



SECRETARIA DE EDUCACIÓN PÚBLICA
TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE MÉRIDA

ITM

**“MONITOREO DE PARÁMETROS ELÉCTRICOS Y
RADIACIÓN SOLAR DE UN SISTEMA FOTOVOLTAICO
VIRTUAL CON IOT”**

PARA OPTAR AL GRADO DE

MAESTRO EN INGENIERÍA

PRESENTA

ABIGAIL IRIDIANA VICTORIA GRANADOS

ASESOR

Dr. Carlos Alberto Luján Ramírez

MÉRIDA, YUCATÁN, MÉXICO

2021



Educación
Secretaría de Educación Pública



TECNOLÓGICO
NACIONAL DE MÉXICO



Instituto Tecnológico de Mérida
División de Estudios de Posgrado e Investigación

DEPENDENCIA: **DIV. DE EST. DE POSG. E INV.**

No. DE OFICIO: X-0125/26

Mérida, Yucatán, **17/abril/2026**

ASUNTO: AUTORIZACIÓN DE IMPRESIÓN.

**C. ABIGAIL IRIDIANA VICTORIA GRANADOS
PASANTE DE LA MAESTRÍA EN INGENIERÍA
PRESENTE.**

De acuerdo al fallo emitido por su director Carlos Alberto Luján Ramírez, su coasesor Jesús Sandoval Gio y la comisión revisora integrada por, José Ramón Atoche Enseñat, Gustavo Alonso Martínez Escalante y Alejandro Arturo Castillo Atoche considerando que cubre los requisitos establecidos en el Reglamento de Titulación de los Institutos Tecnológicos le autorizamos la impresión de su trabajo profesional con la TESIS:

**"MONITOREO DE PARÁMETROS ELÉCTRICOS Y RADIACIÓN SOLAR DE UN SISTEMA FOTOVOLTAICO
VIRTUAL CON IOT."**

ATENTAMENTE

Excelencia en Educación Tecnológica®

**DR. GABRIEL LIZAMA UC
JEFE DE LA DIVISIÓN DE ESTUDIOS DE
POSGRADO E INVESTIGACIÓN**

C.C.p. Archivo
GLU/SCCC/zac



2026
año de
**Margarita
Maza**



Contenido

Agradecimientos	5
Dedicatorias	6
Resumen	7
Capítulo 1.	8
Presentación.....	8
1.1 Introducción.....	8
1.2 Planteamiento del problema	9
1.3 Objetivos.....	9
1.3.1 General.....	9
1.3.2 Específico.	9
1.4 Justificación	9
1.5 Estado del arte	10
2. Capítulo 2. Marco teórico	12
2.1 Energía Fotovoltaica.....	13
2.2 Radiación solar	13
2.2.1 Irradiancia.....	13
2.2.2 Irradiación.....	13
2.2.3 Masa de Aire.....	13
2.3 Geometría solar.....	14
2.4 Módulo fotovoltaico	14
2.4.1 Tensión de salida	14
2.4.2 Potencia de salida	14
2.4.3 Curva I-V	15
2.5 Inversor interconectado a la red.....	15
2.6 Sensores y Actuadores.....	15
2.7Piranometro	15
2.8 Microcontroladores.....	15
2.9 Datalogger	16
2.10 Software de simulación	16
2.11 Proteus	16
2.12 Visual Studio	16

2.13 Python.....	16
2.14 Arduino.....	17
2.15 Nube	17
2.16 Ubidots	17
Capítulo 3. Metodología.....	18
3.1 Características eléctricas de los paneles Solares.	18
3.2 Sensores para el datalogger	18
3.2.1 Sensor de corriente	18
3.2.2 Sensor de Voltaje.....	18
3.2.3 Sensor de Temperatura	19
3.2.3 Sensor de Irradiancia	19
3.3 Sistema Embebido	19
3.4 Programación y prueba de lectura de sensores de temperatura	19
3.5 Transmisión de datos de Proteus a Python	20
3.6 Programa completo del Datalogger en Arduino y Proteus	20
3.7 Configuración de Ubidots.....	21
3.8 Configuración en el programa de Python para Ubidots.	21
Capítulo 4. Pruebas y resultados	22
Figura 18 <i>Interfaz de sensores de Temperatura y voltaje</i>	22
Figura 19 <i>Interfaz de sensores de Voltaje y corriente</i>	23
Figura 20 <i>Ventana de error de ejecución</i>	24
Conclusiones.....	25
Referencias bibliográficas	27
3. Referencias	27
4. Anexo A (Figura)	30
Figura 1 <i>Entrada del Instituto Tecnológico de Mérida</i>	30
Figura 2. <i>Tablilla Literaria excavada en Nínive</i>	30
Figura 3. <i>Libro de los muertos</i>	31
Figura 4 <i>Sadie Pfeifer, la niña icónica</i>	31
Figura 5 <i>Paneles Solares</i>	32
Figura 7 <i>Sistema de monitoreo</i>	33
Figura 8 <i>ACS712</i>	33

Figura 9	<i>Diagrama ACS712 30A.....</i>	34
Figura 10	<i>Diagrama ADC1015</i>	34
Figura 11	<i>Sensor DS18B20</i>	34
Figura 12	<i>Diagrama del DS18B20</i>	35
Figura 13	<i>Circuito de sensores DS18B20</i>	35
Figura 14	<i>Adquisición de direcciones de los sensores de temperatura</i>	36
Figura 15.	<i>Simulación de lectura de sensores de temperatura.....</i>	36
Figura 16	<i>Conexión del Sistema FV al Arduino</i>	36
Figura 17	<i>Interfaz gráfica para variables contables.....</i>	38
5.	Anexo B (Tablas)	39
Tabla 1	<i>Datos del panel solar S72PC-315</i>	39
Tabla 2	<i>Sensores para el sistema Fotovoltaico</i>	39
Tabla 3	<i>Características del ACS712</i>	39
Tabla 4	<i>Circuito Divisor de Voltaje</i>	40
Tabla 5	<i>Sensor DS18B20.....</i>	40
Tabla 6	<i>Sensor de irradiancia</i>	40
6.	Anexo C (Códigos).....	42
Código 1	<i>Lectura de direcciones del DS1</i>	42
Código 2	<i>Lectura de sensores de temperatura</i>	43
Código 3	<i>Comunicación y lectura de puerto COM</i>	46
Código 4	<i>Lectura de datos en Python</i>	48
Código 5	<i>Programa completo de Arduino para el Datalogger</i>	49
Código 6	<i>Tokens de Ubidots.....</i>	54
Código 7	<i>Identificación de variables de Arduino y envío de datos a Ubidots.....</i>	55

Agradecimientos

Al Instituto Tecnológico de Mérida por permitirme sumarme a su programa de Maestría en ingeniería donde tuve la oportunidad de conocer increíbles compañeros de otras áreas y a maestros y doctores expertos en su cátedra. Por dejar las puertas abiertas para ingresar a los laboratorios y aulas antes de los problemas que surgieron por la pandemia, y de las medidas de seguridad de salud que nos llevó resguardarnos en nuestras casas, pero siempre en contacto en línea, viendo nuevas opciones de comunicación para seguir con el programa y proyecto.

Al Conacyt con su programa de apoyo económico con sus becas que me permitió dejar un tiempo mis actividades laborales y concentrarme en la maestría, permitiendo enfocarme en las asignaturas y al proyecto.

A mis tutores Doctor Carlos Luján y Jesús Sandoval Gío quienes estuvieron presentes y pendientes de mi progreso con el proyecto para el monitoreo de paneles, el cual presentó cambios de la disposición física del prototipo a una virtual debido a la pandemia.

A mi padre Raúl Enrique Victoria Delgado por motivarme a iniciar la maestría mientras luchaba de forma silenciosa contra el cáncer y con las esperanzas de seguir con vida un tiempo más pero que lamentablemente tuvo que partir de este mundo y a mi madre

Auridice Granados Sánchez por el apoyo para brindarme paz a mi alma mientras seguíamos adelante afrontando nuestros problemas.

Dedicatorias

Dedico esta tesis para todos los lectores que estén buscando una orientación para su proyecto, para que conozcan los diversos softwares que uno puede contar para las pruebas virtuales antes de pasar al estado físico de los dispositivos y conozcan las limitaciones de los programas libres con que pueden contar un estudiante en sus prácticas.

Resumen

La tesis presenta el desarrollo del proyecto del monitoreo de un sistema fotovoltaico a través de un equipo de adquisición de datos con conexión a Internet para aplicar el sistema de monitoreo en una IoT (Internet de las cosas) mediante el uso de una de las plataformas que implementan el almacenaje de datos en la nube y una aplicación o página web que visualice los datos de monitoreo. El sistema fotovoltaico y los sensores son simulados mediante los programas Proteus y Arduino, logrando la comunicación de una plataforma IoT con Python para vincular el sistema simulado.

Capítulo 1.

Presentación

1.1 Introducción

Los sistemas fotovoltaicos para la generación de energía eléctrica han tenido un auge en la última década, derivado de diversas disposiciones establecidas en el Plan Nacional de Desarrollo 2007-2012 (Virgen, 2007) , en la Ley para el Aprovechamiento de Energías Renovables y el Financiamiento de la Transición Energética, su Reglamento, así como en el Programa Especial de Cambio Climático 2008-2012. De acuerdo con el último Estudio Intercensal realizado por INEGI, en 2015 había casi 32 millones de viviendas particulares habitadas en México y sólo el 0.5 por ciento aprovechaba el potencial del sol con paneles solares en ese entonces (INEGI, 2015). Entre los principales estados para la inversión de proyectos de energía solar están Sonora, Chihuahua, Nuevo León y Yucatán; lo cual ha motivado que más personas de la sociedad civil, empresarios e instituciones inviertan en sistemas fotovoltaicos para la generación de energía eléctrica orientado a cubrir el consumo residencial o para disminuir los costos de consumo en la red eléctrica. Estos puntos llevo al Instituto Tecnológico de Mérida (ITM) invertir en la instalación de un sistema fotovoltaico con diez paneles.

Un sistema de monitoreo se vuelve una parte esencial para los usuarios para conocer el estado del sistema y su comportamiento en el tiempo para implementar acciones de mantenimiento, estudios, accionamiento de actuadores, mejora en la calidad, entre otros, siendo el intercambio de información la parte más importante. Hoy en día, la implementación de internet de las cosas (IoT) ha facilitado la comunicación de los dispositivos del hogar y la industria, logrando ser la transferencia de información alámbrica o inalámbrica, la información llega a una base de datos local o en la nube y pueda ser visualizada a través de una aplicación o software que permite al usuario visualizar los datos de forma comprensible o interactuar con los objetos al enviar comandos para su activación o desactivación desde cualquier punto demográfico que le permita tener acceso a la aplicación. El proyecto de monitoreo del sistema fotovoltaico se basa en el uso de internet, específicamente el uso de WiFi como principal conexión inalámbrica con los paneles solares ubicados en el techo del segundo piso de la institución a través de un datalogger encargado de enviar los datos de los sensores a una plataforma especializada en IoT, debido a la falta de accesibilidad a la institución por la pandemia se implementa una versión virtual del prototipo.

El software de monitoreo del sistema de paneles de la empresa Solartec instalados en el ITM utiliza el sistema operativo Windows XP, lo cual limita la integración a los nuevos equipos computacionales del instituto que están basados en la versión de Windows 10, además

1.2 Planteamiento del problema

Las instalaciones del Instituto Tecnológico de Mérida (ver Figura 1, Anexo A) cuentan con un sistema fotovoltaico, el cual se requiere un nuevo sistema de monitoreo del voltaje, corriente y potencia generado por los paneles solares, ya que el actual requiere de Windows XP y no es compatible con el sistema operativo de las computadoras actuales de la institución. También no hay una buena visualización e interacción con la interfaz del usuario que permita una rápida toma de decisiones, además, se requiere llevar a cabo la medición del nivel de irradiancia y temperatura en que se encuentra sometido cada panel fotovoltaico las 24 horas del día y que se pueda tener acceso a la información en cualquier punto donde se encuentre el supervisor en la instalación para eliminar la falta de información y dependencia de un punto centrado de supervisión.

La pandemia del Covid 19 que se extendió a nivel global en el año de 2020 pero sus inicios se dieron en diciembre 2019, las instituciones educativas junto a otras empresas privadas y públicas de diferente giro se cerraron al paso público, por siguiente se propuso realizar el datalogger de forma virtual para su simulación.

1.3 Objetivos

1.3.1 General

Mediante la creación de un datalogger virtual para la recepción y almacenamiento de las lecturas generadas por los sensores para la medición de parámetros eléctricos que corresponden a la corriente, voltaje y potencia; y la temperatura e irradiancia para los parámetros meteorológicos, llevar a cabo el monitoreo de un sistema fotovoltaico virtual para la conexión y pruebas en la nube a través de una plataforma especializada en IoT que permita comprobar la funcionalidad del proyecto de forma simulada.

1.3.2 Específico.

- a. Crear un datalogger virtual con las entradas necesarias para cubrir la cantidad de sensores necesarios por panel solar y la adquisición de datos de forma alámbrica.
- b. Generar la transmisión de datos por Wi-fi desde el datalogger virtual a la plataforma IoT seleccionada.
- c. Estructurar las variables de la plataforma IoT para representar los datos en tiempo real con gráficas donde el usuario pueda visualizar, interpretar y valorar el comportamiento del sistema fotovoltaico desde una unidad móvil o fija.

1.4 Justificación

En general, un pequeño sistema fotovoltaico autónomo está limitado por factores de tamaño y costo, y es difícil rastrear su funcionamiento principalmente porque los registradores de datos comerciales son excesivamente caros en comparación con el precio total del sistema

fotovoltaico, requieren software especial y generalmente requieren ser continuamente conectado a una PC (MICHEL, 2015) . Por lo tanto, se requiere un mayor desarrollo de dataloggers precisos, autónomos y de bajo costo para monitorear pequeños sistemas fotovoltaicos independientes que permita obtener información relacionada al rendimiento y la producción de energía que tiene los paneles solares a lo largo de año para que el usuario pueda tomar decisiones para intervenir en el mantenimiento del sistema fotovoltaico desde su computador o en cualquier sistema móvil como el celular o tablet, librándose de las limitaciones de un sistema fijo como la PC al disponer de la información en la palma de su mano a través del internet. Debido a los costos del material, falta de acceso de la institución por pandemia se procede a reproducir el prototipo de forma virtual para una futura reproducción física.

