UNIVERSIDAD MODELO

“Escuela de Ingeniería”

"Monitoreo de Temperatura con Raspberry Pi Pico W y ThingSpeak"

ASIGNATURA

Internet de las Cosas

PROFESOR

Freddy Antonio Ix Andrade

ESTUDIANTE

Luis Manuel Pacho Ayora [*15222403*]

ENTREGA

24/02/2025

Introducción

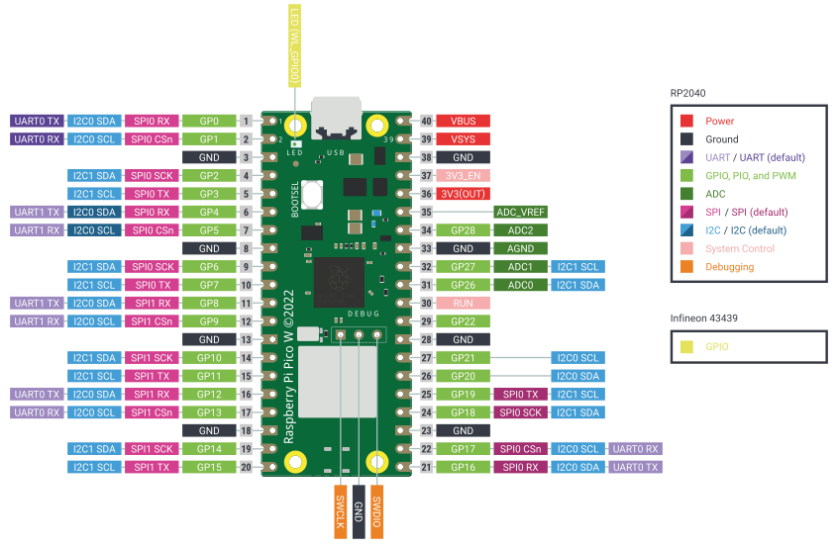
En la actualidad, el Internet de las Cosas (IoT) juega un papel crucial en la monitorización y control de diversos parámetros en múltiples sectores, incluyendo el industrial, doméstico y de investigación. Este proyecto tiene como objetivo desarrollar un sistema de monitoreo de temperatura basado en IoT, utilizando la Raspberry Pi Pico W y el sensor LM35 para capturar datos y enviarlos a la plataforma ThingSpeak para su almacenamiento y visualización.

Para lograrlo, se emplean las siguientes herramientas:

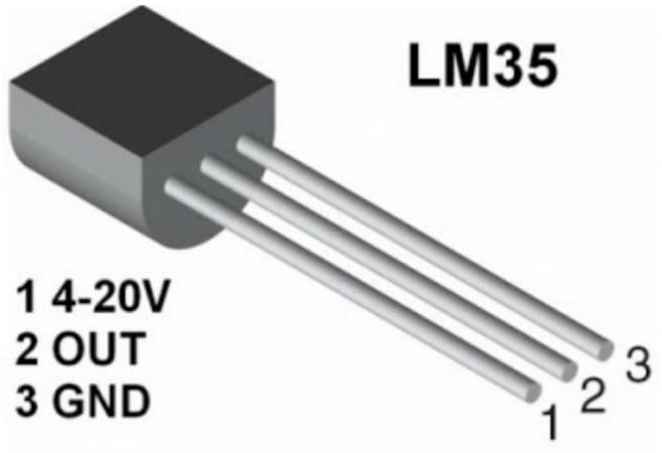
- **Raspberry Pi Pico W**: una microcontroladora con conectividad Wi-Fi.
- **Sensor LM35**: que permite medir la temperatura con una salida analógica proporcional a la temperatura en grados Celsius.
- **ThingSpeak**: una plataforma en la nube utilizada para almacenar y visualizar los datos en tiempo real.
- **MathWorks**: para analizar los datos recopilados y generar alertas en caso de valores fuera de rango.
- **GitHub**: como sistema de control de versiones para documentar el código y compartir el proyecto.

