Teknoloji Fakültesi

MRM4002 – Endüstriyel Haberleşme ve SCADA

DENEY 5 – Web Sitesi Yayınlama - API

Kullanımı

deneyisi

Laboratuvar Raporu

Öğrenci Abdelwahab-Rayis 170219922

1. GİRİŞ

Deneyin Amacı

Web Sitesi Yayınlama - API Kullanımı yapmak

Giriş

Bu rapor, web istekleri aracılığıyla yerel sıcaklık ve API'den alınan sıcaklık verilerini karşılaştırmak için tasarlanmış bir ESP32 tabanlı projenin uygulamasını ve test edilmesini sunar. Proje, bir LM35 sıcaklık sensörü kullanarak yerel sıcaklık verilerini ölçer ve OpenWeatherMap API'sinden alınan sıcaklık verileri ile karşılaştırır.

Kullanılan Bileşenler

- ESP32
- LM35 Sıcaklık Sensörü
- Breadboard
- Jumper kabloları
- Arduino IDE 2.3.2 yüklü dizüstü bilgisayar

Amaçlar

- ESP32 modülünü bir Wi-Fi ağına bağlamak.
- LM35 sıcaklık sensörü ile yerel sıcaklık verilerini ölçmek.
- OpenWeatherMap API'sinden sıcaklık verilerini almak.
- Yerel ve API sıcaklık verilerini bir web sayfasında görüntülemek.

2. DENEY YÖNTEMİ

Donanım Kurulumu

Donanım kurulumu, ekli fotoğraflarda gösterilmektedir. Bağlantılar şu şekildedir:

- LM35 sıcaklık sensörünün VCC pini 5 V'a, GND pini GND'ye ve OUT pini ESP32'nin 34 numaralı analog pinine bağlanmıştır.
- Breadboard üzerindeki bağlantılar jumper kabloları ile sağlanmıştır.

Yazılım Kurulumu

ESP32 modülünü programlamak için Arduino IDE kullanılmıştır. Kod, ESP32'nin Wi-Fi ağına bağlanmasını, yerel sıcaklık verilerini okumasını, OpenWeatherMap API'sinden veri çekmesini ve bu verileri bir web sayfasında görüntülemesini sağlar.

Kütüphaneler

Projede kullanılan kütüphaneler şunlardır:

- WiFi.h: ESP32 modülünü Wi-Fi ağına bağlamak için kullanılır.
- HTTPClient.h: HTTP istemci işlevselliği sağlar ve ESP32'nin HTTP GET ve POST istekleri göndermesine olanak tanır.
- ESPAsyncWebServer.h: ESP32 üzerinde asenkron web sunucusu oluşturmak için kullanılır. Bu, HTTP isteklerini asenkron olarak işleyerek daha hızlı ve verimli bir sunucu uygulaması sağlar.

