

Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y Sistemas
Arquitectura de Computadores y Ensambladores 2



PRACTICA 1
ESTACIÓN METEOROLÓGICA

NOMBRE	CARNET
Luisa María Ortiz Romero	202003381
Marjorie Gissell Reyes Franco	202000560
Carlos Emilio Campos Moran	201612332
Fredy Samuel Quijada Ceballos	202004812
Brayan Alexander Mejia Barrientos	201900576

Introducción

Las estaciones meteorológicas con Internet of Things (IoT) son una solución para recopilar, almacenar y transmitir datos meteorológicos en tiempo real. Estas estaciones pueden ser utilizadas para estudios científicos, monitoreo a largo plazo del clima, rastreo de tormentas, monitoreo ambiental y mucho más. Estas estaciones ofrecen una solución de datos de confianza y confiables para los propietarios de negocios, agricultores, científicos y otros que necesitan información precisa sobre el clima. Una estación meteorológica con Internet of Things se compone de una variedad de sensores conectados a una plataforma de Internet of Things. Estos sensores miden y recopilan datos meteorológicos y los envían a una plataforma de computación en la nube.

Processing es un lenguaje de programación de código abierto que se utiliza para crear gráficos, animaciones y demás proyectos de arte interactivo. Con ayuda de Processing se visualizaron los datos recuperados de la estación meteorológica de los diferentes parámetros que se solicitaron en la práctica. Mediante un Framework de IoT se visualizó de manera correcta cada uno de los datos solicitados como la velocidad del viento, humedad, temperatura, presión barométrica, y dirección del viento.

Objetivos

- Diseñar un dispositivo destinado a medir y registrar regularmente diversas variables meteorológicas.
- Implementar una aplicación en processing que permita visualizar las mediciones y observaciones puntuales de los diferentes parámetros meteorológicos.
- Aprender a desarrollar una manera correcta de la visualización de los datos mediante la implementación de framework de IoT.

Anemómetro

Para las variables relacionadas con el viento se utilizó un anemómetro, este nos permite tomar medidas de la velocidad del aire.

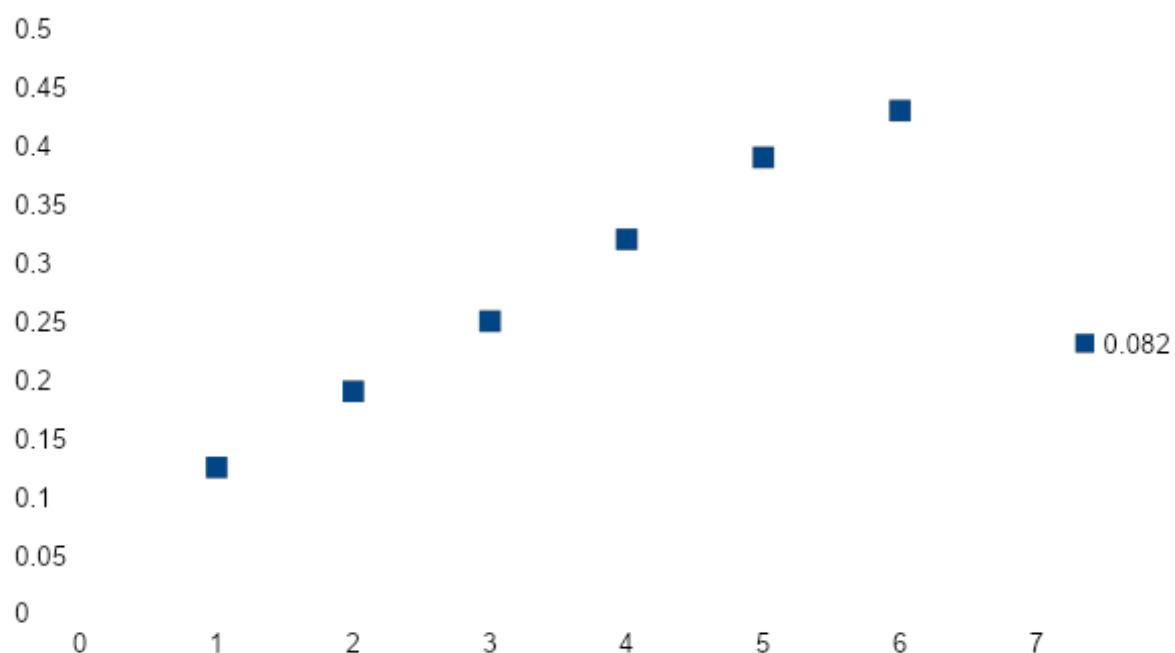
Materiales utilizados

- Un motor DC de 6V
- Cable paralelo
- PVC
- Materiales plásticos para las aspas

Calibración:

Para calibrar el anemómetro se midió el voltaje que devuelve el motor con un tester a diferentes velocidades. Se utilizó un vehículo para realizar estas pruebas dado como resultado los siguientes valores

Corrida	Velocidad del Auto (km/h)	Voltios Registrados durante la prueba con multímetro (v)
1	20	0.082
2	30	0.125
3	40	0.19
4	50	0.25
5	60	0.32
6	70	0.39
7	80	0.43



Al introducir un gráfico de dispersión se puede notar que la gráfica es lineal esto nos permite calcular la pendiente de la curva. Esta pendiente nos servirá para obtener cualquier valor de velocidad según el voltaje que nos devuelva el motor.

$$M = 0.190$$

Ejemplo:

El motor empieza a girar y registra un dato en el arduino con un analogRead de 123 con ayuda de la pendiente de 0.190 se procede a multiplicar 0.190×123 y se obtiene un velocidad de 23.37 km/h.

