

Corso di Fondamenti di Telematica

a.a. 2022-2023

OpenMote B Laboratory

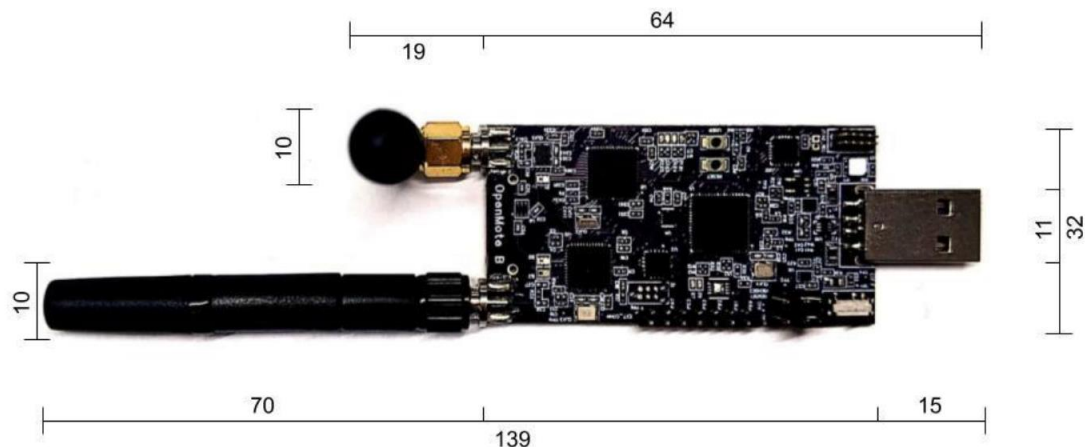
Docente: Prof. Giuseppe Piro

Assistenti: Dott. Antonio Petrosino, Dott. Giancarlo Sciddurlo

Un OpenMote B è un dispositivo di sviluppo hardware e prototipazione per Industrial Internet of Things (IIoT), in particolare per ricercatori e sviluppatori che lavorano su reti wireless di nuova generazione a lungo raggio e a bassa potenza e utilizza IPv6. Si basa sul microcontrollore Texas Instruments CC2538 ARM Cortex-M3 e presenta operazioni dual-band su bande 2.4 GHz e 868/915 MHz con supporto completo per IEEE 802.15.4.

In particolare, in questo laboratorio si utilizza il transceiver CC2538 con antenna trasmittente che opera nella banda 2.4 GHz ISM con supporto per IEEE 802.15.4-2006 con:

- Modulazione OQPSK;
- Data rate: 250 kbps;
- Receiver sensitivity: -97 dBm;
- Potenza di trasmissione: 7 dBm
- Corrente di trasmissione: 24 mA a 0 dBm;
- Receive current: 20 mA.



La scheda è dotata di:

- **Human interfacing:** la scheda principale include 4 LED (verde, giallo, arancione e rosso) e un bottone destinato a scopi di debug. Inoltre, include anche un bottone per il ripristino dell'hardware.

- **Serial communication:** la scheda principale può comunicare con un computer utilizzando una porta UART (Universal Asynchronous Receiver-Transmitter) sul CC2538. Inoltre, il chip FTDI consente di programmare direttamente il CC2538 utilizzando l'interfaccia interna con script Python.
- **Board expansion:** la scheda principale include una porta di espansione (8 pin con spaziatura di 2,54 mm) che può essere utilizzata per il debug o per collegare schede figlie (sensori).
- **Extended security:** la scheda principale include il supporto hardware per funzioni crittografiche SHA2, AES-128/256, ECC-128/256 e RSA.
- **Antenna connectors:** la scheda principale include due connettori per antenna SMA per antenne sub-GHz e 2,4 GHz. Il connettore dell'antenna Sub-GHz è direttamente collegato alla radio Sub-GHz sull'AT86RF215. L'antenna a 2,4 GHz è multiplexata utilizzando un interruttore RF per i ricetrasmittitori radio CC2558 e AT86RF215.
- **Power:** la scheda principale può essere alimentata da una porta USB (5V) o da due batterie AA (3V) poste sul retro. La scheda include un pulsante on/off per scollegare le due batterie AA quando non utilizzate.
- **Current sensing:** la scheda principale include due porte (3 pin con spaziatura di 2,54 mm) per misurare il consumo di corrente dell'intero sistema durante il funzionamento. La prima misura il consumo di corrente del chip CC2538, mentre la seconda misura il consumo di corrente del chip AT86RF215.