Desarrollo del Proyecto

Diagrama de conexión del **LM35 con la Raspberry Pi Pico W**.



Este es el diagrama de la Raspberry Pi Pico W donde nos dice en que pines debemos poner el sensor LM35 en este caso este es el datashet del sensor para tenerlo mas claro:



Pasos para conectar el sensor LM35 a la Raspberry.

1. Agarrar en PIN 40 (**Raspberry**) y conectarlo al PIN 1(VCC) del LM35.
2. Conectar el PIN 31 (GP 26) (**Raspberry**) al PIN 2 (Out) del LM35.
3. Conectar el PIN 38 (GND) (**Raspberry**) al PIN 3 (GND) del LM35.

Explicación del **funcionamiento del LM35** y su salida analógica.

El **sensor LM35** es un sensor de temperatura de precisión cuya salida es una señal analógica proporcional a la temperatura en grados Celsius. Su principal ventaja es que proporciona una lectura lineal con una escala de **10 mV por grado Celsius** (10 mV/°C), lo que facilita su conversión a temperatura sin necesidad de calibración adicional.

Funcionamiento del LM35

El LM35 tiene tres pines:

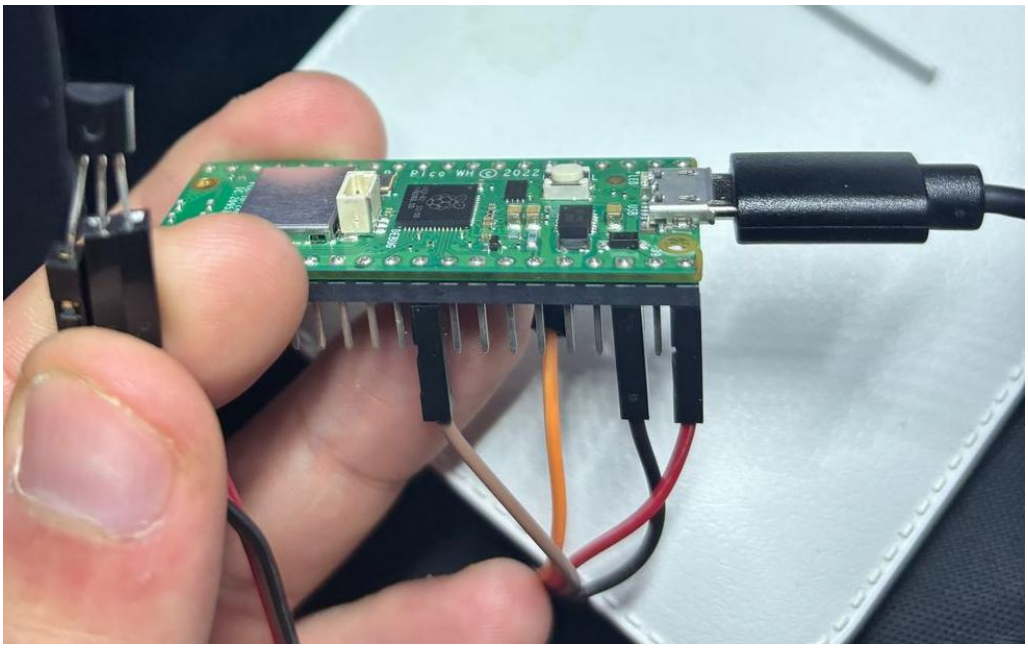
- **VCC**: Alimentación (4V a 30V, en este caso se usa 5V de la Raspberry Pi Pico W).
- **GND**: Tierra.
- **Vout**: Salida analógica que varía en función de la temperatura medida.

La salida del LM35 se puede leer con un **convertidor analógico-digital (ADC)**, como el incorporado en la Raspberry Pi Pico W.