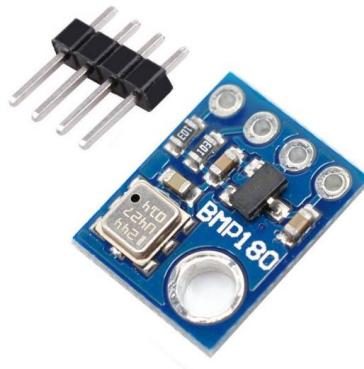


Comprobación

Para comprobar la funcionalidad correcta del sensor se colocó sobre el techo y se procedió a medir con un analogRead, utilizando el Monitor Serial se obtuvo una lectura en tiempo real y se procedió a poner el vehículo en una velocidad de 20 km/h y como resultado se obtuvo una lectura en el monitor serial de 20km/h comprobando que está correctamente calibrado.

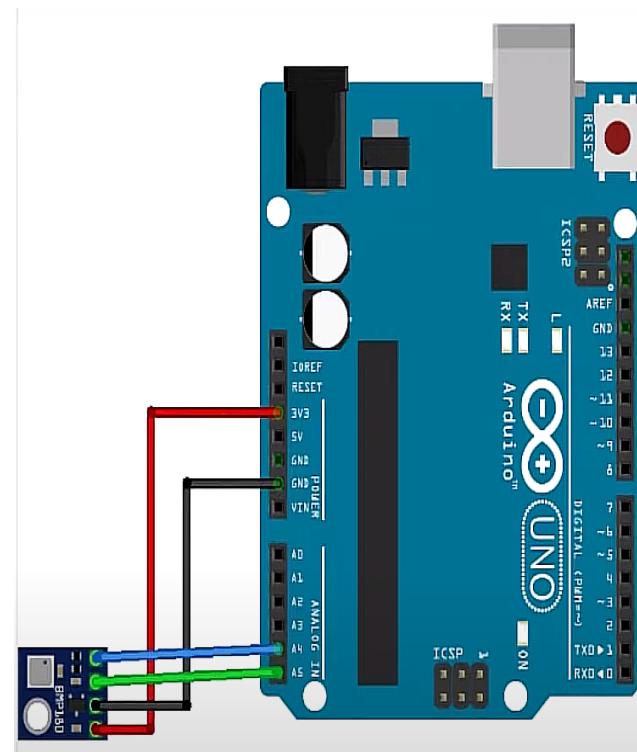
Sensor de Presión Barométrica

El BMP180 es un sensor de presión atmosférica digital de bajo consumo de energía. Está diseñado para medir la presión atmosférica y la temperatura. El BMP180 se puede usar para monitorizar el clima, detectar incendios, controlar el nivel del agua, etc. El BMP180 es un dispositivo de tamaño pequeño y ligero que es fácil de integrar en los proyectos de IoT.



Materiales:

- Estaño
 - Cautín
 - Placa BMP180
 - Pintaderas



Conexiones y configuración:

Implementación de la librerías utilizadas para el sensor.

```
#include <SFE_BMP180.h>
#include <Wire.h>
```

Librería Wire.h:

Librería de arduino que permite a los usuarios conectarse y comunicarse con dispositivos externos a través del protocolo de comunicación I2C. Esta librería proporciona una interfaz simple para enviar y recibir datos entre el Arduino y los dispositivos externos. Esta librería también permite a los usuarios configurar los dispositivos externos, como sensores, actuadores y chips de memoria. Se puede utilizar para controlar la velocidad de comunicación, el nombre de la dirección del dispositivo, el número de direcciones de los dispositivos externos, el número de bytes de los datos, etc.

