PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO

Escuela de Ingeniería Eléctrica
PONTIFICIA UNIVERSIDAD CATOLICA DE VALPARAISO

Diego Rodríguez

Proyecto Final: Estación Meteorológica

EIE507 - Sistemas Embebidos

Escuela de Ingeniería Eléctrica

Valparaíso, 31 de agosto de 2023

Introducción

En un mundo cada vez más enfocado en la recopilación y análisis de datos, la creación de una Estación Meteorológica se presenta como un proyecto apasionante y educativo. En este contexto, la combinación de la versatilidad de Raspberry Pi y la potencia de una base de datos PostgreSQL ofrece una solución robusta y escalable para la captura y almacenamiento de información meteorológica en tiempo real. Este proyecto no solo satisface la curiosidad científica, sino que también sirve como una herramienta práctica para aquellos interesados en comprender y monitorear las condiciones climáticas en su entorno. La combinación de hardware y software permite visualizar y analizar los diferentes datos recibidos. A lo largo de este proyecto, se explora la configuración detallada de la Estación Meteorológica, desde la selección de sensores hasta la instalación y configuración de PostgreSQL para el almacenamiento y análisis de datos meteorológicos. Esta tecnología puede contribuir significativamente al monitoreo y comprensión del clima local.

Palabras claves: Python Web, Web Socket, datos, PostgreSQL.

Índice general

1 Desarrollo

Todo los codigos se pueden encontrar en: https://github.com/diegorodpucv/proyectoeie507

1.1 Arduino

Todo empieza con el microcontrolador, en el cual se conectan los sensores DHT11 y BME280 por conexiones i2c y GPIO permitiendo la detección de temperatura y humedad del ambiente. Además, desde aqui, el Arduino posee un Id qu será traspasado a la base de datos para identificar la procedencia de los datos.

```
#include <DFRobot_DHT11.h>
#define DHT11_PIN 2
DFRobot_DHT11 DHT;
int btnPin = 9;
int btnState = 0;
int prevBtnState = 0;
int ArduinoID = 1;

void setup() {
  Serial.begin(9600);
  pinMode(btnPin, INPUT);
}

void loop(){
  btnState = digitalRead(btnPin);
  if(btnState != prevBtnState){
    if(btnState == HIGH){
      DHT.read(DHT11_PIN);
      int t = DHT.temperature;
      int h = DHT.humidity;

      Serial.print(ArduinoID);
      Serial.print(",");
      Serial.print(t);
      Serial.print(",");
      Serial.println(h);
      delay(1000);
    }
  }
  prevBtnState = btnState;
}
```

1.2 Raspberry Python

Para enviar los datos, es importante que la Raspberry Pi reciba los datos del Arduino, para esto es importante que se utilice la librería ”Serial”. Una vez recibidos los datos, el comando ”split” se usa para separar cada valor requerido. Y finalmente, con las librerías ”time” y ”psycopg2” es posible la transmisión de datos a la base de datos cada 15 minutos.

```
import time
import serial
import psycopg2
import os
RaspID = '1'

ser = serial.Serial('/dev/ttyACM0',9600)
ser.setDTR(False)
time.sleep(1)
ser.flushInput ()
ser.setDTR(True)

conn = psycopg2.connect(
    host="isabelle.db.elephantsql.com",
    database="wgmxepoa",
    user="wgmxepoa",
    password="URKWt_rx7o1HCsjqRONmR9wrB8ep0QI1")

print("Conectado")
cursor = conn.cursor()

while True:
        dato = ser.readline().decode('utf-8').rstrip('\r\n')
        dato2 = dato.split(",")
        ArdID = dato2[0]
        t = dato2[1]
        h = dato2[2]
        timestamp = time.strftime("\'%Y-%m-%d_%H:%M:%S\'")
        ultradatos = (RaspID,ArdID,t,h,timestamp)
        sql = "INSERT INTO public.datos(rasp_id, arduino_id, temp_value,
  hum_value, fecha) VALUES (%s,%s,%s,%s,%s)"
        try:
                cursor.execute(sql,(RaspID,ArdID,t,h,timestamp))
                conn.commit()
                print(f"Datos Subidos Correctamente: {ultradatos}")
                time.sleep(900)

        except Exception as e:
                print(f"Error: {e}")

cursor.close()
conn.close()
ser.close()
```

1.3 postgreSQL

Para la base de datos, como primera instancia se tenían los servicios de Google Cloud para utilizar una máquina virtual que permitiera recibir los datos. La gran desventaja es la alta

vulnerabilidad de los datos una vez subidos, ya que el sistema no poseía ningún tipo de encriptación. Por lo tanto, el servicio de respaldo utilizado finalmente fue elephantsql tal como se muestra en la figura 1.1.

