MONITOREO DE TEMPERATURA CON RASPBERRY PI PICO W Y THINGSPEAK

Internet of Things

JORGE HUMBERTO SOSA GARCÍA
15221311
Ingeniería Mecatrónica 6to Semestre

Mtro. Freddy Antonio Ix Andrade
28/02/2025

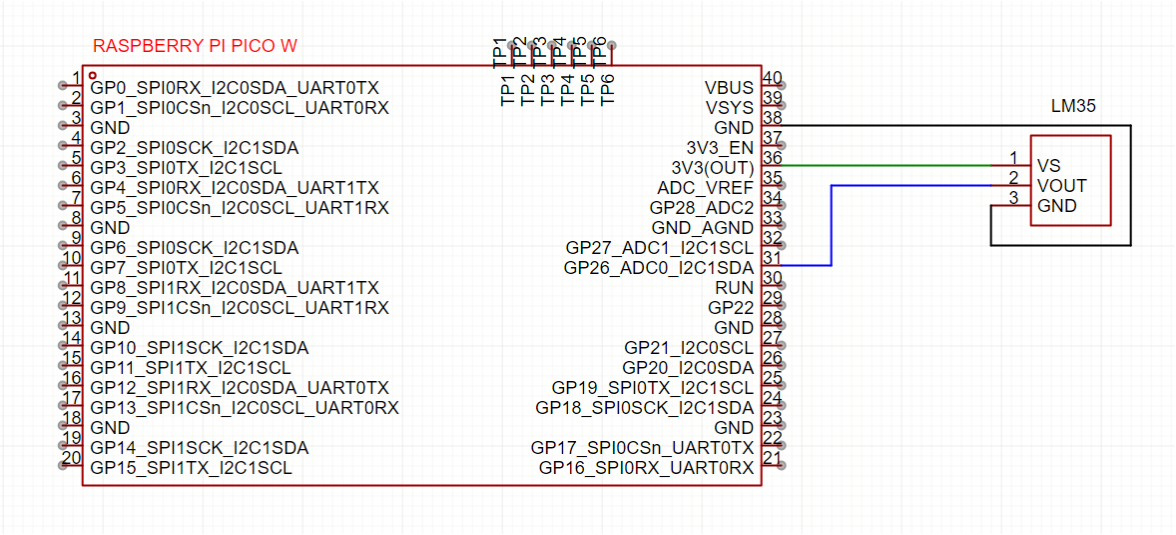
INTRODUCCIÓN

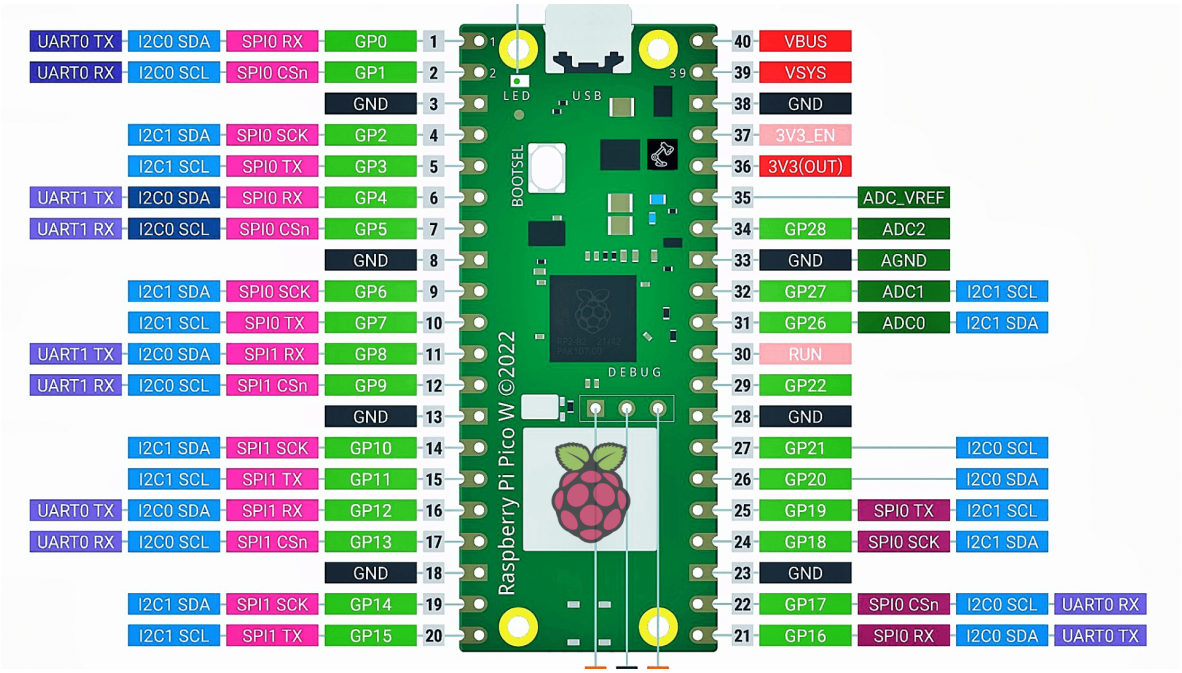
En este proyecto se utilizó una Raspberry Pi Pico W para conectarse a Internet mediante un código en MicroPython. Se conectó un sensor LM35 para la medición de temperatura, y los datos obtenidos fueron enviados a ThingSpeak para su recolección y almacenamiento. Además, se empleó MATLAB Analysis para calcular un promedio de los valores registrados. Finalmente, toda la información y el código del proyecto fueron documentados y compartidos a través de GitHub.