1.5 Estado del arte

La necesidad del hombre de registrar diversas actividades se ha reflejado a lo largo de la historia, desde la implementación de logográficas o pictográficas en civilizaciones antiguas como Mesopotamia(ver Figura 2, Anexo A) y Egipto (ver Figura 3, Anexo A) aproximadamente un siglo antes del año 3000 A.C. (RA, 2010) hasta llegar a la escritura actual a pesar que existen una gran diversidad de idiomas, esto también ha estado a la par de cuantificar lo que vemos, poseemos y percibimos, incluso en cosas no perceptibles al ojo humano. Las mediciones en los tiempos antiguos eran manuales y su margen de error era afectada por error humano. Con el tiempo el campo científico permite crear instrumentos que permitan medir variables físicas con el uso de propiedades de materiales sensibles al entorno a medir y la aplicación de la electricidad generaron con el tiempo los primeros sensores y transductores que permitan la creación de instrumentos de medición complejos.

En el siglo XIX, cuando llega la era de la industrialización, toma el auge registrar los cambios y controlar procesos a lo largo del tiempo en las maquinarias para que puedan trabajar en óptimas condiciones y no alcanzar un estado crítico que pueda afectar de forma negativa a la maquina o al producto (ver Figura 4, Anexo A). La visualización de los parámetros de los equipos de medición era de estación fija, aun percibidas por el hombre, pero las mediciones ya eran realizadas por un equipo de medición especial. El surgimiento del primer transistor dio origen a la electrónica de los semiconductores, generando el primer microcontrolador que daría el origen a las computadoras y sumergirnos a la era digital (Gómez Gómez, 2007).

Para un sistema de control o monitoreo, los datos compilados por los sensores deben pasar por un sistema de adquisición de datos, comúnmente conocido como DAQ que convierte señales eléctricas analógicas de fenómenos físicos en el dominio digital y luego utiliza una computadora para la visualización, almacenamiento y análisis de datos, bajo el control del software asociado (Grant Maloy, 2024). Dentro de este grupo se encuentra los dataloggers que tienen velocidades de muestreo más lentas y están destinados a registrar datos que cambian lentamente durante un largo período de tiempo: días, semanas, meses o incluso años. Además, estos registradores de datos suelen ser mucho más pequeños, menos costosos y altamente portátiles (Grant Maloy, 2024). Por último, se encuentran los instrumentos

virtuales en los sistemas de control de medida que simula sistemas basados en sistemas de adquisición de datos y software como LabView, VEE, entre otros (Mercedes Granda & Mediavilla Bolado, 2015).

El “Instituto Tecnológico de Mérida” es una institución pública de educación superior, actualmente cuenta con dos campus, una ubicada en el norte de la ciudad de Mérida y la otra en el poniente, en el estado de Yucatán, México. El enfoque del proyecto a desarrollar se ubica en el edificio AB del campus Norte, donde ya se cuenta con 10 paneles, cada panel genera una potencia de 315W con una tensión de 37.3V y 8.44A máximo de acuerdo con las especificaciones del proveedor Solartec.

Solartec S.A. es una empresa de capital mexicana fundada en el 2009. Su campo de actividad cubre la fabricación de paneles solares, así como inversores, sistema de monitoreo y medidores bidireccionales (Apogee , 2017). Realizó compra de plantas y negocios en Europa, en 2014 adquiere la empresa francesa Tournaire Solaire Energie para reubicarla en Houston, Texas; también adquirió la alemana Bosch Solar, la cual ahora opera bajo el nombre de Semicon, y a la belga Photovolttech, mismas que llevó al Parque Industrial Apolo en Guanajuato (Manufactura , 2019). Actualmente no se tiene registro de la empresa y su página oficial se encuentra inactiva.

La introducción del internet a las cosas o IoT ha ocasionado el constante monitoreo y control de los dispositivos mediante la interconexión digital de objetos cotidiano. El concepto IoT se ha manejado actualmente para el monitorio de los paneles solares con el fin de mejorar el sistema fotovoltaico, el mantenimiento y evaluar el sistema (MICHEL, 2015). Dependiendo de la perspectiva del usuario los beneficios son los siguientes:

- Fabricantes de componentes: las evaluaciones de rendimiento y calidad son puntos de referencia de calidad para sus productos.
- Equipos de investigación y desarrollo: son un método clave para ayudar a un diseño óptimo.
- Integradores de sistemas y los clientes finales: son herramientas vitales para evaluar productos y la calidad de los productos para guiar la toma de decisiones futuras (SOLARTEC, 2019).

En los proyectos que se han realizado el monitoreo de sistema fotovoltaicos han buscado que los sistemas de transferencia de datos sean de largo alcance para zonas remotas y a bajo costo, en algunos casos, para la medición de variables meteorológicas se han optado instalar una estación meteorológica que es alimentada por la fuente fotovoltaica. Para el actual proyecto, no se requiere una estación meteorológica debido a que en nuestra región existe una en la UADY, UTM y CINVESTAV, también porque se busca enfocarse en la simulación de la temperatura y lúmenes.

Para proyectos de adquisición de datos se ha empleado el dispositivo ArduinoTM Uno, este microcontrolador trabaja con un código abierto y gratuito donde el desarrollador le resulta agradable pero presentó limitaciones de hardware, como la baja resolución de las mediciones

o la ausencia de un reloj, por siguiente, se requirió la integración de nuevas mejoras de hardware en una PCB ad-hoc (placa de circuito impreso) para resolver las principales limitaciones (KAWAMOTO, NISHIYAMA, & KATO, 2014).

Otro dispositivo implementado en adquisición de datos es el controlador de automatización programable (PAC) (SOLARTEC, 2019) Compact RIO de National Instruments, que son controladores embebidos en tiempo real e ideales para aplicaciones avanzadas de monitoreo y control, y cuentan con puertos seriales, Ethernet, USB y pueden operar en temperaturas que van desde 40 °C hasta 70 °C. Debido a su complejidad y a la familia que pertenece al equipo, la precisión y el costo es elevado, este equipo se empleó para el proyecto de una planta solar de 150KVA (11).

El presente proyecto de adquisición de datos se desarrolla en un entorno virtual usando el programa Proteus donde se trabaja con componentes ya previamente seleccionados físicamente que puedan trabajar con las características del sistema de paneles de la institución. Proteus es un programa versátil que cuenta con muchos proyectos de práctica simples y complejos (Agbetuyi Ayoade , Orovwode , Olowoleni , & Nwangwu , 2018).

En el área de transmisión de información se ha empleado en varios proyectos el uso de dispositivos que permiten la conectividad a internet para que el datalogger tenga acceso a la CPU o computadora, como es el ESPO1 WIFI para Arduino, el CC2530 Zigbee que tiene una comunicación de radiofrecuencia, empleada en el área de domótica y el sistema SCADA para grandes escalas; para después guardar los datos en una nube (Zou, Li, Zhai, & Liu, 2012). El programa Proteus no cuenta con un módulo de simulación de conexión a internet, pero cuenta con conexión de puertos COM para hacer la interfaz de comunicación con programa externo como Python que cuenta con librería para conectar directamente con algunas de las plataformas de nube. En el mercado se cuenta varias plataformas de IoT que tienen una versión gratuita y de costo como Google Cloud Platform, Pariciple, IBM y Amazon AWS IoT Core, su ventaja es que es fácil de comprender y más rápido de aplicar que hacer el uso de SQL para almacenar datos.

Para finalizar, el programa en común para la HMI (Interfaz Hombre Maquina) en los proyectos de monitoreo se encuentra el programa LabView, gracias a sus interfaces gráficas simples, muestran la información del proceso en tiempo real y de forma intuitiva. El presente proyecto cubre la plataforma de la nube y la HMI con Ubidots que se describe en el siguiente capítulo.

2. Capítulo 2. Marco teórico

El capítulo desarrolla los principales conceptos necesarios para profundizar en el tema de las características principales que intervienen en un panel solar y la producción de energía de este, mediante la comprensión de los conceptos permite tener idea de los componentes necesarios para implementar en el Datalogger conocido en hispanohablante como Registrador de Datos, concepto que también se habla en este capítulo.

2.1 Energía Fotovoltaica.

El efecto fotoeléctrico permite transformar directamente energía solar (ya sea directa o difusa) en energía eléctrica continua. Para la creación de las celdas solares se suelen utilizar materiales semiconductores, en especial el silicio (el segundo o más abundante en la corteza terrestre que se obtiene de la arena). El elemento base de la energía fotovoltaica es la célula fotovoltaica, suelen ser de silicio mono cristalino, policristalino o amorfo. Los conjuntos de células fotovoltaicas se llaman paneles solares o fotovoltaicas (ver Figura 5, Anexo A) y se orientan hacia el sur para aprovechar más la radiación solar, y son conectadas a un sistema de almacenamiento (baterías) y de conversión de la corriente. Se trata pues de una fuente de energía que puede aprovecharse en cualquier aplicación: red eléctrica, consumo en lugares aislados en zonas rurales.

2.2 Radiación solar

La radiación solar es el conjunto de radiaciones electromagnéticas emitidas por el sol. El sol es una estrella que se encuentra a una temperatura media de 6000 K en cuyo lugar tiene una serie de reacciones de fusión nuclear, que produce una pérdida de masa que se transforma en energía. (Arenas Sánchez & Zapata Castaño, 2011) Esta energía es liberada por el sol y es transmitida hacia el exterior por medio de radiación solar.

La luz generada por el sol o por cualquier fuente de luz está formada por un conjunto de radiaciones electromagnéticas de muy alta frecuencia que tienen un cierto rango y se le llama espectro luminoso.

Las ondas de frecuencias bajas del espectro solar llamadas ondas infrarrojas son las que proporcionan el calor y las de altas frecuencias son las ultravioletas que nos proporcionan la fotosíntesis y el bronceado de la piel. Esos 2 extremos forman parte del espectro solar. La variación de la longitud de onda nos indica la variación de la intensidad luminosa.

2.2.1 Irradiancia

La irradiancia es la magnitud utilizada para describir la potencia incidente por unidad de superficie de todo tipo de radiación electromagnética, también se denomina como constante solar. La magnitud representa la cantidad promedio de energía que incide por unidad de área por unidad de tiempo sobre una superficie. Las unidades en que se mide son las unidades radiométricas y es en W/m^2 (Lira Oliver & Guavara Mon, 2017).

2.2.2 Irradiación

Para evitar confusión con el término irradiancia, la irradiación es la energía por unidad de área (J/m^2 o Wh/m^2), la comparación de los términos se puede apreciar en el Figura 6, Anexo B.

2.2.3 Masa de Aire

Se define la masa de aire (AM) como el cociente entre el recorrido óptico de un rayo solar y el correspondiente a la normal a la superficie terrestre (ángulo cenital cero) y que está relacionada con la altura solar (α). Esta medida se utiliza como referencia para comparar la

eficiencia de los paneles solares en diferentes condiciones atmosféricas. La masa de aire tiene un impacto significativo en la eficiencia de los paneles solares. A medida que la luz solar atraviesa una mayor cantidad de atmósfera, se produce una mayor dispersión y absorción de la radiación solar. Esto reduce la cantidad de energía que llega a los paneles solares y, por lo tanto, disminuye su eficiencia (Vega de Kuyper, 2018).

2.3 Geometría solar

La geometría solar es uno de los elementos más importantes del proceso de diseño de una instalación fotovoltaica ya que a través del conocimiento de la trayectoria de los rayos solares, tanto en su componente térmica como lumínica, se logrará dar la óptima orientación al arreglo fotovoltaico, para aprovechar al máximo el recurso solar disponible del lugar.

2.4 Módulo fotovoltaico

Los módulos fotovoltaicos están formados por un conjunto de celdas (células fotovoltaicas) que producen electricidad a partir de la luz que incide sobre ellos (energía solar fotovoltaica). El parámetro estandarizado para clasificar su potencia se denomina potencia pico, y se corresponde con la potencia máxima que el módulo puede entregar bajo unas condiciones estandarizadas.

Los paneles fotovoltaicos, en función del tipo de célula que los forman, se dividen en:

- Cristalinas
- Mono cristalinas: se componen de secciones de un único cristal de silicio (Si) (reconocibles por su forma circular u octogonal, donde los 4 lados cortos, si se puede apreciar en la imagen, se aprecia que son curvos, debido a que es una célula circular recortada).
- Poli-cristalinas: cuando están formadas por pequeñas partículas cristalizadas.
- Amorfas: cuando el silicio no se ha cristalizado.

2.4.1 Tensión de salida

La tensión de salida de una célula fotovoltaica es de corriente continua. Por lo tanto, hay un lado que es positivo y otro negativo. Esta variable está directamente relacionada con la temperatura que tenga la celda, existe una pérdida de tensión para temperaturas altas y la variación dependerá del factor de pérdidas por temperatura de cada tipo de tecnología.

2.4.2 Potencia de salida

En un instante determinado, la potencia eléctrica proporcionada por el módulo fotovoltaico está dada por el producto de los valores instantáneos de la tensión y la corriente de salida. Este valor es afectado por el comportamiento intrínseco de un material semiconductor, por el nivel de irradiación luminosa, y el método de fabricación de la célula.

2.4.3 Curva I-V

Si los valores de potencia luminosa y orientación del panel permanecen constantes, las corrientes de salida de un panel fotovoltaico varía con el valor de tensión de la carga y su temperatura de trabajo. Esto se debe a la característica intrínseca de los materiales semiconductores.

2.5 Inversor interconectado a la red

Un inversor de interconexión a red es un inversor que convierte la corriente directa (DC) de electricidad en corriente alterna (AC) con una capacidad de sincronizar para interactuar con una red eléctrica local (Teón Vega, 2013).

2.6 Sensores y Actuadores

Los sensores son dispositivos que permiten cuantificar una variable física, por otra parte los actuadores son dispositivos que tienen la capacidad de realizar una acción en consecuencia del resultado de la medición de la variable en cuestión. Los sensores y actuadores están basados en el principio de transducción (principio de transformación de energía), en cual en general se le denomina transductor a todo dispositivo que convierte una variable física en otra correspondiente a un dominio diferente, es decir que los transductores convierten un tipo de energía en otro y por tanto los transductores forman parte de los sensores o actuadores, razón por la cual suelen usarse como sinónimos.

