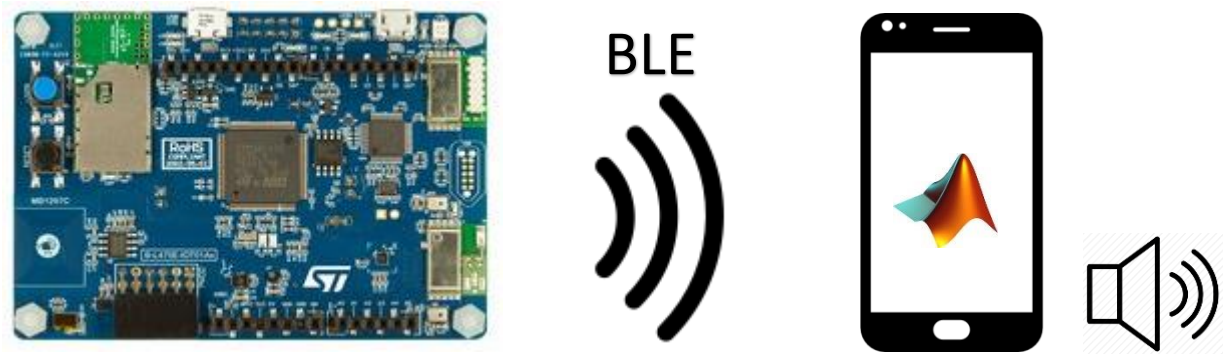
BLE – STM32 & Android (IOS)

BLE (Bluetooth Low Energy) è una tecnologia che permette di inviare dati tra due dispositivi bluetooth.

L'obiettivo è ottenere le misurazioni dei sensori di prossimità e di intensità magnetica dalla scheda STM32 e inviare a intervalli regolari tali informazioni al proprio telefono Android (o IOS) in modo da generare un suono a frequenza e intensità variabile, in base al valore dei sensori.



Set up codice bluetooth su mbed.com

Importare l'esempio https://os.mbed.com/teams/ST/code/mbed-os-example-ble-HeartRate/ nel compilatore online specificando come nome del progetto "iaas_ble_range" o altro e "Program" come tipologia di importazione.

Nel file source/main.cpp modificare la riga 25 scegliendo un nome per il servizio di invio dati. Ad esempio da

```
const static char     DEVICE_NAME[] = "HRM";
```

a

```
const static char     DEVICE_NAME[] = "IAS_12345";
```

È consigliabile fornire un nome univoco a ogni dispositivo, altrimenti saranno visibili 16 device Bluetooth con lo stesso nome.

Una volta compilato e caricato sulla scheda, si può verificare il funzionamento utilizzando l'app nrfConnect

Android: https://play.google.com/store/apps/details?id=no.nordicsemi.android.mcp

IOS: https://apps.apple.com/it/app/nrf-connect/id1054362403

Nell'elenco dei dispositivi bluetooth connettersi al proprio dispositivo scegliendo quello con il nome impostato nel passaggio precedente.

Nella scheda del dispositivo aprire il servizio Heart Rate e iniziare la misurazione premendo il bottone  .

I dati ricevuti sono una sequenza di interi da 100 a 174 con un intervallo di 500 ms tra uno e l'altro.

Aggiunta lettura sensore di prossimità

Aprire l'esempio HelloWorld_ST_sensors e copiare la cartella VL53L0X nella cartella source del progetto iaas_ble_range.

In questo modo i file di libreria per la lettura del sensore di prossimità sono disponibili.

Nel file source/main.cpp aggiungere dopo le altre include (riga 22-23):

```
#include "VL53L0X.h"
static DevI2C devI2c(PB_11,PB_10);
static DigitalOut shutdown_pin(PC_6);
static VL53L0X range(&devI2c, &shutdown_pin, PC_7);
```

E successivamente nella prima riga della funzione main (riga 124):

```
range.init_sensor(VL53L0X_DEFAULT_ADDRESS);
```