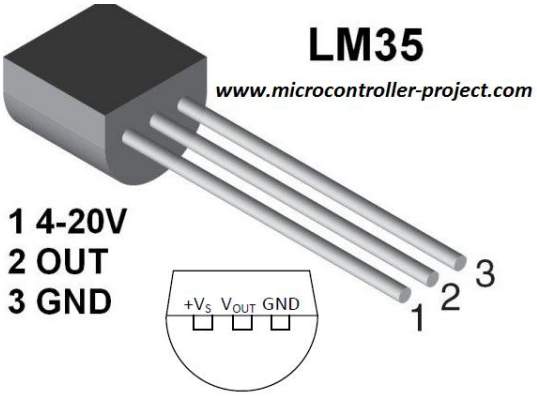
Se detallará la configuración del hardware, la programación en MicroPython, la conexión con la nube y el análisis de los datos obtenidos en ThingSpeak.

DESARROLLO

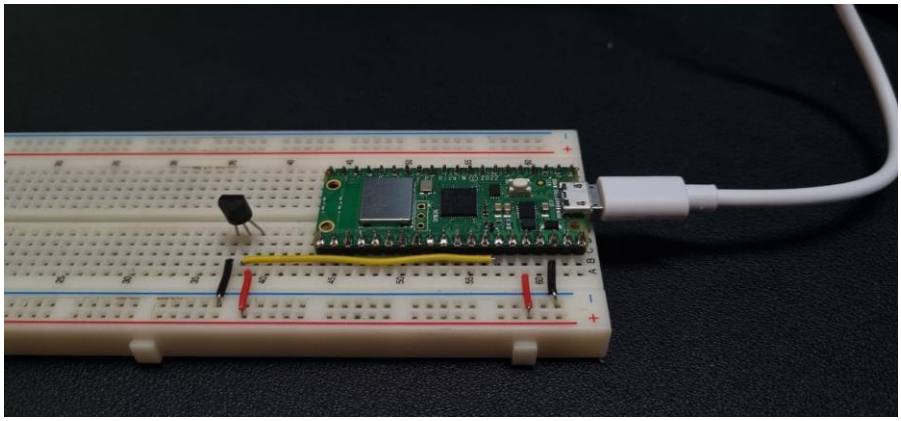
A) CONFIGURACIÓN DEL HARDWARE







El sensor LM35 es un sensor de temperatura de precisión cuya salida es una señal analógica linealmente proporcional a la temperatura en grados Celsius. La salida del sensor varía en 10 mV por cada grado centígrado. En este proyecto, se conecta al pin GP28 de la Raspberry Pi Pico W, el cual está configurado como una entrada analógica.



