Prototipo de monitoreo remoto para medir las condiciones climáticas y parámetros de funcionamiento de un sistema solar-eólico

Yimy Rodolfo Suárez Vergara

James Dean Balaguera Villa

Universidad del Magdalena

Facultad de Ingeniería

Programa Ingeniería Electrónica

Santa Marta, Colombia

2021

Prototipo de monitoreo remoto para medir las condiciones climáticas y parámetros de funcionamiento de un sistema solar-eólico

Yimy Rodolfo Suárez Vergara

James Dean Balaguera Villa

Trabajo presentado como requisito parcial para optar al título de:

Ingeniero Electrónico

Director:

Ing. Carlos Arturo Robles Algarín, PhD

Codirector:

Ing. Diego Andrés Restrepo Leal, Msc

Línea de Investigación:

Ambiente y sostenibilidad

Grupo de Investigación:

MAGMA-Ingeniería (Matemática Aplicada a la Ingeniería)

Universidad del Magdalena

Facultad de Ingeniería

Programa de Ingeniería Electrónica

Santa Marta, Colombia

2021

Nota de aceptación:

Aprobado por el Consejo de Programa en cumplimiento de los requisitos exigidos por la Universidad del Magdalena para optar al título de Ingeniería Electrónica.

_____

Jurado

_____

Jurado

Santa Marta, ____ de ____del _____

A Dios y nuestros padres, gracias

AGRADECIMIENTOS

Agradecemos al grupo al grupo de investigación MAGMA-Ingeniería (Matemática Aplicada a la Ingeniería), por abrirnos sus puertas en este proyecto y brindarnos todos sus recursos humanos y materiales para el desarrollo de la investigación.

A la Universidad del Magdalena, gracias a su financiación este trabajo ha podido llevarse a cabo.

Al director Carlos Roles Algarín y codirector Diego Restrepo Leal, por su acompañamiento, consejos y apoyo durante el proceso de realización de este proyecto, sin su guía esto no sería posible.

Resumen

En esta investigación se presenta el desarrollo de un sistema de adquisición y transmisión inalámbrica de variables climáticas y parámetros de funcionamiento de sistemas híbridos (solar-eólico). Este proyecto está articulado con la investigación denominada “Sistema de Generación Híbrido Solar-Eólico para Brindar Servicios de Primera Necesidad en Instituciones Educativas Distritales de las Zonas Rurales de Santa Marta, Colombia”, financiado por Minciencias, en el cual se busca solventar problemáticas como la falta de agua potable y el flujo intermitente de la energía eléctrica por medio de la instalación de un sistema híbrido, en una institución educativa rural perteneciente al distrito de Santa Marta. Las variables que serán monitoreadas corresponden a irradiación solar, velocidad del viento, potencia de un sistema solar-eólico, temperatura y humedad, todo esto se realizará a través de un microcontrolador de 32 bits. El sistema hace uso de las tecnologías GSM/GPRS y Wi-Fi para su comunicación con una base de datos para realizar el debido almacenamiento y tratamiento de datos, estas tecnologías nos permitirán contar con dos medios de trasmisión de los datos y su utilización dependerá de la disponibilidad de red celular y de la conexión de internet en el lugar de instalación. Como resultado se obtiene un sistema embebido cuya utilidad final será instalarlo en una escuela rural del distrito de Santa Marta, que se podrá monitorear de forma remota a través de una interfaz de usuario que fue diseñada por el equipo de trabajo. Se resalta que la información obtenida por los sensores servirá como apoyo para futuros proyectos que busquen desarrollar técnicas para estimar o predecir el comportamiento de variables climáticas.

ABSTRACT

This research presents the development of a wireless acquisition and transmission system of climatic variables and operating parameters of hybrid systems (solar-wind). This project is articulated with the research called "Solar-Wind Hybrid Generation System to Provide Primary Needs Services in District Educational Institutions in the Rural Zones of Santa Marta, Colombia", financed by Minciencias, which seeks to solve problems such as lack of drinking water and the intermittent flow of electricity through the installation of a hybrid system in a rural educational institution belonging to the district of Santa Marta. The variables that will be monitored correspond to solar irradiation, wind speed, power of a solar-wind system, temperature and humidity, all this will be done through a 32-bit microcontroller. The system makes use of GSM / GPRS and Wi-Fi technologies for communication with a database to carry out the proper storage and processing of data, these technologies will allow us to have two means of data transmission and its use it will depend on the availability of the cellular network and the internet connection at the installation site. As a result, an embedded system is obtained whose final utility will be to install it in a rural school in the Santa Marta district, which can be monitored remotely through a user interface that was designed by the work team. It is highlighted that the information obtained by the sensors will serve as support for future projects that seek to develop techniques to estimate or predict the behavior of climatic variables**.**

Keywords: Sistema de monitoreo, Variables climáticas, Comunicación inalámbrica, Energía hibrida.

Contenido

Introducción

El proceso de medición de las variables climáticas es de vital importancia para modelar el potencial de las zonas de interés, con el fin de realizar diversos estudios como definir la posibilidad de la instalación de un sistema fotovoltaico, granja eólica, o en general cualquier sistema de transformación energética, como los sistemas híbridos.

Con el avance de la tecnología y las telecomunicaciones, se amplía la posibilidad de efectuar el análisis de datos a una considerable distancia, en tiempo real, sin necesidad de cables. Es por esta razón que se planteó desarrollar un sistema de medición inalámbrico versátil, con el fin de obtener una transmisión directa del comportamiento de las variables climáticas y de parámetros de funcionamiento de un sistema solar-eólico, como lo son la temperatura de operación y la potencia suministrada. El sistema de medición envía la información a una plataforma con el fin de almacenar y procesar los datos para facilitar su estudio y comprensión.

El sistema cuenta con la implementación de un circuito para gestionar el suministro energético del prototipo con el fin de dotar al sistema de un funcionamiento con baterías y evitar la dependencia de la red eléctrica.

La implementación de este proyecto permite la recopilación de información relevante de parámetros climáticos esenciales en la región, como la radiación solar y velocidad del viento, esto con el fin de suministrar datos actualizados a las bases de datos de monitoreo climatológico. Estos datos son cruciales para realizar más proyectos de transición energética en el territorio.

Diversos proyectos enfocados en el uso de energías renovables están surgiendo en nuestra región caribe proyectos como: “Sistema de Generación Híbrido Solar-Eólico para Brindar Servicios de Primera Necesidad en Instituciones Educativas Distritales de las Zonas Rurales de Santa Marta, Colombia, Financiado por Minciencias, se contempla la selección de 1 de las 25 escuelas rurales que existen en el distrito.

La creciente tendencia hacia las energías renovables ocasiona que los dispositivos enfocados en la recolección de datos climáticos relacionados con dichas energías tengan un gran impacto en el desarrollo de estas, debido a que permite realizar un estudio más detallado de zonas específicas, ayudando a la creación de base de datos y a determinar la viabilidad de los proyectos con mayor facilidad.

1 Capítulo 1

1.1 Fundamentación teórica

A fin de realizar una mejor integración de los conceptos y de conocer los diferentes desarrollos ya realizados en el marco de la investigación se realiza una búsqueda en las diferentes bases de datos encontrado trabajos de investigación que mantienen una semejanza a la lógica de funcionamiento de este, Pero que, mediante distintos abordajes y metodologías, buscan llegar a resultados, respuestas y conclusiones diferentes, de los cuales se mencionan:

[1] V. Pagola, R. Peña, J. Segundo, A. Ospino, “Rapid Prototyping of a Hybrid PV–Wind Generation System Implemented in a Real-Time Digital Simulation Platform and Arduino”, electronics, vol. 8, pp. 102, January 2019. En este trabajo de investigación, se muestra la implementación de un modelo híbrido completo, es decir un sistema fotovoltaico-eólico en una plataforma de simulación digital en tiempo real, además del desarrollo del sistema de control del convertidor electrónico, implementado en controlador (Arduino Due).

[2] S. Sinha, S.Chandel, ”Review of recent trends in optimization techniques for solar photovoltaic–wind based hybrid energy systems”, elsevier, vol. 50, pp. 755-769, May 2015. Éste, presenta una revisión sobre las tendencias en las técnicas de optimización utilizadas para el diseño y desarrollo de los sistemas híbridos de energía solar fotovoltaica-eólica e identifican 16 métodos de optimización, implementando algoritmos de inteligencia artificial ya que requieren menos tiempo de cálculo y mejor precisión, buena convergencia en comparación con los convencionales.

[3] N. Priyadarshi, V. K. Ramachandaramurthy, S. Padmanaban, F. Azam, “An Ant Colony Optimized MPPT for Standalone Hybrid PV-Wind Power System with Single Cuk Converter”, European journal of physics, vol. 12, pp. 167, January 2019. En éste documento, se presenta la realización de un sistema híbrido de obtención de energía eléctrica con rastreador de punto de máxima potencia (MPPT)[4] en lugares rurales (aplicaciones residenciales), El algoritmo MPPT basado en Ant Colony Optimization (ACO)[5] se emplea con el fin de realizar un seguimiento rápido de la potencia del sistema de energía, en el cual se adopta la estrategia de control del

inversor Fuzzy Logic Control (FLC)[6] en comparación con el control proporcional-integral (PI)[7] clásico.

1.2 Características de un sensor

En la selección de los diferentes sensores para el muestreo de las variables implicadas en el desarrollo del dispositivo, es necesario conocer ciertas características que afectan de una manera u otra la implementación de dichos sensores en el sistema.

Entre algunas de esas características podemos resaltar:

- **Resolución:** La resolución corresponde a la mínima perturbación en la señal de entrada del sensor que produce una alteración en la salida, lo que permite realizar medidas más exactas según la variable que se requiera. Es necesario aclarar que esta característica está ligada a la incertidumbre máxima permitida en el sistema con respecto a la variable en la que se involucre dicho sensor, un ejemplo claro sería el uso de un sensor de temperatura cuya resolución sea de 1°C, esto quiere decir que se necesita una variación mínima de 1°C en la temperatura para que el sensor genere un cambio en su salida.

- **Sensibilidad:** Corresponde a la variación de la salida con respecto a la entrada del sensor, nuevamente esta característica depende la variable que se desea medir, más específicamente de los rangos en los que se espera que se encuentre la medida de dicha variable y es importante tenerla en cuenta a la hora de seleccionar el microcontrolador a utilizar.

- **Spam:** Es el rango de medida del sensor, esta variable delimita los intervalos de medición del sistema, y es necesario tener claro cuáles son esos intervalos para no incurrir en la selección de un sensor que no cumpla con los rangos máximos necesarios o por el contrario los supere de manera exagerada, por ejemplo, retomando el uso del sensor de temperatura, teniendo en cuenta que el sistema estará expuesto a condiciones climáticas de la costa caribe, se espera que los rangos en que se encuentren las diferentes medidas de temperatura sea entre los 20°C y 60°C, esto debido a la exposición continua a la irradiación solar, por ello

un sensor cuyo spam máximo no alcanza los 60°C no realizaría la tarea correctamente, por el contrario un sensor cuyo spam máximo alcance los 1000°C, a pesar de que cumpliría con su tarea no es el uso óptimo del mismo.

- **Precisión:** La precisión corresponde al porcentaje de error del sensor, todo sensor genera un porcentaje de error el cual varía dependiendo del método utilizado para realizar la medida y de los materiales utilizados en el dispositivo, para garantizar una lectura fiable es importante manejar porcentajes de error bajos, lo más cercano al 0% que sea posible.

- **Alimentación de voltaje:** Esta característica, aunque muchas veces es pasada por alto, es necesario tener en cuenta que en el caso de este sistema que será alimentado por un módulo fotovoltaico conectado a una batería de 12V, la alimentación energética de dichos sensores debe estar dentro del consumo energético máximo de la batería.

1.3 Sensores utilizados

En la figura 1 se observa una ilustración de los diferentes dispositivos y sensores utilizados en el desarrollo del proyecto.

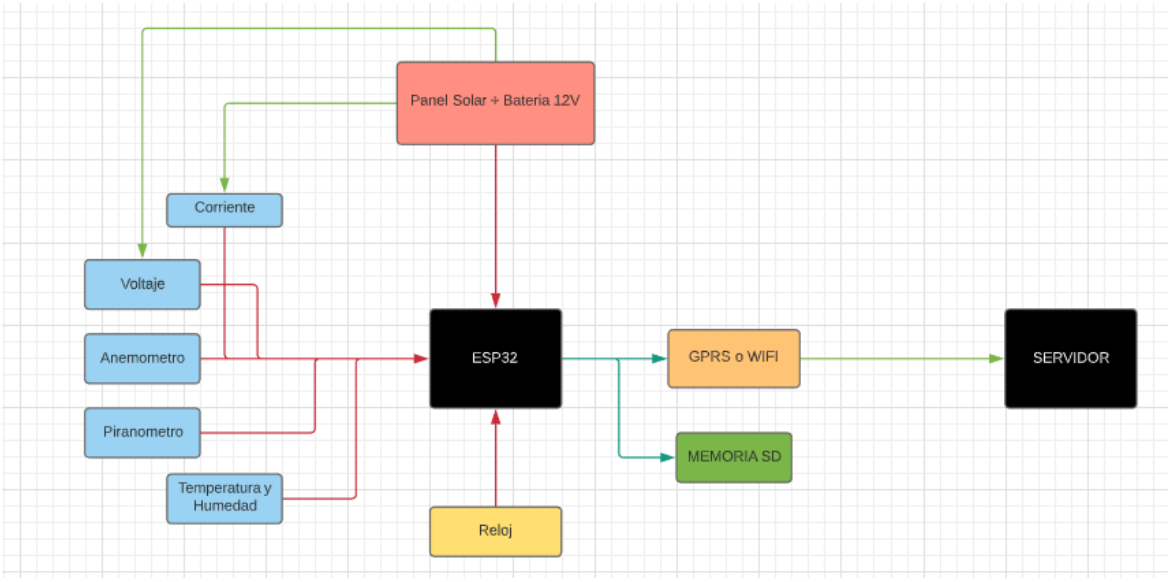

Figura 1. Diagrama básico de conexiones.

