|  |  |
| --- | --- |
| Poli Fabio José  Robra Abel | **UNIVERSIDAD TECNOLÓGICA NACIONAL**  **FACULTAD REGIONAL TUCUMÁN**  Curso MicroPython con el ESP32  **Prof. Maximiliano Simonassi** |

Dataloger generado en base al curso MicroPython con el ESP32

Diagrama, Dibujo de ingeniería

Descripción generada automáticamente

**Título del Proyecto:**

**Monitoreo y Optimización de la Temperatura en Habitaciones Utilizando Sensores DS18B20 y ESP32**

**Resumen:**

Este proyecto tiene como objetivo desarrollar un sistema de monitoreo de temperatura para optimizar el confort térmico en una habitación. Utilizando dos sensores de temperatura DS18B20 conectados a un microcontrolador ESP32, que actúa como un punto de acceso Wi-Fi, se recopilan datos de temperatura en dos puntos específicos: la cámara del cielorraso y debajo del cielorraso. Los datos se transmiten a una notebook conectada al ESP32 para su análisis, permitiendo ajustar el sistema de climatización de manera eficiente.

**Introducción:**

El control de la temperatura en interiores es crucial para el confort y la eficiencia energética. Este proyecto se centra en la implementación de un sistema de monitoreo de temperatura que permite obtener datos precisos sobre las condiciones térmicas en diferentes partes de una habitación. Con esta información, se pueden tomar decisiones informadas para mejorar o disminuir la temperatura interna, optimizando así el uso de sistemas de climatización.

**Metodología:**

1. **Componentes Utilizados:**
   * Dos sensores de temperatura DS18B20.
   * Un microcontrolador ESP32 con capacidad Wi-Fi.
   * Una notebook para la recepción y análisis de datos.
2. **Configuración del Sistema de TX de Datos:**
   * Los sensores DS18B20 se conectan al pin GPIO 4 del ESP32.
   * El ESP32 se configura como un punto de acceso Wi-Fi.

**2. Configuración del Sistema**

En esta sección, se describe cómo se conectan los sensores DS18B20 al ESP32 y cómo se configura el ESP32 como un punto de acceso Wi-Fi para transmitir los datos de temperatura a una notebook.

* **Conexión de los Sensores DS18B20:** Los sensores DS18B20 se conectan al pin GPIO 4 del ESP32. Este pin se utiliza para la comunicación OneWire, que permite la lectura de las temperaturas desde los sensores.
* **Configuración del ESP32 como Punto de Acceso Wi-Fi:** El ESP32 se configura como un punto de acceso Wi-Fi, creando una red a la que la notebook puede conectarse. Esto permite la transmisión de los datos de temperatura desde el ESP32 a la notebook para su análisis.

**Desarrollo del Sistema**

**Importación de Bibliotecas**

import time

import machine

import onewire

import ds18x20

import network

import socket

Estas bibliotecas son esenciales para el funcionamiento del programa:

* time: Para manejar retrasos y tiempos de espera.
* machine: Para interactuar con los pines del ESP32.
* onewire y ds18x20: Para manejar la comunicación con los sensores DS18B20.
* network: Para configurar la red Wi-Fi.
* socket: Para crear un servidor web.

**Configuración de los Pines y Sensores**

ONE\_WIRE\_BUS = 4

dat = machine.Pin(ONE\_WIRE\_BUS)

ds\_sensor = ds18x20.DS18X20(onewire.OneWire(dat))

* ONE\_WIRE\_BUS = 4: Define el pin GPIO 4 para la comunicación OneWire.
* dat = machine.Pin(ONE\_WIRE\_BUS): Configura el pin GPIO 4.
* ds\_sensor = ds18x20.DS18X20(onewire.OneWire(dat)): Inicializa el sensor DS18B20 en el pin definido.