1. GCC Compiler per processore ARM embedded

Prerequisiti: Ubuntu 20.04 LTS oppure WSL con stessa versione.

Nello specifico per installare WSL, da riga di comando Power Shell (è preferibile sempre aprire il terminale come amministratore):

```
wsl -install
```

Una volta installato il sistema operativo, impostare le credenziali dell'utente (Username e Password).

Per poter aprire più schede contemporaneamente è possibile installare Anteprima Terminale come app di Windows.

Scaricare il pacchetto per abilitare l'interfaccia USB su WSL e installare l'msi:

https://github.com/dorssel/usbipd-win/releases/download/v2.4.1/usbipd-win_2.4.1.msi

Su WSL inserire i comandi per importare usbip:

```
:~$ sudo apt install linux-tools-5.4.0-77-generic hwdata
```

```
:~$ sudo update-alternatives --install /usr/local/bin/usbip usbip
/usr/lib/linux-tools/5.4.0-77-generic/usbip 20
```

Per listare tutte le USB connesse a Windows in PowerShell (amministratore) scrivere il comando:

```
usbipd wsl list
```

Prendere nota del Bus-ID in cui viene attaccato l'OpenMote B.

BUSID	VID:PID	DEVICE	STATE
1-1	0403:6010	USB Serial Converter A, USB Serial Converter B	Attached - Ubuntu
1-7	413c:301a	Dispositivo di input USB	Not attached
1-8	0bda:0129	Realtek USB 2.0 Card Reader	Not attached
1-9	413c:2113	Dispositivo di input USB	Not attached
1-14	0cf3:e005	Qualcomm QCA9565 Bluetooth 4.0	Not attached

Abilitare la porta USB relativa alla scheda:

```
usbipd wsl attach --busid <busid>
```

Su WSL è possibile controllare le USB collegate ad Ubuntu correttamente:

```
:~$ lsusb
```

Ogni volta che viene staccato il dispositivo dall'USB potrebbe essere necessario replicare questa procedura per l'abilitazione della porta USB su WSL.

Aggiornare il sistema operativo di Ubuntu:

```
:~$ sudo apt-get upgrade
:~$ sudo apt-get update
```

Installazione Tool Chain per compilare C/C++ su Ubuntu e componenti aggiuntivi addizionali per lavorare con il codice sorgente:

```
:~$ sudo apt-get install gcc-arm-none-eabi
:~$ sudo apt-get install build-essential git scons python3-pip
:~$ sudo pip3 install intelhex pyserial
```

2. Firmware OpenMote B

È possibile scaricare il firmware ufficiale (openmote-fw.tar.bz2) dell'OpenMote B dal sito <https://apps.industrialshields.com/main/openmote/> e passarlo su WSL o Ubuntu per poi modificarlo secondo le preferenze, oppure direttamente da Ubuntu scaricare con il comando:

```
:~$ git clone https://github.com/telematics-lab/openmote-fw
```

Da WSL per gestire l'interfaccia grafica di esplora risorse condivise con Windows per spostare i vari file, è possibile scrivere il comando:

```
:~$ explorer.exe .
```

3. Configurazione Trasmettitore

Spostarsi nella cartella contenente il firmware:

```
:~$ cd ./openmote-fw/projects/template/src/
```

Modificare il file main.cpp con l'editor di testo apribile con il comando (*per salvare CTRL+X*):

```
:~$ sudo nano main.cpp
```