Para realizar las distintas lecturas para el funcionamiento del sistema se utilizan los siguientes instrumentos:

1.3.1 Sensor de intensidad de radiación solar (Piranómetro SR05-A1):

Un piranómetro, mostrado en la figura 2, es un dispositivo utilizado para medir la irradiación solar, el sensor Hukseflux SR05-A1 es un dispositivo que tiene una sensibilidad de $11{,}24 \times 10^{-6}$ V / (W/m^2) y una precisión de ±5% cuya salida de voltaje bajo irradiación solar normal varía entre 0 y 11 mV, con un funcionamiento similar al de los paneles solares ya que es un sensor de tipo pasivo (no requiere alimentación de voltaje). Este dispositivo genera voltaje en función de la irradiación solar que recibe, permite ser utilizado en microcontroladores, pero dada su alta resolución es necesario realizar una amplificación del voltaje que este genera en su salida.

Figura 2. Piranómetro Hukseflux SR05-A1.

Para realizar la antes mencionada amplificación de voltaje se utiliza un amplificador de instrumentación AD623, como el mostrado en la figura 3.

“El amplificador de instrumentación es un amplificador diferencial tensión-tensión cuya ganancia puede establecerse de forma muy precisa y que ha sido optimizado para que opere de acuerdo con su propia especificación aún en un entorno hostil. Es un elemento esencial de los sistemas de medida, en los que se ensambla como un bloque funcional que ofrece características funcionales propias e independientes de los restantes elementos con los que interacciona. Para ello, se requiere:

A) Tengan unas características funcionales que sean precisas y estables.

B) Sus características no se modifiquen cuando se ensambla con otros elementos” [8]

Figura 3. Módulo INA AD623.

La figura 3, presenta el módulo amplificador INA AD623, esta placa permite modificar la ganancia del amplificador de forma manual a través de un potenciómetro, también incluye un generador de onda negativo, permitiendo así alimentar el amplificador con una fuente simple que en este caso es de 5v, soportando una tensión máxima de 5.5v.

Tiene un rango de amplificación de 0 a 150mV, y cuenta con opciones de configuraciones de salida negativa y positiva, es decir, si se requiere una salida positiva es necesario conectar la entrada S+ a la salida positiva del sensor y la entrada S- conectarla a tierra, para que la salida sea negativa es necesario invertir la conexión, la entrada S- se conecta a la salida negativa del sensor y S+ se conecta a tierra. Con la entrada Ref es posible calibrar el nivel de tensión de salida que represente el valor 0 en la medida de la señal, por tal motivo se conecta Ref a tierra para que el nivel de tensión sea 0v en cuanto la medida este en 0, en el caso que se requiera un valor diferente a 0, solo basta con alimentar tal entrada con el valor deseado.

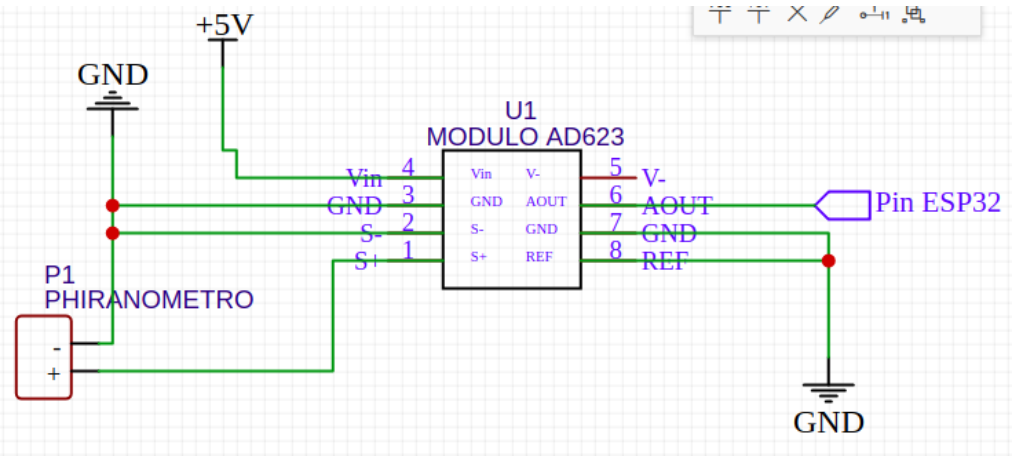

Figura 4. Conexión del módulo INA AD623.

En la figura 4, se puede observar lo antes dicho, la configuración efectuada es la de S- conectada a tierra y S+ a la salida positiva del sensor, con lo que se puede lograr una salida positiva, Ref se conecta a tierra para que marque 0V en cuanto sea 0 el valor de la medida.

Es importante aclarar que, esta última configuración de las entradas S+ y S-, no están en el diseño PCB, pero esto se solucionó sin necesidad de modificar el diseño, de la siguiente manera.

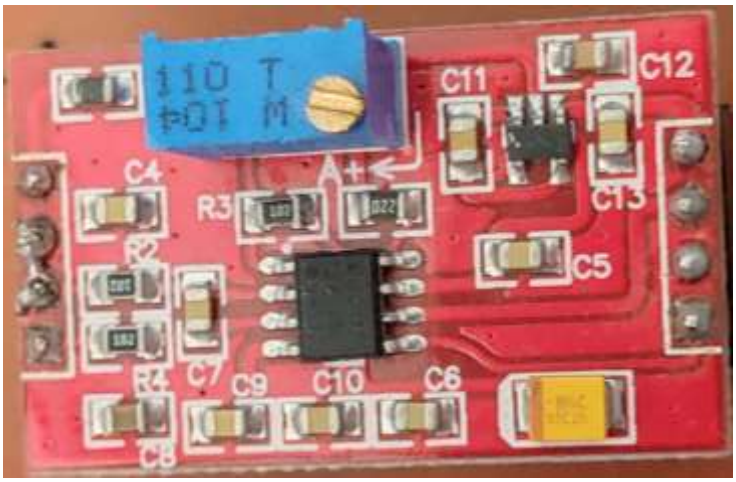

Figura 5. Adaptación del módulo AD623

La figura 5, es el módulo montado en la placa final, y se puede observar los pines de la izquierda del amplificador, los 2 pines del centro que corresponden de arriba hacia abajo a GND y S-, están unidos.

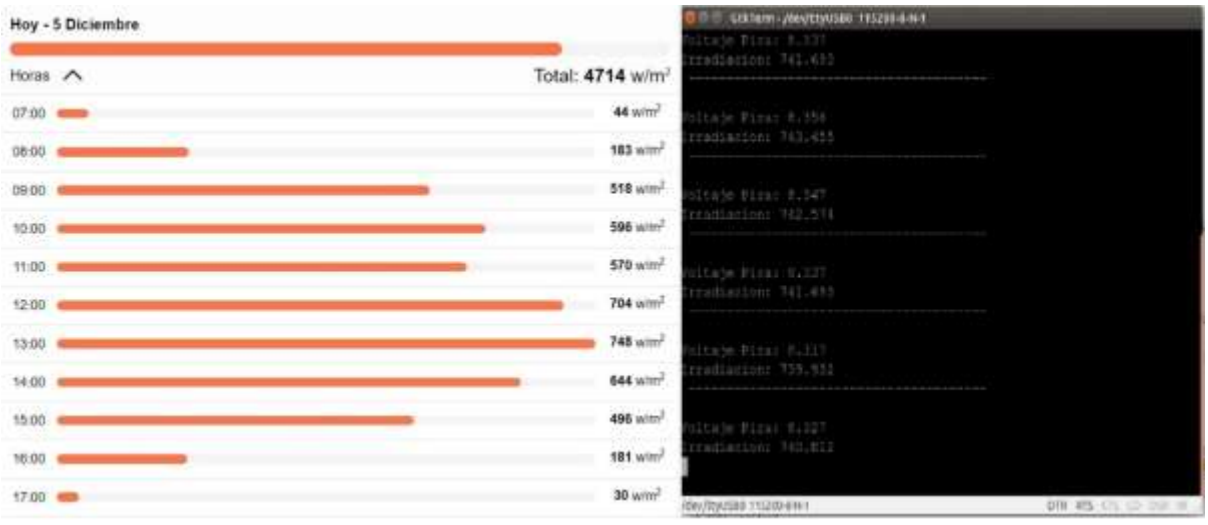

Figura 6. Prueba Piranómetro SR05-A1.

En la figura 6, se observa las pruebas realizadas al piranómetro a las 12:30 del mediodía y comparadas con las predicciones realizadas de la irradiación en la ciudad de Santa Marta

registradas en la página tutiempo.net, con una ganancia ajustada de 101, la variable "ADC_VOLT" es el valor obtenido en el microcontrolador, es decir, la señal amplificada por el AD623, el valor "Voltaje pira" es el voltaje real del piranómetro en milivoltios, por último, la IRRADIACION, que es el valor de "Voltaje pira" dividido entre la sensibilidad del piranómetro ($11.24 \times 10^{-6}\ \frac{\mathrm{V}}{(\mathrm{W/m^2})}$), y sus unidades son, vatios por metro cuadrado.

1.3.2 Anemómetro (QS-FS Adafruit):

El sensor QS-FS de Adafruit es un anemómetro análogo con un spam de 0 a 60 m/s con una salida lineal entre 0 y 32,4m/s con una salida respectiva de 0,4 V a 2 V, teniendo en cuenta que la velocidad de 32,4 m/s equivale aproximadamente a 115 Km/h esa es una velocidad cercana a la de un huracán de categoría 1 (Velocidades entre 118 y 154 Km/) usaremos el sensor dentro de su margen de trabajo lineal lo cual lo hace ideal para su uso con microcontroladores y con un voltaje de alimentación de entre 7 y 24 VDC. El sensor cuenta con una resolución de 0,1 m/s y con un error máximo de 1 m/s. El sensor responde a la ecuación 1

$$Y = (20.3 * X) - 8.7 \qquad (1)$$

Dónde:
Y es la velocidad en m/s.
X es el voltaje en V.

Se puede ver a continuación el sensor antes dicho, en la figura 7.

Figura 7. Anemómetro QS-FS.

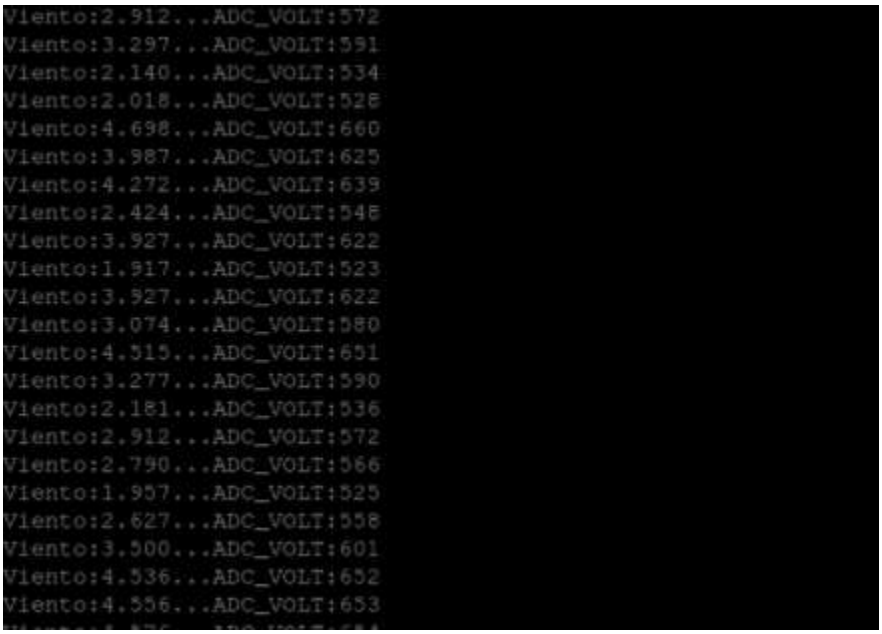

Figura 8. Prueba QS-FS Adafruit.

En la figura 8 se observa las mediciones realizadas en prueba con el anemómetro.

1.3.3 Sensor temperatura y humedad (SHT20):

El sensor digital SHT20 utilizado para medir la temperatura y humedad, mostrado en la figura 9, tiene las siguientes especificaciones:

- **Especificaciones medición de humedad:**

Principio: Capacitivo.
Resolución: 0.04% RH.
Precisión: ±0.1% RH.
Exactitud (20% - 80% RH): ±3% RH típico.
Histéresis: ±1% RH.
Rango de operación: 0% - 100% RH.

- **Especificaciones medición de temperatura:**

Principio: Bandgap.
Resolución: 0.01 °C.
Precisión: ±0.1 °C.
Exactitud (5°C - 60°C): ±0.3°C típico.
Rango de operación: -40°C - 125°C.

Con un voltaje de alimentación de 3.3V y para la entrega de datos utiliza el protocolo I2C.

Figura 9. Sonda SHT20.