Infine modificare la funzione updateSensorValue (riga 43) da:

```
void updateSensorValue() {
    // Do blocking calls or whatever is necessary for sensor polling.
    // In our case, we simply update the HRM measurement.
    hrmCounter++;

    //  100 <= HRM bps <=175
    if (hrmCounter == 175) {
        hrmCounter = 100;
    }

    hrServicePtr->updateHeartRate(hrmCounter);
}
```

a:

```
void updateSensorValue() {
    uint32_t distance;
    uint16_t data;
    int status = range.get_distance(&distance);
    if (status == VL53L0X_ERROR_NONE) {
        data = (uint16_t)distance;
    } else {
        data = 0;
    }
    hrServicePtr->updateHeartRate(data);
}
```

Compilare e verificare tramite nrfConnect che i dati ricevuti siano conformi alla distanza.

Si può anche aggiungere una printf per inviare i dati alla seriale in modo da verificare in tempo reale che la distanza è effettivamente calcolata.

Invio di 2 valori distinti

Lo spazio a disposizione è di 16 bit, quindi è possibile sfruttare 8 bit per l'invio di un valore e 8 bit per l'invio di un altro contemporaneamente. Ad esempio utilizzare un byte per il valore della distanza e un byte per il valore di un altro sensore (ad esempio il magnetometro).

Per fare questo sono necessarie semplici operazioni binarie:

```
uint16_t data = valore1 & 255; // inserisce il primo valore in data, deve
                               // essere inferiore a 256 altrimenti viene eliso.
data = data | (valore2 << 8); // aggiunge il secondo valore a fianco.
```

Ossia dati due numeri, ad esempio: 44 e 172 (0010 1100, 1010 1100)

1) 44 & 255 === 0010 1100 & 1111 1111 === 0010 1100 === 44
2) 44 | (172 << 8) === 0010 1100 | 1010 1100 0000 0000 === 1010 1100 0010 1100 === 44076

44076 è il valore trasferito, e possono essere estratti i due valori dall'applicazione matlab/simulink dividendo i 16 bit in due byte distinti.

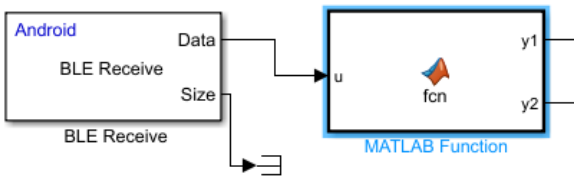
Lettura dati con Android(IOS)/Simulink

1) Creare un progetto Simulink per Android(IOS).
2) Scegliere dalla libreria il blocco BLE Receive dal pacchetto Android (IOS).
3) Configurare il blocco con il bottone "Scan" (tenendo lo smartphone collegato) seguendo le istruzioni. La configurazione carica una app sullo smartphone per individuare i dispositivi disponibili.
4) Scegliere come caratteristica "Heart Rate/Heart Rate Measurament" e data size pari a 3.

Automaticamente il blocco BLE Receive restituisce un array di 3 byte composto nel seguente modo

[flag, byte meno significativo, byte più significativo]

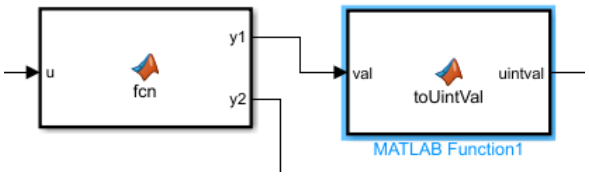
Quindi estraendo il secondo e il terzo elemento dell'array si ottengono i due valori inviati con il metodo indicato nel paragrafo "Invio di 2 valori distinti".




```
function [y1,y2] = fcn(u)

  y1 = u(2);
  y2 = u(3);
```

L'unico problema è dovuto al fatto che matlab interpreta ogni byte come intero a 8 bit con segno, quindi i valori inviati maggiori o uguali a 128 risultano negativi. Per ovviare alla cosa si può fare una semplice conversione.




```
function uintval = toUintVal(val)
  uintval = int16(bitand(val, int8(127)));
  if val < 0
     uintval = bitor(uintval, int16(128));
  end
```

Per verificare i valori si possono usare i blocchi Android Data Display.

I valori ricevuti possono esser utilizzati per generare un'onda sinusoidale alterando la frequenza attraverso i valori ottenuti.