**Ce am :**

- comunicare BLE

- Afisare valori caracteristici pe GUI (ceasuri)

- GUI separat pentru conexiune BLE : scanare( cu filtru ) , conectare la device gasit, conectare la toate serviciile gasite ( serviciu baterie, serviciu AQI , etc )

**Pana voi avea dispozitivul pot sa:**

**1)NEAPARAT** (cu date mock-uite deocamdata) Sa aleg o metoda de calcul AQI - din cele din documentatie : nu va fi o metoda în care datele se acumulează în ore intregi ,

- Vreau sa vad un AQI in timp real => De prefferat o metoda care nu doar ia indicele maxim - ci agregheaza tot poluantii

- Sa clarific cu unitatile de masura : ce primesc de la senzor , cum transform in ug/m3 si mai apoi in SubIndice

- Afisare date in GUI in timp real

- Notificare cand apare o valoare peste o limita de siguranta

**2)In functie de timp ramas :**

- Sa Creez o baza de date : vezi daca FireBase sau SQL Lite -

- Tabela 1:

- Loc Masurare : cheie primara

- Valori sub-indici - cate un camp pentru fiecare poluant

- Valoarea AQI ( in functie de metoda)

- Data si ora masurarii

- Optiuni din GUI: Start Recording to DB - doar cand am date disponibile inregistrez in GUI

- Sa pot sa iau din DB datele dintr-un anumit loc, pentru o anumita perioada + afisarea lor intr-un grafic

Cand am dispozitivul :

NEAPARAT

1- mentinere conexiune pe o durata mai lunga

2- Sa primesc date periodic de la senzor - constant

3- Afisez datele pe GUI

IN FUCNTIE DE TIMP RAMAS:

4 - Inregistrez in baza de date + interogare pe Loc + Perioada

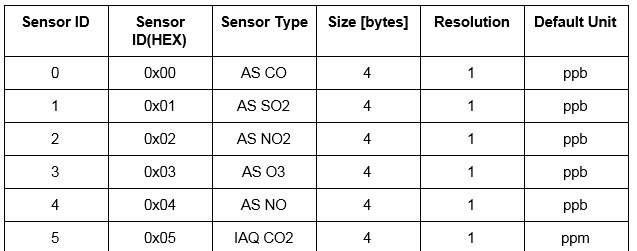
Prezentare PPT Disertatie : IDEI

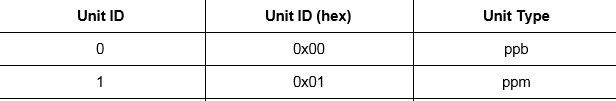
- prezentare problema mare poluare : context actual - CoVid-19 afecteaza mai mult oamenii din zone poluate

1. **CALCUL AQI**

* metoda simpla : Iau Toti sub-indicii la un momentdat si vad care e Maximul )
* afisez in timp real sub-indicii la toti poluantii

* iau PPM si transform in ug/m3
* 4 bytes :32 biti – suficient





ARDUINO :

-<http://www.martyncurrey.com/hm-10-bluetooth-4ble-modules/>

Cod pentru MQ-7 : <https://www.teachmemicro.com/use-mq-7-carbon-monoxide-sensor/>

Rezolvare problema formula ...afiseaza 51 pentru prea multe valori

Mail Prof :

-Ca pt disp pair nu mai faci scan ci getbounddevices.

Am reusit sa ma conectez la dispozitiv, imi citeste caracteristicile, dar n-ai facut partea de conectare la caracteristica de notificare. Poti sa faci si partea asta te rog.

