Universidad de Pamplona

Desarrollo de un prototipo de estación meteorológica autónoma portátil con comunicación inalámbrica para el monitoreo en tiempo real de parámetros micro climatológicos

Autor:

Andres Felipe Hernandez Quiroz

Director:

Julio César Ospino Arias

MsC(c) Controles Industriales

INGENIERIA ELECTRÓNICA

DEPARTAMENTO DE ELECTRÓNICA, ELÉCTRICA, SISTEMAS Y TELECOMUNICACIONES

FACULTAD DE INGENIERÍAS Y ARQUITECTURA

UNIVERSIDAD DE PAMPLONA

PAMPLONA, 14 de diciembre de 2020

Dedicatoria:

A mi madre por todo lo que me ha dado, por ser todo lo que me hace ser, por estar siempre a mi lado dándome fuerzas para salir adelante y alentarme en cada momento a continuar sin permitirme rendir para ella de todo corazón le doy gracias a Dios y la vida por permitirme tenerla junto a mí.

A mi padre por apoyarme y darme la sabiduría necesaria para salir adelante, por enseñarme lo que se, por darme los consejos y la sabiduría necesaria para no caer en el camino equivocado, por prepararme para enfrentar la vida y por soportar los problemas que he podido dar.

A ambos por todo lo que me han dado y por todo lo que me hace ser, porque sin ellos no podría emprender este camino lleno de incertidumbres y de felicidades, de larga trayectoria y de vital importancia para el futuro.

Agradecimientos:

A la Universidad de Pamplona por permitirme estudiar esta carrera y ampliar mis campos del conocimiento.

Al profesor Julio Cesar Ospino por el apoyo brindado a lo largo de la carrera, y en especial, en estos últimos días de tan importancia, su comprensión ha sido una guía para buscar una salida y conseguir los objetivos propuestos a lo largo de la carrera, gracias por enseñarnos y prepararnos para la vida laboral y profesional con una ética y profesionalismo, pero al mismo tiempo dándonos la confianza para poder creer en usted.

A los profesores de la carrera de Ingeniería Electrónica por ayudarme en el trayecto para completar un paso más en la búsqueda del conocimiento y del aprendizaje, por dar los conocimientos necesarios para poder emprender esta carrera en los futuros proyectos de mi persona.

Resumen

En este proyecto se plantea el desarrollo de una estación meteorológica portable para el monitoreo de las variables del clima en tiempo real. Cuenta con sensores para medir temperatura, humedad, presión atmosférica y dióxido de carbono (CO2) para aplicaciones en las que solo se necesita el monitoreo básico de un microclima, y además cuenta con un mástil para añadir las mediciones de radiación UV, precipitación, velocidad y dirección del viento para usarse en aplicaciones en las que se necesita monitorear de una mayor cantidad de condiciones climáticas. Su software permite visualizar en tiempo real los datos del clima, además de dar la opción al usuario de exportar un registro de datos para su análisis. Con esto se busca de una forma económica y flexible habilitar y mejorar las condiciones de vida de las personas en áreas como la agricultura, cultivos en ciudades verdes, calidad de vida, entre otras.

Abstract

In this project the development of a portable weather station is proposed for near real time monitoring of climate variables. The station can measure temperature, humidity, pressure, and equivalent carbon dioxide (eCO2) for applications in which only a basic microclimate monitoring system is needed. It also has a mast to add UV, rain-gauge, wind speed and direction measurements to be used in applications where more data about climate variables is needed. The software for the weather station allows to visualize data in near real time, it also gives the user the ability to export a data logging for further analysis. With this, we look for an economical, and flexible way to enable and improve the living conditions of people in cities, remote places, countryside, or productivity in areas like agriculture, quality of life, among others

Índice

Índice de Figuras

Índice de Ilustraciones

Índice de Tablas

Índice de Ecuaciones

Introducción

1.1 Problema

El estudio y análisis de las variables climatológicas se ha convertido en asunto de gran importancia, no solo por las afectaciones que puedan presentarse para las actividades en un sitio especifico, sino por lo que dichos cambios puedan implicar para la sociedad en general.

El calentamiento global ha traído la necesidad de monitorear el clima en todo momento, puesto que los efectos de la humanidad en el ambiente están trayendo como consecuencia todo tipo de condiciones climatológicas que no se habían visto anteriormente, sumado a esto, la necesidad de prevenir las consecuencias económicas o sociales de un clima que puede cambiar de forma rápida y cuyos efectos pueden ser costosos e incluso fatales.

El aumento de población en ciudades, suburbios y en el campo, junto con la industrialización traen consigo la necesidad de conocer el microclima de un lugar, esto en busca de un desarrollo sostenible en medio de una globalización que cada vez más lleva a las sociedades a estar conectadas en un ámbito en que el concepto de clima es cambiante y cada vez más importante para el sostenimiento de estas.

1.2 Justificación

La implementación de una estación autónoma de monitoreo del clima de forma rápida, económica y sencilla trae consigo la posibilidad de tener en cuenta estos parámetros meteorológicos locales en tiempo real para actividades ganaderas, aeronáuticas, de cultivo, e incluso como alertas tempranas ante un riesgo inminente.

Al ser portable es posible monitorear el microclima de un lugar, permitiendo tomar medidas localizadas y confiables en lugares de alto riesgo, así como también al ser económica permite no solo su implementación en lugares del campo, sino en sitios dentro de una ciudad que se dediquen a cultivos hidropónicos donde las condiciones del microclima tienen prioridad sobre aquellas que puedan pronosticarse para un área general.

En el campo de la electrónica el monitoreo del clima permite profundizar en el aprendizaje, acople e interpretación de señales de sensores en microcontroladores y su integración mediante hardware y software con el fin de proporcionar al usuario una interfaz que le permita visualizar históricos de registros además de los datos en tiempo real, ofreciendo así un sinfín de servicios que permiten la toma de decisiones con base en la información obtenida.

1.3 Objetivos

1.3.1 Objetivo General

Desarrollar un prototipo de estación meteorológica autónoma portátil con comunicación inalámbrica para el monitoreo en tiempo real de parámetros micro climatológicos

1.3.2 Objetivos Específicos

- Determinar las características técnicas de los componentes que conforman la estación meteorológica

- Diseñar un sistema de comunicación para obtener datos de la estación meteorológica de forma inalámbrica

- Diseñar un software para monitoreo y visualización de los datos obtenidos por la estación meteorológica

- Diseñar un sistema para alimentación por panel solar y batería de forma que se asegure la operación incluso sin conexión a red eléctrica

- Implementar los sistemas y puesta en marcha de la estación meteorológica

Marco Teórico

2.1 Antecedentes

A nivel nacional

En el 2016, Daniel Camilo Ruiz Ayala, en la Universidad de Pamplona, presento el trabajo de grado titulado "DESARROLLO DE UN SISTEMA DE MONITOREO INALÁMBRICO DE ALGUNAS VARIABLES CLIMÁTICAS" [1]. En este trabajo se desarrolló una estación meteorológica la cual mide temperatura, humedad, presión, radiación, velocidad y dirección del viento, empleando un PIC 18F2550 y un PIC 18F4550, empleando Wifi para él envió de datos a un servidor, además de emplear conexión USB para enviar datos a un computador, también tiene aplicación en Android descargando los datos del servidor web, y puede almacenar los datos en una memoria SD, siendo esta una de las más completas en conectividad pero con un alto costo.

En el 2017, Rafael David Ferrer Sanabria, en la Universidad Santo Tomás sede Bucaramanga, presento el Trabajo de grado titulado "Desarrollo de una estación meteorológica autónoma de bajo costo" [2]. En este trabajo se desarrolló una estación meteorológica de bajo costo para el sector agrícola empleando un Arduino MEGA, la cual mide temperatura, humedad relativa, humedad del suelo, luminosidad, presión barométrica, precipitación, gases, velocidad y dirección del viento, empleando comunicación GPRS para enviar la información a una base de datos con interfaz HTML y PHP.

En el 2017, Brian Yesid Garzón Guzmán y María Fernanda Rincón Cerón, en la Universidad Francisco José de Caldas, presentaron el proyecto de grado titulado "DISEÑO E IMPLEMENTACIÓN DE UN PROTOTIPO DE ESTACIÓN METEOROLÓGICA PARA LA MEDICIÓN DE VARIABLES AMBIENTALES" [3]. En este proyecto se construye una estación meteorológica automática la cual mide temperatura, humedad, presión atmosférica, radiación, precipitación, dirección y velocidad del viento, empleando un Arduino DUE, el cual envía datos a un servidor local empleando Wifi, además de almacenar los datos en una memoria SD.

En el año 2018, Raul Alberto Gavíria Valencia, en la Universidad Libre seccional Pereira, presento el proyecto de grado e investigación titulado "PROTOTIPO FUNCIONAL ESTACIÓN METEOROLÓGICA PORTABLE CON DISPOSITIVOS DE BAJO COSTO (ARDUINO)" [4]. En este proyecto se construye una estación meteorológica la cual mide temperatura, humedad, presión atmosférica, nivel del agua y precipitación a un bajo costo empleando un Arduino UNO, empleando comunicación GPRS e integrando los datos a la plataforma ThingSpeak.

A nivel internacional

En el año 2016, Alberto Tobajas García, de la Universidad de Cataluña, presento el trabajo de grado titulado “Diseño e implementación de una estación meteorológica con Raspberry Pi” [5]. En este proyecto se construye una estación meteorológica la cual mide temperatura, humedad, presión barométrica, precipitación, luminosidad, dirección y velocidad del viento, empleando una Raspberry Pi 3, en la cual se encuentra un servidor WEB instalado, la información de los sensores puede ser visualizada al conectarse a la Raspberry Pi al conectarse a la dirección IP de esta con lo que se obtiene una página WEB.

En el año 2017, Mahmoud Shaker Nasr y Walla’a Y.Y. Alebady, de la Universidad de Babilonia, publicaron un artículo titulado “Design and Implementation of a Smart Weather Station Based on Internet of Things” [6] (Diseño e implementación de una estación meteorológica inteligente basada en el internet de las cosas). En este artículo se describe la construcción de una estación meteorológica para monitorear el ambiente en la ciudad de Babilonia, dicha estación puede medir temperatura, humedad, concentraciones de gases e intensidad de luz, empleando un Arduino MEGA, la cual puede comunicarse por medio de GSM y Ethernet para enviar datos a un servidor, de manera que estos puedan ser visualizados en una página web.

En el 2018, Carlos Morón, Jorge Pablo Diaz, Daniel Ferrández y Pablo Saiz, de la Universidad Politécnica de Madrid y de la Universidad REY Juan Carlos, publicaron un artículo titulado “Design, Development and Implementation of a Weather Station Prototype for Renewable Energy Systems” [7] (Diseño, desarrollo e implementación de un prototipo de estación meteorológica para sistemas de energía renovable). En este artículo se desarrolló una estación meteorológica, la cual puede medir temperatura, humedad, presión barométrica, precipitación, granizo, calidad del aire, velocidad y dirección del viento, empleando un Arduino MEGA, con el propósito de comparar los resultados obtenidos con los de estaciones meteorológicas profesionales.

En el 2019, Mary Nsabagwaa, Maximus Byamukamab, Emmanuel Kondelaa y Julianne Sansa Otima, de la Universidad Makerere en Uganda, publicaron un artículo titulado “Towards a robust and affordable Automatic Weather Station” [8] (Hacia una estación meteorológica automática robusta y asequible). En este artículo se desarrolló una estación meteorológica automática (AWS), la cual puede medir temperatura, humedad, temperatura y humedad del suelo, presión atmosférica, velocidad y dirección del viento, con el propósito de que países en desarrollo puedan adquirir estaciones meteorológicas a menores costos y de esta manera mejorar los pronósticos del clima.

2.2 Meteorología

Es el estudio de la atmosfera, sus fenómenos y efectos en el clima a corta escala [9]. El estudio de estos parámetros en el tiempo puede ser usado para pronosticar las condiciones climatológicas en un lugar [10], dependiendo del número de estaciones y los parámetros obtenidos puede usarse para estudiar el clima en una microescala, meso escala o en una escala global [11].

2.3 Estación meteorológica

La estación meteorológica es una construcción o lugar donde se colecta información acerca de los diferentes parámetros del clima local [12]. Entre la información que colectan suele estar la temperatura, humedad, presión barométrica, precipitación, velocidad y dirección del viento [13]. Esta puede ser automática con telemetría, donde la información que colecta es enviada automáticamente a un centro de procesamiento usando medios de transmisión (Wifi, GPRS, RF, etc.) o automática sin telemetría, donde la información colectada es guardada en un medio de almacenamiento local y al cual debe acceder una persona para poder obtener los datos de esta. También se pueden encontrar estaciones no automáticas, las cuales dependen de personal que se encargue de la toma y registro de los parámetros del clima, para luego enviarlos a su lugar de procesamiento [14].

2.4 IDEAM

Instituto de Hidrología, Meteorología y Estudios Ambientales, es la entidad gubernamental encargada del estudio, registro y procesamiento del estado del medio ambiente, el clima y los recursos naturales de Colombia, con el fin de que en base a esta información se generen las políticas institucionales y gubernamentales en materia del medio ambiente, además de brindar acceso de los datos recolectados a la población en general [15].

2.5 Arduino

Plataforma de código abierto desarrollada para facilitar la programación de los microcontroladores por parte de personas que inician en la electrónica o que busquen una forma rápida de realizar prototipos funcionales basados en la familia ATMEL, siendo el más popular el ATMEGA328 incluido en el Arduino UNO.

2.6 Protocolo SPI

La interfaz serial de periféricos es un protocolo de comunicación serial síncrono que usa 4 cables [16] usado para comunicaciones primariamente entre dispositivos embebidos a cortas distancias, entre los dispositivos que lo utilizan se encuentran algunos como las SD (Secure Digital) o las pantallas LCD [17].

2.7 Protocolo I2C

El Protocolo de Circuito Inter-Integrado (I2C) es un protocolo Serial para comunicación vía interfaz de dos cables para conectar dispositivos de baja velocidad [18] como microcontroladores, EEPROMs, Sensores, Interfaces E/S, y dispositivos embebidos similares, diseñado por Philips Semiconductor (Ahora NXP) en 1982 [19].

Permite que múltiples dispositivos periféricos se comuniquen con uno o más controladores en distancias cortas que hagan parte de un mismo dispositivo. Su velocidad de reloj máxima inicial era de 100kbit/s, pero hoy en día se encuentran modos que permiten llegar hasta los 3.4Mbit/s o 5Mbit/s (Unidireccional) [20].

2.8 Protocolo 1-Wire

Protocolo de comunicación Serial que solo requiere un solo cable adicional a la referencia de tierra [21], similar a I2C, pero con menores velocidades y mayores distancias. Generalmente es usado para comunicar dispositivos sencillos como sensores de temperatura, además algunos dispositivos pueden disponer de la energía por medio del mismo cable (Energía parásita) [22]

2.9 ADC

Un conversor análogo a digital es un circuito electrónico que permite obtener un valor digital que es representativo de la magnitud análoga con el objetivo de realizar procesamiento con el valor resultante (por ejemplo, para leer la temperatura, o para realizar una acción en caso de un exceso de líquido en un tanque). Esto se realiza por medio de un muestreo en el tiempo (la acción de medir en un intervalo de tiempo) con lo que se obtiene una representación aproximada de la señal.

2.10 Interrupciones

Las interrupciones son una señal que presenta la más alta prioridad en un controlador, lo que hace que el controlador deje de realizar el proceso en el que se encuentra temporalmente para realizar el proceso asociado a la interrupción antes de regresar al proceso actual.

Esto puede ser usado para realizar acciones o tomar datos en base a una señal sea periódica o no, que requiere de su registro inmediato (por ejemplo, para medir velocidad, o para leer datos de un teclado)

2.11 Bluetooth

Bluetooth se define como una tecnología inalámbrica para el intercambio de datos entre diferentes dispositivos en un rango corto usando la frecuencia de 2.40GHz en la que se crean redes de área personal (PAN). Por lo general es usado para enviar datos de un dispositivo servidor a un cliente en forma de paquetes con comunicación bidireccional.

2.12 GSM

Sistema para comunicaciones móviles globales es un estándar que define los protocolos para la segunda generación de redes móviles (2G) utilizados en dispositivos como celulares y tablets, desplegado por primera vez en Finlandia en diciembre de 1991

2.13 Windows

Sistema operativo de interfaz gráfica propietario desarrollado por Microsoft. Es el sistema operativo de escritorio con mayor uso en el mundo, su núcleo permite adaptarlo a distintos dispositivos (Tablets, IoT, Portátiles, Servidores, Consolas, entre otros) además de distintas arquitecturas (x86, x64 (Intel Itanium), x86_64, ARM, Power PC, entre otros) dado que su núcleo está pensado para poderse adaptar a distintos entornos tanto de software como de hardware (además de proveer de capas de abstracción de hardware que básicamente lo vuelven agnóstico (no depende de una sola arquitectura, hardware o driver)) Permite ejecutar aplicaciones, juegos, navegadores, reproductores de música y videos, entre otros.

2.14 Android

Sistema operativo de código abierto derivado de un kernel modificado de Linux y desarrollado por Google. Es el sistema operativo con mayor uso en el mundo, su aplicación principal es en Smartphones, pero el mismo puede ser usado en tablets, portátiles, televisores, entre otros.

2.15 Visual Studio

Entorno de desarrollo para crear software que puede ser utilizado principalmente en la plataforma Microsoft Windows, pero que recientemente también ha implementado soporte para desarrollo de software en Android y iOS, desarrollado por Microsoft y disponible en versión gratuita para empresas y desarrolladores pequeños, además de licencias de pago para entornos de desarrollo más grandes.

2.16 Android Studio

Entorno de desarrollo para crear software que puede ser utilizado en la plataforma Android (TV, Watch, Smartphone) soportado en el lenguaje de programación java y kotlin, desarrollado por Google y disponible de forma gratuita para desarrolladores.

Metodología

1. • Revisión Bibliografica
2. • Determinación de caracteristicas tecnicas y programación de sensores
3. • Programación del sistema de comunicación
4. • Programación del software de monitoreo y visualización de datos
5. • Dimensionamiento del sistema energetico
6. • Validación de la funcionalidad

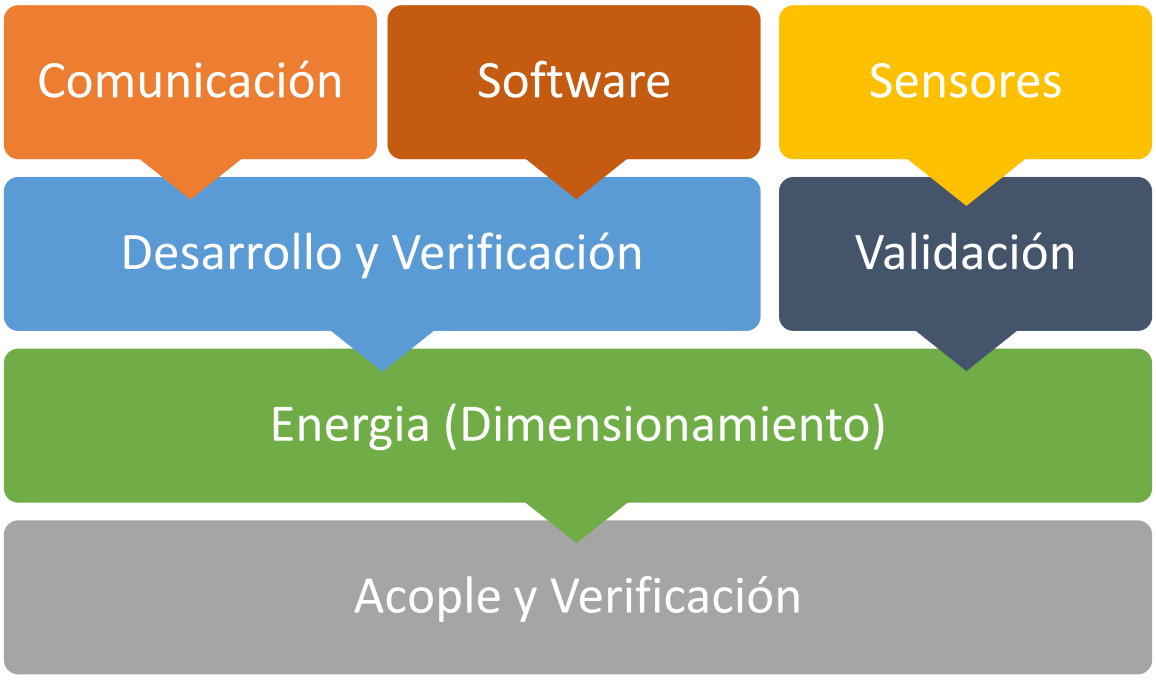


Ilustración 1 – Metodología del trabajo de grado. [Fuente: Autor].

3.1 Sensores

Se hace la selección de los sensores dependiendo de sus características técnicas las cuales están representadas a continuación:

3.1.1 Temperatura

Para la temperatura se tuvieron en cuenta los siguientes sensores con sus respectivas características técnicas, exceptuando los que ya fueron elegidos para presión y humedad.

	Voltaje	**Rango**	**Precisión**	**Resolución**	**Comunicación**	**Calibrado**
LM35	4 – 30V	-55 - 150°C	0.5°C	10mV(°C)	Análogo	O
DHT22	5V	-40 – 80°C	0.5°C	0.1°C	OneWire	O
PT100/3	-	-	-	-	Análogo	X
HTU21D	3.7V	-40 – 125°C	0.3°C	0.01°C	I2C	O
AM2301	5V	-40 – 80°C	0.5°C	0.1°C	OneWire	O
BME280	1.7 – 3.6V	-40 – 85°C	0.5°C	0.01°C	I2C	O
SHT30	2.5 – 5V	-40 – 125°C	0.3°C	0.015°C	I2C	O
SHT20	2.1 – 3.6V	-40 – 125°C	0.3°C	0.01°C	I2C	O
BMP280	1.7 – 3.6V	-40 – 85°C	1°C	0.01°C	I2C	O
DS18B20	3 – 5.5V	-55 – 125°C	0.5°C	Varia	OneWire	O
AM2305	3.3 – 5.5V	-40 – 125°C	0.4°C	0.1°C	OneWire	O
Termocupla tipo K	-	-	-	-	Análogo	X

Tabla 1 – Sensores de temperatura encontrados durante búsqueda bibliográfica. [Fuente: Autor].

El sensor elegido es el DS18B20, puesto que ofrece unas características que son más que suficientes para la estación meteorológica, el mismo viene en una sonda para protegerlo del ambiente, además de tener un precio que permite su obtención fácil, lo que puede ser tenido en cuenta en casos de falla de los sensores

Imagen 1 – Sensor de temperatura con sonda DS18B20. [Fuente: FERRETRÓNICA].

DS18B20 – Características Técnicas

Voltaje	3 – 5.5 V
Corriente	1 – 1.5 mA
Rango	-55 – 125 °C
Precisión (-10°C a +85°C)	±0.5°C
Resolución	Varia, Programable de 9 a 12 Bits
Comunicación	1-Wire
Costo	1.93 USD

Tabla 2 – Características técnicas del sensor DS18B20. [Fuente: Datasheet, MAXIM].

Diagrama de Flujo para el Sensor de Temperatura:

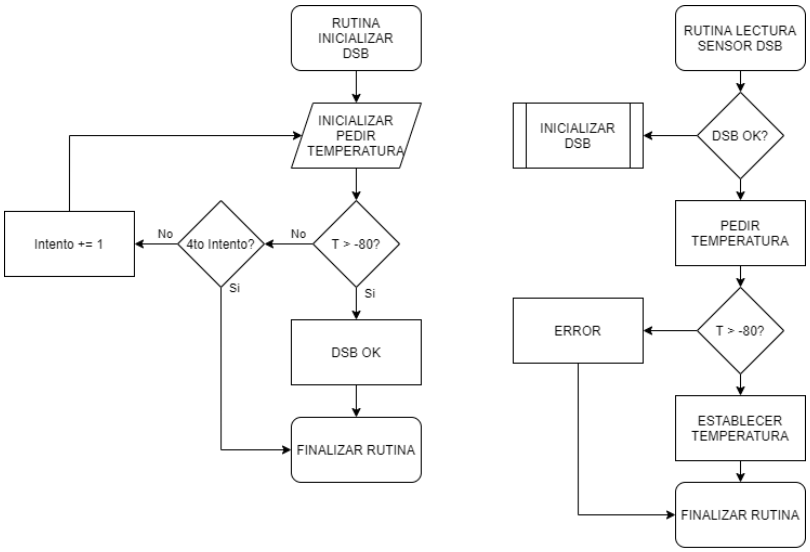


Ilustración 2 – Diagrama de Flujo para el DS18B20. [Fuente: Autor].

3.1.2 Humedad

Para la humedad se tuvieron en cuenta los siguientes sensores, algunos de ellos también permiten la medición de la temperatura, pero en este caso se busca principalmente la humedad, y solo se considera la temperatura como un diferenciador.

	Voltaje	Rango	Precisión	Resolución	Comunicación	Calibrado
HS1101	5 – 10V	0 – 100%	2%	1.50%	Análogo	O
DHT22	5V	0 – 100%	2%	0.10%	OneWire	O
HTU21D	3.7V	0 – 100%	2%	0.04%	I2C	O
AM2301	5V	0 – 100%	2%	0.10%	OneWire	O
AM2315	5V	0 – 100%	2%	0.10%	OneWire	O
HIH4030	5V	0 – 100%	4%	4mV	Análogo	O
BME280	1.7 – 3.6V	0 – 100%	3%	0.01%	I2C	O
SHT30	2.5 – 5V	0 – 100%	2%	0.01%	I2C	O
SHT20	2.1 – 3.6V	0 – 100%	3%	0.04%	I2C	O
AM2305	3.3 – 5.5V	0 – 100%	2%	0.10%	OneWire	O

Tabla 3 – Sensores de Humedad encontrados durante búsqueda bibliográfica. [Fuente: Autor].

El sensor elegido es el SHT30, en una revisión bibliografía se encuentra que la familia de sensores SHT es conocida por su calidad y precisión, además de todo esto, el hecho de que también mide temperatura lo cual permite dar redundancia a la medición en una forma compacta y funcional.

Imagen 2 – Sensor SHT30 de Sensirion en Baquela. [Fuente: MACTRONICA].

SHT30 – Características Técnicas

Voltaje	2.4 – 5.5 V
Corriente	2 – 800 µA
Rango	-40 – 125 °C
Precisión (10°C a +55°C)	±0.3°C
Resolución	±0.05°C
Rango [Humedad]	0 – 100% RH
Precisión [Humedad]	±3% RH
Resolución [Humedad]	±0.05% RH
Comunicación	I2C
Costo	5.50 USD

Tabla 4 – Características técnicas del sensor SHT30. [Fuente: Datasheet, Sensirion].

Diagrama de Flujo para el Sensor de Humedad:

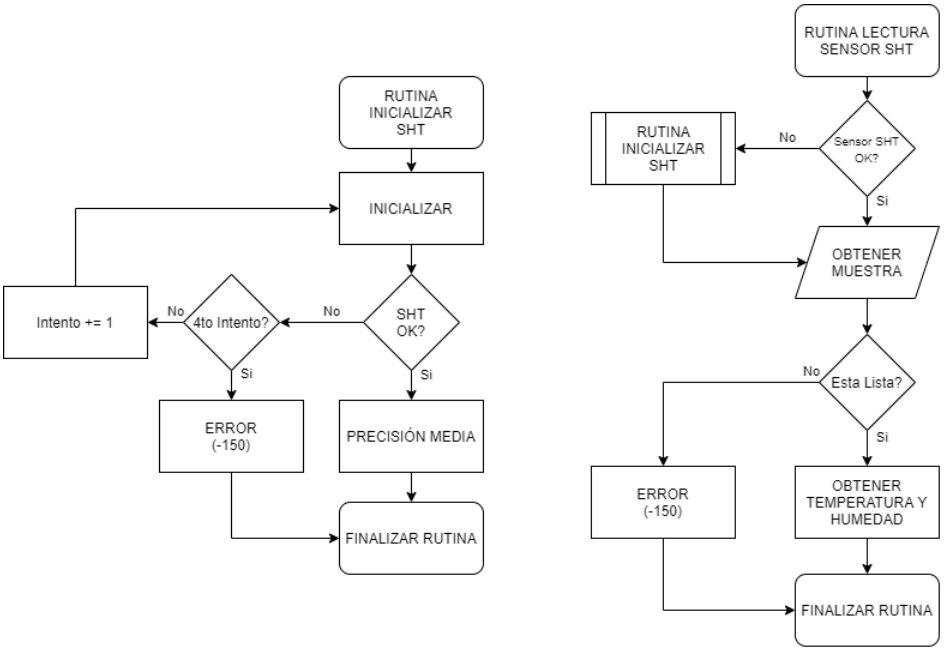


Ilustración 3 – Diagrama de Flujo para el SHT30. [Fuente: Autor].

3.1.3 Presión

Para la presión, si bien se tuvieron en cuenta los siguientes sensores, se encontró que muchos de ellos están ya descontinuados por el fabricante, además, algunos de estos permiten la medición de temperatura, la cual solo se considera como un diferenciador.

	Voltaje	**Corriente**	**Rango [hPa]**	**Resolución**	**Comunicación**	**Calibrado**
BMP280	1.7 – 3.6V	2.8 µA	300 – 1100	0.01	I2C	O
MPL3115A2	2.0 – 3.6V	8.5 µA	200 – 1100	Varia	I2C	O
MPL115A2	2.4 – 5.5V	5 µA	500 – 1150	1.5	I2C	O
BME280	1.7 – 3.6V	714 µA	300 – 1100	0.01	I2C	O

Tabla 5 - Sensores de Presión Barométrica encontrados durante búsqueda bibliográfica. [Fuente: Autor].

El sensor elegido es el BMP280, fabricado por BOSCH. Es un sensor de presión barométrica absoluta que suele ser usado en dispositivos como celulares, módulos GPS o relojes. El sensor también permite obtener temperatura y altitud la cual se calcula en base a la presión del ambiente compensando con el valor de la presión a nivel del mar.

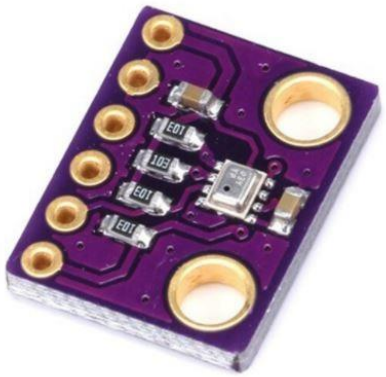


Imagen 3 – Sensor Bosch BMP280 en Baquela. [Fuente: MACTRONICA].

BMP280 – Características Técnicas

Voltaje	1.7 – 3.6 V [5V en módulo disponible para compra]
Corriente	2.8 – 1120 (Pico Máximo) µA
Rango	-40 – 85 °C
Precisión (0°C a +65°C)	±1.0°C
Resolución	±0.01°C [En modo ultra alta resolución]
Rango [Humedad]	300 – 1100 hPa
Precisión [Presión] (0 – 65°C)	±1.7 hPa
Resolución [Humedad]	±0.0016 hPa [En modo ultra alta resolución]
Comunicación	I2C
Costo	2.20 USD

Tabla 6 - Características técnicas del sensor BMP280. [Fuente: Datasheet, BOSCH].

Diagrama de Flujo para el Sensor de Presión y Altitud:

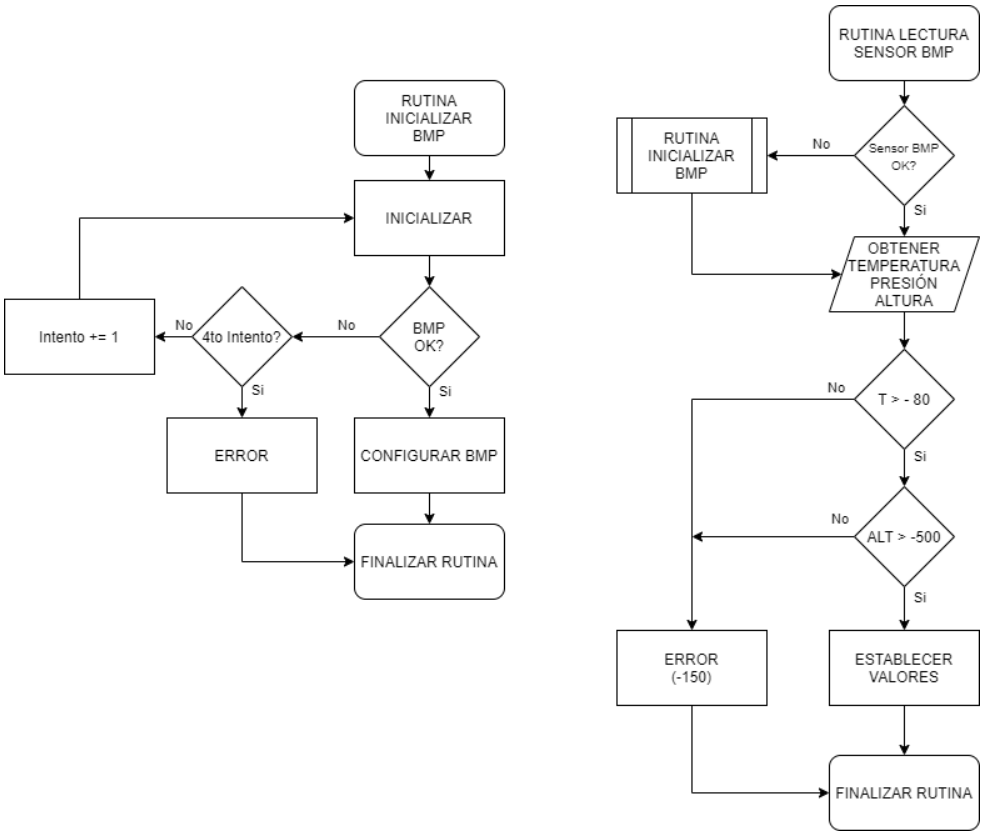


Ilustración 4 – Diagrama de Flujo para el BMP280. [Fuente: Autor].