**Configuración del ESP32 como Punto de Acceso Wi-Fi**

ssid = "ESP32-Access-Point"

password = "prov15888"

ap = network.WLAN(network.AP\_IF)

ap.config(essid=ssid, password=password)

ap.active(True)

while not ap.active():

time.sleep(1)

print("Configurando punto de acceso Wi-Fi...")

print("Punto de acceso Wi-Fi configurado")

print("Dirección IP:", ap.ifconfig()[0])

* ssid y password: Definen el nombre y la contraseña del punto de acceso Wi-Fi.
* ap = network.WLAN(network.AP\_IF): Configura el ESP32 como un punto de acceso.
* ap.config(essid=ssid, password=password): Establece el SSID y la contraseña del punto de acceso.
* ap.active(True): Activa el punto de acceso.
* while not ap.active(): Espera hasta que el punto de acceso esté activo.
* print("Dirección IP:", ap.ifconfig()[0]): Imprime la dirección IP del punto de acceso.

**Configuración del Servidor Web**

addr = socket.getaddrinfo('0.0.0.0', 80)[0][-1]

server = socket.socket()

server.bind(addr)

server.listen(1)

print('Servidor escuchando en', addr)

* addr = socket.getaddrinfo('0.0.0.0', 80)[0][-1]: Obtiene la dirección y el puerto para el servidor.
* server = socket.socket(): Crea un socket para el servidor.
* server.bind(addr): Asocia el socket a la dirección y puerto obtenidos.
* server.listen(1): Configura el servidor para escuchar conexiones entrantes.

**Manejo de Conexiones y Lectura de Temperaturas**

while True:

cl, addr = server.accept()

print('Cliente conectado desde', addr)

cl\_file = cl.makefile('rwb', 0)

while True:

line = cl\_file.readline()

if not line or line == b'\r\n':

break

roms = ds\_sensor.scan()

ds\_sensor.convert\_temp()

time.sleep\_ms(750)

temp1 = ds\_sensor.read\_temp(roms[0])

temp2 = ds\_sensor.read\_temp(roms[1])

response = """\

HTTP/1.1 200 OK\r\nContent-Type: text/plain\r\nConnection: close\r\n\r\n

Temp1: {} C\nTemp2: {} C\n""".format(temp1, temp2)

cl.send(response)

cl.close()

print('Cliente desconectado')

* while True: Bucle infinito para manejar conexiones entrantes.
* cl, addr = server.accept(): Acepta una conexión entrante.
* cl\_file = cl.makefile('rwb', 0): Crea un archivo para leer y escribir datos del cliente.
* while True: Bucle para leer datos del cliente.
* if not line or line == b'\r\n': Verifica si la línea está vacía o es un salto de línea.
* roms = ds\_sensor.scan(): Escanea los sensores conectados.
* ds\_sensor.convert\_temp(): Inicia la conversión de temperatura.
* time.sleep\_ms(750): Espera 750 ms para que la conversión termine.
* temp1 = ds\_sensor.read\_temp(roms[0]): Lee la temperatura del primer sensor.
* temp2 = ds\_sensor.read\_temp(roms[1]): Lee la temperatura del segundo sensor.
* response: Crea la respuesta HTTP con las temperaturas.
* cl.send(response): Envía la respuesta al cliente.
* cl.close(): Cierra la conexión con el cliente

Este programa configura un ESP32 como un punto de acceso Wi-Fi y un servidor web. Los sensores DS18B20 miden la temperatura en dos ubicaciones y los datos se envían a una notebook conectada al ESP32. Este sistema permite monitorear y analizar las temperaturas para tomar decisiones informadas sobre el aislamiento térmico y la eficiencia energética

1. **Proceso de Monitoreo**
   * La notebook recibe los datos y los almacena para su posterior análisis.

En esta sección, se describe cómo la notebook se conecta al Wi-Fi del ESP32 y recibe los datos de temperatura, almacenándolos para su posterior análisis.

* **Recepción de Datos en la Notebook:** La notebook se conecta al punto de acceso Wi-Fi del ESP32 y ejecuta un programa en Python que solicita los datos de temperatura cada 30 segundos. Los datos recibidos se almacenan en un archivo Excel para su análisis diario.