2.7 Piranometro

Los piranómetros son dispositivos de medición de la radiación solar, que generalmente se utilizan para investigaciones climatológicas o para monitorear el desempeño de estaciones meteorológicas en redes meteorológicas, invernaderos, automatización de edificios, estudios de reflectancia, meteorización, etc (Seven, 2022).

2.8 Microcontroladores

Un microcontrolador es un dispositivo electrónico que contiene un microprocesador, memoria, interfaces de entrada y dispositivos periféricos como los temporizadores para almacenar programas y datos con el fin de permitirle un modo de comunicación con el exterior desde donde pueda recibir señales. Dichas señales servirán de insumo para que funcione como un sistema aplicado al control. En general los microcontroladores tienen líneas o terminales para alimentación eléctrica, conexión de entradas y salidas (puertos), señales de reloj, señales de control.

Los PIC de alta gama incluyen en su configuración un módulo de comunicación serial que les permite utilizar dos modos de transmisión: el SSP (Puerto Serie Síncrono) y el USART (Universal Synchronous Asynchronous Receiver Transmitter).

En el modo SSP, el módulo de transmisión serie cuenta con dos interfaces de trabajo, la SPI (Interfaz Serial de Periféricos), la cual fue desarrollada por Motorola para consumo de sus mismos productos, y la I2C (Interfaz Inter-Circuitos), desarrollada por Philips y de uso más extendido entre otros fabricantes.

Los tipos de comunicación serial menos comunes que utilizan los PIC son: el USB (Universal Serial Bus), CAN (Controller Area Network), Ethernet, LIN (Local Interconnect Network) y 1-Wire bus (Ortega, 2019).

2.9 Datalogger

Un registrador de datos (datalogger) es un dispositivo electrónico que registra datos en el tiempo o en relación con la ubicación por medio de instrumentos y sensores propios o conectados externamente. Casi todos están basados en microcontroladores. Por lo general son pequeños, con pilas, portátiles, y equipados con un microprocesador, memoria interna para almacenamiento de datos y sensores. Algunos registradores de datos se comunican con un ordenador personal y utilizan software específico para activar el registrador de datos, ver y analizar los datos recogidos, mientras que otros tienen un dispositivo de interfaz local (teclado, pantalla LCD) y puede ser utilizado como un dispositivo independiente (Final Test, s.f.).

2.10 Software de simulación

Un software de simulación de sistemas sirve para conocer cuál será el comportamiento de un sistema en determinadas condiciones del mundo real, sin necesidad de que estas se produzcan de facto. Es decir, se trata de un software que “simula” una situación plausible en el mundo físico, permitiendo así comprobar cómo funciona el sistema en estos supuestos (Click, 2021).

2.11 Proteus

Proteus es un sistema completo de diseño electrónico que combina un avanzado programa de captura de esquemas, un sistema de simulación mixto (analógico y digital) basado en Spice, y un programa para disposición de componentes en placas de circuito impreso y auto-ruteado. Se trata de un software comercial fabricado por Labcenter Electronics, caracterizado por su potencia y facilidad de uso (Barrón Ruiz).

2.12 Visual Studio

Visual Studio es un entorno de desarrollo integrado (IDE) desarrollado por Microsoft que permite a los programadores crear aplicaciones para distintas plataformas (incluyendo Windows, Android, iOS y Linux). Este software ofrece diversas herramientas para facilitar el proceso de desarrollo, como depuración de código, integración con sistemas de control de versiones, pruebas automatizadas, y asistencia en la escritura de código (ARIMETRICS, 2022).

2.13 Python

Python es un lenguaje de programación ampliamente utilizado en las aplicaciones web, el desarrollo de software, la ciencia de datos y el machine learning (ML). Los desarrolladores utilizan Python porque es eficiente y fácil de aprender, además de que se puede ejecutar en muchas plataformas diferentes. El software Python se puede descargar gratis, se integra bien a todos los tipos de sistemas y aumenta la velocidad del desarrollo. (AWS, s.f.)

2.14 Arduino

Arduino es una plataforma de creación de electrónica de código abierto, la cual está basada en hardware y software libre, flexible y fácil de utilizar para los creadores y desarrolladores. Esta plataforma permite crear diferentes tipos de microordenadores de una sola placa a los que la comunidad de creadores puede darles diferentes tipos de uso (Fernández, 2022).

2.15 Nube

La nube es una red de servidores distribuidos por todo el mundo que están operativos 24 horas al día los 365 días del año y que funcionan como un único ecosistema para que podamos almacenar todo tipo de datos, programas y plataformas informáticas. De esta forma, no tienes que almacenar toda esta información, software o sistemas en tu disco duro, sino que al estar alojada en esta red conectada de servidores, siempre tienes acceso a la información desde cualquier lugar o dispositivo (Alonso, 2023).

2.16 Ubidots

Ubidots es una Plataforma habilitadora del Internet de las Cosas que permite a miles de ingenieros, desarrolladores y empresas, desplegar soluciones IoT de manera rápida y sin necesidad de contratar un equipo de desarrollo de software o tardar demasiado tiempo en desplegar soluciones a sus clientes finales (Procolombia, s.f.).

Capítulo 3. Metodología

3.1 Características eléctricas de los paneles Solares.

Los principales parámetros de medición del sistema fotovoltaico son el voltaje, corriente, temperatura y la irradiancia del ambiente, siendo los dos primeros los más relevantes para determinar la potencia total del panel solar.

Un panel solar de Solartec del modelo S72PC-315 genera 8.44 amperes en condiciones normales, en su tabla de datos (ver Tabla 1, Anexo B) el máximo voltaje se da cuando está en circuito abierto con 44.37 Volts, estos valores son tomados de referencia para la selección de los sensores físicos de cada panel solar que formaran parte del sistema adquisición de datos, siendo un total de 10 paneles solares que contiene el sistema fotovoltaico del instituto. En la Tabla 1 se puede observar las siguientes abreviaciones:

- STC: Standard Test Condition. Mediciones obtenidas bajo condiciones de pruebas estándar con una irradiancia de 1,000W/m, temperatura de 25°C, AM 1.5.
- NOTC: Normal Operation Cell Temperature. Mediciones realizadas con una irradiancia de 800W/m, temperatura 45 ±2°C, AM 1.5, considerado como un estado no ideal y más cercano a condiciones reales.

3.2 Sensores para el datalogger

Para el monitoreo de los parámetros eléctricos de los paneles eléctricos, se divide la adquisición de datos en 2 dataloggers (ver figura 7, anexo A). Cada datalogger recopilará datos de 5 paneles y el sensor de irradiancia debe estar conectado a uno de los dataloggers, en la tabla 2 del Anexo B se observa un total de 31 sensores que se requieren para el sistema fotovoltaico. A continuación, se especifica el tipo de sensor seleccionado para su representación física y que posteriormente se simula en el programa Proteus.

3.2.1 Sensor de corriente

Para medir la corriente generada por el panel solar se selecciona el módulo ACS712 (ver figura 8, anexo A) que permite medir hasta 20 amperes con corriente continua y corriente alterna, eso nos dejar un gran margen de trabajo en caso de que se presente alguna inconsistencia en el panel solar y se presente algún pico de corriente al alza. Este sensor usa el efecto Hall y su voltaje de trabajo es de 5Volts (ver tabla 3, anexo B). Este dispositivo cuenta con una biblioteca para su simulación en el programa de Proteus, en caso de contar con él en el programa de Proteus se puede adquirir la biblioteca en internet de forma libre para su instalación (ver Figura 9, anexo A).

3.2.2 Sensor de Voltaje

La mayoría de los sensores para medir el voltaje tiene su aplicación para corriente alterna, otra opción en el mercado es el sensor FZ0430, que es un módulo que puede medir hasta 25 V DC, pero el voltaje que se necesita medir debe ser aproximadamente los 45 Volts, lo cual no hace viable para el proyecto. Para cubrir ese margen grande voltaje se opta por emplear el circuito divisor de voltaje para medir el voltaje de corriente continua del panel solar. La selección de componentes se observa en la tabla 4. Para medir el voltaje de CD de los paneles solares se genera con una señal analógica, el sistema embebido debe contar con 5

entradas analógicas, esto puede aumentar el costo, para poder acoplar las señales analógicas se selecciona el dispositivo ADC1115, que cuenta con comunicación I2C y permite la lectura de 4 señales analógicas, en la tabla 4 se puede observar en la columna de “Diagrama” el componente ADC115, el cual, también se cuenta con su librería en Proteus (ver Figura 10, anexo A) con el nombre ADC1015.

3.2.3 Sensor de Temperatura

La medición de la temperatura se realiza con el sensor DS18B20 (ver Figura 11, anexo A) que cuenta con su suficiente margen para la medición de temperatura ambiente. Los sensores DS18B20 están disponibles para la simulación en Proteus, la ventaja de estos sensores es que cuentan con un protocolo de comunicación llamado One-wire (ver tabla 5, anexo B) que permite conectar más de un sensor en el mismo bus de datos mediante una dirección única que se le da en Proteus (ver figura 12, anexo A).

3.2.3 Sensor de Irradiancia

Para medir la irradiancia se selecciona el equipo digital BGT-SR1, la desventaja es que solo se encontraba en mercado asiático a un precio accesible pero llevaría mínimo tres meses su entrega hasta 6 meses (ver Tabla.2. Sensores para paneles solares), el piranómetro cuenta con una fuente de alimentación de 5, 12 y 24 V, según se requiera. Este equipo no cuenta con librerías en Proteus para su simulación, pero existe una biblioteca, pero se toma un LDR para su representación en la simulación en Proteus.

3.3 Sistema Embebido

Como sistema embebido se selecciona el WeMos D1 Mini ya que cuenta con conexión WiFi-integrada y es compacta, lo que lo hace ideal para su implementación física. En la simulación virtual de Proteus no hay biblioteca para el el WeMos D1 Mini, por siguiente, se opta implementar el Arduino Uno que comparte características similares al equipo seleccionado. El Arduino uno virtual es la base del sistema de adquisición de datos de los paneles solares.

3.4 Programación y prueba de lectura de sensores de temperatura

Primero se implementa el desarrollo del programa para medir las variables de temperatura de 5 sensores conectados a Arduino Uno simulado en el programa Proteus, cada sensor de temperatura cuenta con una dirección para la conexión Onewire, para esto, es necesario descargar unas librerías libres que tiene el Arduino para la comunicación OneWire y configuración para la lectura de la temperatura. Las bibliotecas que implementar son las siguientes:

- OneWire.h
- DallasTemperature.h
- SoftwareSerial

Para detectar la dirección de los sensores de temperatura es necesario realizar un programa en Arduino para que en el monitor serial de Proteus obtenga su dirección. El programa en Arduino para leer las direcciones de las señales de temperatura virtual se puede apreciar en

el Anexo C, código 1. En la figura 13 se aprecia el circuito de conexión para los sensores de temperatura en Proteus para las pruebas de simulación de conexión One-wire e impresión de datos. En Proteus, en el simulador de Arduino se carga la dirección del código 1 y se reproduce la simulación para visualizar las direcciones que le asigna el programa Proteus a cada sensor de temperatura, en la figura 14 del Anexo A se aprecia las direcciones de cada sensor de temperatura.

Una vez se obtiene las direcciones de los sensores de temperatura, en el programa de Arduino se configura la lectura para 5 sensores de temperatura, donde cada variable se le asigna la dirección obtenida del programa anterior, el programa de lectura de sensores se puede observar en el código 2, Anexo C. El código 2 se carga en el Arduino de Proteus para observar que se ejecuta de forma correcta la lectura de los sensores. En la figura 15 del anexo A se puede observar el resultado de la simulación.

En el código 3 del Anexo C, es el programa de lectura de temperatura de Arduino, pero se agrega otra terminal comunicación serial, esto es para asignar un puerto virtual desde el programa de Proteus. En el programa de Proteus la terminal COM se configura con el nombre por puerto COM3, un nombre que no está asignado de forma física a otro puerto.

3.5 Transmisión de datos de Proteus a Python

Llevado a cabo el proceso de tomar lecturas de los sensores de temperatura con el Arduino de Proteus, se procede a comenzar las pruebas de comunicación entre el programa Proteus y Python usando a Visual Studio para ejecutar Python.

El enlace de comunicación entre Python y Proteus se realiza por medio de un software llamado Configure Virtual Serial Port Driver, este programa permite generar dos puertos virtuales que tendrán enlace de comunicación, donde uno de los puertos se le asigna el nombre “COM3” para emplearlo en el programa de Proteus y el otro puerto “COM5” se asigna a Python.

En Python se genera un programa sencillo para comprobar que se recibe la información y revisar la estructura que se recibe los datos desde el puerto COM de Proteus (código 4, Anexo C) durante la simulación. El programa que se ejecuta en el Arduino de Proteus es el mismo de temperatura del código 3.

3.6 Programa completo del Datalogger en Arduino y Proteus

En el programa de adquisición de datos de Arduino, se agrega la librería Adafruit_ADS1x15.H para programar los componentes ADS1015 de Proteus y poder asignar cada pin de entrada analógica las señales de voltaje y corriente. Los ADS usan la comunicación 12C y se le pueden asignar hasta cuatro direcciones, es decir, se pueden conectar sin problema 4 ADS al Arduino, las direcciones empleadas en el programa para los ADS son “0x48”, “0x4A” y “0x4B”, y un pin analógico del Arduino se emplea para leer el LDR, sus valores se representan en forma de porcentaje (ver código 5, Anexo C).

En Proteus se agregan los 5 paneles solares que van conectados a un equipo de adquisición de datos (Arduino) como se aprecia en la figura 16. Los paneles que proporciona Proteus

funcionan como baterías donde se puede asignar el voltaje que se requiera, en este caso siempre se respetara un máximo de 45 Volts. Para la configuración de los ADS en Proteus, en propiedades del bloque ADS se agrega la dirección empleada en el programa de Arduino para cada elemento. Una vez que se termine de configurar los programas en el Arduino y en Proteus se procede hacer la prueba de simulación para verificar que todos los elementos muestren sus mediciones en tiempo real.

