IOT adatgyűjtés és adatelemzés

Az ESP-k és a Grafana közötti kapcsolat megvalósítás

A feladat

Részvevő személyek:

- **Kurdi Barnabás**
- **Kedves Áron Csanád**

A feladat:

A féléves feladat során a csapat tagjai különböző adatokat gyűjtenek ESP-kre szerelt szenzorok segítségével adatokat gyűjtenek egy esztergagép működéséről. Ebben a részfeladatban azt kellett megvalósítanunk, hogy az ESP-k által mért adatokat eltároljuk és megjelenítsük. A megvalósításhoz kaptunk két szervert. A feladat megoldására az MQTT, InfluxDb és Grafana eszközöket ajánlották nekünk.

- **MQTT**-t használtuk, hogy az ESP-kről megérkező sok adatot könnyen össze tudjuk gyűjteni ezzel a publish-subscribe rendszerrel
- **InfluxDb**-t használtuk, hogy az összegyűjtött adatokat eltároljuk, valamint erre könnyű illeszteni a Grafana programot
- **Grafana**-t használtuk, hogy az összegyűjtött adatokat egy könnyen értelmezhető dashboardon megjelenítsük

Az előzetes tudásunk:

A feladat megkezdésekor csak a két szerver IP címét ismertük és nem tudtuk pontosan mik vannak ezeken. Az ajánlott eszközök közül eddigre mindketten megismertük az MQTT működését, hiszen azt már megvalósítottuk a kölcsönbe kapott ESP-n. A többi rendszert nem ismertük.

A megvalósítás

Szerverek megismerése:

A szerverekhez a belső hálózatról SSH segítségével kapcsolódtunk. Frissítések után linux parancsokkal megnéztük melyik gépen melyik program található. Ebből megtudtuk, hogy:

- **192.168.33.212:** InfluxDb található rajta és már vannak benne adatbázisok
- **192.168.33.211:** Fut rajta egy MQTT szerver, erre később MQTT explorer segítségével kapcsolódtunk

ESP-MQTT kapcsolat megvalósítása

Az ESP MQTT szerverhez kapcsolása egy viszonylag egyszerű feladat. Első lépésként azt ESP-t a hozzá tartozó könyvtár segítségével csatlakoztatni kell Wi-Fi hálózathoz.

ESP 8285 esetén példa:

```
#include "ESP8266WiFi.h"

const char* ssid = "vlan152"; //A WiFi hálózat neve
const char* password = "alma"; //A WiFi hálózathoz tartozó jelszó

WiFiClient espClient;

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

        // WiFi csatlakozás
        WiFi.begin(ssid, password);
        while (WiFi.status() != WL_CONNECTED)
        {
                delay(500);
                Serial.print("*");
        }
}
```

Ha már megvan az internet kapcsolat csatlakozhatunk a hálózatunkon levő MQTT szerverhez. Ismerünk kell ehhez természetesen az MQTT szerver IP címét és port számát. A default MQTT szerver port az 1883.

Ennek a kapcsolatnak a létrehozása és tesztelése során érdemes lehet MQTT explorerrel figyelni, hogy megjelenik-e a szerveren a csatlakoztatni kívánt eszköz. Ha nincs esetleg MQTT explorer használata nélkül vizsgálni a szerverre beérkezett üzeneteket definiálhatunk callback függvényt, ami a soros porton kiírja az MQTT szerverre megérkezett üzeneteket.

Először létre kell hoznunk egy client-et, majd ehhez publish-subscribe módokon tudunk csatlakozni. Az publish és subsrice parancsokban megadjuk hova szeretnénk csatlakozni. Ha nincs ilyen, akkor az MQTT szerver létrehozza nekünk.

