
*Relazione dell'attività di progetto
del corso di Sensori a.a. 2022/2023*

*Dalini Gianpaolo 1045938
Maffeis Isaac 1041473*

*Weather station
(BLE Central and Peripheral)*

Obiettivo:

- Elaborare un programma in Arduino per la gestione di un sensore ambientale, con campionamento di valori relativi alla temperatura, umidità, pressione atmosferica e altitudine;
- Integrare la connettività bluetooth low energy al sensore di ambiente;
- Collezionare i dati campionati su MATLAB.

Componenti:

- 2x Arduino Nano 33 BLE.

Software

- Arduino IDE con installato:
 - Libreria "Nano33BLESensor";
 - Libreria ArduinoBLE;
 - Firmware [DEPRECATED – Please install standalone packages] Arduino Mbed OS Boards v.1.1.6;
- MATLAB;
- Applicazione smartphone per scanner BLE (consigliata nRF Connect).

Cenni Teorici:

La scheda "Arduino Nano 33 BLE" ha già integrato il sensore di temperatura e di pressione che servono per il campionamento dei valori, pertanto non sono necessari componenti aggiuntivi, la comunicazione con tali sensori avviene tramite protocollo I2C e nello specifico sono:

- **Sensore di temperatura e umidità HTS221**
i range dei diversi valori del sensore sono i seguenti:
 - Precisione della temperatura: $\pm 0,5\text{ }^{\circ}\text{C}$, da $15\text{ a }+40\text{ }^{\circ}\text{C}$
 - Intervallo di temperatura: da $-40\text{ a }120^{\circ}\text{C}$
 - Precisione umidità: $\pm 3,5\%\text{ rH}$, da $20\text{ a }+80\%\text{ rH}$
 - Intervallo di umidità: da $0\text{ a }100\%$
- **Sensore barometrico LPS22HB**
 - Precisione relativa rispetto alla pressione: $P = 80 - 110\text{ KPa}$ $T = 25\text{ }^{\circ}\text{C} \pm 0,1\text{ hPa}$
 - Intervallo di pressione: da $26\text{ a }126\text{ KPa}$

Il sensore barometrico contiene un diaframma formato da una piastra resistiva a contatto con l'atmosfera, per effetto della forza esercitata dalla pressione risultante la membrana si deforma e in

base alla quantità della deformazione viene rilevata la pressione atmosferica. Maggiore è la pressione, più il diaframma si muove, il che si traduce in una lettura del barometro più elevata. I valori del sensore della pressione atmosferica sono in kPa (unità di misura), grazie a questi si può calcolare l'altitudine in metri per mezzo della formula matematica $H = 44330 * [1 - (P/p_0)^{(1/5.255)}]$ dove "H" sta per altitudine, "P" la pressione misurata (kPa) dal sensore e "p0" è la pressione di riferimento a livello del mare (101.325 kPa).

L'interazione con questi sensori integrati avviene per mezzo della libreria Nano33BLESensor, che mette a disposizione oggetti e metodi per semplificarne la gestione. I valori dei sensori di temperatura, umidità e pressione vengono quindi campionati, viene calcolata l'altitudine, un buffer stringa dovrà quindi contenere questi dati che vengono infine mostrati su monitor seriale.

Una segnalazione luminosa viene integrata per permettere di capire quando la scheda effettua una misura.

La **connettività bluetooth** si basa sullo standard BLE (Bluetooth Low Energy) e ha lo scopo di fornire un consumo energetico e un costo notevolmente ridotto, mantenendo un intervallo di comunicazione simile al bluetooth standard.

I dispositivi connessi tramite Bluetooth per comunicare svolgono i ruoli di centrale e periferico, in cui il primo esegue una scansione e ascolta qualsiasi dispositivo in trasmissione e il secondo trasmette informazioni su di esso a qualsiasi dispositivo vicino, una volta stabilita una connessione il dispositivo centrale interagirà con le informazioni disponibili di cui dispone il dispositivo periferico utilizzando i servizi (elenco di caratteristiche con un codice identificativo univoco UUID).

Per integrare la connettività bluetooth vengono quindi utilizzati 2 Arduino Nano 33 BLE, uno che ricopre la funzionalità di centrale, uno quella di periferica e la libreria ArduinoBLE. Il dispositivo periferico campiona i dati ambientali dai propri sensori e li invia al dispositivo centrale che si occupa di collezionarli adeguatamente con il supporto di uno script di MATLAB.