Librería SFE_BMP180

La librería SFE_BMP180 de Arduino proporciona una interfaz para el sensor de presión BMP180. Esta librería permite a los usuarios leer la presión atmosférica y la temperatura del sensor, así como configurar los parámetros del sensor, como el modo de medición, el nivel de oversampling, el tiempo de espera, etc. Esta librería también permite a los usuarios obtener datos de altitud, calcular la presión a nivel del mar y obtener información sobre el estado del sensor. Esta librería es fácil de usar, segura y estable, lo que la convierte en una excelente opción para los proyectos de IoT.

```
#include <SFE_BMP180.h>
#include <Wire.h>

char status;
double tem,presion;

SFE_BMP180 sensorPresion;

void setup() {
    // put your setup code here, to run once:
    Serial.begin(9600);
    sensorPresion.begin();
}

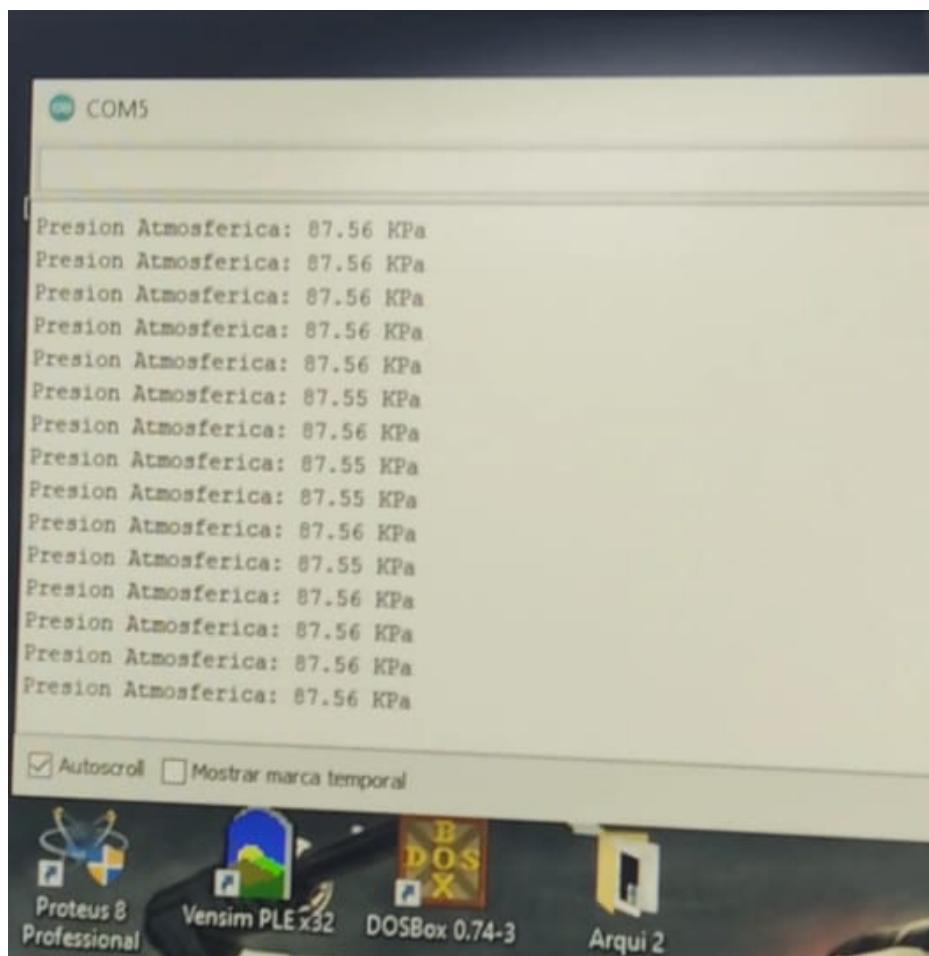
void loop() {
    // put your main code here, to run repeatedly:
    status = sensorPresion.startTemperature();
    delay(status);

    sensorPresion.getTemperature(tem);
    status = sensorPresion.startPressure(3);
    delay(status);

    sensorPresion.getPressure(presion,tem);
    Serial.print("Presion Atmosferica: ");
    Serial.print(presion*0.1,2);
    Serial.println(" KPa");
    // Para convertirlo se debe de multiplicar
    // 1KPa = 7.50062 mmHg
}
```

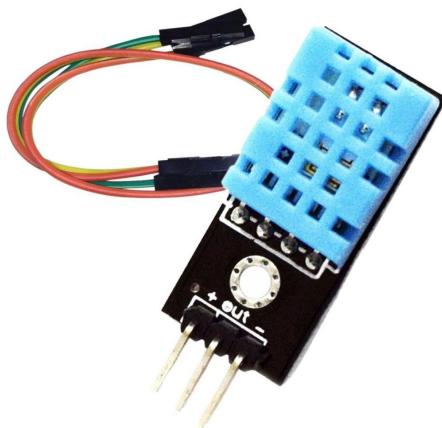
Ejemplo:

Se probó el sensor dentro de una habitación en la ciudad de Guatemala, San Miguel Petapa a las 00:26 horas, dando como resultado una presión de 87.56 kPa que equivalen a 647.44 mmHg.



Sensor de Temperatura y humedad

Un DHT11 es un sensor de temperatura y humedad relativa. Funciona midiendo la temperatura y la humedad del aire usando una resistencia térmica y un capacitor polimérico. El sensor proporciona datos a través de una señal de salida de digital de señal de bajo voltaje. La señal de salida contiene los datos de temperatura y humedad. Estos datos se pueden leer con un microcontrolador, como Arduino. El DHT11 es un sensor de lectura simple, por lo que no requiere calibración. Es asequible, fácil de usar y muy precisa.



Materiales:

- Módulo DHT11
- Jumpers

Librerías utilizadas:

```
#include "DHT.h"
```

Libreria DTH.h

La librería DHT.h es una librería de código abierto para Arduino que se utiliza para comunicarse con sensores de temperatura y humedad. Esta librería contiene funciones para leer los datos del sensor, calcular la temperatura y la humedad, y enviar los datos recogidos a una aplicación de lectura. Esta librería es compatible con varios tipos de sensores, incluyendo el DHT11, DHT22 y AM2302. Esta librería también proporciona funciones de control de errores para asegurarse de que los datos recibidos sean confiables.

Para usar la librería, primero hay que instalarla en el IDE de Arduino. Luego, se deben incluir los archivos necesarios de la librería en el sketch de Arduino. Después de eso, se pueden usar las funciones de la librería para configurar el sensor, leer los datos del sensor, calcular la temperatura y la humedad, y enviar los datos a la aplicación de lectura.

A continuación se muestran las funciones utilizadas para obtener los datos solicitados:

Humedad Absoluta:

La humedad absoluta se calcula con una fórmula usando la temperatura y la humedad relativa. La humedad absoluta es el contenido de vapor de agua en el aire expresado como una fracción de la cantidad de vapor de agua que se necesita para saturar el aire a la misma temperatura. A continuación se muestra el código que contiene las fórmulas para el cálculo de la humedad absoluta.

```
float humedadAbsoluta() {
    //Presión de vapor de saturación a partir de la temperatura en Celcius
    float presionA = 611 * exp((17.27 * temperaturaC)/(237.3 + temperaturaC));
    //Presión de vapor a partir de la presión de vapor de saturación y la humedad relativa
    float presion = (humedadR * presionA) / 100;
    //Temperatura en Celcius a Kelvin
    float temperaturaK = temperaturaC + 273.15;
    //Cálculo de la humedad absoluta en kg/m^3
    humedadA = presion / (temperaturaK * 461.5);
    //Cálculo de la humedad absoluta en k/m^3
    humedadA = humedadA * 1000;
    return humedadA;
}
```

Punto de Rocío:

A continuación se muestra el código que contiene las fórmulas para el cálculo del punto de rocío.

```
float puntoRocio(){
    float number = humedadR/100.0;
    float alpha = log(number) + ((17.625 * temperaturaC) / (243.04 + temperaturaC));
    puntoR = (243.04 * alpha) / (17.625 - alpha);
    return puntoR;
}
```

Dirección del viento

Para conocer la dirección del viento, se realizó utilizando efecto Hall.