ESP 8285 esetén példa:

```
const char *mqtt_broker = "192.168.33.211"; //MQTT szerver IP címe
const int mqtt_port =  1883; //MQTT port
const char* user = ""; //MQTT csatlakozáshoz használt felhasználó, erre nem volt szükségünk
const char* pass = ""; //MQTT csatlakozáshoz használt jelszó, erre nem volt szükségünk

WiFiClient espClient;
PubSubClient client(espClient);

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

        client.setServer(mqtt_broker, mqtt_port);
        client.setCallback(callback); //Callback függvény hozzádása, ha így szeretnénk debuddolni

        //Csatlakozás az MQTT szerverhez
        while (!client.connected()) {
                        Serial.println("Connecting to public emqx mqtt broker.....");
                if (client.connect("esp8266-client",user,pass)) {
                        Serial.println("Public emqx mqtt broker connected");
                } else {
                        Serial.print("failed with state ");
                        Serial.print(client.state());
                        delay(2000);
                }
        }
        // publish és subscribe kapcsolatok megvalósítása az MQTT szerverrel
        client.publish("esp8266/control", "hello emqx");
        client.subscribe("esp8266/control");
}

        // Egy konkrét üzenet küldés
                char message[100];
                sprintf(message,"eszterga,loc=alma rpm=%f", rpm);
                client.publish("esp8266/rpm", message);
```

Példa callback függvényre:

```
void callback(char *topic, byte *payload, unsigned int length) {
        Serial.print("Message arrived in topic: ");
        Serial.println(topic);
        Serial.print("Message:");
        for (int i = 0; i < length; i++) {
                Serial.print((char) payload[i]);
        }
        Serial.println();
        Serial.println("-----------------------");
}
```

InfluxDb szerver előkészítése

Miután MQTT szerverre már jól tudtunk adatokat küldeni, elkezdtünk az InfluxDB működésével foglalkozni. Az SSH csatlakozás után az influx utasítással könnyedén be tudtunk lépni az influxd-be.

Itt a SHOW DATABASES paranccsal megnéztük milyen adatbázisok vannak, majd a CREATE DATABASE esztergagep paranccsal létrehoztuk a számunkra szükséges adatbázist.

InfluxDb elkezdéséhez hasznos lehet a hivatalos dokumentáció get-started része ezen a linken.

Az MQTT szerver InfluxDb szerver összekapcsolása

Ez a feladat rész okozta nekünk a legtöbb gondot. Elsőre rossz úton indultunk el a megoldás felé.

MQTT, InfluxDb bridge készítése Python kóddal

Mivel nem ismertük a rendszer működését és mikéntjét először ezen Visualize MQTT Data with InfluxDB and Grafana tutorial alapján kezdtünk el dolgozni. Az oldal segítséget nyújt az influxdb megéréséhez és installálásához.

A tutorial alapján a következő lépés az InluxDb és az MQTT szerver összekötése. Ehhez a tutorialban egy Python kódot használ a szerző. Az általa írt script Az MQTTből érkező információkat egy előre definiált REGEX kifejezéssel darabolja, majd az ebből kinyert adatokat egy JSON fájl-á alakítja, amit utána belerak az InfluxDb szerverbe.

Ezt a python kódot mi is létrehoztuk és kicsit változtattunk rajta, hogy a mi adatainkhoz jobban illjen. Második, harmadik átírásra működni látszott. Egy folyamatosan működő szenzorról küldtünk RPM értékeket pontosan kiírta a futó script.

Következő indításkor azonban minden indításkor leállt a script futása. Ez azért következett, be mert az ESP csatlakozásakor egy üdvözlő üzenetet küldtünk az ESP-ről, amit nem tudott az előre megadott float adattá alakítani. Ezután derült ki az a probléma is, hogy hiba érkezett meg sikeresen a scripthez az adat, az nem került be az adatbázisba.

Ezeken a hibákon kívül, azért is vetettük el ennek a módszernek a használatát, mert ha még helyesen működik is nagyon nehézkes kibővíteni a feldolgozott adatok struktúráját és mivel jóval több eszköz fog jóval többféle adatot küldeni, ezért ez egy problémás megoldás lenne.

