Xiaomi hőmérséklet és páratartalom szenzor rekreálása

A projekt célja egy Xiaomi hőmérséklet- és páratartalom-érzékelő rendszer rekreálása, amely adatokat küld egy felhőalapú platformra. A rendszer az ESP8266 mikrokontroller segítségével rögzíti a DHT11 szenzortól kapott hőmérsékleti és páratartalom adatokat, majd ezeket egy felhőszolgáltatásba továbbítja, ahol az adatok hisztogram formában jeleníthetők meg. Az OLED kijelző az aktuális értékeket helyben is megjeleníti, míg a rendszer működését egy nyomógomb vezérli, és akkumulátorról működik, így hordozható megoldást kínál.

Csapattagok:

Perjési Dániel

- Eszközök beszerzése, berendezés összeállítása, funkcionalitást biztosító kód megírása

Gyarmati Gábor

- Cloud szolgáltatás hozzáadása, kód kiegészítése, dokumentáció elkészítése

Felhasznált eszközök típusai és szerepük:

Eszköz	Típus	Szerep
ESP8266 (ESP-12E)	Mikrokontroller	A központi vezérlő egység, amely adatokat gyűjt a szenzortól, feldolgozza azokat, és továbbítja a felhőbe.
0.91 OLED kijelző 128x32 I2C	Kijelző	A szenzor által mért hőmérséklet- és páratartalom értékek helyi megjelenítésére szolgál.
DHT11 Hőmérséklet és páratartalom érzékelő	Szenzor	Mérési adatokat szolgáltat a környezeti hőmérsékletről és páratartalomról.
DC-DC konverter (LM2596)	Feszültségstabilizátor	Az ESP8266 és a kijelző tápellátásának biztosítása az akkumulátorról.
18650 Akku cella tartó (3 db)	Akkumulátortartó	A rendszer energiaellátásához szükséges 18650-es akkumulátorok tárolására és rögzítésére szolgál.
Nyomógomb	Vezérlő	Lehetővé teszi a rendszer indítását és leállítását
Breadboard és Prototype Board	Kapcsolási felület	Az áramkör teszteléséhez és fejlesztéséhez használatos felület a kapcsolások kialakításához.
Vezetékek	Kábelezés	Az eszközök közötti elektromos összeköttetést biztosítják az eszközben.

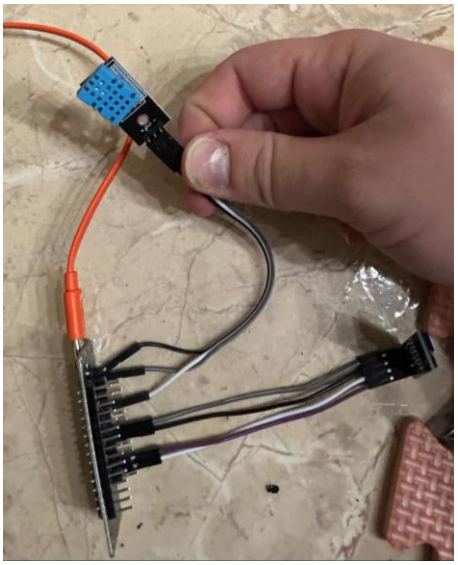
Felhasznált szoftver kódja

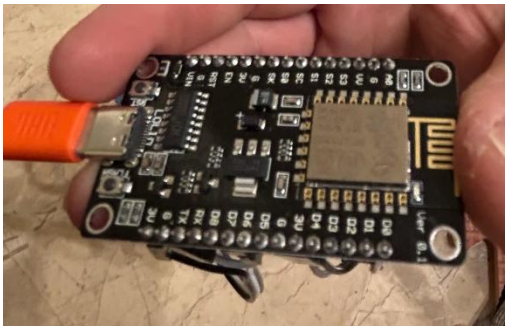
```
#define BLYNK_TEMPLATE_ID "TMPL4_84jEg0Y"
#define BLYNK_TEMPLATE_NAME "Temperature and Humidity Monitor"
#define BLYNK_AUTH_TOKEN "2MVrOFsBpG2kUGK0tGn12Ih2_92srx9G"

#include <Wire.h>  // I2C kommunikációhoz szükséges könyvtár
#define BLYNK_PRINT Serial  // Blynk debuggoláshoz a soros monitort használjuk
#include <ESP8266WiFi.h>  // ESP8266 WiFi könyvtár
#include <BlynkSimpleEsp8266.h>  // Blynk könyvtár ESP8266-hoz
#include <Adafruit_GFX.h>  // OLED képernyőhöz szükséges könyvtár
#include <Adafruit_SSD1306.h>  // OLED képernyő kezelése
#include <DHTesp.h>  // DHT szenzor könyvtár

// WiFi beállítások
const char* ssid = "DIGI_d7e278";  // WiFi SSID (hálózat neve)
const char* password = "********";  // WiFi jelszó

// OLED képernyő beállítása
#define SCREEN_WIDTH 128  // Képernyő szélessége
#define SCREEN_HEIGHT 64  // Képernyő magassága
#define SCL_PIN 5  // I2C SCL pin
#define SDA_PIN 4  // I2C SDA pin
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);  // OLED
kijelző inicializálása

// DHT11 szenzor beállítása
DHTesp dht;  // DHT szenzor objektum létrehozása
#define DHT_PIN 13  // DHT11 szenzor adat lábának beállítása

void setup() {
  // Soros monitor beállítása
  Serial.begin(9600);  // Soros kommunikáció inicializálása (debugging)

  // WiFi kapcsolat beállítása
  Blynk.begin(BLYNK_AUTH_TOKEN, ssid, password);  // Blynk kapcsolat
inicializálása a megadott WiFi adatokkal

  // DHT11 szenzor inicializálása
  dht.setup(DHT_PIN, DHTesp::DHT11);  // A DHT11 szenzort beállítjuk a
megfelelő pinre

  // OLED képernyő inicializálása
  if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {  // Ellenőrizzük, hogy az
OLED képernyő megfelelően inicializálódott-e
    Serial.println(F("SSD1306 allocation failed"));  // Hibaüzenet, ha nem
sikerült a képernyő inicializálása
    for (;;);  // Végtelen ciklus, ha hiba történt
```