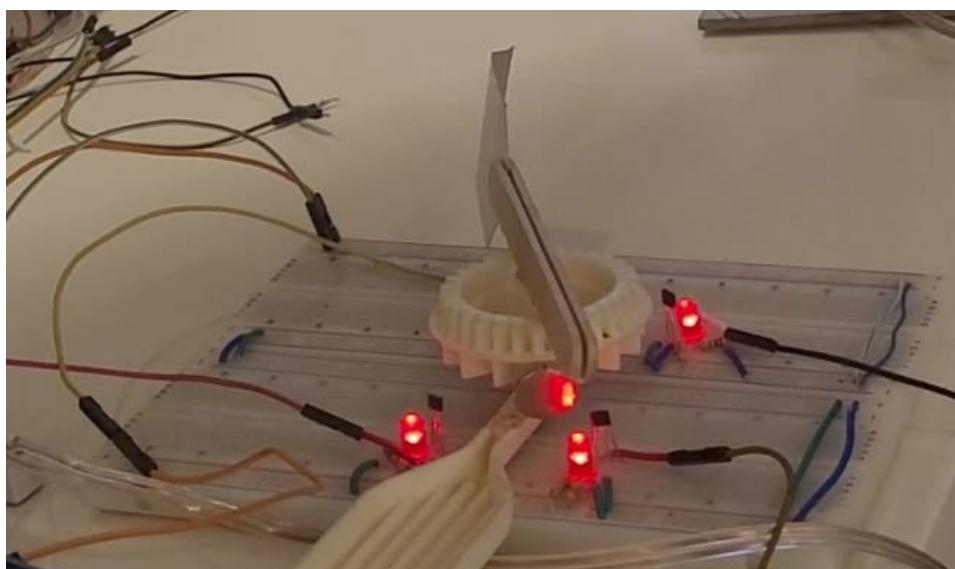


Materiales utilizados

- 4 sensores A3144
- 1 imán de neodimio
- 8 resistencias
- 4 leds
- 1 veleta
- Base con rotación

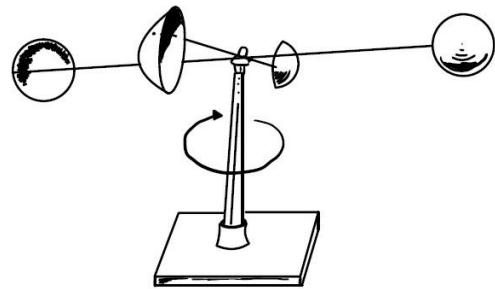
Implementación:

Se conectaron los 4 sensores simulando un punto cardinal (Norte, Sur, Este, Oeste) y se colocó la base con rotación al centro de los sensores. A esta base se le conectó el imán y la veleta. Al iniciar, los 4 sensores estarán activos (se tendrán los 4 leds conectados encendidos) y la veleta, que también está conectada a la base con rotación, girará según sea la dirección del viento. El imán estará girando con este, y al pasar frente a cualquier sensor, este creará un campo magnético que desactivará el sensor (lo apagará junto con el led). El sensor que se encuentre apagado, indicará hacia dónde está apuntando la dirección del viento.