**Programa en Python para la Notebook**

**Parte 1: Solicitud de Datos de Temperatura**

Este programa se conecta al servidor web del ESP32, solicita los datos de temperatura cada 30 segundos y los imprime en la consola.

import requests

import time

# URL del servidor ESP32

url = "http://192.168.4.1/" # Asegúrate de usar la IP correcta del ESP32

def get\_temperature():

try:

# Enviar una solicitud GET al servidor

response = requests.get(url)

# Verificar si la solicitud fue exitosa

if response.status\_code == 200:

# Imprimir el texto de la respuesta (datos de temperatura)

print("Datos de temperatura recibidos del ESP32:")

print(response.text)

else:

print(f"Error al recuperar los datos. Código de estado HTTP: {response.status\_code}")

except requests.exceptions.RequestException as e:

print(f"Ocurrió un error: {e}")

# Bucle principal para obtener datos cada 30 segundos

while True:

get\_temperature()

# Esperar 30 segundos antes de la próxima solicitud

time.sleep(30)

**Parte 2: Almacenamiento de Datos en un Archivo Excel**

Este programa crea un archivo Excel y guarda los datos de temperatura recibidos del ESP32.

import os

import pandas as pd

from datetime import datetime

# Ruta de la carpeta y archivo

folder\_path = "D:/DHT11"

file\_path = os.path.join(folder\_path, "ESP32.xlsx")

# Crear la carpeta si no existe

if not os.path.exists(folder\_path):

os.makedirs(folder\_path)

def save\_temperature(temp1, temp2):

# Crear un DataFrame con los datos actuales

now = datetime.now()

date\_time = now.strftime("%Y-%m-%d %H:%M:%S")

df = pd.DataFrame([[date\_time, temp1, temp2]], columns=["Fecha", "Temp1", "Temp2"])

# Guardar los datos en el archivo Excel

if not os.path.exists(file\_path):

df.to\_excel(file\_path, index=False)

else:

existing\_df = pd.read\_excel(file\_path)

updated\_df = pd.concat([existing\_df, df], ignore\_index=True)

updated\_df.to\_excel(file\_path, index=False)

print(f"Datos guardados en {file\_path}")

# Ejemplo de uso

save\_temperature(25.5, 26.0)

**Descripción del Sistema**

**Importación de Bibliotecas**

import time

import machine

import onewire

import ds18x20

import network

import socket

Estas bibliotecas son esenciales para el funcionamiento del programa:

* time: Para manejar retrasos y tiempos de espera.
* machine: Para interactuar con los pines del ESP32.
* onewire y ds18x20: Para manejar la comunicación con los sensores DS18B20.
* network: Para configurar la red Wi-Fi.
* socket: Para crear un servidor web.

**Configuración de los Pines y Sensores**

ONE\_WIRE\_BUS = 4

dat = machine.Pin(ONE\_WIRE\_BUS)

ds\_sensor = ds18x20.DS18X20(onewire.OneWire(dat))

* ONE\_WIRE\_BUS = 4: Define el pin GPIO 4 para la comunicación OneWire.
* dat = machine.Pin(ONE\_WIRE\_BUS): Configura el pin GPIO 4.
* ds\_sensor = ds18x20.DS18X20(onewire.OneWire(dat)): Inicializa el sensor DS18B20 en el pin definido.

**Configuración del ESP32 como Punto de Acceso Wi-Fi**

ssid = "ESP32-Access-Point"

password = "prov15888"

ap = network.WLAN(network.AP\_IF)

ap.config(essid=ssid, password=password)

ap.active(True)

while not ap.active():

time.sleep(1)

print("Configurando punto de acceso Wi-Fi...")

print("Punto de acceso Wi-Fi configurado")

print("Dirección IP:", ap.ifconfig()[0])

* ssid y password: Definen el nombre y la contraseña del punto de acceso Wi-Fi.
* ap = network.WLAN(network.AP\_IF): Configura el ESP32 como un punto de acceso.
* ap.config(essid=ssid, password=password): Establece el SSID y la contraseña del punto de acceso.
* ap.active(True): Activa el punto de acceso.
* while not ap.active(): Espera hasta que el punto de acceso esté activo.
* print("Dirección IP:", ap.ifconfig()[0]): Imprime la dirección IP del punto de acceso.