Il dispositivo periferico offre il servizio "Environmental Sensing" con UUID 0x181A che mette a disposizione le caratteristiche di temperatura (temperatureCharacteristic con UUID 0x2A6E), umidità (humidityCharacteristic con UUID 0x2A6F), pressione (pressureCharacteristic con UUID 0x2901), altitudine (elevationCharacteristic con UUID 0x2A6C) e un buffer globale globalBLE con UUID 0x2A25, tutte con il proprio descrittore.

Analisi del codice semplificato:

➤ Arduino Periferico:

Per semplificarne la lettura ci si focalizza solamente sulla rilevazione e trasmissione della temperatura.

Nella parte iniziale vengono importate le librerie e definiti alcuni parametri della comunicazione ble, tra cui il servizio **environmentalSensingService** e la caratteristica offerta

temperatureCharacteristic con proprietà di lettura e di notifica. **temperatureDescriptor** è il descrittore che verrà allegato alla caratteristica di temperatura che ha il compito di descrivere il valore e l'unità di misura. Il primo parametro di questi oggetti è il codice identificativo univoco UUID, è importante scegliere il valore esadecimale corretto come descritto in precedenza, si può fare riferimento alla seguente libreria github per avere un elenco completo di tutti i codici

<https://github.com/noble/noble/tree/master/lib>.

```
#include "Arduino.h"
#include <ArduinoBLE.h>
#include "Nano33BLETemperature.h"

#define BLE_DEVICE_NAME    "Arduino Nano 33 BLE 1"    // Nome dispositivo
```

```

#define BLE_LOCAL_NAME    "Stazione Meteo (BLE) 1" // Nome Locale

Nano33BLETemperatureData temperatureData; //oggetto di tipoNano33BLETemperatureData
// Caratteristiche BLE
BLEService environmentalSensingService("181A"); // UUID 181A Environmental Sensing
// formato BLECharacteristic(uuid, properties, stringValue)
BLEShortCharacteristic temperatureCharacteristic("2A6E", BLERead | BLENotify);
// Temperature UUID 2A6E
BLEDescriptor temperatureDescriptor("2901", "Temperature mesure [*C]");
// Characteristic User Description UUID 2901

```

Nella parte di setup eseguita solamente una volta all'avvio del codice viene inizializzato il sensore della temperatura e la comunicazione bluetooth low energy.

```

void setup(){
    Temperature.begin(); // setup sensore temperatura

    if (!BLE.begin()){ // Setup BLE
        //Errore - Procedura di setup del modulo Bluetooth Low Energy (BLE) fallita
        while (1);
    }
    else{
        BLE.setDeviceName(BLE_DEVICE_NAME);
        BLE.setLocalName(BLE_LOCAL_NAME);
        BLE.setAdvertisedService(environmentalSensingService);
        environmentalSensingService.addCharacteristic(temperatureCharacteristic);
        temperatureCharacteristic.addDescriptor(temperatureDescriptor);
        BLE.addService(environmentalSensingService);
        BLE.advertise();
    }
}

```

La parte del ciclo loop viene eseguita in ripetizione e inizialmente si occupa di ricercare un dispositivo centrale a cui connettersi, una volta trovato viene campionato il valore di temperatura dal sensore e inserito all'interno della caratteristica `temperatureCharacteristic` grazie alla funzione `writeValue()`. Il formato del variabile volta ad immagazzinare il valore di temperatura è sint16 ovvero signed short.

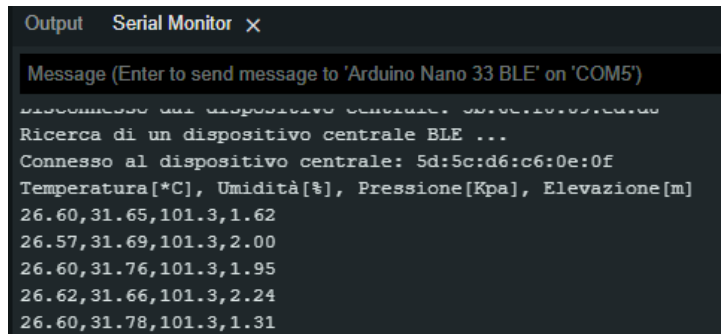
```

void loop(){
    // centrale è un dispositivo che ricerca i dispositivi Bluetooth per connettersi
    BLEDevice central = BLE.central();
    //ascolta le periferiche Bluetooth Low Energy da connettere:
    if(central){ // se una centrale è collegata alla periferica:
        // Quando si connette un dispositivo si esegue il campionamento dei dati
        while(central.connected()){ // mentre il centrale è ancora connesso
            if(Temperature.pop(temperatureData)){
                // se è presente un valore del sensore di temperatura
                // BLE defines Temperature UUID 2A6E Type sint16
                // Il valore è in gradi Celsius con una sensibilità di 0.01
                int16_t temperature = round( temperatureData.temperatureCelsius * 100.0 );
                temperatureCharacteristic.writeValue(temperature);
                // scrive il valore della caratteristica
            }
        }
    }
}