B) PROGRAMACIÓN EN MICROPYTHON

```
import network  # Librería para manejar la conexión Wi-Fi
import time  # Librería para manejar tiempos y pausas
import urequests  # Librería para hacer solicitudes HTTP a ThingSpeak
import machine  # Librería para manejar el hardware de la Raspberry Pi Pico W

# ◇ Configuración de Wi-Fi
SSID = "Casa Pinos 2G"  # Nombre de la red Wi-Fi (SSID)
PASSWORD = "jsruz4040"  # Contraseña de la red Wi-Fi

# ◇ Configuración de ThingSpeak
THINGSPEAK_API_KEY = "E1NSFB1YJ5X7LT4X"  # API Key para escribir en ThingSpeak
THINGSPEAK_URL = "https://api.thingspeak.com/update?api_key=E1NSFB1YJ5X7LT4X&field1=0"  # URL base para enviar datos a ThingSpeak

# ◇ Configuración del sensor LM35
adc = machine.ADC(28)  # Configura el ADC en el pin GP28 (ADC2) para leer el sensor LM35

# ◇ Función para conectar a Wi-Fi
def connect_wifi():
    wlan = network.WLAN(network.STA_IF)  # Configura la Wi-Fi en modo estación
    wlan.active(True)  # Activa la Wi-Fi
    if not wlan.isconnected():  # Si no está conectada
        print("Conectando a Wi-Fi...")
        wlan.connect(SSID, PASSWORD)  # Intenta conectar con las credenciales dadas
        tiempo_inicio = time.time()  # Guarda el tiempo de inicio de la conexión
        while not wlan.isconnected():  # Espera hasta que se conecte
            if time.time() - tiempo_inicio > 10:  # Si tarda más de 10 segundos
                print("❌ No se pudo conectar a Wi-Fi")  # Muestra error
                return False  # Retorna falso si no se conecta
            time.sleep(1)  # Espera 1 segundo antes de volver a intentar
    print("✅ Conectado a Wi-Fi:", wlan.ifconfig())  # Muestra la IP asignada
    return True  # Retorna verdadero si la conexión fue exitosa

# ◇ Función para leer el LM35 y enviar datos a ThingSpeak
def enviar_a_thingspeak():
    while True:  # Bucle infinito para seguir enviando datos
        lectura = adc.read_u16()  # Lee el valor del ADC (0-65535)
        voltaje = lectura * (3.3 / 65535)  # Convierte el valor leído a voltaje (0-3.3V)
        temperatura = voltaje * 100  # Convierte el voltaje en temperatura (°C)

        print(f"🌡️  Temperatura: {temperatura:.2f} °C")  # Muestra la temperatura en la consola

        # Construye la URL con la temperatura medida
        url = f"{THINGSPEAK_URL}?api_key={THINGSPEAK_API_KEY}&field1={temperatura}"

        # Intenta enviar los datos a ThingSpeak
        try:
            respuesta = urequests.get(url)  # Hace la solicitud HTTP GET a ThingSpeak
            print(f"✅ Enviado a ThingSpeak: {respuesta.text}")  # Muestra la respuesta del servidor
            respuesta.close()  # Cierra la conexión HTTP
        except Exception as e:  # Captura cualquier error en la solicitud
            print(f"❌ Error al enviar datos: {e}")  # Muestra el error

        time.sleep(60)  # Espera 60 segundos antes de enviar el siguiente dato

# ◇ Ejecutar el código principal
if connect_wifi():  # Si se conecta a Wi-Fi correctamente
    enviar_a_thingspeak()  # Comienza a leer la temperatura y enviarla a ThingSpeak
else:
    print("❌ No se pudo conectar a Wi-Fi")  # Muestra error si no hay conexión
```