In particolare, nella funzione **printSensor** per creare una trasmissione diversa per ogni trasmettitore che installiamo nella rete, caratterizzare i parametri evidenziati in rosso che rappresentano l'ID del trasmettitore e la frequenza di invio dei pacchetti:

```
static void printSensor(void *pvParameters) {

    Bme280Data bme280_data;
    SensorData sensorData;

    static RadioResult result;

    bool status;
    bool sent;
    size_t len = 0;

    /* Initialize the bme280 */
    bme280.init();

    while (true) {

        //NEW
        uint16_t tx_buffer_len;

        /* Check whether we can read data from bme280 */
        status = bme280.read(&bme280_data);

        if (status)
        {
            /* Fill-in sensor data */
            sensorData.temperature = (uint16_t) (bme280_data.temperature * 1.0f);
            sensorData.humidity = (uint16_t) (bme280_data.humidity * 1.0f);
            sensorData.pressure = (uint16_t) (bme280_data.pressure * 1.0f);

            led_green.on();
            len = sprintf((char*) uartBuffer,
                "ID\t%i\tTemperature\t%u\tHumidity\t%u\tPressure\t%u\t\r\n", 1, sensorData.temperature,
                sensorData.humidity, sensorData.pressure);
        }

        if (len > 0)
        {
            /* Write bytes to Serial */
            uart0.writeBytes(uartBuffer, len);
            len = 0;
        }

        led_green.on();
        //bme280.reset();
        Scheduler::delay_ms(50);

        // Take the txSemaphre, block until available
        if (txSemaphore.take())
        {
            // Turn on the radio transceiver
            radio.on();
        }
    }
}
```

```

        // Turn the yellow LED on when the packet is being loaded
        led_yellow.on();

        // Load the packet to the transmit buffer
        len = sprintf((char*) radio_buffer,
"ID\t%i\tTemperature\t%u\tHumidity\t%u\tPressure\t%u\t\r\n", 1, sensorData.temperature,
sensorData.humidity, sensorData.pressure);
        radio_ptr = radio_buffer;
        radio_len = sizeof(radio_buffer);
        radio_len = len;

        len = 0;

        result = radio.loadPacket(radio_ptr, radio_len);

        if (result == RadioResult_Success)
        {
            // Put the radio transceiver in transmit mode
            radio.transmit();
            Scheduler::delay_ms(500);
            // Turn the yellow LED off when the packet has been loaded
            led_yellow.off();
        }

        // Delay the transmission of the next packet 250 ms
        //vTaskDelay(10 / portTICK_RATE_MS);
        Scheduler::delay_ms(50);
    }

    led_green.off();

    Scheduler::delay_ms(1500);
}
}

```

Collegare la scheda OpenMote B trasmettitore via USB ed installare il firmware appena modificato nella scheda con il comando (assicurarsi di essere nella root del progetto `./openmote-fw`, per andare *backward* nelle cartelle utilizzare il comando *cd..*):

```

:~$ sudo make template

```

Ripetere questa operazione per tutti i trasmettitori che si desidera connettere alla rete.

4. Configurazione Gateway

La configurazione del centro-stella è molto simile a quella dei trasmettitori. Spostarsi nella root contenente il firmware (`./openmote-fw`).

Collegare la scheda OpenMote B ricevitore (centro-stella) via USB ed installare il firmware nella scheda con il comando:

```

:~$ sudo make test-radio-cc2538

