LAB 5: WEATHER STATION

OVERVIEW

In this lab we will design a mini weather station which collects environmental data such as temperature, humidity, air pressure and device cardinal direction, and then sends on a mobile device via Bluetooth Low Energy. We will learn how to modularize and reuse the code we have developed in the previous labs and integrate them into a final application.

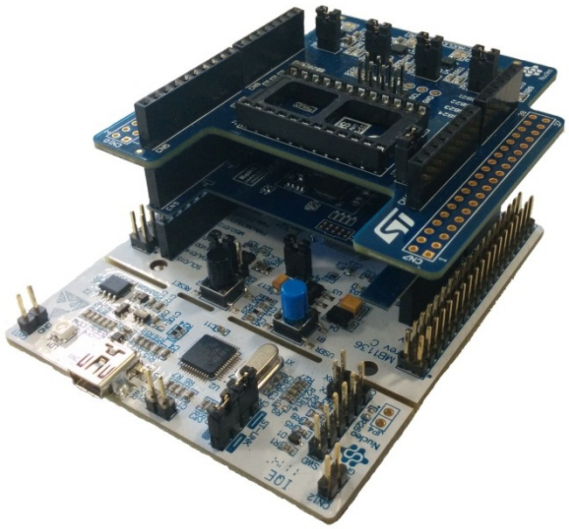


Figure.1 ST Nucle F401RE with X-NUCLEO-IDB04A and X-NUCLEO-IKS01A1 expansion shields

REQUIRED MATERIALS

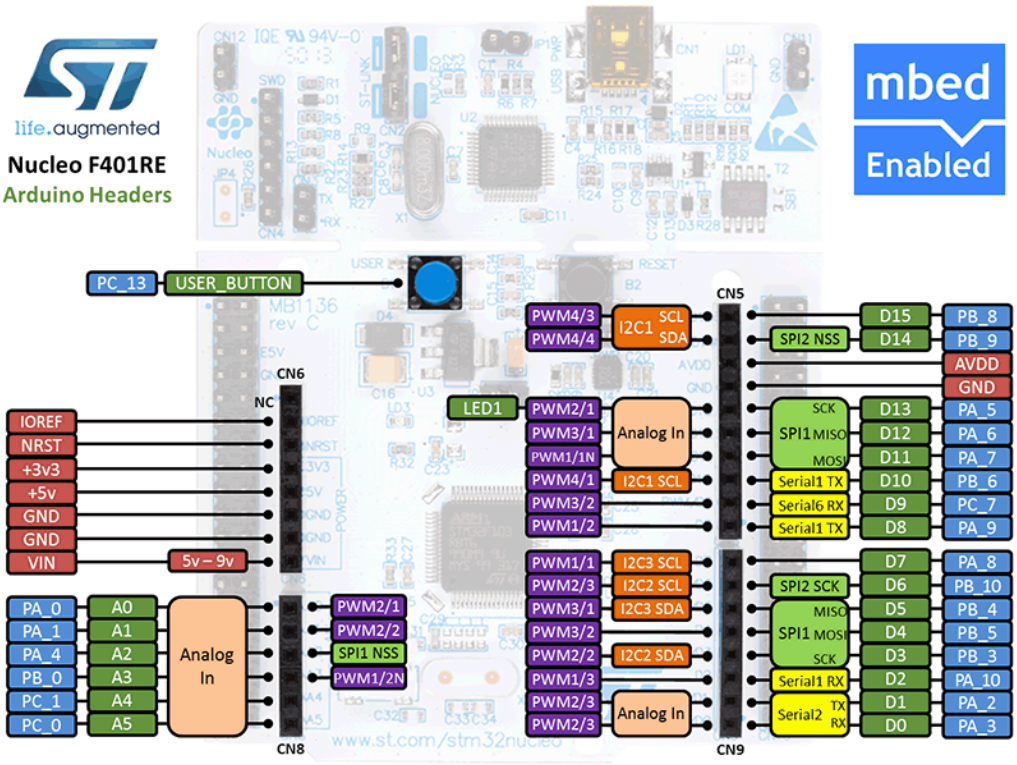
HARDWARE

All the hardware to be used in this lab has been introduced in the previous labs. This includes the Nucleo board, the BLE expansion board, the sensor board and a smartphone:

- An Android device with Bluetooth Low Energy support and Android 4.3 (API Level 18) or above.
  You can check a list of Bluetooth Smart Ready devices using the following link:
  http://www.bluetooth.com/Pages/Bluetooth-Smart-Devices-List.aspx
- The inertial and environmental sensors evaluation board X-NUCLEO-IKS01A1.
  http://www.st.com/web/catalog/tools/FM116/SC1248/PF261191
- The Bluetooth Low Energy evaluation board X-NUCLEO-IDB04A1.
  http://www.st.com/web/catalog/tools/FM116/SC1075/PF260517

NUCLEO F401RE BOARD

The Nucleo F401RE board pin descriptions are shown below:



Line description	Pin name in mbed API
LED	LED1/PA_5
I2C SCL	PB_8
I2C SDA	PB_9
SPI MOSI	PA_7
SPI MISO	PA_6
SPI SCLK	PB_3
SPI CS	PA_1
UART MOSI	USBTX/PA_2
UART MISO	USBRX/PA_3

PROGRAMMING EMBEDDED BLE WEATHER STATION

In this lab exercise we will program the microcontroller to interact with the sensors on the X-NUCLEO-IKS01A1 expansion board using I2C serial communication (as we did in the "Sensors Expansion Board" lab exercise), and then broadcast the data via BLE using the X-NUCLEO-IDB04A1 expansion board (similar to the "Bluetooth Low Energy Expansion Board" lab exercise).