- Conexión a Wi-Fi: Configuración de la Raspberry Pi Pico W para conectarse a una red Wi-Fi utilizando credenciales establecidas en el código.
- Lectura del sensor LM35 usando el ADC: Conversión de la señal analógica proporcionada por el sensor LM35 en valores digitales utilizando el convertidor ADC de la Raspberry Pi Pico W.
- Conversión de la señal analógica a temperatura en °C: Cálculo de la temperatura a partir de la señal analógica basada en la relación de voltaje del LM35.
- Envío de datos a ThingSpeak: Comunicación con la plataforma ThingSpeak a través de solicitudes HTTP para registrar los valores de temperatura.

C) VISUALIZACIÓN EN THINGSPEAK

Evaluación Primer Parcial

Channel ID: **2854534**
Author: mwa0000036854159
Access: Public

Desarrollar un sistema de monitoreo de temperatura basado en Internet de las Cosas (IoT) utilizando una Raspberry Pi Pico W y un sensor LM35, con el propósito de capturar, procesar y visualizar datos en la nube mediante ThingSpeak. Además, se implementará
lm35, temperatura, temperature, unimodelo, modelo, universidad modelo, iot

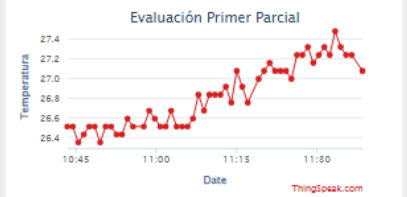
Private View | Public View | Channel Settings | Sharing | API Keys | Data Import / Export

Add Visualizations | Add Widgets | Export recent data | MATLAB Analysis | MATLAB Visualization