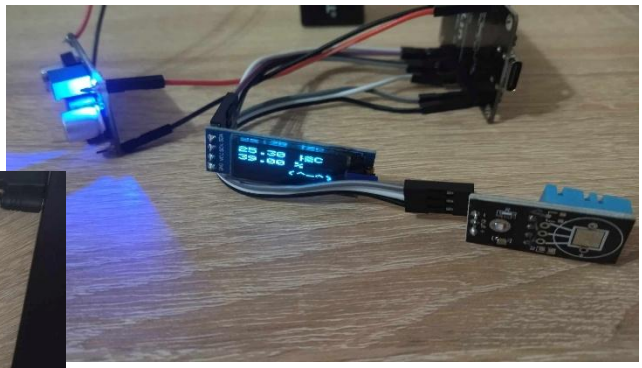
```
  }
  delay(2000);  // 2 másodperces késleltetés

  // Képernyő törlése és szöveg színének beállítása
  display.clearDisplay();  // Képernyő törlése
  display.setTextColor(WHITE);  // Fehér színű szöveg beállítása
}

void loop() {
  // Blynk hívás folyamatos működtetése
  Blynk.run();  // Folyamatosan kommunikálunk a Blynk felhővel

  // Minimum mintavételezési idő biztosítása
  delay(dht.getMinimumSamplingPeriod());  // A DHT szenzor mintavételezési
idejének betartása

  // Páratartalom és hőmérséklet adatainak lekérése
  float humidity = dht.getHumidity();  // Páratartalom lekérése a DHT
szenzorból
  float temperature = dht.getTemperature();  // Hőmérséklet lekérése a DHT
szenzorból

  // OLED kijelző frissítése
  display.clearDisplay();  // Képernyő törlése, hogy új adatokat
jeleníthessünk meg

  // Hőmérséklet kijelzése
  display.setTextSize(2);  // A szöveg méretének beállítása
  display.setCursor(0, 0);  // A szöveg kezdő pozíciója
  display.print(temperature);  // Hőmérséklet kiírása
  display.print(" °C");  // Hőmérséklet mértékegység kiírása

  // Páratartalom kijelzése
  display.setTextSize(2);  // A szöveg méretének beállítása
  display.setCursor(0, 25);  // A szöveg kezdő pozíciója
  display.print(humidity);  // Páratartalom kiírása
  display.print(" %");  // Páratartalom mértékegység kiírása

  // Komfortérzet számítása
  byte perception = dht.computePerception(temperature, humidity, false);  // A
komfortérzet számítása
  display.setTextSize(2);  // A szöveg méretének beállítása
  display.setCursor(64, 52);  // A szöveg kezdő pozíciója
  if (perception == 1 || perception == 2 || perception == 3) {  // Ha a
komfortérzet jó
    display.print("(^-^)");  // Pozitív ikon
  } else {  // Ha a komfortérzet nem megfelelő
    display.print("(-_-)");  // Negatív ikon
  }
```

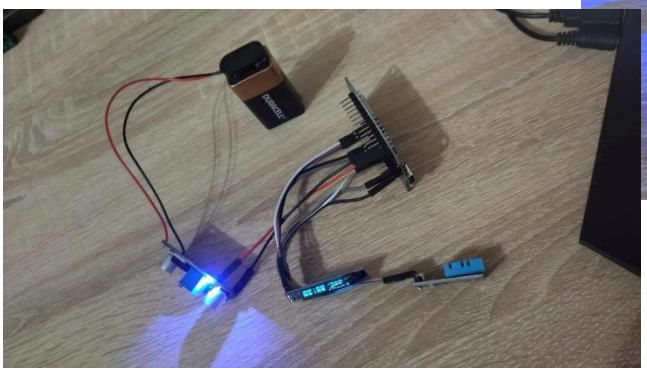
```
  // Képernyő frissítése
  display.display();  // Az összes adat frissítése és kijelzése a képernyőn

  // Adatok küldése a Blynk felhőbe
  Blynk.virtualWrite(V0, temperature);  // Hőmérséklet adat küldése a Blynk felhőbe (V0 virtuális pin)
  Blynk.virtualWrite(V1, humidity);  // Páratartalom adat küldése a Blynk felhőbe (V1 virtuális pin)

  delay(2000);  // 2 másodperces késleltetés a következő mérés előtt
}
```

Média









Áramköri rajz