TECNICATURA SUPERIOR

Telecomunicaciones

Proyecto Integrador

CAPA DE PROCESAMIENTO

ISPC INSTITUTO SUPERIOR POLITÉCNICO CÓRDOBA

Dirección General de EDUCACIÓN TÉCNICA Y FORMACIÓN PROFESIONAL — Ministerio de EDUCACIÓN — CÓRDOBA entre todos

Profesor: Gonzalo Vera

Explicación del código para el Sensor de Temperatura y Humedad AM2320

El AM2320 también se comunica utilizando el protocolo I2C, por lo que es compatible con la configuración actual del ESP32. A continuación te proporciono el código completo para el AM2320.

Explicación paso a paso

1. **Importar Librerías**

```
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_AM2320.h>
#include <WiFi.h>
#include <HTTPClient.h>
```

- **Wire.h**: Librería que permite la comunicación I2C entre el microcontrolador y el sensor.
- **Adafruit_Sensor.h** y **Adafruit_AM2320.h**: Librerías específicas para el sensor AM2320 que facilitan la lectura de datos de temperatura y humedad.
- **WiFi.h**: Librería para conectar el ESP32 a una red Wi-Fi.
- **HTTPClient.h**: Librería que permite enviar solicitudes HTTP (como POST) desde el ESP32 hacia un servidor (API).

2. **Credenciales Wi-Fi**

```
const char* ssid = "tu_SSID";
const char* password = "tu_PASSWORD";
```

- Aquí defines las credenciales de tu red Wi-Fi a la cual el ESP32 se conectará.

3. **URL del Servidor (API)**

```
const char* serverName = "https://api.gonaiot.com/plata/datos_dispositivos";
```

   - Se define la URL del servidor donde se enviarán los datos del sensor en formato JSON. En este caso, es la dirección de tu API.

4. **Objeto del Sensor**

```
Adafruit_AM2320 am2320 = Adafruit_AM2320();
```

   - Se crea un objeto **am2320** que representa el sensor AM2320. Este objeto permitirá interactuar con el sensor para leer los valores de temperatura y humedad.

5. **Función setup()**

```
void setup() {
  Serial.begin(115200);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Conectando a WiFi...");
  }
  Serial.println("Conectado a la red WiFi");

  if (!am2320.begin()) {
    Serial.println("No se puede encontrar el sensor AM2320.");
    while (1);
  }
}
```

   - **Serial.begin(115200);**: Inicializa la comunicación serie para que puedas ver los mensajes de depuración en el monitor serie.
   - **Wi-Fi**: El ESP32 intenta conectarse a la red Wi-Fi con las credenciales definidas anteriormente. Si se conecta correctamente, muestra un mensaje.
   - **Sensor AM2320**: Se inicializa el sensor AM2320. Si no se encuentra el sensor, el ESP32 detiene la ejecución del programa con un bucle infinito (while(1);).

6. **Función loop()**

   - Esta función se ejecuta en bucle y es donde se recopilan los datos del sensor y se envían a la API.

7. **Lectura de Temperatura y Humedad**

```
float temperature = am2320.readTemperature();
float humidity = am2320.readHumidity();
```

   - El ESP32 lee los valores de temperatura y humedad usando las funciones propias del objeto **am2320**.

8. **Mostrar los Valores en el Monitor Serie**

```
Serial.print("Temperatura: ");
Serial.print(temperature);
Serial.println(" °C");

Serial.print("Humedad: ");
Serial.print(humidity);
Serial.println(" %");
```

- Estos mensajes muestran los valores de temperatura y humedad en el monitor serie, útil para depuración.

9. **Verificar Conexión Wi-Fi y Enviar Datos a la API**

```
if (WiFi.status() == WL_CONNECTED) {
  HTTPClient http;
  http.begin(serverName);
  String httpRequestData = "{\"temperatura\":" + String(temperature) + ", \"humedad\":
  http.addHeader("Content-Type", "application/json");
```

- **Verificación de conexión**: Solo si el ESP32 está conectado a Wi-Fi, se procede a crear una instancia de HTTPClient para realizar la solicitud HTTP.
- **URL del servidor**: La función http.begin(serverName); prepara el ESP32 para enviar datos a la URL especificada.
- **Crear el cuerpo de la solicitud**: Se construye un string en formato JSON que contiene los valores de temperatura y humedad.

10. **Enviar los Datos a la API**

```
int httpResponseCode = http.POST(httpRequestData);
```

- Envía una solicitud **POST** a la API con los datos en formato JSON.

11. **Mostrar el Código de Respuesta**

```
if (httpResponseCode > 0) {
  String response = http.getString();
  Serial.println(httpResponseCode);
  Serial.println(response);
} else {
  Serial.print("Error en la solicitud POST: ");
  Serial.println(httpResponseCode);
}
```

- Si la solicitud fue exitosa, se muestra el código de respuesta y el mensaje de la API. Si hubo un error, se muestra el código de error.

12. **Finalizar Conexión HTTP**

```
http.end();
```

- Esta función cierra la conexión HTTP para liberar recursos.

13. **Retraso Antes de la Siguiente Lectura**

```
delay(10000);
```

- Espera 10 segundos antes de volver a leer y enviar los datos.