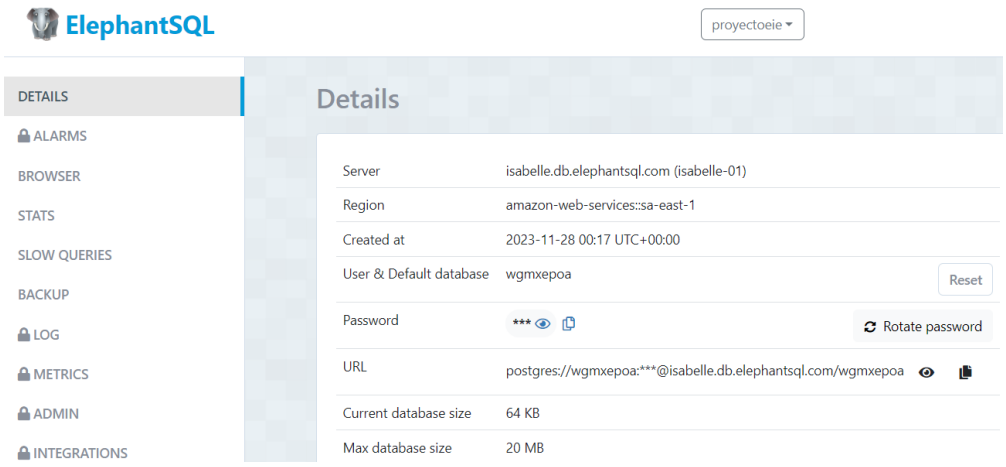
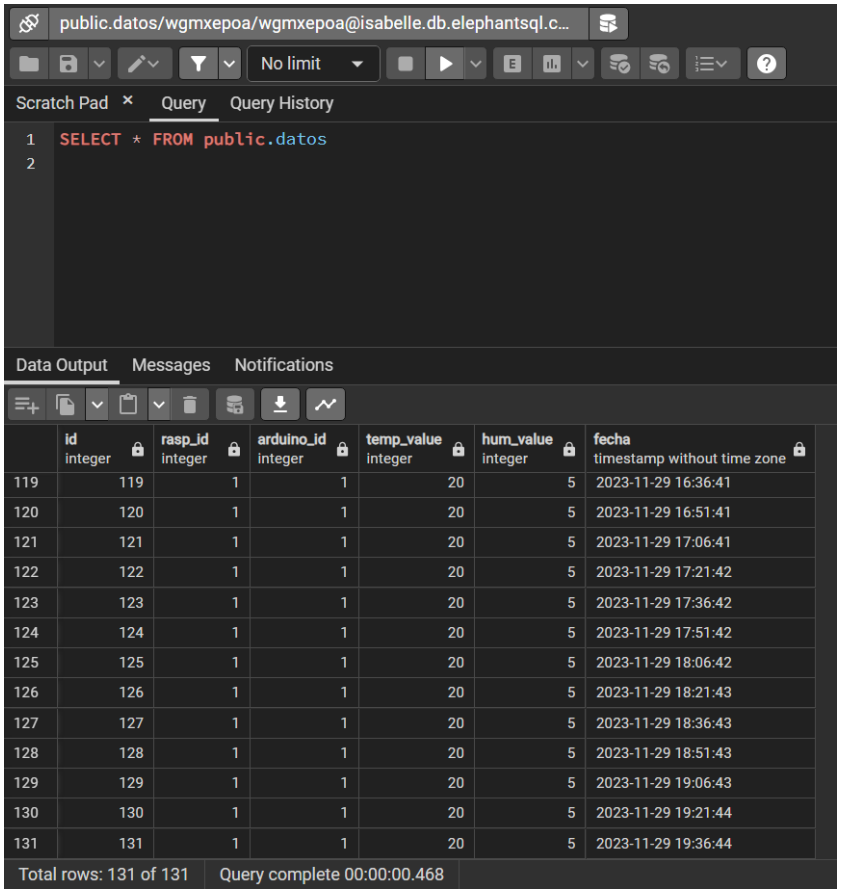