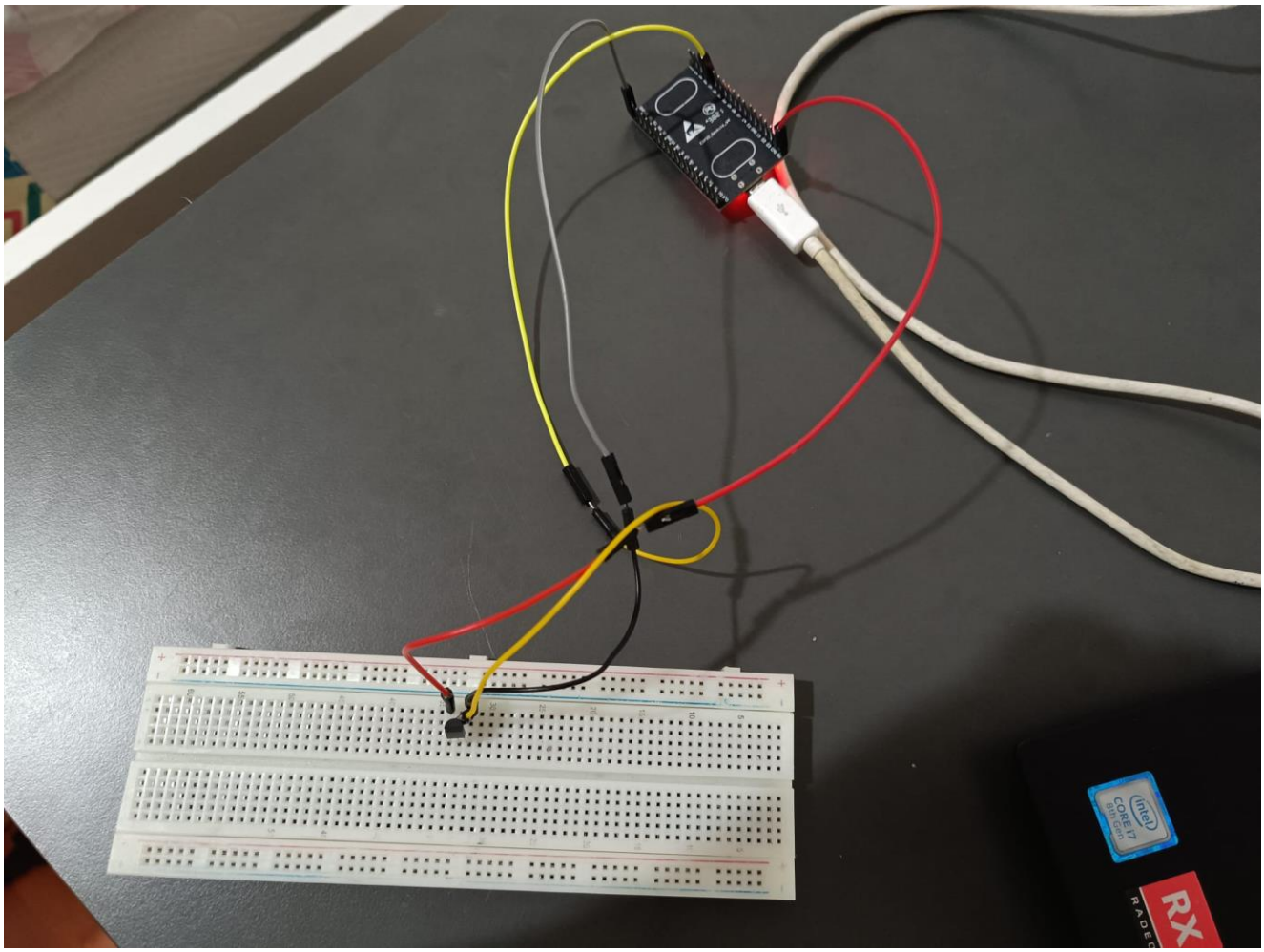
KODLAR

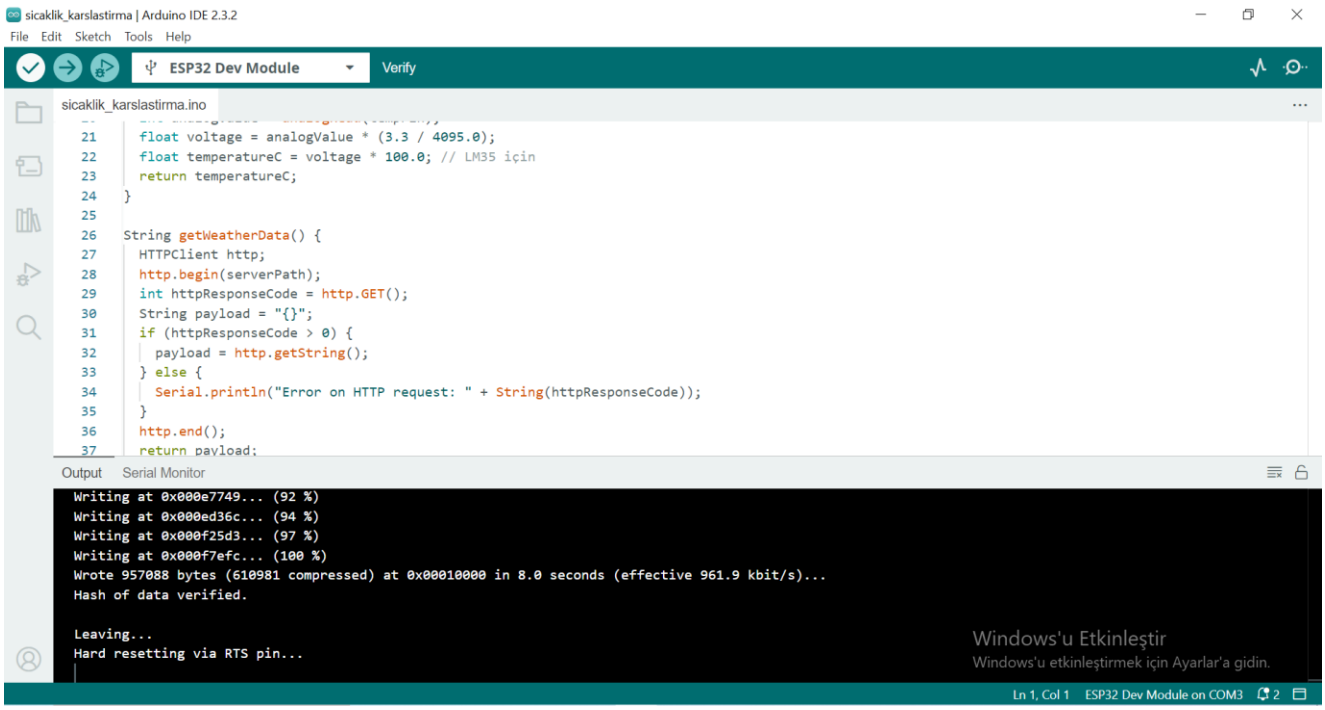
```
#include <WiFi.h>
#include <ESPAsyncWebServer.h>
#include <HTTPClient.h>

// Wi-Fi credentials
const char* ssid = "HUAWEI P30 lite";
const char* password = "00000000";
```