```

Osservazioni:

Questo codice ha lo scopo di campionare i dati ambientali tramite i sensori di temperatura e pressione, scriverli all'interno delle corrette caratteristiche del servizio offerto ed occuparsi del ruolo di dispositivo periferico di una comunicazione bluetooth, pubblicizzando se stesso nella rete cercando un dispositivo centrale.



```
Output  Serial Monitor x
Message (Enter to send message to 'Arduino Nano 33 BLE' on 'COM5')
Disconnesso dal dispositivo centrale: 5d:5c:d6:c6:0e:0f
Ricerca di un dispositivo centrale BLE ...
Connesso al dispositivo centrale: 5d:5c:d6:c6:0e:0f
Temperatura[*C], Umidità[%], Pressione[Kpa], Elevazione[m]
26.60,31.65,101.3,1.62
26.57,31.69,101.3,2.00
26.60,31.76,101.3,1.95
26.62,31.66,101.3,2.24
26.60,31.78,101.3,1.31
```

Figura 1: Stampa su monitor seriale della stringa buffer contenente i Valori di Temperatura [°C], Umidità [%], Pressione atmosferica [Kpa] e Altitudine [m] di un singolo campionamento per riga.

Questo dispositivo periferico può essere alimentato tramite una power bank così da avere una maggiore libertà nel calcolo dei dati con misure dell'ambiente esterno.

➤ Arduino Centrale:

Per semplificare la lettura del codice ci si focalizza solamente sulla ricezione del buffer globale e della sua scrittura sulla porta seriale.

Il buffer `bleBuffer` è costituito da un massimo di 27 byte ed è composto dai dati in serie di temperatura, umidità, pressione ed elevazione.

Il servizio a cui si è interessati è l' Environmental Sensing del periferico UUID=0x181A.

```
#define BLE_BUFFER_SIZES 27 // massimo 27 byte in un pacchetto BLE
const char* deviceServiceUuid = "181A"; // UUID 181A Environmental Sensing
char bleBuffer[BLE_BUFFER_SIZES]; // Buffer BLE per le caratteristiche offerte
// 6 byte max Temperatura , 6 byte max per Umidità, 5 byte max Pressione, 7 byte
max Altitudine = 24 + 3(',') = 27byte
```

E' importante far corrispondere il codice identificativo univoco della caratteristica `globalBLE` del dispositivo centrale con il rispettivo UUID di quella periferica.

Una volta definita la caratteristica da usare si controlla se contiene tutte le proprietà definite nel periferico e se il controllo da esito positivo ci si sottoscrive alle notifiche, successivamente si leggono i valori presenti all'interno della caratteristica e si scrivono nella porta seriale.

```
void loop() {
  // [...]
  BLECharacteristic globalBLE = peripheral.characteristic("2A25");

  if (!globalBLE) {
    Serial.println("* Peripheral device does not have globalBLE characteristic!");
    peripheral.disconnect();
    return;
  } else if (!globalBLE.canRead()) {
```

```

    Serial.println("* Peripheral does not have a readable globalBLE
characteristic!");
    peripheral.disconnect();
    return;
} else if (!globalBLE.canSubscribe()) {
    Serial.println("* globalBLE characteristic is not subscribable!");
    peripheral.disconnect();
    return;
} else if (!globalBLE.subscribe()) {
    Serial.println("* Subscription failed!");
    peripheral.disconnect();
    return;
}

while (peripheral.connected()) {
if (globalBLE.valueUpdated()) {
    globalBLE.readValue(bleBuffer, BLE_BUFFER_SIZES);
    Serial.println(bleBuffer); // stampa su monitor seriale il buffer
}}}

```

Osservazioni:

Questo codice ha lo scopo di occuparsi del ruolo di dispositivo centrale in una comunicazione bluetooth, stabilendo una comunicazione con il dispositivo periferico, legge poi i dati relativi al campionamento e li scrive sulla porta seriale.

Questo dispositivo centrale necessita di un collegamento al PC via USB in modo da poter collezionare i dati.