Figura 1.1: Detalle Server postgreSQL

Entonces, una vez configurado y listo para la recepción, se puede utilizar el código python de la raspberry para añadir la base de datos y correr el programa de manera intermitente.



	id integer	rasp_id integer	arduino_id integer	temp_value integer	hum_value integer	fecha timestamp without time zone
119	119	1	1	20	5	2023-11-29 16:36:41
120	120	1	1	20	5	2023-11-29 16:51:41
121	121	1	1	20	5	2023-11-29 17:06:41
122	122	1	1	20	5	2023-11-29 17:21:42
123	123	1	1	20	5	2023-11-29 17:36:42
124	124	1	1	20	5	2023-11-29 17:51:42
125	125	1	1	20	5	2023-11-29 18:06:42
126	126	1	1	20	5	2023-11-29 18:21:43
127	127	1	1	20	5	2023-11-29 18:36:43
128	128	1	1	20	5	2023-11-29 18:51:43
129	129	1	1	20	5	2023-11-29 19:06:43
130	130	1	1	20	5	2023-11-29 19:21:44
131	131	1	1	20	5	2023-11-29 19:36:44

Figura 1.2: Servidor SQL en pgAdmin

1.4 Diseño Web - Metodo GET y POST

El siguiente código utiliza HTML con la url **"http://127.0.0.1/5000"** para que el usuario pueda realizar la petición de datos según las fechas/horas de inicio y fin como rango con el método POST visto y realizado en clases anteriormente.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Generador de Graficos</title>
</head>
<body>
    <h1>Generador de Graficos</h1>
    <form action="/grafico" method="post">
        <label for="fecha_inicio">Fecha de inicio:</label>
        <input type="text" name="fecha_inicio" required>
        <label for="fecha_fin">Fecha de fin:</label>
        <input type="text" name="fecha_fin" required>
        <label for="raspi_id">ID Raspberry:</label>
```

```
        <input type="text" name="raspi_id" required>
        <label for="ardu_id">ID Arduino:</label>
        <input type="text" name="ardu_id" required>
        <button type="submit">Generar Gr fico</button>
    </form>
</body>
</html>
```

Figura 1.3: Página Web: Petición de Datos

Como programa backend, un archivo python recibe el rango de horario ingresado por el usuario del código anterior. Lo siguiente es realizar la conexión con el servidor SQL para realizar la consulta con los siguientes comandos ´´SELECT FROM WHERE BETWEEN´´. Finalmente, gracias a la librería "matplotlib . pyplot" se puede realizar el gráfico de datos con la temperatura y la humedad con sus escalas respectivas. Asimismo, el programa se encarga de guardar esta gráfica en una imagen de formato "PNG".

```
from flask import Flask, render_template, request
import psycopg2
import matplotlib.pyplot as plt
from io import BytesIO
import base64

app = Flask(__name__)

# Conexion base de datos PostgreSQL
conn = psycopg2.connect(
    host="isabelle.db.elephantsql.com",
    database="wgmxepoa",
    user="wgmxepoa",
    password="URKWt_rx7o1HCsjqRONmR9wrB8ep0QI1")

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/grafico', methods=['POST'])
def generar_grafico():
    fecha_inicio = request.form['fecha_inicio']
    fecha_fin = request.form['fecha_fin']
    raspi_id = request.form['raspi_id']
    ardu_id = request.form['ardu_id']

    # Consulta base de datos
    cursor = conn.cursor()
    query = "SELECT fecha, temp_value, hum_value FROM datos WHERE fecha BETWEEN %
  s AND %s AND rasp_id = %s AND arduino_id = %s"
```

```
    cursor.execute(query, (fecha_inicio, fecha_fin, raspi_id, ardu_id))
    data = cursor.fetchall()
    cursor.close()