3.1.4 Intensidad Lumínica

Para el sensor de intensidad lumínica se consideraron diferentes referencias, sin embargo, al consultar disponibilidad, se llegó a la conclusión de que el único sensor disponible para la compra era el BH1750, por lo que este fue el escogido en una coraza que lo protege.

	Voltaje	Corriente	Longitud de Onda	Resolución	Comunicación	Temperatura
BH1750	3.3 – 5 V	120 µA	1 – 65535	1 – 4 Lx	I2C	-40 – 85°C
TSL-2561	3.6 V	240 µA	188µ - 88000	Varia	I2C	-40 – 85°C
BOB-08688	6 V	20000 µA	390 – 700 nm	Varia	Análogo	-40 – 85°C
APDS-9301	3.6 V	240 µA	1 – 65535	1 Lx	I2C	-30 – 85°C

Tabla 7 - Sensores de intensidad lumínica encontrados durante búsqueda bibliográfica. [Fuente: Autor].

Imagen 4 – Sensor BH1750FVI de ROHM Semiconductor en coraza protectora. [Fuente: MACTRONICA].

BH1750 – Características Técnicas

Voltaje	2.4 – 3.6 V [5V en módulo disponible para compra]
Corriente	120 – 190 µA
Longitud de Onda Pico	560 nm
Rango	1 – 65535 lx
Precisión	0.96 – 1.44 Veces [Salida Sensor / lx Actuales]
Resolución	1 – 4 lx
Comunicación	I2C
Costo	3.58 USD

Tabla 8 - Características técnicas del sensor BH1750FVI. [Fuente: Datasheet, ROHM Semiconductor].

Diagrama de Flujo para el sensor de Intensidad Lumínica

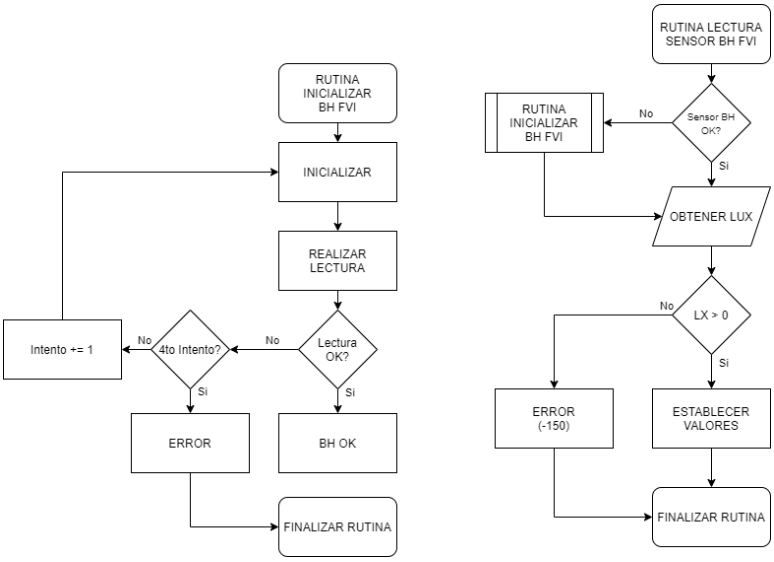


Ilustración 5 – Diagrama de Flujo para el BH1750FVI. [Fuente: Autor].

3.1.5 Índice Ultravioleta

Para el sensor de índice ultravioleta, se encontró que los únicos sensores disponibles son el GUVA-S12SD, UVM-30A y el ML8511, de los cuales una búsqueda llevo a la conclusión de que el UVM-30A y el GUVA-S12SD utilizan el mismo sensor, seguidamente del datasheet se establece una salida directamente relacionada entre el nivel de intensidad UV y los valores dados por el sensor por lo que se elige el UVM-30A.

	Voltaje	**Corriente**	**Longitud de Onda**	**Resolución**	**Comunicación**	**Temperatura**
ML8511	3.3 – 5 V	300 µA	365	-	Análogo	-40 – 85°C
UVM-30A	3.6 V	60 µA	200 – 370	0 – 1200 mV	Análogo	-40 – 85°C
VEML6070	6 V	100 µA	320 – 410	-	I2C	-40 – 85°C
GUVA-S12SD	3.6 V	1000 µA	240 – 370	0.14 A/W	Análogo	-30 – 85°C

Tabla 9 – Sensores ultravioleta encontrados durante revisión bibliográfica. [Fuente: Autor].

Imagen 5 – Sensor UVM-30A en Baquela. [Fuente: MACTRONICA].

UVM-30A – Características Técnicas

Voltaje	3.0 – 5.0 V
Corriente	60 – 100 µA
Longitud de Onda de Respuesta (Rango)	200 – 370 nm
Precisión	±1 Índice UV
Comunicación	I2C
Costo	11.83 USD

Tabla 10 - Características técnicas del sensor UVM-30A. [Fuente: Wiltronics].

Diagrama de Flujo para el Sensor de Intensidad Ultravioleta

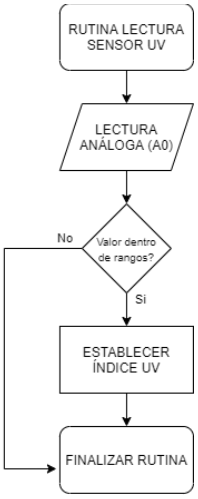


Ilustración 6 – Diagrama de Flujo para el sensor UVM-30A. [Fuente: Autor].

3.1.6 Dirección del Viento

Se utiliza una veleta como el sensor utilizado para medir la dirección del viento, funciona por medio de resistencias las cuales se encuentran distribuidas de manera que se obtiene un valor de voltaje especifico en los 8 puntos cardinales. Dicho valor se lleva a un ADC para poder realizar el procesamiento con un microcontrolador y de este modo saber la dirección del viento en un lugar específico.

Imagen 6 – Veleta para medir la dirección del viento. [Fuente: SparkFun].

Para este sensor, se utiliza una resistencia externa de 10kΩ de manera que se genera un divisor de voltaje, con el cual, se obtiene un voltaje de salida que es medido por el microcontrolador a través del ADC integrado. Para poder establecer los valores de voltaje que corresponden a las diferentes direcciones posibles, se realiza primero una fórmula para obtener los valores aproximados de R2.

Teniendo en cuenta la fórmula para el divisor de voltaje:

$$Vout = \frac{Vin * R2}{R1 + R2}$$

Ecuación 1 – Divisor de Voltaje. [Fuente: Autor].

Despejando R2, se obtiene:

$$R2 = \frac{\left(\frac{R1 * Vout}{Vin}\right)}{\left(1 - \frac{Vout}{Vin}\right)}$$

Ecuación 2 – Valor de resistencia R2. [Fuente: Autor].

Además, para obtener un valor aproximado del voltaje que llega al arduino, se tiene en cuenta la siguiente fórmula para convertir de los valores dados por el ADC a voltaje:

$$V = \frac{ADC_{IN} * 5V}{1023}$$

Ecuación 3 – Voltaje equivalente al valor dado por el ADC del Arduino. [Fuente: Autor].

Esto para establecer los valores de voltaje que correspondan a la salida del divisor de voltaje, con lo que, al comparar con los del fabricante se obtiene. Al realizar las mediciones en los distintos valores correspondientes, se obtiene:

Ángulo (°)	R2 Dado por el Fabricante (Ω)	R2 Obtenido (Ω)	Voltaje del divisor (V)
0	33k	33K	3.837
45	8.2k	8.2K	2.253
90	1k	1K	0.454
135	2.2k	2.2K	0.902
180	3.9k	3.9K	1.403
225	16k	16K	3.077
270	120k	120K	4.615
315	64.9k	64.9K	4.332

Tabla 11 – Valores de Resistencia y Voltaje de salida respectivo según el ángulo. [Fuente: Autor].

Verificando los voltajes en el Arduino, se establece un rango para cada voltaje (para dar lecturas más estables) de 0.12V (±25 en la lectura del ADC)

Ángulo (°)	Voltaje (V)	Valor ADC Mín	Valor ADC Medio	Valor ADC Máx
0	3.837	760.0502	785.0502	810.0502
45	2.253	435.9638	460.9638	485.9638
90	0.454	67.8884	92.8884	117.8884
135	0.902	159.5492	184.5492	209.5492
180	1.403	262.0538	287.0538	312.0538
225	3.077	604.5542	629.5542	654.5542
270	4.615	919.229	944.229	969.229
315	4.332	861.3272	886.3272	911.3272

Tabla 12 – Valores mínimos, medios y máximos para establecer el rango correspondiente a cada grado en el Arduino. [Fuente: Autor].

Diagrama de Flujo para el sensor de Dirección del Viento:

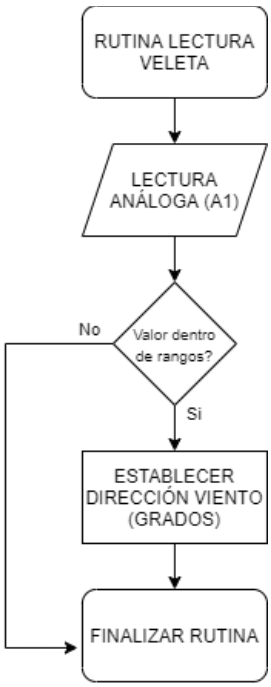


Ilustración 7 – Diagrama de Flujo para la Veleta. [Fuente: Autor].

3.1.7 Velocidad del Viento

Se utiliza un anemómetro como el sensor utilizado para medir la velocidad del viento, el mismo posee un switch magnético, el cual genera un pulso al ser excitado por un campo magnético generado por un imán. Para este caso el sensor genera dos pulsos por rotación, el número de rotaciones en un tiempo y con un radio determinado es lo que se utiliza para determinar la velocidad del viento.

Imagen 7 – Anemómetro para medir la velocidad del viento. [Fuente: SparkFun].

La fórmula para calcular la velocidad del viento es:

$$V = N\left(\frac{2\pi r}{T}\right) \quad \rightarrow \quad V = N\left(\frac{2\pi(0.0525m)}{T}\right) \quad \rightarrow \quad V = N\left(\frac{0.329867m}{3s}\right)$$

$$donde\ N = \frac{numero\ de\ pulsos}{2}, \qquad V = \frac{m}{s}, \qquad T = seg, \qquad R = metros$$

Ecuación 4 – Velocidad del viento en N rotaciones. [Fuente: Autor].

Diagrama de Flujo para el Sensor de Velocidad del Viento

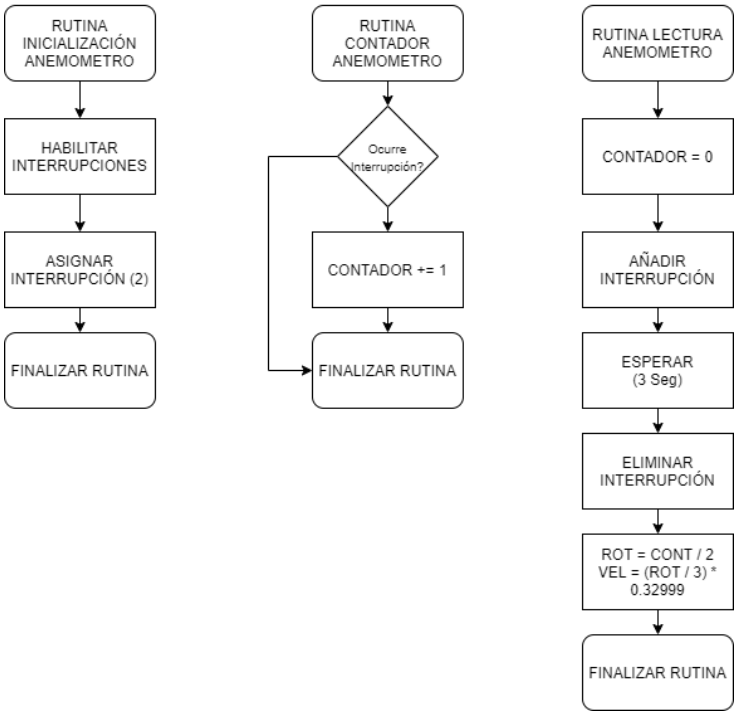


Ilustración 8 – Diagrama de Flujo para el Anemómetro. [Fuente: Autor].

3.1.8 Sensor de precipitación (Pluviómetro)

Se utiliza un pluviómetro como el sensor utilizado para calcular la precipitación en un momento dado. El sensor posee un switch magnético el cual se activa bajo la presencia de un campo generado por un imán (un principio similar al del anemómetro). En este caso, el movimiento es generado por la lluvia, la cual llega a un reservorio (acumulador) y de la cual cae por un balancín, el cual "gira" de un lado a otro cada 0.2794mm de lluvia (datos del fabricante, falta verificación).

Esta activación momentánea genera un pulso, el cual es detectado por el microcontrolador, que genera un conteo, el cual es relativo, pues depende de la cantidad de lluvia que se quiera registrar en un tiempo determinado (Ej: si es por minuto, el conteo se reinicia cada minuto, si es por hora, entonces se reinicia

por hora). El número de pulsos detectados indica la cantidad de lluvia registrada en un tiempo determinado.

$$[Cantidad\ en\ mm] = Número\ de\ pulsos * 0.2794mm$$

Ecuación 5 – Cantidad de milímetros de agua en base al número de pulsos. [Fuente: Autor].

Imagen 8 – Sensor de precipitación (Pluviómetro). [Fuente: SparkFun].

Diagrama de Flujo para el Sensor de Lluvia

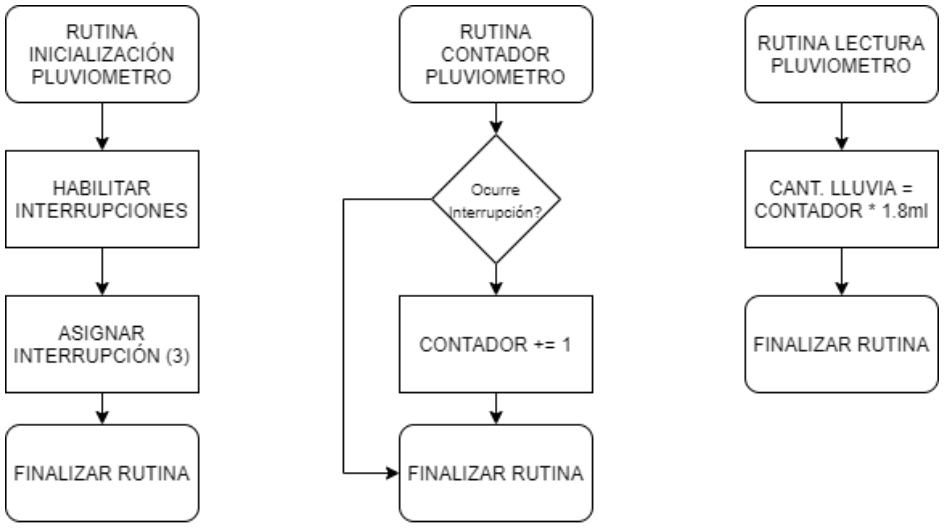


Ilustración 9 – Diagrama de Flujo para el Pluviómetro. [Fuente: Autor].

3.1.9 Sensor de calidad del aire (eCO2 y TVOC)

Para el sensor de dióxido de carbono, se utiliza el CCS811, un sensor que permite obtener los niveles equivalentes a dióxido de carbono y de compuestos orgánicos volátiles.

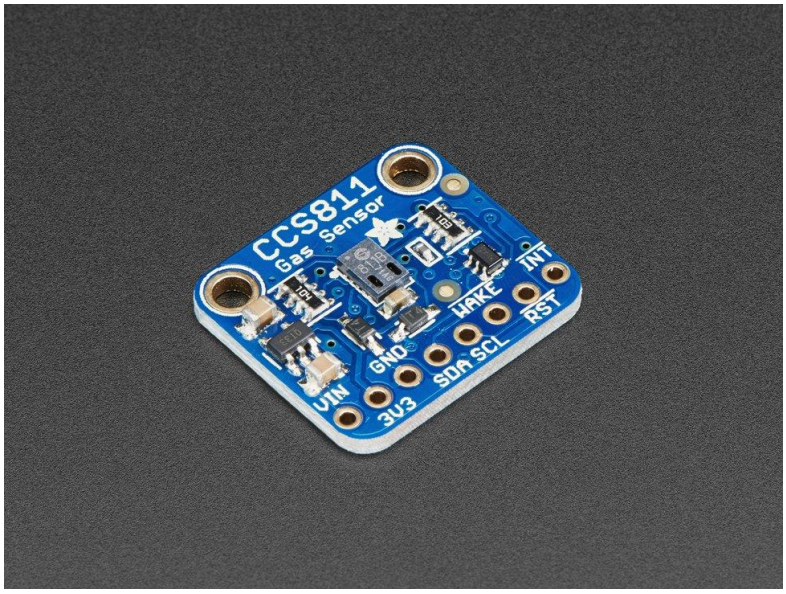


Imagen 9 – Módulo de calidad del aire CCS811 de Adafruit. [Fuente: Adafruit].

CCS811 – Características Técnicas	
Voltaje	1.8 – 3.6 V
Corriente	0.7 - 30 mA (Máximo)
Rango (eCO2)	400 – 8192 ppm
Rango (TVOC)	0 – 1187 ppb
Comunicación	I2C
Costo	23.11 USD

Tabla 13 - Características técnicas del módulo CCS811. [Fuente: Adafruit].

Diagrama de Flujo para el Sensor de Dióxido de Carbono

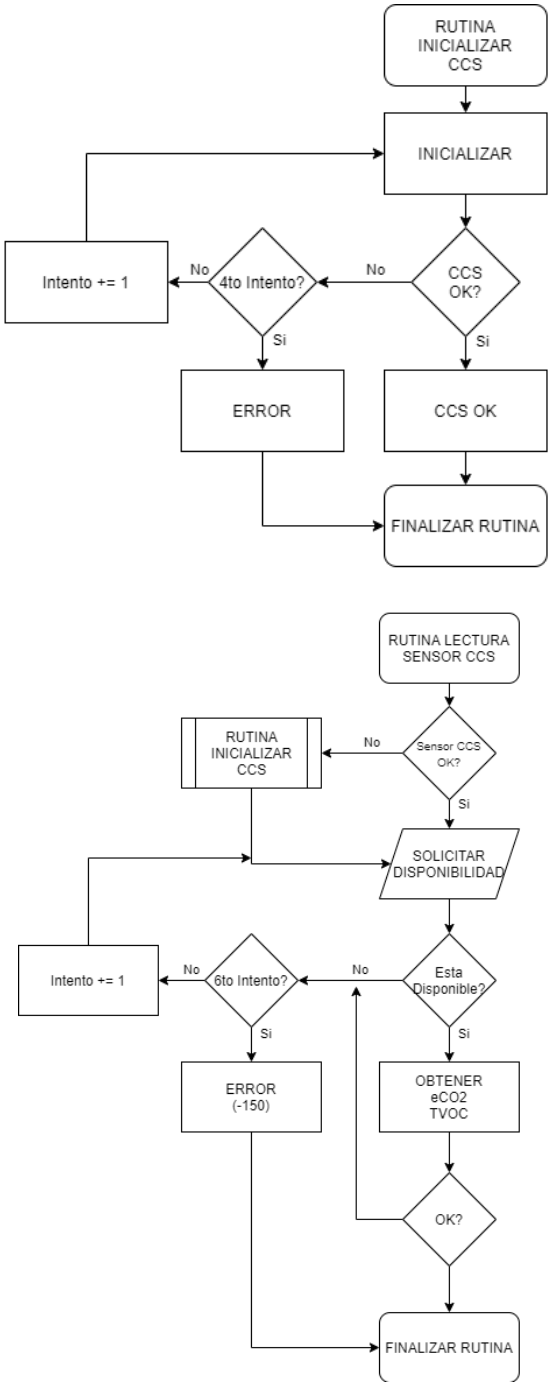


Ilustración 10 – Diagrama de Flujo para el CCS811. [Fuente: Autor].

3.2 Comunicación Inalámbrica

Para el desarrollo de la comunicación inalámbrica se determinan los datos a enviar, los parámetros de la comunicación y su desarrollo respectivo, se tiene en cuenta que se va a utilizar el módulo SIM808, el cual permite no solo la utilización de redes móviles (sea por SMS, llamada o GPRS), sino que además provee de interfaces bluetooth y GPS, convirtiéndolo en la opción más practica para la implementación del sistema de comunicación inalámbrica.

3.2.1 Datos a enviar

Los datos que la estación meteorológica va a enviar por medio de comunicación inalámbrica son:

- Temperatura
- Humedad
- Presión
- Altitud
- Velocidad del Viento
- Dirección del Viento
- Precipitación
- Intensidad de Luz
- Intensidad UV
- Equivalente CO2
- TVOC
- Latitud
- Longitud
- Fecha y Hora UTC
- Tiempo desde Encendido
- Batería

3.2.2 Parámetros de la comunicación

En primer lugar, se establece que la comunicación no debe ser “ciega”, esto significa que la estación debe ser capaz de identificar si el envío de datos fue exitoso por parte del módulo a utilizar (con una respuesta de dicho módulo).

El envío de datos solo se genera una vez se han realizado todas las mediciones, de este modo se asegura que la comunicación no interrumpa otros procesos, además de enviar siempre los datos más completos para el ciclo de lectura realizado (en lugar de, por ejemplo, enviar un dato de temperatura que sea más reciente que uno de humedad que también se envié en el momento)

Los datos serán enviados en un solo mensaje, esto quiere decir que todos los datos tomados por la estación son enviados uno tras otro en una misma rutina, y no por partes de forma esporádica (Ej: <temperatura, humedad y presión> en lugar de <temperatura>, <humedad>, <presión>, siendo el primero enviado en un momento dado con valores de un ciclo de medición completo en lugar de ir enviando conforme se va midiendo).

La recepción de datos puede darse en cualquier momento, siempre y cuando la estación meteorológica haya completado la labor que se encuentre realizando (Ej: si se encuentra midiendo temperatura, mide temperatura y luego procede a leer los datos, no se interrumpe la lectura del sensor)

La estación meteorológica debe tener una forma de guardar los datos en caso de que no haya comunicación inalámbrica, esto mediante un registrador de datos (datalogger). Además, dado que la conectividad puede no ser inmediata, y que aun en este caso la sincronización puede tomar tiempo, se debe mantener el tiempo que la estación lleva activa para poder usarse en caso de no tener una forma confiable de obtener la fecha y hora para su registro.

Desarrollo de la comunicación inalámbrica

3.2.3 SIM808

Imagen 10 – Módulo SIMCOM SIM808 con antena GPS y GSM incluida. [Fuente: SSDielect].

SIMCOM SIM808 – Características Técnicas	
Bandas	Quad 850/900/1800/1900 MHz
Bluetooth	Compatible con 3.0+EDR (Enhanced Data Rate)
Control por comandos AT	3GPP TS 27.007, 27.005 y Comandos mejorados AT de SIMCOM
Voltaje	3.4V – 4.4V
Comunicación	Serie
GPS	66 Canales de Adquisición, 22 de Rastreo
Costo	31.90 USD

Tabla 14 – Características técnicas SIM808. [Fuente: SIMCOM]

3.2.4 Comandos AT

Los comandos AT son comandos usados para controlar módems, fueron derivados de los comandos Hayes y se utilizan en todos los modem alámbricos o inalámbricos.

Los comandos AT utilizados y una descripción breve se dan en la siguiente tabla:

Comando AT	**Descripción**
AT	Comando para prueba de comunicación
AT+BTACPT	Acepta conexiones entrantes bluetooth
AT+BTCONNECT	Conecta a un dispositivo bluetooth
AT+BTCONNECTING	Indica una solicitud de conexión entrante bluetooth
AT+BTDISCONN	Desconecta la sesión actual de bluetooth
AT+BTPAIR	Acepta la solicitud o genera un emparejamiento con un dispositivo bluetooth
AT+BTPAIRING	Indica una solicitud de emparejamiento bluetooth
AT+BTPOWER	Enciende o Apaga el bluetooth
AT+BTSPPDATA	Indica datos entrantes por bluetooth
AT+BTSPPSEND	Envía datos por bluetooth (protocolo serie)
AT+BTSTATUS	Obtener el estado del bluetooth
AT+CGNSINF	Obtener datos del GPS
AT+CGNSPWR	Encender o Apagar el GPS
AT+CMGD	Eliminar mensajes de texto (SMS) en el dispositivo
AT+CMGF	Configurar modo de envío de mensaje de texto (SMS)
AT+CMGL	Obtener mensajes de texto (SMS) en el dispositivo
AT+CMGS	Envió de mensaje de texto (SMS)
CLIP	información de la llamada entrante
CMTI	Indica un mensaje de texto (SMS) entrante
RING	Indica una llamada entrante

Tabla 15 – Descripción de los comandos AT utilizados. [Fuente: Autor].

3.2.5 Forma de trabajo de la comunicación AT

Ilustración 11 – Flujo de trabajo de la comunicación AT. [Fuente: Autor].

Módulo SIM

3.2.6 Inicialización

Se envía un comando "AT" para verificar si hay comunicación con el módulo SIM, si lo hay, se procede a realizar la inicialización de los componentes bluetooth y GPS, para que el módulo SIM pueda funcionar correctamente. Si los componentes están en funcionamiento (ej, si se produjera una desconexión temporal entre el Arduino y el módulo SIM), simplemente toma la respuesta y continua con la rutina.

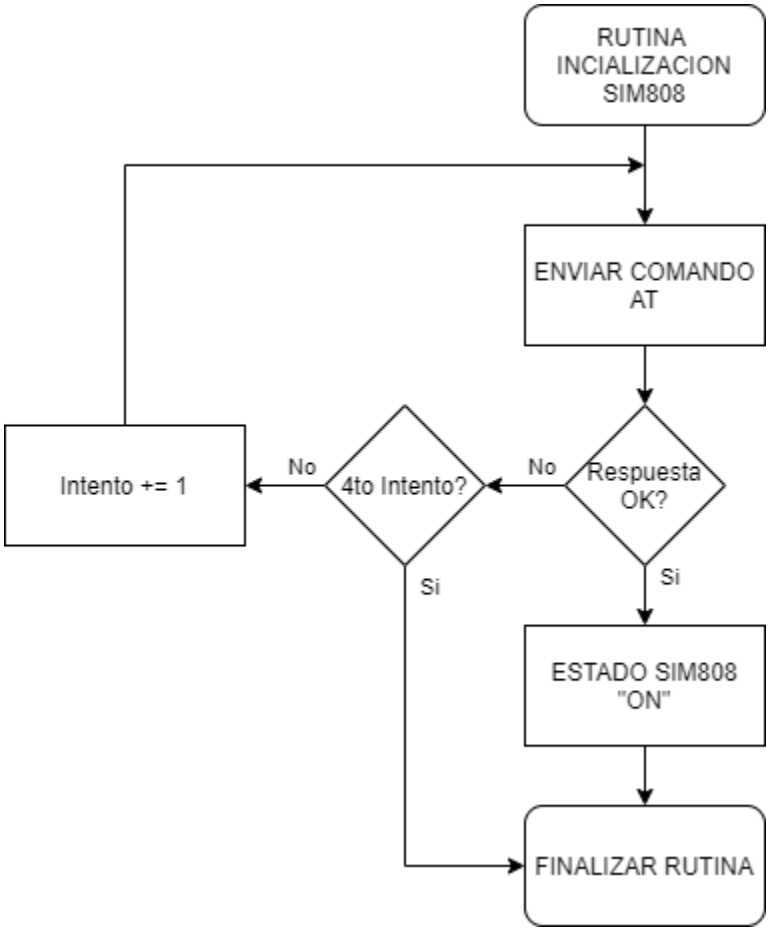


Ilustración 12 – Diagrama de Flujo para Inicialización del Módulo SIM. [Fuente: Autor].

3.2.7 Comunicación con el módulo SIM

Para comunicarse con el módulo SIM, se establecen dos rutinas, una de envió de datos, y la otra de recepción de datos. La recepción de datos incluye categorización de estos, lo que significa que se establece el parámetro recibido y se envía a la rutina respectiva. El envío de datos puede ser usado por cualquier rutina para enviar datos al módulo. De este modo se tiene una comunicación bidireccional entre el Arduino y el módulo SIM, asegurando un correcto funcionamiento de este. Es importante esto pues el módulo SIM es el encargado de enviar y recibir todo lo referente a bluetooth, SMS o llamadas para procesar en el Arduino.

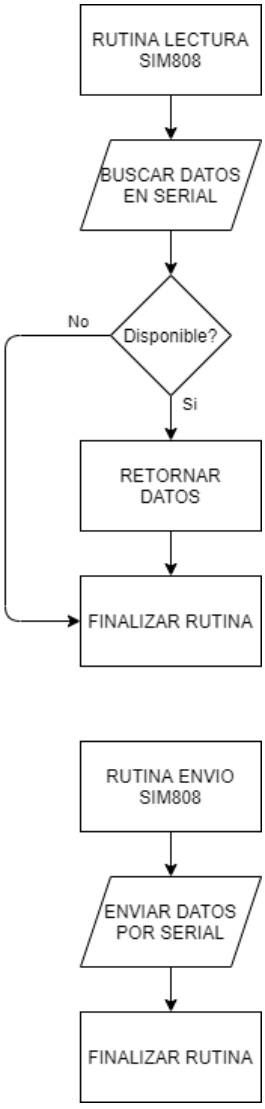


Ilustración 13 – Rutina de Envió y Recepción de datos SIM808. [Fuente: Autor].

Bluetooth

3.2.8 Inicialización

Se envía un comando para conocer el estado del bluetooth en el módulo SIM, si el mismo es indicativo de que esta encendido, no se hace nada, de lo contrario se envía el comando para encender el bluetooth y se espera a la respuesta.

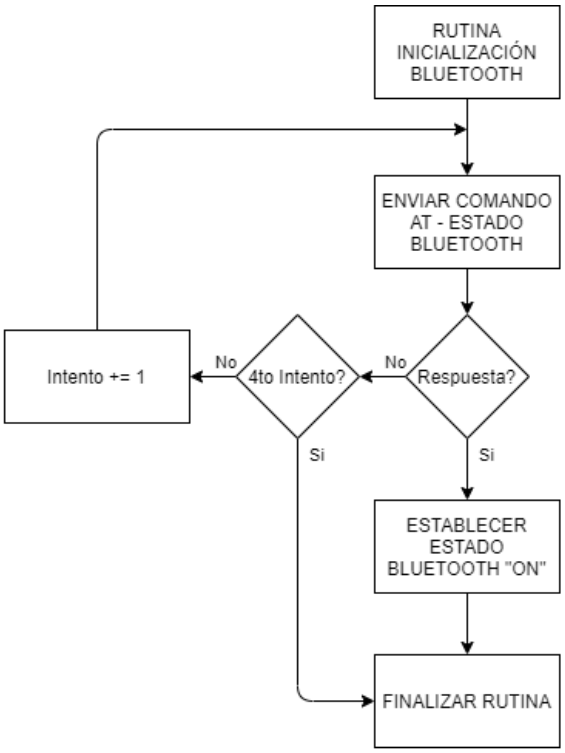


Ilustración 14 – Diagrama de Flujo inicialización Bluetooth. [Fuente: Autor].

3.2.9 Emparejamiento

Para que dos dispositivos bluetooth puedan conectarse y comunicarse, primero deben realizar un emparejamiento, esto es, el proceso por el que los dispositivos realizan una solicitud inicial de comunicación, la cual puede ser aceptada (lo que permite que se conecten y comuniquen) o negada (lo cual se puede considerar como un bloqueo de comunicaciones hasta que se realice un emparejamiento exitoso).

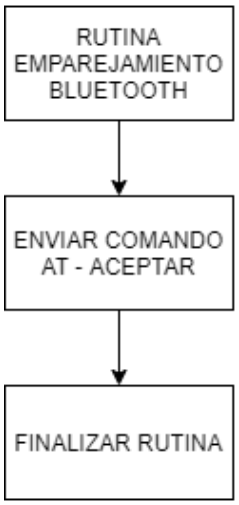


Ilustración 15 – Diagrama de Flujo Emparejamiento Bluetooth. [Fuente: Autor].