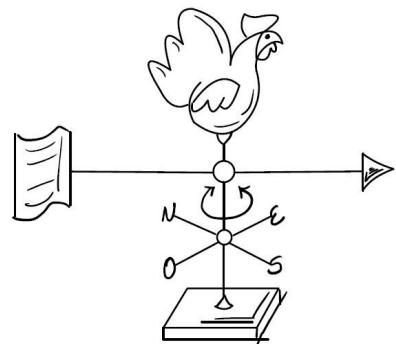


Bocetos del prototipo

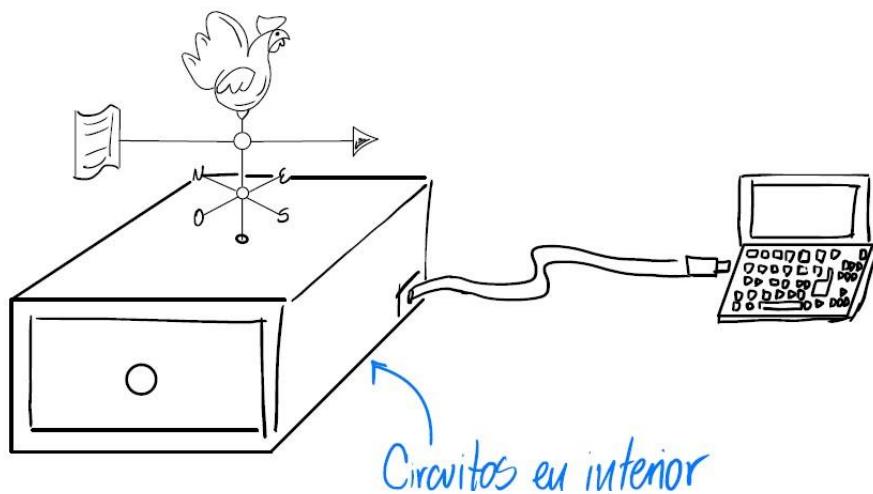
Anemómetro



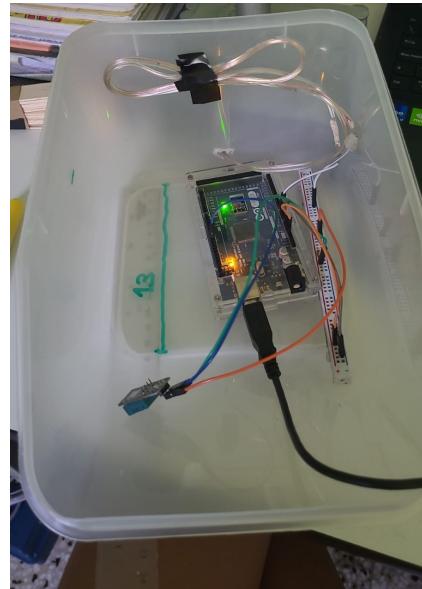
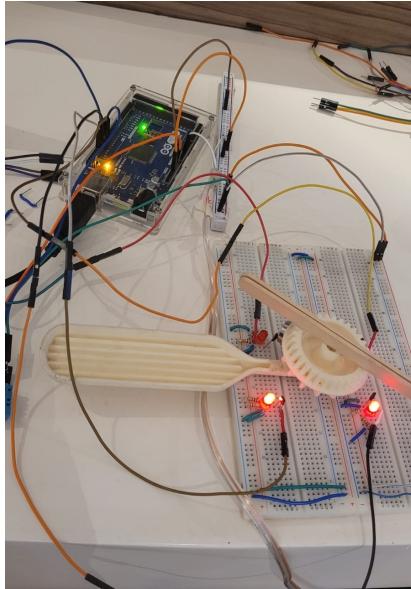
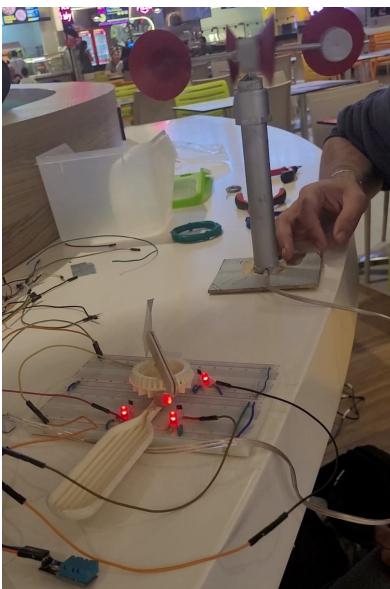
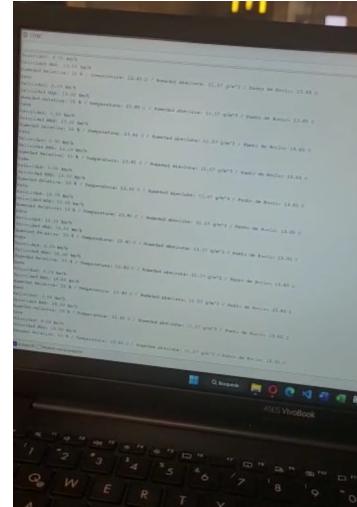
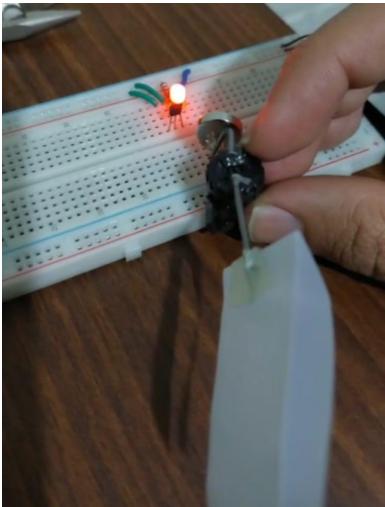
Veleta



Dispositivo para intemperie



Imágenes de la construcción del prototipo



Dispositivo terminado de estación meteorológica:

Al unificar cada uno de los sensores anteriormente detallados y con la ayuda de un arduino mega se obtuvo el primer prototipo. La siguiente imagen muestra nuestro prototipo de una estación meteorológica que nos permite medir y registrar regularmente, diversas variables meteorológicas.