In plus nu prea sunt sigur ca sunt ok functiile de connect si callback, ca nu imi merge de fiecare data. Ai incercat sa mergi pe modul de implementare de la

<https://developer.android.com/guide/topics/connectivity/bluetooth-le>

sau

<https://github.com/googlearchive/android-BluetoothLeGatt>

In plus:

-          Trateaza si paired devices:

Set<BluetoothDevice> pairedDevices = **mBluetoothAdapter**.getBondedDevices();  
**for**(BluetoothDevice result: pairedDevices) {  
    **if**(result.getAddress().equals(**"00:A0:50:1A:D6:A3"**)) {  
        **bluetoothDevice**= result;  
        **deviceAddress**.setText(**bluetoothDevice**.getAddress());  
        **deviceName**.setText(**bluetoothDevice**.getName());  
        **progressBar**.setVisibility(View.***INVISIBLE***);  
        **return**;  
    }  
}

-          Si mai trateaza si cazul cand nu ai activat BLE, in onResume, de ex

*// Ensures Bluetooth is available on the device and it is enabled. If not,  
// displays a dialog requesting user permission to enable Bluetooth.***if** (**mBluetoothAdapter** == **null** || !**mBluetoothAdapter**.isEnabled()) {  
    Intent enableBtIntent = **new** Intent(BluetoothAdapter.***ACTION\_REQUEST\_ENABLE***);  
    startActivityForResult(enableBtIntent, ***REQUEST\_ENABLE\_BT***);  
}

Ar fi important sa te conectezi la notificare si sa iti vina date de acolo.

Si dupa aia le interpretam.

Iti trimit un device tie si eu mai am altele pe care lucrez

public static string SERVICE\_BATTERY\_UUID = "0000180f-0000-1000-8000-00805f9b34fb";

        public static string CHARACTERISTIC\_BATTERY\_LEVEL\_UUID = "00002a19-0000-1000-8000-00805f9b34fb"; //READ & NOTIFY

        public static string SERVICE\_SENSORS\_UUID = "26d2a5e0-7b2c-11e6-8b77-86f30ca893d3";

        public static string CHARACTERISTIC\_BME\_DATA\_UUID = "26d2a5e1-7b2c-11e6-8b77-86f30ca893d3"; //READ

        public static string CHARACTERISTIC\_PMS\_DATA\_UUID = "26d2a5e2-7b2c-11e6-8b77-86f30ca893d3"; //READ

        public static string CHARACTERISTIC\_SPEC\_DATA\_UUID = "26d2a5e3-7b2c-11e6-8b77-86f30ca893d3"; //READ

        public static string CHARACTERISTIC\_SENSORS\_DATA\_UUID = "26d2a5e4-7b2c-11e6-8b77-86f30ca893d3"; //NOTIFY

        public static string CHARACTERISTIC\_DEVICE\_CONFIG\_UUID = "26d2a5e5-7b2c-11e6-8b77-86f30ca893d3"; //WRITE

        public static string CHARACTERISTIC\_TIMESTAMP\_UUID = "26d2a5e6-7b2c-11e6-8b77-86f30ca893d3"; //READ

        public static string CHARACTERISTIC\_DEVICE\_INFO\_UUID = "26d2a5e7-7b2c-11e6-8b77-86f30ca893d3"; //READ

astea cunt codurile si ar tb sa urmaresti notificarile CHARACTERISTIC\_SENSORS\_DATA\_UUID

Ai mai jos si structura pachetelor

           if (receivedBytes.Length == 17) /\*\* SPEC data \*/

            {

                measurementsList.Add(new RawMeasurement { UserToken = user, DeviceId = did, Timestamp = HistoricalDataTimestamp[0], IsOld = true, ParamId = ParamIds.CarbonMonoxide, RawValue = FunctionUtils.ConvertBytesToInt(**receivedBytes, 1, 4)** });

                measurementsList.Add(new RawMeasurement { UserToken = user, DeviceId = did, Timestamp = HistoricalDataTimestamp[0], IsOld = true, ParamId = ParamIds.NitrogenDioxide, RawValue = FunctionUtils.ConvertBytesToInt(**receivedBytes, 5, 4)** });

                measurementsList.Add(new RawMeasurement { UserToken = user, DeviceId = did, Timestamp = HistoricalDataTimestamp[0], IsOld = true, ParamId = ParamIds.SulphurDioxide, RawValue = FunctionUtils.ConvertBytesToInt(**receivedBytes, 9, 4)** });

                measurementsList.Add(new RawMeasurement { UserToken = user, DeviceId = did, Timestamp = HistoricalDataTimestamp[0], IsOld = true, ParamId = ParamIds.Ozone, RawValue = FunctionUtils.ConvertBytesToInt(**receivedBytes, 13, 4**) });