I2C, mencionado anteriormente, es un puerto y protocolo de comunicación serial, define la trama de datos y las conexiones físicas para transferir bits entre 2 dispositivos digitales. El puerto incluye dos cables de comunicación, SDA y SCL. Además, el protocolo permite conectar hasta 127 dispositivos esclavos con esas dos líneas, con hasta velocidades de 100, 400 y 1000 kbits/s. También es conocido como IIC o TWI – Two Wire Interfaz.

El protocolo I2C es uno de los más utilizados para comunicarse con sensores digitales, ya que a diferencia del puerto Serial, su arquitectura permite tener una confirmación de los datos recibidos, dentro de la misma trama, entre otras ventajas.

La conexión de tantos dispositivos al mismo bus es una de las principales ventajas. Además, si comparamos a I2C con otro protocolo serial, como Serial TTL, este incluye más bits en su trama de comunicación que permite enviar mensajes más completos y detallados.” [9], la figura 10, muestra el diagrama de conexión en el protocolo I2C.

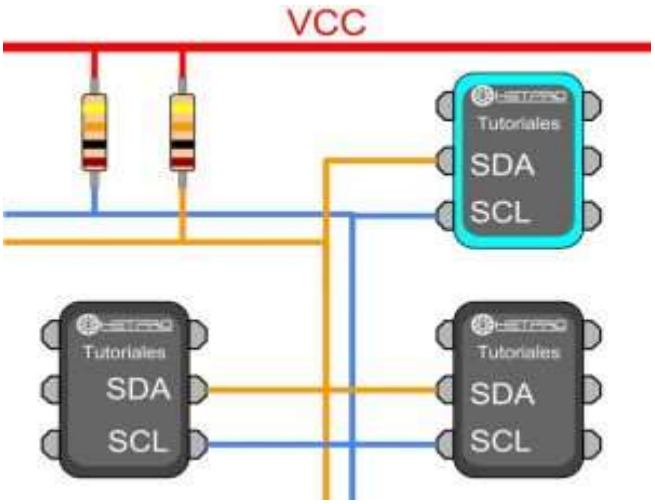

Figura 10. Esquema de comunicación I2C.

El SHT20 tiene un protocolo de comunicación especifico, primero tenemos que observar la secuencia de Start y Stop, que nos permiten identificar cuando se empieza o termina la transmisión de los datos del Slave(SHT20) al master, estas secuencias las podemos observar en la figura 11 y 12 respectivamente para Start y Stop.

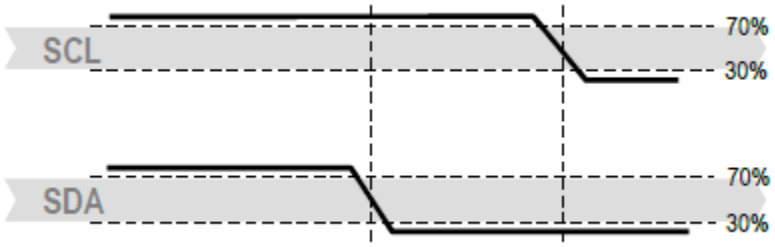

Figura 11. Secuencia Start SHT20

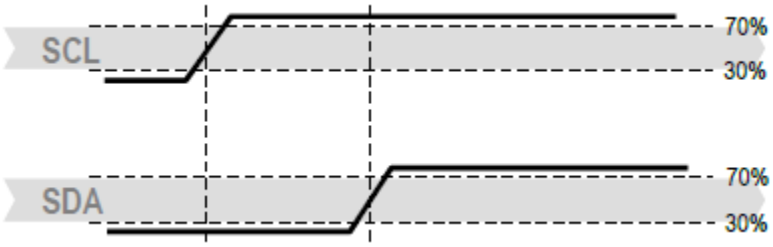

Figura 12. Secuencia Stop SHT20

Teniendo en cuenta las secuencias anteriores, luego de enviar la secuencia de Start, la siguiente secuencia está compuesta de 7 bits que corresponden a la dirección del dispositivo y un bit de dirección de SDA (Write:'0' o Read:'1'), esperamos la respuesta del sensor el cual pone el SDA pin en Low. Luego, se procede a enviar el comando correspondiente a lo que se desea realizar, los comandos básicos del sensor se observan en la figura 13.

Command	Comment	Code
Trigger T measurement	hold master	1110'0011
Trigger RH measurement	hold master	1110'0101
Trigger T measurement	no hold master	1111'0011
Trigger RH measurement	no hold master	1111'0101
Write user register		1110'0110
Read user register		1110'0111
Soft reset		1111'1110

Figura 13. Tabla de comandos básicos SHT20

Luego de esto el sensor nos envía una trama de datos de 14 bits, además de dos bits (43 y 44) que nos determinan si se está realizando la lectura de humedad o temperatura (bit 43 en '0'

corresponde a temperatura y si está en ‘1’ corresponde a humedad, el bit 44 no tiene uso actualmente).

El proceso completo de manera resumida se observa en la figura 14.

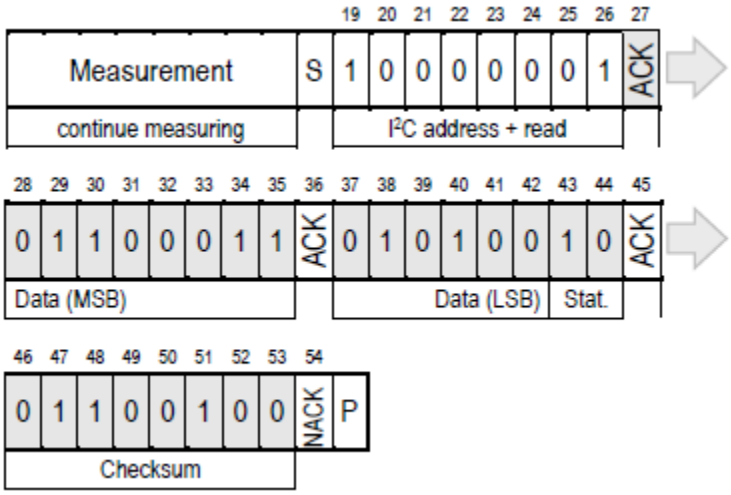

Figura 14. Comunicación SHT20

Después de tener los datos, se utilizan las siguientes ecuaciones para calcular sus valores reales.

Para la humedad se emplea la ecuación 2.

$$\mathrm{RH} = -6 + \left(125 \times \frac{S_{RH}}{2^{16}}\right) \quad (2)$$

Y para la temperatura se utiliza la ecuación 3.

$$\mathrm{T} = -46{,}85 + \left(175{,}72 \times \frac{S_T}{2^{16}}\right) \quad (3)$$

En la figura 15 se observa las pruebas de funcionamiento realizadas con el sensor

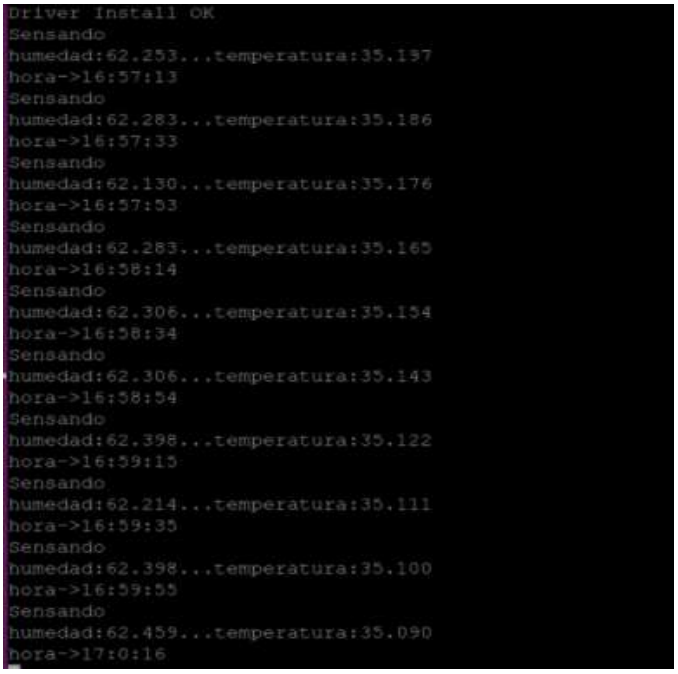

Figura 15. Prueba sensor SHT20.

1.3.4 Sensor de voltaje FZ0430: El módulo sensor de voltaje FZ0430 es un módulo bastante simple, el cual consiste en un divisor de tensión conformado por dos resistores uno de 7.5K y uno de 30K, con el cual es capaz de reducir el voltaje medido en una razón de 0.2 y por lo tanto permite medir una tensión máxima de 25 V con procesador de 5V.

Utiliza tecnología AVR de microchip de 10 Bits, por lo que la resolución de este módulo es de aproximadamente 0.00489V (5V/1023), por lo que el mínimo valor de tensión a detectar es de 0.02445V y se encuentra en la figura 16.

Figura 16. Sensor de voltaje FZ0430.

En la figura 17 se pueden observar las pruebas de funcionamiento realizadas con el sensor.

Figura 17. Pruebas sensor FZ0430 y WCS1800.

1.3.5 Sensor de Corriente WCS1800: Se obtiene el valor de la corriente a través del sensor de efecto hall WCS1800 que a 5 voltios de alimentación tiene un rango de medida lineal de -30 a 30 A, y una alta sensibilidad de 66 mV/A, puede leer corriente AC y DC y también puede alimentarse con una fuente de 3.3v, dicho sensor de corriente se puede observar a continuación en la figura 18.

Figura 18. Sensor de corriente WCS1800.

Como se mencionó anteriormente, este es un sensor de efecto hall, es decir, si una corriente atraviesa el sensor y la misma está en presencia de un campo magnético que fluye de forma vertical al sensor, éste obtendrá en su salida un voltaje que será proporcional al producto de la fuerza del campo magnético y la corriente, a esto se le denomina como FEM (Fuerza Electromotriz). La tensión que se induce en la salida del sensor será cuantificada con un pin (I/O)

del microcontrolador y será leída con un conversor de señal analógica a digital (ADC) que se configura en el mismo, a través de código.

Por practicad, se eligió una fuente poco mayor de 3.3v para alimentar el sensor, ya qué, es al nivel de tensión que se alimentan la mayoría de los componentes, y debido también a que el consumo del sistema se encuentra dentro del rango de medida lineal del sensor, comprendido entre -20 y 20 A.

A continuación, se muestra en la figura 19, la ecuación de trabajo del sensor alimentado a 3.3v:

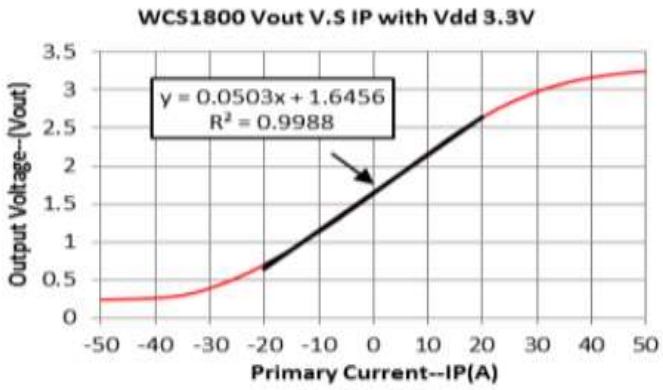

Figura 19 Grafica Corriente vs Voltaje WCS1800.

La ecuación 4 que se presenta en la figura 19 es:

$$Y = 0.0503 * x + 1.6456V \tag{4}$$

Donde Y corresponde al voltaje de salida del sensor y X a la corriente que está pasando a través de este.

Para calcular la corriente en función del voltaje de salida procedemos a despejar X de la ecuación (4).

$$X = \frac{(Y - 1.6456)}{0.0503} A \tag{5}$$

Un factor para tener en cuenta es, la manera correcta de cómo se debe conectar este sensor, la cual se muestra en la figura 20.

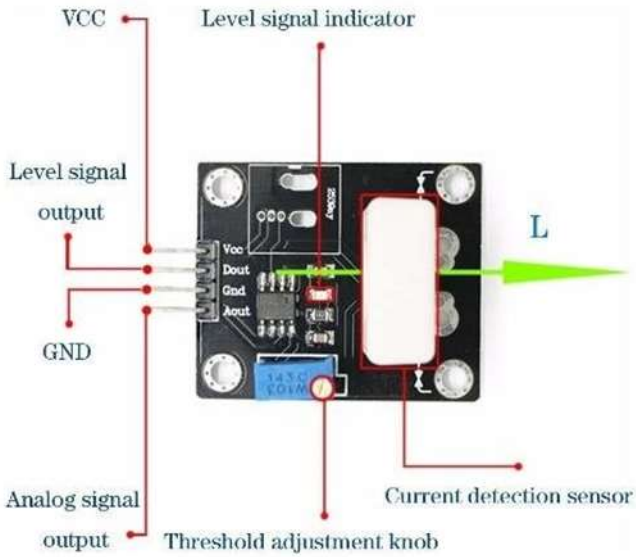

Figura 20. Esquema de conexión WCS1800.

Como podemos visualizar anterior mente en la figura 20, es la correcta conexión del dispositivo, la corriente L que va en dirección hacia la derecha, pasa a través de la pinza blanca. Éste tiene 1 led azul que indica el nivel de la señal de salida, el cual se puede ajustar con el potenciómetro, o sea, que el led enciende una vez que detecta el nivel ajustado, en la salida.