MQTT, influxDb megvalósítása Telegraf használatával

A script alapú megoldás elvetése után találtuk meg a Telegraf nevű eszközt, amely egy influxdb-hez kapcsolódó eszköz. A telegraf egyszerűvé teszi a kapcsolat létrahozását, mert MQTT subscribe-ként csatlakozik az MQTT szerverre és utána könnyedén helyezi el az influxdb-ben az adatokat.

A telegrafot telepítettük és konfiguráltuk. A helyes konfigurációk megadása és az MQTT üzenet felépítésének javítása után az adatbázisban elkezdtek megjelenni az ESP által küldött adatok.

A telegaf-ot és a grafanat ez alapján az tutorial alapján csináltuk: link.

Grafana telepítése és összekapcsolása az InfluxDb szerverrel

Végső lépés az adatok megjelenítése volt. Erre a korábban említett Grafana-t használtuk. A Grafana egy könnyen telepíthető alkalmazás. Telepítés után a Grafa a localhost 3000-es portján jelenik meg.

Ekkor a szervert már nem SSH segítségével kezeltük, hanem csatlakoztattunk képernyőt és egeret, billentyűzetet a szerverhez. A grafana grafikus felületét a localhost 3000-es protján értük el.

Itt megadtuk az admin felhasználót.

- **Felhasználó**: admin
- **Jelszó**: temalabor

Bejelentkezés után létrehoztunk egy Dashboardot, ahol a küldött RPM értékeket jelenítettük meg egy gafikonon. Ezt a konkrét dashboardot 5 másodperces frissítésekkel használtuk.

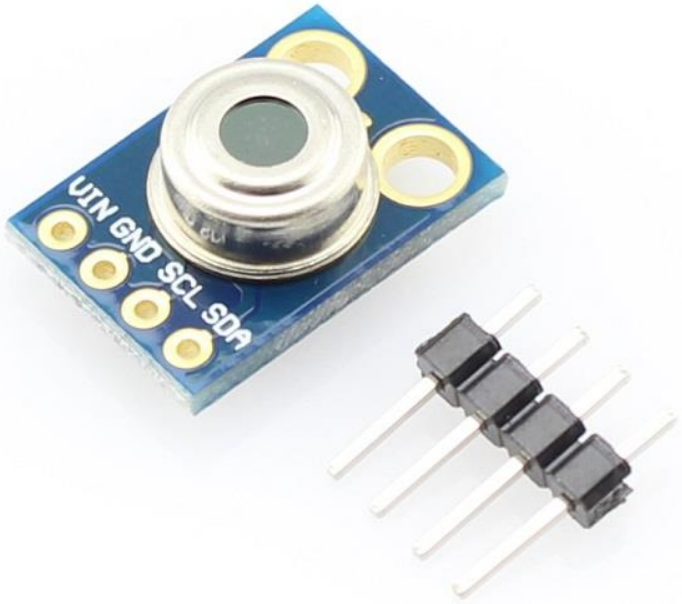
2020. 12. 13. Kurdi Barnabás F0DJNW

Felhasznált szenzorok

A munka során két szenzort állítottam össze, ezek rövid leírása következik:

mlx90614

Az első szenzor egy HW-691 mlx90614 hőérzékelő szenzor.



https://i1.wp.com/arduinolearning.com/wp-content/uploads/2015/11/GY-906-2-300x200.jpg?resize=300%2C200

A szenzornak 4 lába van: áramellátás, földelés, órajel és az adatjel lába. Az adatlábról két különböző jelet lehet beolvasni: a szenzor belső hőmérsékletét és a szenzor által mért hőmérsékletet. Az adatok posztprocesszálására nem volt szükség, egyszerűen továbbítottam a Celsiusban beolvasott adatokat MQTT-n keresztül.

A szenzor relative magas hőmérsékleten képes működni, -40-től +125 Celsiusig van kalibrálva a belső hőmérsékletének mérése, és -70-től +380 fokig képes a külső hőmérséklet mérésére.