```

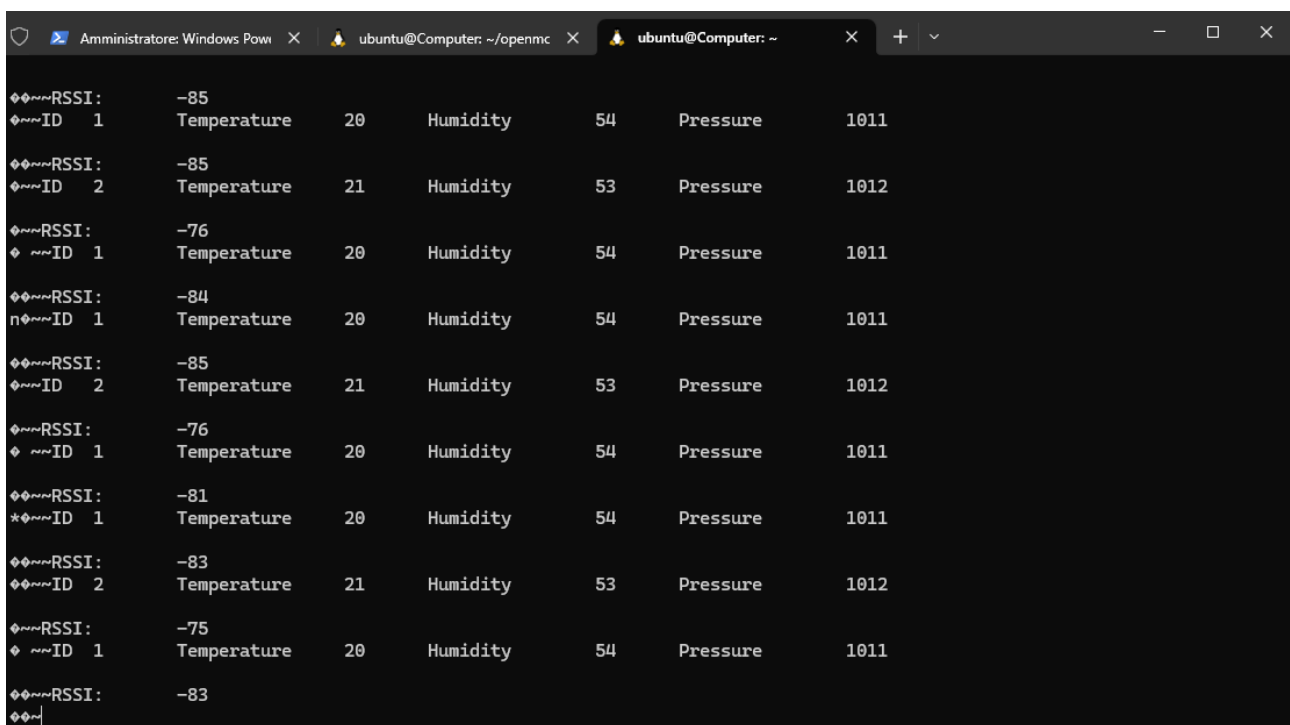
5. Lettura dei dati

A questo punto occorre leggere i dati che arrivano dai vari trasmettitori sul gateway sfruttando l'interfaccia UART USB. L'Universal Asynchronous Receiver and Transmitter (UART) è protocollo in grado di interfacciare dispositivi seriali. Questo protocollo fornisce operazioni come la conversione dei byte in un singolo flusso di bit per la trasmissione in uscita, viceversa per la trasmissione in entrata, tramite l'aggiunta di un bit di Start e Stop per evidenziare l'inizio e la fine di una word di dati, e un bit di parità per il rilevamento degli errori. Non è necessario un segnale di clock, poiché l'UART è asincrono, ma entrambe le estremità della linea devono operare con lo stesso baud rate.

Sul terminal a cui è collegato il gateway digitare il seguente comando:

```
:~$ sudo screen /dev/ttyUSB1 115200
```

In particolare, il primo parametro indica l'interfaccia seriale da cui si vuole leggere i dati, mentre il secondo indica il baudrate a cui la porta seriale trasmette (indicato nell'header del firmware del dispositivo nella funzione main di test radio *#define UART_BAUDRATE (115200)*).



```
#####RSSI:      -85
#####ID  1      Temperature      20      Humidity      54      Pressure      1011

#####RSSI:      -85
#####ID  2      Temperature      21      Humidity      53      Pressure      1012

#####RSSI:      -76
#####ID  1      Temperature      20      Humidity      54      Pressure      1011

#####RSSI:      -84
#####ID  1      Temperature      20      Humidity      54      Pressure      1011

#####RSSI:      -85
#####ID  2      Temperature      21      Humidity      53      Pressure      1012

#####RSSI:      -76
#####ID  1      Temperature      20      Humidity      54      Pressure      1011

#####RSSI:      -81
#####ID  1      Temperature      20      Humidity      54      Pressure      1011

#####RSSI:      -83
#####ID  2      Temperature      21      Humidity      53      Pressure      1012

#####RSSI:      -75
#####ID  1      Temperature      20      Humidity      54      Pressure      1011

#####RSSI:      -83
```

L'RSSI (Receive Signal Strength Indicator) è un parametro utilizzato nelle telecomunicazioni, ed in generale misura la potenza di un segnale. Nell'ambito dell'RFID, il parametro è utilizzato principalmente dal controller per misurare la potenza del segnale ricevuto dal trasmettitore e si misura in dBm.