➤ MATLAB

```

s = serial('COM5','BAU',115200,'terminator','LF');
fopen(s);
str = fscanf(s);
expression = '\d*.\d*.\d*.\d*.\d*.\d*.\d*.\d*';

arr = nan(10,4);
data = fscanf(s,'%f, %f, %f, %f').';
prev = data;
arr(1,:) = data;
i=2;

while 1 % ciclo do-while
    data = fscanf(s,'%f, %f, %f, %f').'; % acquisisco i valori in continuazione
    if(max(data ~= prev)==1) % se c'è almeno un valore diverso salvo la lettura
        arr(i,:) = data;
        prev = data;
        i=i+1;
    end
    if(i>10) % salvo solo 10 letture
        break;
    end
end

fclose(s);
data = mean(arr); % una misura è la media di 10 letture

```

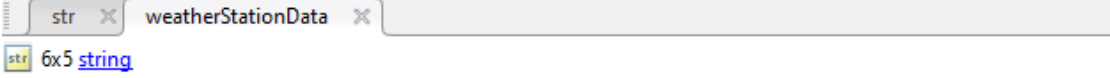
```
% weatherStationData_description = ["Time", "Temperatura [°C]", "Umidità [%]",
"Pressione [Kpa]", "Elevazione [m]"];
% weatherStationData = [weatherStationData_description;string(datetime),data];

% carica la matrice e aggiunge la misura
load('WeatherStationData.mat');
weatherStationData = [weatherStationData;string(datetime),data];

save("WeatherStationData.mat","weatherStationData");
```

Osservazioni:

Questo codice si occupa di leggere i dati dalla porta seriale e collezionarli in una struttura matriciale.



	1	2	3	4	5
1	Time	Temperatura [°C]	Umidità [%]	Pressione [Kpa]	Elevazione [m]
2	06-Feb-2023 23:25:26	26.398	33.299	100.93	31.501
3	06-Feb-2023 23:34:28	26.318	33.278	101	30.734
4	07-Feb-2023 00:00:56	26.451	32.164	101	26.135
5	07-Feb-2023 09:57:58	24.455	34.176	101.4	-3.36
6	07-Feb-2023 12:03:06	27.036	30.461	101.2	8.571

Figura 2: Tabella weatherStationData.mat

In particolare prima controlla se sulla porta seriale sono presenti i dati nel formato corretto (e non ad esempio messaggi relativi alla connessione ble), poi scrive i dati all'interno della matrice persistente weatherStationData facendo una media tra 10 campionamenti effettuando così un filtraggio.

Analisi dei dati

L'analisi dei dati è stata eseguita su MATLAB, in un primo momento raccogliendo i dati ed eseguendo una media dei valori per i 4 tipi di dati che andavamo ad analizzare:

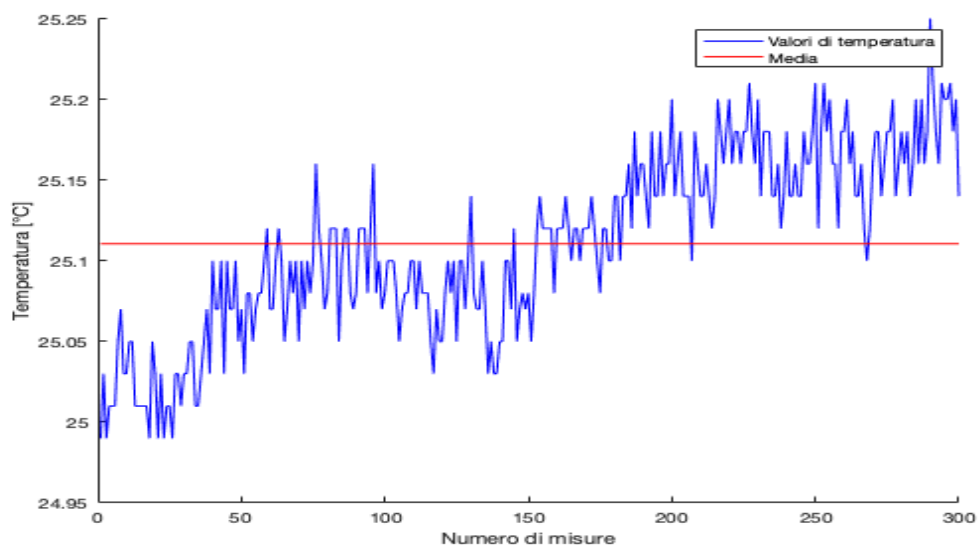


Figura 3: Valori di Temperatura

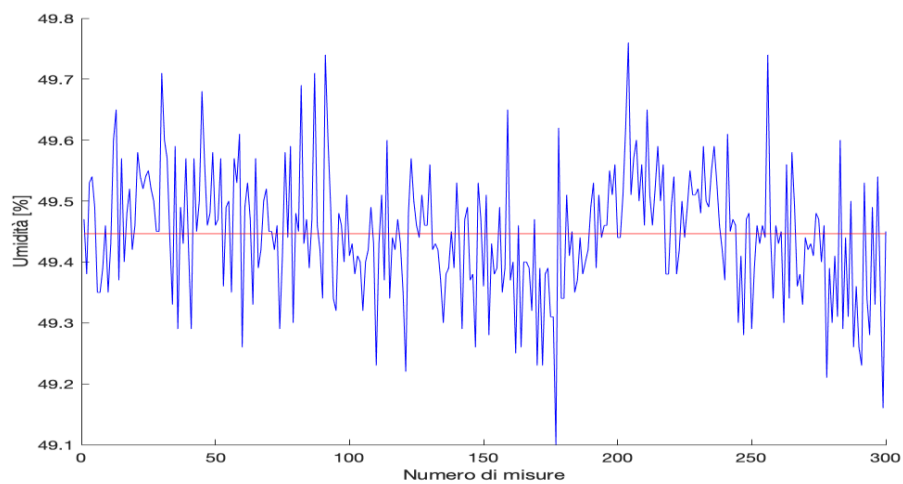


Figura 4: Valori di Umidità

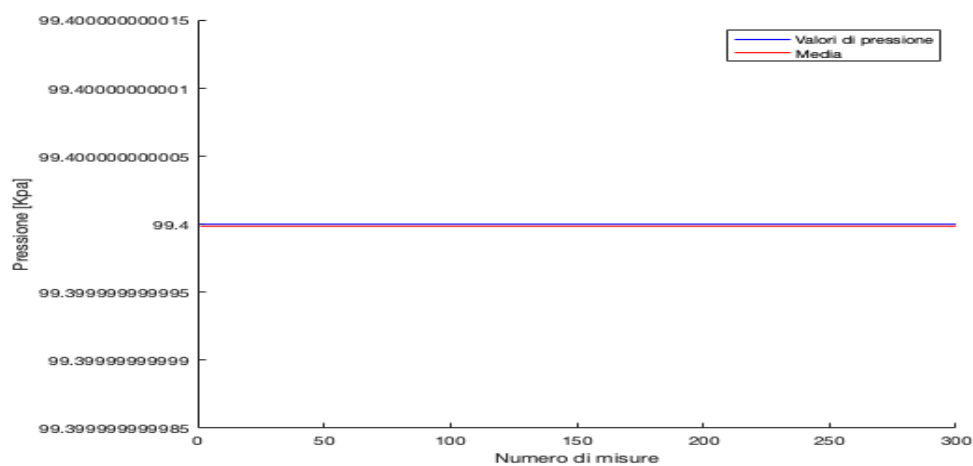


Figura 5: Valori di Pressione

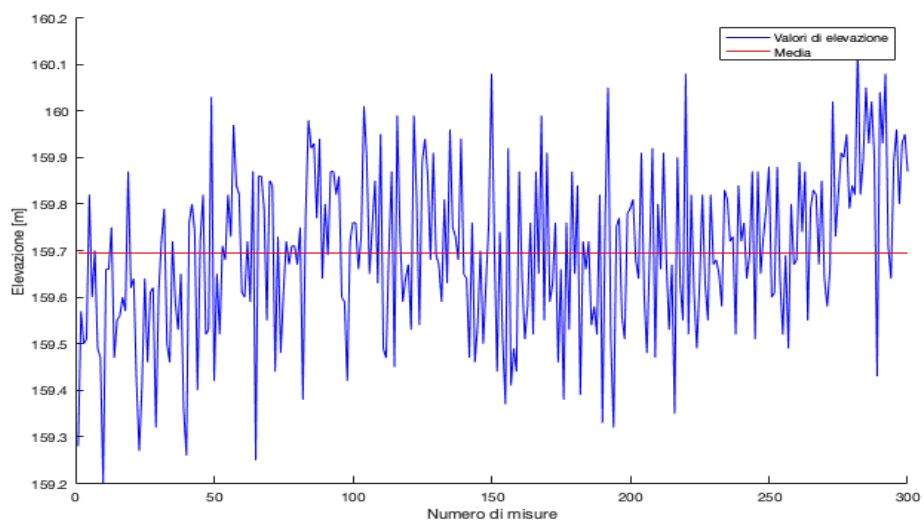


Figura 6: Valori di Elevazione

Ci siamo poi concentrati in un'analisi più approfondita per quanto riguardava i dati relativi alla temperatura, considerando anziché la media, la mediana la quale è meno influenzata da valori estremi che potrebbero portare a variare la media.