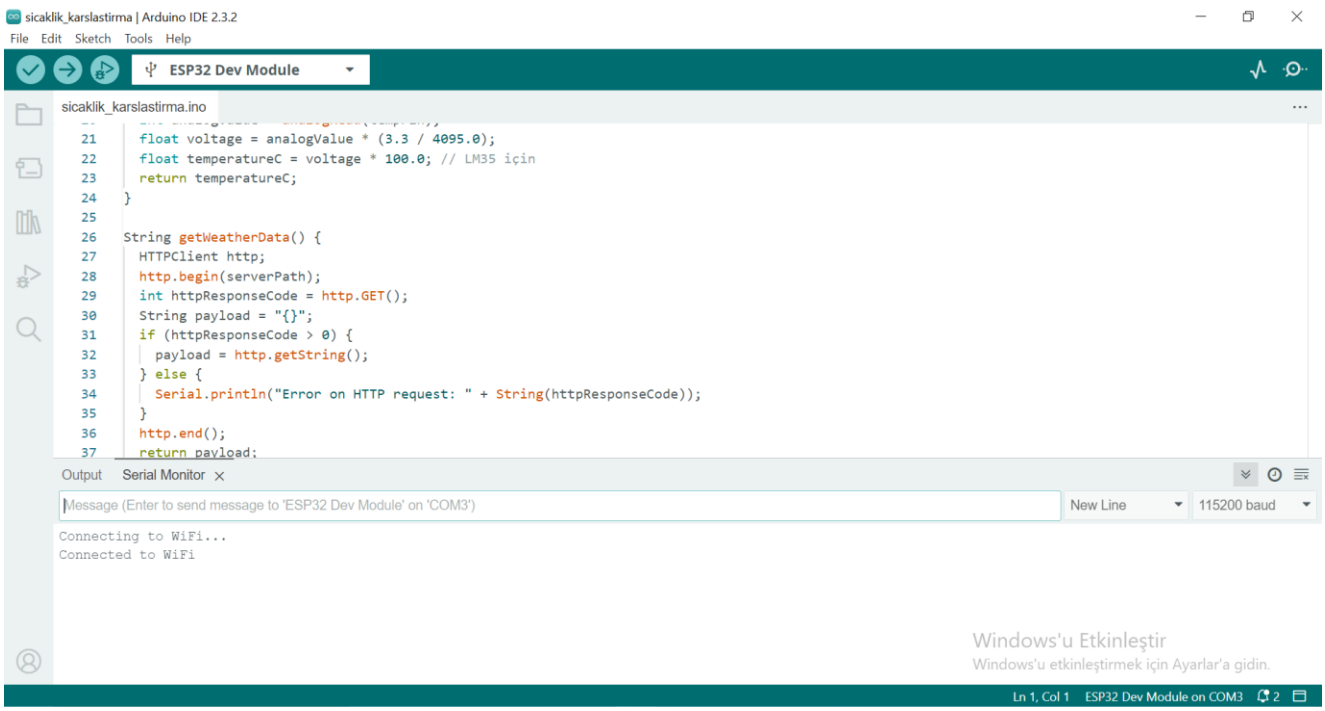
```
// OpenWeatherMap API details
const char* apiKey = "7a78b3eb7ef463c424d2ba210c54c218";
const char* location = "Istanbul,tr"; // Konum bilgisi
String serverPath = "http://api.openweathermap.org/data/2.5/weather?q=" +
String(location) + "&appid=" + String(apiKey) + "&units=metric";

AsyncWebServer server(80);

// Temperature sensor pin
const int tempPin = 34; // LM35 bağlı olduğu pin

float readTemperature() {
  int analogValue = analogRead(tempPin);
  float voltage = analogValue * (3.3 / 4095.0);
  float temperatureC = voltage * 100.0; // LM35 için
  return temperatureC;
}

String getWeatherData() {
  HTTPClient http;
  http.begin(serverPath);
  int httpResponseCode = http.GET();
  String payload = "{}";
  if (httpResponseCode > 0) {
    payload = http.getString();
  } else {
    Serial.println("Error on HTTP request: " + String(httpResponseCode));
  }
  http.end();
  return payload;
}

String processor(const String& var) {
  if (var == "LOCAL_TEMP") {
    return String(readTemperature());
  } else if (var == "API_TEMP") {
    String weatherData = getWeatherData();
    int tempIndex = weatherData.indexOf("\"temp\":");
    if (tempIndex != -1) {
      tempIndex += 7;
      String apiTemp = weatherData.substring(tempIndex, weatherData.indexOf(",",
tempIndex));
      return apiTemp;
    } else {
      return "N/A"; // Eğer veri bulunamazsa
    }
  }
  return String();
```