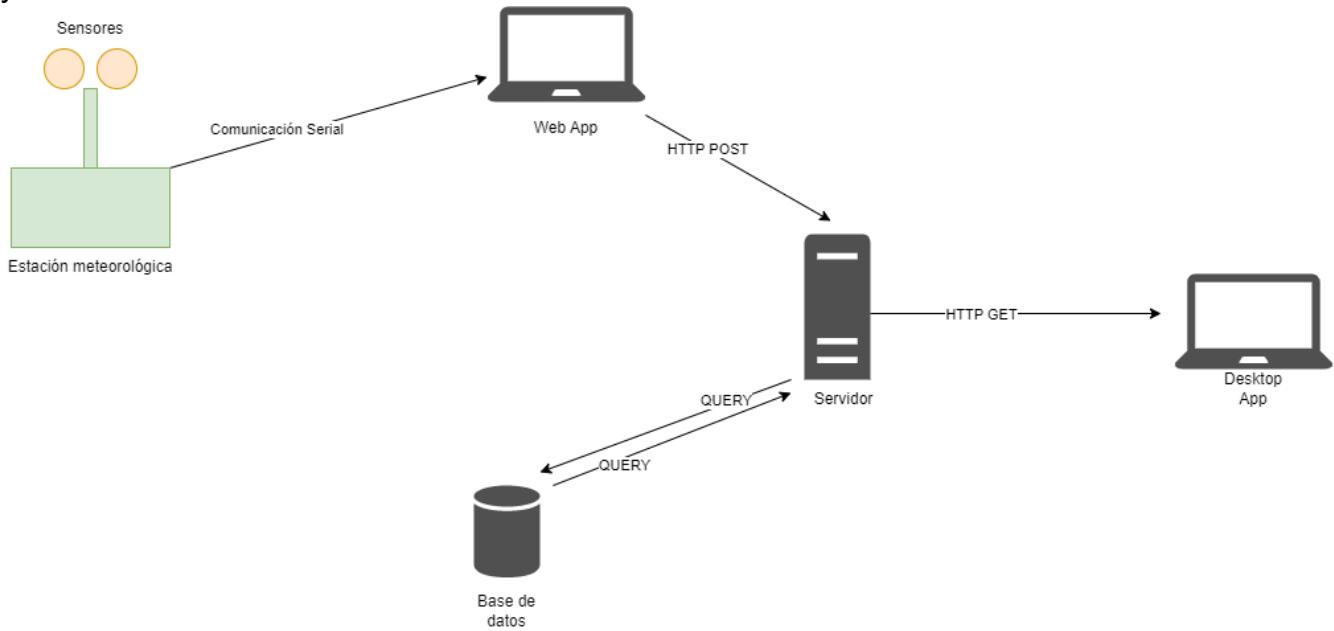


En el siguiente link encontrará un video donde el prototipo está en funcionamiento.

<https://youtu.be/qald08A3v8Y>

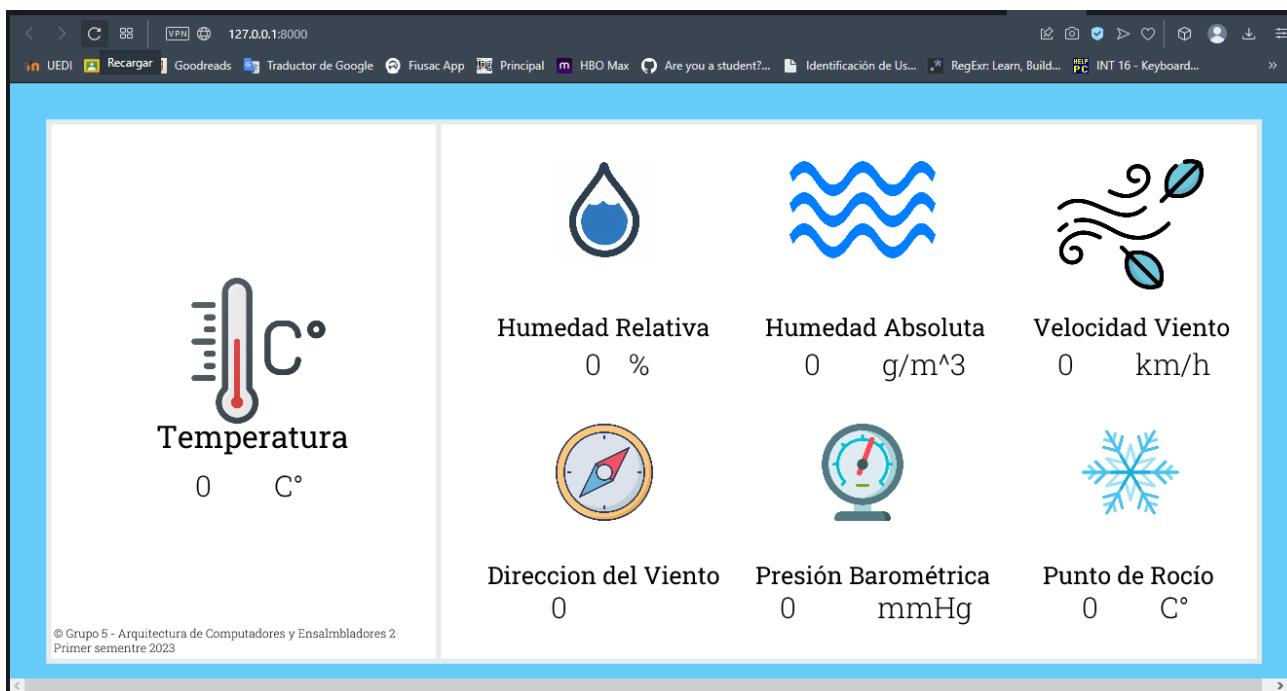
Conectividad

A continuación, se muestra un diagrama de la conectividad utilizada, mostrando las entradas y salidas de la información.



Web App

Se despliega una aplicación web en processing mediante el modo p5.js, el cual permite crear aplicaciones processing mediante scripts y mostrarlas en el navegador. Para levantar la aplicación se utiliza Django 4.1.6, un framework escrito en Python.