    # Genera el grafico con Matplotlib
    fig, ax = plt.subplots()
    fechas = [item[0] for item in data]
    temp = [item[1] for item in data]
    hum = [item[2] for item in data]

    ax.set_xlabel('Fecha')
    ax.set_ylabel('Temperatura', color='tab:blue')
    ax.plot(fechas, temp, color='tab:blue')
    ax.tick_params(axis='y', labelcolor='tab:blue')

    ax2 = ax.twinx()
    ax2.set_ylabel('Humedad', color='tab:red')
    ax2.plot(fechas, hum, color='tab:red')
    ax2.tick_params(axis='y', labelcolor='tab:red')

    ax.set_title('Grafico de Datos Temperatura y Humedad')
    img_buffer = BytesIO()
    plt.savefig(img_buffer, format='png')
    img_buffer.seek(0)
    img_str = base64.b64encode(img_buffer.read()).decode('utf-8')
    plt.close()

    # Renderiza la plantilla con la imagen del grafico
    return render_template('grafico.html', img_str=img_str)

if __name__ == '__main__':
    app.run(debug=True)
```

1.5 Graficos

Por último, en el código HTML anterior, existe el botón de generar gráfico el cual redirige al siguiente código que se encarga de mostrar la imagen con los datos.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Grafico</title>
</head>
<body>
    <h1>Grafico de Datos</h1>
    <img src="{{ img_str }}" alt="Grafico">
</body>
</html>
```

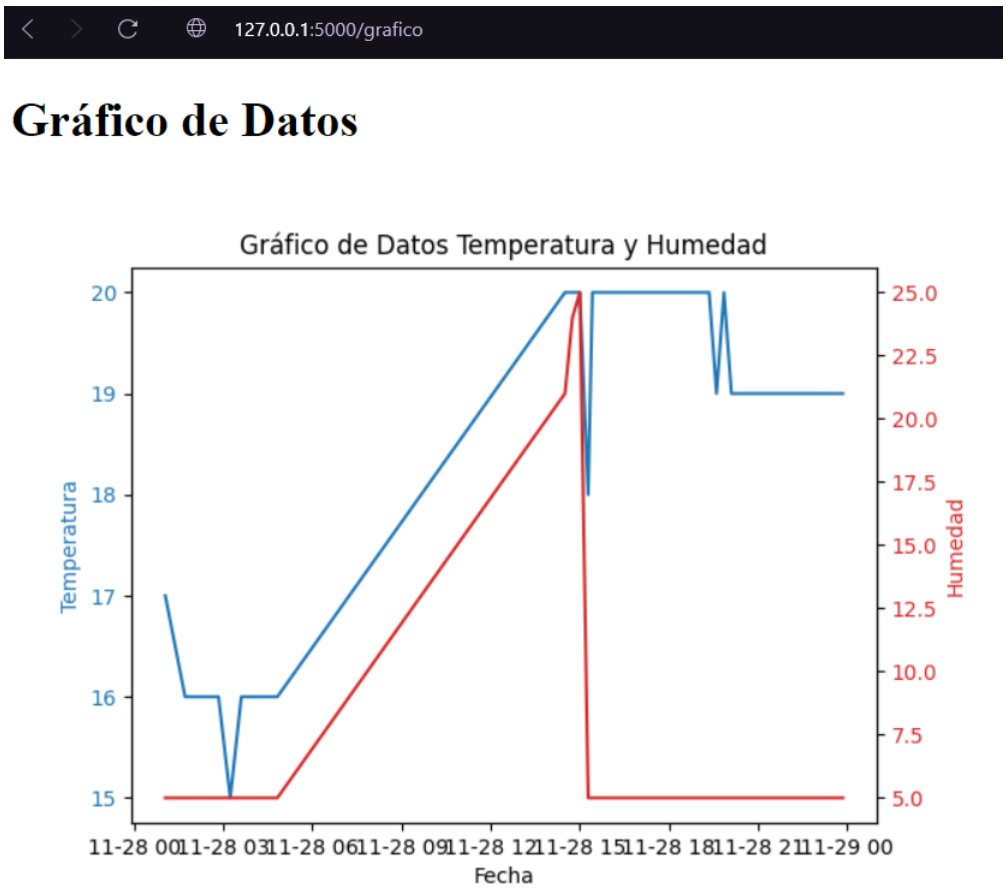


Figura 1.4: Resultados Grafica Temperatura

1.6 Diagrama y Conclusiones

Es de importancia saber el modo de transmisión de los datos y definir que es lo que se desea enviar. Para este proyecto, los sensores poseen los datos que recibe el Arduino, que a su vez posee una ID lo que permite más de un Arduino conectado a una Raspberry. Lo mismo aplica para la Raspberry donde la RaspID puede ser útil para una búsqueda de datos más personalizada.