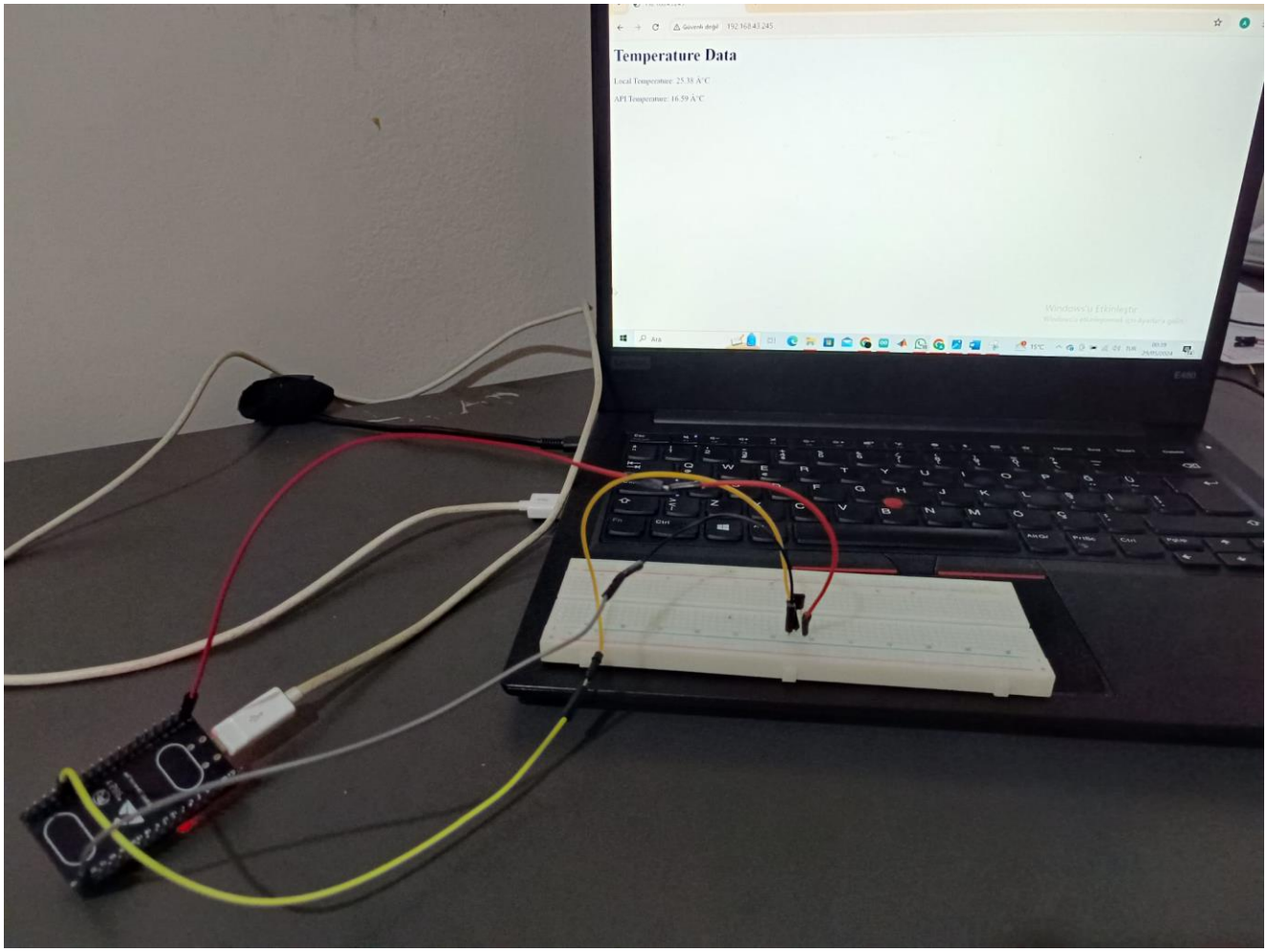
```
}

void setup() {
  Serial.begin(115200);

  // Connect to Wi-Fi
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi...");
  }
  Serial.println("Connected to WiFi");

  // Start server
  server.on("/", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send_P(200, "text/html", "<html><body><h1>Temperature Data</h1><p>Local
Temperature: %LOCAL_TEMP% °C</p><p>API Temperature: %API_TEMP% °C</p></body></html>",
processor);
  });

  server.begin();
}

void loop() {
  // Nothing to do here, everything is handled in the background
}
```

3. DENEY SONUÇLARI




sicaklik_karslastirma | Arduino IDE 2.3.2
File Edit Sketch Tools Help
ESP32 Dev Module
Verify
sicaklik_karslastirma.ino
21 float voltage = analogValue * (3.3 / 4095.0);
22 float temperatureC = voltage * 100.0; // LM35 için
23 return temperatureC;
24 }
25
26 String getWeatherData() {
27 HTTPClient http;
28 http.begin(serverPath);
29 int httpResponseCode = http.GET();
30 String payload = "{}";
31 if (httpResponseCode > 0) {
32 payload = http.getString();
33 } else {
34 Serial.println("Error on HTTP request: " + String(httpResponseCode));
35 }
36 http.end();
37 return payload;
Output Serial Monitor
Writing at 0x000e7749... (92 %)
Writing at 0x000ed36c... (94 %)
Writing at 0x000f25d3... (97 %)
Writing at 0x000f7efc... (100 %)
Wrote 957088 bytes (610981 compressed) at 0x00010000 in 8.0 seconds (effective 961.9 kbit/s)...
Hash of data verified.
Leaving...
Hard resetting via RTS pin...
Windows'u Etkinleştir
Windows'u etkinleştirmek için Ayarlar'a gidin.
Ln 1, Col 1 ESP32 Dev Module on COM3 2