Donde:

- VoutV_{out}Vout es el voltaje en voltios medido en la salida del sensor.
- El factor de conversión 100100100 proviene de la escala de 10 mV/°C.

Circuito Ensamblado



Explicación del código fuente

Para establecer la conexión a Internet, se utiliza la librería **network**.

1. Se activa la interfaz Wi-Fi con network.WLAN(network.STA_IF).
2. Si no hay conexión, intenta conectarse con wlan.connect(SSID, PASSWORD).
3. Se espera un máximo de 15 segundos para conectarse, verificando el estado con wlan.isconnected().
4. Si la conexión es exitosa, imprime la dirección IP obtenida; de lo contrario, muestra un mensaje de error.

Lectura del sensor LM35 usando el ADC

El sensor LM35 genera una señal analógica proporcional a la temperatura.

1. Se configura el **ADC (Convertidor Analógico-Digital)** en el pin **GP26** con machine.ADC(26).
2. Se lee la señal analógica con sensor.read_u16(), que devuelve un valor de 16 bits (0-65535).
3. Se convierte el valor en voltios usando CONVERSION_FACTOR = 3.3 / 65535.

```
# Configuración del sensor LM35
sensor = machine.ADC(26)  # LM35 conectado a GP26 (ADC0)
CONVERSION_FACTOR = 3.3 / 65535  # Conversión ADC a Voltios
```

Conversión de la señal analógica a temperatura en °C

1. Se multiplica la lectura del ADC por el factor de conversión para obtener el voltaje.
2. Se convierte el voltaje en temperatura con la fórmula:

$$T(℃) = \frac{V\ x\ 100}{10}$$

Dado que el **LM35 tiene una sensibilidad de 10mV/°C**, se divide el resultado entre 10 para obtener la temperatura en grados Celsius.

Envío de datos a ThingSpeak

1. Se obtiene la temperatura llamando a leer_temperatura().
2. Se genera la URL de la API de ThingSpeak con el valor de temperatura en field1.
3. Se envía la solicitud HTTP con urequests.get(url).
4. Si la solicitud es exitosa, imprime la respuesta del servidor; si falla, se captura la excepción y se muestra un mensaje de error.

```
def enviar_datos():
    temperatura = leer_temperatura()
    print(f"Temperatura: {temperatura}°C")

    url = f"{THINGSPEAK_URL}&field1={temperatura}"
    try:
        response = urequests.get(url)
        print("Respuesta HTTP:", response.text)
        response.close()
    except Exception as e:
        print("Error enviando datos:", e)
```

Ejecución del programa principal

1. Se establece la conexión Wi-Fi con conectar_wifi().
2. Se inicia un bucle infinito donde:
    - Se verifica que la conexión a Wi-Fi sigue activa.
    - Se envía la temperatura a **ThingSpeak**.
    - Se espera **180 segundos** antes de la siguiente lectura.

```
# Ejecutar
wlan = conectar_wifi()

while True:
    if not wlan.isconnected():  # Reintentar conexión si se pierde
        wlan = conectar_wifi()

    enviar_datos()
    utime.sleep(180)  # Esperar 180 segundos
```

GitHub

README: https://github.com/xLuisPacho/IoT-Temperature-Monitor/blob/main/README.md