La fuente de alimentación se debe conectar entre VCC y GND, para conectarlo al microcontrolador, se debe conectar AOUT y GND a una entrada analógica de este, lo que reduce todo el trabajo a tomar una medida de corriente DC.

Al realizar varias pruebas, los datos arrojan que hay cierto error y dispersión en la medida de la señal, y eso es debido a la alimentación del sensor que en base a ella se determina el punto de partida, es decir, el voltaje equivalente a 0 amperios, como podemos apreciar en la ecuación (3), dice que el punto cero amperios es 1.6456 v, pero si alimentamos el sensor a 3.8v, el punto

de partida sería 1.9 (VCC/2), es decir, tendríamos un incremento en el error de la media. Motivo por el cual, hace dependiente la medida del sensor, a su fuente de alimentación.
Una vez dicho lo anterior se procede a hacerle un ajuste a la función teniendo en cuenta la fuente de alimentación y la ecuación (3), de lo cual se obtiene la siguiente ecuación:

$$X = \frac{\left(Y - \frac{V_{cc}}{2}\right)}{0.0503} \quad (6)$$

Como se pudo observar, el resultado de las pruebas de este sensor se encuentra en la figura 17.

1.3.6 Modulo reloj en tiempo real DS3231:

El DS3231 es un reloj en tiempo real de alta precisión que cuenta con un oscilador a cristal con compensación de temperatura (TCXO). La integración del oscilador a cristal en el propio circuito integrado, en conjunto con la compensación de temperatura, asegura la precisión a largo plazo. El RTC mantiene registro de segundos, minutos, horas, día de la semana, fecha, mes y año, la fecha es ajustada automáticamente a final de mes para meses con menos de 31 días, incluyendo las correcciones para año bisiesto.

El módulo se comunica con el microcontrolador a través del bus I2C con solamente 2 pines que pueden ser compartidos por varios dispositivos, véase el módulo a continuación en la figura 21.

Figura 21. Módulo DS3231.

En la figura 15, se puede observar las pruebas realizas al módulo y al sensor SHT20 pruebas que se realizaron simultáneamente.

1.3.7 Modulo lectura y escritura SD:

El módulo de lectura y escritura SD, es un dispositivo que permite el uso de almacenamiento externo utilizando memorias SD o microSD, las características de este módulo son:

- **Referencia:** ShieldMicroSD.
- **Compatible:** Tarjeta MicroSD y Micro SDHC.
- **Interfaz de comunicación:** SPI.
- **Pines:** 6 (GND, VCC, MISO, MOSI, SCK, CS).
- **Fuente de alimentación:** 4.5 V-5.5 V, 3.3 V circuito regulador de voltaje.
- **Dimensiones:** 4.5 x 2.4 x 0.5 cm.
- **Peso:** 6 g.

Figura 22. Módulo de lectura y escritura microSD.

En la figura 22, se puede observar el módulo de lectura y escritura microSD, el cual utiliza el protocolo de comunicación SPI:

“SPI es un bus que establece la comunicación entre máster y esclavo mediante 4 tipos de hilos que contiene señales diferentes:

• MISO, Máster In/Slave Out. Esta línea es una de las dos líneas unidireccionales. A través de esta línea se realiza la transmisión de datos de forma unidireccional desde la salida del esclavo a la entrada del maestro. Cuando el dispositivo no ha sido seleccionado para la comunicación, esta línea es puesta en un estado de alta impedancia para evitar interferencias.

• MOSI, Master Out/Slave In. Esta línea es la segunda de las dos líneas unidireccionales. A través de esta línea se realiza la transmisión de datos de forma unidireccional desde la salida del máster a la entrada del esclavo. El dispositivo máster pone los datos sobre la línea MOSI medio ciclo antes del final del flanco de alta impedancia para evitar interferencias.

• SCLK, Es el reloj del bus con el que los dispositivos sincronizaran el flujo de datos a través de las líneas. Se pueden configurar 2 parámetros que definen 4 modos de sincronización.

Estos parámetros son:

- CPOL (Clock polarity): Determina si el estado IDLE de la línea SCLK es en nivel bajo (CPOL = 0) o en nivel alto (CPOL = 1). No tiene efecto significativo en el formato de transferencia.
- CPHA (Clock Phase): Determina en cual flanco del reloj los datos son leídos o escritos. Si CPHA = 0 los datos sobre la línea MOSI son puestos en el primer flanco de reloj y los datos sobre la línea MISO son leídos en el segundo flanco de reloj. Cuando CPHA = 1 sucede lo contrario, la transferencia de datos sucede en el segundo flanco de reloj. Por lo tanto, según como se combinen estos dos parámetros, tendremos 4 modos ´ diferentes de trabajo. Es muy importante que todos los dispositivos dentro del bus trabajen con el mismo modo.
- CS o SS, Chip select o Slave Select. Los dispositivos en SPI no se seleccionan por software utilizando direcciones, como ocurre en I2C. En este protocolo es necesario utilizar una línea más llamada Chip Select, (CS) o Select Slave (SS). De forma ´que, si estamos utilizando 3 dispositivos, el bus estará compuesto por 2 hilos de transmisión de datos, 1 de reloj y 3 de CS.

Cuando se establece la comunicación, el bus solo puede ser ocupado por un Master y un Esclavo. Cualquier dispositivo que no haya sido seleccionado deberá deshabilitar se por medio del chip select para evitar interferencia.

La selección de los dispositivos es muy sencilla: El máster pone a nivel lógico bajo ’0’ el chip select del dispositivo en cuestión. El resto pasan a modo de alta impedancia para no interferir la comunicación.

A partir de este momento se inicia la comunicación entre los dispositivos. Es algo parecido a la condición de inicio de I2C. Una vez seleccionados los dispositivos, los datos pueden ser transferidos hasta una velocidad máxima
de varios MHz (Mbps).” [10], se muestra a continuación en la figura 23, el diagrama de bloques de las conexiones del maestro esclavo en el protocolo SPI.

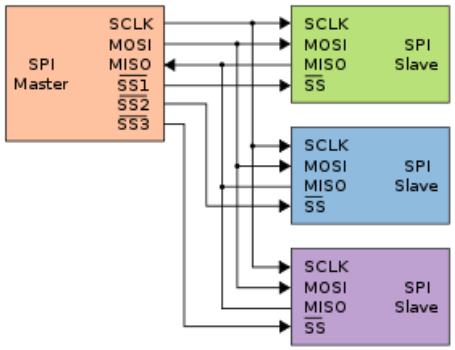

Figura 23. Conexiones SPI maestro y SPI esclavo.

1.3.8 Modulo GSM/GPRS SIM800L:

El módulo SIM800L es un dispositivo quad-band GSM/GPRS, trabaja en las frecuencias GSM850MHz, EGSM900MHz, DCS1800MHz y PCS1900MHz. Este módulo de telefonía celular que permite el uso de voz, texto, datos y SMS de texto, el dispositivo cuenta con las siguientes características:

- **Voltaje de Operación:** 3.7V ~ 4.4V DC (RECOMENDABLES).
- **Nivel lógico:** de 3V a 5V.
- **Consumo de corriente (Max):** 500 mA.
- **Consumo de corriente (sleep):** 0.7 mA.
- **interfaz:** Serial UART.
- **Quad-band** 850/900/1800/1900MHz – se conectan a cualquier red mundial GSM con cualquier SIM 2G.
- Trabaja solo con tecnología 2G.
- Enviar y recibir mensajes SMS.
- Enviar y recibir datos GPRS (TCP/IP, HTTP, entre otros protocolos).
- Receptor FM.

- Controlado por comandos AT (3GPP TS 27.007, 27.005 y SIMCOM enhanced AT Commands).
- Interfaz de comandos AT con detección “automática” de velocidad de transmisión.
- Soporta A-GPS.

- **Datos GPRS:**
  - Velocidad máxima de transmisión 85.6 Kbps.
  - Protocolo TCP/IP en chip.
  - Codificación: CS-1, CS-2, CS-3 y CS-4.
- Soporta USSD.
- Soporta Reloj en tiempo real (RTC).
- Velocidades de transmisión serial desde 1200bps hasta 115200bps.
- **Tamaño de la SIM:** Micro SIM.

Figura 24. Módulo GSM/GPRS SIM800L.

EL módulo presentado en la figura 24, utiliza el protocolo de comunicación UART o también conocida como comunicación serial.

Un puerto Serial es un módulo de comunicación digital para un sistema embebido. Es decir, permite la comunicación entre dos dispositivos digitales. Cuenta con dos conexiones, RX y TX. Lo que nos indica los modos de comunicación que puede manejar, Full-dúplex, Dúplex y Simplex. Además, podemos considerar como su principal ventaja a la sencillez de su protocolo de

comunicación. Sin embargo, también tiene desventajas como que sólo se puede comunicar a un puerto dos dispositivos.

Full dúplex. Significa que puede recibir y enviar información digital simultáneamente.

Dúplex o Half-dúplex. Es cuando sólo podemos transmitir o recibir información, una cosa a la vez.

Simplex. Cuando sólo podemos ya sea recibir o transmitir.

La función principal de un puerto serial, es la de empacar y desempacar paquetes de datos binarios seriales. Como resultado, la serialización significa convertir un dato paralelo (byte) a un conjunto de pulsos seriales que puedan ser recibidos y enviados por una línea de transmisión. En primer lugar, el protocolo serial opera mediante tres condiciones digitales básicas: inicio de transmisión (IT), paridad (P) y fin de transmisión (FT). Estas condiciones son sincronizadas mediante un oscilador interno. El generador permite controlar la velocidad del puerto serial. Por lo tanto, la velocidad se mide en BAUD 's." [11]

COMANDOS AT: Una vez establecida la comunicación UART, el lenguaje por así decirlo, con el que el dispositivo atiende las solicitudes y peticiones es, a través de los comandos AT, es decir, con estos, se logra la configuración del módulo para, bien sea, enviar mensajes de texto, realizar llamadas o hacer una conexión a internet; la cual es ésta última, la de principal interés para este proyecto, ya que, es éste el medio de transmisión por el cual se envían los datos de todos los sensores, siendo más precisos, a través de una conexión TCP/IP por el protocolo HTTP haciendo una petición GET, dicho esto, se debe tener presente cuales son los comandos con los que se puede hacer la anterior configuración.

La idea es configurar el módulo y hacer efectiva una conexión TCP/IP a un servidor, luego que la conexión haya sido exitosa, se procede a enviar una trama de configuración con la que se identifica el protocolo HTTP junto con una petición GET, posteriormente, es posible enviarle toda la información recolectada de los sensores, al servidor que procesara los datos a fin de almacenarlos, graficarlos y posterior mente visualizarlos, por ejemplo, el servidor con el cual se realizaron las pruebas fue "api.thingspeak.com", para el cual solo se necesita crearse un usuario

y contraseña para poder usarlo. Los comandos AT para realizar la configuración antes dicha, se presentan a continuación:

Gramática	Descripción
AT+CIPSHUT Respuesta esperada: **SHUT OK**	Desactiva el contexto PDP de GPRS, respuesta “SHUT OK ò ERROR”. Tiempo máximo de respuesta 65 segundos.
AT+CPIN? Respuesta esperada: **+CPIN: READY** **OK**	Verifica si se requiere alguna contraseña o no. Respuesta “+CPIN: READY” y “OK”
AT+CSQ Respuesta esperada: **+CSQ: 11,0** **OK**	Consulta el nivel de señal recibida. Respuesta “+CSQ: (Nivel)” Y “OK”.
AT+CREG? Respuesta esperada: **+CREG: 0,1** **OK**	Muestra si la red ha indicado actualmente el registro del ME. Respuesta “+CREG: 0,1” y “OK” Si el dispositivo no está registrado, responde “+CME ERROR: <err>”.
AT+CIPMUX=0 Respuesta esperada:	Inicia una conexión IP simple, respuesta “OK”.

OK	
AT+CGATT=1 Respuesta esperada: **OK**	Conecta o desconecta del servicio GPRS, 1 para conectarse del servicio GPRS, respuesta “OK”. Tiempo máximo de respuesta 10 segundos.
AT+CSTT="web.colombiamovil .com.co","Tigo Web","" Respuesta esperada: **OK**	Inicia la tarea y configure APN, nombre de usuario, contraseña. Después de que se ejecute este comando, el estado cambiará a IP START. Respuesta “OK”.
AT+CIICR Respuesta esperada: **OK**	Abre la conexión inalámbrica (GPRS o CSD), respuesta “OK”. Tiempo de respuesta máximo 85 segundos. Después de que el módulo acepta la operación activada, si se activa con éxito, el estado del módulo cambiará a IP GPRSACT, y responde OK, de lo contrario responderá ERROR.
AT+CIFSR Respuesta esperada: **100.114.104.250**	Obtiene una dirección IP local, respuesta <IP address> un parámetro de cadena que indica la dirección IP asignada desde GPRS o CSD. Solo después de que se activa el contexto PDP, AT + CIFSR puede

	obtener la dirección IP local; de lo contrario, responderá ERROR.
AT+CIPSTART="tcp","api.thingspeak.com","80" Respuesta esperada: **OK** **CONNECT OK**	Inicia una conexión TCP o UDP. Respuesta “OK” y “CONNECT OK”.
AT+CIPSEND Respuesta esperada: **>**	Enviar datos a través de una conexión TCP o UDP. Respuesta “>”.
GET https://api.thingspeak.com/update?api_key=VQOHZSPLH9SX0O8M&field1=13.750&field2=5.510&field3=0.516899&field4=31.97&field5=69.29 HTTP/1.1 **Host: 184.106.153.149** **Connection: close** Respuesta esperada: **SEND OK** **HTTP/1.1 200 OK** **Date: Mon, 31 May 2021 23:28:05 GMT** **Content-Type: text/plain; charset=utf-8**	El usuario debe escribir datos solo después de que el módulo responda con esta marca “>", y luego use CTRL+Z (0x1a) para enviar. Respuesta del módulo “SEND OK”, después de este mensaje se espera la trama de confirmación del servidor, con la que se indica si se recibieron los datos exitosamente.