sicaklik_karslastirma | Arduino IDE 2.3.2
File Edit Sketch Tools Help
ESP32 Dev Module
sicaklik_karslastirma.ino
21 float voltage = analogValue * (3.3 / 4095.0);
22 float temperatureC = voltage * 100.0; // LM35 için
23 return temperatureC;
24 }
25
26 String getWeatherData() {
27 HTTPClient http;
28 http.begin(serverPath);
29 int httpResponseCode = http.GET();
30 String payload = "{}";
31 if (httpResponseCode > 0) {
32 payload = http.getString();
33 } else {
34 Serial.println("Error on HTTP request: " + String(httpResponseCode));
35 }
36 http.end();
37 return payload;
Output Serial Monitor
Message (Enter to send message to 'ESP32 Dev Module' on 'COM3')
New Line
115200 baud
Connecting to WiFi...
Connected to WiFi
Windows'u Etkinleştir
Windows'u etkinleştirmek için Ayarlar'a gidin.
Ln 1, Col 1 ESP32 Dev Module on COM3 2



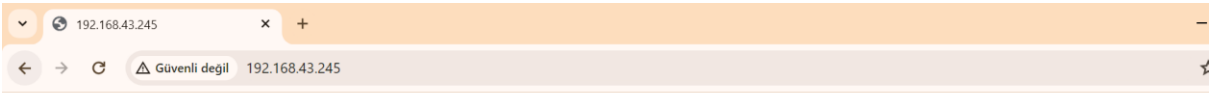
192.168.43.245

Güvenli değil 192.168.43.245

Temperature Data

Local Temperature: 25.79 Â°C

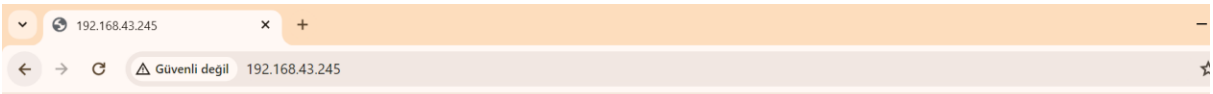
API Temperature: 16.59 Â°C



Temperature Data

Local Temperature: 24.50 Â°C

API Temperature: 16.59 Â°C



Temperature Data

Local Temperature: 19.34 Â°C

API Temperature: 16.59 Â°C

,

4. DEĞERLENDİRME

Test ve Sonuçlar

Donanım ve yazılım kurulumları, doğru çalıştığından emin olmak için test edilmiştir. Sağlanan görüntülerde görüldüğü üzere:

- ESP32 modülü Wi-Fi ağına başarıyla bağlanmıştır.
- Yerel sıcaklık verileri LM35 sensörü ile okunmuş ve web sayfasında görüntülenmiştir.
- OpenWeatherMap API'sinden alınan sıcaklık verileri de web sayfasında başarıyla gösterilmiştir.

Sonuç

Bu proje, ESP32 modülünün HTTP istekleri kullanarak bir Wi-Fi ağı üzerinden sıcaklık verilerini nasıl toplayabileceğini ve bu verileri bir web sayfasında nasıl görüntüleyebileceğini başarılı bir şekilde göstermektedir. Kurulum, daha karmaşık IoT uygulamaları için genişletilebilir, böylece birden fazla sensör verisi uzaktan izlenebilir ve kontrol edilebilir.

Kaynaklar

- OpenWeatherMap API dökümantasyonu: https://openweathermap.org/api