3.7 Configuración de Ubidots

Ubidots es una plataforma enfocada a proyectos de IoT que proporciona almacenamiento en la nube, cuenta con 4 versiones, 3 en modalidad de pago que corresponda al profesional, industrial y empresarial y la parte gratuita que recibe el nombre de Ubidots STEM, esta versión permite adquirir 2 dispositivos, cada dispositivo cuenta con su dashboards y en cada dispositivo solo se acepta 10 variables. Un dispositivo en Ubidots es una unidad que representa el proyecto general, en él se encuentra la dirección para vincular el dispositivo que cuenta con internet con el prototipo que enviara la información en tiempo real, también cada variable creada en Ubidots tiene su propia dirección para recibir la información y es necesaria para el programa de Python. Ya generada la cuenta en Ubidots, en la sección de Dispositivos, se generan 2 dispositivos, uno integrara las variables de temperatura y voltaje y el otro las variables de corriente y el sensor LDR. En cada dispositivo de Ubidots en el parte superior derecho cuenta con un signo de suma que es para agregar variables, una vez creada la nueva variable, se puede cambiar el nombre y el rango valores. Cuando se terminen de crear todas las variables, uno puede ir al Dashboard para ir generando cada icono que representara a cada variable (ver figura 17, Anexo A), Ubidots clasifica a cada icono de acuerdo a sus funciones, como se puede ser de control, mapa, displays, entre otros; para el proyecto se emplea una interfaz de un termómetro para las temperaturas, una pila para el voltaje y un gráfico de nivel para la corriente.

3.8 Configuración en el programa de Python para Ubidots.

En Python se requiere agregar la librería de Ubidots para lograr usar la interfaz de programación de aplicaciones (API), el primer segmento trata probar si se tiene éxito con la conexión API de Ubidots, para ello en Ubidots se copia el token del proyecto para que puede identificar a que dispositivo se debe conectar, seguido se prueba la conexión de cada variable de los sensores con sus respectivos tokens correspondientes a cada variable de Ubidots, en el código 6 del Anexo C, se puede observar la estructura que debe seguir la asignación de la variable con el token de Ubidots.

La parte más importante del programa radica en lograr identificar y clasificar cada variable que se están leyendo en Python para luego redireccionarla y ser enviada a Ubidots, para ello, primero generamos una lista con la función Index para guardar la lectura en una variable y luego se divide la cadena de datos que se recibe en Python mediante las siguientes funciones: `[pos]` y `[pos+1:]`, el primero tiene como función almacenar o guardar el fragmento de la cadena que esta antes de los dos puntos (:) porque en Arduino se generó la impresión de datos con el nombre de una variable corto y después de los dos puntos (:) se asigna el valor real de la lectura, en Python se preguntara si la variable leída es idéntica a la variable que se conoce, en caso de ser cierto, con el comando `[pos+1:]` indicamos que se

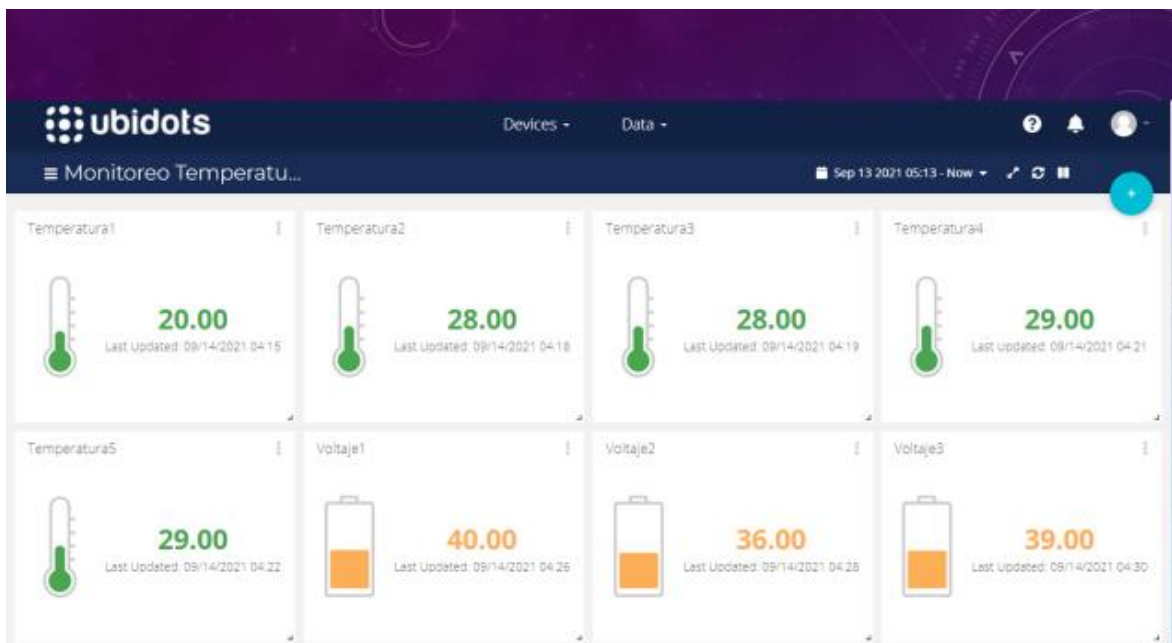
guarde todo lo está después de los dos puntos en una variable que guardara el valor en Ubidots (ver código 7, Anexo C), y así para todas las variables que se requieren del proyecto, en caso de no identificar algo, volverá a preguntar hasta que logre identificar la variable.

Capítulo 4. Pruebas y resultados

En este capítulo se presenta la simulación y registro de datos de los sensores de temperatura, corriente y voltaje desde la simulación del circuito en Proteus, transfiriendo los datos con el puerto COM con Python, ejecutando el programa con Visual Studio, donde se conecta con internet mediante a librería que hay Ubidots en Python. En las siguientes imágenes se puede apreciar la interfaz que brinda Ubidots.

Figura 18

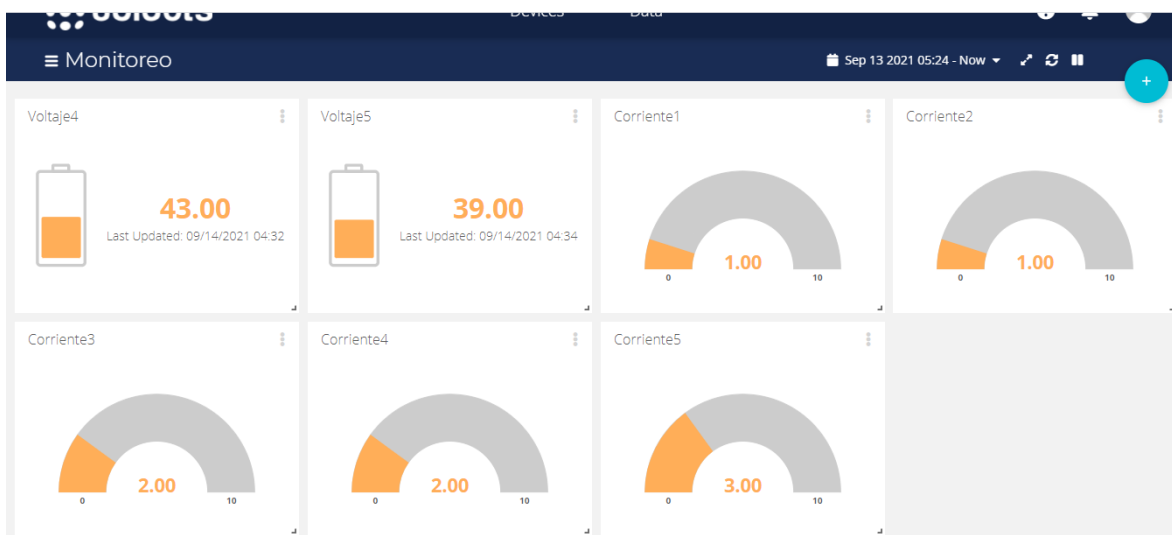
Interfaz de sensores de Temperatura y voltaje



Nota: Visualización de medición de temperatura y voltaje desde Ubidots de la simulación de adquisición de datos del circuito de Proteus.

Figura 19

Interfaz de sensores de Voltaje y corriente

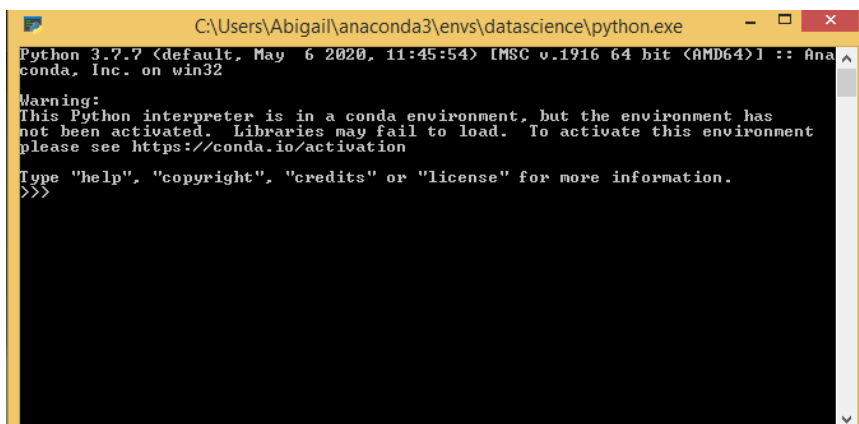


Nota: Visualización de la interfaz de los sensores de voltaje y corriente en Ubidots.

Durante el desarrollo del programa para seguir simulando, el programa de Python en Visual Studio se desconfigura las bibliotecas y el software de ejecución de Python, esto se debió a la intervención de otro programa que anteriormente se contaba en la computadora llamada “Anaconda”, en la figura 20 se tiene una imagen de la ventana que se generó del error al ejecutar el programa, después de detectar que el programa “Anaconda” es el programa que está afectando a Visual Studio se procede a eliminar el programa, pero seguía apareciendo error en la ejecución, por lo cual, se procedió a desinstalar el programa de Visual Studio y Python para identificar la causa o por lo menos lograr una buena configuración para su conexión.