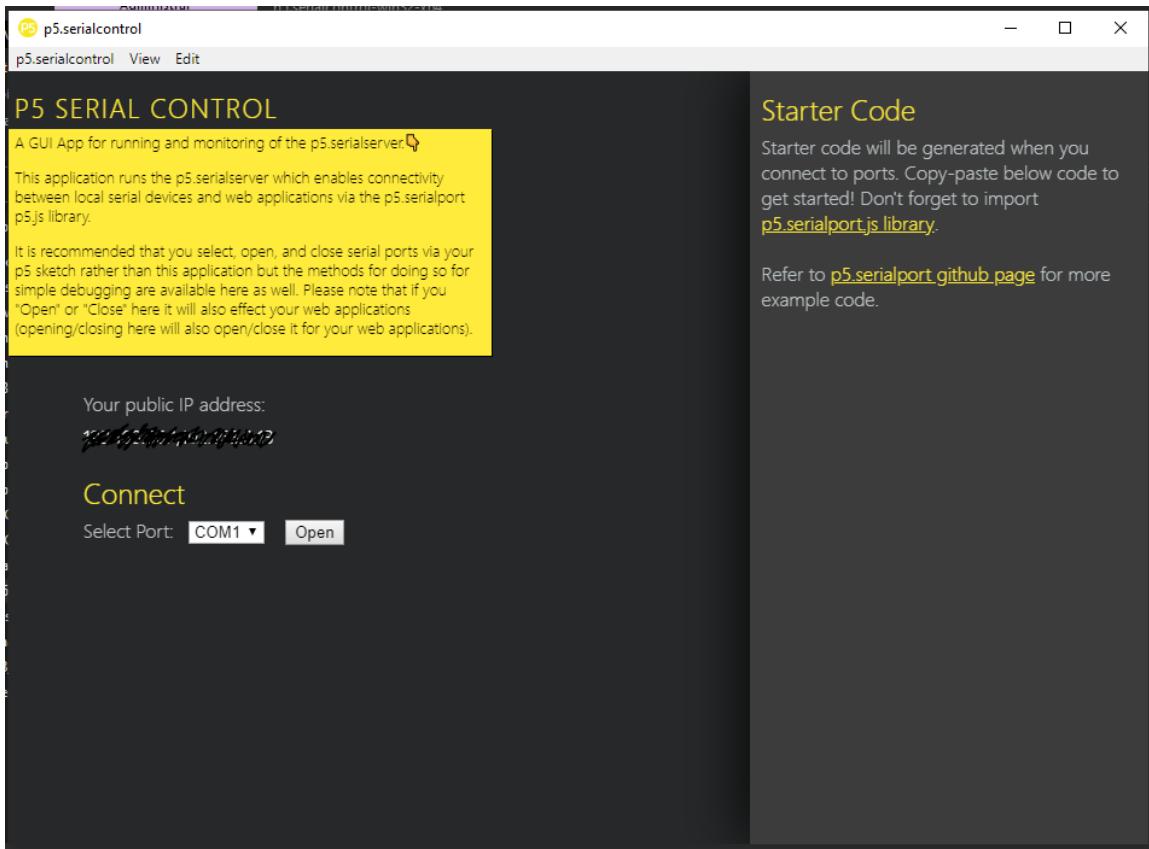
Librería p5.serialport

Esta librería de Processing permite leer y enviar mensajes mediante comunicación serial, extremadamente útil para que la aplicación pueda mostrar los datos de las lecturas en tiempo real.

```
// Función que obtiene líneas de información enviadas mediante comunicación serial
function getData() {
    let currentString = serial.readLine();
    trim(currentString);
    if (!currentString){ latestData="0,0,0,0,0,0,0"; }
    console.log(currentString);
    latestData = currentString;
}
```

Aplicación p5.serialcontrol

Serial Control es una aplicación en javascript que permite al navegador acceder a los puertos seriales de la computadora. Por motivos de seguridad, los navegadores no tienen permitido acceder a los puertos de los dispositivos, pero Serial Control actúa como un intermediario entre los puertos seriales y la aplicación, debe estar en ejecución si se quieren leer datos y mostrarlos en tiempo real.



Servidor

Para recibir las peticiones de la aplicación web, enviar respuestas a la aplicación de escritorio y guardar información en la base de datos, se implementa un servidor local en Python con la librería Flask.

```
● ● ●

#Servidor levantandose en el servidor local, puerto 5000

app = Flask(__name__)

@app.route('/')
def index():

    return 'Hello, World!'

if __name__=='__main__':
    print("Iniciando el servidor")
    main()
    port = int(os.environ.get('PORT', 5000))
    app.run(host='localhost', port=port)
    app.run(debug=True)
```

Base de datos

La base de datos utilizada es SQLite3, para poder realizar operaciones sobre ella desde el servidor en Python. La base de datos consta de 8 campos, 7 para almacenar las diferentes lecturas de la estación meteorológica y 1 para la fecha y la hora de la medición.

```
/*Creación de la tabla desde el servidor*/
clima_table = """ CREATE TABLE IF NOT EXISTS clima (
    fechayHora DATETIME NOT NULL,
    temperatura text NOT NULL,
    humedadRel text NOT NULL,
    humedadAbs text NOT NULL,
    velocidadV text NOT NULL,
    direccionV text NOT NULL,
    presionBar text NOT NULL,
    puntoRocio text NOT NULL
); """
```

Aplicación Processing de Escritorio

La aplicación de escritorio se encarga de realizar consultas a la base de datos mediante peticiones HTTP al servidor, para mostrar gráficamente el historial de las lecturas almacenadas.