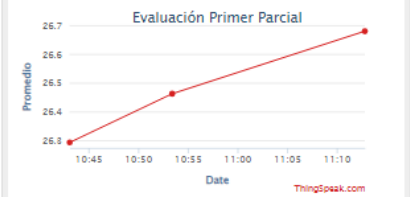
Channel Stats

Created: 3 days ago
Last entry: about a minute ago
Entries: 16500

Field 1 Chart



Field 2 Chart



Field 3 Chart

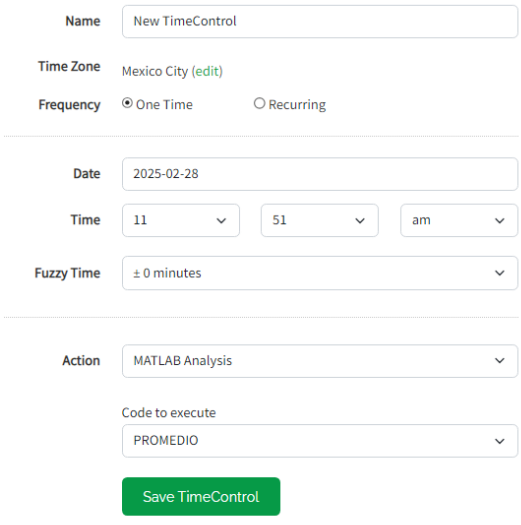
CÓDIGO DE PROMEDIO

PROMEDIO

MATLAB Code

```
% CONFIGURACIÓN
readChannelID = 2854534; % Reemplaza con el ID de tu canal
writeChannelID = 2854534; % ID del canal donde escribirás el promedio
readAPIKey = 'CBP2QHLVW9L44LS5'; % API Key de lectura (déjalo vacío si el canal es público)
writeAPIKey = 'E1NSFB1YJ5X7LT4X'; % API Key de escritura (verifica que sea correcta)
fieldID = 1; % Campo donde están los datos originales

% INTENTAR LEER LOS ÚLTIMOS 10 DATOS VARIAS VECES
maxAttempts = 3;
attempt = 0;
data = [];

while attempt < maxAttempts && isempty(data)
    attempt = attempt + 1;
    data = thingSpeakRead(readChannelID, 'Fields', fieldID, 'NumPoints', 10, 'ReadKey', readAPIKey);
    pause(2); % Espera 2 segundos entre intentos
end

% VERIFICAR SI HAY SUFICIENTES DATOS
if isempty(data) || any(isnan(data))
    disp("⚠️ No se pudieron obtener suficientes datos después de " + attempt + " intentos.");
    return;
end

% CALCULAR EL PROMEDIO
avgValue = mean(data);

% VERIFICAR QUE EL PROMEDIO SEA VÁLIDO
if isnan(avgValue) || ismissing(avgValue)
    disp("⚠️ El promedio no se pudo calcular correctamente.");
    return;
end

% MOSTRAR EL PROMEDIO CALCULADO
disp("???? Promedio calculado: " + avgValue);

% VERIFICAR QUE EL CANAL DE ESCRITURA Y LA CLAVE API SEAN CORRECTOS
if isempty(writeAPIKey) || writeChannelID == 0
    disp("❌ Error: El canal de escritura o la clave API no están configurados correctamente.");
    return;
end

% ESCRIBIR EL PROMEDIO EN THINGSPEAK
thingSpeakWrite(writeChannelID, 'Fields', [2], 'Values', avgValue, 'WriteKey', writeAPIKey);
disp("✅ Promedio enviado a ThingSpeak.");

