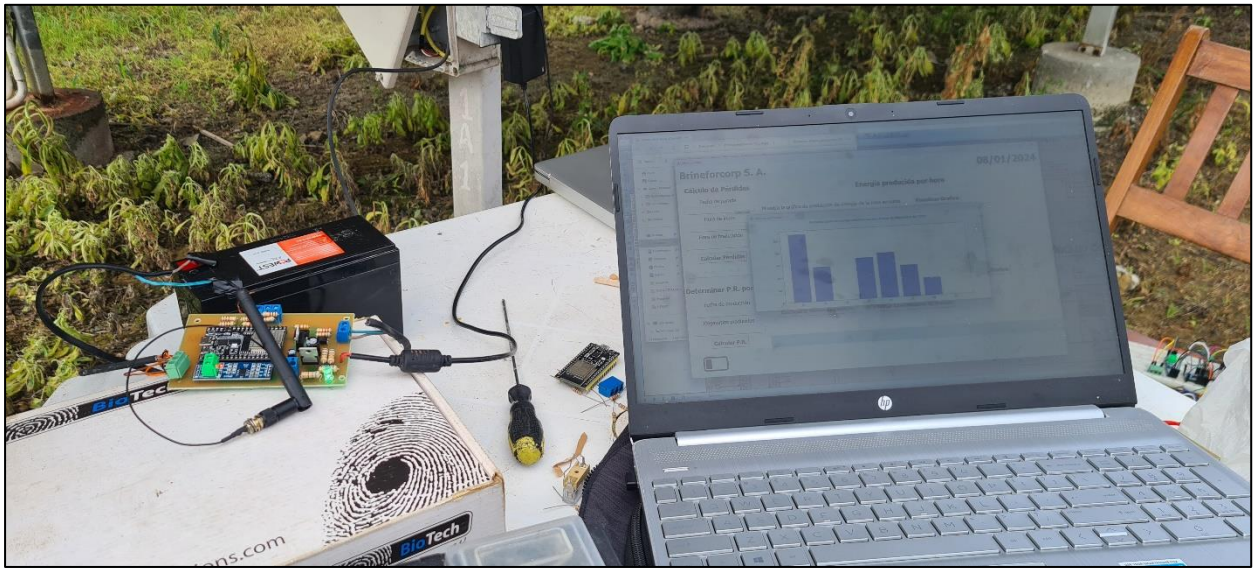


Ilustración 80. Configuración y visualización de datos en línea cargados por el dispositivo.