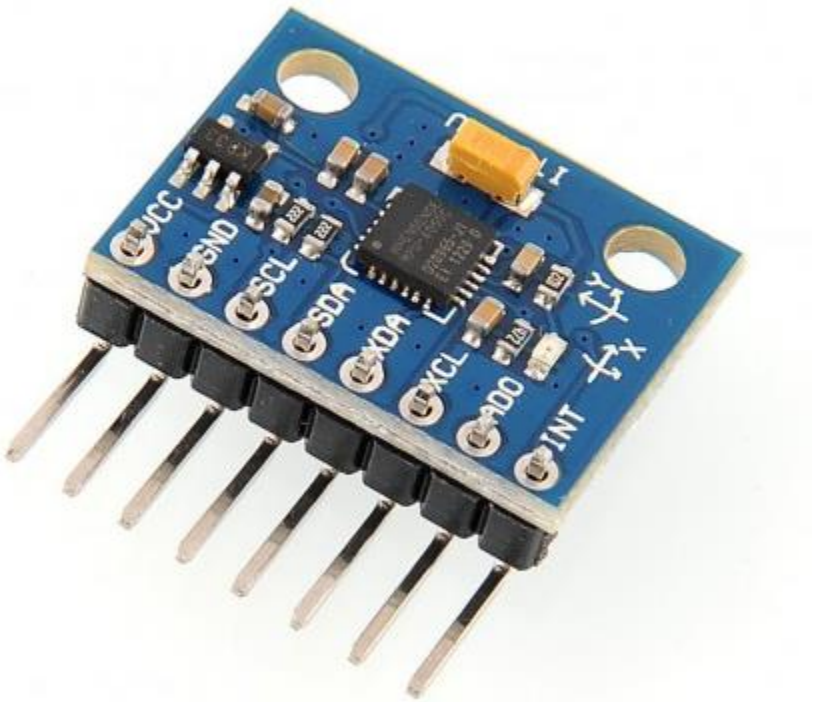
A könnyebb beolvasás érdekében a következő könyvtárt használtam fel:

https://github.com/adafruit/Adafruit-MLX90614-Library

2020. 12. 13. Kurdi Barnabás F0DJNW

MPU6050

Az MPU6050 GY-521 egy 3 tengelyű gyorsulásmérő és 3 tengelyű giroszkóp.



https://5.imimg.com/data5/WH/XQ/GLADMIN-23108818/gy-521-mpu6050-accelerometer-and-gyroscope-sensor-500x500.png

Ez a szenzor 6 különböző lábbal rendelkezik, azonban a mérések elvégzéséhez ezek közül csupán négyet használtam fel, az előző szenzorhoz hasonlóan az áramellátást, a földelést, az óra-és adatjelet.

Az adatjelről egyszerre 6 különböző adatot tudunk kiolvasni: a gyorsulást és az elfordulás mértékét az X, Y és Z tengelyek mindegyikén. Ezek közül a gyorsulást használtam fel.

A feladat a rezgés frekvenciájának kiszámítása volt a gyorsulás változásának segítségével. Ezt a feladatot FFT-vel, azaz Fast Fourier Transformation segítségével oldottam meg. 1024 minta beolvasása után végeztem az adatokon Fourier-transzformációt. Sajnos a használt szenzor félre van kalibrálva, így az adatok és a számításom helyességéről nem tudtam meggyőződni.

A felhasznált könyvtárak:

- https://github.com/kosme/arduinoFFT
- https://github.com/adafruit/Adafruit_MPU6050

2020. 12. 13. Kurdi Barnabás F0DJNW

Eredményünk

A feladat elvégzése sok szempontból eredményesnek tekinthető: sikerült vezeték nélkül adatokat küldeni az MQTT-re, és onnan az InfluxDB adatbázisba, majd ezeket az adatokat megjeleníteni vizuálisan a Grafana segítségével.

Sajnos a vírushelyzetre való tekintettel a szenzorokat végül nem tudtuk egy dobozba összeállítani, így a végleges produktum nem állt elő, azonban az összes