Content-Length: 2
Connection: close
Status: 200 OK

Tabla 1. Tabla de comandos AT para una conexión TCP/IP.

La tabla 1, representa la serie de comandos AT con los que se logra configurar en el módulo SIM800L, una conexión TCP/IP con el protocolo HTTP por el método GET, en la columna de gramática se encuentran todos los comandos AT en negrita y sus respectivas respuestas en azul, en la columna de descripción se explica cómo funciona cada comando, las posibles respuestas y en algunos casos, el tiempo máximo de respuesta, si se desea tener más información se puede consultar en el Datasheet del módulo, series de aplicaciones TCP/IP [13] y el manual de series de comandos AT [14] del mismo.

Con respecto a la cadena de transmisión, los datos que se desean procesar y almacenar en el servidor “api.thingspeak.com” se envían en las variables (field) seguidas del número que indica la posición de la gráfica que es visualizada en el servidor y separadas por el símbolo ampersand “&”. Es posible transmitir máximo 8 datos.

Un aspecto importante a tener en cuenta es, el tema de la cobertura de la red 2g en Colombia, lo cual podemos ver en la siguiente figura.

Figura 25. Mapeo de cobertura de redes en Colombia [15]

Como se puede observar en la figura 25, afortunadamente en Colombia se puede contar con esta conexión por lo menos un par de años, pero queda claro que con el paso del tiempo y con el avance de la tecnología día tras día esta quedará obsoleta.

1.3.9 Microcontrolador de 32 bits ESP32:

La ESP32 es un microcontrolador de 32bits con una gran capacidad de procesamiento y diseñado para trabajo robusto, lo que lo hace un dispositivo confiable para trabajar en diferentes ambientes.

“ESP32 es la denominación de una familia de chips SoC de bajo costo y consumo de energía, con tecnología Wi-Fi y Bluetooth de modo dual integrada. El ESP32 emplea un microprocesador Tensilica Xtensa LX6 en sus variantes de simple y doble núcleo e incluye interruptores de antena, balun de radiofrecuencia, amplificador de potencia, amplificador receptor de bajo ruido, filtros, y módulos de administración de energía. El ESP32 fue creado y desarrollado por Espressif Systems y es fabricado por TSMC utilizando su proceso de 40 nm.” [12]

Sus características son:

- **Procesador:**
  - **CPU:** Microprocesador de 32-bit Xtensa LX6 de doble núcleo (o de un solo núcleo), operando a 160 o 240 MHz y rindiendo hasta 600 DMIPS.
  - Coprocesador de ultra baja energía (ULP).
- **Memora:** 520 KiB SRAM.
- **Conectividad inalámbrica:**
  - **Wi-Fi:** 802.11 b/g/n.
  - **Bluetooth:** v4.2 BR/EDR y BLE.
- **Interfaces periféricas:**
  - 12-bit SAR ADC de hasta 18 canales.
  - 2 × 8-bit DACs.
  - 10 × sensores de tacto (sensores capacitivos GPIOs).
  - 4 × SPI.
  - 2 × interfaces I^2S.
  - 2 × interfaces I^2C.
  - 3 × UART.
  - Controlador host SD/SDIO/CE-ATA/MMC/eMMC.
  - Controlador esclavo SDIO/SPI.
  - Interfaz Ethernet MAC con DMA dedicado y soporte para el protocolo IEEE 1588 Precision Time Protocol.
  - Bus CAN 2.0.
  - Controlador remoto infrarrojo (TX/RX, hasta 8 canales).
  - Motor PWM.
  - LED PWM (hasta 16 canales).
  - Sensor de efecto Hall.
  - Preamplificador analógico de ultra baja potencia.
- **Seguridad:**
  - Soporta todas las características de seguridad estándar de IEEE 802.11, incluyendo WFA, WPA/WPA2 y WAPI.
  - Arranque seguro.

- Cifrado flash.
- 1024-bit OTP, hasta 768-bit para clientes.
- Criptografía acelerada por hardware: AES, SHA-2, RSA, criptografía de curva elíptica (ECC), generador de números aleatorios (RNG).

- **Administración de energía:**
  - Regulador interno de baja caída.
  - Dominio de poder individual para RTC.
  - Corriente de 5μA en modo de suspensión profundo.
  - Despierta por interrupción de GPIO, temporizador, medidas de ADC, interrupción por sensor de tacto capacitivo.

Figura 26. Módulo ESP32.

El dispositivo mostrado en la figura 26, es el microcontrolador descrito por las anteriores características.

1.4 Diseño del circuito impreso:

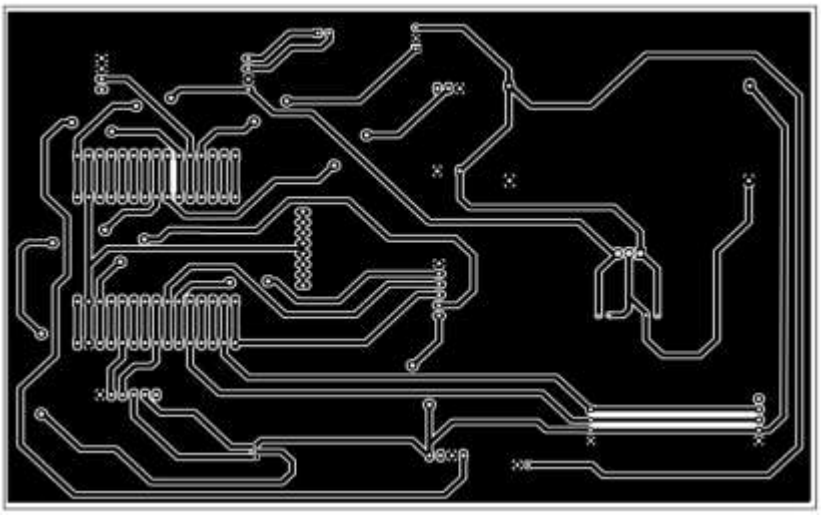

Figura 27. Diseño PCB del sistema

En la figura 27, se presenta el diseño impreso del sistema de adquisición y transmisión de datos, el cual fue realizado en el software libre EasyEDA que es una herramienta basada en la web que permite diseñar, simular, compartir y desarrollar placas de circuito impreso.

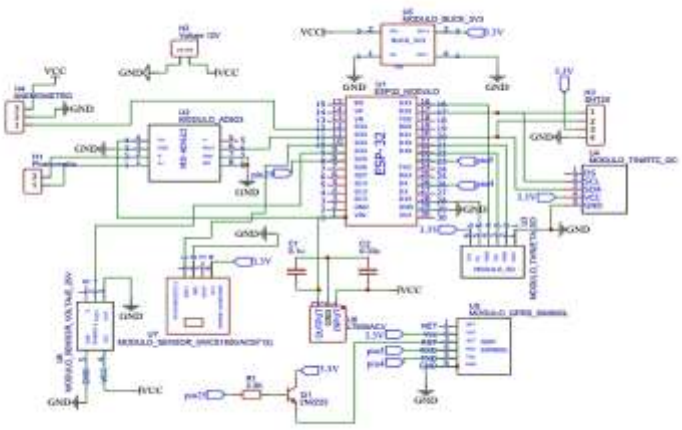

Figura 28. Diagrama de bloques del sistema de adquisición y transmisión de datos.

El esquema mostrado en la figura 28, es el diagrama de conexiones del sistema de adquisición y transmisión de la información, el cual es necesario para facilitar el diseño de la placa impresa, ya que el programa una vez realizado el diagrama de bloques, da una opción que permite generar una pestaña que señala las conexiones que hay que hacer; consta solo de ir uniendo los puntos en toda la placa de manera sencilla, reduciendo la posibilidad de error, hay que tener presente que, hay componentes que no se encuentran en el programa, por tal motivo hay que crearlos, como por ejemplo el módulo de la ESP32, el amplificador de instrumentación AD623 y el SIM800L.

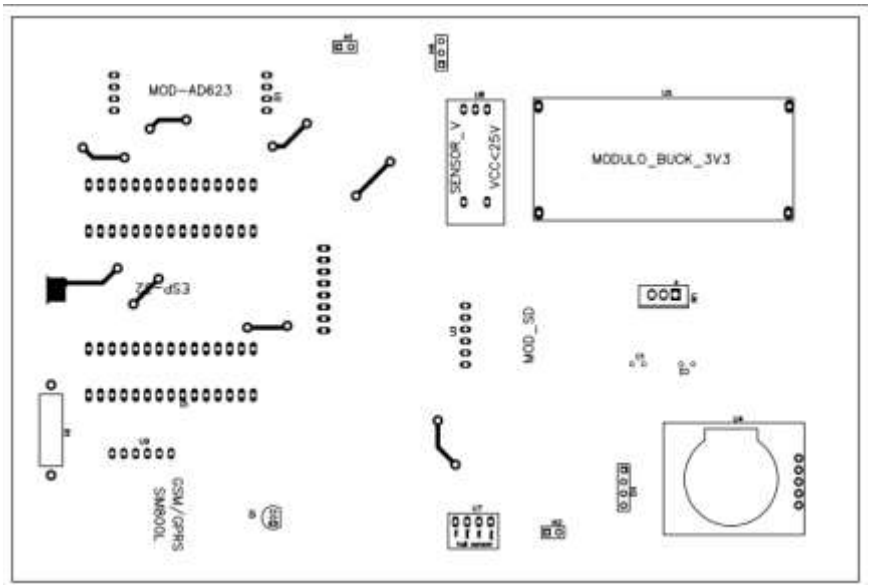

Figura 29. Serigrafía de la parte superior de la placa.

En la figura 29, se puede identificar la serigrafía de la parte superior, que es la que identifica la posición en la que van los componentes y en este caso, también permite visualizar los puntos que necesitan puentearse.

Un aspecto a tener en cuenta es que se removió el transistor 2N2222, que conectaba el reset del módulo SIM800L al pin 25 de la ESP32, porque el uso de este no es necesario, simplemente se conectó con un puente en donde sería la base y emisor del transistor.

1.5 Funcionamiento del software

El funcionamiento del dispositivo consta de varias tareas que se van realizando algunas de manera simultánea y otras que depende de que ciertas condiciones se cumplan como se observa en la figura 30.

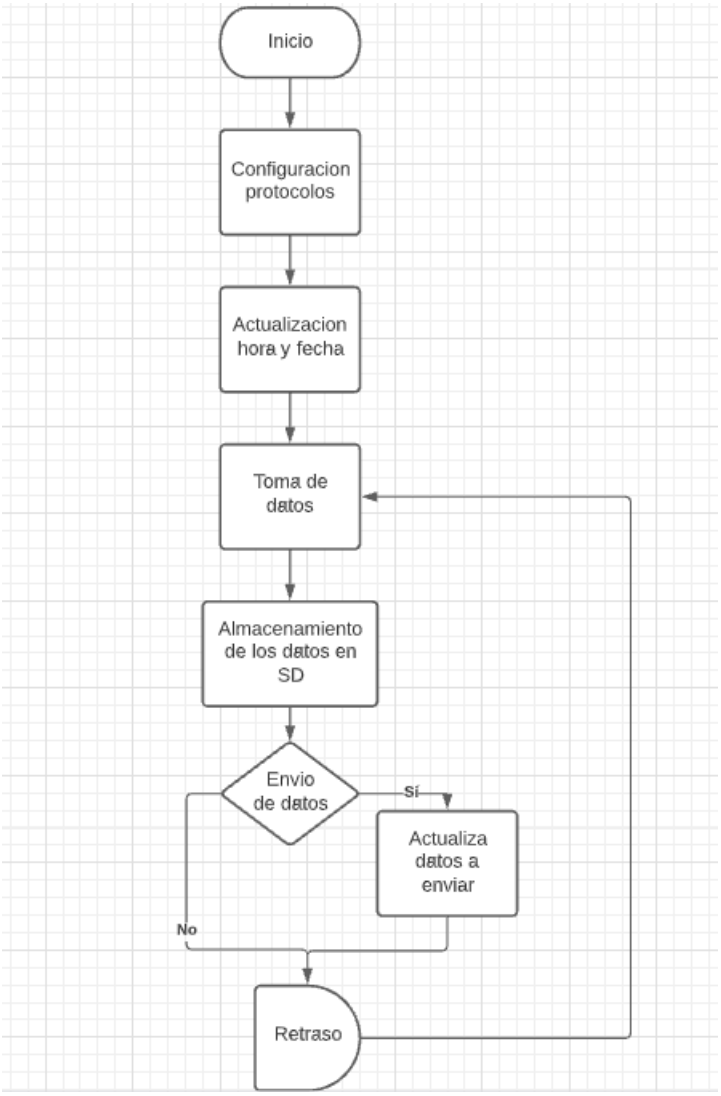

Figura 30. Diagrama de flujo del funcionamiento del software.

Podemos resumir el funcionamiento del hardware en cinco tareas, que son:

- **GPRS:** En primera instancia el dispositivo hace un reseteo del módulo GPRS y prepara los comandos para la configuración y conectividad para garantizar el correcto funcionamiento del módulo.