                HistoricalDataTimestamp[0] = HistoricalDataTimestamp[0].AddSeconds(-64);

            }

            else if (receivedBytes.Length == 18) /\*\* BME and PMS data\*/

            {

                measurementsList.Add(new RawMeasurement { UserToken = user, DeviceId = did, Timestamp = HistoricalDataTimestamp[1], IsOld = true, ParamId = ParamIds.ParticlesPM1, RawValue = FunctionUtils.ConvertBytesToInt**(receivedBytes, 1, 2)** });

                measurementsList.Add(new RawMeasurement { UserToken = user, DeviceId = did, Timestamp = HistoricalDataTimestamp[1], IsOld = true, ParamId = ParamIds.ParticlesPM25, RawValue = FunctionUtils.ConvertBytesToInt**(receivedBytes, 3, 2)** });

                measurementsList.Add(new RawMeasurement { UserToken = user, DeviceId = did, Timestamp = HistoricalDataTimestamp[1], IsOld = true, ParamId = ParamIds.ParticlesPM10, RawValue = FunctionUtils.ConvertBytesToInt**(receivedBytes, 5, 2)** });

                measurementsList.Add(new RawMeasurement { UserToken = user, DeviceId = did, Timestamp = HistoricalDataTimestamp[1], IsOld = true, ParamId = ParamIds.Temperature, RawValue = FunctionUtils.ConvertBytesToInt**(receivedBytes, 8, 2)** });

                measurementsList.Add(new RawMeasurement { UserToken = user, DeviceId = did, Timestamp = HistoricalDataTimestamp[1], IsOld = true, ParamId = ParamIds.Humidity, RawValue = FunctionUtils.ConvertBytesToInt**(receivedBytes, 10, 2)** });

                measurementsList.Add(new RawMeasurement { UserToken = user, DeviceId = did, Timestamp = HistoricalDataTimestamp[1], IsOld = true, ParamId = ParamIds.Pressure, RawValue = FunctionUtils.ConvertBytesToInt**(receivedBytes, 12, 4)** });

                measurementsList.Add(new RawMeasurement { UserToken = user, DeviceId = did, Timestamp = HistoricalDataTimestamp[1], IsOld = true, ParamId = ParamIds.IndoorAirQualityIndex, RawValue = FunctionUtils.ConvertBytesToInt**(receivedBytes, 16, 2)** });

                HistoricalDataTimestamp[1] = HistoricalDataTimestamp[1].AddSeconds(-64);

            }

Ok, am reusit sa citesc ceva

Ai cateva greseli:

-        Testezi masca de biti cu & nu cu |

**connectService**.setOnClickListener(**new**View.OnClickListener() {  
    @Override  
    **public void**onClick(View v) {  
  
        **for**(BluetoothGattCharacteristic mNotifyCharacteristic: **mNotifyCharacteristics**) {  
  
            **if**(mNotifyCharacteristic != **null**) {  
                **final int**charaProp = mNotifyCharacteristic.getProperties();  
                **if**((charaPro**p & B**luetoothGattCharacteristic.***PROPERTY\_READ***) > 0) {  
                    **mBluetoothLEService**.readCharacteristic(mNotifyCharacteristic);  
                }  
                **if**((charaPro**p & B**luetoothGattCharacteristic.***PROPERTY\_NOTIFY***) > 0) {  
                    **mBluetoothLEService**.setCharacteristicNotification(mNotifyCharacteristic, **true**);  
                }  
            }  
        }  
    }  
});

-        Apoi cand faci

serviceString = SampleGattAttributes.*lookup*(uuid);

sau

charaString = SampleGattAttributes.*lookup*(uuid);

poti obtine si null, pt cazul serviciilor si caracteristicilor ce nu sunt standard, cum  e cazul dispozitivului nostru

tu obtii ceva != null doar pentru batteryservice, ce e un serviciu standard ble

trebuie sa tratezi si cazurile cand valorile respective sunt nule