6. Memorizzazione dei dati in time-series database

Poiché lo scenario IoT proposto teoricamente connette tanti tipi di oggetti e dispositivi, si ottengono enormi quantità di dati da un'ampia gamma di fonti, tra cui i sensori. Questi dati possono essere processati, analizzati e usati per prendere una decisione in tempo reale. L'analisi dei dati allora diventa una risorsa fondamentale in quanto essa può far sì che, un insieme di dati acquisti un valore concreto. L'idea è quella di utilizzare i big data attraverso un framework IoT per aumentare l'efficacia dell'applicazione realizzata.

Per questo motivo installiamo un database time series, nello specifico Influxdb, su WSL digitare i seguenti comandi:

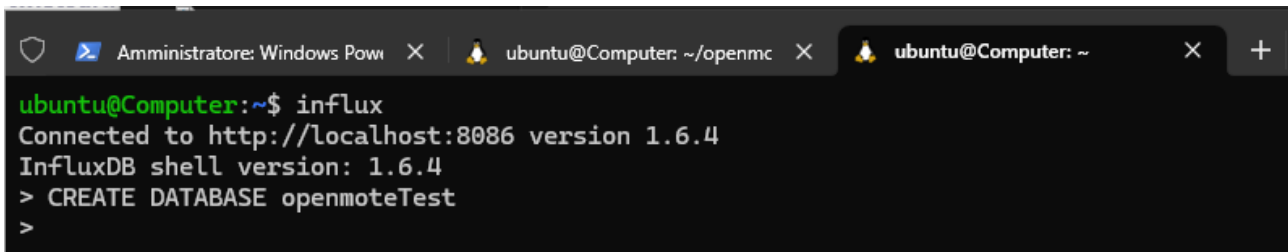
```
:~$ sudo apt install influxdb-client
:~$ sudo apt install influxdb
:~$ sudo pip install influxdb-client
:~$ sudo pip install influxdb
```

In una nuova finestra del terminale lanciare il gestore del database con il comando:

```
:~$ sudo influxd run
```

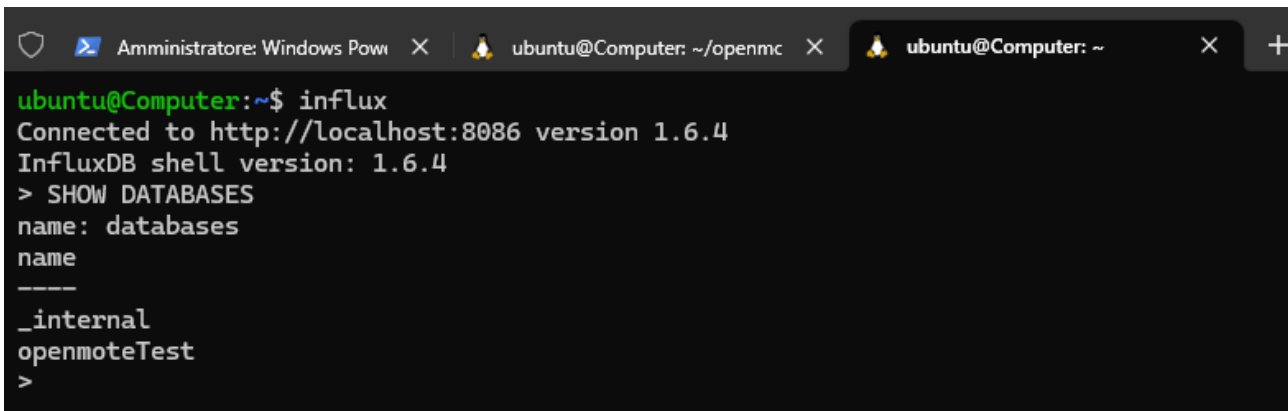
Nella finestra del terminale principale creare il database con i comandi:

```
:~$ influx
> CREATE DATABASE openmoteTest
```

A screenshot of a terminal window with three tabs. The active tab is titled 'ubuntu@Computer: ~'. The terminal shows the command 'influx' being executed, which connects to 'http://localhost:8086' and shows 'InfluxDB shell version: 1.6.4'. Then, the command '> CREATE DATABASE openmoteTest' is entered and executed, followed by a prompt '>'.