- **Actualización de hora y fecha:** La siguiente tarea que se ejecuta es la encargada de actualizar la hora y fecha para tener estos datos actualizados a la hora de tomar las diferentes mediciones, esta tarea adquiere la hora y fecha que se encuentra actualmente en el módulo DS3231 y la almacena en una variable.

  Luego de que la hora es almacenada, es utilizada en otra tarea que se ejecuta una vez por segundo, actualizando esta variable y llevar el respectivo control del tiempo.

  La función que actualiza la hora del módulo DS3231 se ejecuta cuna vez cada veinticuatro horas con el fin de garantizar una mejor precisión de la hora y la fecha.

- **Toma de datos y almacenamiento:** En esta tarea se ejecutan las diferentes funciones para cada dispositivo de medición, recordemos que los dispositivos tienen protocolos diferentes a la hora de realizar la respectiva toma de datos, como el I2C y UART.

  Estas mediciones son almacenadas en sus respectivas variables y luego anexadas a una trama de datos para realizar su respectivo almacenamiento, en esta trama se incluye los nombres y valores de las mediciones además de la hora y fecha en la que fueron tomadas.

  Esta trama va almacenada en dos archivos diferente, un archivo que se utiliza para tener un pequeño back up en caso de pérdida de datos y en otro archivo que es el utilizado para garantizar de que todos los datos tomados son enviados correctamente.

1.4.9 **Envío de datos:** La tarea de envío de datos se ejecuta una vez que el dispositivo ya ha tomado los datos correspondientes, esta función realiza la lectura de un archivo en el cual se almacenan los datos a publicar y realiza a través del protocolo TCP-IP el envío de la trama de datos almacenada en el archivo, cuando el envío es realizado esta espera una

confirmación de recepción, cuando esta confirmación es recibida procede a eliminar del archivo la trama de datos correspondiente ya que esta fue enviado de manera exitosa.

De esta manera se garantiza en todo momento que todos las mediciones realizadas son enviadas correctamente, si llegase a ocurrir que el sistema pierde la conexión con el servidor y esta función no recibe la respectiva confirmación mantiene el dato en lista para enviar, como esta función se ejecuta de manera periódica cuando la conexión sea restablecida todos los datos que estén en lista serán enviados, teniendo en cuenta que la trama de datos incluye la hora y la fecha en las que la medición fue realizada no existe inconveniente mayor para el envío de dichos datos en un momento posterior.

2 Resultados

Durante el desarrollo de este proyecto de investigación, se diseñó e implemento un sistema de detección de variables climáticas, con el fin de que este sea la base para futuros proyectos enfocados en el área de las energías renovables. Basado en los objetivos logramos de manera satisfactoria el desarrollo del dispositivo planteado como base para el fin de este proyecto.

A pesar de tener ciertos problemas enfocados en la transmisión de datos, causados por bajo nivel de señal recibida en el módulo SIM800L, ya que, por motivos de la pandemia, las pruebas se realizaron desde casa y el lugar donde se realizaron se encontraba rodeado de muchos edificios, todo eso sumando a que la cobertura 2g, poco a poco se reduce, problemas que se pudieron solucionar, buscando un sector cercano un poco más despejado, para evitar interferencias en la comunicación.

La manera en que estuvo dirigido el proyecto hizo posible que se obtuviera un producto con las siguientes características:

- Independencia energética debido al diseño para alimentación energética del dispositivo basada en energía solar, garantiza la continuidad del funcionamiento sin depender de la red eléctrica.
- Bajo consumo energético, dado que se desarrolló en un microcontrolador de 32 bits, nos permite de realizar de manera eficiente un mayor control del consumo de nuestro dispositivo, además los sensores utilizados son de bajo consumo lo que garantiza que el sistema tenga una autonomía mayor.
- Facilidad de envió de datos, teniendo en cuenta que el método de envió de datos utilizado es a través de la tecnología GPRS la cual cuenta con una amplia cobertura nos da una mayor garantía para el vio de datos.
- Portable y de fácil instalación, debido a las características anteriormente mencionadas y que el dispositivo es de tamaño pequeño es realmente sencillo instalarlo en el lugar donde se desea realizar la medición de las variables.

- Confiable, gracias a la arquitectura del código y que cuenta con un almacenamiento de respaldo para los datos el dispositivo es bastante confiable a la hora de garantizar que todas las muestras tomadas sean enviadas.
- Practico, al ser modular es fácil realizan el remplazo o el diagnostico de los instrumentos utilizados además de que es posible cambiar los sensores (teniendo en cuenta los respectivos cambios en el código) dependiendo a las necesidades del lugar por lo cual hace muy práctico su uso para las diferentes situaciones.

En resumen, el dispositivo permite de manera satisfactoria:

- Realizar la medición de las distintas variables planteadas en los objetivos del proyecto.
- Almacenar estos datos en un respaldo con su respectiva hora y fecha como medio de seguridad para evitar la pérdida de información.
- Realizar el envío de estos datos a una base de datos de elección en este caso utilizando http.

Figura 31. Montaje del prototipo de monitoreo remoto de variables climáticas (exterior).

Figura 32. Montaje del prototipo de monitoreo remoto de variables climáticas (interior).

En la figura 31, se visualiza el sistema en la parte externa con todos los sensores conectados que monitorean las 4 variables de interés, que son, velocidad del viento, temperatura, humedad e intensidad de radiación solar. En la figura 32, se puede observar la placa PCB con sus distintos componentes conectados, como los son: el módulo SIM800L, el amplificador AD623, la ESP32, el módulo de la SD, el módulo del reloj y para poder conocer el consumo energético del sistema, el sensor de corriente y el de voltaje, que se usaron para calcular la potencia.

A continuación, se presentan evidencias de los datos transmitidos al servidor, los cuales han sido almacenados y procesados en distintas gráficas, con ellas se puede observar detalladamente el comportamiento de cada una de las variables a lo largo del tiempo, con el registro de fecha y hora:

Figura 33. Graficas del comportamiento del voltaje y velocidad de viento.

En la figura 33, se puede observar las gráficas del comportamiento del sensor de voltaje el cual da las unidades en voltios (V), y del anemómetro que es el sensor de velocidad del viento, y su unidad es en metros por segundo (m/s)

Figura 34. Graficas del comportamiento de la corriente y temperatura.

En la figura 34, se encuentran graficados los valores que obtiene el sensor de corriente de efecto hall WCS1800, que tiene como unidad el amperio (A) en especial amperios DC, debido a que se mide corriente continua, y también podemos observar la variable temperatura del sensor SHT20 que entrega la medida en grados Celsius (°C)

Figura 35. Graficas del comportamiento de la humedad e intensidad de radiación solar.

En la figura 35, está presente el comportamiento de la humedad, que mide el sensor SHT20, mencionado anteriormente, y su unidad está dada en porcentaje (%), también se encuentra el comportamiento del piranómetro que mide la intensidad de radiación solar, que se da en W por metro cuadrado (W/m^2).

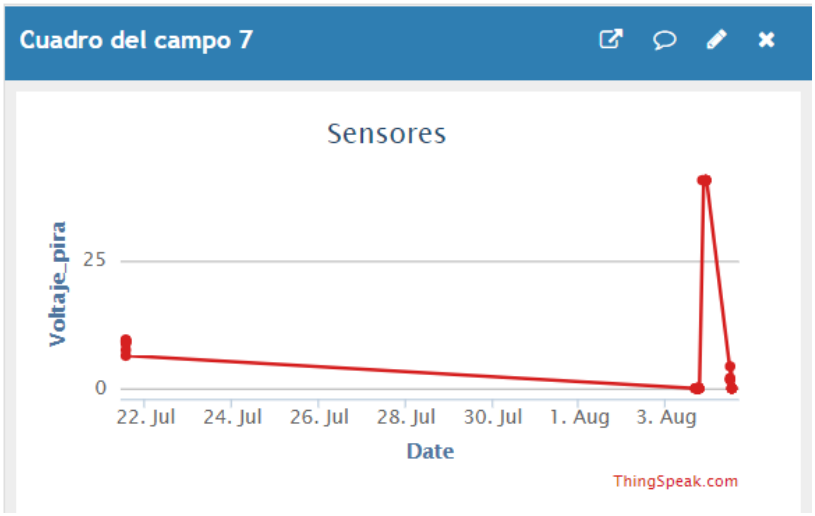

Figura 36. Graficas del comportamiento del voltaje del piranómetro.

Por último, se tiene en la figura 36, la gráfica del voltaje que representa la medida del piranómetro, claramente sus unidades son en voltios (V), esta variable es importante tenerla presente, para efectos de calibración de la medida de dicho sensor.

3 Conclusión

Queda evidenciado en éste trabajo, la gran ventaja que presenta, la implementación de sistemas embebidos y sus distintos periféricos, en el desarrollo del sistema de recopilación de datos climáticos, teniendo en cuenta qué; fue posible adquirir dichos datos de manera remota, a una distancia considerable, permitiendo con ello, llevar a cabo un sinnúmero de aplicaciones tales como operación, control y en este caso, el análisis de una zona que cumpla con las condiciones necesarias para implementar un sistema de energías renovables.

Por otra parte, lo antes mencionado, deja como principal ventaja, la recolección de información en una plataforma digital para procesamiento de datos, de forma, que deje abierta, la oportunidad de implementar proyectos a futuro, que comprendan la realización de estaciones de energías renovables híbridas, análisis de datos y optimización.

En conclusión, el desarrollo de sistemas embebidos para el monitoreo de variables climáticas, son una alternativa viable que aporta grandes beneficios como, su adaptabilidad y la capacidad de personalizar las distintas funciones con respecto a las necesidades en cuestión.

Estos beneficios facilitan su implementación en los diferentes ambientes y la selección de sensores dependiendo de las necesidades finales, es una gran ventaja debido a su adaptabilidad y bajo coste, ya que permite mayor liberta en la compra de los dispositivos, además, también brinda facilidad a la hora de replicar los dispositivos, intercambiar sensores segundos la necesidad previa, e incluso hacer actualizaciones en software o en hardware.

El uso de la rede 2g, es una herramienta muy útil a la hora de implementar proyectos de transmisión de datos, en especial, en sitios en donde no se tiene conectividad de wifi, solo basta con que haya cobertura de la red en el lugar en donde se desee realizar la implementación, pero como pudimos observar en el mapa de redes en Colombia [15], poco a poco la cobertura va reduciendo con el paso del tiempo y el avance de la tecnología, por lo que se debería ir optando por otros módulos y dispositivos que presenten otro tipos de conexiones y redes como es el caso de las 3g y 4g.

Anexo: Códigos sensores

Código sensor de temperatura SHT20:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "freertos/semphr.h"
#include "freertos/queue.h"

#include "esp_wifi.h"
#include "esp_system.h"
#include "esp_event.h"
#include "esp_log.h"
#include "esp_event_loop.h"
#include "nvs_flash.h"
#include "esp_vfs_fat.h"
#include "esp_heap_caps.h"

#include "driver/gpio.h"
#include "driver/adc.h"
#include "esp_adc_cal.h"

#include "driver/uart.h"
#include "soc/uart_struct.h"

#include "driver/sdmmc_host.h"
#include "driver/sdspi_host.h"
#include "sdmmc_cmd.h"
```

```
#include "lwip/netdb.h"
#include "lwip/sockets.h"
#include "driver/i2c.h"

//#include "DS1307.h"
//#include "UnixTime.h"

#define TIME_MUESTRA 60

#define PANTALLA 0
#define CLK_CONFIG 0
#define GPRS 1
#define SAVE_SD 1

#define lectura 1
#define escritura 1

#define SDA 21
#define SCL 22
#define FREC 100000
#define I2C_PORT I2C_NUM_1

#define PIN_NUM_MISO 19
#define PIN_NUM_MOSI 23
#define PIN_NUM_CLK 18
#define PIN_NUM_CS_SD 15

#define ACK_CHECK_EN 0x01
#define ACK_CHECK_DIS 0x00
#define TEMPERATURE_DIREC 0XF3
#define HUMEDAD_DIREC 0XF5
```

```
#define TAM_COLA 20
#define TAM_MSG_DATA 180
#define TAM_COLA_S 5
#define TAM_MSG_DATA_S 20

#define TIME_WAIT 1000

#define NUM_MUESTRAS 500
#define V_REF 1099

#define TXD_PIN GPIO_NUM_10
#define RXD_PIN GPIO_NUM_9

#define DESTINO_DATOS 0

#define SPI_BUS TFT_VSPI_HOST


//variables para sensor de temperatura: SHT20
float Hum=0,Tem=0;
//--------------------


void i2c_init(){
        int i2c_master_port=I2C_PORT;
        i2c_config_t config_i2c;
        config_i2c.mode=I2C_MODE_MASTER;
        config_i2c.scl_io_num=SCL;
        config_i2c.sda_io_num=SDA;
        config_i2c.master.clk_speed=FREC;
        config_i2c.scl_pullup_en=GPIO_PULLUP_ENABLE;
        config_i2c.sda_pullup_en=GPIO_PULLUP_ENABLE;
```

```
        if(i2c_param_config(i2c_master_port,        &config_i2c)==ESP_OK)printf("Param
Config OK\r\n");
        if(i2c_driver_install(i2c_master_port,
config_i2c.mode,0,0,0)==ESP_OK)printf("Driver Install OK\r\n");
}