We will receive and display the data (temperature, humidity, atmospheric pressure and cardinal direction) on the mobile device. Note that the cardinal direction can be used to indicate the direction of the wind, if we can attach the device to a wind vane (or a weathercock).

In order to make our device compatible to other BLE app we will use the standard Universally Unique Identifier (UUIDs) for the BLE services and characteristics. More information on the standard UUID can be found at: https://developer.bluetooth.org

The table below describes the details of the Environmental Sensing Service and the characteristics relevant to our application.

Name	UUID	Format	Description
Environmental Sensing (Service)	0x181A	-	The Environmental Sensing Service (ESS) exposes measurement data from an environmental sensor.
Humidity (Characteristic)	0x2A6F	uint16	Unit is in percent with a resolution of 0.01 percent.
Temperature(Characteristic)	0x2A6E	sint16	Unit is in degrees Celsius with a resolution of 0.01º.
Pressure(Characteristic)	0x2A6D	uint32	Unit is in pascals with a resolution of 0.1 Pa.
True Wind Direction(Characteristic)	0x2A71	uint16	Direction from which the wind originates. Angle measured clockwise relative to Geographic North. Unit is in degrees with a resolution of 0.01 degrees.

CREATE CUSTOMIZED SERVICES AND CHARACTERISTICS

In the previous heart rate monitor program we used predefined services that are part of the BLE_API. In this exercise we will create our own customized services and characteristics.

We first need to define the service and its functions in a separate header file as in the MBED_API. We recommend using BatteryService.h file as a template as it has a fairly simple notification service. The file can be found in the project folder > Nucleo_BLE_API > services.

We also need to declare our BLE device and characteristics as GattCharacteristic objects.

```
private:
    BLEDevice                   &ble;
    GattCharacteristic          my_characteristic;
```

Then let's create the service constructor and pass it to our characteristics and its attributes. These include UUID, first value, size of the packet in bytes, size of the whole message in bytes and notification properties.

```
my_characteristic (GattCharacteristic::UUID_HUMIDITY_CHAR,
                  (uint8_t *)&humidity, sizeof(uint16_t), sizeof(uint16_t),
                   GattCharacteristic::BLE_GATT_CHAR_PROPERTIES_NOTIFY),
```

After that, we need to add all the characteristics to a GattCharacteristic table and declare the GattService with parameters including UUID, the table of characteristics and the number of characteristics.

```
GattCharacteristic *charTable[] = {&my_characteristic1, &my_characteristic2};
GattService     my_service(GattService::UUID_SERVICE, charTable, No.characteristics));
```

Finally we need to add our service to the Bluetooth Low Energy device object.

```
ble.addService(my_service);
```

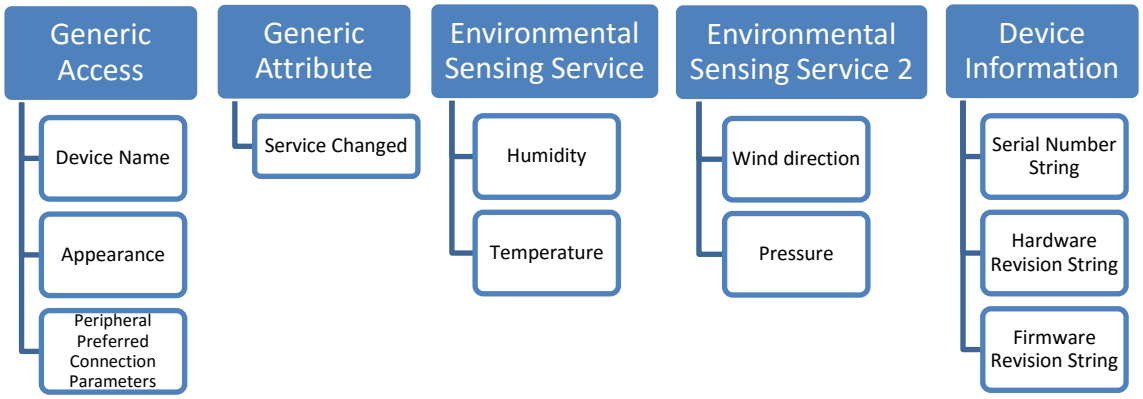
We also need to write some methods to interact with those characteristics easily. Each notification characteristic needs only one function to update the value. We use function updateCharacteristicValue to send the new value to the client.

```
ble.updateCharacteristicValue(handle, (uint8_t *) &variable, size of the variable);
```

PROGRAM STRUCTURE

The program has to configure the system as a discoverable BLE device in the same way we did for the heart rate monitor exercise. The program structure is going to be the same however the services and characteristics broadcasted by the device will be different.

The current release version of the BLE_API only allows us to have two notification characteristics in each service. Therefor to be able to subscribe all the four parameters we have to create two Environmental Sensing Services with two notification characteristics each. We will also add the Device Information Service, and optionally the Battery Service. The structure will be as follows. We can use a BLE analyser app to check that if the program structure is correct.



ANEX

The following graphs are approximate representations of the magnetometer sensors for the compensated values of axis X and Y.