**Documentación del Código en MicroPython para proyecto con ESP32**

**Descripción General**

Este código en MicroPython para ESP32 se ha desarrollado para un proyecto de IoT que supervisa diversos sensores, incluyendo sensores de humedad en el suelo, sensores UV (interior y exterior), un sensor de flujo de agua, y sensores DHT22 para medir la temperatura y humedad tanto en el interior como en el exterior. La información recolectada se envía a través de mensajes a bots de Telegram.

**Configuración de Conexión Wi-Fi**

El script comienza configurando la conexión Wi-Fi utilizando las credenciales proporcionadas (ssid y password). La ejecución espera a que la conexión sea exitosa antes de avanzar, asegurando una conexión estable para la transmisión de datos.

python

ssid = 'test'

password = 'server00001'

station = network.WLAN(network.STA\_IF)

station.active(True)

station.connect(ssid, password)

while station.isconnected() == False:

pass

print('Connection successful')

print(station.ifconfig())

**Tokens de los Bots de Telegram**

Se definen tokens para los bots de Telegram, cada uno asociado con un sensor específico. Estos tokens se utilizan más adelante para enviar mensajes a través de la API de Telegram.

python

TOKEN\_BOT\_SUELO = "6611079948:AAH81aBvL9pmFnXA8VB4spo3INBQLESIKc4"

TOKEN\_BOT\_UV = "6739077746:AAHxCQiN3Hy0b7vgCES5YKhO6YbgzJXnM9Q"

TOKEN\_BOT\_FLUJO\_AGUA = "6407146772:AAFwTO-xjL225hLocDTmL28RHrkOBgqABXU"

TOKEN\_BOT\_DHT22 = "6941247101:AAGNo61FBSCMwewgeHfoWMkIBlrEsf7edfY"

**Control de Electroválvula**

Se define el pin para controlar una electroválvula que regula el riego en función de la humedad del suelo. La electroválvula se abre o cierra según las condiciones del suelo.

python

PIN\_ELECTROVALVULA = 19

electrovalvula = Pin(PIN\_ELECTROVALVULA, Pin.OUT)

electrovalvula.value(0) # Inicialmente cerrada

**Función para Enviar Mensajes a Telegram**

La función enviar\_mensaje se encarga de enviar mensajes a través de la API de Telegram. Está diseñada para ser general y reutilizable, tomando como parámetros el token del bot, el mensaje a enviar y el chat ID del grupo o usuario de destino. A continuación, se proporciona una explicación más detallada de esta función:

python

def enviar\_mensaje(token, mensaje):

url = "https://api.telegram.org/bot{}/sendMessage".format(token)

payload = {"chat\_id": "-1002124461151", "text": mensaje}

try:

response = requests.post(url, json=payload)

# Verificar el estado de la respuesta antes de continuar

if response.status\_code == 200:

print("Mensaje enviado exitosamente.")

else:

print("Error al enviar mensaje a Telegram. Código de estado:", response.status\_code)

except Exception as e:

print("Error al enviar mensaje a Telegram:", e)

finally:

# Cerrar la conexión y liberar recursos

if response:

response.close()

**Descripción Detallada de envio a Telegram**

1. **Construcción de la URL y Payload:**
   * La URL se construye concatenando el token del bot al inicio de la URL base de la API de Telegram.
   * El payload (carga útil) del mensaje se define como un diccionario que incluye el chat\_id y el text del mensaje.

python

url = "https://api.telegram.org/bot{}/sendMessage".format(token)

payload = {"chat\_id": "-1002124461151", "text": mensaje}

1. **Envío de la Petición POST:**
   * Se utiliza la biblioteca requests para enviar una solicitud POST a la URL de la API de Telegram.
   * La carga útil se envía como datos JSON en el cuerpo de la solicitud.

python

response = requests.post(url, json=payload)

1. **Verificación del Estado de Respuesta:**
   * Se verifica si la respuesta tiene un código de estado HTTP 200, que indica que la solicitud se procesó correctamente.
   * En caso de cualquier otro código de estado, se imprime un mensaje de error.

python

if response.status\_code == 200:

print("Mensaje enviado exitosamente.")

else:

print("Error al enviar mensaje a Telegram. Código de estado:", response.status\_code)

1. **Manejo de Excepciones:**
   * Se implementa un bloque try-except para manejar excepciones que puedan ocurrir durante el envío del mensaje.
   * Si hay un error, se imprime el mensaje de error.

python

except Exception as e:

print("Error al enviar mensaje a Telegram:", e)

1. **Cierre de la Conexión y Liberación de Recursos:**
   * En el bloque finally, se cierra la conexión para liberar recursos, asegurándose de que se realice incluso si hay un error.

python

finally:

if response:

response.close()

Esta función proporciona un mecanismo robusto para enviar mensajes a Telegram, con manejo de errores y liberación adecuada de recursos. Puedes usar esta función en diferentes partes del código para enviar actualizaciones y alertas a los grupos de Telegram asociados con cada sensor.

**Sensores de Humedad en el Suelo**

Se definen pines ADC para los sensores de humedad en el suelo y se ajustan para una sensibilidad específica. Se realiza la lectura de ADC para calcular el porcentaje de humedad en cada sensor. La función también evalúa el estado general del suelo y controla la electroválvula en consecuencia.

python

# Pines ADC para los sensores de suelo

AOUT\_PINS = [34, 35, 39, 32]

adcs = [ADC(Pin(pin)) for pin in AOUT\_PINS]

for adc in adcs:

adc.atten(ADC.ATTN\_11DB)

La función medir\_sensores\_humedad\_suelo recorre los sensores, calcula el porcentaje de humedad y determina el estado del suelo. También controla la electroválvula según las condiciones del suelo.

python

def medir\_sensores\_humedad\_suelo():

# ...

**Sensor de Flujo de Agua**

Se configura un pin para medir la frecuencia de pulsos generados por el sensor de flujo de agua. La frecuencia se utiliza para calcular el flujo y la cantidad total de litros. La función medir\_flujo\_agua envía los datos a un bot de Telegram específico.

python

# Configuración de la interrupción para el flanco de subida (rising edge)

sf.irq(trigger=Pin.IRQ\_RISING, handler=conteo)

def medir\_flujo\_agua():

# ...

**Sensores UV Interior y Exterior**

Dos funciones (uv\_i y uv\_e) miden la radiación UV interior y exterior respectivamente. Los resultados se clasifican en índices de UV y niveles de riesgo, y se envían a bots de Telegram específicos.

python

def uv\_i():

# ...

def uv\_e():

# ...

**Sensores DHT22 (Temperatura y Humedad)**

La función dht22 mide la temperatura y humedad interior y exterior utilizando sensores DHT22. Los datos se envían a un bot de Telegram específico.

python

def dht22():

# ...

**Bucle Principal**

En el bucle principal, se llaman secuencialmente todas las funciones que obtienen datos de los sensores. Después de cada iteración, el ESP32 espera 15 minutos antes de comenzar una nueva medición.

python

while True:

uv\_i()

uv\_e()

dht22()

medir\_flujo\_agua()

medir\_sensores\_humedad\_suelo()

sleep(60\*15)

Este documento proporciona una visión detallada de cada sección del código, destacando la funcionalidad de cada parte del script. Asegúrate de ajustar los tiempos de espera y frecuencias de medición según las necesidades específicas de tu proyecto.