3.2.10 Conexión

Cuando los dispositivos bluetooth han realizado el emparejamiento, pero no se encuentran conectados, uno de los dos puede iniciar una solicitud de conexión, la cual puede ser aceptada (permitiendo el inicio de la transferencia de datos) o negada hasta otro intento exitoso.

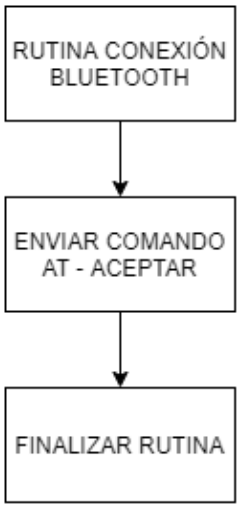


Ilustración 16 – Diagrama de Flujo Conexión Bluetooth. [Fuente: Autor].

3.2.11 Desconexión

Si la estación meteorológica se encuentra conectada a un dispositivo y se desea que otro se pueda conectar sin tener que desactivar la conexión bluetooth en el primero, se puede enviar un comando de desconexión con el cual la estación meteorológica se desconectara automáticamente del dispositivo actual.

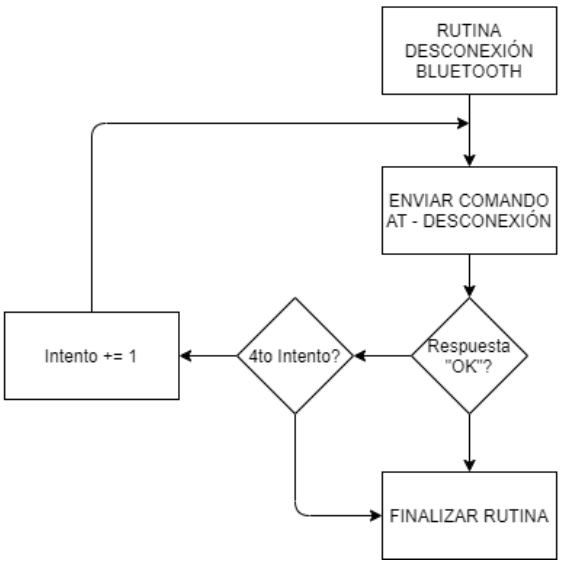


Ilustración 17 – Diagrama de Flujo Desconexión Bluetooth. [Fuente: Autor].

3.2.12 Comunicación Bluetooth

Una vez la estación meteorológica se encuentra emparejada y conectada al dispositivo, se procede a realizar el envío y recepción de datos por medio de bluetooth, la forma en que se realiza él envió de datos es similar a la que se emplea para él envió por medio de mensajes de texto.

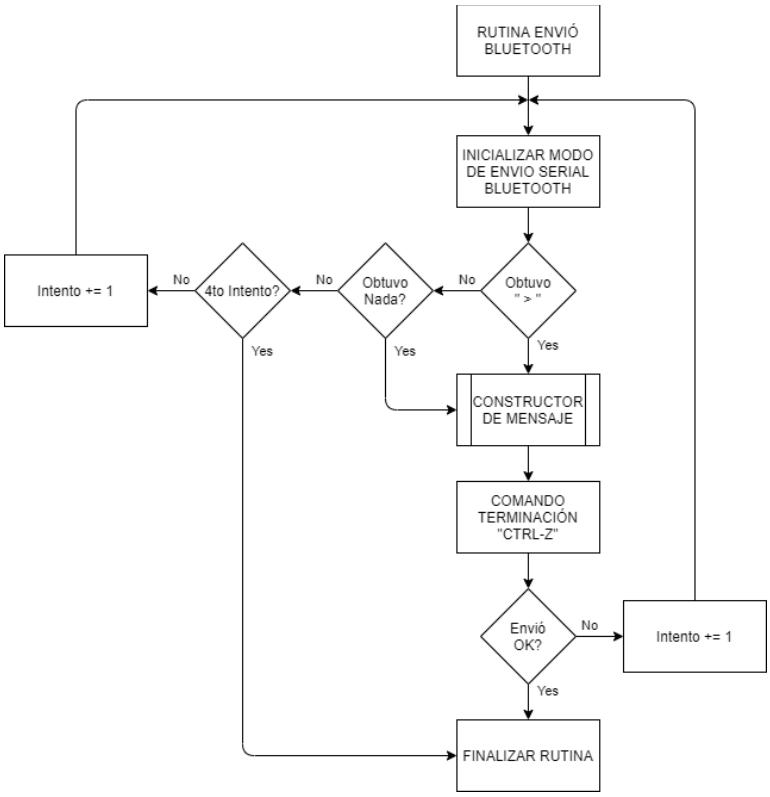


Ilustración 18 – Diagrama de Flujo para envió de datos Bluetooth. [Fuente: Autor].

3.2.13 Auto Respuesta

La estación meteorológica puede generar respuestas automáticas por medio de mensajes de texto a un número celular, este número puede ser obtenido por un comando bluetooth (enviado desde la aplicación), una llamada telefónica o un mensaje de texto enviado al número que se encuentre en uso en la estación meteorológica.

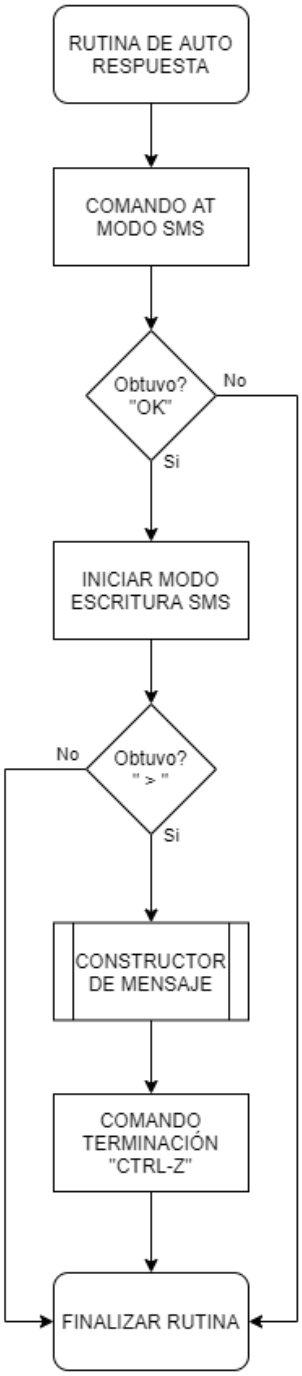


Ilustración 19 – Diagrama de Flujo para Auto respuesta. [Fuente: Autor].

La estación meteorológica puede recibir y auto colgar llamadas, tomando esto como un comando para enviar un mensaje de texto al número telefónico del que recibió la llamada (generar una auto respuesta). Así mismo, si la estación meteorológica recibe un mensaje de texto, este también será tomado como un comando para generar una auto respuesta al número del que proviene el SMS.

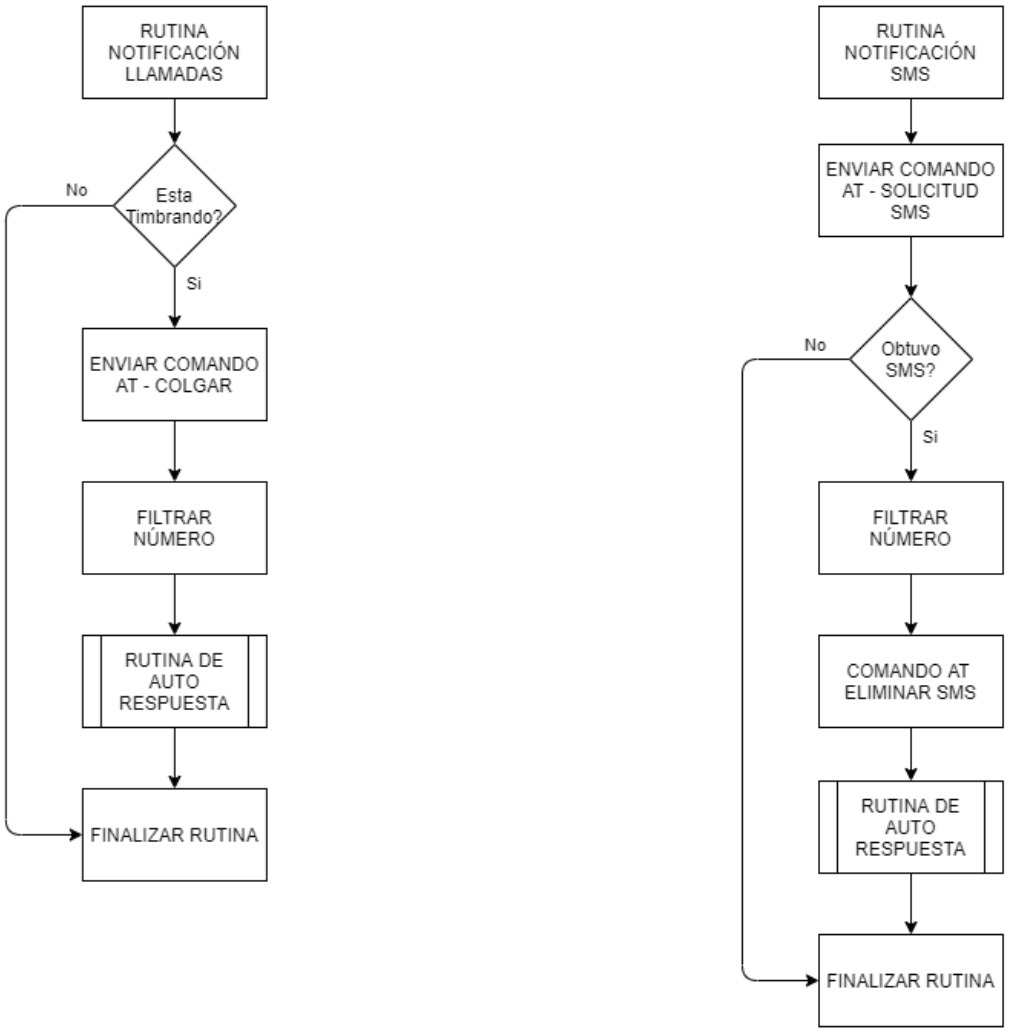


Ilustración 20 – Diagrama de Flujo Notificación de Llamada y SMS. [Fuente: Autor].

3.2.14 GPS

El módulo SIM808 integra un receptor de señales GPS, con lo que, al conectar una antena y energizar, se puede establecer comunicación con satélites GPS para obtener datos como latitud, longitud, altitud, velocidad, dirección, entre otros.

3.2.15 Trama NMEA

El módulo SIM808 permite la entrega de distintas sentencias NMEA, las cuales contienen distintos tipos de información respecto al estado y ubicación del GPS:

Mensaje	**Descripción**
GGA	Posición, Hora y tipo de arreglo “enganche”
GSA	Modo de operación del receptor GNSS, satélites usados en la posición y valores DOP
GSV	Numero de satélites GNSS en vista, ID, elevación, azimut y valores SNR
RMC	Fecha, Hora, Posición, Dirección y Velocidad

Tabla 16 – Tipo de mensajes NMEA proporcionados por el SIM808 y sus descripciones. [Fuente: SIMCOM]

Para este caso se va a hacer uso de la sentencia RMC, puesto que es la que permite obtener todos los datos esenciales para el usuario.

3.2.16 Trama de datos GPS del Módulo SIM

Para poder obtener los datos provenientes del GPS, se hace uso del comando avanzado del módulo SIM 808, al emplear los comandos GNS para la obtención de los datos de forma efectiva, con lo que el módulo entrega la siguiente trama de datos:

Información de navegación GNSS obtenida de sentencia NMEA	
Comando a Enviar	Respuesta
AT+CGNSINF	**+CGNSINF: <Estado del GNSS>, <>, <Fecha y Hora UTC>, <Latitud>, <Longitud>, <Altitud MSL>, <Velocidad respecto al suelo>, <Dirección respecto al suelo>, <Modo de >, <Reservado 1>, <HDOP>, <PDOP>, <VDOP>, <Reservado 2>, <Satélites GNSS en vista>, <Satélites GNSS Usados>, <Satélites GLONASS usados>, <Reservado 3>, <C/N0 máximo>, <HPA>, <VPA>**
	OK

Tabla 17 – Respuesta del SIM808 a una solicitud de información del GPS. [Fuente: SIMCOM].

Como se puede observar, el identificador “+CGNSINF” indica que se está recibiendo la respuesta adecuada al comando dado, luego, la trama de datos viene separada (delimitada) por coma “,” por lo que usamos este como el indicativo para obtener las variables respectivas, teniendo en cuenta que el módulo siempre entrega los datos en el orden mostrado.

Teniendo en cuenta lo anterior, los datos de interés a obtener del GPS y sus posibles valores son:

Entero (INT)	
GPS_RunStatus	1 = ON, 0 = OFF
GPS_FixStatus	1 = Location Fix, 0 = Location Not Fix
UTC_DateTime	YYYYMMDDHHMMSS
GPS_InView	0 – 99
GNSS_InView	0 – 99
FLOAT	
Latitude	-99.9999
Longitude	±99.9999
MSL_Altitude	999.99 [Metros]
Course_Ground	359.99 [Grados]

Tabla 18 – Datos a tener en cuenta del GPS. [Fuente: Autor].

De esta forma se define la comunicación con el módulo y la obtención de datos referentes al GPS de la siguiente forma:

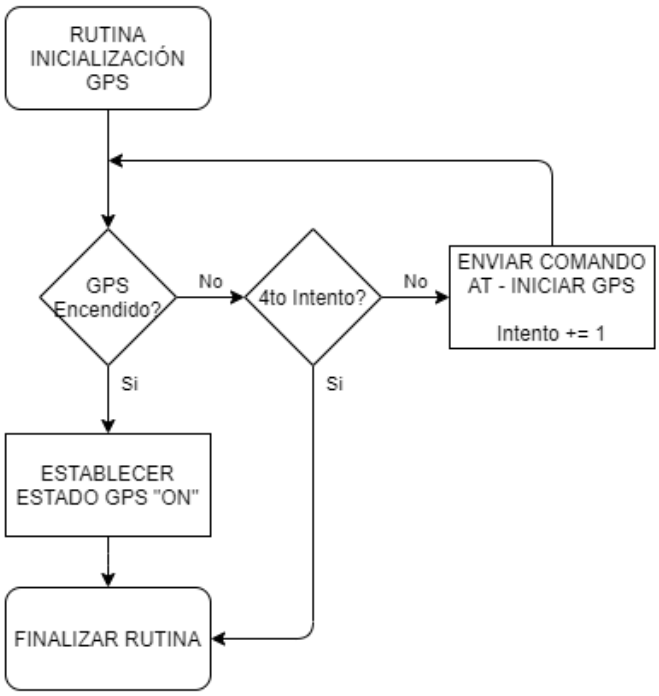


Ilustración 21 – Diagrama de flujo para inicialización del GPS. [Fuente: Autor].

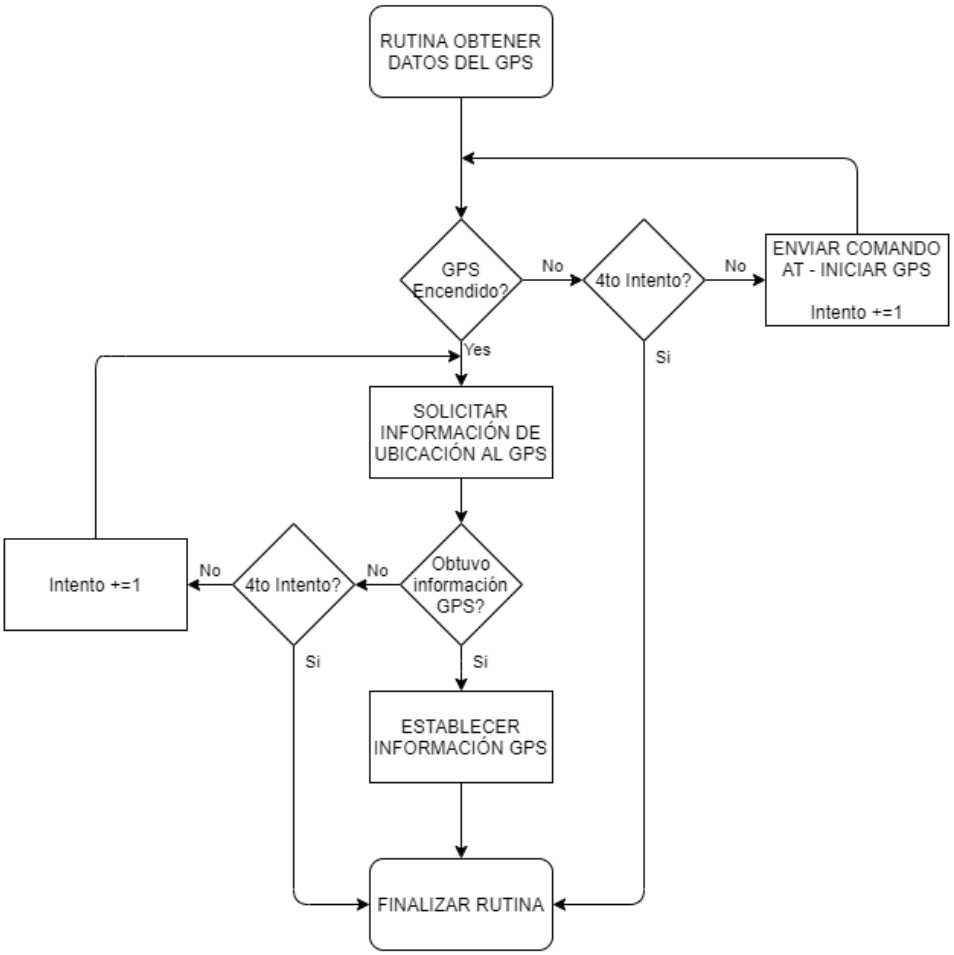


Ilustración 22 – Diagrama de flujo para obtener datos del GPS. [Fuente: Autor].

3.2.17 Tiempo de Actividad (Uptime)

Dado que, en ocasiones, la fecha y hora disponibles pueden no ser las correctas (ej: primer uso, batería de módulo RTC descargada y módulo SIM no sincronizado), se añade el tiempo el tiempo de actividad a los datos guardados, de manera que, en caso de no tener una fecha y hora definida, y/o que ocurra una sincronización que cambie la misma, se pueda tener una forma de establecer la relación entre estas y de esa forma mantener los registros del clima de forma ordenada.

Código

```
void int_TIME(){                          Declaración de la función.
  curr_mill = millis();                   Se establece el tiempo de actividad actual (ms).
  S = curr_mill / 1000;                   Se obtienen los segundos.
  if (S > 59){
    S %= 60;                                          1s = 1000ms
  }
```

```
  M = S / 60;                                   Se obtienen los minutos.
  if(M > 59){
    M %= 60;                                                  1mín = 60s
  }
  H = M / 60;                                   Se obtienen las horas.
  if (H > 24){
    H %= 24;                                                  1h = 60mín
  }
 }
```

Tabla 19 – Código para el cálculo del tiempo de actividad de la estación meteorológica. [Fuente: Autor].

3.2.18 MicroSD

Para poder dar autonomía al sistema, se implementa un datalogger por medio de guardado de los datos en una MicroSD, con lo cual, en caso de no disponer de conectividad bluetooth, o de no disponer de una Sim Card, permita el registro de datos para su posterior análisis. La MicroSD es obligatoria para el funcionamiento de la estación, puesto que es la única forma de garantizar el registro de los datos.

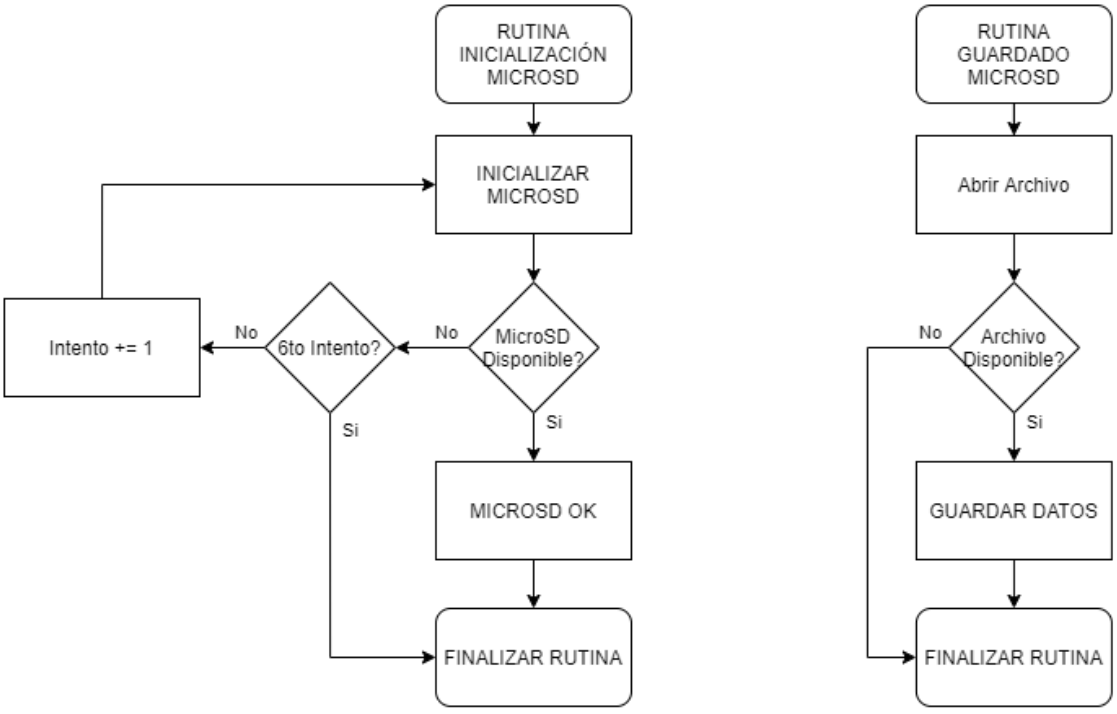


Ilustración 23 – Diagrama de flujo para inicializar y guardar datos en la MicroSD. [Fuente: Autor].

3.2.19 Microcontrolador

En base a los sensores escogidos, y una vez definido el diagrama de flujo para la estación, se hace una búsqueda de microcontroladores disponibles en el mercado

Microcontrolador	E/S Digital	Entradas Análogas	Corriente por Pin	FLASH	SRAM	EEPROM	CLK MHz
PIC18F4550	35	13	25mA	32KB	2KB	256 b	48
PSoC 5LP – CY8CKIT-059	62	3	25mA	256KB	64KB	2KB	80
EK -TM4C123GXL	40	12	25mA	256KB	32KB	2KB	80
Arduino UNO R3 (ATmega328P)	14	6	20mA	32KB	2KB	1KB	16
Arduino MEGA R3 (ATmega2560)	54	16	20mA	256KB	8KB	4KB	16

Tabla 20 – Microcontroladores encontrados durante revisión bibliográfica. [Fuente: Autor / Datasheet de cada dispositivo].

Al realizar la selección, se descartan el PIC18F4550 y el Arduino UNO R3, puesto que poseen grandes limitaciones en cuanto a memoria RAM y almacenamiento FLASH que podrían limitar o imposibilitar el desarrollo de lo planteado. La PSoC si bien una buena elección, se encontró durante la búsqueda que la misma se encuentra fuera de stock, finalmente la Tiva C (EK-TM4C123GXL) si bien estaba en disponibilidad, no ofrecía prestaciones muy diferentes de la del Arduino Mega R3, y por el contrario podía costar más del doble de lo que cuesta el Arduino Mega.

Arduino MEGA R3 (ATmega2560)

Imagen 11 – Arduino MEGA R3. [Fuente: Arduino.cc].

Arduino Mega R3 – Características Técnicas

Voltaje de Entrada	7 – 12V [Regulado DC], 5V [USB o entrada]
Voltaje de Operación	5V
Entradas / Salidas	54 Digitales, 16 Análogas, 20mA por Pin

Memoria Flash	256KB
Memoria SRAM	8KB
Memoria EEPROM	4KB
Velocidad de Reloj	48MHz
Costo	14USD (China) – 40.30USD (Fabricada en Italia)

Tabla 21 – Características técnicas Arduino Mega R3. [Fuente: Arduino.cc / Autor].

3.3 Software

La estación meteorológica cuenta con un software que le permite comunicarse con esta, y poder visualizar los datos de forma inalámbrica.

Para esto, se hace necesario determinar las plataformas a las que se va a ofrecer el software, las especificaciones de este y su funcionalidad

3.3.1 Determinación de plataformas a ofrecer

Para determinar las plataformas (Sistemas operativos) a ofrecer, se tienen en cuenta los siguientes aspectos:

- Facilidad para desarrollar en la plataforma
- Porcentaje de Mercado
- Disponibilidad de los elementos para desarrollo por parte del Autor

Teniendo en cuenta estos aspectos, se procede a establecer las plataformas que actualmente se encuentran en el mercado con algún porcentaje significativo de uso, a lo que se obtienen:

3.3.2 Facilidad para desarrollar en la plataforma

Se encontró que todas las plataformas ofrecen diversidad de entornos de desarrollo (Visual Studio, Android Studio, XCode, Xamarin, entre otros) que se adaptan a las necesidades además de proveer de documentación.

3.3.3 Porcentaje de Mercado

Para determinar las plataformas, se tendrá en cuenta el porcentaje de mercado de estas, esto es, la cantidad de usuarios que tiene cada plataforma, de forma que se maximiza la cantidad de personas que pueden acceder al software.

Si bien no hay muchas plataformas distintas que sean usadas, las versiones de estas si son muchas y con distintos porcentajes de mercado (Ej: Windows 7, Windows 10, Android 5.0, Android 7.0, iOS13, entre otros).

Buscando en la página StatCounter se encuentra la siguiente distribución del mercado global:

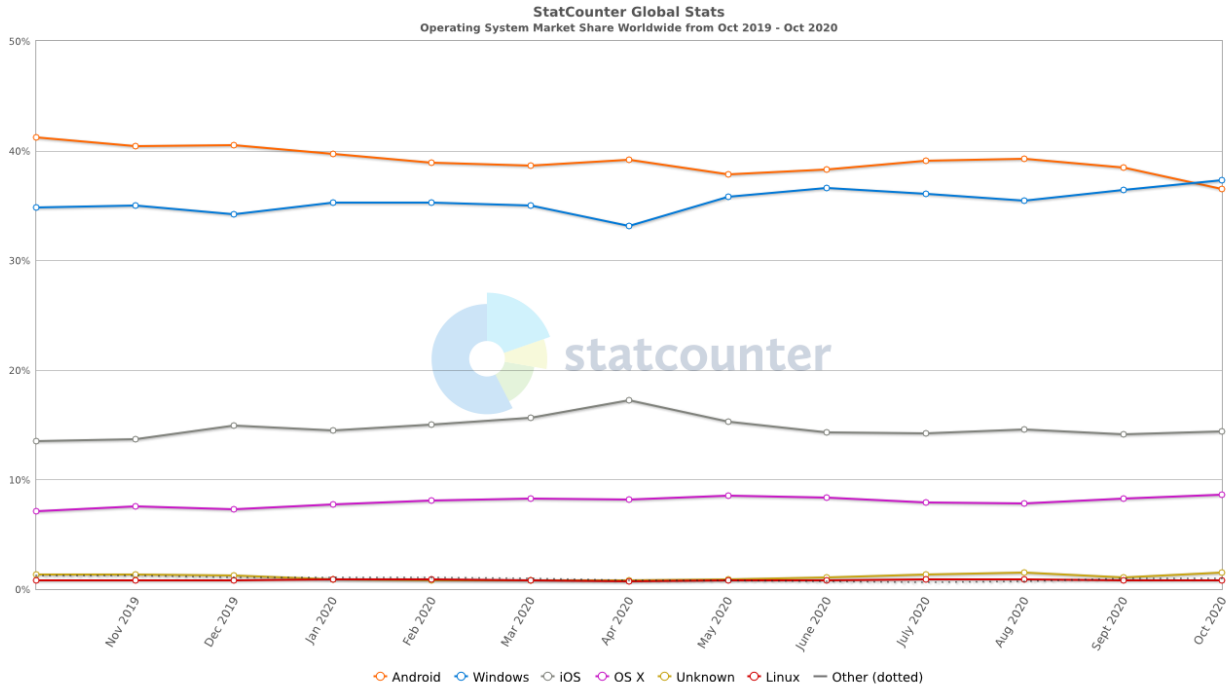


Imagen 12 – Datos de porcentaje de mercado de sistemas operativos (global). [Fuente: StatCounter].

- Windows (37.3% Mundial a octubre de 2020)
- Android (36.46% Mundial a octubre de 2020)
- iOS (14.36% Mundial a octubre de 2020)
- OS X (8.62% Mundial a octubre de 2020)

Antes de continuar con la distribución del mercado, se a de tener en cuenta el siguiente factor para determinar las plataformas a ofrecer, teniendo en cuenta que las posibilidades de adquisición de herramientas para desarrollo y prueba de las diferentes plataformas no necesariamente tienen una facilidad de acceso para el autor y/o las personas.

Disponibilidad de los elementos para desarrollo por parte del Autor

Si bien las plataformas ofrecen facilidades para desarrollo y poseen todas un porcentaje de mercado significativo, la posibilidad de ejecutar el código y probar la funcionalidad de este se ve limitada a tener el hardware, lo que conlleva a la dificultad de acceso al desarrollo de ciertas plataformas si no se cuenta previamente con dispositivos de estas

- Windows: Disponible, costos de adquisición suelen ser variados, incluyendo costos bajos para cualquier persona.

- Android: Disponible, costos de adquisición suelen ser variados, incluyendo costos bajos para cualquier persona.
- iOS: No disponible, costo estimado de adquisición alto para algunos sectores de la población.
- OS X: No disponible, costo estimado de adquisición alto para algunos sectores de la población.

Teniendo en cuenta lo anterior, se elige trabajar con las plataformas Windows y Android, sin embargo, dado que ambas plataformas poseen demasiadas versiones, se hace necesario mirar el porcentaje de mercado para identificar las versiones de sistema operativo a las que se les ofrece soporte.

3.3.4 Windows

Para seleccionar la versión mínima a soportar, se busca el porcentaje de mercado global y local, obteniendo los siguientes resultados:

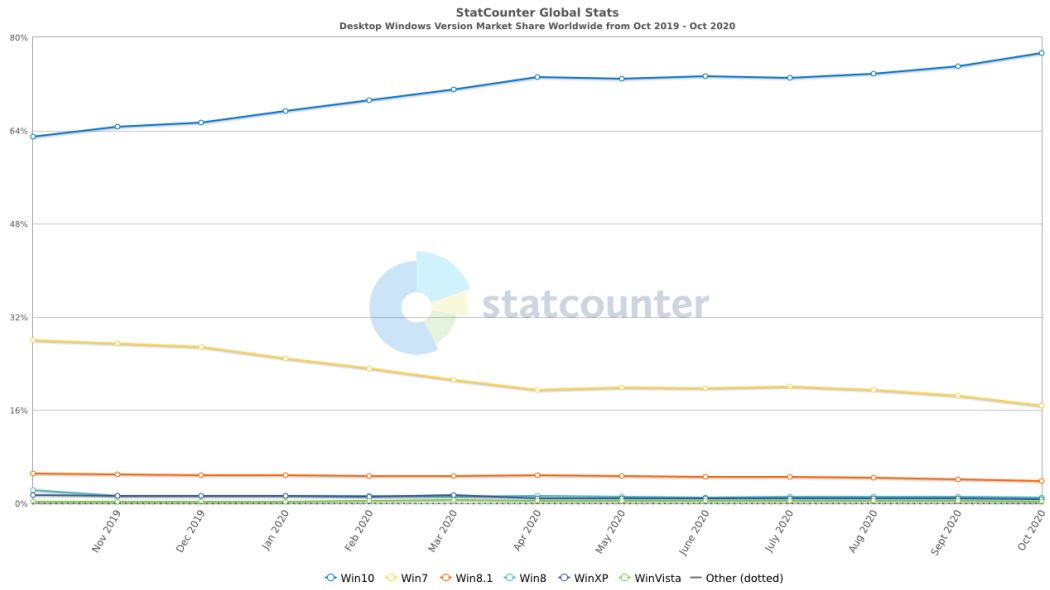


Imagen 13 – Datos de porcentaje de mercado (global) para sistemas operativos Windows. [Fuente: StatCounter].