Figura 20

Ventana de error de ejecución

A screenshot of a Windows terminal window. The title bar at the top reads 'C:\Users\Abigail\anaconda3\envs\datascience\python.exe'. The terminal content shows the Python 3.7.7 prompt, followed by a warning message: 'Warning: This Python interpreter is in a conda environment, but the environment has not been activated. Libraries may fail to load. To activate this environment please see https://conda.io/activation'. Below the warning, it says 'Type "help", "copyright", "credits" or "license" for more information.' and ends with the prompt '>>>'.

```
C:\Users\Abigail\anaconda3\envs\datascience\python.exe
Python 3.7.7 (default, May 6 2020, 11:45:54) [MSC v.1916 64 bit (AMD64)] :: Ana
conda, Inc. on win32

Warning:
This Python interpreter is in a conda environment, but the environment has
not been activated. Libraries may fail to load. To activate this environment
please see https://conda.io/activation

Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Nota: Ventana de terminal de Windows que informa de problemas para cargar las librerías de Python

El siguiente problema que surge es la actualización de librerías, donde se presenta problemas de identificar los datos que envía el Arduino virtual a Python, donde anteriormente se seccionaba la cadena de datos que se envía desde Arduino Virtual a Python, pero en la conversión del segmento de datos que son números, en unidades o en float seguía presentando error.

Conclusiones

El diseño virtual del sistema de adquisición de datos de los paneles solares se logra diseñar de forma virtual mediante los programas Visual Studio, Python, Isis Proteus, Arduino y la plataforma de Ubidots. El uso del Arduino Uno virtual de Proteus se debe a que se tiene una gran gama de librerías y el WeMos D1 Mini comparte el mismo software de programación, en caso haber desarrollado el prototipo de forma física, también se podría agregar la librería de Ubidots directo al programa de Arduino porque el We Mos D1 Mini si cuenta con WiFi integrado. En Proteus no había ningún componente que integrara el WiFi en la tarjeta y solo comparte las características de entrada y salida de las tarjetas de programación. Entre las pruebas que se iban realizando en los programas para lograr establecer la comunicación entre Proteus y Python, se optó siempre comenzar con la lectura de una cadena pequeña para comprobar y ubicar más rápido los errores que se presentara al ejecutar el programa en Python, cuando se empieza implementar la lectura de más de 2 variables se presentó el problema de como identificar cada variable y como relacionarla con su valor, por lo cual, se investigó diversos proyectos que implementara el Python y su enfoque fuera la extracción de datos específicos, algunos mapeaban los datos es una matriz y otros es una lista, siendo la lista una versión sencilla y más fácil de comprender, también se requiere ubicar que es importante convertir la cadena en un valor flotante al manejar valores decimales en la mediciones de los sensores. El uso de Ubidots en Python permitió crear una interfaz en línea que permitiera acceder a la información en cualquier punto que cuente con internet. Se permitió agregar sin problemas las mediciones de los sensores de corriente y voltaje y temperatura, estos valores se registran con la fecha y hora que se recibe los datos en Ubidots. A pesar que se requería agregar el cálculo de potencia en la visualización de variables del dashboard, al no permitir más de 10 se optó por no generarlo, ya que estaría incompleto la representación de las variables, aunque de ser requerido más adelante se puede optar por la modalidad de paga para poder agregar más widgets en el dashboear de Ubidots. El programa de Proteus a pesar de ser un programa versátil presento problemas para la generación de mediciones con las direcciones internas de los sensores de temperatura que se conectaban al OneWire, causando que no se imprima los valores de todos los sensores después transcurrir el tiempo de espera, a veces se presentaba tres valores y volvía a comenzar desde inicio el valor del primer sensor. Entre más grande era el programa que se cargaba al Arduino de Proteus más lenta era la generación de lecturas. de usar la representación del primer bloque de paneles que van conectado al primer dispositivo de monitoreo en el programa Proteus mediante el uso de bibliotecas de dispositivos libres en internet que están programadas para interactuar como lo haría un dispositivo físico, También para evitar una sobresaturación de procesos en la computadora y en la tarjeta de video al ejecutarla simulación en Proteus, se optó por solo representar y simular la mitad de los paneles que lleva el sistema fotovoltaico de la institución.

Para complementar este proyecto para una simulación adecuada sería averiguar cómo ejecutar todos los sensores del proyecto en Proteus para que cambien su valor con el tiempo sin la intervención manual del usuario. Hay que recalcar que la mayor dificultad de este proyecto fue no encontrar documentos o videos que permitieran la orientación en la

identificación de múltiples variables de Arduino a Python, habían ejemplos de solo un sensor y usando otros programas, o solamente la conexión desde el Arduino Uno con librerías a Ubidots, muchos proyectos de IoT están implementados directamente con el uso de dispositivos físicos y no virtuales.

Referencias bibliográficas

3. Referencias

- Barrón Ruiz, M. (s.f.). USO DIDÁCTICO DEL SOFTWARE DE AYUDA AL DISEÑO.
- Agbetuyi Ayoade , F., Orovwode , H., Olowoleni , J., & Nwangwu , C. (2018). Design and Implementation of a Solar Energy Measurement.
- Alonso, M. (7 de Febrero de 2023). *Asana*. (Asana) Recuperado el 29 de Febrero de 2024, de Qué es la nube y cómo puede ayudarte en la gestión de proyectos: <https://asana.com/es/resources/what-is-the-cloud>
- Apogee . (2017). *Soluciones energía renovable*. (Apogee) Recuperado el 4 de Mayo de 2020, de <http://apogeego.com/web/index.php/soluciones/soluciones-energia-renovable>
- Arenas Sánchez, D. A., & Zapata Castaño, H. S. (2011). *LIBRO INTERACTIVO SOBRE ENERGIA SOLAR Y SUS APLICACIONES*. Pereira: UNIVERSIDAD TECNOLÓGICA DE PEREIRA.
- ARIMETRICS. (2022). *Qué es Visual Studio*. (ARIMETRICS) Recuperado el 12 de Mayo de 2023, de Qué es Visual Studio: <https://www.arimetrics.com/glosario-digital/visual-studio>
- AWS. (s.f.). *¿Qué es Python?* (AWS) Recuperado el 12 de Mayo de 2023, de ¿Qué es Python?: <https://aws.amazon.com/es/what-is/python/>
- Bravo, G. (Diciembre de 2020). *Profesor de fotografía*. Obtenido de Lewis Hine: <https://fotogasteiz.com/blog/fotografos/lewis-hine-biografia-vida-obra/>
- Click, O. (23 de Noviembre de 2021). *Indielec*. Recuperado el 12 de Mayo de 2023, de ¿Para qué sirve un software de simulación?: <https://www.indielec.com/para-que-sirve-un-software-de-simulacion-blog-4-50-291/>
- Cuneiform Digital Library Initiative (CDLI). . (9 de Septiembre de 2023). *Cuneiform Digital Library Initiative (CDLI)*. . Obtenido de Gilgamesh II 124ff, J (P273210): <https://cdli.ucla.edu/P273210>
- Fernández, Y. (23 de Septiembre de 2022). (Xataka Basics) Recuperado el 15 de Mayo de 2023, de Qué es Arduino, cómo funciona y qué puedes hacer con uno: <https://www.xataka.com/basics/que-arduino-como-funciona-que-puedes-hacer-uno>
- Final Test*. (s.f.). Recuperado el 12 de Mayo de 2023, de ¿Que es un Datalogger?: <https://www.finaltest.com.mx/product-p/art-4.htm>
- Gómez Gómez, M. (2007). *Electrónica General*. Madrid: RA-MA.
- Grant Maloy, S. (5 de Febrero de 2024). *Historia de la adquisición de datos: desde registradores gráficos de tira hasta DAQ's digitales*. Recuperado el 29 de 02 de

2024, de DEWESoft: <https://dewesoft.com/es/blog/historia-de-la-adquisicion-de-datos>

HelioEsfera. (10 de Octubre de 2020). *Irradiancia, irradiación y radiación solar*.

(HelioEsfera) Recuperado el 15 de Octubre de 2021, de

<https://www.helioesfera.com/irradiancia-irradiacion-y-radiacion-solar/#>

INEGI. (2015). Porcentaje de viviendas que disponen de panel solar para tener electricidad (Porcentaje). Cd.de Méxio : INEGI.

KAWAMOTO, Y., NISHIYAMA, H., & KATO, N. (2014). Internet of Things (IoT): Present State and Future Prospects. *The Institute of Electronics, Information and Communication Engineers*, E97–D(10), 8.

Lira Oliver, D., & Guavara Mon, A. (2017). IRRADIANCIA Y RADIANCIA. México: LES.

Manufactura . (10 de Agosto de 2019). *Desde México, Solartec se abre paso por el mundo*.

(Manufactura) Recuperado el 4 de Mayo de 2020, de

<https://manufactura.mx/industria/2016/08/10/desde-mexico-solartec-se-abre-paso-por-el-mundo>

Mercedes Granda , M., & Mediavilla Bolado, E. (2015). *Instrumentación electrónica: transductores y actuadores de señal*. Cantabria: Universidad de Cantabria.

Miatello, R. M., & Salomón, S. E. (2010). EL TERMÓMETRO: HISTORIA DE UNO DE LOS INSTRUMENTOS. 6(1).

MICHEL, I. F. (2015). *DISEÑO, INSTALACIÓN Y ANÁLISIS ENERGÉTICO DEL SISTEMA FOTOVOLTAICO*. Distrito Federal: UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO.

Naylamp Mechatronics . (23 de Octubre de 2019). *Naylamp Mechatronics* . Obtenido de Tutorial sensor digital de temperatura DS18B20:

https://naylampmechatronics.com/blog/46_tutorial-sensor-digital-de-temperatura-ds18b20.html

Ojeda, C. (25 de Abril de 2017). *World History Encyclopedia*. Obtenido de Papiro de Ani: <https://www.worldhistory.org/image/6550/papiro-de-ani/>

Ortega, E. Á. (2019). Microcontroladores. En *Sistema de monitoreo y* (pág. 15). Cd. México: UNAM.

Procolombia. (s.f.). *PLATAFORMA DEL INTERNET DE LAS COSAS*. (Procolombia)

Recuperado el 16 de Abri de 2023, de

<https://b2bmarketplace.procolombia.co/es/software-servicios-ti/software-empresarial/plataforma-del-internet-de-las-cosas-23817>

RA, J. (2010). History of Writing-TRAD.

- Seven. (18 de Octubre de 2022). *Sevensensor*. Recuperado el 14 de Mayo de 2023, de ¿Qué es un piranómetro y cómo mide la irradiancia?: <https://www.sevensensor.com/es/que-es-un-piranometro-y-como-mide-la-irradiancia>
- SOLARTEC. (1 de Julio de 2019). *SOLATEC*. (SOLARTEC) Recuperado el 15 de Marzo de 2020, de <http://www.solartec.com.ar/nosotros.html>
- Teón Vega, A. (2013). SISTEMAS FOTOVOLTAICOS. En *Implementación de un Sistema* (págs. 29-43). Chihuahua: cimar.
- UNIT ELECTRONICS. (12 de 05 de 2020). *UNIT ELECTRONICS*. Obtenido de Sensor de Corriente ACS712 30A: <https://uelectronics.com/producto/sensor-de-corriente-acs712-30a/?srsltid=AfmBOoq6VpVvW96dLrFbAA2hyLtRepnQ4CZuXs3wbO-3ZKIHU5ssXG4r>
- Vega de Kuyper, J. C. (2018). *Principios y aplicaciones de la energía fotovoltaica y de las baterías*. Santiago: Ediciones Universidad Católica de Chile.
- Virgen, M. A. (2007). *PLAN NACIONAL DE DESARROLLO 2007-2012*. Cd. de México: CEFP.
- Zou, X., Li, B., Zhai, Y., & Liu, H. (2012). Performance Monitoring and Test System for Grid-Connected Photovoltaic Systems. *IEEE*, 1-4.

4. Anexo A (Figura)

Figura 1

Entrada del Instituto Tecnológico de Mérida



Nota: Adaptado de TecNM-Campus Mérida [Fotografía], por Jorge Luis Sixto Roberto, 2020, GoogleMaps (<https://maps.app.goo.gl/WAqZRoCMvYh5bZxWA>).

Figura 2.

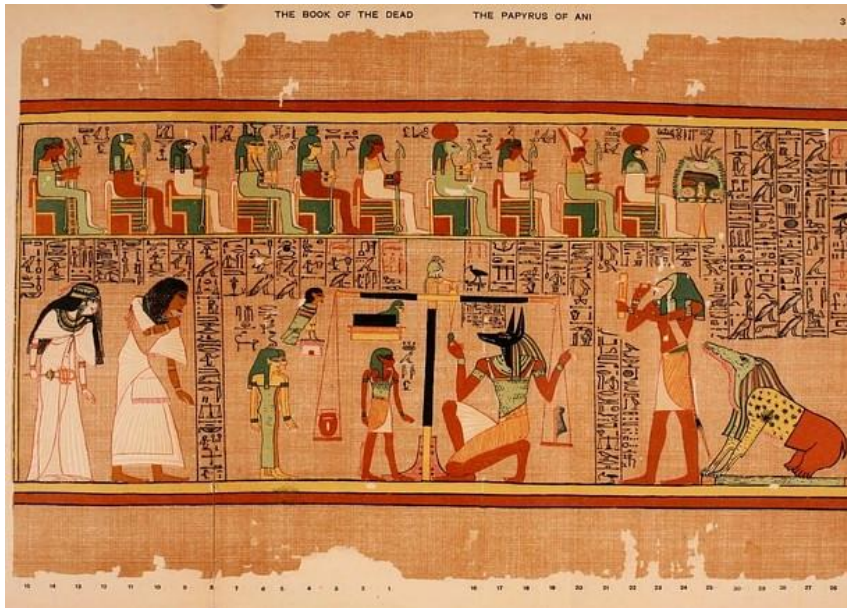
Tablilla Literaria excavada en Nínive



Nota: Adaptada de **Gilgamesh II 124ff**. Tablilla literaria excavada en Nínive, se aprecia una escritura pictográfica de tipo cuneiforme (Cuneiform Digital Library Initiative (CDLI). , 2023)

Figura 3.

Libro de los muertos



Nota: Adaptada de El Papiro de Ani. Texto del libro de la muerte. (Ojeda, 2017)

Figura 4

Sadie Pfeifer, la niña icónica



Nota: La niña supervisaba una máquina de husillo industrial, atenta a los hilos rotos y al cambio de carretes. (Bravo, 2020).

Figura 5

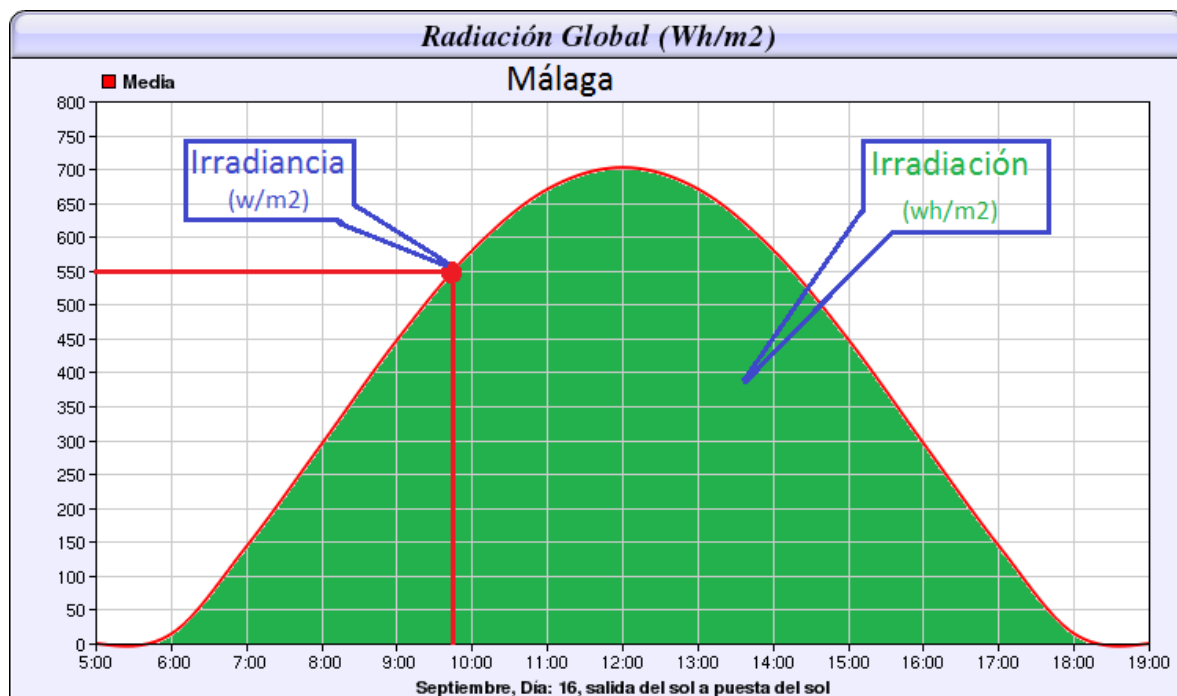
Paneles Solares



Nota: Adaptado de **Sistema fotovoltaico, Solar y Energía solar**, por Sebastian Ganso, 2017. Pixabay (<https://pixabay.com/es/photos/sistema-fotovoltaico-solar-2742302>)

Figura 6

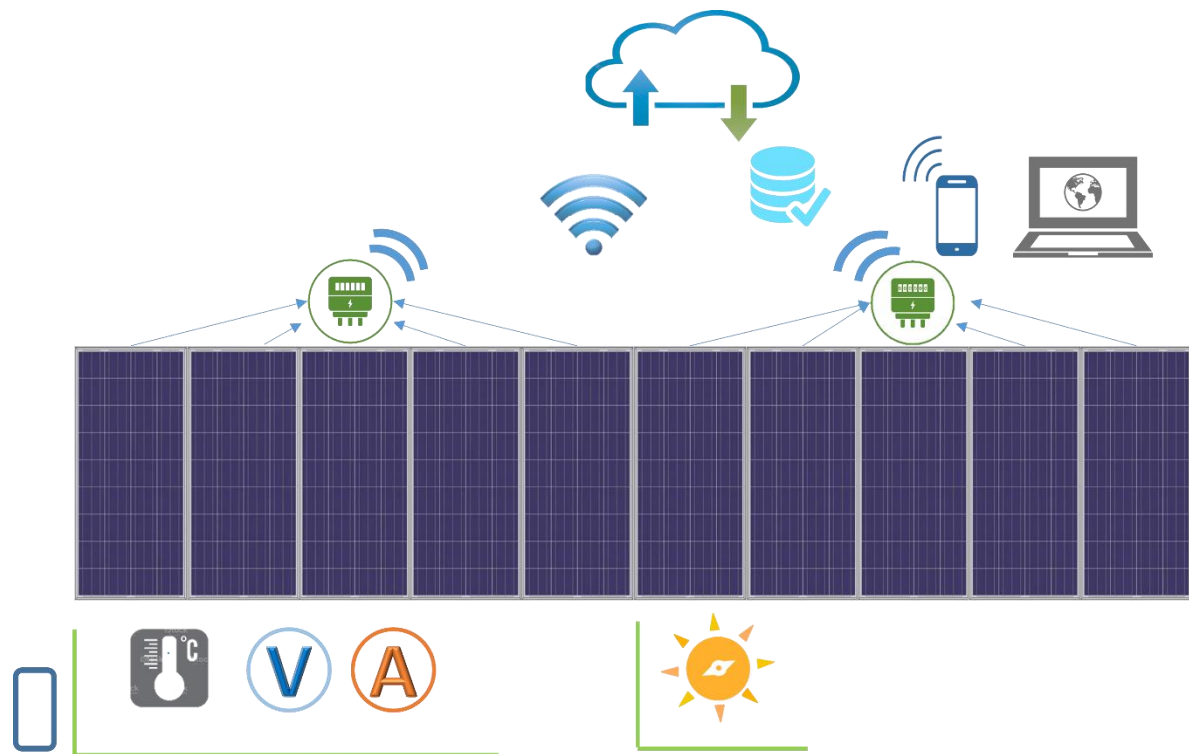
Gráfica de Radiación global.



Nota: Adaptada de Representación gráfica de la irradiancia e irradiación. (HelioEsfera, 2020)

Figura 7

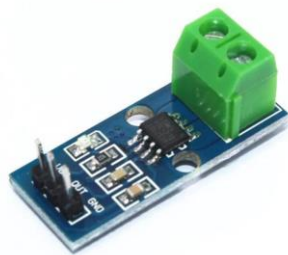
Sistema de monitoreo



Nota: Visualización esquemática de los componentes del sistema monitoreo físico.

Figura 8

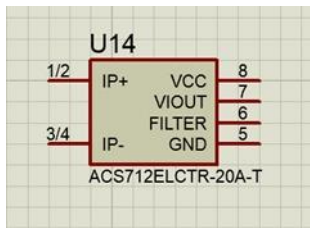
ACS712



Nota: Aspecto físico del sensor de corriente. (UNIT ELECTRONICS, 2020)

Figura 9

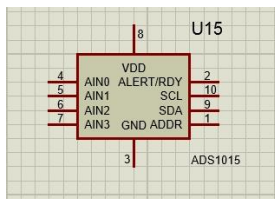
Diagrama ACS712 30A



Nota: Diagrama del sensor de corriente de Proteus.

Figura 10

Diagrama ADC1015



Nota: Diagrama del convertidor analógico digital en Proteus.

Figura 11

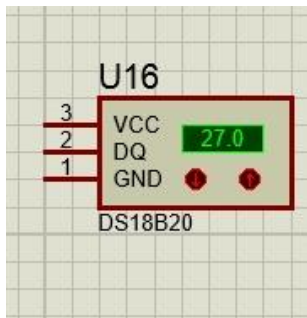
Sensor DS18B20



Nota: Aspecto físico del sensor de temperatura. (Naylamp Mechatronics , 2019)

Figura 12

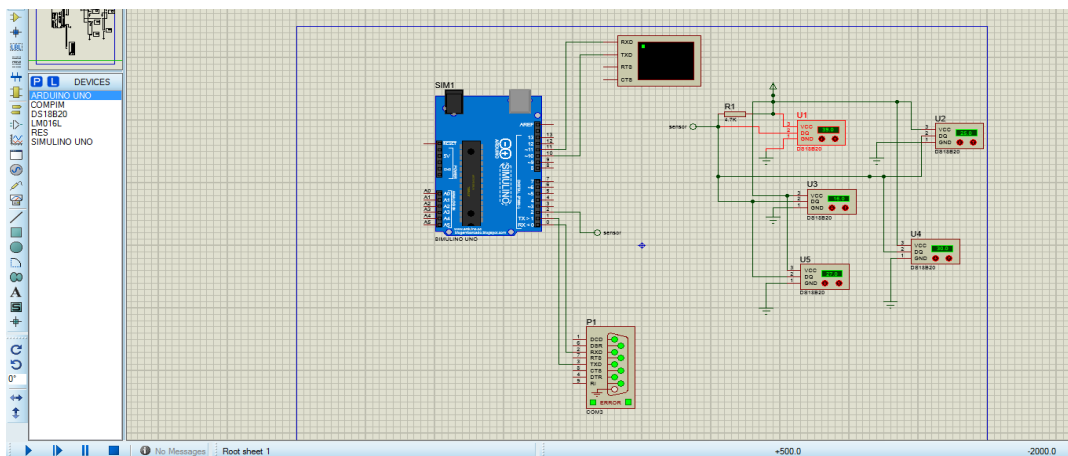
Diagrama del DS18B20



Nota: Diagrama del sensor de temperatura

Figura 13

Circuito de sensores DS18B20



Nota: Circuito implementado en el programa de Proteus.

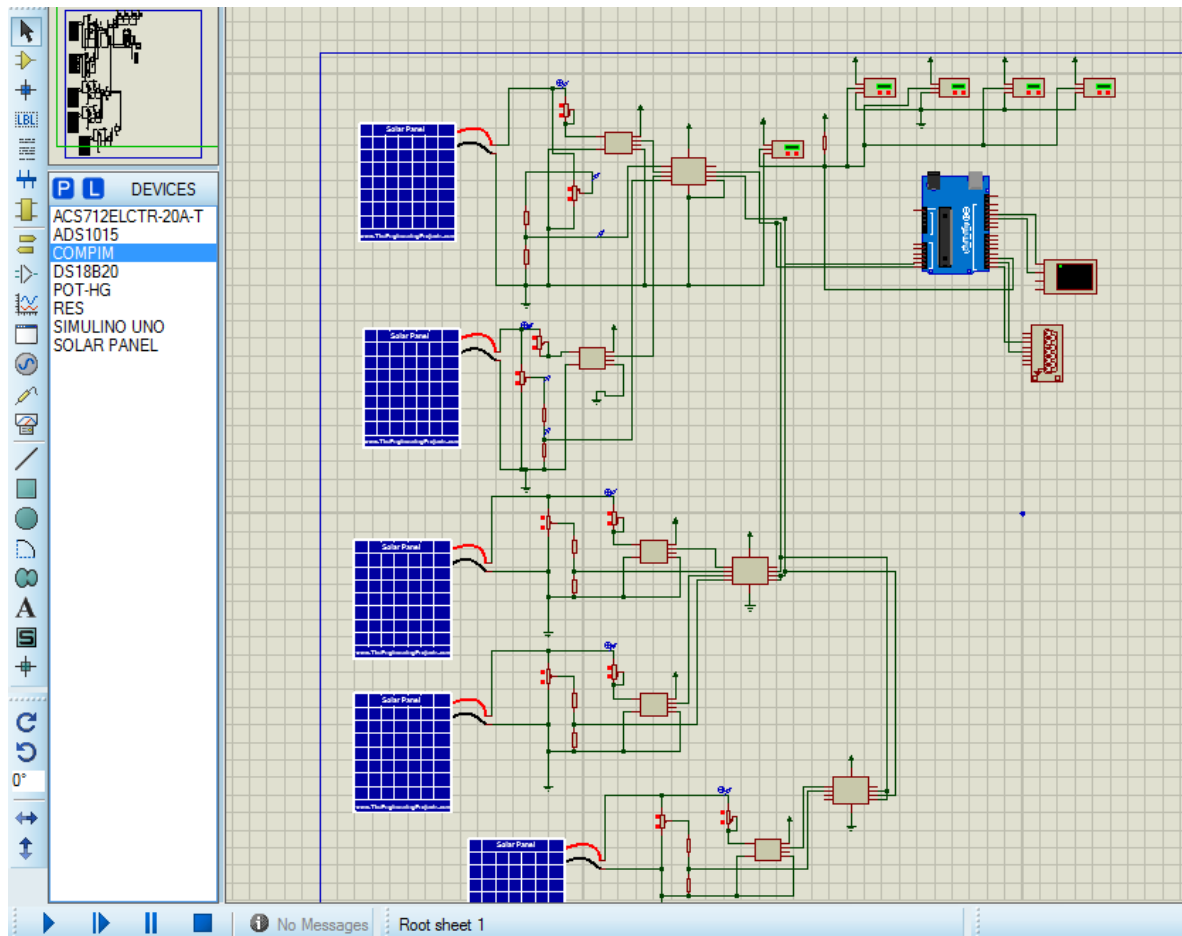
Adquisición de direcciones de los sensores de temperatura



Simulación de lectura de sensores de temperatura



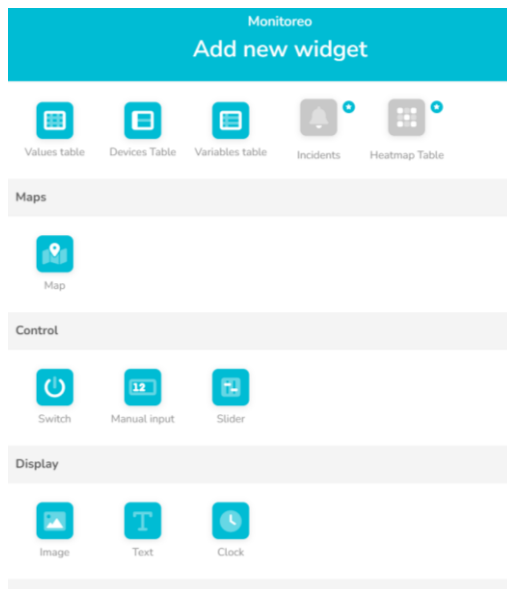
Conexión del Sistema FV al Arduino



Nota: Arreglo de 5 Paneles solares y sus sensores al Arduino.

Figura 17

Interfaz gráfica para variables contables



Nota: Tipos de interfaces gráficas para la adquisición de datos. Adaptada de Ubidots.

5. Anexo B (Tablas)

Tabla 1

Datos del panel solar S72PC-315

Panel Solar con Celdas de 156mm de Silicio Policristalino		
Modelo	S72PC-315	
Condiciones de Medición	STC	NOCT
Voltaje de Circuito Abierto (Voc)	44.3 V	44.37 V
Voltaje en el punto de máxima potencia (Vmpp)	37.3 V	37.23 V
Corriente de cortocircuito (Isc)	8.93 A	7.22 V
Corriente en el punto de máximo de potencia (Impp)	8.44 A	6.82 A
Potencia máxima (Pmax)	315 W	253 W
Eficiencia del módulo*STC	16.26%	

Nota. Adaptada del datasheet S72PC, recopilación de características eléctricas del panel solar. Fuente: Solartec, Energía Renovable.

Tabla 2

Sensores para el sistema Fotovoltaico

Sensores	Cantidad por panel	Cantidad por datalogger	Cantidad por arreglo Fotovoltaico.
Temperatura	1	5	10
Corriente	1	5	10
Voltaje	1	5	10
Irradiancia	0	0	1
Total	5	15	31

Nota. Sensores para el sistema fotovoltaico de 10 paneles solares. Fuente: Autor.

Tabla 3

Características del ACS712

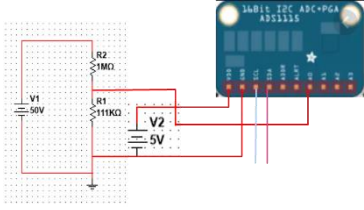
Sensor	Características	Tipo
--------	-----------------	------

Módulo ACS712	Max. Amperaje: 20A Alimentación: 5,0 V, Ancho de banda de 50 kHz. Sensibilidad de salida de 66 a 185 mV / A	Analógico
------------------	---	-----------

Nota: Características eléctricas del sensor de corriente.

Tabla 4

Circuito Divisor de Voltaje

Circuito	Material	Diagrama
Divisor de voltaje	*Resistencia de 1M Ω *Potenciómetro de 200K Ω *ADS1115	

Nota: Lista de componentes para medir el voltaje del panel solar.

Tabla 5

Sensor DS18B20

Sensor	Descripción	Tipo
DS18B20	Rango: -50 °C -+125 °C. Alimentación: 3.3-5V Chip: DS18B20. Sensor: resistente al agua. Comunicación: One wire	Digital

Nota: Característica eléctricas del sensor de temperatura

Tabla 6

Sensor de irradiancia

Sensor	Descripción	Tipo
--------	-------------	------

	Alimentación: 12/24V	
	Rango espectral: 0-2000W / m2	
	Rango de longitud de onda: 400-1100nm	
BGT-SR1	Sensibilidad: 200 ~ 800 μ v * w-1 * m2	Digital
	Longitud estándar del cable: 1,5 m.	
	Señal de salida: 4~ 20mA /0 ~	
	200V/RS485	

Nota: Características eléctricas del sensor BGT-SR1.

6. Anexo C (Códigos)

Código 1

Lectura de direcciones del DS1