**Configuración del Servidor Web**

addr = socket.getaddrinfo('0.0.0.0', 80)[0][-1]

server = socket.socket()

server.bind(addr)

server.listen(1)

print('Servidor escuchando en', addr)

* addr = socket.getaddrinfo('0.0.0.0', 80)[0][-1]: Obtiene la dirección y el puerto para el servidor.
* server = socket.socket(): Crea un socket para el servidor.
* server.bind(addr): Asocia el socket a la dirección y puerto obtenidos.
* server.listen(1): Configura el servidor para escuchar conexiones entrantes.

**Manejo de Conexiones y Lectura de Temperaturas**

while True:

cl, addr = server.accept()

print('Cliente conectado desde', addr)

cl\_file = cl.makefile('rwb', 0)

while True:

line = cl\_file.readline()

if not line or line == b'\r\n':

break

roms = ds\_sensor.scan()

ds\_sensor.convert\_temp()

time.sleep\_ms(750)

temp1 = ds\_sensor.read\_temp(roms[0])

temp2 = ds\_sensor.read\_temp(roms[1])

response = """\

HTTP/1.1 200 OK\r\nContent-Type: text/plain\r\nConnection: close\r\n\r\n

Temp1: {} C\nTemp2: {} C\n""".format(temp1, temp2)

cl.send(response)

cl.close()

print('Cliente desconectado')

* while True: Bucle infinito para manejar conexiones entrantes.
* cl, addr = server.accept(): Acepta una conexión entrante.
* cl\_file = cl.makefile('rwb', 0): Crea un archivo para leer y escribir datos del cliente.
* while True: Bucle para leer datos del cliente.
* if not line or line == b'\r\n': Verifica si la línea está vacía o es un salto de línea.
* roms = ds\_sensor.scan(): Escanea los sensores conectados.
* ds\_sensor.convert\_temp(): Inicia la conversión de temperatura.
* time.sleep\_ms(750): Espera 750 ms para que la conversión termine.
* temp1 = ds\_sensor.read\_temp(roms[0]): Lee la temperatura del primer sensor.
* temp2 = ds\_sensor.read\_temp(roms[1]): Lee la temperatura del segundo sensor.
* response: Crea la respuesta HTTP con las temperaturas.
* cl.send(response): Envía la respuesta al cliente.
* cl.close(): Cierra la conexión con el cliente.

**Simulaciones**

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Recepción en la notebook

Captura de pantalla de computadora

Descripción generada automáticamente

**Conclusión**

Este sistema permite monitorear y analizar las temperaturas en diferentes partes de una habitación, proporcionando datos precisos para tomar decisiones informadas sobre el aislamiento térmico y la eficiencia energética. La notebook se conecta al ESP32, recibe los datos de temperatura y los almacena en un archivo Excel para su análisis diario.

**Importancia del Monitoreo:**

El monitoreo de la temperatura es fundamental para obtener datos reales y precisos sobre las condiciones térmicas en la habitación. Con esta información, se pueden tomar decisiones informadas para aplicar diferentes soluciones de aislamiento térmico, tales como:

* Pinturas reflectivas.
* Aislantes como Isover.
* Revoques térmicos.
* Paredes con aislaciones térmicas.

Estas soluciones ayudan a mejorar la eficiencia energética y el confort térmico, reduciendo la necesidad de sistemas de climatización y, por ende, el consumo de energía.

**Resultados Esperados:**

Se espera obtener un conjunto de datos detallado sobre las variaciones de temperatura en la habitación. Con esta información, se podrán identificar patrones y tomar decisiones para ajustar el sistema de climatización, mejorando así el confort térmico y la eficiencia energética.

A continuación se muestra los archivos adquiridos por el sistema

Interfaz de usuario gráfica, Texto, Tabla

Descripción generada automáticamente

**Tomando el archivo ESP32\_30\_11\_24**

**Interfaz de usuario gráfica, Aplicación, Tabla, Excel

Descripción generada automáticamente**

Un análisis general Gracias por compartir el gráfico. A continuación, te doy una propuesta para mantener la temperatura dentro de la habitación (línea celeste) a 23 grados Celsius de manera constante:

Análisis de la Situación:

1. Línea Celeste: La temperatura dentro de la habitación varía alrededor de los 23°C, con algunas fluctuaciones.

2. \*Línea Roja\*: La temperatura entre la chapa y el cielorraso muestra un aumento significativo, alcanzando los 25.5°C en algunos momentos.