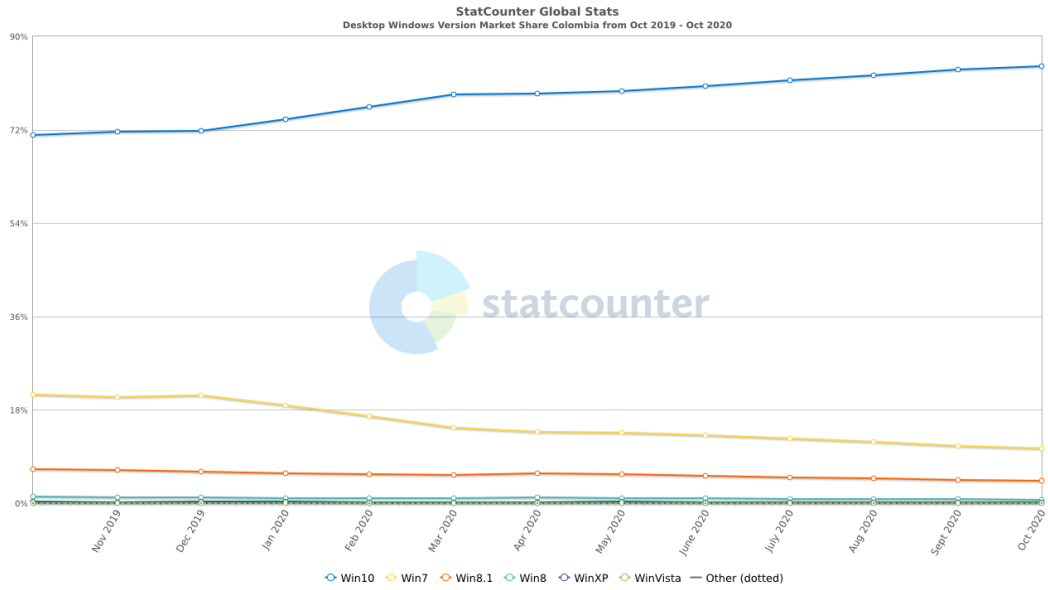


Imagen 14 – Datos de porcentaje de mercado (Colombia) para sistemas operativos Windows. [Fuente: StatCounter].

Versión del S.O.	Porcentaje de Mercado Global	Porcentaje de Mercado Local
Windows 10	77.31%	84.29%
Windows 7	16.8%	10.56%
Windows 8.1	3.79%	4.33%
Windows 8	1%	0.71%
Windows XP	0.71%	0.09%
Windows Vista	0.35%	0.02%

Tabla 22 – Porcentaje de mercado Global y Local (Colombia) para Windows en sus diferentes versiones. [Fuente: StatCounter].

Viendo los datos, se observa que la versión mínima para poder acceder a la mayor cantidad de personas es Windows 7, con la cual se puede dar soporte a Windows 8, Windows 8.1 y Windows 10, cubriendo así el 98.9% de los usuarios globales y el 99.89% de los usuarios del mercado local (Colombia).

3.3.5 Android

Para seleccionar la versión mínima a soportar, se busca el porcentaje de mercado global y local, obteniendo los siguientes resultados:

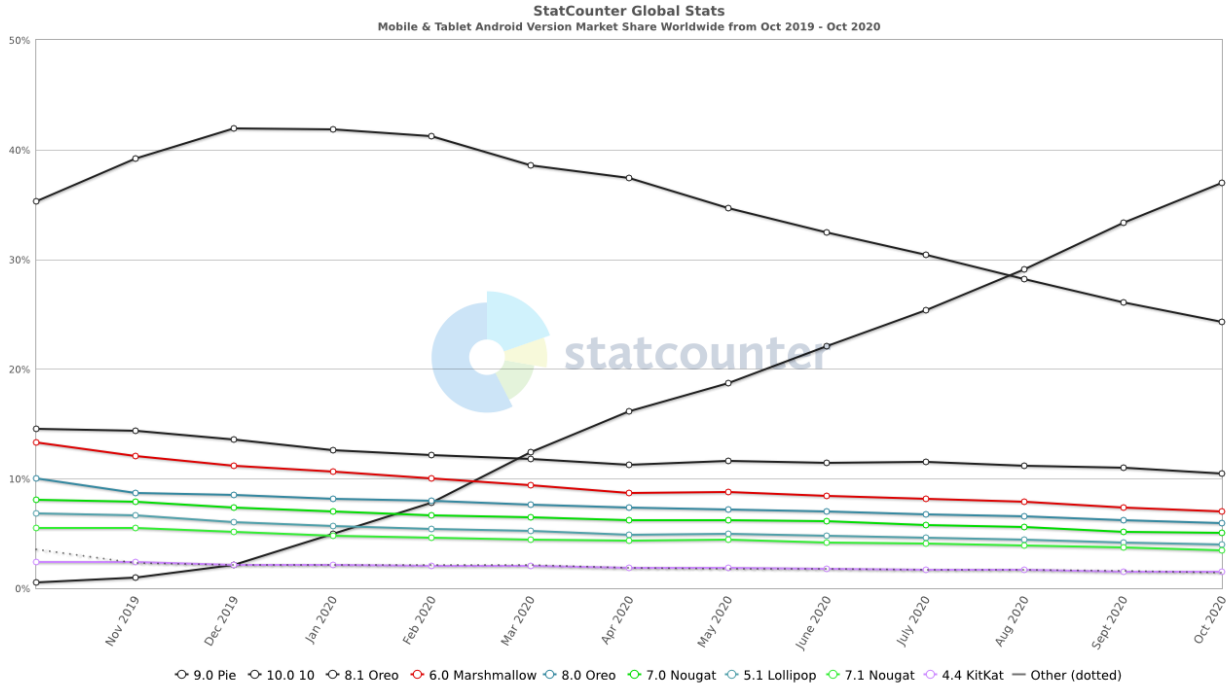


Imagen 15 – Datos de porcentaje de mercado (Global) para Android. [Fuente: StatCounter].

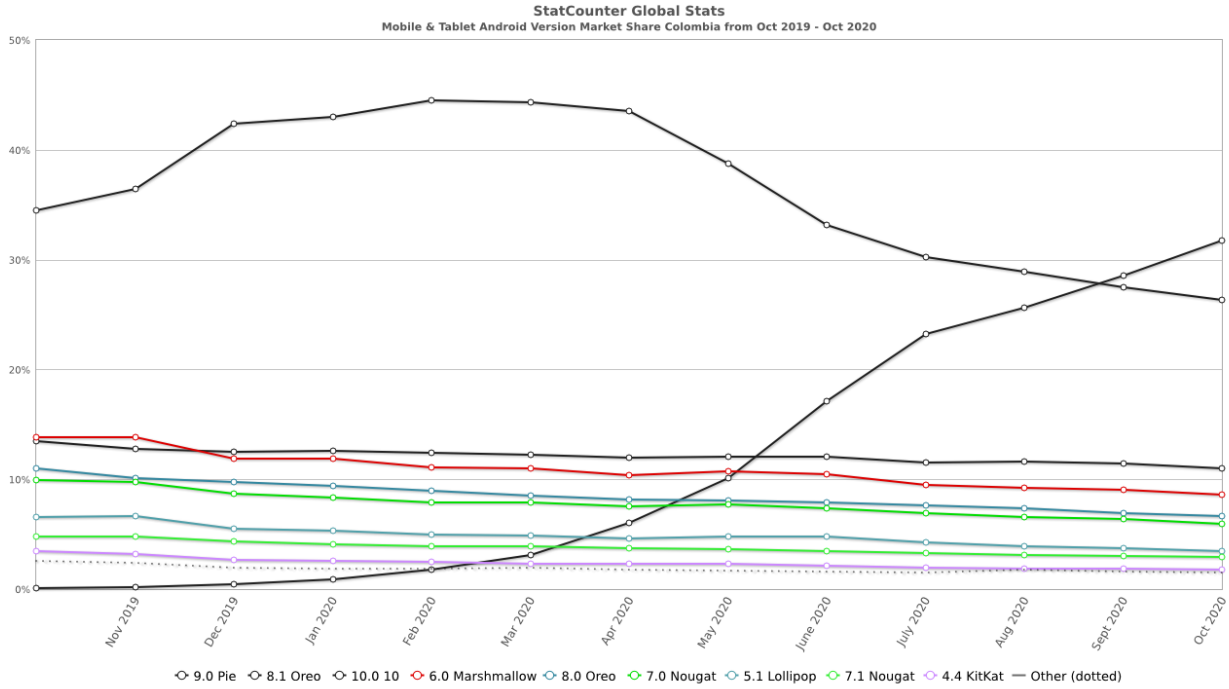


Imagen 16 – Datos de porcentaje de mercado (Colombia) para Android. . [Fuente: StatCounter].

Versión del S.O.	Porcentaje de Mercado Global	Porcentaje de Mercado Local
10.0.10	36.93%	31.77%
9.0	24.3%	26.31%
8.1	10.43%	10.99%
6.0	7.01%	8.64%
8.0	5.98%	6.67%
7.0	5.01%	5.96%
5.1	3.98%	3.47%

Tabla 23 – Porcentaje de mercado Global y Local (Colombia) para Android en sus diferentes versiones. [Fuente: StatCounter].

En base a estos datos, se determina que la plataforma mínima de Android para dar soporte es 5.1 Lollipop, puesto que aun cuenta un porcentaje significativo de usuarios, que las aplicaciones diseñadas para este sirven en las otras versiones y de este modo se cubre un 93.64% de los usuarios globales y un 93.81% de usuarios locales (Colombia).

3.3.6 Especificación de las aplicaciones

El software debe obtener de la estación meteorológica los siguientes datos más esenciales, además de mostrar al usuario los datos más esenciales del mismo.

- Temperatura
- Humedad
- Presión
- Altitud
- Velocidad del Viento
- Dirección del Viento
- Precipitación
- Intensidad de Luz
- Intensidad UV
- Equivalente CO2
- TVOC
- Latitud
- Longitud
- Fecha y Hora UTC
- Tiempo desde Encendido
- Batería

Adicional a esto, como fue mencionado anteriormente, debe poder funcionar en los sistemas operativos Windows 7 y Android 5.1 (en condiciones de fabrica) además de verificar el funcionamiento en sistemas más modernos para garantizar que el software funciona de forma correcta.

3.3.7 Determinación de la funcionalidad a ofrecer

Para ambas plataformas se plantea la siguiente funcionalidad

- Visualizar en tiempo real los datos de la estación meteorológica
- Comunicación por Bluetooth con la estación

Adicionalmente para la aplicación en Windows se considera la siguiente funcionalidad

- Visualizar de forma gráfica los datos
- Visualizar los datos tomados en intervalos de 10 minutos
- Lista de datos obtenidos desde que se inició la aplicación
- Exportar dichos datos a un archivo .csv (valores separados por coma)

3.3.8 Aplicación en Windows

La aplicación en Windows se desarrolla usando el entorno de desarrollo Visual Studio, se ha decido por utilizar el sistema de interfaz de usuario WPF (Fundación de Presentación de Windows) el cual hace el renderizado de la interfaz por Hardware de forma nativa (lo que implica un mejor rendimiento y una menor complejidad de código al no tener que realizar interfaces con DirectX o OpenGL de forma directa).

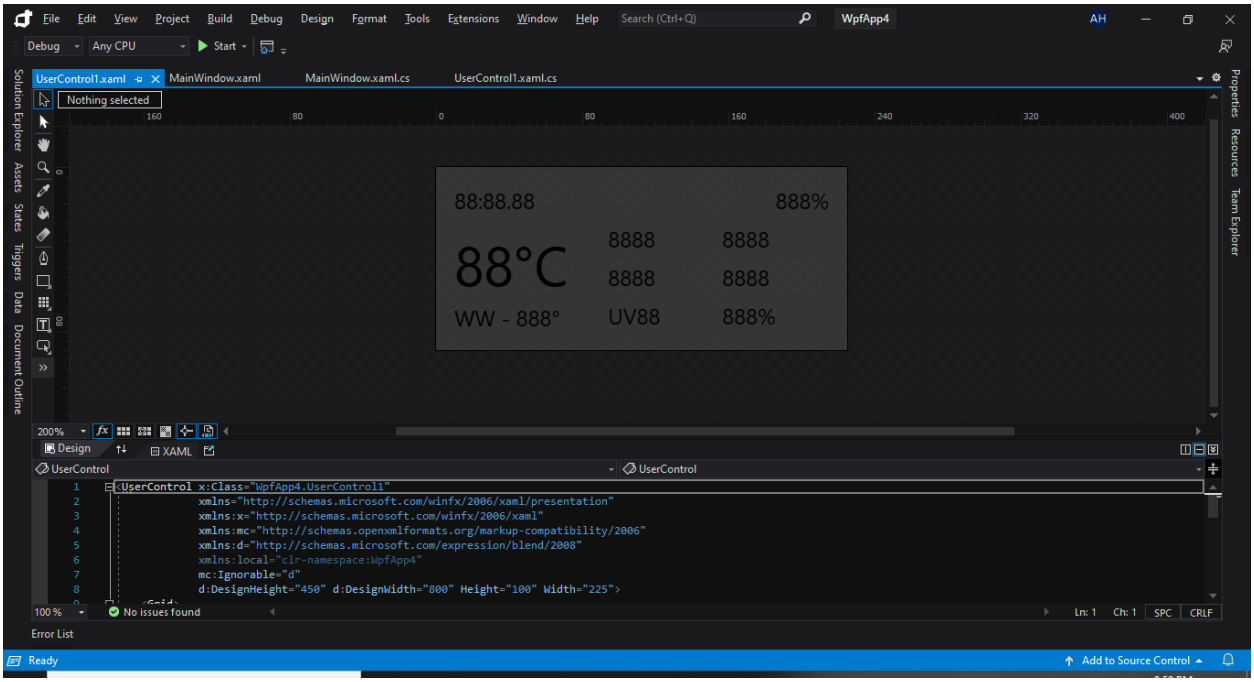


Imagen 17 – Interfaz de Usuario de Visual Studio Blend. [Fuente: Autor].

Otra característica de WPF, es que permite programar la parte visual con código XAML, lo que proporciona mayor posibilidad de personalizar las interfaces y/o controles, mientras el código de la aplicación sigue siendo en C# o en Visual Basic, dando la posibilidad en algunos casos de permitir separar al programador del diseñador gráfico.

Finalmente se hace uso de la librería 32feet.net, esta permite utilizar la interfaz bluetooth desde aplicaciones de Visual Basic o C#, dado que ninguno de los dos provee de una forma para acceder a la misma (solamente se provee por medio de C++ o en aplicaciones UWP donde se entrega una interfaz para trabajar con dispositivos Bluetooth LE).

3.3.9 Aplicación en Android

La aplicación en Android se desarrolla usando el entorno de desarrollo Android Studio, el mismo permite el desarrollo de aplicaciones para distintos tamaños de pantalla, además de mostrar visualmente el desarrollo de la aplicación, el mismo permite que la lógica del programa sea en el lenguaje de programación JAVA o Kotlin de forma intercambiada, para este proyecto se usa JAVA.

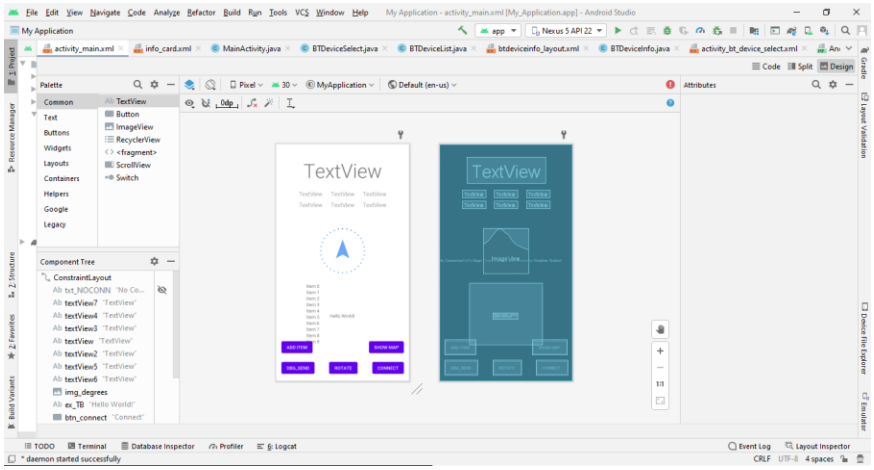


Imagen 18 – Interfaz de Usuario de Android Studio. [Fuente: Autor].

La interfaz de usuario utiliza código XML, permitiendo que el diseñador y el programador puedan trabajar en un mismo proyecto sin necesidad de tener el conocimiento completo del otro, a diferencia de Windows, se ofrece una interfaz integrada para interactuar con el bluetooth desde el dispositivo sin necesidad de librerías extra.

3.4 Energía

Puesto que la estación meteorológica requiere de un funcionamiento constante, y dado que no siempre se garantiza la presencia de fluido eléctrico, se dispone de un sistema fotovoltaico de baja escala para proporcionar una autonomía a este, de forma que incluso de noche, en lugares remotos o sin fluido eléctrico se pueda seguir utilizando la estación meteorológica para la captura de datos. Para esto, es necesario realizar el dimensionamiento de los componentes, de forma que sabiendo su consumo se pueden estimar los parámetros necesarios para el panel solar y las baterías.

3.4.1 Dimensionamiento

El dimensionamiento consiste en calcular el consumo de los dispositivos que componen un sistema, de manera que al saber el consumo se puede conocer las características que debe tener un panel solar (voltaje, amperaje, cantidad), regulador, baterías, etc. Dado que el voltaje máximo que se requiere en cualquier componente de la estación meteorológica es de 5V, se procede a realizar el dimensionamiento del consumo (corriente) el cual es el siguiente:

Ítem	Típica (µA)	Máxima (µA)
SHT30	2	1500
BH1750	120	190
BMP280	2.8	1120
BMM150	170	500
DS18B20	1000	1500
UVM-30A	60	100
PCF8563T	0.25	1.7
MicroSD	200	200000
SIM808	200000	2000000
Reserva	50000	50000
Total	251555.1	2254912
Total (en mA)	251.5551	2254.914

Tabla 24 – Dimensionamiento del consumo de corriente de la estación meteorológica. [Fuente: Autor].

Nota: El amperaje del módulo SIM808 (2A) solo es requerido en disparos (no es constante, solo ocurre durante la comunicación celular), sin embargo, dado que es requerido garantizarlos, se decide usar como valor constante para el dimensionamiento.

Del dimensionamiento anterior, se obtiene que:

Capacidad del Panel Solar	Número de Paneles Solares
250mAh	1
300mAh	0.83
500mAh	0.5

Capacidad de la Batería	Número de Baterías
1000mAh	2.26
2000mAh	1.13
3000mAh	0.75

Tabla 25 – Estimado de la cantidad de baterías y paneles solares en base a su capacidad. [Fuente: Autor].

Con lo cual se obtiene lo necesario para implementar el sistema de energía de la estación meteorológica.

Resultados

4.1 Validación de los sensores

Inicialmente se verifica con el monitor serial de Arduino, la comunicación con cada uno de los sensores, al verificar su funcionamiento se procede a la toma de datos en forma global (esto es, obtener la lectura de todos los sensores al mismo tiempo) durante un tiempo indeterminado para verificar que efectivamente funcionen de forma correcta, y se determine si es necesario o no, establecer cambios que permitan "normalizar" la salida de los sensores comparado con los datos del ambiente.

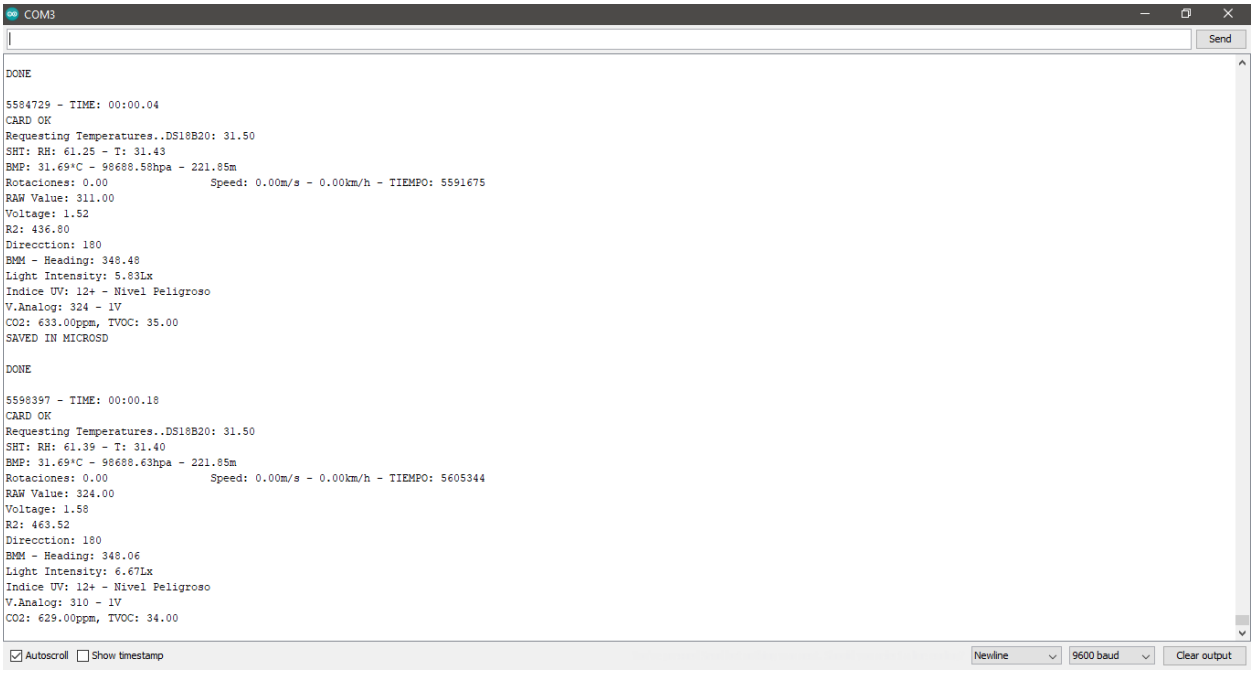


Imagen 19 – Datos obtenidos por el monitor serial de Arduino. [Fuente: Autor].

Para todo caso, se tomaron datos cada 10 minutos por un intervalo de 2 horas, una muestra en la mañana y otra en la noche.

4.1.1 Temperatura

De los datos se puede ver que el comportamiento de los tres sensores (DS18B20, SHT30 y BMP280) es similar con el comportamiento del control, siendo el DS18B20 el que más susceptibilidad presenta a las variaciones de temperatura, y en todo caso, estando siempre a unos ±0.5°C del control. De destacar es la termocupla, la cual no mostro respuesta a las variaciones de la temperatura (posiblemente por el tipo de calibración que tiene).

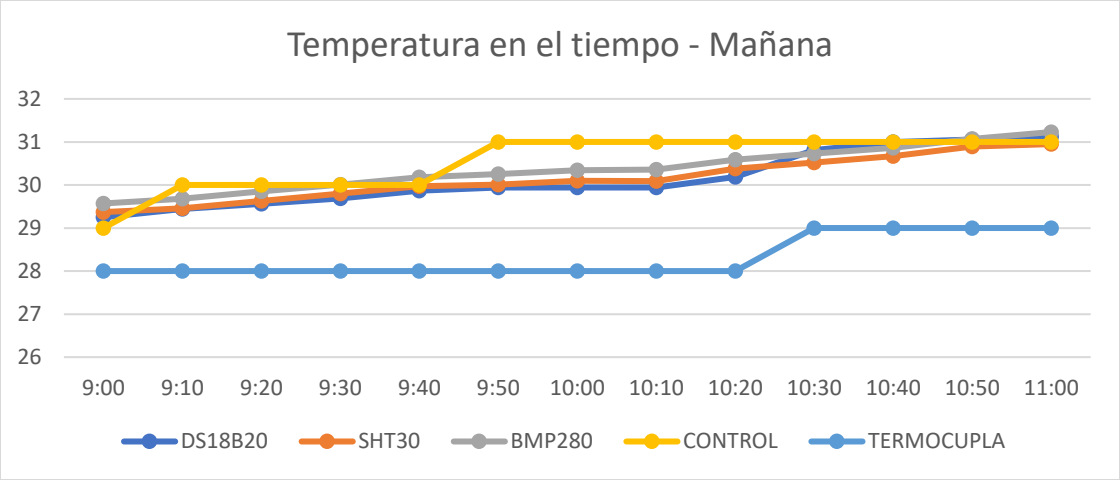


Ilustración 24 – Temperatura en un intervalo de 2 horas en una mañana. [Fuente: Autor].

Al ver los datos de la temperatura en la noche, se evidencia que la diferencia entre los distintos sensores disminuye, un posible factor es el tiempo de funcionamiento que permite que los sensores tengan medidas más estables y precisas. Dado lo mostrado por los datos, se decide realizar corrección alguna respecto de estas mediciones.

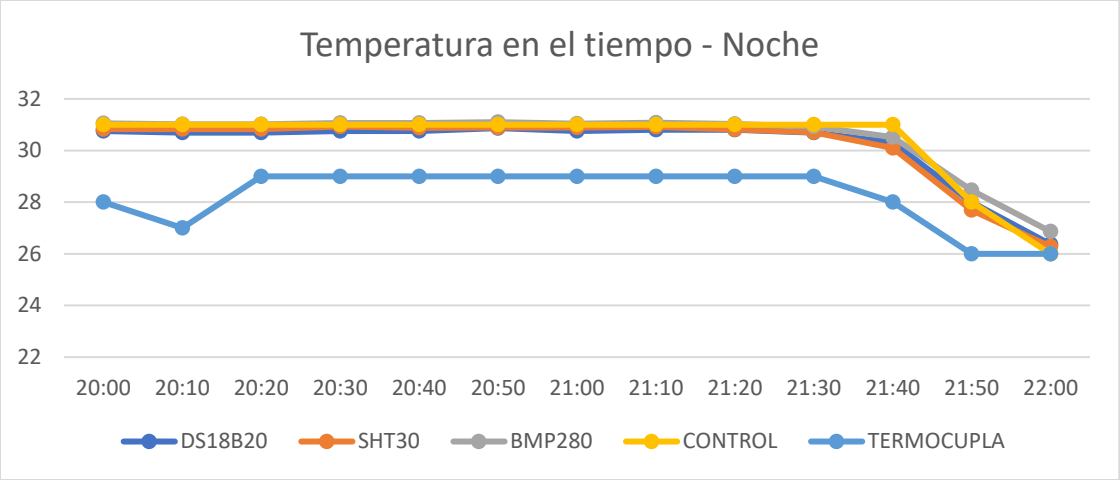


Ilustración 25 – Temperatura en un intervalo de 2 horas en una noche. [Fuente: Autor].

Código para el sensor de temperatura:

MAXIM DS18B20

Inclusión de Librería

```
#include <OneWire.h>
#include <DallasTemperature.h>
```

Se declaran las librerías OneWire y Dallas para comunicación con el sensor.

Inicialización del dispositivo	
OneWire oneWire(7);	Se declara una instancia de comunicación OneWire en el pin 7 del Arduino.
DallasTemperature DSB(&oneWire);	Se crea y vincula un objeto DSB en la librería Dallas con la instancia OneWire definida anteriormente
DSB.begin();	Inicializa el sensor
Lectura del dispositivo	
void DS18B20_read(){	Declaración de la función.
DSB.requestTemperatures();	Solicitar lectura de temperatura
DS_TEMP = DSB.getTempCByIndex(0);	Obtener valor medido de temperatura
}	

Tabla 26 – Código básico requerido para utilizar el sensor DS18B20. [Fuente: Autor].

4.1.2 Humedad

El mismo comportamiento observado con los sensores de temperatura se puede observar en el sensor de humedad (SHT30), donde la diferencia entre el control y el sensor disminuye en la noche, conforme el sensor en funcionamiento a tenido tiempo para estabilizar las mediciones y mejorar la precisión de este.

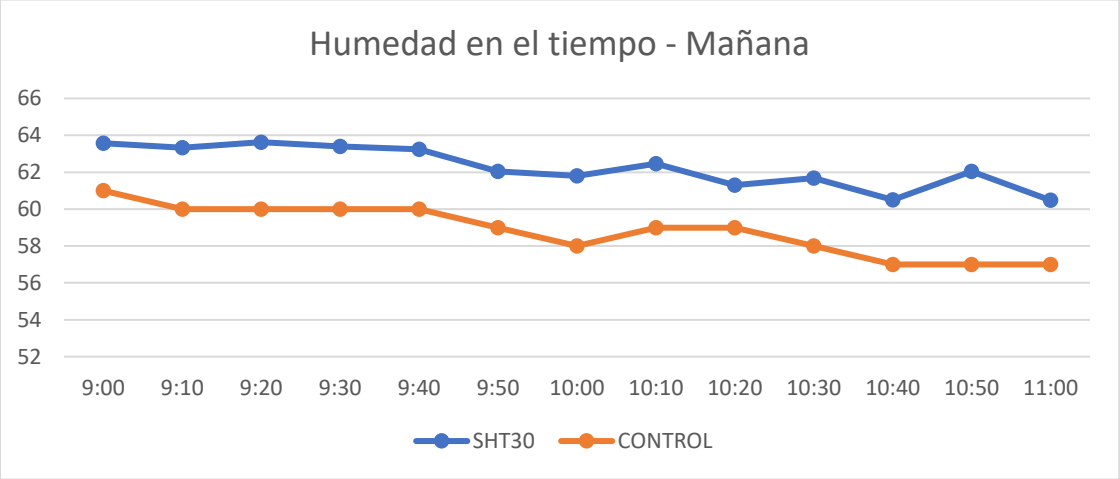


Ilustración 26 – Humedad en un intervalo de 2 horas en una mañana. [Fuente: Autor].

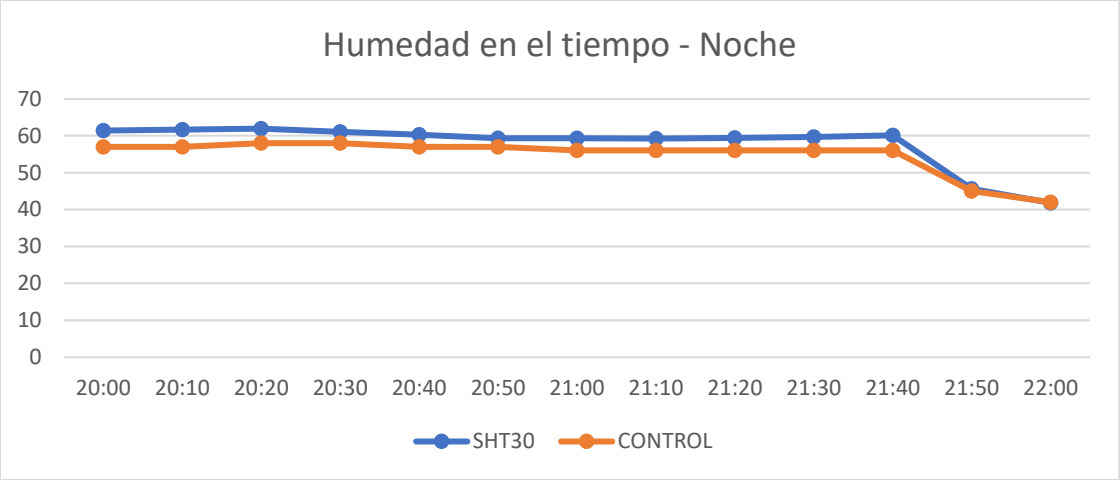


Ilustración 27 – Humedad en un intervalo de 2 horas en una noche. [Fuente: Autor].

Código para el sensor de humedad:

Sensirion SHT30

Inclusión de Librería

#include "SHTSensor.h"	Se declara la librería para incluir durante la compilación del código.

Inicialización del dispositivo

SHTSensor SHT;	Se crea y vincula un objeto SHT con la librería

if(SHT.init()){
  SHT_AVAILABLE = true;
}else{
  SHT_AVAILABLE = false;
}	

SHT.setAccuracy(SHTSensor::SHT_ACCURACY_MEDIUM);	Configura la exactitud del sensor (entre más exacto más consumo de corriente y más calor generado en el sensor)

Lectura del dispositivo

void SHT_read(){	Declaración de la función.

```
  if(SHT.readSample()){
    SHT_TEMP = SHT.getTemperature();
    SHT_HUM = SHT.getHumidity();
  }
}
```

Si el sensor está disponible, obtiene la lectura de la temperatura y humedad y las asigna a dos variables.

Tabla 27 – Código básico requerido para utilizar el sensor SHT30. [Fuente: Autor].

4.1.3 Presión

Dado la falta de un instrumento calibrado que pueda medir presión de forma precisa, se observa el comportamiento del sensor a lo largo del tiempo.

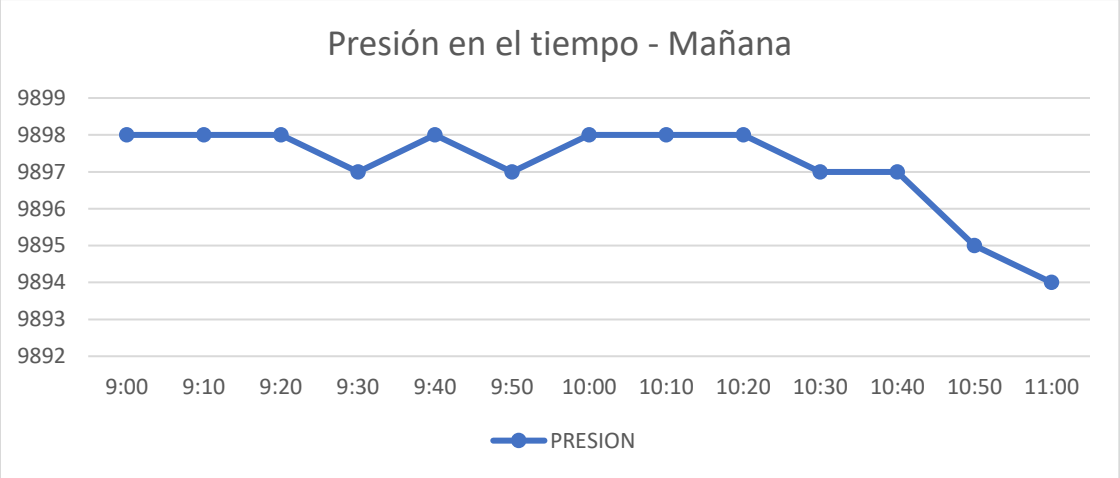


Ilustración 28 – Presión en un intervalo de 2 horas en una mañana. [Fuente: Autor].