Código base para los reportes

```
1 import http.requests.*;
2
3 PImage leftPanel, rightPanel;
4
5 String baseURL = "http://localhost:5000";
6
7 JSONObject json;
8
9
10 void setup(){
11     size(1280,720);
12     frameRate(60);
13
14     background(255);
15 }
16
17 void draw(){
18
19     //image(rightPanel, 0, 0,rightPanel.width/4, rightPanel.height/4);
20
21     String variable = <variable>;
22     String url_temp = "/" + variable;
23     GetRequest get = new GetRequest(baseURL + url_temp);
24     get.send();
25
26     JSONObject response = parseJSONObject(get.getContent());
27
28     int altura = 2;
29
30     int vals_group = response.size() / 10;
31
32
33     for (int i = 0; i < response.size(); i++){
34         json = new JSONObject();
35         json = response.getJSONObject(Integer.toString(i));
36
37         String date = json.getString("fecha");
38         String temp = json.getString(variable);
39
40         //fill colour based on temp
41         //convert string to int
42         int tempInt = Applet.parseInt(temp);
43
44         //make an spiral with the temp
45         int x = 100 + (int) (Math.cos(altura) * altura + width/3);
46         int y = 100 + (int) (Math.sin(altura) * altura + height/3);
47         altura += 1;
48
49
50         // change color by temp
51         if (tempInt < 10){
52             fill(0, 0, 255);
53         } else if (tempInt < 20){
54             fill(0, 255, 0);
55         } else if (tempInt < 30){
56             fill(255, 255, 0);
57         } else if (tempInt < 40){
58             fill(255, 165, 0);
59         } else {
60             fill(255, 0, 0);
61         }
62
63         ellipse(x, y, 10, 10);
64
65         fill(0);
66         if (i % 10 == 0 || i == 0 || i == response.size() - 1){
67             text(date, x, y);
68
69             text(temp, x, y + 10);
70         }
71     }
72 }
73 }
```

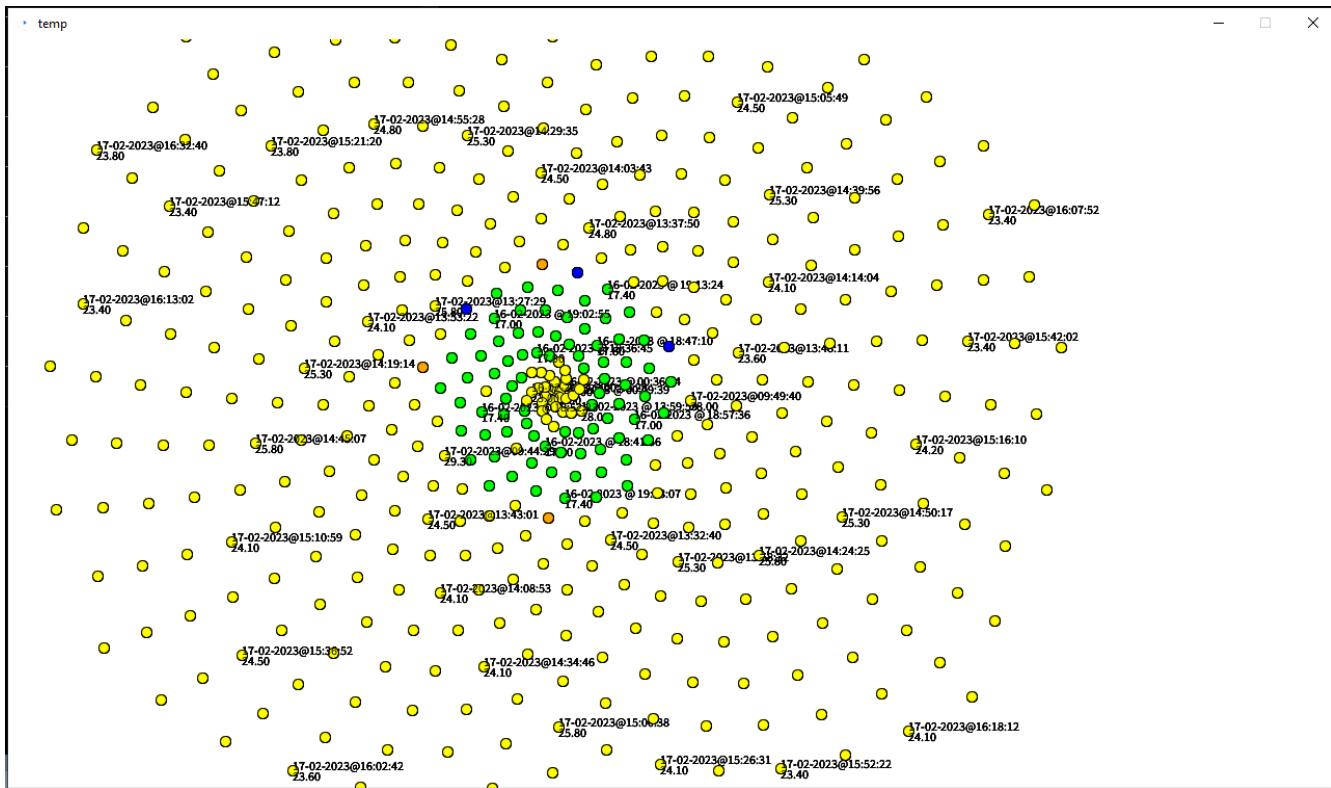
Este código fue el base para la generación de reportes

Realiza una consulta http en la que retorna un json con los datos de la <variable> enviada para cada reporte

Este se hace una conversión y muestra mapeado en forma de espiral los datos, debido a la gran cantidad de datos se muestra solo aquellos dentro de mas de 60s de diferencia entre ellos

al momento de graficarlos solo se da una muestra cada 10 unidades debido a que era mucha información por procesar y mostrar.

El resultado es una gráfico muestral identificado por colores y etiquetado en un rango de cada 10 elementos disponibles



Repositorio en GitHub

https://github.com/LuisaMariaO/ACE2_1S23_G5