```
#include <OneWire.h>

#include <DallasTemperature.h>

const int pinDatosDQ = 2;

OneWire oneWireObjeto(pinDatosDQ);

DallasTemperature sensorDS18B20(&oneWireObjeto);

void setup() {

  Serial.begin(9600);

  sensorDS18B20.begin();

}

void loop() {

  sensorDS18B20.requestTemperatures();

  Serial.print("Temperatura sensor 1: ");

  Serial.print(sensorDS18B20.getTempCByIndex(0));

  Serial.println(" C");

  Serial.print("Temperatura sensor 2: ");

  Serial.print(sensorDS18B20.getTempCByIndex(1));

  Serial.println(" C");

  Serial.print("Temperatura sensor 3: ");

  Serial.print(sensorDS18B20.getTempCByIndex(2));

  Serial.println(" C");

  Serial.print("Temperatura sensor 4: ");

  Serial.print(sensorDS18B20.getTempCByIndex(3));

  Serial.println(" C");

  Serial.print("Temperatura sensor 5: ");

  Serial.print(sensorDS18B20.getTempCByIndex(4));

  Serial.println(" C");

  Serial.print("Temperatura sensor 6: ");

  Serial.print(sensorDS18B20.getTempCByIndex(5));

  Serial.println(" C");

  delay(1000);

}
```

Código 2

Lectura de sensores de temperatura

```
#include <OneWire.h>

#include <DallasTemperature.h>

const int pinDatosDQ = 2;

OneWire oneWireObjeto(pinDatosDQ);

DallasTemperature sensorDS18B20(&oneWireObjeto);

void setup() {

  Serial.begin(9600);

  sensorDS18B20.begin();

}

void loop() {

  sensorDS18B20.requestTemperatures();

  Serial.print("Temperatura sensor 1: ");
  Serial.print(sensorDS18B20.getTempCByIndex(0));
  Serial.println(" C");