Tanto en la mañana como en la noche se pueden evidenciar ligeros cambios en la presión reportada, nunca excediendo los 100kPa (aproximadamente 0.99 atm ó 1 bar), lo que sugiere que son cambios dados por el ambiente que no tienen afectación directa en la medición.

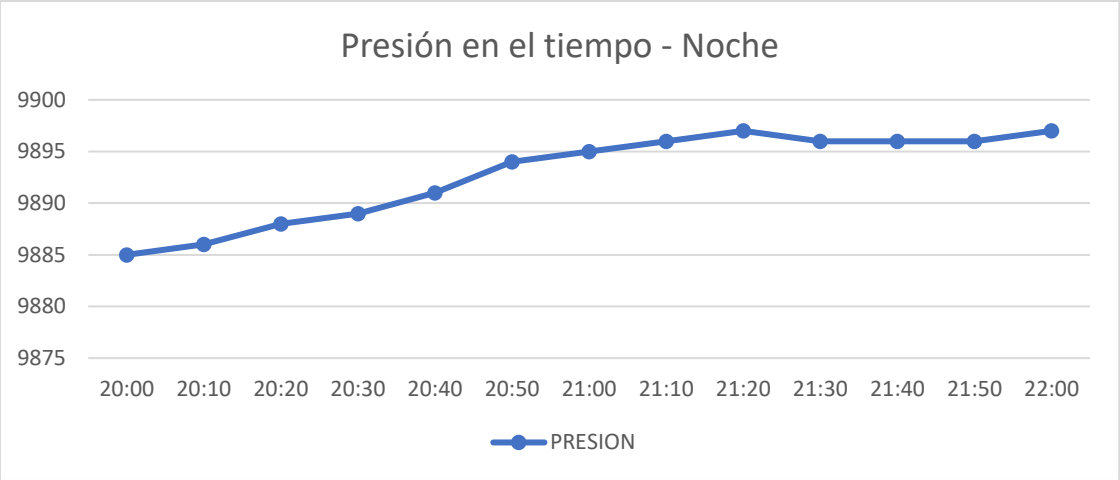


Ilustración 29 – Presión en un intervalo de 2 horas en una noche. [Fuente: Autor].

4.1.4 Altitud

Dado que la altitud reportada por el BMP280 se calcula de la presión, se puede observar un comportamiento similar en los datos.

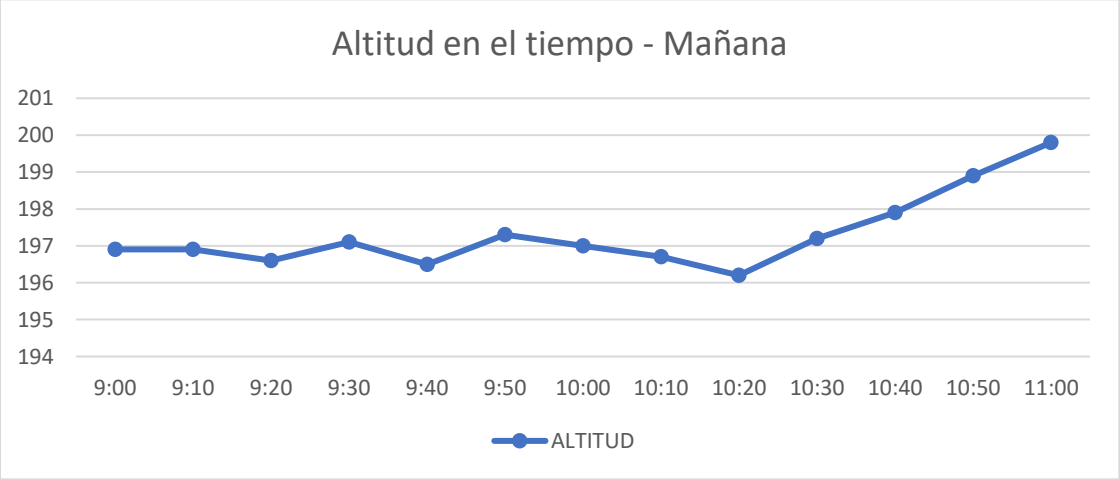


Ilustración 30 – Altitud reportada en un intervalo de 2 horas en una mañana. [Fuente: Autor].

La variación de los datos en el tiempo sigue siendo dentro de los parámetros normales, por lo que se considera no necesario emplear modificaciones en los datos entregados por el BMP280.

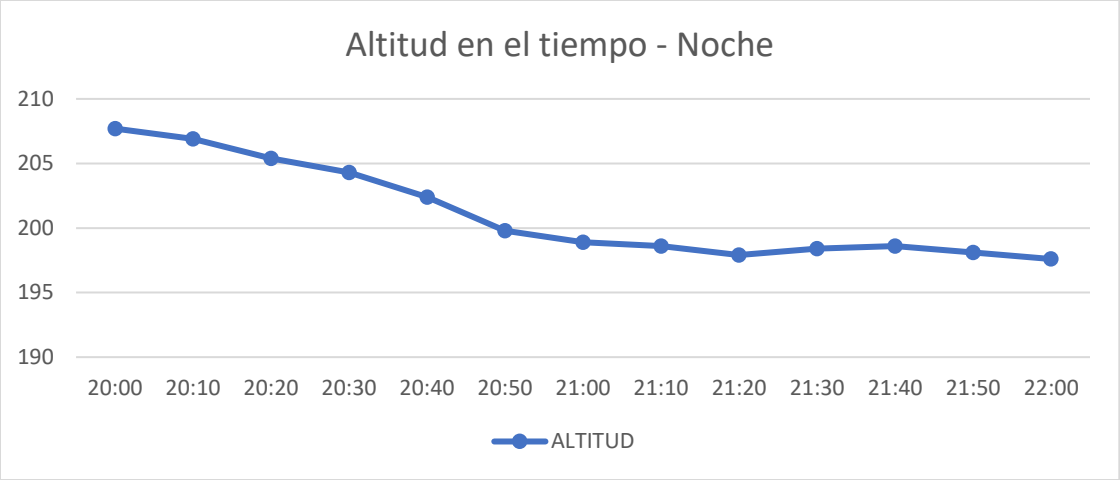


Ilustración 31 – Altitud reportada en un intervalo de 2 horas en una noche. [Fuente: Autor].

Código para el sensor de presión y altitud:

BOSCH BMP280

Inclusión de Librería

```
#include <Adafruit_BMP280.h>
```

Se declara la librería de Adafruit para incluir durante la compilación del código.

Inicialización del dispositivo

```
Adafruit_BMP280 BMP;
```

Se declara un objeto BMP para vincular con la librería de Adafruit

```
if (!BMP.begin(0x76)) {
  Serial.println(F("Could not find a valid BMP280
                   sensor, check wiring!"));
  while (1);
}
```

Si el dispositivo no ha sido inicializado, entra en un ciclo repetitivo hasta que el mismo sea inicializado.

```
BMP.setSampling(
            Adafruit_BMP280::MODE_NORMAL,
            Adafruit_BMP280::SAMPLING_X2,
```

Configura los parámetros de medición del sensor

- Modo de Operación
- Sobre muestra de temperatura
- Sobre muestra de humedad
- Filtrado
- Tiempo de espera

Adafruit_BMP280::SAMPLING_X16, Adafruit_BMP280::FILTER_X16, Adafruit_BMP280::STANDBY_MS_500);	
Lectura del dispositivo	

void BMP_read(){
BMP_TEMP = BMP.readTemperature();
BMP_PRESS = BMP.readPressure();
BMP_ALT = BMP.readAltitude(1013.25);
}
	Declaración de la función.
	
	Obtiene las lecturas de temperatura, presión y altitud (Compensada con presión a nivel del mar, debe modificarse si se usa en ambientes distintos)

Tabla 28 - Código básico requerido para utilizar el sensor BMP280. [Fuente: Autor].

4.1.5 Dirección del Viento

Dado que solo se reportan 8 posiciones respecto al sensor (representadas en ángulos de 45°), se observa la estabilidad de la lectura en el tiempo.

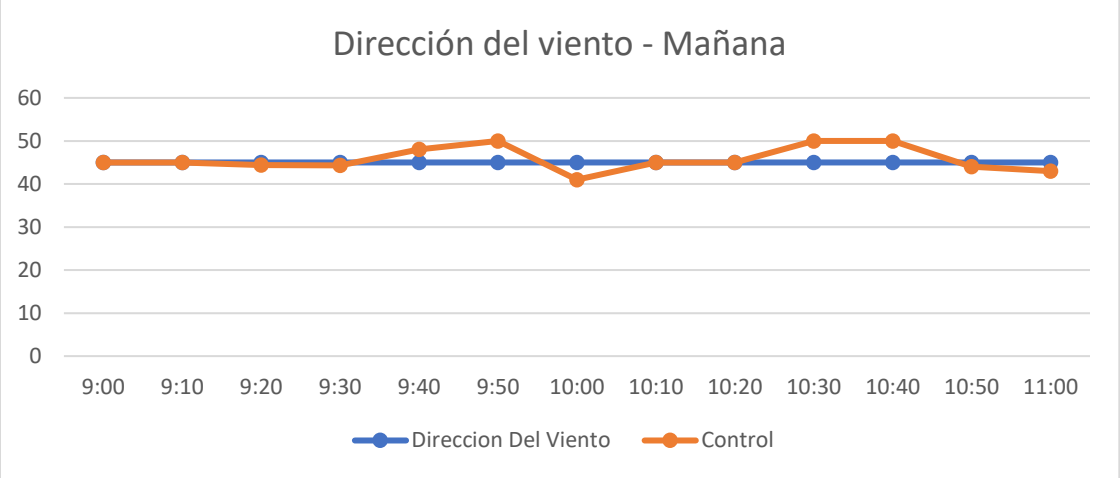


Ilustración 32 – Dirección del viento en un intervalo de 2 horas en una mañana. [Fuente: Autor].

Durante la mañana se observa una estabilidad respecto a la lectura, dado que la diferencia no es mucha, se procede en la medición de la noche a alterar la dirección (esto es, moviendo manualmente la veleta) para observar su comportamiento.

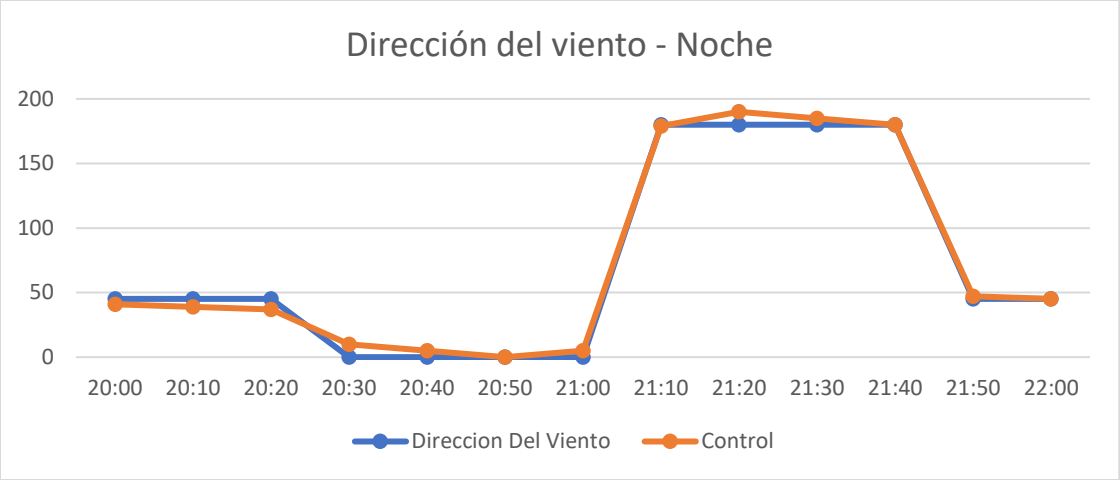


Ilustración 33 – Dirección del viento manipulada en un intervalo de 2 horas. [Fuente: Autor].

Adicionalmente se encontró que los valores dados por el sensor están rotados 180°, por lo que se hace la respectiva corrección en el código.

Código para el sensor de dirección del viento:

SEN-08942 – Veleta

Inclusión de Librería

No requiere librería puesto que es un sensor análogo.

Inicialización del dispositivo

No requiere inicialización puesto que es un sensor análogo.

Lectura del dispositivo

```
void DIR_Meas(){
```

Declaración de la función.

```
 V = analogRead(A0);
 A = V*(5.0/1023.0);
```

Realiza la lectura análoga, obteniendo de este modo el valor equivalente del ADC, y procede a clasificar en el ángulo respectivo.

```
 if(V > 67 && V < 118){
  W_Dir = "270";
 }else if(V > 159 && V < 210){
  W_Dir = "315";
 }else if(V > 262 && V < 312){
```

```
    W_Dir = "0";
  }else if(V > 435 && V < 486){
    W_Dir = "225";
  }else if(V > 604 && V < 655){
    W_Dir = "45";
  }else if(V > 760 && V < 810){
    W_Dir = "180";
  }else if(V > 861 && V < 911){
    W_Dir = "135";
  }else if(V > 919 && V < 969){
    W_Dir = "90";
  }
}
```

Tabla 29 – Código para el sensor de dirección de viento (Veleta). [Fuente: Autor].

4.1.6 Calidad del Aire (eCO2 y TVOC)

Para la estación meteorológica, el sensor CCS811 no tiene como objetivo dar una medida exacta del nivel de CO2 en el ambiente, el objetivo de este es dar al usuario una medida aproximada relativa de los niveles de CO2 (eCO2) en comparación con el ambiente (Por ejemplo, registrando una nominal de 800ppm en condiciones de baja actividad, comparado con el de 2400ppm durante una actividad dada)

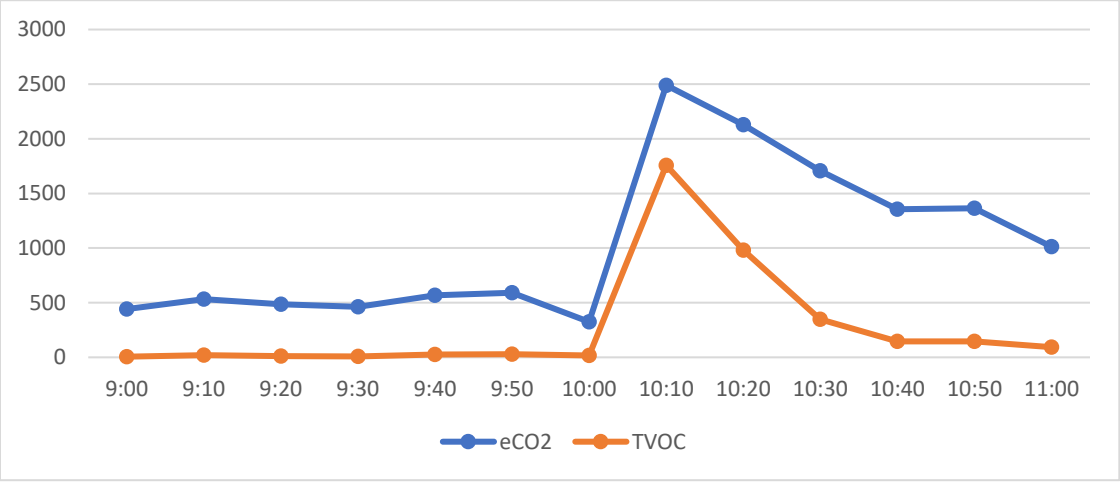


Ilustración 34 – Comportamiento de los niveles de eCO2 y TVOC en un intervalo de 2 horas en una mañana. [Fuente: Autor].

Durante las pruebas se observa que el sensor reacciona en base a distintas variables de las que no necesariamente es posible llevar el control, un ejemplo de esto, es que a puerta cerrada (con aire acondicionado en un ambiente “semi” controlado) se evidencia en los datos una especie de “repetición” respecto a los niveles de eCO2 en el ambiente, los cuales se ven afectados solamente cuando hay cambios en el mismo (por ejemplo: al realizar movimientos, respirar cerca al mismo, encender humo, etc.) Mientras que en un ambiente abierto el sensor se ve afectado por parámetros que no son necesariamente controlables (ejemplo: automóviles que pasen por el lugar).

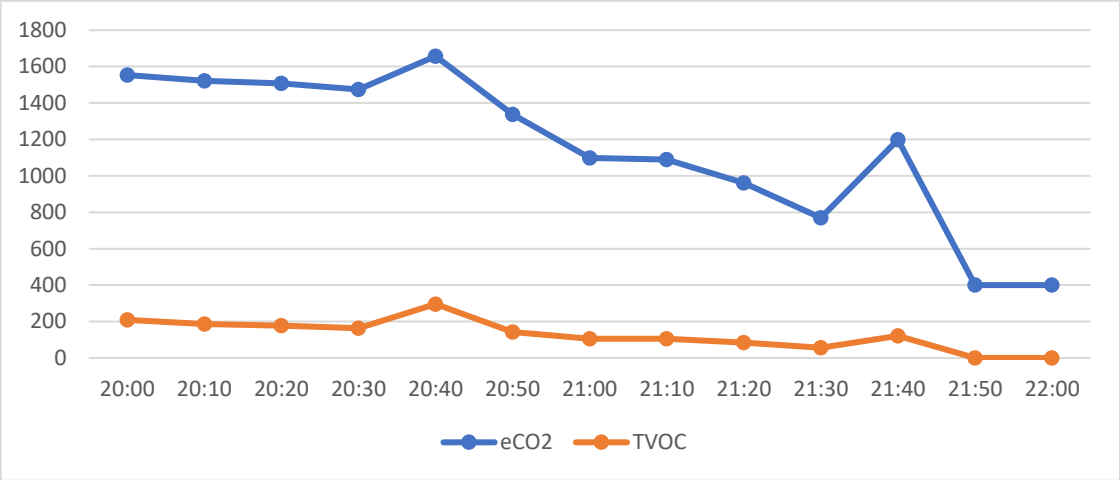


Ilustración 35 – Comportamiento de los niveles de eCO2 y TVOC en un intervalo de 2 horas en una noche. [Fuente: Autor].

Código para el sensor de dióxido de carbono:

Adafruit Industries CCS811

Inclusión de Librería

#include “Adafruit_CCS811.h”	Se declara la librería de Adafruit para incluir durante la compilación del código.

Inicialización del dispositivo

Adafruit_CCS811 CCS;	Se declara la variable CCS como un objeto a referir a la librería de Adafruit.

If(!CCS.begin()){
while(1);
}
	Si el dispositivo no a sido inicializado, entra en un ciclo repetitivo hasta que el mismo sea inicializado.
	
	Espera a que el dispositivo esté disponible para su uso.

while(!ccs.available());	
Lectura del dispositivo	
void ccs_read(){	Declaración de la función.
if(CCS.available()){	
_eCO2 = CCS.geteCO2();	Si el dispositivo está disponible, obtiene las lecturas de eCO2, TVOC y las asigna a dos variables.
_TVOC = CCS.getTVOC();	
}	
}	

Tabla 30 - Código básico requerido para utilizar el módulo CCS811. [Fuente: Autor].

Por recomendación del fabricante, en caso de no usar el sensor por un tiempo, se recomienda que se haga un periodo de "quemado" equivalente a 48 horas, esto porque durante este tiempo los datos entregados del sensor no son estables puesto que el mismo realiza una auto calibración en base al ambiente en el que se encuentra. Para casos en que el sensor se usa de forma seguida, se recomienda un periodo de "quemado" de 30 minutos para estabilizar los datos del sensor.

Adicionalmente se tienen los códigos para los sensores de luminosidad, intensidad ultravioleta, velocidad del viento y precipitación, los cuales mostraron el comportamiento esperado:

4.1.7 Intensidad lumínica

Código para el sensor de intensidad lumínica

ROHM Semiconductor BH1750FVI

Inclusión de Librería

#include <BH1750.h>	Se declara la librería para incluir durante la compilación del código.
Inicialización del dispositivo	
BH1750 BH;	Se crea y vincula un objeto BH con la librería.
BH.begin();	Inicializa el sensor
Lectura del dispositivo	
void Meas_Lux(){	Declaración de la función.
LX = BH.readLightLevel();	Obtiene la lectura de intensidad de luz
}	

Tabla 31 - Código básico requerido para utilizar el sensor BH1750FVI. [Fuente: Autor].

4.1.8 Intensidad Ultravioleta

Código para el sensor de intensidad ultravioleta:

UVM-30A

Inclusión de Librería

No requiere librería puesto que es un sensor análogo.

Inicialización del dispositivo

No requiere inicialización puesto que es un sensor análogo.

Lectura del dispositivo

```
void UV_read(){
  UV_Analog = analogRead(UV);
  if(UV_Analog < 10){
    UV_Index = 0;
  }else if(UV_Analog < 46){
    UV_Index = 1;
  }else if(UV_Analog < 65){
    UV_Index = 2;
  }else if(UV_Analog < 83){
    UV_Index = 3;
  }else if(UV_Analog < 103){
    UV_Index = 4;
  }else if(UV_Analog < 124){
    UV_Index = 5;
  }else if(UV_Analog < 142){
    UV_Index = 6;
  }else if(UV_Analog < 163){
    UV_Index = 7;
  }else if(UV_Analog < 180){
```

Declaración de la función.

Realiza la lectura análoga del sensor (por medio de un ADC incluido en el Arduino) y categoriza dicha lectura en los índices UV correspondientes (según datos del fabricante).

Índice UV	**Voltaje**
0	49mV
1	225mV
2	318mV
3	406mV
4	503mV
5	606mV
6	694mV
7	797mV
8	880mV
9	977mV
10	1.08V
11	1.17mV

UV_Index = 8;	11+	1.171mV+
}else if(UV_Analog < 200){		
UV_Index = 9;		
}else if(UV_Analog < 221){		
UV_Index = 10;		
}else if(UV_Analog < 239){		
UV_Index = 11;		
}else{		
UV_Index = 12;		
}		

Tabla 32 - Código básico requerido para utilizar el sensor UVM-30A. [Fuente: Autor].

4.1.9 Velocidad del Viento

Código para el sensor de velocidad de viento (Anemómetro):

SEN-08942 Anemómetro

Interrupción

void isr_rotation () {	Declaración de la función.
if ((millis() - AContactBounceTime) > 3) {	Aumenta un contador en 1 por cada pulso detectado, además registra el tiempo actual, de modo que si al activarse la función, el tiempo transcurrido es inferior a 3ms, ignora el pulso (Anti reboté).
Counter++;	
AContactBounceTime = millis();	
}	
}	

Inicialización del dispositivo

void ANEM_Init() {	Declaración de la función.
pinMode(Anem_Pin, INPUT);	Habilita la interrupción en el pin del Anemómetro y lo configura para activar la función de conteo durante cada pulso de “bajada”
attachInterrupt(digitalPinToInterrupt(Anem_Pin), isr_rotation, FALLING);	
}	

Lectura del dispositivo

void SPEED_Meas() {	Declaración de la función.
Rotations = 0; Counter = 0; attachInterrupt(digitalPinToInterrupt(Anem_Pin), isr_rotation, FALLING); delay (3000);	Añade y habilita la interrupción del Anemómetro, espera por 3 segundos, y elimina la interrupción. Durante esos 3 segundos cuenta la cantidad de pulsos detectados (interrupciones).
detachInterrupt(digitalPinToInterrupt(Anem_Pin)); Rotations = Counter / 2; WindSpeed = (Rotations / 3) * 0.32999;	Calcula la velocidad del viento en base al número de rotaciones por segundo.
}	

Tabla 33 - Código básico requerido para utilizar el Anemómetro. [Fuente: Autor].

4.1.10 Precipitación

Código para el sensor de precipitación (pluviómetro):

SEN-08942 Pluviómetro

Interrupción

void GAUGE_Meas(){	Declaración de la función.

if ((millis() - RContactBounceTime) > 15){
Rain_Global++;
RContactBounceTime = millis();
}
	Cuenta cada vez que la función sea llamada y actualiza un registro, de modo que si el tiempo entre la activación actual y la anterior es menor a 15ms ignora el conteo (Anti reboté).

}	

Inicialización del dispositivo

void GAUGE_Init() {	Declaración de la función.

pinMode(RGaug, INPUT_PULLUP);
attachInterrupt(digitalPinToInterrupt(RGaug), GAUGE_Meas, FALLING);
}
	Habilita la interrupción en el pin del pluviómetro y lo configura para activar la función de conteo durante cada pulso de “bajada”

Lectura del dispositivo	
void GAUGE_Read(){	Declaración de la función.
Serial.print("Number of Interrupts: "); Serial.print(Rain_Global); Serial.print(" - "); Serial.print(Rain_Global * 1.8); Serial.println("ml"); }	Muestra la cantidad de lluvia en ml que a ocurrido desde que empezó la medición. (Número de pulsos * equivalente en ml)

Tabla 34 - Código básico requerido para utilizar el pluviómetro. [Fuente: Autor].

4.1.11 Datos en el Tiempo

Se tomo una muestra de ciertos parámetros en un periodo de tiempo de 10 horas (no continuas), con el propósito de observar el comportamiento en el tiempo de los diferentes parámetros, con lo que se obtuvo:

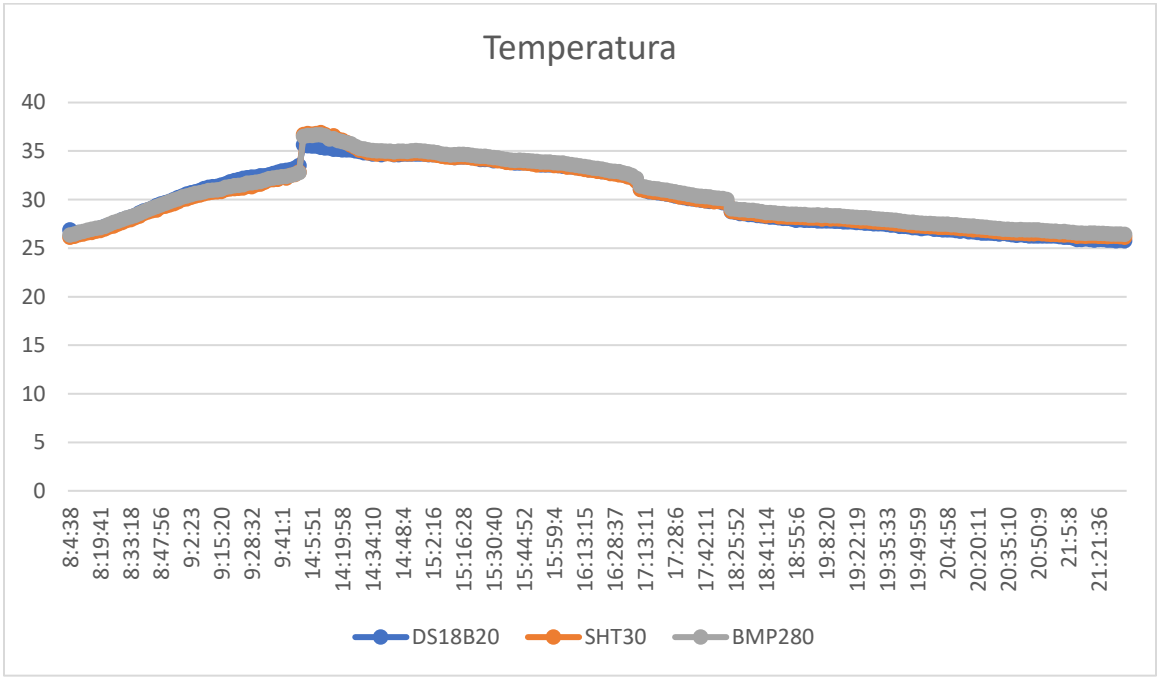


Ilustración 36 – Temperatura en un intervalo de 10 horas no continuas. [Fuente: Autor].

Al ver los datos se puede estimar una temperatura máxima de 37 °C (aproximados), y una mínima de 25 °C (aproximados) lo cual concuerda con los estimados que se pueden encontrar en línea (weather.com). Las horas de mayor temperatura siendo de 9:46 a.m. con una temperatura superior a los 32 °C, y terminando a las 4:50 p.m. con una temperatura inferior a los 32 °C (Horas en que la sensación térmica es alta, en las horas antes y después de este intervalo la misma disminuye).

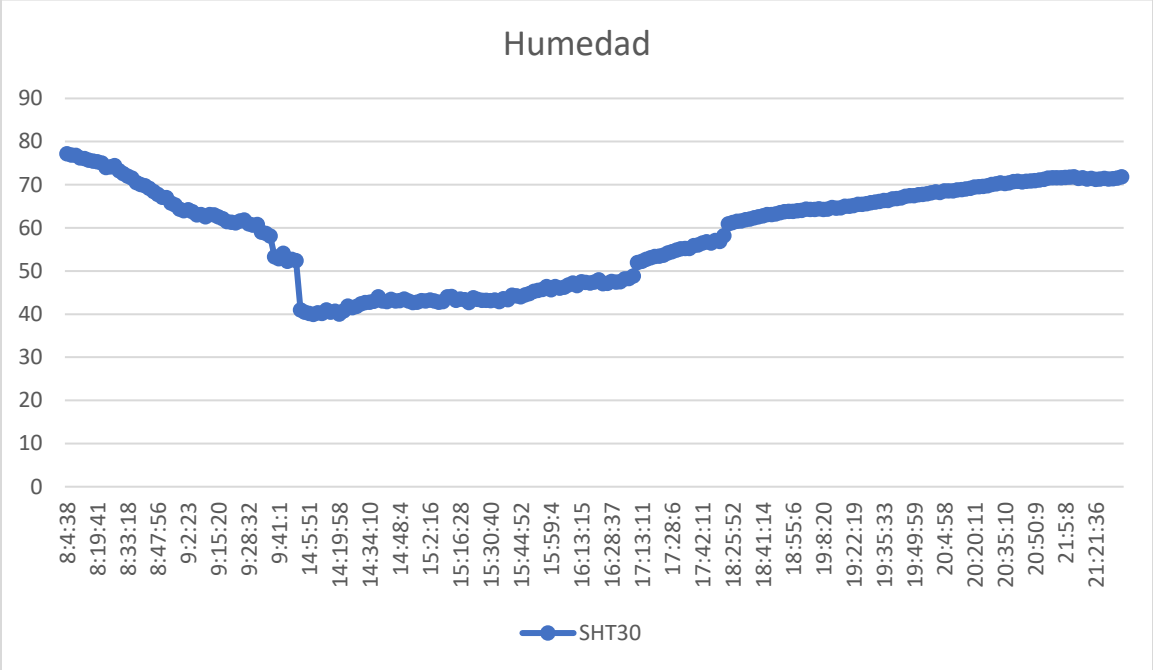


Ilustración 37 – Humedad en un intervalo de tiempo de 10 horas no continuas. [Fuente: Autor].

Así mismo, se observa un comportamiento de la humedad que es proporcional a la temperatura (pudiendo ser afectada por otros parámetros como el viento que en este caso no fueron considerados). Durante las horas en las que se evidencio mayor temperatura, se evidencia también el menor porcentaje de humedad, siendo este inferior al 50%RH, y alcanzando valores cercanos al 80%RH durante la madrugada.

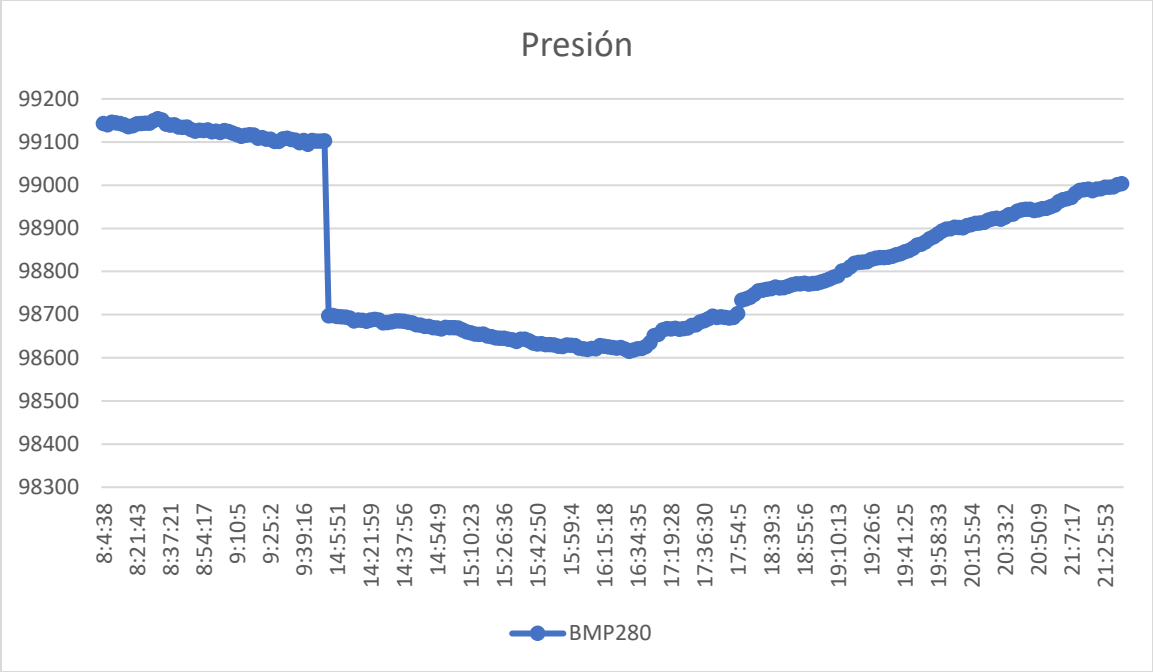


Ilustración 38 – Presión en un intervalo de tiempo de 10 horas no continuas. [Fuente: Autor].