```

TIMER DEL PROMEDIO



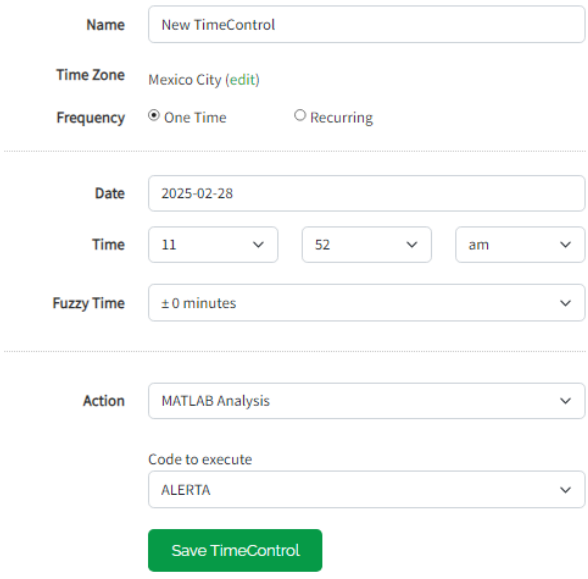
CÓDIGO DE LAS ALERTAS

ALERTA

MATLAB Code

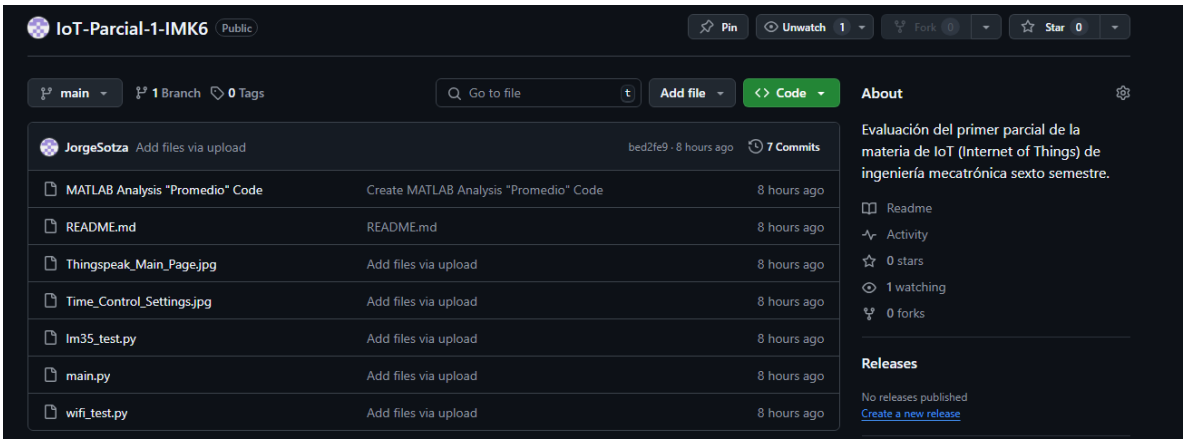
```
% ???? CONFIGURACIÓN INICIAL

readChannelID = 2854534; % ???? ID del canal en ThingSpeak donde se almacenan los datos de temperatura
writeChannelID = 2854534; % ???? ID del canal donde se escribirá la alerta
readAPIKey = 'CBP2QHLVW9L44LS5'; % ???? API Key para leer datos (déjalo vacío si el canal es público)
writeAPIKey = 'E1NSFB1YJ5X7LT4X'; % ???? API Key para escribir datos en ThingSpeak
fieldID = 1; % ???? Campo donde se almacena la temperatura en ThingSpeak
alertFieldID = 3; % ???? Campo donde se guardará la alerta en ThingSpeak

% ???? LEER EL ÚLTIMO VALOR DE TEMPERATURA DESDE THINGSPEAK

temperature = thingSpeakRead(readChannelID, 'Fields', fieldID, 'NumPoints', 1, 'ReadKey', readAPIKey);

% ???? VERIFICAR SI LA TEMPERATURA ES VÁLIDA
if isempty(temperature) || isnan(temperature)
    disp("⚠️ No hay datos recientes de temperatura.");
    return; % ???? Si no hay datos, detiene la ejecución
end

% ???? Muestra la temperatura obtenida en la consola
disp("???? Última temperatura registrada: " + temperature + "°C");

% ???? COMPROBAR SI LA TEMPERATURA SUPERA LOS 35°C
if temperature > 35
    alertMessage = 1; % ???? Activa la alerta con un valor de 1 en ThingSpeak
    disp("???? Alerta: Temperatura alta detectada!");
else
    alertMessage = 0; % ???? No hay alerta, se mantiene en 0
    disp("✅ Temperatura dentro del rango normal.");
end

% ???? ESCRIBIR LA ALERTA EN THINGSPEAK
thingSpeakWrite(writeChannelID, 'Fields', alertFieldID, 'Values', alertMessage, 'WriteKey', writeAPIKey);
disp("✅ Alerta enviada a ThingSpeak.");

```

TIMER DE LAS ALERTAS

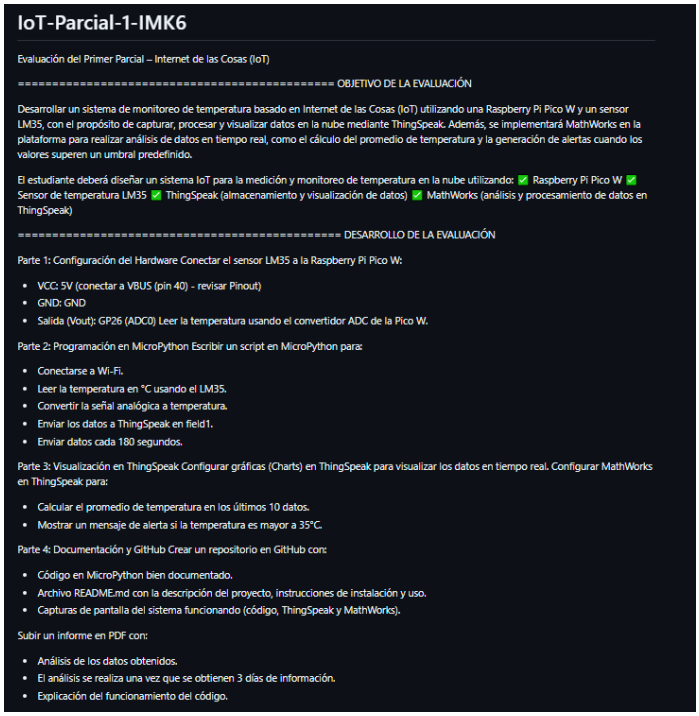


D) USO DE GITHUB

El código y toda la documentación del proyecto fueron gestionados a través de GitHub, una plataforma ampliamente utilizada para el control de versiones y la colaboración en proyectos de software.