🌡️ Sistema de Monitoreo de Temperatura IoT con Raspberry Pi Pico W - WH

📖 Descripción del Proyecto

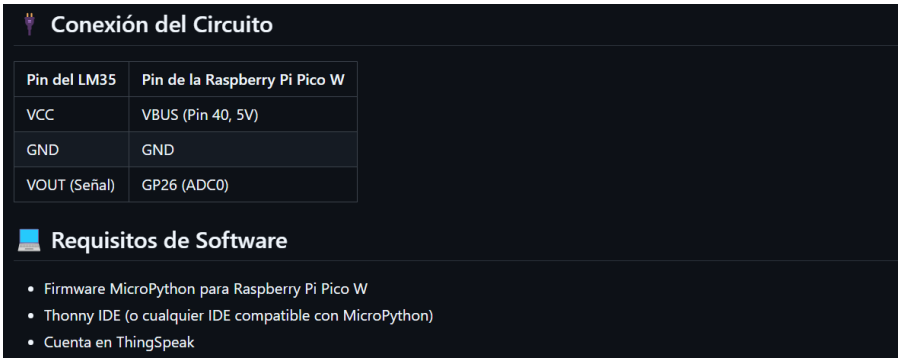
Este proyecto demuestra un sistema de monitoreo de temperatura basado en IoT utilizando una Raspberry Pi Pico W y un sensor de temperatura LM35. El sistema captura datos de temperatura en tiempo real, los sube a la nube mediante ThingSpeak y realiza análisis de datos con MathWorks. Se generan alertas si la temperatura supera un umbral predefinido.

🎯 Objetivos del Proyecto

1. **Monitoreo de Temperatura:** Leer continuamente los datos de temperatura del sensor LM35.
2. **Carga de Datos en la Nube:** Enviar lecturas de temperatura a ThingSpeak cada 180 segundos.
3. **Análisis de Datos:** Calcular el promedio de las últimas 10 lecturas utilizando MathWorks.
4. **Alertas:** Generar una alerta si la temperatura supera los 35°C.

🛠️ Requisitos de Hardware

- Raspberry Pi Pico W
- Sensor de Temperatura LM35
- Protoboard y cables de conexión
- Cable USB para alimentación y programación

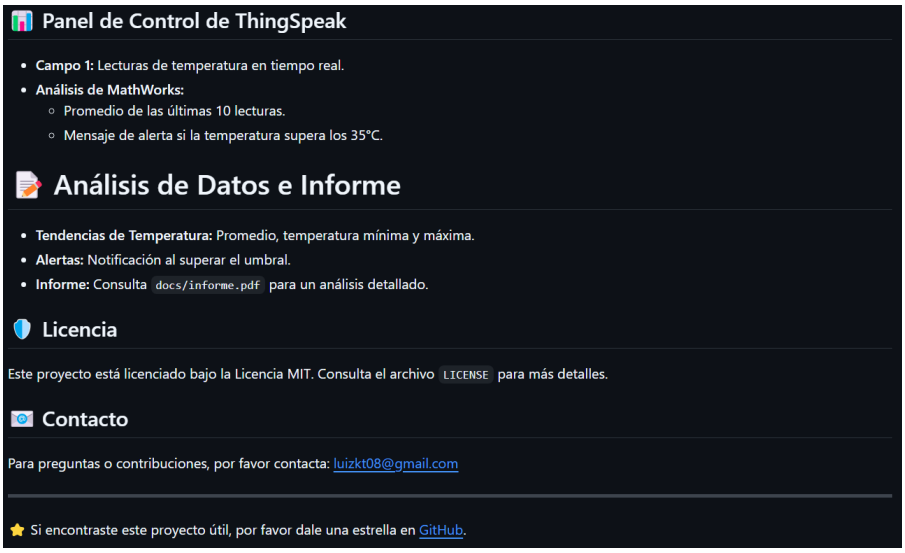


🔌 Conexión del Circuito

Pin del LM35	Pin de la Raspberry Pi Pico W
VCC	VBUS (Pin 40, 5V)
GND	GND
VOUT (Señal)	GP26 (ADC0)

💻 Requisitos de Software

- Firmware MicroPython para Raspberry Pi Pico W
- Thonny IDE (o cualquier IDE compatible con MicroPython)
- Cuenta en ThingSpeak



📊 Panel de Control de ThingSpeak

- **Campo 1:** Lecturas de temperatura en tiempo real.
- **Análisis de MathWorks:**
  - Promedio de las últimas 10 lecturas.
  - Mensaje de alerta si la temperatura supera los 35°C.