El comportamiento de los datos obtenidos por el sensor de presión demuestra una estabilidad de esta, nótese que, si bien hay diferencia en los datos, los mismos no superan una diferencia de ± 250hPa (0.25%), lo cual denota, teniendo en cuenta las horas en que ocurren los cambios, que las variaciones en la presión son el resultado de la compensación por temperatura, además de factores externos que logren cambiar la lectura del sensor.

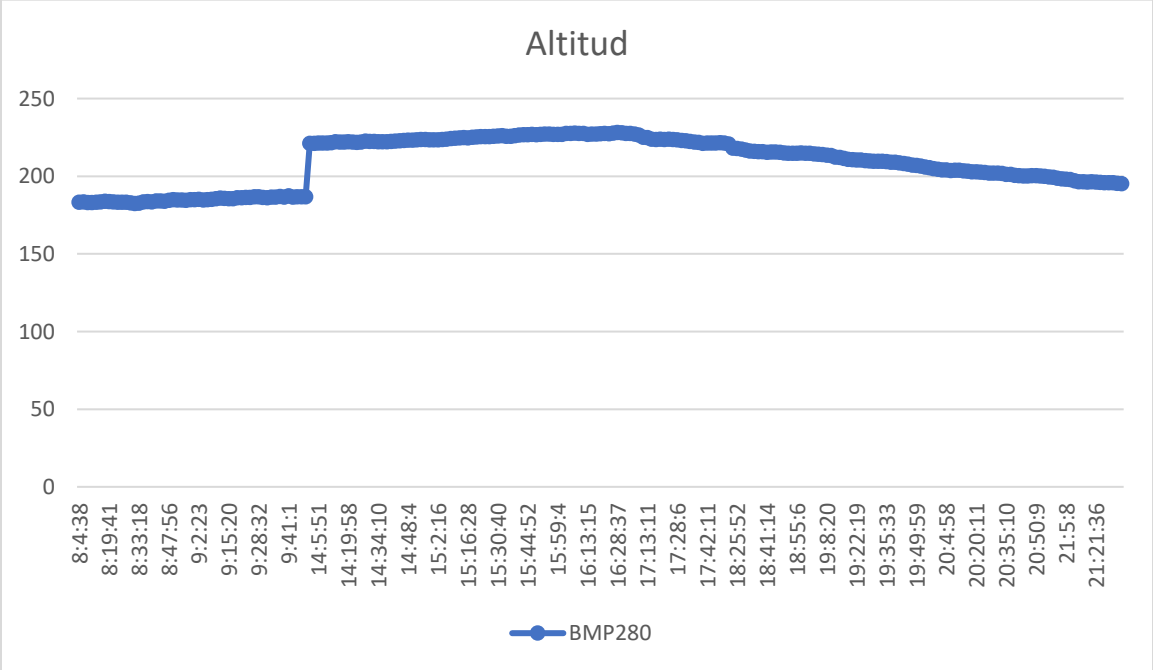


Ilustración 39 – Altitud en un intervalo de tiempo de 10 horas no continuas. [Fuente: Autor].

Del mismo modo, se evidencia una variación en la altitud reportada por el sensor, teniendo en cuenta que los datos de altitud son calculados internamente por el sensor de los datos de presión, se puede evidenciar entonces el mismo comportamiento inversamente proporcional, con una diferencia de ± 22.5m.

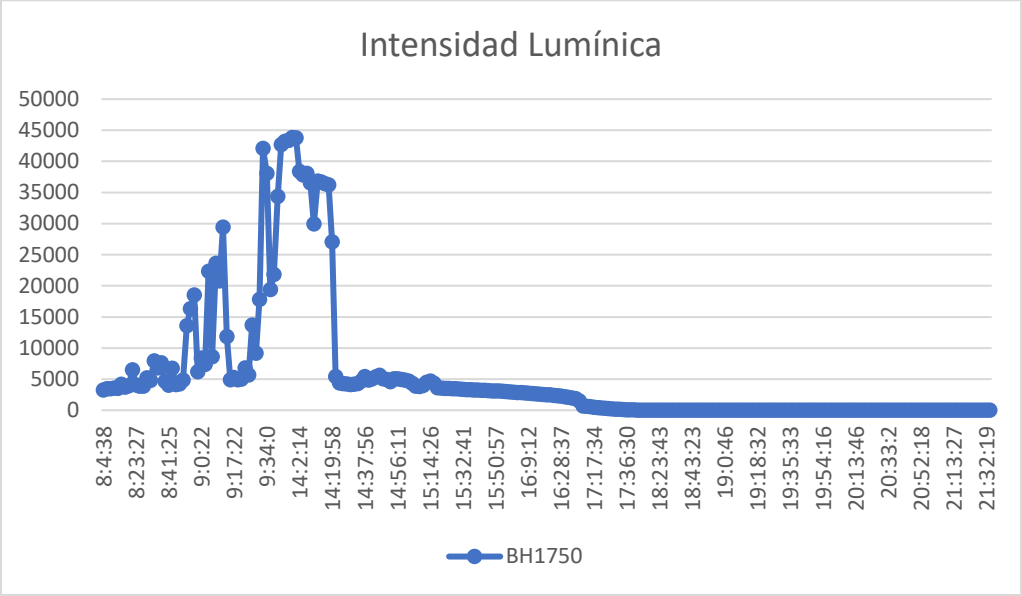


Ilustración 40 – Intensidad Lumínica en un intervalo de tiempo de 10 horas no continuas. [Fuente: Autor].

Para la intensidad lumínica se observa un comportamiento que denota picos en las mediciones, esto por factores como el ángulo en que llegue la luz, nubes, aves, entre otros. La mayor intensidad se encuentra en el intervalo de 10:00 a.m. y 3:00 p.m., sin embargo, dada la ubicación de la estación, esta intensidad disminuye por las sombras que disminuyen la intensidad registrada por el sensor, siendo este un error que se puede mejorar al conseguir una mejor ubicación de la estación (en un lugar más abierto de ser necesario).

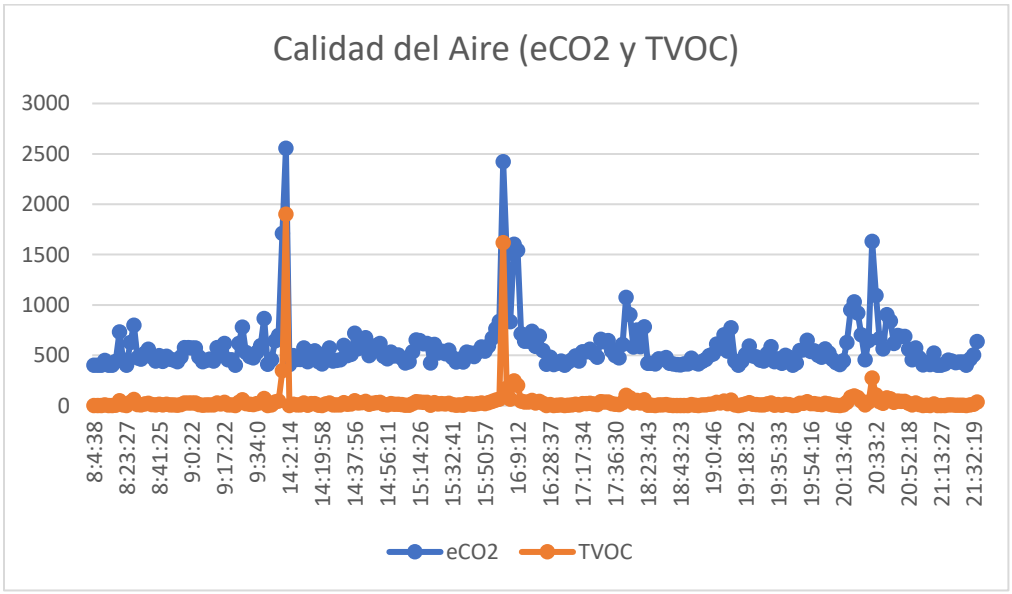


Ilustración 41 – Calidad del Aire en un intervalo de 10 horas no continuas. [Fuente: Autor].

Para la calidad del aire, se evidencian picos en distintos momentos del día, donde la actividad más registrada se encuentra a las 9:48 a.m., 4:01 p.m. y 8:30 p.m. siendo estos datos sujetos a la actividad que se realice en los alrededores, y por lo mismo, al lugar en el que se encuentre ubicada.

Los datos de velocidad y dirección del viento no fueron tenidos en cuenta para su análisis puesto que el lugar en el que se tomaron los datos no era el ideal, lo cual generaba variaciones constantes en la dirección y velocidad que no necesariamente eran descriptivos del lugar en el que se encuentran, adicionalmente tampoco se hace análisis a los datos del pluviómetro puesto que no se registraron precipitaciones de ningún tipo durante la medición de este.

4.1.12 Comparando con otra fuente de información.

Así mismo, se realizó un registro de datos en un intervalo de 4 horas, comparando las variables de temperatura y humedad con las de los datos dados por un higrómetro, y por los datos obtenidos en Google (Cuya fuente reporta ser Weather.com). No fue tenido en cuenta los datos de la estación meteorológica ubicada en el Aeropuerto Alfonso López Pumarejo, puesto que sus datos solo son dados en intervalos de 1 hora, y los mismos no son reportados 24 horas, sino en un intervalo de 6 horas, a diferencia de otras estaciones (ej. Barranquilla, Bogotá) donde los datos METAR se pueden obtener constantemente, siendo esta la única estación meteorológica de la ciudad de Valledupar.

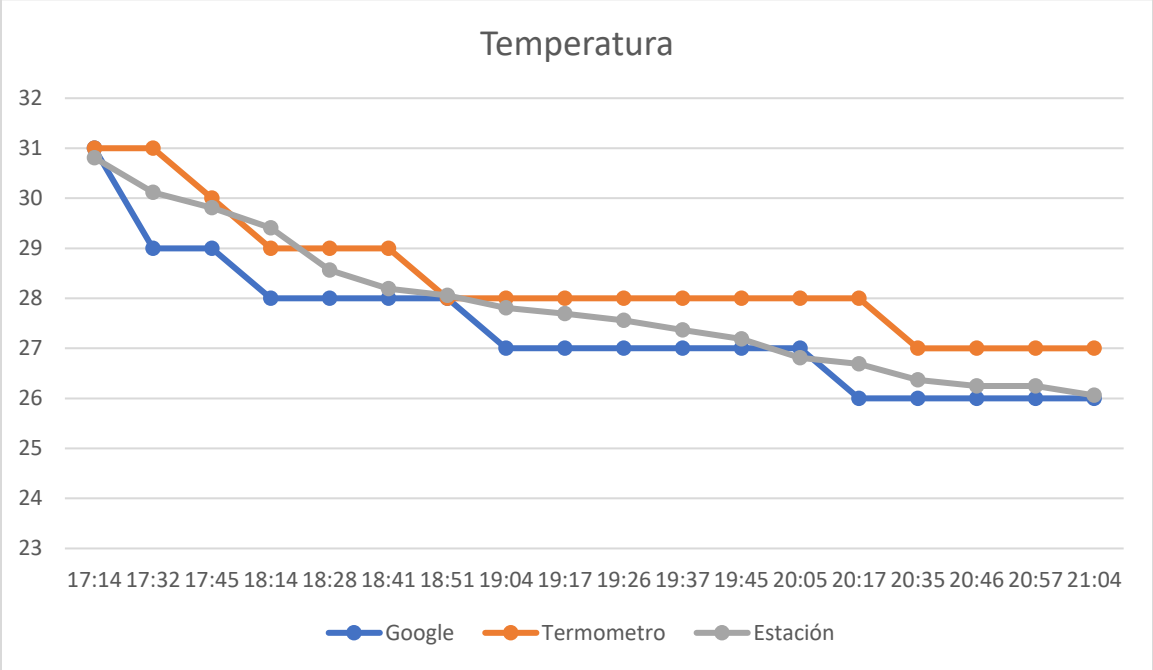


Ilustración 42 – Temperatura en un intervalo de 3 horas. [Fuente: Autor / Weather.com (vía Google)].

Se puede observar un comportamiento similar en los datos obtenidos de la estación meteorológica, siendo esta la que más muestra el comportamiento de la temperatura durante la disminución de esta en los horarios de 5:14 p.m. a 9:04 p.m. es importante tener en cuenta que la temperatura del higrómetro, y la reportada por Google, es dada sin números decimales, lo cual explica la diferencia vista en la gráfica.

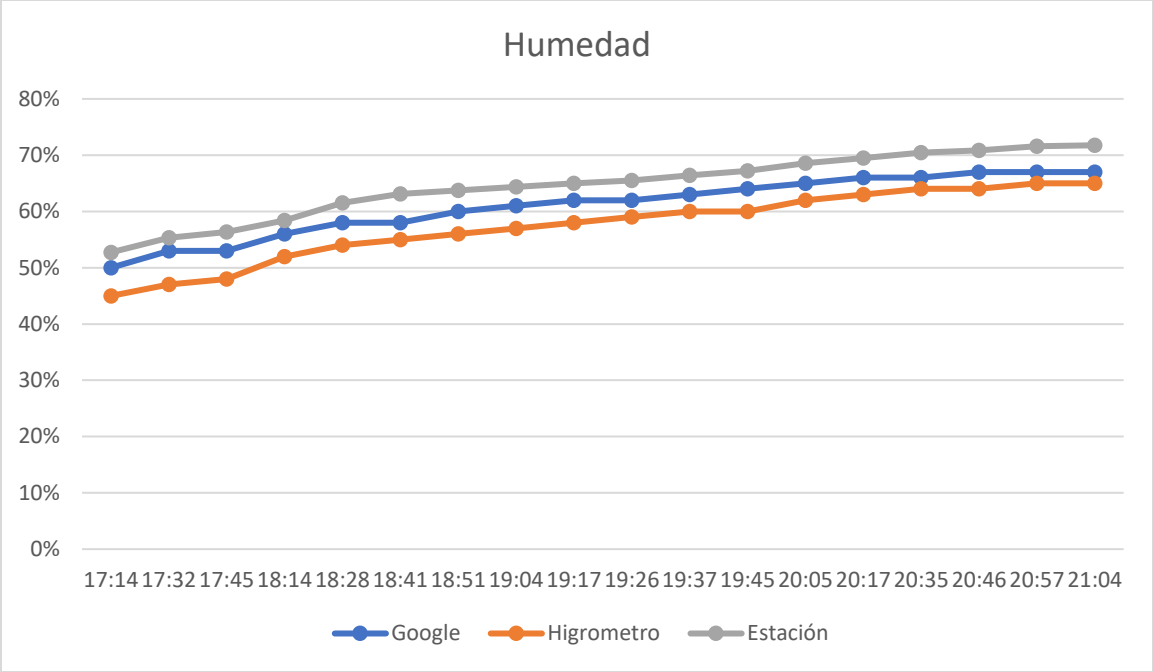


Ilustración 43 – Humedad en un intervalo de 3 horas. [Fuente: Autor /Weather.com (vía Google)].

Así mismo, se puede observar un comportamiento similar en los datos obtenidos de la humedad, siendo la diferencia de un ± 5%RH, la cual puede ser asociada al sitio de medición, y a factores como la sensibilidad del sensor, los vientos y la ubicación en el lugar.

Finalmente se obtiene el siguiente diagrama esquemático denotando las conexiones entre los módulos I2C y el Arduino.

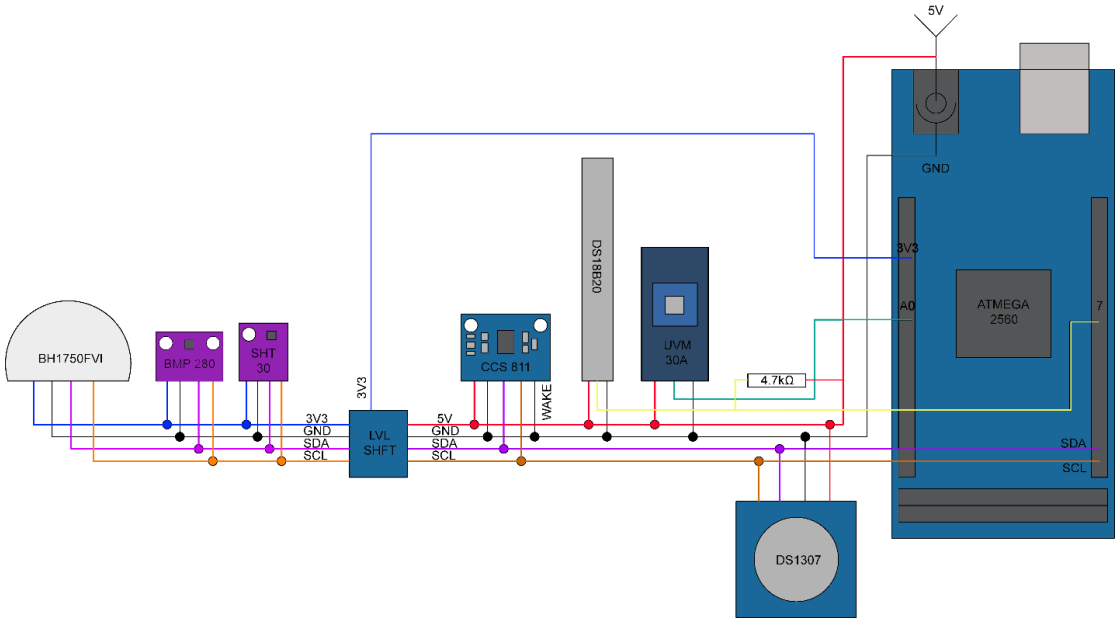


Ilustración 44 – Diagrama esquemático para conexión de dispositivos I2C con Arduino. [Fuente: Autor].

Junto con el diagrama esquemático para los sensores de velocidad, dirección del viento y precipitación.

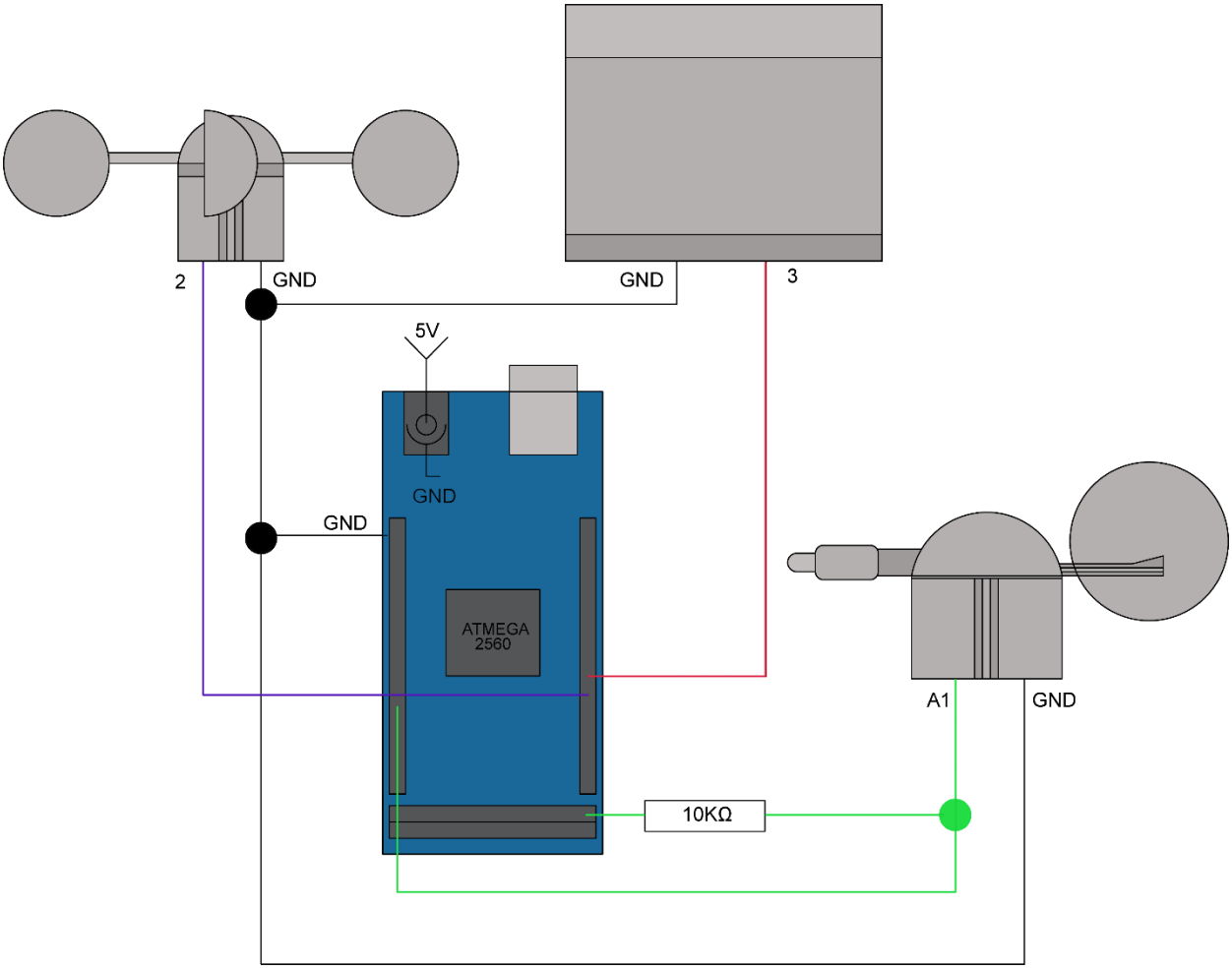


Ilustración 45 – Diagrama esquemático para la conexión entre los sensores de precipitación, velocidad y dirección del viento. [Fuente: Autor].

4.2 Comunicación inalámbrica

Como resultado del proceso y en base a los diagramas de flujo, se obtuvo el siguiente código para el funcionamiento de la comunicación y guardado de datos de la estación meteorológica:

4.2.1 Inicialización SIM808

SIMCOM SIM808

Inclusión de Librería

#include <SoftwareSerial.h>	Se declara la librería Software Serial para incluir durante la compilación del código.

Declaración del dispositivo

SoftwareSerial SIM808(10,8);	Se declara un objeto SIM808 con la librería Software Serial, indicando que se utilizan los pines 8, 10 como Tx y Rx respectivamente.
Inicialización del dispositivo	
void SIM_init(){	Declaración de la función.
String Result = "NULL";	
int Retry = 0;	Si el dispositivo está disponible, obtiene las lecturas de eCO2, TVOC y las asigna a dos variables.
while(SIM_PWR == false){	
SIM_send("AT\r\n");	
delay(20);	
Result = SIM_read();	
if(Result.startsWith("OK")){	
SIM_PWR = true;	
}else{	
Retry += 1;	
}	
if(Retry == 4){	
Serial.println(F("BREAK SIM INIT"));	
break;	
}	
}	
//Serial.println(F("BREAK SIM INIT LOOP"));	
}	

Tabla 35 – Código Inicialización del SIM808. [Fuente: Autor].

4.2.2 Envió y Recepción de Datos:

Código

Recepción (Lectura) de datos

String SIM_read(){	Declaración de la función
String R_DATA;	Se declara una cadena para los datos a leer

```
 while(SIM808.available() > 0){
  R_DATA = SIM808.readString();
 }
 R_DATA.trim();
 if(R_DATA != ""){
  Serial.print("SIM READ: ");
  Serial.println(R_DATA);
 }
 return(R_DATA);
}
```

Si hay datos disponibles, leerlos y asignarlos a la cadena anteriormente declarada.

Se eliminan los espacios innecesarios o caracteres de terminación.

Envió de datos

```
void SIM_send(String S_DATA){
   SIM808.write(S_DATA.c_str());
}
```

Declaración de la función

Envió de datos al módulo SIM

Categorización

```
void SIM_parser(){
 String Parse = "NULL";
 Parse = SIM_read();
 delay(500);
 if(Parse != ""){
  Serial.print("PARSE: ");
  Serial.println(Parse);
 }
 if(Parse.startsWith("+BTPAIRING")){
  BT_pair();
 }else if(Parse.startsWith("+BTCONNECTING")){
  BT_connect();
 }else if(Parse.startsWith("+BTSPPDATA")){
```

Categoriza automáticamente en base a:

- Emparejamiento Bluetooth
- Conexión Bluetooth
- Datos entrantes Bluetooth
- Llamada Entrante
- Nuevo Mensaje de Texto.

De encontrar la categoría, procede a llamar la función respectiva para ejecutar la acción necesaria.

```
    BT_read(Parse);
  }else if(Parse.startsWith("+BTDISCONN")){
    BT_disconn();
  }else if(Parse.startsWith("RING")){
    CALL_notify(Parse);
  }else if(Parse.startsWith("+CMTI")){
    SMS_notify(Parse);
  }
}
```

Tabla 36 – Código para la comunicación por Serial con el SIM808. [Fuente: Autor].

4.2.3 Inicialización Bluetooth

Código

Inicialización

`void BT_init(){`	Declaración de la función.
`  String Result = "NULL";`	Declaración de una cadena con valor “NULL”
`  int Retry = 0;`	
`  while(BT_PWR == false){`	Se crea un bucle en el cual se pregunta el estado del bluetooth, si el mismo está activo entonces sale del bucle, de lo contrario envía un comando para activación del bluetooth y continua en el bucle.
`    SIM_send("AT+BTSTATUS?\r\n");`	
`    delay(20);`	
`    Result = SIM_read();`	
`    Serial.print("RESULT STATUS: ");`	
`    Serial.println(Result);`	
`    if(Result.startsWith("+BTSTATUS: 0")){`	
`      SIM_send("AT+BTPOWER=1\r\n");`	
`      Retry += 1;`	
`    }else if(Result.startsWith("+BTSTATUS")){`	
`      BT_PWR = true;`	
`    }else{`	

```
    Retry += 1;
   }
   if(Retry == 4){
    Serial.println(F("BREAK BT INIT"));
    break;
   }
  }
}
```

Tabla 37 – Código para inicialización del bluetooth en el SIM808. [Fuente: Autor].

4.2.4 Emparejamiento Bluetooth

Código

Respuesta

```
void BT_pair(){
 SIM_send("AT+BTPAIR=1,1\r\n");
 delay(50);
}
```

Declaración de la función.

Si la cadena entrante indica una solicitud de emparejamiento, enviar comando de aceptación.

Esta función es accedida por el categorizador.

Tabla 38 – Código para aceptar solicitudes de emparejamiento por bluetooth. [Fuente: Autor].

4.2.5 Conexión Bluetooth

Código

Lectura y Respuesta

```
void BT_connect(){
 SIM_send("AT+BTACPT=1\r\n");
 delay(50);
}
```

Declaración de la función.

Si la cadena entrante indica una solicitud de conexión, enviar comando de aceptación. (El

módulo SIM rechaza automáticamente si ya se encuentra conectado a otro dispositivo)

Esta función es accedida por el categorizador.

Tabla 39 – Código para aceptar conexiones entrantes por bluetooth. [Fuente: Autor].

4.2.6 Desconexión Bluetooth

Código

Lectura y Respuesta

```
void BT_disconn(){
  String Result = "NULL";
  int Retry = 0;
  while(1){
    SIM_send("AT+BTDISCONN=1\r\n");
    delay(20);
    Result = SIM_read();
    if(Result.startsWith("OK")){
      break;
    }else{
      Retry += 1;
      if(Retry == 4){
        break;
      }
    }
  }
}
```

Declaración de la función.

Si la cadena entrante indica una solicitud de auto desconexión, se envía el comando de desconexión y se espera a una respuesta exitosa.

Esta función es accedida por el categorizador.

Tabla 40 – Código para desconectarse del dispositivo bluetooth. [Fuente: Autor].

4.2.7 Envió y Recepción de datos por Bluetooth

Código

Envió (Escritura) de datos

```
void BT_send(){
  String Result = "NULL";
  int Retry = 0;
  bool BT_M = false;
  bool MSG_M = false;
  while(BT_M == false){
   SIM_send("AT+BTSPPSEND\r\n");
   delay(20);
   Result = SIM_read();
   if(Result.startsWith(">")){
    MSG_M = true;
   }else if(Result != ""){
    Retry += 1;
    if(Retry == 4){
     break;
    }
   }else{
    MSG_M = true;
   }
   while(MSG_M == true){
    MSG_CONSTRUCTOR();
    delay(50);
    SIM_send("");
    delay(20);
    Result = SIM_read();
```

Declaración de la función.

Declaración de variables para llevar el control del estado en que se encuentra el módulo.

Se indica al módulo que se van a enviar datos por bluetooth, se espera a la respuesta de este (que sea exitosa) y una vez hecho esto se envían los datos por medio de la función de construcción de mensaje, seguido de esto se envía el comando de terminación (Ctrl Z). Si no se hace esto y el módulo se encuentra en modo de envió entonces todos los comandos que el módulo reciba (con excepción del de terminación) serán interpretados como parte del mensaje a enviar. Finalmente se espera a recibir el resultado exitoso por parte del módulo.

```
    if(Result.startsWith("SEND OK")){
      MSG_M = false;
      BT_M = true;
    }else{
      MSG_M = false;
      Retry += 1;
    }
   }
  }
}
```

Recepción (Lectura) de datos

```
void BT_read(String BT_DATA){
  int MSG_Length = BT_DATA.length();
  char TMP_MSG[MSG_Length];
  BT_DATA.toCharArray(TMP_MSG, MSG_Length);
  strtok(TMP_MSG, ",");
  strtok(NULL, ",");
  String BT_rMSG = strtok(NULL, ",");
  if(BT_rMSG.startsWith("+SEND:")){
    BT_rMSG.toCharArray(TMP_MSG,
BT_rMSG.length());
    strtok(TMP_MSG, "\"");
    String Number = strtok(NULL, "\"");
    auto_Answer(Number);
  }
}
```

Declaración de la función

(Se presenta un ejemplo para auto respuesta, puede cambiar)

Si la cadena de datos indica que se han recibido datos por bluetooth, crear un vector de caracteres para procesar la información, luego crear una cadena con el dato de interés (en este caso el numero celular) y proceder a dar auto respuesta.

Esta función es accedida por el categorizador.

Tabla 41 – Código para el envío y recepción de datos por bluetooth (Serial o RFCOMM). [Fuente: Autor].

4.2.8 Auto Respuesta

Código

Auto Respuesta

```
void auto_Answer(String Number){
  delay(100);
  bool rout_wait = true;
  bool sms_wait = true;
  String Result = "";
  while(rout_wait == true){
   SIM_send("AT+CMGF=1\r\n");
   delay(20);
   Result = SIM_read();
   if(Result.startsWith("OK")){
    bool rou2_wait = true;
    while(rou2_wait == true){
     String S_Number =
                        "AT+CMGS=\""+Number+"\"";
     SIM_send("AT+CMGS=\"");
     SIM_send(Number);
     SIM_send("\"\r\n");
     delay(20);
     Result = SIM_read();
     if(Result == ">"){
      MSG_CONSTRUCTOR();
      SIM_send("");
      while(sms_wait == true){
       Result = SIM_read();
       delay(20);
```

Declaración de la función.

Declaración de variables para llevar el control del estado en que se encuentra el módulo y declaración de una cadena de respuesta.

Se entra en un bucle inicial que encarga de enviar el comando para establecer el módulo SIM en modo de mensaje de texto, se espera a la respuesta y de ser exitosa se entra en otro bucle para realizar él envió de los datos al número celular especificado (“XXXXXXXXXX”), se espera a la respuesta del módulo, de ser exitosa “>” se procede a enviar el mensaje usando la función de constructor de mensaje, seguidamente se envía el comando de terminación (Ctrl Z) y se espera a la respuesta.

```
        if(Result.startsWith("+CMGS")){
          sms_wait = false;
          rou2_wait = false;
          rout_wait = false;
        }else if(Result == "ERROR"){
          sms_wait = false;
          rou2_wait = false;
          rout_wait = false;
        }
        delay(20);
      }
    }else{
      sms_wait = false;
      rou2_wait = false;
      rout_wait = false;
    }
   }
  }
 }
}
```

Tabla 42 – Código Rutina de auto respuesta. [Fuente: Autor].