Se creó un repositorio en GitHub donde se almacenaron los archivos principales del proyecto, incluyendo el código en MicroPython utilizado en la Raspberry Pi Pico W para la conexión con ThingSpeak. Además, se incluyó un archivo README.md, el cual proporciona una descripción detallada del proyecto.



IoT-Parcial-1-IMK6

Evaluación del Primer Parcial – Internet de las Cosas (IoT)

========== OBJETIVO DE LA EVALUACIÓN

Desarrollar un sistema de monitoreo de temperatura basado en Internet de las Cosas (IoT) utilizando una Raspberry Pi Pico W y un sensor LM35, con el propósito de capturar, procesar y visualizar datos en la nube mediante ThingSpeak. Además, se implementará MathWorks en la plataforma para realizar análisis de datos en tiempo real, como el cálculo del promedio de temperatura y la generación de alertas cuando los valores superen un umbral predefinido.

El estudiante deberá diseñar un sistema IoT para la medición y monitoreo de temperatura en la nube utilizando: ✅ Raspberry Pi Pico W ✅ Sensor de temperatura LM35 ✅ ThingSpeak (almacenamiento y visualización de datos) ✅ MathWorks (análisis y procesamiento de datos en ThingSpeak)

========== DESARROLLO DE LA EVALUACIÓN

Parte 1: Configuración del Hardware Conectar el sensor LM35 a la Raspberry Pi Pico W:

- VCC: 5V (conectar a VBUS (pin 40) - revisar Pinout)
- GND: GND
- Salida (Vout): GP26 (ADC0) Leer la temperatura usando el convertidor ADC de la Pico W.

Parte 2: Programación en MicroPython Escribir un script en MicroPython para:

- Conectarse a Wi-Fi.
- Leer la temperatura en °C usando el LM35.
- Convertir la señal analógica a temperatura.
- Enviar los datos a ThingSpeak en field1.
- Enviar datos cada 180 segundos.

Parte 3: Visualización en ThingSpeak Configurar gráficas (Charts) en ThingSpeak para visualizar los datos en tiempo real. Configurar MathWorks en ThingSpeak para:

- Calcular el promedio de temperatura en los últimos 10 datos.
- Mostrar un mensaje de alerta si la temperatura es mayor a 35°C.

Parte 4: Documentación y GitHub Crear un repositorio en GitHub con:

- Código en MicroPython bien documentado.
- Archivo README.md con la descripción del proyecto, instrucciones de instalación y uso.
- Capturas de pantalla del sistema funcionando (código, ThingSpeak y MathWorks).

Subir un informe en PDF con:

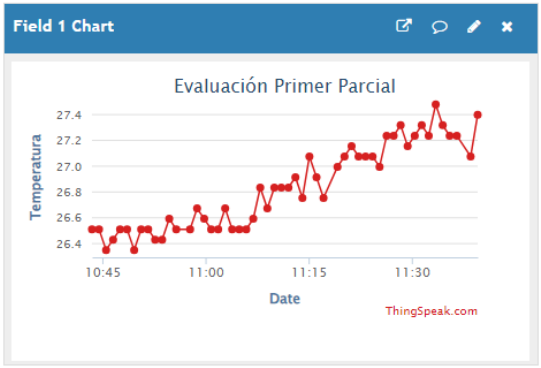
- Análisis de los datos obtenidos.
- El análisis se realiza una vez que se obtienen 3 días de información.
- Explicación del funcionamiento del código.