📝 Análisis de Datos e Informe

- **Tendencias de Temperatura:** Promedio, temperatura mínima y máxima.
- **Alertas:** Notificación al superar el umbral.
- **Informe:** Consulta `docs/informe.pdf` para un análisis detallado.

🛡️ Licencia

Este proyecto está licenciado bajo la Licencia MIT. Consulta el archivo `LICENSE` para más detalles.

📧 Contacto

Para preguntas o contribuciones, por favor contacta: luizkt08@gmail.com

---

⭐ Si encontraste este proyecto útil, por favor dale una estrella en GitHub.

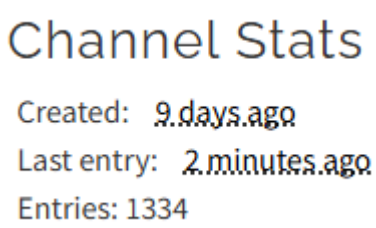
Promedio y alerta de Temperatura

```
% Configuración del canal
channelID = 2844673; % Reemplaza con tu ID de canal
fieldID = 1; % Campo de temperatura
readAPIKey = 'JCO6DVV5XKAK1BU4'; % API Key de lectura (si es necesaria)
writeAPIKey = 'M29KDFI41XEDLX2R'; % API Key de escritura para actualizar ThingSpeak

% Leer los últimos 10 valores del campo de temperatura
data = thingSpeakRead(channelID, 'Fields', fieldID, 'NumPoints', 10, 'ReadKey', readAPIKey);

% Verificar si hay suficientes datos
if numel(data) < 10
    fprintf('No hay suficientes datos para calcular el promedio.\n');
else
    % Calcular el promedio de temperatura
    temp_promedio = mean(data);

    % Mostrar el resultado en MATLAB
    fprintf('Promedio de temperatura: %.2f°C\n', temp_promedio);

    % Escribir el promedio en un nuevo campo en ThingSpeak
    thingSpeakWrite(channelID, 'Fields', 2, 'Values', temp_promedio, 'WriteKey', writeAPIKey);
end

```

Este código en .m nos nos saca el promedio de los últimos 10 datos que sacamos y lo promedia, y también nos manda una alerta si la temperatura pasa de los 35 grados todo esto mostrado en este código.

Análisis de los resultados

Hoy 27 de febrero del 2025 a las 5:30pm se han registrado **1334,** el sensor de temperatura fue conectado el lunes 24 de febrero a las 8:30pm y eso han sido los datos que se han reportado.

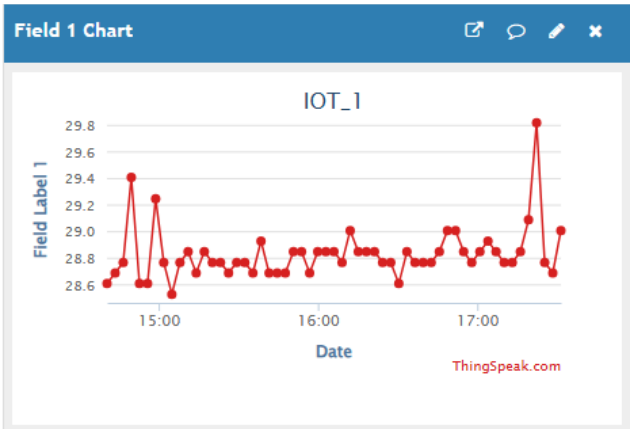


Hay 1334 datos registrado que a captado en sensor conectado.

Preguntas

1. **¿Qué tendencias o patrones se observaron en la temperatura?**

   Lo que notaba en los resultados es que tuve un promedio de temperatura de 28.88 grados y si notaba picos de temperatura.