  Serial.print("Temperatura sensor 2: ");
  Serial.print(sensorDS18B20.getTempCByIndex(1));
  Serial.println(" C");

  Serial.print("Temperatura sensor 3: ");
  Serial.print(sensorDS18B20.getTempCByIndex(2));
  Serial.println(" C");

  Serial.print("Temperatura sensor 4: ");
  Serial.print(sensorDS18B20.getTempCByIndex(3));
  Serial.println(" C");

  Serial.print("Temperatura sensor 5: ");
  Serial.print(sensorDS18B20.getTempCByIndex(4));
  Serial.println(" C");

  Serial.print("Temperatura sensor 6: ");
  Serial.print(sensorDS18B20.getTempCByIndex(5));
  Serial.println(" C");

  delay(1000);
```

```

#include <SoftwareSerial.h>

SoftwareSerial pserial(10,11); //RX y TX

#include <OneWire.h>

#include <DallasTemperature.h>

OneWire ourWire(2);          //Se establece el pin 2 como bus OneWire

DallasTemperature sensors(&ourWire); //Se declara una variable u objeto para nuestro sensor

DeviceAddress sensor1 = {0x28, 0x32, 0xC5, 0xB8, 0x0, 0x0, 0x0, 0xE0}; //dirección del sensor 1
DeviceAddress sensor2 = {0x28, 0x31, 0xC5, 0xB8, 0x0, 0x0, 0x0, 0xB9}; //dirección del sensor 2
DeviceAddress sensor3 = {0x28, 0x35, 0xC5, 0xB8, 0x0, 0x0, 0x0, 0x65}; //dirección del sensor 3
DeviceAddress sensor4 = {0x28, 0x34, 0xC5, 0xB8, 0x0, 0x0, 0x0, 0x52}; //dirección del sensor 4
DeviceAddress sensor5 = {0x28, 0x33, 0xC5, 0xB8, 0x0, 0x0, 0x0, 0xD7}; //dirección del sensor 5

float temp1, temp2, temp3, temp4, temp5;

void setup() {

  Serial.begin(9600);
  pserial.begin(9600);
  sensors.begin(); //Se inicia el sensor
}

void loop() {
  sensors.requestTemperatures(); //envía el comando para obtener las temperaturas
  delay(500);

  temp1= sensors.getTempC(sensor1); //Se obtiene la temperatura en °C del sensor 1
  temp2= sensors.getTempC(sensor2); //Se obtiene la temperatura en °C del sensor 2
  temp3= sensors.getTempC(sensor3); //Se obtiene la temperatura en °C del sensor 3
  temp4= sensors.getTempC(sensor4); //Se obtiene la temperatura en °C del sensor 4
  temp5= sensors.getTempC(sensor5); //Se obtiene la temperatura en °C del sensor 5

  if (temp1== -127)
  {}
  else {
    pserial.print("Temperatura 1 = ");
    pserial.print(temp1);
    pserial.println(" C");
  }
}

```

```
if (temp2== -127 )
{
else{
pserial.print("Temperatura 2 = ");
pserial.print(temp2);
pserial.println(" C");
}
if (temp3== -127 )
{
else{
pserial.print("Temperatura 3 = ");
pserial.print(temp3);
pserial.println(" C");
}
if (temp4== -127 )
{
else{
pserial.print("Temperatura 4 = ");
pserial.print(temp4);
pserial.println(" C");
}

if (temp5== -127 )
{
else{
pserial.print("Temperatura 5 = ");
pserial.print(temp5);
pserial.println(" C");
}
```

Código 3

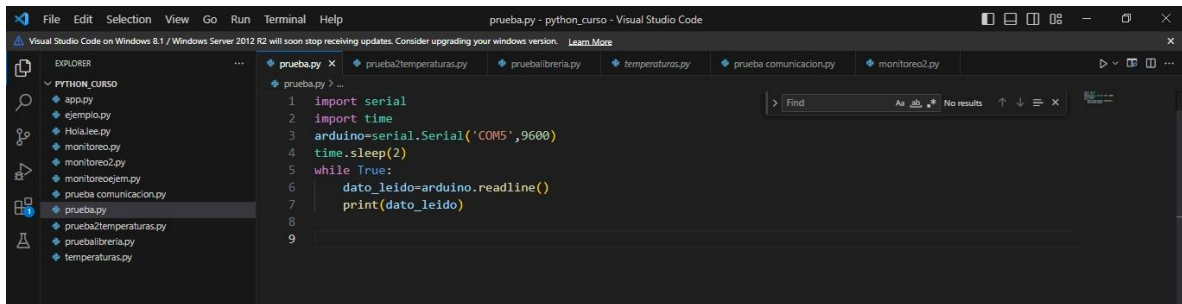
Comunicación y lectura de puerto COM

```
#include <SoftwareSerial.h>
SoftwareSerial pserial(10,11); //RX y TX
#include <OneWire.h>
#include <DallasTemperature.h>
OneWire ourWire(2);          //Se establece el pin 2 como bus OneWire
DallasTemperature sensors(&ourWire); //Se declara una variable u objeto para nuestro sensor
DeviceAddress sensor1 = {0x28, 0x32, 0xC5, 0xB8, 0x0, 0x0, 0x0, 0xE0}; //dirección del sensor 1
DeviceAddress sensor2 = {0x28, 0x31, 0xC5, 0xB8, 0x0, 0x0, 0x0, 0xB9}; //dirección del sensor 2
DeviceAddress sensor3 = {0x28, 0x35, 0xC5, 0xB8, 0x0, 0x0, 0x0, 0x65}; //dirección del sensor 3
DeviceAddress sensor4 = {0x28, 0x34, 0xC5, 0xB8, 0x0, 0x0, 0x0, 0x52}; //dirección del sensor 4
DeviceAddress sensor5 = {0x28, 0x33, 0xC5, 0xB8, 0x0, 0x0, 0x0, 0xD7}; //dirección del sensor 5
float temp1, temp2, temp3, temp4, temp5;
void setup() {
    // put your setup code here, to run once:
    Serial.begin(9600);
    pserial.begin(9600);
    sensors.begin(); //Se inicia el sensor
}
void loop() {
    // put your main code here, to run repeatedly:
    sensors.requestTemperatures(); //envía el comando para obtener las temperaturas
    temp1= sensors.getTempC(sensor1); //Se obtiene la temperatura en °C del sensor 1
    temp2= sensors.getTempC(sensor2); //Se obtiene la temperatura en °C del sensor 2
    temp3= sensors.getTempC(sensor3); //Se obtiene la temperatura en °C del sensor 3
    temp4= sensors.getTempC(sensor4); //Se obtiene la temperatura en °C del sensor 4
    temp5= sensors.getTempC(sensor5); //Se obtiene la temperatura en °C del sensor 5
    delay(500);
    if (temp1== -127)
    {}
    else {
        pserial.print("Temperatura 1 = ");
        pserial.print(temp1); // ESTE ES PARA IMPRIMIR EN EL MONITOR SERIAL
        pserial.println(" C");
        Serial.println("t1:" + String(temp1));
        delay(500);
    }
    if (temp2== -127 )
    {}
    else{
        pserial.print("Temperatura 2 = ");
        pserial.print(temp2);
```

```
pserial.println(" C");
Serial.println("t2:"+String(temp2));
delay(500);
}
if (temp3== -127 )
{
}
else{
pserial.print("Temperatura 3 = ");
pserial.print(temp3);
pserial.println(" C");
Serial.println("t3:"+String(temp3));
delay(500);
}
if (temp4== -127 )
{
}
else{
pserial.print("Temperatura 4 = ");
pserial.print(temp4);
pserial.println(" C");
Serial.println("t4:"+String(temp4));
delay(500);
}
if (temp5== -127 )
{
}
else{
pserial.print("Temperatura 5 = ");
pserial.print(temp5);
pserial.println(" C");
Serial.println("t5:"+String(temp5));
delay(500);
}
delay(500);
}
```

Código 4

Lectura de datos en Python



The image shows a screenshot of the Visual Studio Code editor interface. The title bar at the top reads "prueba.py - python_curso - Visual Studio Code". The menu bar includes File, Edit, Selection, View, Go, Run, Terminal, and Help. A notification banner at the top states: "Visual Studio Code on Windows 8.1 / Windows Server 2012 R2 will soon stop receiving updates. Consider upgrading your windows version. [Learn More](#)".

The Explorer sidebar on the left shows a project named "PYTHON_CURSO" with the following files: `app.py`, `ejemplo.py`, `HolaLee.py`, `monitoreo.py`, `monitoreo2.py`, `monitoreo3em.py`, `prueba comunicacion.py`, `prueba.py` (selected), `prueba2temperaturas.py`, `pruebalibreria.py`, and `temperaturas.py`.

The main editor window displays the content of `prueba.py` with the following Python code:

```
1 import serial
2 import time
3 arduino=serial.Serial('COM5',9600)
4 time.sleep(2)
5 while True:
6     dato_leido=arduino.readline()
7     print(dato_leido)
8
9
```

A search bar in the top right corner of the editor shows "> Find" and "No results".

Nota: Programa en Python para recibir información desde Proteus

Código 5

Programa completo de Arduino para el Datalogger

```
#define PIN_ANALOG A0
#define ESPERA_LECTURAS 1000
#include <SoftwareSerial.h>
SoftwareSerial pserial(10,11); //RX y TX
#include <Wire.h>
#include <Adafruit_ADS1X15.h>
#include <OneWire.h>
#include <DallasTemperature.h>
OneWire ourWire(2);
DallasTemperature sensor(&ourWire);
Adafruit_ADS1015 ads;
Adafruit_ADS1015 ads1;
Adafruit_ADS1015 ads2;
DeviceAddress sensor1 = {0x28, 0x32, 0xC5, 0xB8, 0x0, 0x0, 0x0, 0xE0}; //dirección del sensor de temperatura 1
DeviceAddress sensor2 = {0x28, 0x31, 0xC5, 0xB8, 0x0, 0x0, 0x0, 0xB9}; //dirección del sensor de temperatura 2
DeviceAddress sensor3 = {0x28, 0x35, 0xC5, 0xB8, 0x0, 0x0, 0x0, 0x65}; //dirección del sensor de temperatura 3
DeviceAddress sensor4 = {0x28, 0x34, 0xC5, 0xB8, 0x0, 0x0, 0x0, 0x52}; //dirección del sensor de temperatura 4
DeviceAddress sensor5 = {0x28, 0x33, 0xC5, 0xB8, 0x0, 0x0, 0x0, 0xD7}; //dirección del sensor de temperatura 5

float Sensibilidad=0.100; //sensibilidad en Voltios/Amperio para sensor de 5A
float I=0;
int16_t adc0, adc1, adc2, adc3, adc4, adc5, adc6, adc7, adc8, adc9;
float volts0, volts1, volts2, volts3, volts4, volts5, volts6, volts7, volts8, volts9;
float volt1, volt2, volt3, volt4, volt5 ;
float corriente1, corriente2, corriente3, corriente4, corriente5;
float vcc = 5; // valor del Vcc
float r1 = 1000000; // 1M
float r2 = 111000; // 111K
// sensor de lux
long cronometro_lect=0;
long tiempo_transc;
unsigned int luminosidad;
float coeficiente_porc=100.0/1023.0;
void setup() {
  // put your setup code here, to run once:
  pserial.begin (9600);
  Serial.begin(9600);
  pserial.println("Lectura ADS");
  delay (200);
  // Iniciamos ADS1015
  ads.begin (0x48);
  ads1.begin (0x4A);
  ads2.begin (0x4B);
  sensor.begin();
}
```

```

void loop() {
  // put your main code here, to run repeatedly:
  // primer ads
  adc0=ads.readADC_SingleEnded(0);
  adc1=ads.readADC_SingleEnded(1);
  adc2=ads.readADC_SingleEnded(2);
  adc3=ads.readADC_SingleEnded(3);
  //segundo ads
  adc4=ads1.readADC_SingleEnded(0);
  adc5=ads1.readADC_SingleEnded(1);
  adc6=ads1.readADC_SingleEnded(2);
  adc7=ads1.readADC_SingleEnded(3);
  //tercer ads
  adc8=ads2.readADC_SingleEnded(0);
  adc9=ads2.readADC_SingleEnded(1);
  delay (40);
  //declaramos cronometro
  tiempo_transc=millis()-cronometro_lect;
  // A continuacion se obtiene su valor relativo a voltaje
  volts0 = ads.computeVolts(adc0);
  volts1 = ads.computeVolts(adc1);
  volts2 = ads.computeVolts(adc2);
  volts3 = ads.computeVolts(adc3);

  volts4 = ads1.computeVolts(adc4);
  volts5 = ads1.computeVolts(adc5);
  volts6 = ads1.computeVolts(adc6);
  volts7 = ads1.computeVolts(adc7);
  volts8 = ads2.computeVolts(adc8);
  volts9 = ads2.computeVolts(adc9);
  // Solicitamos la temperaturas leidas.
  sensor.requestTemperatures();
  float temp1= sensor.getTempC(sensor1);//Se obtiene la temperatura en °C del sensor 1
  float temp2= sensor.getTempC(sensor2);//Se obtiene la temperatura en °C del sensor 2
  float temp3= sensor.getTempC(sensor3);//Se obtiene la temperatura en °C del sensor 3
  float temp4= sensor.getTempC(sensor4);//Se obtiene la temperatura en °C del sensor 4
  float temp5= sensor.getTempC(sensor5);//Se obtiene la temperatura en °C del sensor 5
  // Iniciamos lectura de temperatura

  pserial.println("-----");
  pserial.print("Temperatura 1: ");
  while (temp1== -127)

```

```

    {
        sensor.requestTemperatures();
        temp1= sensor.getTempC(sensor1);
    }
    pserial.print(temp1);
    pserial.println(" C");
    Serial.println("t1:"+String(temp1,2));
    delay (500);

    pserial.println("-----");
    pserial.print("Temperatura 2: ");
    while (temp2== -127)
    {
        sensor.requestTemperatures();
        temp2= sensor.getTempC(sensor2);
    }
    pserial.print(temp2);
    pserial.println(" C");
    Serial.println("t2:"+String(temp2,2));
    delay (500);
    pserial.println("-----");
    pserial.print("Temperatura 3: ");
    while (temp3== -127)
    {
        sensor.requestTemperatures();
        temp3= sensor.getTempC(sensor3);
    }
    pserial.print(temp3);
    pserial.println(" C");
    Serial.println("t3:"+String(temp3,2));
    delay (500);

    pserial.println("-----");
    pserial.print("Temperatura 4: ");
    while (temp4== -127)
    {
        sensor.requestTemperatures();
        temp4= sensor.getTempC(sensor4);
    }
    pserial.print(temp4);
    pserial.println(" C");
    Serial.println("t4:"+String(temp4,2));
    delay (500);

    pserial.println("-----");
    pserial.print("Temperatura 5: ");
    while (temp5== -127)
    {
        sensor.requestTemperatures();
        temp5= sensor.getTempC(sensor5);
    }
    pserial.print(temp5);
    pserial.println(" C");
    Serial.println("t5:"+String(temp5,2));
    delay (500);

```

//// Lectura de voltaje!!!!