//---------------------FUNCION PARA SHT20-----------------------------------
uint16_t GetDataSensor(uint8_t sen){
        uint8_t dato[7];
        uint8_t direc=0x40;
        i2c_cmd_handle_t cmd;

        cmd = i2c_cmd_link_create();
        i2c_master_start(cmd);
        i2c_master_write_byte(cmd,  (  direc  <<  1  )  |  I2C_MASTER_WRITE,
ACK_CHECK_EN);
        i2c_master_write_byte(cmd, sen, ACK_CHECK_EN);

        i2c_master_stop(cmd);
        i2c_master_cmd_begin(I2C_PORT, cmd, 2000 / portTICK_RATE_MS);
        i2c_cmd_link_delete(cmd);

        vTaskDelay(150);

        cmd = i2c_cmd_link_create();
        i2c_master_start(cmd);
        i2c_master_write_byte(cmd,  (  direc  <<  1  )  |  I2C_MASTER_READ,
ACK_CHECK_EN);
        i2c_master_read(cmd, dato,3,I2C_MASTER_LAST_NACK);
        i2c_master_stop(cmd);
        i2c_master_cmd_begin(I2C_PORT, cmd, 1000 / portTICK_RATE_MS);
        i2c_cmd_link_delete(cmd);
```

```
        if(sen==HUMEDAD_DIREC)
                return(((int)(dato[0]<<8))|((int)(dato[1]&0xFC)));
        if(sen==TEMPERATURE_DIREC)
                return(((int)(dato[0]<<8))|((int)(dato[1]&0xFC)));
        return 0;
}


void sensor(){
float corriente=0,voltaje_adc;
  // uint32_t voltaje_adc;
   //int j=0;

while(1){

        Hum = (-6 + (125 * (GetDataSensor(HUMEDAD_DIREC) / 65536.0)));
                        Tem = (-46.85 + (175.72 *
(GetDataSensor(TEMPERATURE_DIREC)/ 65536.0)));

                        if (Hum<0)
                                Hum = 0;

                        if (Tem<0)
                                Tem = 0;


//printf("CORRIENTE:%0.6f\r\n VOLT_ADC:%0.6f\r\n----------\r\n",corriente,voltaje_adc);

 printf("Temperatura: %0.2f\r\n Humedad:%0.2f\r\n----------\r\n",Tem,Hum);
vTaskDelay(1000/portTICK_PERIOD_MS);
}
}
```

```
void app_main(){
        i2c_init();

xTaskCreate(&sensor,"sensor",20000,NULL,1,NULL);
}
```

Código sensor de Voltaje y corriente:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "freertos/semphr.h"
#include "freertos/queue.h"

#include "esp_wifi.h"
#include "esp_system.h"
#include "esp_event.h"
#include "esp_log.h"
#include "esp_event_loop.h"
#include "nvs_flash.h"
#include "esp_vfs_fat.h"
#include "esp_heap_caps.h"

#include "driver/gpio.h"
#include "driver/adc.h"
#include "esp_adc_cal.h"

#include "driver/uart.h"
#include "soc/uart_struct.h"

#include "driver/sdmmc_host.h"
#include "driver/sdspi_host.h"
#include "sdmmc_cmd.h"

#include "lwip/netdb.h"
```

```
#include "lwip/sockets.h"
#include "driver/i2c.h"

//#include "DS1307.h"
//#include "UnixTime.h"

#define TIME_MUESTRA 60

#define PANTALLA 0
#define CLK_CONFIG 0
#define GPRS 1
#define SAVE_SD 1

#define lectura 1
#define escritura 1

#define SDA 21
#define SCL 22
#define FREC 100000
#define I2C_PORT I2C_NUM_1

#define PIN_NUM_MISO 19
#define PIN_NUM_MOSI 23
#define PIN_NUM_CLK 18
#define PIN_NUM_CS_SD 15

#define ACK_CHECK_EN 0x01
#define ACK_CHECK_DIS 0x00
#define TEMPERATURE_DIREC 0XF3
#define HUMEDAD_DIREC 0XF5

#define TAM_COLA 20
#define TAM_MSG_DATA 180
```

```
#define TAM_COLA_S 5
#define TAM_MSG_DATA_S 20

#define TIME_WAIT 1000

#define NUM_MUESTRAS 500
#define V_REF 1099

#define TXD_PIN GPIO_NUM_10
#define RXD_PIN GPIO_NUM_9

#define DESTINO_DATOS 0

#define SPI_BUS TFT_VSPI_HOST


//--------------------LECTURA ANALOGA-------------------------------------
float analog_lec(adc1_channel_t canal, adc_bits_width_t resolucion, adc_atten_t atenuacion){
        static float muestra=0;
        static long suma=0;
        static int i=0;
        static esp_adc_cal_characteristics_t characteristics;

        adc1_config_width(resolucion);
        adc1_config_channel_atten(canal,atenuacion);
        esp_adc_cal_get_characteristics(V_REF, atenuacion, resolucion, &characteristics);

   suma=0;
        for(i=0;i<NUM_MUESTRAS;i++){
                //esp_adc_cal_get_voltage(canal, &characteristics,&p1);
                suma+=adc1_to_voltage(canal, &characteristics);
```

```
        }

        muestra=suma/NUM_MUESTRAS;

        return muestra;
}


//---------------------------------------------------------------------------


void sensor(){
float corriente=0,voltaje_adc;
  // uint32_t voltaje_adc;
   //int j=0;

while(1){

//-------sensor de corriente

        voltaje_adc=analog_lec(ADC1_CHANNEL_4,ADC_WIDTH_BIT_12,ADC_ATTE
N_DB_11);
                voltaje_adc = voltaje_adc/1000.00000;

                //WCS1700:
                //corriente= (voltaje_adc-1.7323)/0.0269;

                //condicion para corregir error :

                if(voltaje_adc < vcc/2){
                        voltaje_adc = voltaje_adc + ((vcc/2) - voltaje_adc);
                }
```

```
                corriente= (voltaje_adc-(vcc/2))/0.0269;

                if(corriente < 0){

                        corriente *= (-1);
                }
                corriente = corriente + 0.22;

                printf("        VOLT_ADC:%0.6f        CORRIENTE:%0.6f\r\n        ----------
\r\n",voltaje_adc,corriente);

//---------- sensor de voltaje:

        volta                                                                   =
analog_lec(ADC1_CHANNEL_5,ADC_WIDTH_BIT_12,ADC_ATTEN_DB_11);
        volta = (volta*5)/1000.0;

        printf(" Voltaje_bateria:%0.3f",volta);

vTaskDelay(1000/portTICK_PERIOD_MS);

}
}

void app_main(){


xTaskCreate(&sensor,"sensor",20000,NULL,1,NULL);
}
```

Código sensor anemómetro:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "freertos/semphr.h"
#include "freertos/queue.h"

#include "esp_wifi.h"
#include "esp_system.h"
#include "esp_event.h"
#include "esp_log.h"
#include "esp_event_loop.h"
#include "nvs_flash.h"
#include "esp_vfs_fat.h"
#include "esp_heap_caps.h"

#include "driver/gpio.h"
#include "driver/adc.h"
#include "esp_adc_cal.h"

#include "driver/uart.h"
#include "soc/uart_struct.h"

#include "driver/sdmmc_host.h"
#include "driver/sdspi_host.h"
#include "sdmmc_cmd.h"

#include "lwip/netdb.h"
```

```
#include "lwip/sockets.h"
#include "driver/i2c.h"

//#include "DS1307.h"
//#include "UnixTime.h"

#define TIME_MUESTRA 60

#define PANTALLA 0
#define CLK_CONFIG 0
#define GPRS 1
#define SAVE_SD 1

#define lectura 1
#define escritura 1

#define SDA 21
#define SCL 22
#define FREC 100000
#define I2C_PORT I2C_NUM_1

#define PIN_NUM_MISO 19
#define PIN_NUM_MOSI 23
#define PIN_NUM_CLK 18
#define PIN_NUM_CS_SD 15

#define ACK_CHECK_EN 0x01
#define ACK_CHECK_DIS 0x00
#define TEMPERATURE_DIREC 0XF3
#define HUMEDAD_DIREC 0XF5

#define TAM_COLA 20
#define TAM_MSG_DATA 180
```

```
#define TAM_COLA_S 5
#define TAM_MSG_DATA_S 20

#define TIME_WAIT 1000

#define NUM_MUESTRAS 500
#define V_REF 1099

#define TXD_PIN GPIO_NUM_10
#define RXD_PIN GPIO_NUM_9

#define DESTINO_DATOS 0

#define SPI_BUS TFT_VSPI_HOST


//--------------------LECTURA ANALOGA-----------------------------------
float analog_lec(adc1_channel_t canal, adc_bits_width_t resolucion, adc_atten_t atenuacion){
        static float muestra=0;
        static long suma=0;
        static int i=0;
        static esp_adc_cal_characteristics_t characteristics;

        adc1_config_width(resolucion);
        adc1_config_channel_atten(canal,atenuacion);
        esp_adc_cal_get_characteristics(V_REF, atenuacion, resolucion, &characteristics);

   suma=0;
        for(i=0;i<NUM_MUESTRAS;i++){
                //esp_adc_cal_get_voltage(canal, &characteristics,&p1);
                suma+=adc1_to_voltage(canal, &characteristics);
```

```
        }

        muestra=suma/NUM_MUESTRAS;

        return muestra;
}


//------------------------------------------------------------------------
void sensor(){


float viento=0;
uint32_t voltaje_adc_ch6;

while(1){

//-------sensor anemómetro

        voltaje_adc_ch6=analog_lec(ADC1_CHANNEL_6,ADC_WIDTH_BIT_12,ADC_A
TTEN_DB_6);
                viento=(20.3*(voltaje_adc_ch6/1000.0))-8.7;
                if(viento<0)
                viento=0;

printf(" Velocidad viento:%0.3f",viento);

void app_main(){


xTaskCreate(&sensor,"sensor",20000,NULL,1,NULL);
}
```

Código tarjeta SD:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "freertos/semphr.h"
#include "freertos/queue.h"

#include "esp_wifi.h"
#include "esp_system.h"
#include "esp_event.h"
#include "esp_log.h"
#include "esp_event_loop.h"
#include "nvs_flash.h"
#include "esp_vfs_fat.h"
#include "esp_heap_caps.h"

#include "driver/gpio.h"
#include "driver/adc.h"
#include "esp_adc_cal.h"

#include "driver/uart.h"
#include "soc/uart_struct.h"

#include "driver/sdmmc_host.h"
#include "driver/sdspi_host.h"
#include "sdmmc_cmd.h"

#include "lwip/netdb.h"
#include "lwip/sockets.h"
#include "driver/i2c.h"
```

```
#include "DS1307.h"
#include "UnixTime.h"

#define TIME_MUESTRA 60

#define PANTALLA 0
#define CLK_CONFIG 0
#define GPRS 1
#define SAVE_SD 1

#define lectura 1
#define escritura 1

#define SDA 21
#define SCL 22
#define FREC 100000
#define I2C_PORT I2C_NUM_1

#define PIN_NUM_MISO 19
#define PIN_NUM_MOSI 23
#define PIN_NUM_CLK 18
#define PIN_NUM_CS_SD 15

#define ACK_CHECK_EN 0x01
#define ACK_CHECK_DIS 0x00
#define TEMPERATURE_DIREC 0XF3
#define HUMEDAD_DIREC 0XF5

#define TAM_COLA 20
#define TAM_MSG_DATA 180
#define TAM_COLA_S 5
#define TAM_MSG_DATA_S 20

#define TIME_WAIT 1000

#define NUM_MUESTRAS 500
```

```
#define V_REF 1099

#define TXD_PIN GPIO_NUM_10
#define RXD_PIN GPIO_NUM_9

#define DESTINO_DATOS 0

#define SPI_BUS TFT_VSPI_HOST


sdmmc_host_t host = SDSPI_HOST_DEFAULT();
sdspi_slot_config_t slot_config = SDSPI_SLOT_CONFIG_DEFAULT();
sdmmc_card_t* card;
esp_vfs_fat_sdmmc_mount_config_t mount_config;

char buff[TAM_MSG_DATA];

void SDMemoryConfig();
void cambio_archivo();
char SDMemoryWrite(char dato[TAM_MSG_DATA]);

void SDMemoryConfig() {
        host.max_freq_khz = 10000;
        slot_config.gpio_miso = PIN_NUM_MISO;
        slot_config.gpio_mosi = PIN_NUM_MOSI;
        slot_config.gpio_sck = PIN_NUM_CLK;
        slot_config.gpio_cs = PIN_NUM_CS_SD;

        mount_config.format_if_mount_failed = false;
        mount_config.max_files = 2;
        mount_config.allocation_unit_size = 4 * 1024;

        esp_err_t ret = esp_vfs_fat_sdmmc_mount("/sdcard", &host, &slot_config,
                        &mount_config, &card);

        while (ret != ESP_OK) {
```

```
                ret = esp_vfs_fat_sdmmc_mount("/sdcard", &host, &slot_config,
                                &mount_config, &card);
                printf("Aqui estoy... \n");
                vTaskDelay(TIME_WAIT);
        }
#if PANTALLA
        printf("SD Config\r\n");
#endif
}



void cambio_archivo(){

        file = fopen("/sdcard/datos.txt", "w");
        Temp = fopen("/sdcard/temporal.txt", "r+");
        while (!feof(Temp)){
                memset(buff,0,sizeof(buff));
                if (fgets(buff, sizeof(buff),Temp) == NULL) break;
                //printf("%s",buff);
                fprintf(file, "%s", buff);
        }

        fclose(file);
        fclose(Temp);
        remove("/sdcard/temporal.txt");
        esp_vfs_fat_sdmmc_unmount();
}



char SDMemoryWrite(char *dato) {
        //FILE* file=NULL;
        file = fopen("/sdcard/datos.txt", "w");
#if PANTALLA
        printf("Abriendo Datos\r\n");
```

```
#endif

        if (file == NULL) {
#if PANTALLA
                printf("Fallo al abrir archivo\r\n");
#endif
                return -1;
        }
        vTaskDelay(200);
        fseek(file, 0, SEEK_END);
        vTaskDelay(200);
        fprintf(file, dato);
        vTaskDelay(200);
        fclose(file);
#if PANTALLA
        printf("Escritura exitosa\r\n");
#endif
        esp_vfs_fat_sdmmc_unmount();
        return 0;
}

void read_SD(){

        static char  j = 0;
        xSemaphoreTake(Wifi_C, portMAX_DELAY);
        while(1){
                xSemaphoreTake(control_archivo, portMAX_DELAY);
                SDMemoryConfig();
                file = fopen("/sdcard/datos.txt", "r+");
                Temp = fopen("/sdcard/temporal.txt", "w");
                if (file != NULL){
                        printf("Archivo encontrado \r\n");
                        while (!feof(file)){
                                memset(buff,0,sizeof(buff));
                                if (fgets(buff, sizeof(buff),file) == NULL) {
                                        printf("Sin datos para publicar \n");
```

```
                                        break;
                                    }
                                    printf("DATO A PUBLICAR : \n ");
                                    printf("%s \n",buff);
                                    //-------- PUBLICO DATO -----------------------
                                    for (j = 0; j < 5; j++) {
                                        if(sendUbidotsWiFi(buff)==0){
                                            //-------- ESCRIBO DATOS EN TEMP ----------------
---
                                            while (!feof(file)){
                                                memset(buff,0,sizeof(buff));
                                                if (fgets(buff, sizeof(buff),file) == NULL)
break;
                                                    printf("%s",buff);
                                                    fprintf(Temp, buff);
                                                }
                                            fclose(file);
                                            fclose(Temp);
                                            vTaskDelay(100);
                                            //-------- CAMBIO LOS DATOS  DE ARCHIVO ----
---------------
                                            cambio_archivo();
                                            break;
                                            }
                                            vTaskDelay(TIME_WAIT / portTICK_RATE_MS);
                                        }
                                            fclose(file);
                                            fclose(Temp);
                                            vTaskDelay(2000);
                                    }
                                }
                            else {
                                printf("Error al buscar el archivo\r\n");
                            }
                    esp_vfs_fat_sdmmc_unmount();
                    xSemaphoreGive(control_archivo);
```

```
        vTaskDelay(10000);
        }
}

void app_main() {

        //---------------SEMAFORO --------------------

        control_archivo= xSemaphoreCreateMutex();

#if lectura
        xTaskCreate(&read_SD, "sensor",4000, NULL, 5, NULL);
#endif

}
```