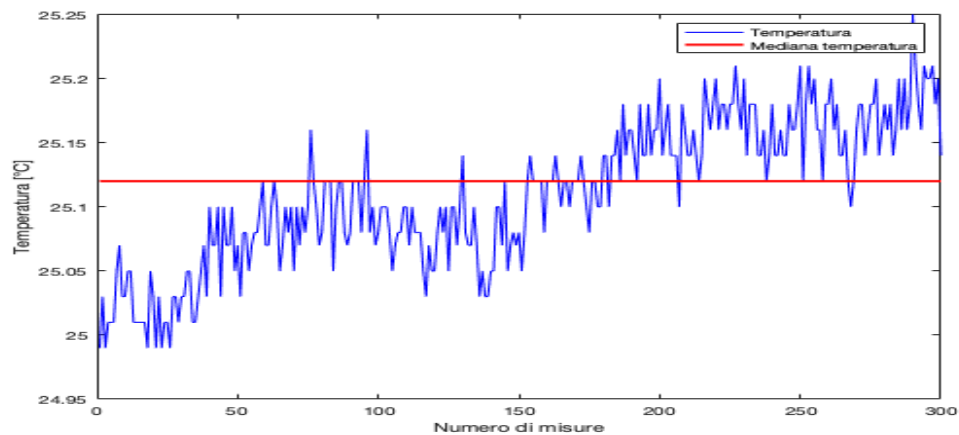


Figura 7: valori di temperatura e mediana

In seguito è stato applicato un filtraggio mediano come smoothing, il quale sostituisce i valori nella serie con la mediana dei valori circostanti per rimuovere possibili picchi, questo tipo di filtraggio è più robusto rispetto ad altri in quanto basandosi sulla mediana è appunto meno soggetto al risentire di picchi seppur mantenendo comunque i valori di picco in modo più preciso rispetto ad un filtraggio tramite media

Segue il codice relativo allo smoothing:

```
window_size = 10; % dimensione della finestra di filtraggio

filtered_data = zeros(1, length(temperatures)); % array per i dati filtrati

for i = 1:length(temperatures)
    start = max(1, i - (window_size - 1) / 2); % inizio della finestra di filtraggio
    stop = min(length(temperatures), i + (window_size - 1) / 2); % fine della finestra di filtraggio
    window = temperatures(start:stop); % finestra di filtraggio
    filtered_data(i) = median(window); % valore mediano della finestra
end
```

La finestra di 10 valori è stata scelta dopo una serie di tentativi osservando il comportamento del sensore in relazione a dei disturbi, con questa finestra di valori siamo in grado di migliorare la qualità del segnale conservando comunque una buona fedeltà rispetto al segnale originale

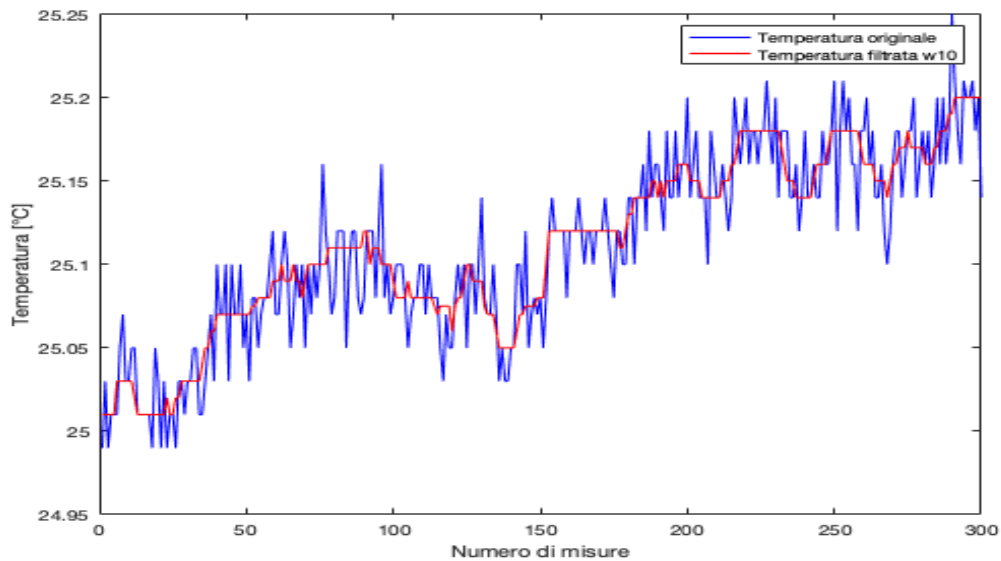


Figura 8: Valori di temperatura con smoothing con finestra w10

Si può quindi osservare un segnale più pulito con molti meno picchi che però mantiene il profilo del segnale non filtrato

Esperimenti:

Per decidere quale tipo di filtraggio utilizzare e che finestra di campionatura considerare sono stati eseguiti degli esperimenti nei quali si andava a perturbare il segnale applicando una fonte di calore al sensore durante l'acquisizione dei valori

Per non rischiare l'integrità del sensore si è deciso di semplicemente soffiare sul sensore in quanto l'aria più calda proveniente dal corpo umano aveva una temperatura più elevata di quella dell'ambiente circostante, abbastanza elevata per avere dei picchi ma non abbastanza per causare danni

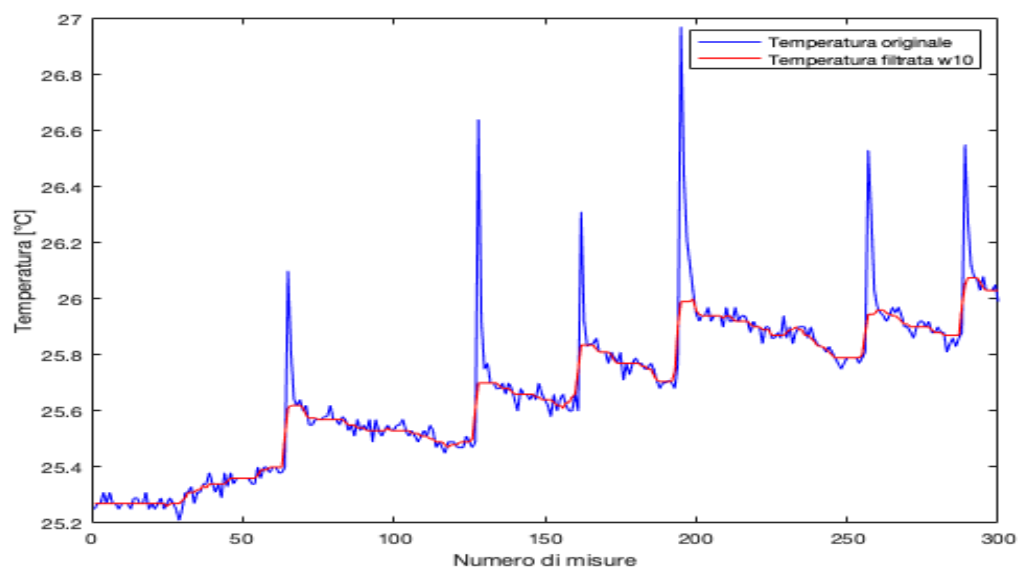


Figura 9: Valori di temperatura con perturbazione esterna

Lettura valori smartphone:

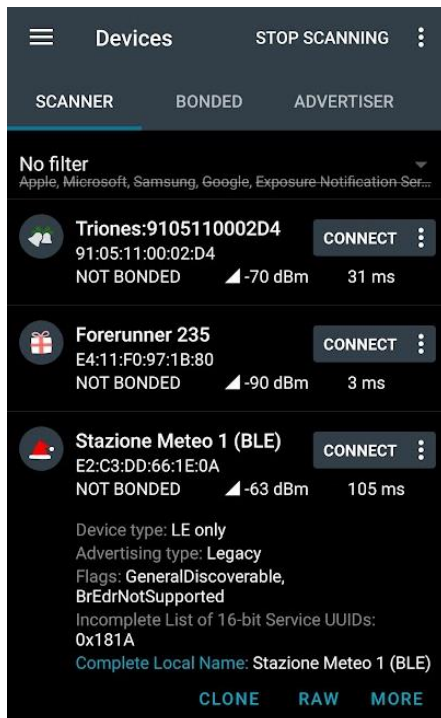


Figura 10: Screenshot schermata iniziale dell'app nRF Connect

Per poter leggere i valori rilevati dalla stazione meteo su smartphone è necessaria un'applicazione che consenta la comunicazione con un dispositivo BLE, in questo caso è stata utilizzata l'app nRF Connect, dopo averla aperta ed avviato la ricerca dei dispositivi BLE ci si può connettere all'Arduino Nano 33 BLE selezionando il dispositivo "Stazione Meteo 1 (BLE)", ossia l'Arduino periferico.

E' possibile capire quando la scheda periferica viene e resta connessa ad un dispositivo centrale BLE (in questo caso lo smartphone) per via di una segnalazione luminosa del led arancione che lampeggia per 200ms ogni 2s circa.

Una volta effettuata la connessione i valori di interesse si trovano nella sezione del servizio "Environmental Sensing".

Per ogni singola caratteristica è possibile leggerne il nome, l'identificatore UUID, le proprietà definite, il valore e il descrittore.