Verificare la corretta creazione del database con il comando:

```
> SHOW DATABASES
```

A screenshot of a terminal window with three tabs. The active tab is titled 'ubuntu@Computer: ~'. The terminal shows the command 'influx' being executed, which connects to 'http://localhost:8086' and shows 'InfluxDB shell version: 1.6.4'. Then, the command '> SHOW DATABASES' is entered and executed. The output shows 'name: databases' followed by a table listing the databases: '_internal' and 'openmoteTest'. The prompt '>' is shown at the bottom.

Per uscire dal database è necessario il comando *exit*.

Nella root del progetto ./openmote-fw lanciare lo script Python per decodificare i dati ricevuti e trasformarli in dati persistenti da salvare sul database.

```
:~$ sudo python3 serialComTest.py
```

```
Amministratore: Windows Pow... ubuntu@Computer: ~/openmc... ubuntu@Computer: ~/openmc... ubuntu@Computer: ~/openm... + v
ubuntu@Computer:~/openmote$ sudo python3 serialComTest.py
JSON: [{"measurement": "transmissionData", "tags": {"openmoteID": "2"}, "fields": {"temp": "23", "humidity": "48", "pressure": "1012"}}]
INFO CANALE -> RSSI: -68

JSON: [{"measurement": "transmissionData", "tags": {"openmoteID": "1"}, "fields": {"temp": "22", "humidity": "48", "pressure": "1011"}}]
INFO CANALE -> RSSI: -77

JSON: [{"measurement": "transmissionData", "tags": {"openmoteID": "1"}, "fields": {"temp": "22", "humidity": "48", "pressure": "1011"}}]
INFO CANALE -> RSSI: -77

JSON: [{"measurement": "transmissionData", "tags": {"openmoteID": "2"}, "fields": {"temp": "23", "humidity": "48", "pressure": "1012"}}]
INFO CANALE -> RSSI: -67

JSON: [{"measurement": "transmissionData", "tags": {"openmoteID": "1"}, "fields": {"temp": "22", "humidity": "48", "pressure": "1011"}}]
INFO CANALE -> RSSI: -78
```

Per la visualizzazione e applicazione di eventuali query sul database su base temporale, aprire una nuova scheda nel terminale WSL e digitare il comando:

```
> USE openmoteTest
```

```
Amministratore: Windows Pow... ubuntu@Computer: ~/openmc... ubuntu@Computer: ~/openmc... ubuntu@Computer: ~... + v
ubuntu@Computer:~$ influx
Connected to http://localhost:8086 version 1.6.4
InfluxDB shell version: 1.6.4
> USE openmoteTest
Using database openmoteTest
> Select * From transmissionData
name: transmissionData
time                humidity openmoteID pressure temp
-----
1669399954532160900 47      2          1012     24
1669399968628522200 47      2          1012     24
1669399973330315100 47      2          1012     24
1669399978033529200 47      2          1012     24
1669399982737096600 47      2          1012     24
1669399987441341400 47      2          1012     24
1669399992144461500 47      2          1012     24
1669399996832988000 47      2          1012     24
1669400001537452000 47      2          1012     24
1669400006241042200 47      2          1012     24
1669400010941833900 47      2          1012     24
1669400015644257000 47      2          1012     24
1669400020347995700 47      2          1012     23
1669400025051603400 47      2          1012     23
1669400029739683300 47      2          1012     23
1669400034447297100 47      2          1012     23
1669400039149365100 48      2          1012     23
1669400043852055000 47      2          1012     23
1669400046202460800 53      1          1011     22
1669400048412146600 51      1          1011     22
1669400048554718200 47      2          1012     23
1669400050603210300 49      1          1011     22
1669400052010512000 48      1          1011     22
```

Nell'esempio specifico mostrato, la tabella *transmissionData* indica tutti i dati trasmessi dai sensori al Gateway.