Código del reloj:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "freertos/semphr.h"
#include "freertos/queue.h"

#include "esp_wifi.h"
#include "esp_system.h"
#include "esp_event.h"
#include "esp_log.h"
#include "esp_event_loop.h"
#include "nvs_flash.h"
#include "esp_vfs_fat.h"
#include "esp_heap_caps.h"

#include "driver/gpio.h"
#include "driver/adc.h"
#include "esp_adc_cal.h"

#include "driver/uart.h"
#include "soc/uart_struct.h"

#include "driver/sdmmc_host.h"
#include "driver/sdspi_host.h"
#include "sdmmc_cmd.h"

#include "lwip/netdb.h"
```

```
#include "lwip/sockets.h"
#include "driver/i2c.h"

#include "DS1307.h"
#include "UnixTime.h"

#define TIME_MUESTRA 60

#define PANTALLA 0
#define CLK_CONFIG 0
#define GPRS 1
#define SAVE_SD 1

#define lectura 1
#define escritura 1

#define SDA 21
#define SCL 22
#define FREC 100000
#define I2C_PORT I2C_NUM_1

#define PIN_NUM_MISO 19
#define PIN_NUM_MOSI 23
#define PIN_NUM_CLK 18
#define PIN_NUM_CS_SD 15

#define ACK_CHECK_EN 0x01
#define ACK_CHECK_DIS 0x00
#define TEMPERATURE_DIREC 0XF3
#define HUMEDAD_DIREC 0XF5

#define TAM_COLA 20
#define TAM_MSG_DATA 180
```

```
#define TAM_COLA_S 5
#define TAM_MSG_DATA_S 20

#define TIME_WAIT 1000

#define NUM_MUESTRAS 500
#define V_REF 1099

#define TXD_PIN GPIO_NUM_10
#define RXD_PIN GPIO_NUM_9

#define DESTINO_DATOS 0

#define SPI_BUS TFT_VSPI_HOST


sdmmc_host_t host = SDSPI_HOST_DEFAULT();
sdspi_slot_config_t slot_config = SDSPI_SLOT_CONFIG_DEFAULT();
sdmmc_card_t* card;
esp_vfs_fat_sdmmc_mount_config_t mount_config;

char buff[TAM_MSG_DATA];

void Take_Hour();
void time_task();

void Take_Hour(){
        clk_config_t tiempo;
#if CLK_CONFIG
        tiempo.segundos=00;
        tiempo.minutos=19;
        tiempo.hora=15;
        tiempo.formato=0;
```

```
        tiempo.state12h=0;
        tiempo.dia_semana=3;
        tiempo.dia_mes=13;
        tiempo.mes=11;
        tiempo.ano=19;
        config_clk(I2C_PORT,tiempo);
#endif
        tiempo=read_clk(I2C_PORT);
#if PANTALLA
        printf("%d:%d:%d",tiempo.hora,tiempo.minutos,tiempo.segundos);
#endif
        if(tiempo.formato==1){
                if(tiempo.state12h){
#if PANTALLA
                        printf(" PM");
#endif
                }
                else{
#if PANTALLA
                        printf(" AM");
#endif
                }
        }
#if PANTALLA
        printf("  -  %d/%d/20%d \r\n",tiempo.dia_mes,tiempo.mes,tiempo.ano);
        printf("------------------------------------------------------------\r\n");
#endif
        datos_tiempo.hora=tiempo.hora;
        datos_tiempo.minuto=tiempo.minutos;
        datos_tiempo.segundo=tiempo.segundos;
        memset(datos_tiempo.fecha,0,sizeof(datos_tiempo.fecha));
        sprintf(datos_tiempo.fecha,"%d-%d-
%d",tiempo.ano,tiempo.mes,tiempo.dia_mes);
```

```
}

void time_task() {

	//xSemaphoreTake(Take_T, portMAX_DELAY);

	//xSemaphoreGive(Time_Ready);
	while (datos_tiempo.segundo == 0 && datos_tiempo.minuto == 0 &&
datos_tiempo.hora == 0);
	while (1) {

		datos_tiempo.segundo++;

		if (datos_tiempo.segundo == 60) {
			datos_tiempo.minuto++;
			datos_tiempo.segundo = 0;
		}

		if (datos_tiempo.minuto == 60) {
			datos_tiempo.hora++;
			datos_tiempo.minuto = 0;
			datos_tiempo.segundo = 0;
		}

		if (datos_tiempo.hora == 24 && datos_tiempo.minuto == 1) {
			Take_Hour();
		}
#if PANTALLA
		printf("hora->%d:%d:%d\r\n", datos_tiempo.hora,
datos_tiempo.minuto,datos_tiempo.segundo);
#endif
		vTaskDelay(1000 / portTICK_RATE_MS);
```

```
        }

void app_main() {

        Take_Hour();

        xTaskCreate(&time_task, "Tiempo",4000, NULL, 4, NULL);
}
```

Código piranómetro:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "freertos/semphr.h"
#include "esp_wifi.h"
#include "esp_event.h"
#include "esp_log.h"
#include "esp_event_loop.h"
#include "nvs_flash.h"
#include "mqtt_client.h"
#include "esp_system.h"
#include "rom/ets_sys.h"
#include "sdkconfig.h"

#include "esp_vfs.h"
#include "esp_vfs_fat.h"

#include "driver/gpio.h"
#include "driver/adc.h"
#include "esp_adc_cal.h"

#define WIFI_SSID "HOME-36D2"
#define WIFI_PASS "A426B5ECFFCE874A"

float viento=0,Hum=0,Tem=0;
char buffer1[300];
xSemaphoreHandle Wifi_C;
xSemaphoreHandle mqtt_on;
```

```
xSemaphoreHandle data_take;
xSemaphoreHandle take_DHT;
#define V_REF 1100
#define NUM_MUESTRAS 500


float analog_lec(adc1_channel_t canal, adc_bits_width_t resolucion, adc_atten_t atenuacion){
static float muestra=0;
static long suma=0;
static int i=0;
static esp_adc_cal_characteristics_t characteristics;
uint32_t voltaje_tem=0;
adc1_config_width(resolucion);
adc1_config_channel_atten(canal,atenuacion);
esp_adc_cal_get_characteristics(V_REF, atenuacion, resolucion, &characteristics);

   suma=0;
    for(i=0;i<NUM_MUESTRAS;i++){
    esp_adc_cal_get_voltage(canal,&characteristics,&voltaje_tem);
    suma=suma+voltaje_tem;

    }

muestra=suma/NUM_MUESTRAS;

return muestra;
}


void anemometro(){
```

```
    uint32_t voltaje_adc;

    float irradiacion=0, voltaje_pira =0;
      while(1){
      voltaje_adc=analog_lec(ADC1_CHANNEL_7,ADC_WIDTH_BIT_12,ADC_ATTEN_DB_1
1); //Lectura analoga piranometro

      printf("ADC_VOLT:%d\n",voltaje_adc);
      voltaje_pira=voltaje_adc / 101.0; //Ganancia del AD623 (101)
      printf("Voltaje Pira: %0.3f\n",voltaje_pira);

      irradiacion = (voltaje_pira / (0.00001124); //Ganancia con ecuacion del piranometro
      printf("Irradiacion: %0.3f\n ",irradiacion);
      vTaskDelay(5000/portTICK_PERIOD_MS);

      }
}

void app_main(){


xTaskCreate(anemometro,"Tarea_Anemometro",5000,NULL,4,NULL);

}
```

4 Referéncias

[1] V. Pagola, R. Peña, J. Segundo, A. Ospino, “Rapid Prototyping of a Hybrid PV–Wind Generation System Implemented in a Real-Time Digital Simulation Platform and Arduino”, electronics, vol. 8, pp. 102, January 2019.

[2] S. Sinha, S.Chandel, ”Review of recent trends in optimization techniques for solar photovoltaic–wind based hybrid energy systems”, elsevier, vol. 50, pp. 755-769, May 2015.

[3] N. Priyadarshi, V. K. Ramachandaramurthy, S. Padmanaban, F. Azam, “An Ant Colony Optimized MPPT for Standalone Hybrid PV-Wind Power System with Single Cuk Converter”, European journal of physics, vol. 12, pp. 167, January 2019.

[4] Mellit, A.; Rezzouk, H.; Messai, A.; Medjahed, B. FPGA-based real time implementation of MPPT-controller for photovoltaic systems. Renew. Energy 2011, 36, 1652–1661.

[5] Okdem, S.; Karaboga, D. Routing in Wireless Sensor Networks Using Ant Colony Optimization. In First NASA/ESA Conference on Adaptive Hardware and Systems - AHS 2006; pp. 401–404.

[6] Liu, C.-L.; Chen, J.-H.; Liu, Y.-H.; Yang, Z.-Z. An Asymmetrical Fuzzy-Logic-Control-Based MPPT Algorithm for Photovoltaic Systems. Energies 2014, 7, 2177–2193.

[7] L. Morales, H Chamorro, J Soriano,"Análisis y comparación entre un controlador PI difuso y un controlador PI óptimo convencional para un conversor reductor", Ingeniería e Investigación, vol. 29, pp 61-66, diciembre de 2009.

[8] J. D. Moyano, «Tema III: El amplificador de instrumentación, Instrumentación electrónica de comunicaciones,» Santander, 2005.

[9] «Hetpro,» 28 Octubre 2017. [En línea]. Available: https://hetpro-store.com/TUTORIALES/i2c/. [Último acceso: 15 10 2020].

[10] C. Miranda, J. Ronquillo, “Diseño y construcción de bus de datos y sensores para las prácticas de NACC ”, universidad politécnica de cataluya, pp. 18-20, 15 enero 2008.

[11] «Hetpro,» 27 octubre 2017. [En línea]. Available: https://hetpro-store.com/TUTORIALES/puerto-serial/. [Último acceso: 15 octubre 2020].

[12] «Espressif,» [En línea]. Available: https://www.espressif.com/en/products/socs/esp32. [Último acceso: 10 octubre 2020].

[13] «SIMCom,» [En línea]. Available: https://cdn-shop.adafruit.com/product-files/2637/SIM800+Series_TCPIP_Application+Note_V1.01.pdf [Último acceso: 4 junio 2021].

[14] «SIMCom,» [En línea]. Available: https://www.elecrow.com/wiki/images/2/20/SIM800_Series_AT_Command_Manual_V1.09.pdf [Último acceso: 4 junio 2021].

[15] «nperf.com,» [En línea]. Available: https://www.nperf.com/es/map/CO/-/11016.Tigo-Movil/signal/?ll=11.034211786832714&lg=-74.34860229492189&zoom=10 [Último acceso: 6 agosto 2021].