4.2.9 Notificación de llamadas y SMS

Código

Notificación de Llamada

void CALL_notify(String Result){	Declaración de la función.
if(Result.startsWith("RING")){	
SIM_send("ATH\r\n");	Si la cadena indica una notificación de llamada, colgar la llamada, filtrar el número telefónico y generar una auto respuesta (Mensaje de Texto).
delay(50);	

```
   String TEMP_MSG = Result.substring(8,
                                      Result.length());
   int MSG_Length = TEMP_MSG.length();
   char CLIP[MSG_Length];
   TEMP_MSG.toCharArray(CLIP, MSG_Length –
                                                2);
   strtok(CLIP, "\"");
   String Number = strtok(NULL, "\"");
   auto_Answer(Number);
  }
}
```

Notificación de Mensaje de Texto (SMS)

```
void SMS_notify(String Result){
```

Declaración de la función

```
  delay(100);
  if(Result.startsWith("+CMTI:")){
```

Si la cadena indica una notificación de mensaje de texto (SMS), enviar un comando al módulo para recibir el mensaje de texto, filtrar el número, eliminar el mensaje de texto y generar una auto respuesta (Mensaje de Texto).

```
   delay(50);
   SIM_send("AT+CMGL\r\n");
   delay(20);
   String SMS_Store = SIM_read();
   if(SMS_Store.startsWith("+CMGL:")){
    String Number;
    int SMS_StoreLength = SMS_Store.length();
    char SMS_CStore[SMS_StoreLength];
    SMS_Store.toCharArray(SMS_CStore,
                                  SMS_StoreLength - 2);
    strtok(SMS_CStore, ",");
    strtok(NULL, ",");
    Number = strtok(NULL, ",");
```

```
    Number.remove(0,1);
    Number.remove(Number.length()-1,1);
    SIM_send("AT+CMGD=1,0\r\n");
    auto_Answer(Number);
   }
   delay(50);
  }
 }
```

Tabla 43 – Código para manejo de llamadas y mensajes de texto (SMS) entrantes. [Fuente: Autor].

4.2.10 GPS

Código

Inicialización del GPS

```
void GPS_init(){
 String Result = "NULL";
 while(_GPSStat == 0){
  SIM_send("AT+CGNSPWR?\r\n");
  delay(20);
  Result = SIM_read();
  Result.trim();
  if(Result.startsWith("+CGNSPWR: 0", 0)){
   SIM_send("AT+CGNSPWR=1\r\n");
   delay(20);
  }else if(Result.startsWith("+CGNSPWR: 1", 0)){
   _GPSStat = 1;
  }
 }
}
```

Declaración de la función

Declaración de una cadena para respuesta

Dentro de un bucle, envía el comando preguntando por el estado del GPS, espera la respuesta y de estar encendido sale del bucle, de lo contrario envía el comando para encender el GPS y repite el bucle.

Lectura de datos del GPS

```
void GPS_getData(){
  int GPS_done = 0;
  int S_Length = 0;
  String Result = "NULL";
  SIM_send("AT+CGNSINF\r\n");
  delay(20);
  Result = SIM_read();
  Result.trim();
  while(GPS_done == 0){
    if(Result.startsWith("+CGNSINF:")){
      Result = Result.substring(10);
      S_Length = Result.length();
      char GPS_DATA[S_Length];
      Result.toCharArray(GPS_DATA, S_Length - 2);
      GPS_RunStatus = atoi(strtok(GPS_DATA, ","));
      GPS_FixStatus = atoi(strtok(NULL, ","));
      char ddate[18];
      char *date = ddate;
      date = strtok(NULL, ",");
      Latitude = atof(strtok(NULL, ","));
      Longitude = atof(strtok(NULL, ","));
      MSL_Altitude = atof(strtok(NULL, ","));
      strtok(NULL, ",");
      Course_Ground = atof(strtok(NULL, ","));
      strtok(NULL, ",");
      strtok(NULL, ",");
      strtok(NULL, ",");
```

Declaración de la función.

(Código sujeto a cambios)

Se envía el comando para solicitar los datos del GPS, se espera la respuesta y se entra en un bucle, dentro del bucle se procede a procesar los datos del GPS de manera que se obtienen los datos definidos anteriormente que son:

- Estado del GPS
- Estado de Arreglo del GPS
- Latitud
- Longitud
- Altitud respecto al nivel del mar
- Dirección (Orientación)
- Numero de satélites GPS en vista
- Numero de satélites GNSS en vista
- Año UTC
- Mes UTC
- Día UTC
- Hora UTC
- Minuto UTC
- Segundo UTC

Una vez hecho esto se da por finalizado el bucle y se continua con el código de la estación.

```
    strtok(NULL, ",");
    GPS_InView = atoi(strtok(NULL, ","));
    GNSS_InView = atoi(strtok(NULL, ","));
    String TEST(date);
    String TEMP;
    TEMP = TEST.substring(0,4);
    _Year = TEMP.toInt();
    TEMP = TEST.substring(4,6);
    _Month = TEMP.toInt();
    TEMP = TEST.substring(6,8);
    _Day = TEMP.toInt();
    TEMP = TEST.substring(8,10);
    _Hour = TEMP.toInt();
    TEMP = TEST.substring(10,12);
    _Minute = TEMP.toInt();
    TEMP = TEST.substring(12,14);
    _Second = TEMP.toInt();
   GPS_done = 1;
  }else{

  }
 }
}
```

Tabla 44 – Código para la comunicación con el GPS. [Fuente: Autor].

4.2.11 Auto respuesta

El funcionamiento de la auto respuesta, el manejo de llamadas y mensajes de texto (SMS) entrantes, termina en él envió de un mensaje de texto con los datos que posee la estación en el momento.

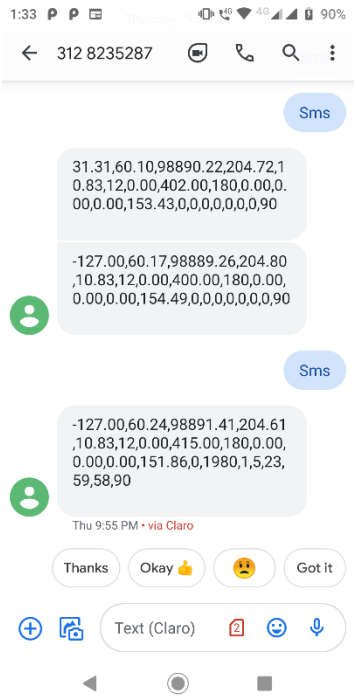


Imagen 20 – Recepción de mensaje de texto (SMS) después de llamada y después de envió de SMS. [Fuente: Autor].

Adicionalmente se obtiene el siguiente diagrama esquemático para la conexión con el módulo SIM.

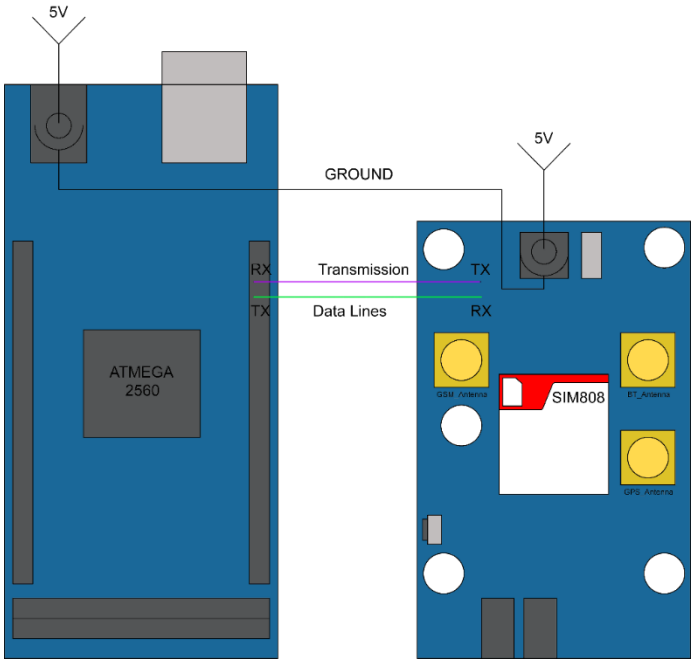


Ilustración 46 – Diagrama esquemático de conexión entre el módulo SIM808 y el Arduino. [Fuente: Autor].

Como otro resultado se obtiene el código para el manejo de la MicroSD a través de un módulo adaptador para el arduino.

4.2.12 MicroSD

Código

Inclusión de Librería

#include <SPI.h> #include <SD.h>	Se declaran las librerías SD (para interactuar con la MicroSD) y SPI (para utilizar el protocolo de comunicación).

Inicialización de la MicroSD

void init_SD() {	Declaración de la función.
if (!SD.begin(CS)) {	Inicializa la MicroSD, si no es posible (sea que no haya MicroSD o que se genere un error en la comunicación) entonces entrar en un bucle infinito hasta que haya comunicación (es obligatorio el uso de MicroSD).
while (1);	
}	
}	

Guardado de datos en la MicroSD

void save_MicroSD(){	Declaración de la función.
String dataString = sd_CONSTRUCTOR();	Llamada al constructor de cadena y asignación del resultado a una variable.
File dataFile = SD.open("dataLog.txt", FILE_WRITE);	Abrir archivo “dataLog.txt” en modo escritura.
if(dataFile){	Si el archivo está abierto, guardar datos y cerrar el archivo, de lo contrario generar un error.
dataFile.println(dataString);	
dataFile.close();	
}else{	
Serial.println("ERROR TRYING TO SAVE INTO FILE!");	
}	
}	

Tabla 45 – Código para comunicación y guardado de datos en la MicroSD. [Fuente: Autor].

Adicionalmente se obtiene el siguiente diagrama esquemático para la conexión con el módulo de MicroSD

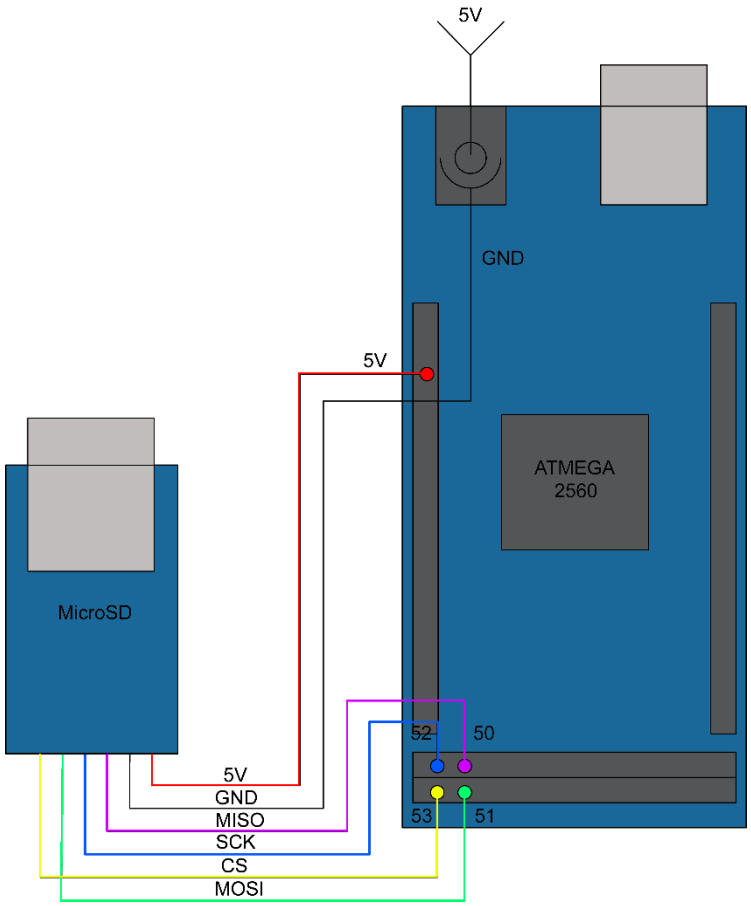


Ilustración 47 – Diagrama esquemático de conexiones para el módulo MicroSD. [Fuente: Autor].

4.3 Software

Se da como resultado dos aplicaciones para el monitoreo de datos de la estación meteorológica, una para sistemas Windows y la otra para sistemas Android, además de la posibilidad de monitorear los datos captados por medio de un monitor serial (Con conexión USB).

4.3.1 Aplicación WeatherView (Windows)

La aplicación en Windows funciona como un monitor para la estación meteorológica que además permite visualizar los datos de forma gráfica o de lista (histórico) además de permitir exportar los mismos a un archivo separado por coma (csv) para su procesamiento posterior por parte del usuario.

Pantalla inicial

Al abrir la aplicación, solo se visualiza un mensaje “¡Para comenzar, presione conectar y seleccione la estación meteorológica!” junto con el botón “Conectar”. Las únicas opciones en este momento son conectarse o salir de la aplicación.

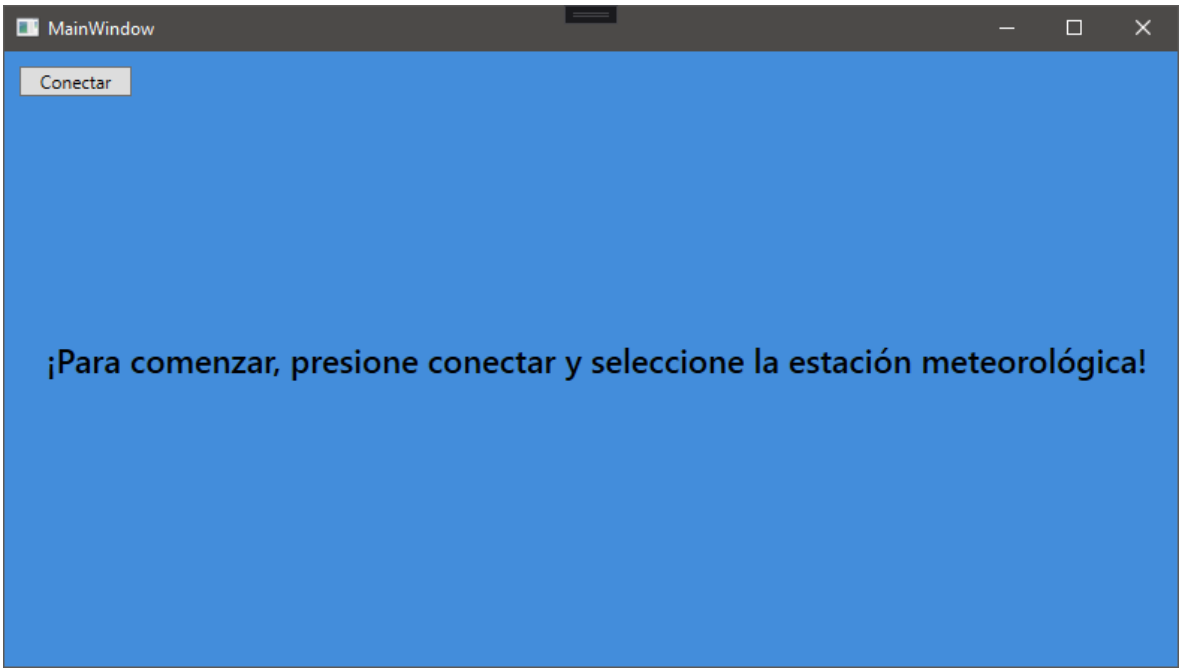


Imagen 21 – Pantalla Inicial de la aplicación WeatherView. [Fuente: Autor].

Pantalla de dispositivos Bluetooth

Al dar clic en conectar, se muestra una pantalla con los diferentes dispositivos bluetooth visibles por el sistema. Se debe seleccionar la estación meteorológica y esperar a que se realice la conexión (puede ser necesario completar pasos extra, en este caso seguir las indicaciones en pantalla). Si la conexión no fue posible, se regresa a la pantalla principal, ahora con la adición de un texto que indica el estado de la conexión al borde inferior derecho de la aplicación.

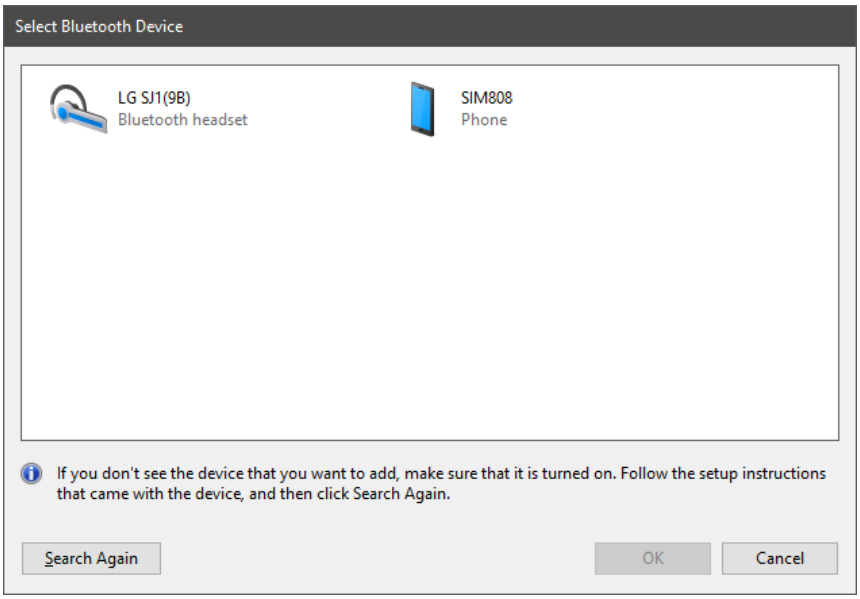


Imagen 22 – Pantalla de selección de dispositivos bluetooth. [Fuente: Autor].

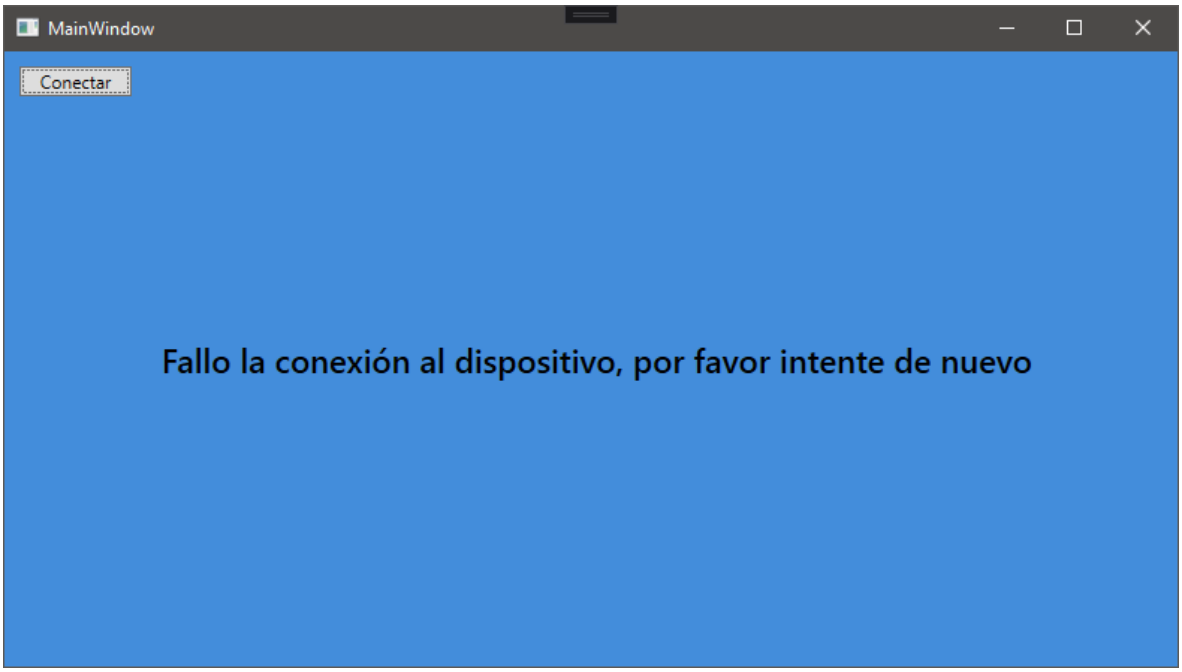


Imagen 23 – Pantalla inicial (Falla en la conexión bluetooth). [Fuente: Autor].

Pantalla Principal

De ser exitosa la conexión con la estación meteorológica, se muestra ahora la pantalla principal, donde se visualizan los datos tomados en tiempo real de esta, además de unas “cartas” que se van añadiendo cada 10 minutos como un histórico rápido de los datos tomados.

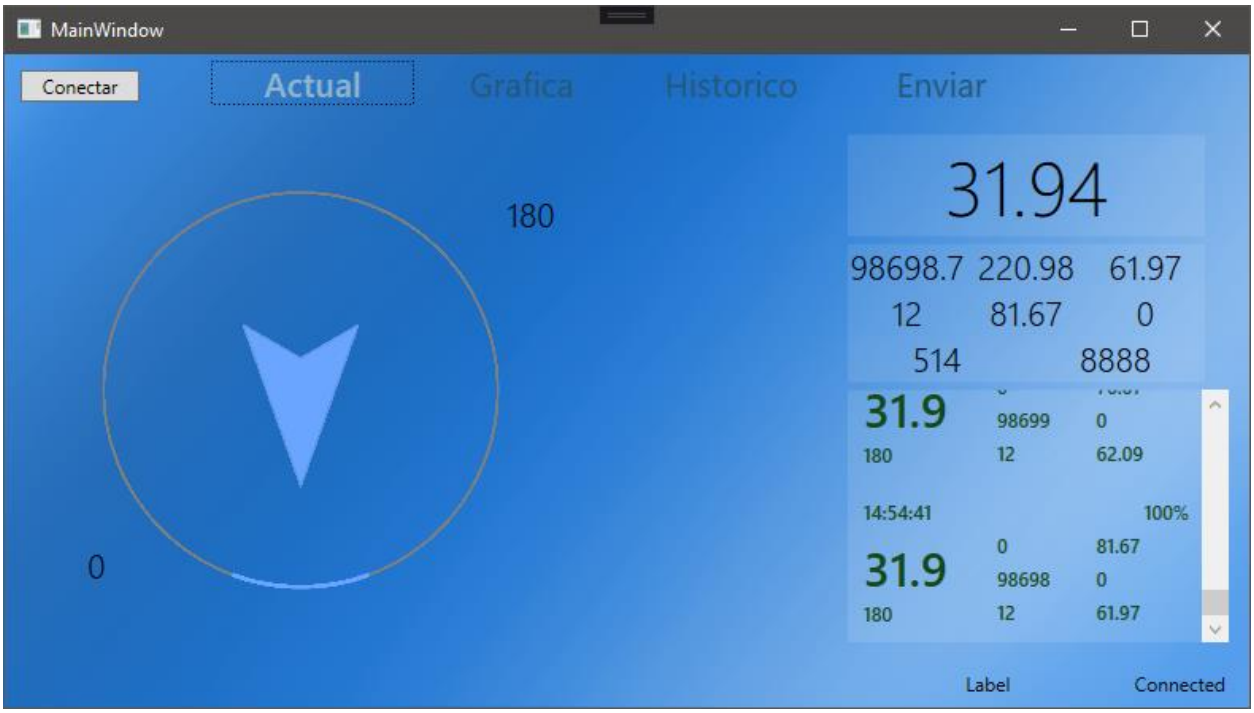


Imagen 24 – Pantalla Principal (Datos actuales). [Fuente: Autor].

Pantalla grafica

En esta pantalla se pueden ver los datos que son tomados por la estación en forma de gráfica, una por cada variable del clima, además, dicha grafica puede ser guardada para su posterior uso por parte del usuario.

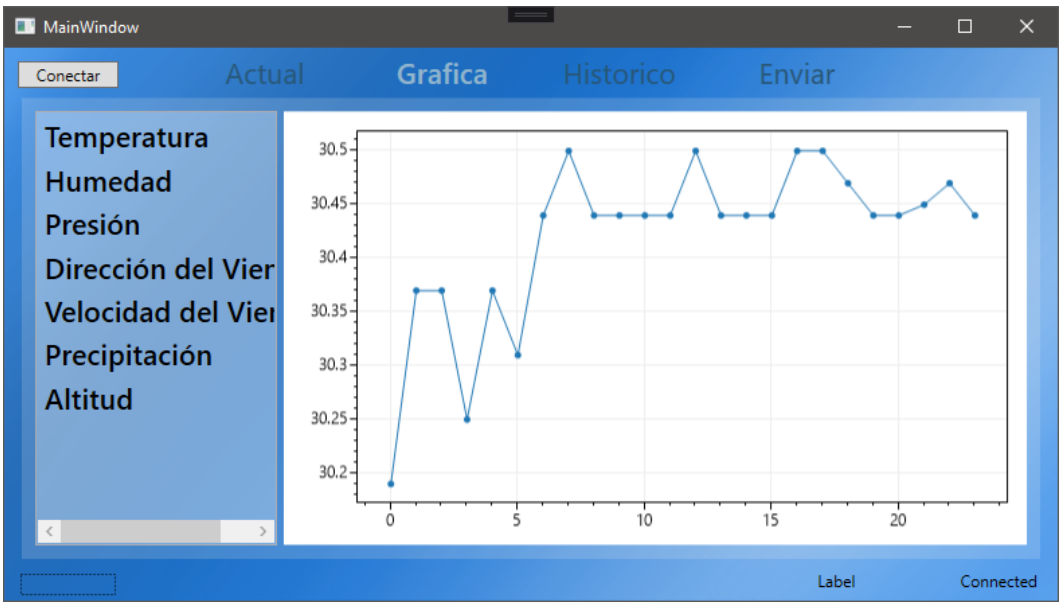
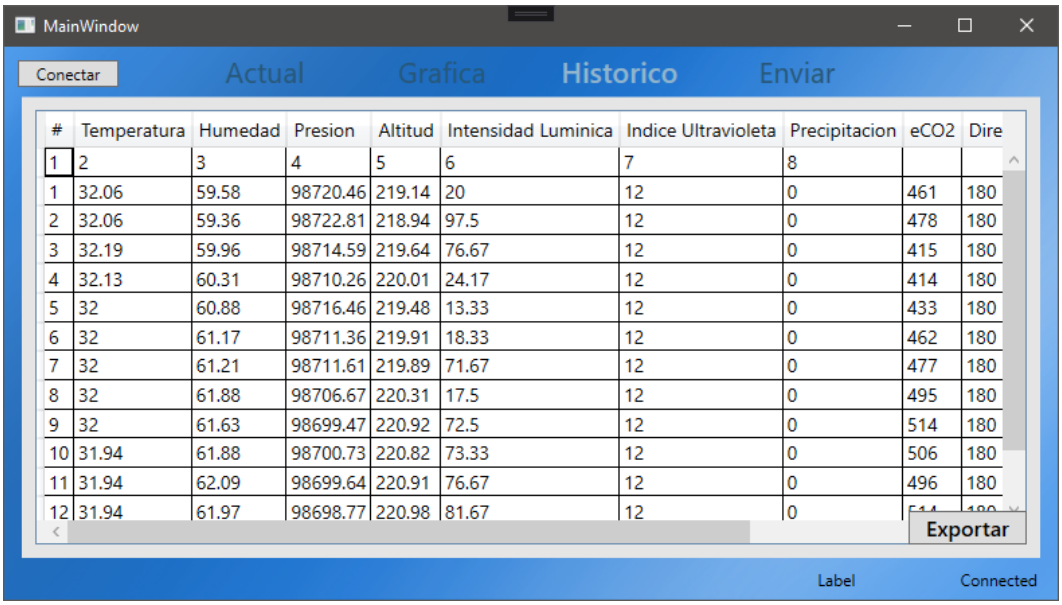


Imagen 25 – Pantalla de grafica (temperatura). [Fuente: Autor].

Histórico

El histórico muestra todos los datos que han sido reportados por la estación desde el momento en que la aplicación fue abierta. Además, muestra también los datos correspondientes al tiempo de actividad y a la fecha y hora UTC (sin importar si hay o no “arreglo” del GPS).



#	Temperatura	Humedad	Presion	Altitud	Intensidad Luminica	Indice Ultravioleta	Precipitacion	eCO2	Dire
1	2	3	4	5	6	7	8		
1	32.06	59.58	98720.46	219.14	20	12	0	461	180
2	32.06	59.36	98722.81	218.94	97.5	12	0	478	180
3	32.19	59.96	98714.59	219.64	76.67	12	0	415	180
4	32.13	60.31	98710.26	220.01	24.17	12	0	414	180
5	32	60.88	98716.46	219.48	13.33	12	0	433	180
6	32	61.17	98711.36	219.91	18.33	12	0	462	180
7	32	61.21	98711.61	219.89	71.67	12	0	477	180
8	32	61.88	98706.67	220.31	17.5	12	0	495	180
9	32	61.63	98699.47	220.92	72.5	12	0	514	180
10	31.94	61.88	98700.73	220.82	73.33	12	0	506	180
11	31.94	62.09	98699.64	220.91	76.67	12	0	496	180
12	31.94	61.97	98698.77	220.98	81.67	12	0		

Imagen 26 – Pantalla de datos Históricos. [Fuente: Autor].

Pantalla Envió

Permite escribir un número de teléfono para enviar mensajes de texto (SMS) con los datos de la estación meteorológica (para cada intento se debe presionar “Enviar”)

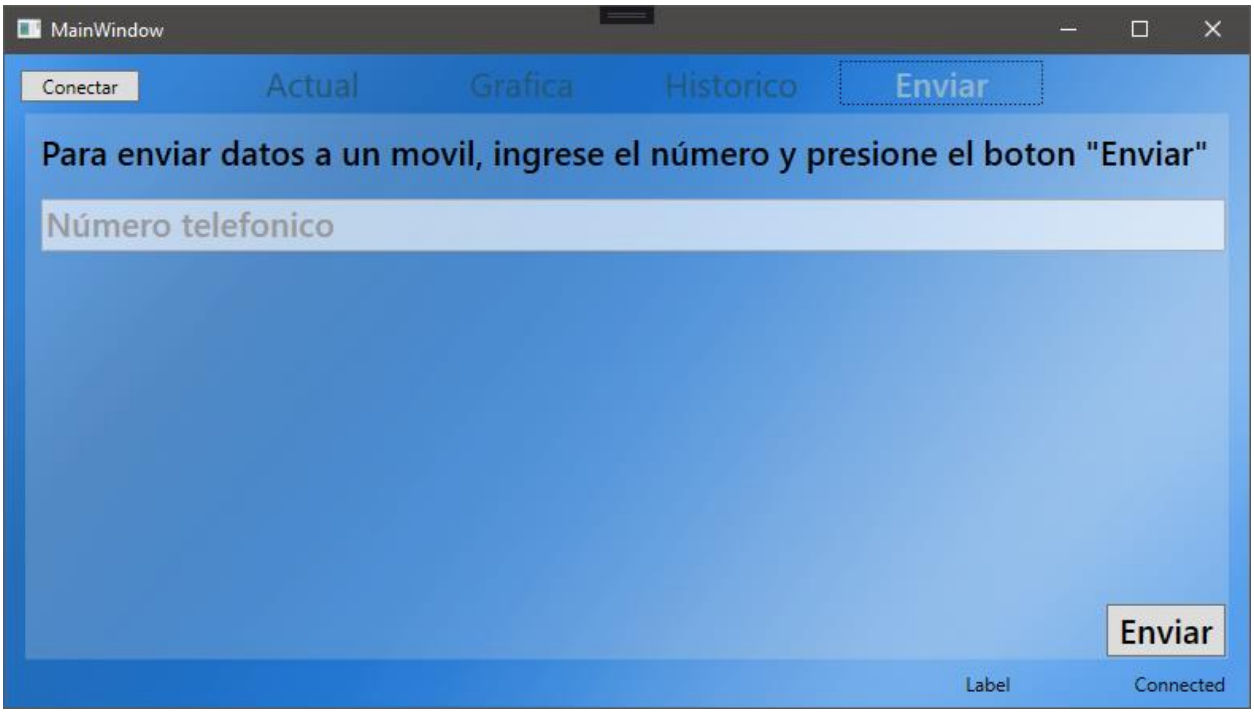


Imagen 27 – Pantalla para enviar datos de la estación meteorológica a un número móvil dado desde la aplicación por mensaje de texto (SMS). [Fuente: Autor].

4.3.2 Aplicación WeatherView (Android)

La aplicación en Android funciona como un monitor para la estación meteorológica, la misma se puede usar en dispositivos con Android 5.1 Lollipop en adelante. La aplicación se conecta a la estación por medio de bluetooth y una vez conectada permite ver los datos en tiempo real de la misma. Para visualizar los datos del GPS (ubicación en el mapa) se hace uso de la aplicación Google Maps, o la aplicación de manejo de mapas que se encuentre instalada en el dispositivo del usuario, donde al dar clic en el botón “GPS” se abre la aplicación disponible y se envía la solicitud para mostrar las coordenadas actuales de la estación. Téngase en cuenta que el botón “GPS” no aparecerá hasta no obtener un “arreglo” del GPS en la estación meteorológica.

Pantalla de Inicio

La primera pantalla que muestra la aplicación se encuentra en blanco, salvo por un mensaje que indica “¡Para comenzar, presione conectar y seleccione la estación meteorológica!” junto con el botón “Conectar”. Las únicas opciones en este momento son conectarse o salir de la aplicación.