Tramite la freccia verso il basso è possibile leggere il valore della caratteristica, mentre con le 3 frecce è possibile sottoscrivere al canale per una lettura continua (è possibile solo se è presente la proprietà NOTIFY).

Di seguito sono riportate le letture delle 5 caratteristiche implementate.

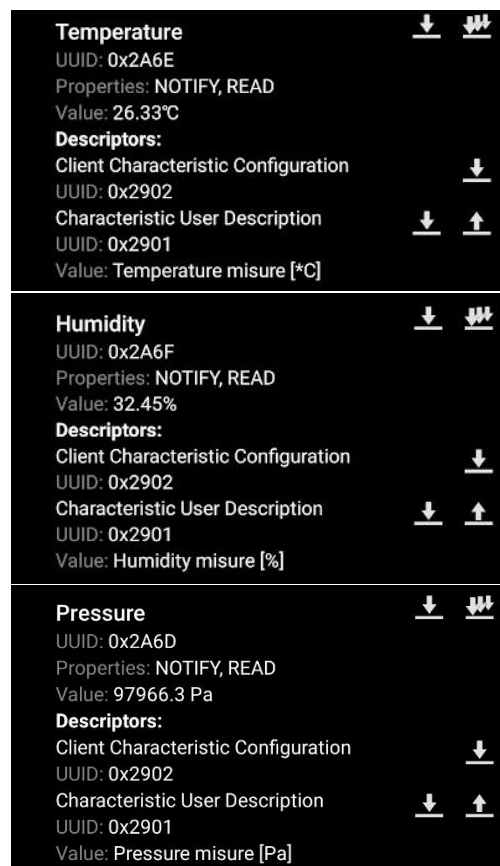
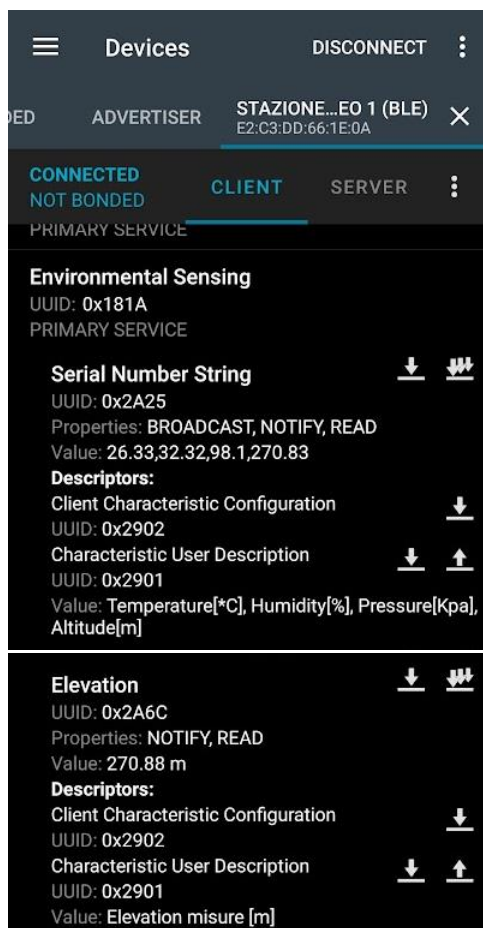


Figura 11: Screenshot delle caratteristiche del servizio Environmental Sensing della Stazione Meteo 1 (BLE) su app nRF Connect

Il valore del buffer globale è una stringa, mentre tutte le altre caratteristiche presentano un tipo specifico che è stato correttamente implementato nel codice, in particolare è stato reso possibile utilizzare il formato con le giuste cifre decimali e l'unità di misura utilizzando lo specifico UUID standard BLE per ogni caratteristica.

Problemi riscontrati:

- Problema nella comunicazione bluetooth low energy con windows 11

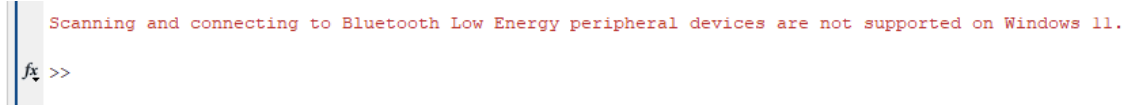


Figura 12: Screenshot messaggio di errore del terminale di MATLAB

(messaggio di errore di MATLAB a seguito del comando >> blelist per cercare i dispositivi ble). Questa problematica è stata risolta scegliendo di utilizzare 2 Arduino Nano 33 BLE nel ruolo di central e peripheral, comunicando con matlab via porta seriale con il central.

- Necessaria una calibrazione, i valori letti dai sensori non sono corretti e variano a seconda della scheda Arduino utilizzata, risulta necessaria una calibrazione.