   Entonces por las mañanas notaba una temperatura promedio, pero en la noche era cuando bajaba mas y puede ser por la temperatura que en la noche hay frio, o prendo el aire acondicionado puede variar por muchas cosas

2. **¿Se activaron alertas en MathWorks? ¿En qué condiciones?**

   El código que tengo esta puesto que si la temperatura llega a 35 grados se active una alerta con el sensor, este es la parte del código:

```
% Configuración del canal
channelID = 2844673; % Reemplaza con tu ID de canal
fieldID = 1; % Campo de temperatura
readAPIKey = 'JCO6DVV5XKAK1BU4'; % API Key de lectura (si es necesaria)
writeAPIKey = 'M29KDFI41XEDLX2R'; % API Key de escritura para actualizar ThingSpeak

% Leer los últimos 10 valores del campo de temperatura
data = thingSpeakRead(channelID, 'Fields', fieldID, 'NumPoints', 10, 'ReadKey', readAPIKey);

% Verificar si hay suficientes datos y no son NaN
if isempty(data) || numel(data) < 10 || any(isnan(data))
    fprintf('No hay suficientes datos válidos para calcular el promedio.\n');
else
    % Calcular el promedio de temperatura
    temp_promedio = mean(data);

    % Mostrar el resultado en MATLAB
    fprintf('Promedio de temperatura: %.2f°C\n', temp_promedio);

    % Escribir el promedio en un nuevo campo en ThingSpeak
    thingSpeakWrite(channelID, 'Fields', 2, 'Values', temp_promedio, 'WriteKey', writeAPIKey);
    pause(16); % Evitar límite de frecuencia de ThingSpeak

    % Verificar si la temperatura promedio supera los 35°C
    if temp_promedio > 35
        fprintf('⚠️ ALERTA: La temperatura promedio supera los 35°C! ⚠️\n');
    end
end

```

3. **¿Se cumplieron los objetivos del proyecto?**

   Por mi parte como estudiante y participe del proyecto yo creo que si cumpli con todos los objetivos del proyecto porque:

   - Logre conectar el LM35 con la Raspberry
   - Logre hacer una conexión con la Raspberry y el Thingspeak
   - Logre hacer el github de mi proyecto

Conclusiones y Reflexión

¿Qué aprendí sobre IoT y ThingSpeak?

- Aprendí a conectar un **sensor de temperatura LM35** a una **Raspberry Pi Pico W** y leer datos utilizando **MicroPython**.
- Descubrí cómo enviar estos datos a **ThingSpeak**, almacenarlos en la nube y visualizarlos en tiempo real mediante gráficos.
- Usé **MATLAB Analysis** en ThingSpeak para calcular el promedio de temperatura y generar alertas cuando se superan ciertos umbrales.
- Comprendí los **límites de frecuencia de ThingSpeak** y la importancia de manejar correctamente las solicitudes para evitar errores.

¿Cómo podría mejorar el proyecto en el futuro?

- **Optimizar el código**, mejorando el manejo de errores y añadiendo una reconexión automática en caso de fallos de conexión WiFi.
- **Implementar notificaciones en tiempo real**, como alertas por **correo electrónico o SMS** cuando la temperatura supere un umbral crítico.
- **Desarrollar una interfaz gráfica más avanzada**, integrando los datos con **Power BI** o **Grafana** para mejorar la visualización.
- **Aplicar Machine Learning** en MATLAB para analizar tendencias y predecir cambios de temperatura antes de que ocurran problemas.

Posibles aplicaciones prácticas en la industria:

- **Monitoreo de temperatura en almacenes y laboratorios**, asegurando condiciones óptimas para productos sensibles.
- **Control de sistemas HVAC (calefacción, ventilación y aire acondicionado)** en edificios inteligentes para mejorar la eficiencia energética.
- **Supervisión de maquinaria industrial**, detectando sobrecalentamientos y previniendo fallos antes de que ocurran.
- **Agricultura inteligente**, optimizando el riego y la producción mediante el monitoreo de temperatura y humedad.