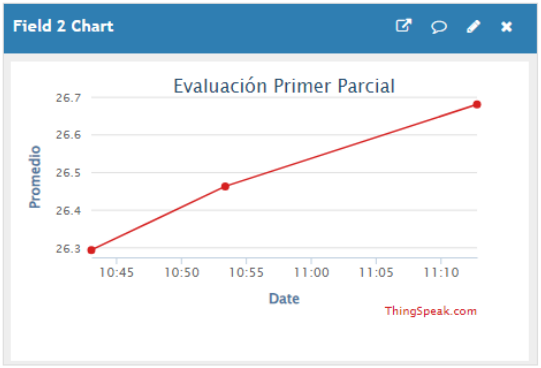
Durante el desarrollo del proyecto, se realizaron varios commits, cada uno documentando cambios específicos en el código o la configuración del sistema. Esto permitió un mejor seguimiento del progreso y facilitó la identificación de errores en caso de ser necesario.

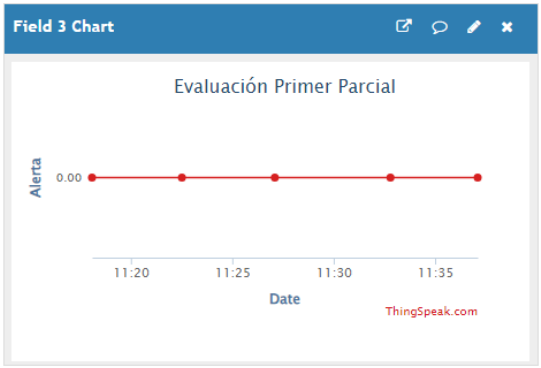
Asimismo, se subieron capturas de pantalla que muestran el sistema en funcionamiento, incluyendo la correcta recolección de datos en ThingSpeak y la ejecución de los análisis en MATLAB Analysis.

El uso de GitHub no solo facilitó la organización del código, sino que también permitió la posibilidad de compartir el proyecto con otros usuarios y colaboradores de manera sencilla.

ANÁLISIS DE LOS RESULTADOS







A lo largo del desarrollo de este proyecto, se adquirieron diversos conocimientos y habilidades clave. Uno de los aprendizajes más significativos fue la programación en MicroPython, un lenguaje que, aunque basado en Python, nunca había sido utilizado con tanta profundidad. Además, se logró integrar un sensor LM35 con una Raspberry Pi Pico W, algo completamente nuevo dentro de la experiencia previa del proyecto.

Otro aspecto importante fue la conexión de la Raspberry Pi Pico W a ThingSpeak para el envío y almacenamiento de datos en la nube. Este proceso permitió entender cómo funcionan las plataformas de IoT para la recolección y análisis de datos en tiempo real. En particular, se exploraron las capacidades de MATLAB Analysis dentro de ThingSpeak, lo que permitió la implementación de cálculos automáticos, como el promedio de temperatura, y la generación de alertas cuando se alcanzaban ciertos umbrales.

Sin embargo, el proyecto no estuvo exento de desafíos. Una de las principales limitaciones encontradas fue la restricción de ThingSpeak, que solo permite el envío de datos cada 15 segundos, lo que provocó varios problemas de sincronización y saturación del sistema. Para solucionar esto, se estableció un intervalo de 3 minutos entre cada envío de datos desde la Raspberry, lo que permitió un funcionamiento más estable. Aun así, en algunas ocasiones, el sistema presentó fallos intermitentes en la generación de alertas y cálculos de promedios.

En cuanto a las posibles aplicaciones de este proyecto, el análisis y monitoreo remoto de datos es una herramienta extremadamente útil dentro de la industria. Este tipo de implementación se puede utilizar en el seguimiento de máquinas, temperatura, humedad, presión, y otros parámetros críticos en entornos industriales. Poder acceder a información en tiempo real desde cualquier ubicación es una ventaja que facilita la toma de decisiones y mejora la eficiencia en diversos sectores.

En conclusión, este proyecto representó una oportunidad de aprendizaje invaluable, no solo en términos de programación y electrónica, sino también en la integración de tecnologías IoT para la recopilación y análisis de datos en la nube. La experiencia obtenida abre la puerta a futuras implementaciones más complejas y robustas en entornos reales.

ANEXOS

https://github.com/JorgeSotza/IoT-Parcial-1-IMK6