Para mantener la temperatura dentro de la habitación constante a 23°C, se pueden tomar las siguientes medidas:

Propuesta:

1. Mejorar el Aislamiento Térmico:

- Aislamiento de Techo: Agregar material aislante entre la chapa y el cielorraso puede ayudar a reducir la transferencia de calor. Materiales como lana de vidrio, poliestireno expandido (EPS), o poliuretano son efectivos.

- Aislamiento de Paredes: Reforzar el aislamiento en las paredes exteriores para minimizar la ganancia de calor desde el exterior.

2. Uso de Ventilación Pasiva y Activa:

- \*Ventilación Pasiva\*: Instalar ventiladores de techo o extractores de aire que permitan la circulación del aire caliente hacia afuera y el ingreso de aire fresco.

- Ventilación Activa: Utilizar sistemas de aire acondicionado o climatización que permitan mantener una temperatura constante. Configura el termostato del aire acondicionado a 23°C.

3. Control Automático de Temperatura:

- Termostatos Inteligentes: Implementar termostatos inteligentes que ajusten automáticamente la temperatura del aire acondicionado basado en las lecturas de temperatura dentro de la habitación.

- Sensores de Temperatura: Colocar sensores en diferentes partes de la habitación para monitorear y controlar la temperatura de manera más precisa.

4. Uso de Cortinas Térmicas y Persianas: - Cortinas Térmicas: Utilizar cortinas térmicas o persianas reflectantes para reducir la ganancia de calor a través de las ventanas. Esto ayuda a mantener la habitación fresca durante el día.

5. Ventilación Cruzada: - Apertura de Ventanas: Si es posible, abre ventanas estratégicamente para permitir la ventilación cruzada, lo que puede ayudar a mantener una temperatura más uniforme.

Implementación:

1. \*Materiales de Aislamiento\*: Compra e instala materiales de aislamiento en el techo y las paredes.

2. \*Sistemas de Ventilación\*: Instala ventiladores de techo, extractores y, si es necesario, sistemas de aire acondicionado.

3. \*Termostatos y Sensores\*: Configura termostatos inteligentes y coloca sensores de temperatura en la habitación.

4. \*Cortinas y Persianas\*: Instala cortinas térmicas y persianas reflectantes en las ventanas.

**Conclusión:**

El proyecto de monitoreo de temperatura utilizando sensores DS18B20 y un ESP32 que actúa como punto de acceso Wi-Fi proporciona una solución efectiva para la optimización del confort térmico en interiores. La capacidad de recopilar y analizar datos en tiempo real permite realizar ajustes precisos en el sistema de climatización, contribuyendo a un uso más eficiente de la energía.

**Evolución del Proyecto con el Sensor BMP280**

El sensor BMP280 mide temperatura, humedad y presión, lo que puede proporcionar datos más completos sobre las condiciones ambientales. Aquí tienes algunos pasos para integrarlo:

1. **Componentes Adicionales:**
   * Sensor BMP280.
2. **Conexión del BMP280 al ESP32:**
   * Conecta el BMP280 a los pines I2C del ESP32 (SDA y SCL).
3. **Actualización del Código:**
   * Importa la biblioteca para el BMP280.
   * Configura el BMP280 en el código y lee los valores de temperatura, humedad y presión.
   * Modifica el servidor web para incluir estos nuevos datos en la respuesta.
4. **Ejemplo de Código:**
5. import machine
6. import bmp280
7. i2c = machine.I2C(scl=machine.Pin(22), sda=machine.Pin(21))
8. bmp = bmp280.BMP280(i2c)
9. temp = bmp.temperature
10. pressure = bmp.pressure
11. humidity = bmp.humidity # Si el BMP280 tiene sensor de humedad
12. print("Temperature:", temp)
13. print("Pressure:", pressure)
14. print("Humidity:", humidity)
15. **Actualización del Servidor Web:**
    * Incluye los nuevos datos en la respuesta HTTP.
16. response = """\
17. HTTP/1.1 200 OK\r\nContent-Type: text/plain\r\nConnection: close\r\n\r\n
18. Temp1: {} C\nTemp2: {} C\nBMP280 Temp: {} C\nPressure: {} hPa\nHumidity: {} %
19. """.format(temp1, temp2, bmp.temperature, bmp.pressure, bmp.humidity)