```
volt1 = volts0 / (r2 / (r1 + r2));  
pserial.println("-----");  
pserial.print("Voltaje 1: ");pserial.print(" "); pserial.print(volt1); pserial.println("V");  
Serial.println("v1:"+String(volt1,2));  
delay (500);
```

```
volt2 = volts3 / (r2 / (r1 + r2));  
pserial.println("-----");  
pserial.print("Voltaje 2: ");pserial.print(" "); pserial.print(volt2); pserial.println("V");  
Serial.println("v2:"+String(volt2,2));  
delay (500);
```

```
volt3 = volts5 / (r2 / (r1 + r2));  
pserial.println("-----");  
pserial.print("Voltaje 3: ");pserial.print(" "); pserial.print(volt3); pserial.println("V");  
Serial.println("v3:"+String(volt3,2));  
delay (500);
```

```
volt4 = volts7 / (r2 / (r1 + r2));  
pserial.println("-----");  
pserial.print("Voltaje 4: ");pserial.print(" "); pserial.print(volt4); pserial.println("V");  
Serial.println("v4:"+String(volt4,2));  
delay (500);
```

```
volt5 = volts9 / (r2 / (r1 + r2));  
pserial.println("-----");  
pserial.print("Voltaje 5: ");pserial.print(" "); pserial.print(volt5); pserial.println("V");  
Serial.println("v5:"+String(volt5,2));  
delay (500);
```

```
///Lectura de la corriente  
corriente1=corriente1+(volts1-2.5)/Sensibilidad; //Ecuación para obtener la corriente  
pserial.println("-----");  
pserial.print("Corriente 1: ");pserial.print(" "); pserial.print(corriente1); pserial.println("A");  
Serial.println("c1:"+String(corriente1,2));  
delay (500);
```

```
corriente2=corriente2+(volts2-2.5)/Sensibilidad; //Ecuación para obtener la corriente  
pserial.println("-----");  
pserial.print("Corriente 2: ");pserial.print(" "); pserial.print(corriente2); pserial.println("A");  
Serial.println("c2:"+String(corriente2,2));  
delay (500);
```

```
corriente3=corriente3+(volts4-2.5)/Sensibilidad; //Ecuación para obtener la corriente  
pserial.println("-----");  
pserial.print("Corriente 3: ");pserial.print(" "); pserial.print(corriente3); pserial.println("A");  
Serial.println("c3:"+String(corriente3,2));  
delay (500);
```

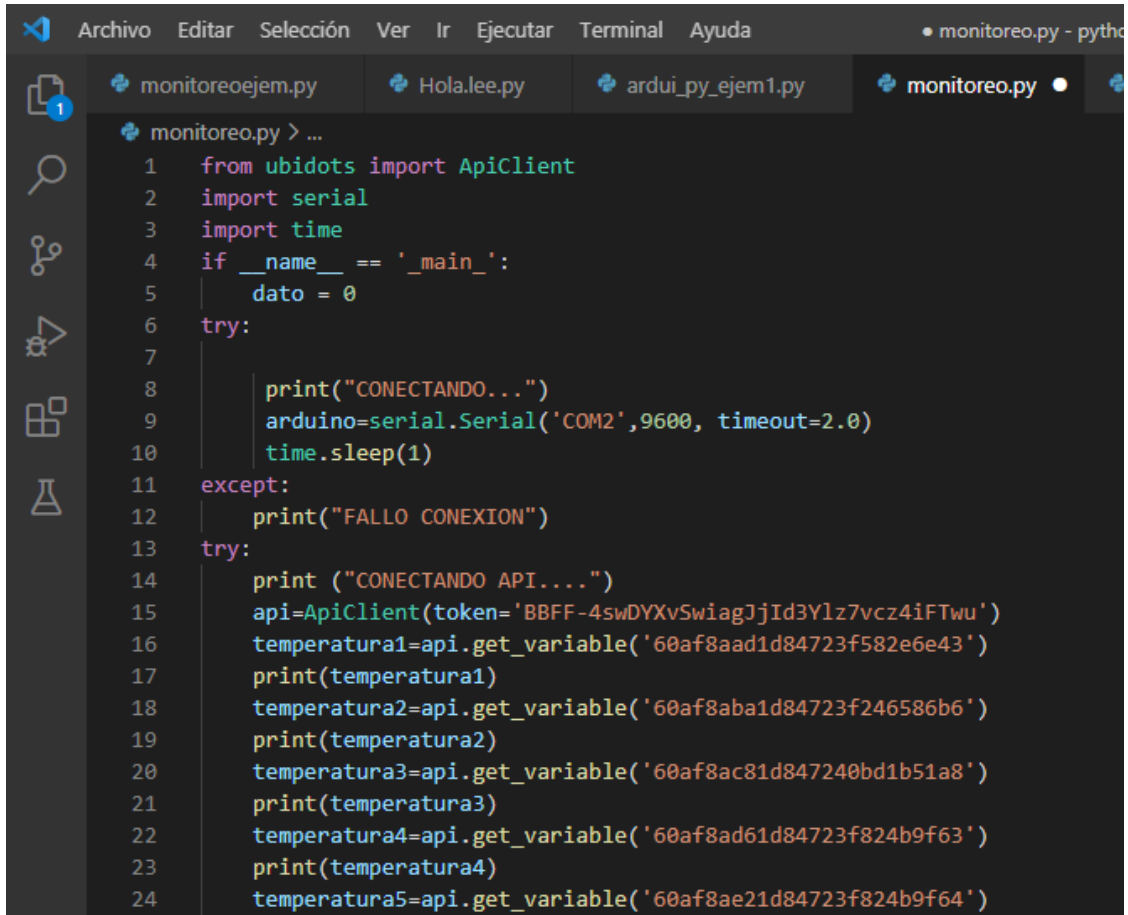
```
corriente4=corriente4+(volts6-2.5)/Sensibilidad; //Ecuación para obtener la corriente  
pserial.println("-----");  
pserial.print("Corriente 4: ");pserial.print(" "); pserial.print(corriente4); pserial.println("A");  
Serial.println("c4:"+String(corriente4,2));  
delay (500);
```

```
corriente5=corriente5+(volts8-2.5)/Sensibilidad; //Ecuación para obtener la corriente
pserial.println("-----");
pserial.print("Corriente 5: ");pserial.print(" "); pserial.print(corriente5); pserial.println("A");
Serial.println("c5:"+String(corriente5,2));
delay (500);

//Iniciamos lecturas de LDR
if(tiempo_transc>ESPERA_LECTURAS)
{
    cronometro_lect=millis();
    luminosidad=analogRead(PIN_ANALOG);
    float finalirra=luminosidad*coeficiente_porc;
    pserial.print ("La luminosidad es ");
    pserial.print(finalirra);
    pserial.println("%");
    Serial.println("irr:"+String(finalirra,2));
}
    delay(500);
}
```

Código 6

Tokens de Ubidots

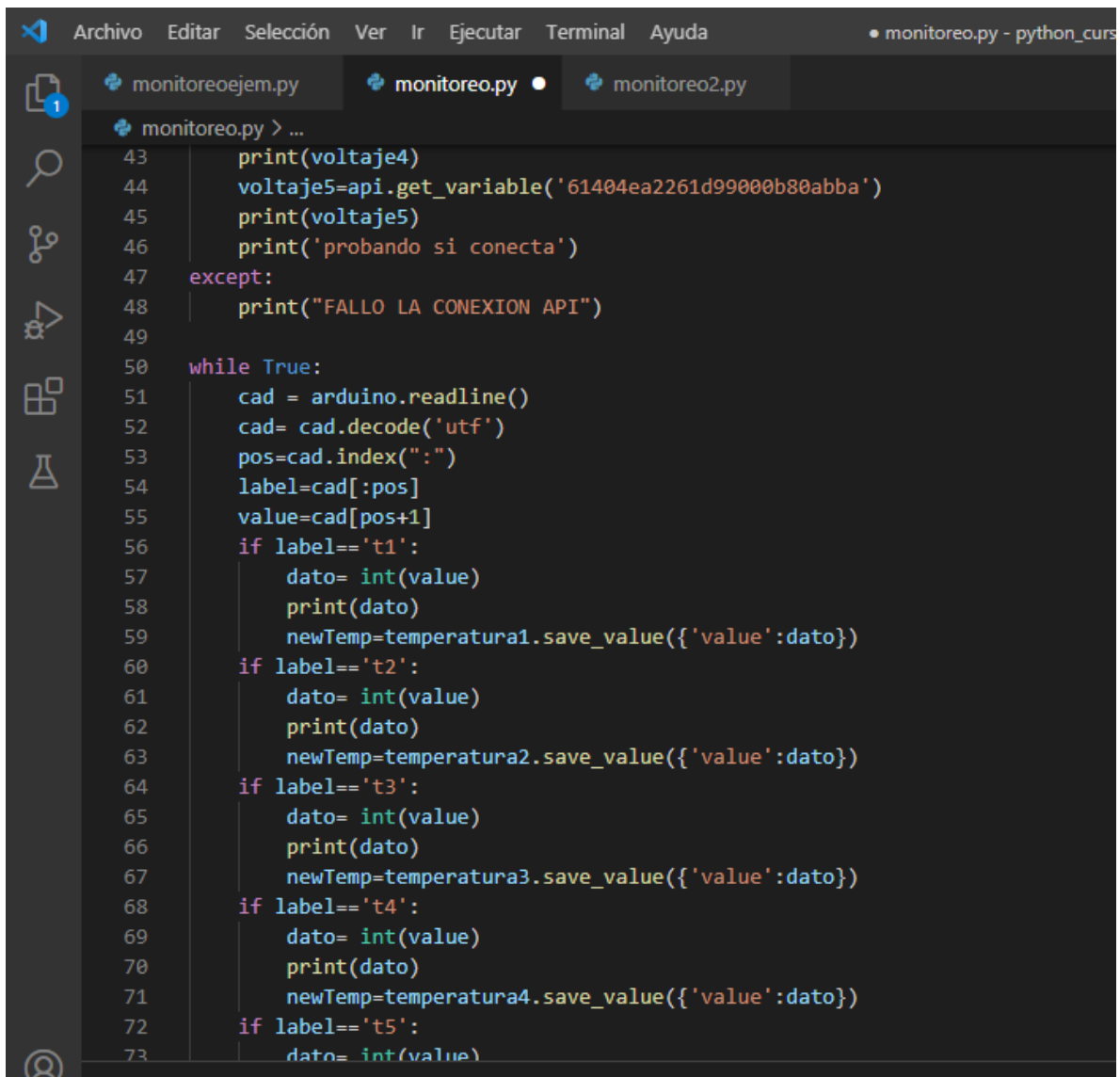
A screenshot of a code editor window with a dark theme. The editor has a menu bar at the top with options: Archivo, Editar, Selección, Ver, Ir, Ejecutar, Terminal, Ayuda. Below the menu bar is a tab bar showing four open files: monitoreoejem.py, Hola.lee.py, ardui_py_ejem1.py, and monitoreo.py (which is the active file). The main editing area shows a Python script for monitoring Ubidots data. The script imports ApiClient from ubidots, serial, and time. It has a main function that tries to connect to a serial port (COM2) and then tries to connect to the Ubidots API using a token. It then retrieves five temperature variables from the API and prints them. The script is as follows:

```
1 from ubidots import ApiClient
2 import serial
3 import time
4 if __name__ == '__main__':
5     dato = 0
6     try:
7
8         print("CONECTANDO...")
9         arduino=serial.Serial('COM2',9600, timeout=2.0)
10        time.sleep(1)
11    except:
12        print("FALLO CONEXION")
13    try:
14        print ("CONECTANDO API...")
15        api=ApiClient(token='BBFF-4swDYXvSwiagJjId3Ylz7vcz4iFTwu')
16        temperatura1=api.get_variable('60af8aad1d84723f582e6e43')
17        print(temperatura1)
18        temperatura2=api.get_variable('60af8aba1d84723f246586b6')
19        print(temperatura2)
20        temperatura3=api.get_variable('60af8ac81d847240bd1b51a8')
21        print(temperatura3)
22        temperatura4=api.get_variable('60af8ad61d84723f824b9f63')
23        print(temperatura4)
24        temperatura5=api.get_variable('60af8ae21d84723f824b9f64')
```

Nota: Visualización para asignar cada variable la dirección de Ubidots.

Código 7

Identificación de variables de Arduino y envío de datos a Ubidots.



```
Archivo  Editar  Selección  Ver  Ir  Ejecutar  Terminal  Ayuda  • monitoreo.py - python_curs
monitoreoejem.py  monitoreo.py  monitoreo2.py
monitoreo.py > ...
43     print(voltaje4)
44     voltaje5=api.get_variable('61404ea2261d99000b80abba')
45     print(voltaje5)
46     print('probando si conecta')
47 except:
48     print("FALLO LA CONEXION API")
49
50 while True:
51     cad = arduino.readline()
52     cad= cad.decode('utf')
53     pos=cad.index(":")
54     label=cad[:pos]
55     value=cad[pos+1]
56     if label=='t1':
57         dato= int(value)
58         print(dato)
59         newTemp=temperatura1.save_value({'value':dato})
60     if label=='t2':
61         dato= int(value)
62         print(dato)
63         newTemp=temperatura2.save_value({'value':dato})
64     if label=='t3':
65         dato= int(value)
66         print(dato)
67         newTemp=temperatura3.save_value({'value':dato})
68     if label=='t4':
69         dato= int(value)
70         print(dato)
71         newTemp=temperatura4.save_value({'value':dato})
72     if label=='t5':
73         dato= int(value)
```

Nota: Fragmento de código para la identificación de cada variable proveniente de Proteus.