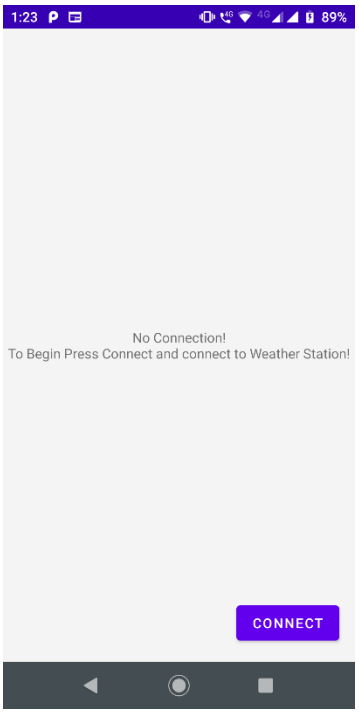


Imagen 28 – Pantalla Inicial aplicación Android (Sin conexión). [Fuente: Autor].

Pantalla de dispositivos Bluetooth

Una vez se presiona el botón "Conectar" aparece una nueva ventana en la que se muestran los dispositivos bluetooth que ya han sido emparejados con el dispositivo. De no ser el caso, aparecerá el mensaje de "¡Activa el bluetooth o Empareja con la estación meteorológica!" indicando que se debe realizar este proceso antes de poder seguir con la conexión a la misma.

Imagen 29 – Pantalla de conexión (Bluetooth apagado o sin emparejar dispositivos). [Fuente: Autor].

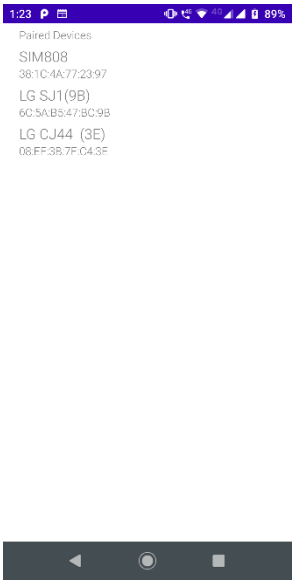


Imagen 30 – Pantalla de conexión (Lista de dispositivos emparejados). [Fuente: Autor].

Si al seleccionar la estación meteorológica no es posible establecer la conexión bluetooth, la aplicación regresa a la pantalla inicial, mostrando ahora el mensaje de “No ha sido posible comunicarse con la estación meteorológica, por favor intenta de nuevo”

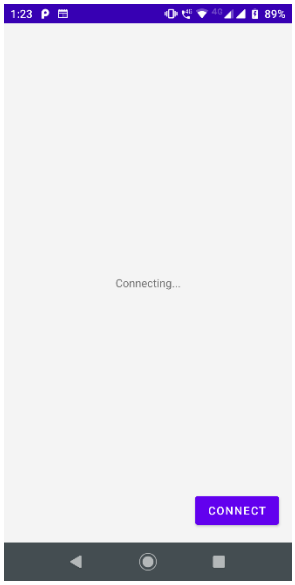


Imagen 31 – Pantalla principal (Intentando Conectar a dispositivo). [Fuente: Autor].

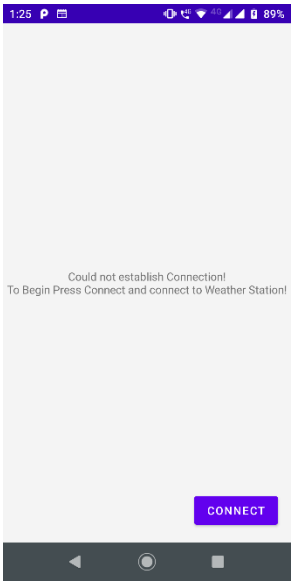


Imagen 32 – Pantalla principal (Falla en la conexión al dispositivo). [Fuente: Autor].

Pantalla Principal

Establecida la conexión, la aplicación muestra los datos de temperatura, humedad, presión, altitud, velocidad y dirección del viento, precipitación, intensidad de luz, índice UV, eCO2 y TVOC. No es necesario mostrar la hora puesto que la misma ya es visible en el dispositivo. Así mismo, como se mencionó anteriormente, los datos de la ubicación GPS (Latitud y Longitud) no están disponibles hasta que la estación meteorológica no tenga un “arreglo”. Una vez se tenga, se muestra un botón “GPS”, con el cual se procede a abrir la aplicación de mapas predeterminada y mostrar la ubicación de esta.

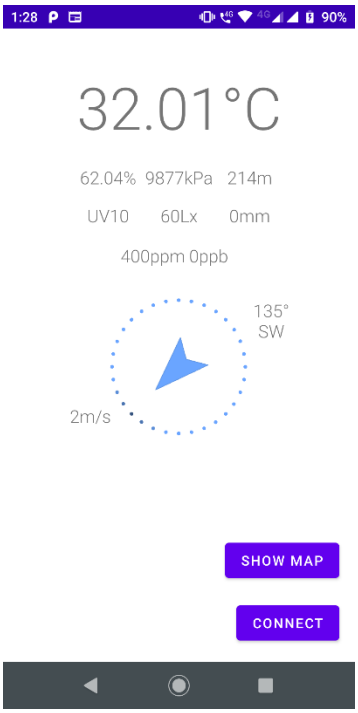


Imagen 33 – Pantalla principal (Recibiendo datos de la estación meteorológica). [Fuente: Autor].

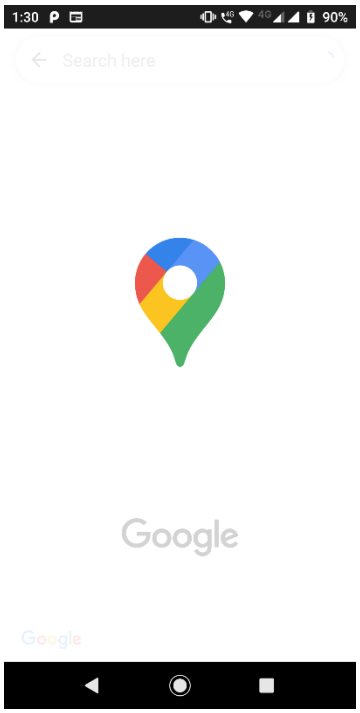


Imagen 34 – Invocando Google Maps para visualización del GPS. [Fuente: Autor].

Imagen 35 – Visualización de las coordenadas del GPS de la estación meteorológica (en azul, coordenadas del celular actual). [Fuente: Autor].

Durante la etapa de verificación se comprueba que tanto la aplicación de Windows como la aplicación de Android funcionan de forma correcta y se comunican con el módulo SIM sin problema alguno.

4.4 Energía

Durante la etapa de verificación se comprueba que el sistema puede dar energía a la estación sin problema alguno, más aún, al probar el funcionamiento de los sensores junto al microcontrolador (sin el módulo SIM), se evidencia un consumo mucho más reducido de lo esperado (50mAh), con lo que se asegura el funcionamiento correcto de este.

Imagen 36 – Consumo del Arduino con MicroSD y Sensores. [Fuente: Autor].

Para soportar la carga por panel solar y por conexión DC, se utiliza un relé, el cual será auto enclavado cuando se conecte una fuente DC. Esto quiere decir que el panel solar es la fuente de energía por defecto, y que el relé se alimenta directamente de la fuente DC. Además, el módulo TP4056 protege las baterías al poder desconectar la carga cuando el voltaje es crítico, o desconectar la fuente cuando el voltaje es muy bajo (se reúsa a cargar). El diagrama de conexiones para el sistema es el siguiente.

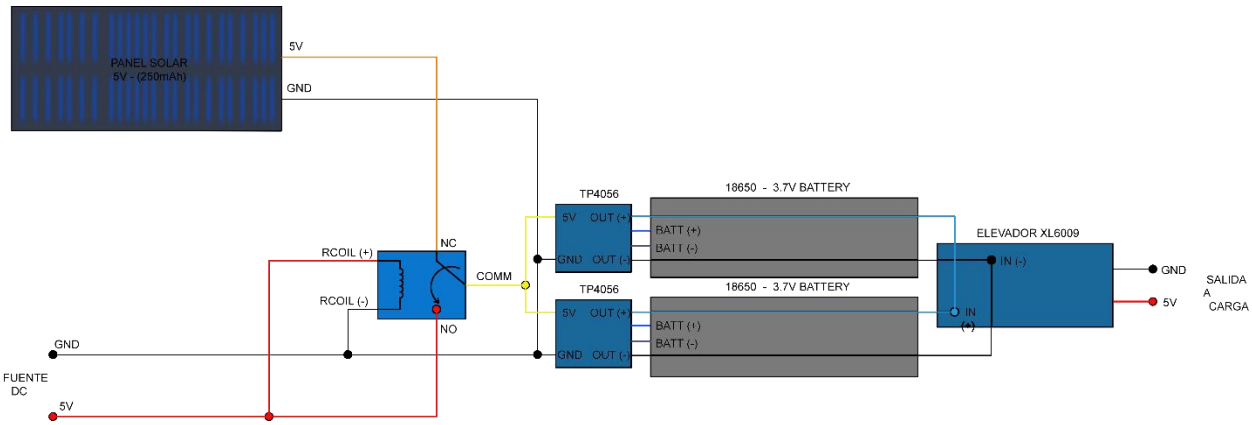


Ilustración 48 – Diagrama esquemático de conexiones para el circuito de energía. [Fuente: Autor].

4.5 Ensamble

Al acoplar los diferentes sistemas y armar la estructura se obtiene:

Imagen 37 – Estación Meteorológica. [Fuente: Autor]

Imagen 38 – Estación Meteorológica. [Fuente: Autor].

Imagen 39 – Estación Meteorológica. [Fuente: Autor].

Imagen 40 – Estación Meteorológica. [Fuente: Autor].

Conclusiones

La medición de los parámetros del clima permite preparar al interesado para disponer de medidas respecto a los planes que tenga establecidos

El clima es variable, sin embargo, en ciertas zonas puede haber periodos de “estabilidad”, seguidos de periodos donde se pueden presentar variaciones en el clima que sean frecuentes.

El constante monitoreo de los gases permite detectar momentos de alta actividad en un ambiente, lo cual permite tomar medidas para mantener un ambiente controlado

La comunicación inalámbrica permite que una persona no tenga que estar físicamente al lado del dispositivo para leer los datos de este

El monitoreo del clima es esencial en el estudio de cómo cambian las variables a lo largo del día, meses, años, sin embargo, esto depende de que la estación pueda estar funcionando todo el tiempo, lo cual no es posible si la estación solo opera en horarios de servicio (como en el caso de la estación del Aeropuerto Alfonso López Pumarejo)

Es ideal disponer de un amplio espacio libre de obstáculos o paredes si se quiere tener con certeza mediciones respecto al viento o a la intensidad de luz, pues estas medidas pueden ser afectadas por el ambiente en el que se encuentran (sin necesariamente ser afectadas las otras mediciones).

Bibliografía

[1] Ruiz Ayala, Daniel Camilo. "*DESARROLLO DE UN SISTEMA DE MONITOREO INALÁMBRICO DE ALGUNAS VARIABLES CLIMÁTICAS*". Pamplona, 2016, Trabajo de grado (título de Ingeniero Electrónico). Universidad de Pamplona. Facultad de Ingenierías y Arquitectura.

[2] Ferrer Sanabria Rafael David. "*Desarrollo de una estación meteorológica autónoma de bajo costo*". Bucaramanga, 2017, Trabajo de grado (título de Ingeniero de telecomunicaciones). Universidad Santo Tomás Bucaramanga. Facultad de Ingeniería de Telecomunicaciones.

[3] Garzón Guzmán, Brian Yesid y Rincón Cerón, María Fernanda. "*DISEÑO E IMPLEMENTACIÓN DE UN PROTOTIPO DE ESTACIÓN METEOROLÓGICA PARA LA MEDICIÓN DE VARIABLES AMBIENTALES*". Bogotá, 2017. Proyecto de grado (título de Ingeniero Eléctrico). Universidad Distrital Francisco José de Caldas. Facultad de Ingeniería.

[4] Gavíria Valencia, Raul Alberto. "*PROTOTIPO FUNCIONAL ESTACIÓN METEOROLÓGICA PORTABLE CON DISPOSITIVOS DE BAJO COSTO (ARDUINO)*". Pereira, 2018. Proyecto de grado e investigación (título de Ingeniero de Sistemas). Universidad Libre seccional Pereira. Facultad de Ingeniería

[5] Tobajas Garcia, Alberto. "*Diseño e implementación de una estación meteorológica con Raspberry Pi* ". Barcelona, 2016. Trabajo final (Máster en Ingeniería de Telecomunicaciones). Universidad abierta de Cataluña.

[6] Mahmoud Shaker Nasr, Walla'a Y.Y. Alebady. *Design and Implementation of a Smart Weather Station Based on Internet of Things*. Journal of Babylon University. 2017. Vol. (25), No. (5); 1769.

[7] Morón, C.; Diaz, J.P.; Ferrández, D.; Saiz, P. *Design, Development and Implementation of a Weather Station Prototype for Renewable Energy Systems*. Energies. 2018. 11, 2234

[8] Mary Nsabagwa, Maximus Byamukama, Emmanuel Kondela, Julianne Sansa Otim. *Towards a robust and affordable Automatic Weather Station*. Development Engineering, 2019. Vol (4).

[9] NATIONAL GEOGRAPHIC, Encyclopedia, "*Meteorology*" [En línea]. Disponible: https://www.nationalgeographic.org/encyclopedia/meteorology/ [Visto por última vez: 14/09/2020]

[10] SCIENCE DAILY, Reference Terms, "*Meteorology*" [En línea]. Disponible: https://www.sciencedaily.com/terms/meteorology.htm [Visto por última vez: 14/09/2020]

[11] SCIENCE DIRECT, Earth and Planetary Sciences, "*Meteorology*" [En línea]. Disponible: https://www.sciencedirect.com/topics/earth-and-planetary-sciences/meteorology [Visto por última vez: 14/09/2020]

[12] CAMBRIDGE DICTIONARY, "*weather station*" [En línea]. Disponible: https://dictionary.cambridge.org/dictionary/english/weather-station [Visto por última vez: 14/09/2020]

[13] PCE Instruments, "*What are Weather Stations?*" [En línea]. Disponible: https://www.industrial-needs.com/technical-data/link/what-are-weather-stations.htm [Visto por última vez: 14/09/2020]

[14] IDEAM, "*DEFINICIONES DEL CATÁLOGO NACIONAL DE ESTACIONES*" [En línea]. Disponible: http://www.ideam.gov.co/documents/10182/557765/Definiciones+CNE.pdf/25e1cca5-ee47-4eaf-86c0-c4f5192a9937 [Visto por última vez: 14/09/2020]

[15] IDEAM, "*ACERCA DE LA ENTIDAD*" [En línea]. Disponible: http://www.ideam.gov.co/web/entidad/acerca-entidad [Visto por última vez: 14/09/2020]

[16] Circuit Basics, *"BASICS OF THE SPI COMMUNICATION PROTOCOL"* [En línea]. Disponible: https://www.circuitbasics.com/basics-of-the-spi-communication-protocol/ [Visto por última vez: 28/11/2020]

[17] Sparkfun, *"Serial Peripheral Interface"* [En línea]. Disponible: https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi/all [Visto por última vez: 28/11/2020]

[18] Circuit Basics, *"BASICS OF THE I2C COMMUNICATION PROTOCOL"* [En línea]. Disponible: https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/ [Visto por última vez: 28/11/2020]

[19] NXP Semiconductor, *"I2C bus specification and user manual"* [En línea]. Disponible: https://www.nxp.com/docs/en/user-guide/UM10204.pdf [Visto por última vez: 28/11/2020]

[20] Sparkfun, *"I2C - Tutorial"* [En línea]. Disponible: https://learn.sparkfun.com/tutorials/i2c/all [Visto por última vez: 28/11/2020]

[21] Maxim integrated, *"Guide to 1-Wire Communication"* [En línea]. Disponible: https://www.maximintegrated.com/en/design/technical-documents/tutorials/1/1796.html [Visto por última vez: 28/11/2020]

[22] Microchip, *"1-Wire® Communication with PIC® Microcontroller"* [En línea]. Disponible: http://ww1.microchip.com/downloads/en/appnotes/01199a.pdf [Visto por última vez: 28/11/2020]

Anexos

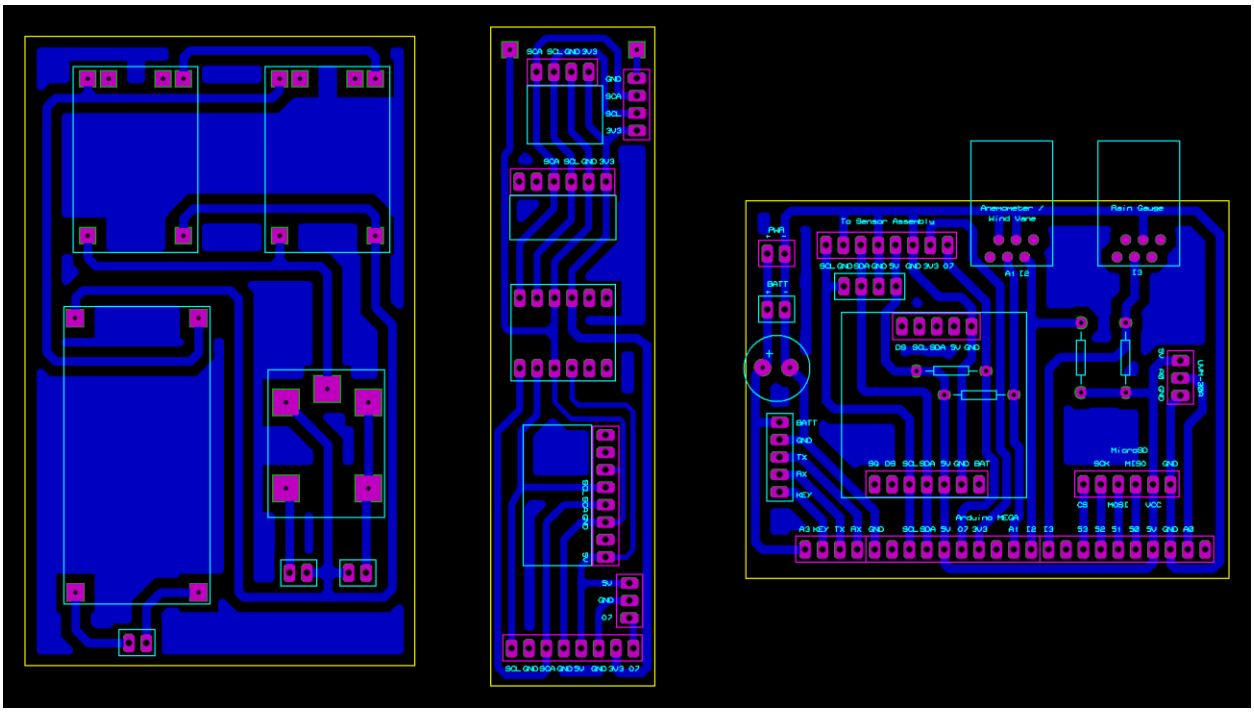


Imagen 41 – Diseño en Baquela (PCB) de las interconexiones para la Estación Meteorológica. [Fuente: Autor].

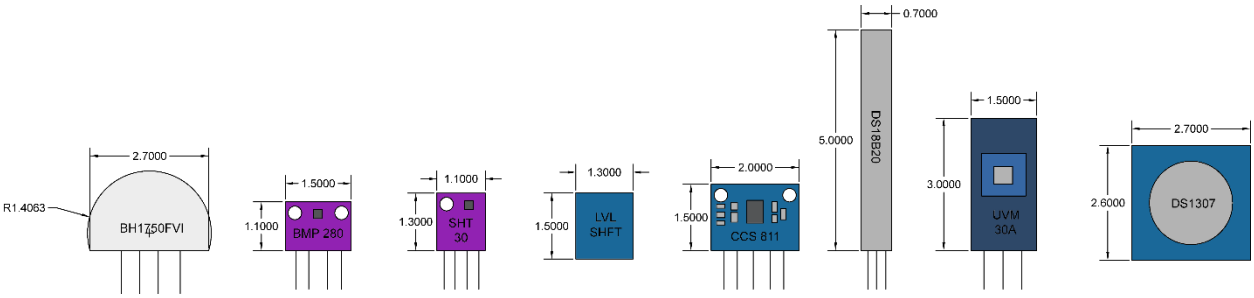


Ilustración 49 – Dimensiones de los dispositivos I2C. [Fuente: Autor].

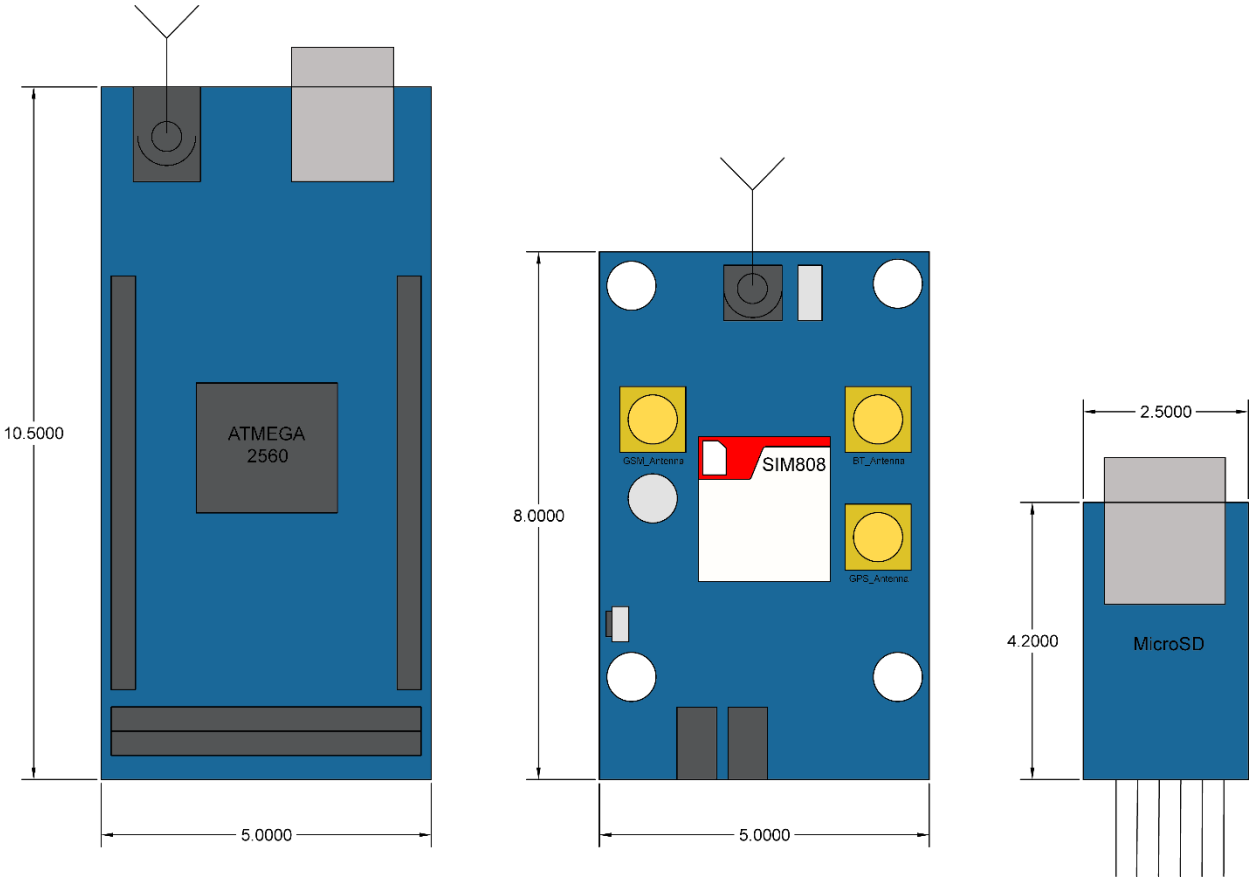


Ilustración 50 – Dimensiones del Arduino MEGA R3, SIM808 y módulo MicroSD. [Fuente: Autor].

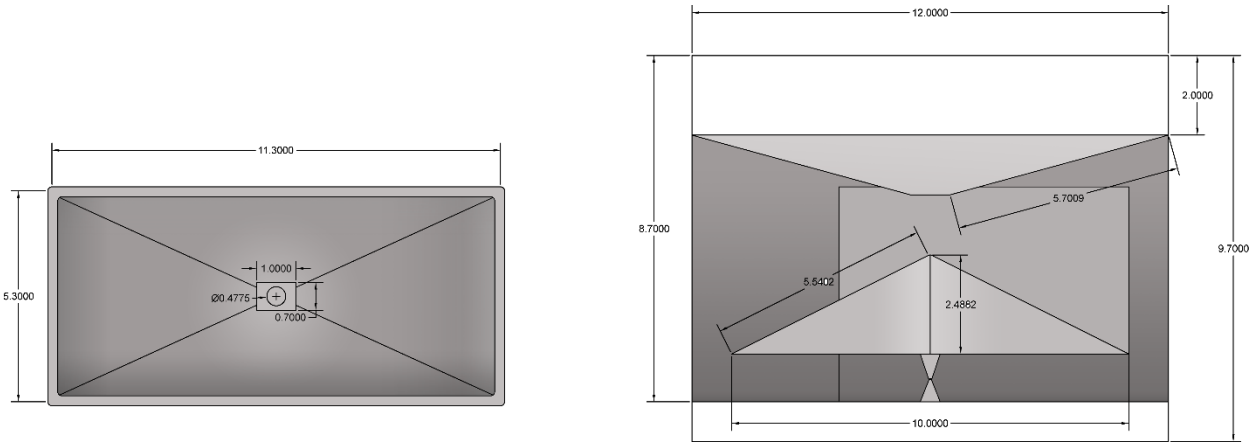


Ilustración 51 – Dimensiones del pluviómetro (Vista Superior y Lateral con Corte Interno). [Fuente: Autor].

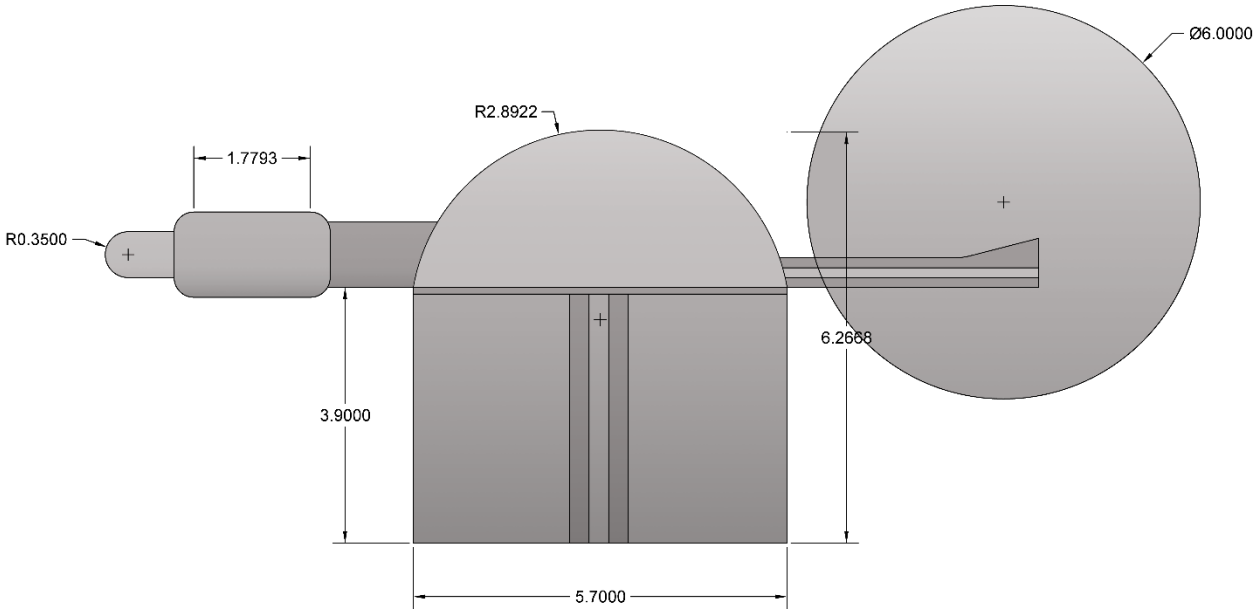


Ilustración 52 – Dimensiones de la Veleta (Vista Lateral). [Fuente: Autor].

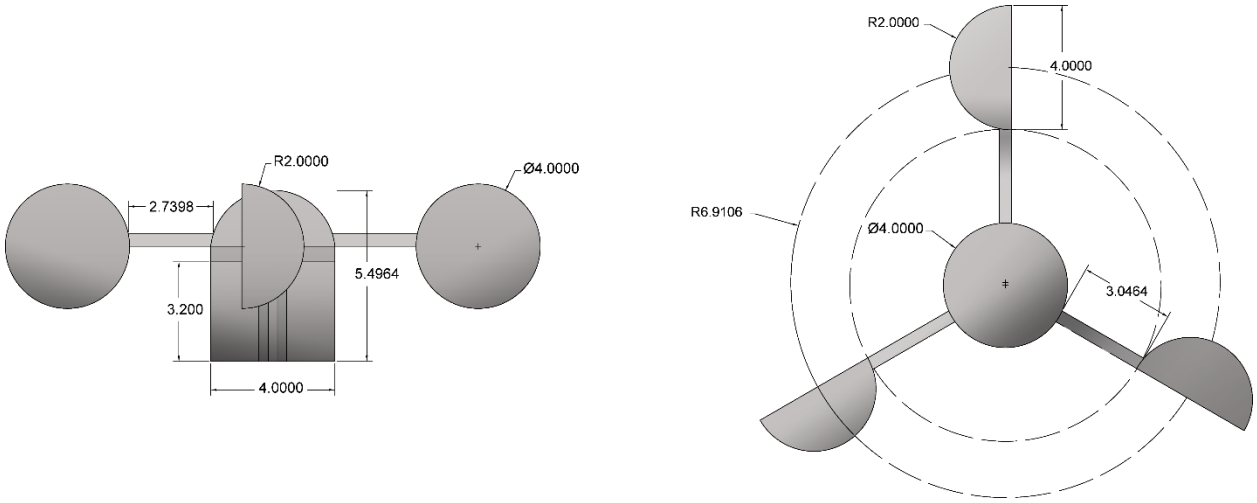


Ilustración 53 – Dimensiones del Anemómetro (Vista Superior y Lateral). [Fuente: Autor].

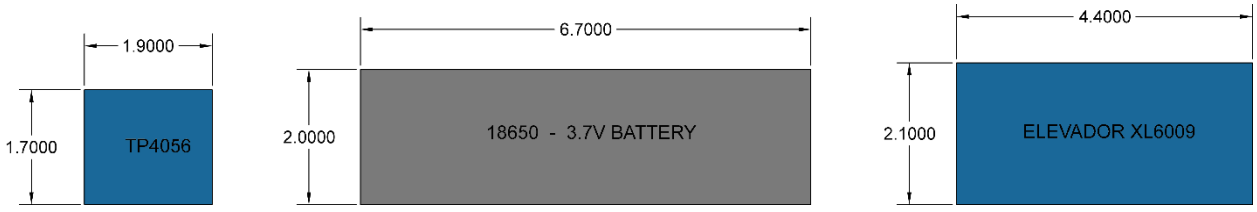


Ilustración 54 – Dimensiones del módulo de carga de baterías TP4056, batería 18650 (con Porta batería) y módulo elevador XL6009. [Fuente: Autor].

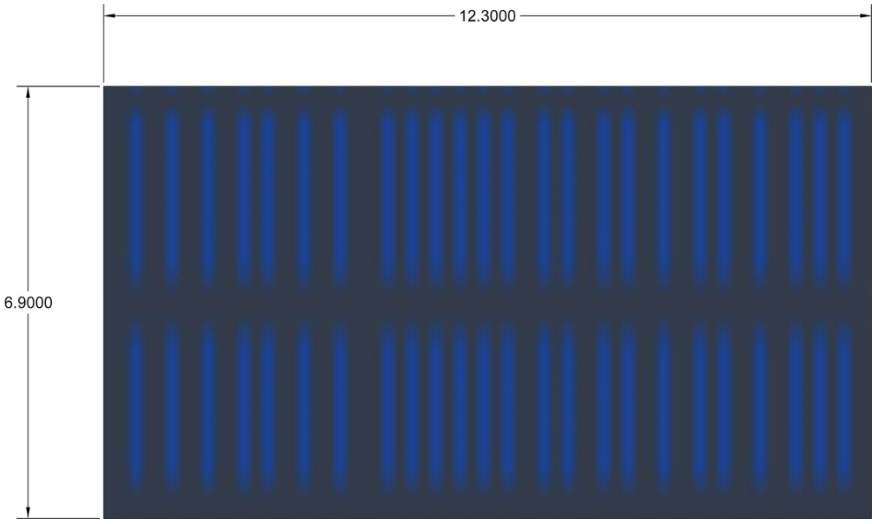


Ilustración 55 – Dimensiones panel solar (250mAh). [Fuente: Autor].