A continuación, se muestra el diagrama del funcionamiento del sistema:

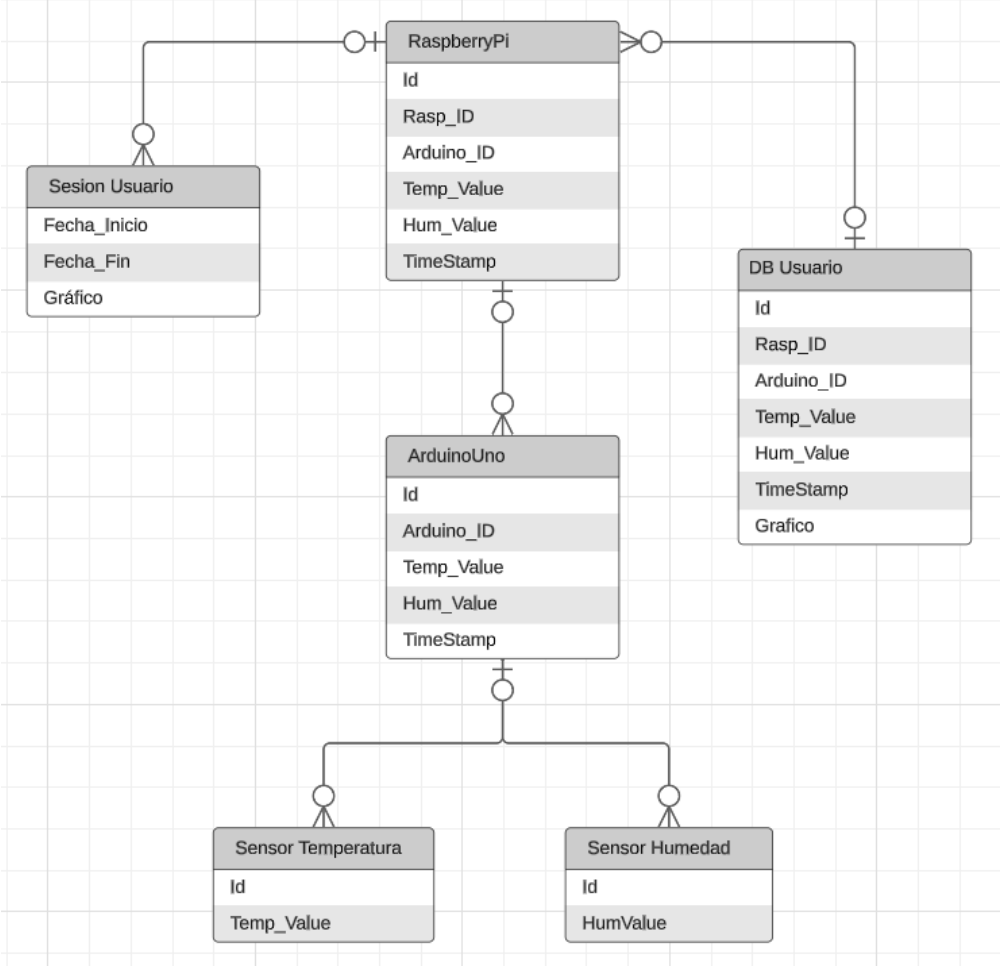


Figura 1.5: Diagrama Sistema Embebido

La implementación de un proyecto de estación meteorológica utilizando Raspberry Pi junto con una base de datos PostgreSQL ofrece una solución robusta y personalizable para la recopilación y gestión de datos meteorológicos. La combinación de la versatilidad de Raspberry Pi y la confiabilidad de PostgreSQL proporciona una plataforma efectiva para monitorear y analizar las condiciones meteorológicas locales.

Asimismo, el diseño web permite al usuario buscar y analizar datos a gusto, generando gráficos gracias a los comandos SQL ofreciendo una experiencia sencilla de navegar.

Bibliografía

[1] (2023) Python web. [En línea]. Disponible en: https://realpython.com/python3-object-oriented-programming/

[2] (2023) Python. [En línea]. Disponible en: https://luca-d3.com/es/data-speaks/diccionario-tecnologico/python-lenguaje

[3] (2021) Psycopg2. [En línea]. Disponible en: https://pypi.org/project/psycopg2/