**Beneficios de la Integración**

* **Datos más completos:** Con la adición del BMP280, podrás monitorear no solo la temperatura, sino también la humedad y la presión, proporcionando una visión más completa del ambiente.
* **Mejora en la eficiencia energética:** Con datos más precisos, podrás tomar decisiones más informadas para optimizar el sistema de climatización.

Estado actual del trabajo con el bmp280

programa en MicroPython para el Arduino Uno que utiliza un sensor BMP280 para medir la humedad relativa, la temperatura y la presión, y muestra estos datos en una pantalla OLED de 0.96 pulgadas.

**Código en MicroPython**

import machine

import ssd1306

import bmp280

import time

# Configuración del I2C para el BMP280 y la pantalla OLED

i2c = machine.I2C(scl=machine.Pin(5), sda=machine.Pin(4))

# Inicialización del sensor BMP280

bmp = bmp280.BMP280(i2c)

# Inicialización de la pantalla OLED

oled\_width = 128

oled\_height = 64

oled = ssd1306.SSD1306\_I2C(oled\_width, oled\_height, i2c)

while True:

# Lectura de datos del BMP280

temp = bmp.temperature

pressure = bmp.pressure

humidity = bmp.humidity # Si el BMP280 tiene sensor de humedad

# Limpieza de la pantalla

oled.fill(0)

# Mostrar datos en la pantalla OLED

oled.text("Temp: {:.2f} C".format(temp), 0, 0)

oled.text("Pres: {:.2f} hPa".format(pressure), 0, 10)

oled.text("Hum: {:.2f} %".format(humidity), 0, 20)

# Actualizar la pantalla

oled.show()

# Esperar un segundo antes de la próxima lectura

time.sleep(1)

**Explicación del Código**

1. **Configuración del I2C:**
   * Se configuran los pines I2C para el BMP280 y la pantalla OLED. En este caso, se utilizan los pines GPIO 5 (SCL) y GPIO 4 (SDA).
2. **Inicialización del BMP280:**
   * Se inicializa el sensor BMP280 utilizando la biblioteca bmp280.
3. **Inicialización de la Pantalla OLED:**
   * Se configura la pantalla OLED con un ancho de 128 píxeles y una altura de 64 píxeles.
4. **Bucle Principal:**
   * En el bucle principal, se leen los datos de temperatura, presión y humedad del BMP280.
   * Se limpia la pantalla OLED y se muestran los datos leídos.
   * La pantalla se actualiza y el programa espera un segundo antes de realizar la próxima lectura.

**Prueba en Wokwi, no lo puedo hacer porque el sensor BMP280 no está en el panel.**

**Foto de funcionamiento actual**

**Un circuito electrónico

Descripción generada automáticamente con confianza baja**

¡Profesor Maximiliano Simonassi!

Quería expresarle mi más sincero agradecimiento por el curso de MicroPython con el ESP32. Ha sido una experiencia increíblemente enriquecedora que me ha permitido actualizar mis conocimientos y llevar a cabo el proyecto "Monitoreo y Optimización de la Temperatura en Habitaciones Utilizando Sensores DS18B20 y ESP32".

Gracias a su enseñanza, he podido desarrollar un sistema de monitoreo de temperatura que está funcionando conforme a lo esperado. Este proyecto ha sido una oportunidad para aplicar de manera práctica los conceptos aprendidos en su curso.

Aprecio mucho su dedicación y el esfuerzo que pone en cada clase. ¡Gracias por compartir su conocimiento y por inspirarnos a seguir aprendiendo y creciendo en este campo!

Saludos cordiales