h e p i a

Haute école du paysage, d'ingénierie
et d'architecture de Genève
Laboratoire de systèmes numériques
L. Gantel, A. Upegui, F. Vannel

Hes·so
Haute Ecole Spécialisée
de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences
Western Switzerland

Introduction to Waspmotes : BLE sensors
IoT - HES-SO Master
2015 - 2016 Spring Semester

Objectives

The goal of this laboratory is to get an overview of the notion of client and service on a Bluetooth Low Energy *(BLE)* architecture, and to deploy a simple application making use of BLE sensors for implementing an activity recognition system.

For achieving so, you will familiarize with the *Waspmote* device provided by *Libelium*. It will be used as a GATT client communicating with a BLE module called *Sensor Tag*.

1 Introduction

The first steps consists in interacting with a BLE module from Texas Instrument, in this case a SensorTag module, which will play the role of the GATT server. The final step will be to implement an activity pattern recognition application based on accelerometer service measurements.

1.1 Hardware

For this lab, a box with the following hardware is provided :
- 1 Waspmote PRO module
- 1 BLE daughter board
- 1 BLE antenna
- 1 BLE SensorTag module
- 1 mini-USB cable

Also provided, not necessary for this first lab :
- 1 LoRa daughter board
- 1 LoRaWan daughter board
- 2 LoRa antennas
- 1 Waspmote module
- 1 extension radio board

1.2 Waspmote and development tools

Preliminary note : all the technical Libelium documentations that may be used for these labs are available on Cyberlearn
`https://cyberlearn.hes-so.ch/mod/folder/view.php?id=553019`

Before starting read carefully the following Libelium documentation : "Waspmote quickstart guide" (specific chapters 2, 3, 4, 6, and 7). In chapter 3, please note that we will not use the battery, SD card, nor sensor board. We will only use the BLE board `http://www.libelium.com/development/waspmote`

Waspmote and all hardware should be used with care.

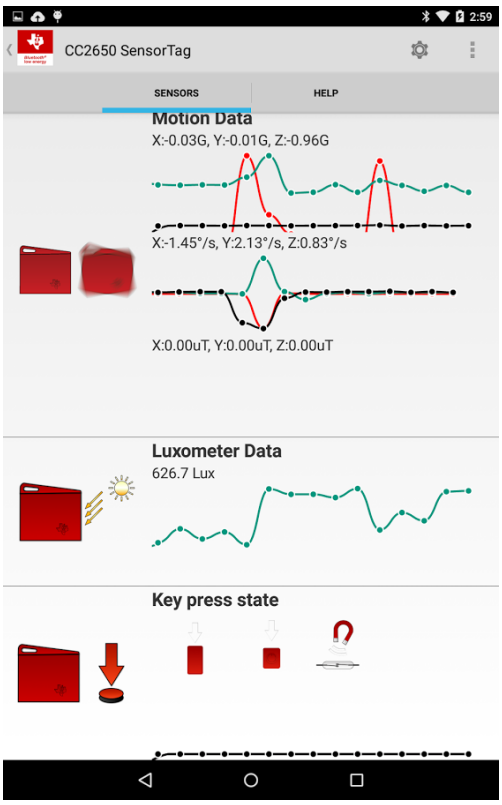
You can find all the available methods on the API documentation : `http://www.libelium.com/api/waspmote/`.

2 Exercises

2.1 Hands-on with SensorTag

If you have a smartphone or tablet equipped with Bluetooth 4.0, install the mobile application for the SensorTag provided by TI. It is called *Simplelink SensorTag*. If you don't join a colleague only for this part.

Once the application is installed, launch it and connect to your SensorTag device. Refer to the document in Cyberlearn for getting the MAC address of your Bluetooth device in order to select the correct one. PLEASE avoid selecting any other, it is the one of your neighbor.



Have a look at the different services available on this module. This is the type of information that you will try to gather later from your Libelium board by accessing yourself the services.

2.2 Hands-on with Waspmote

— Place the BLE daughter board on the Waspmote along with its antenna. The antenna should be outside of the board.
— Connect it to your computer with the USB cable.
— Launch the Waspmote Pro IDE.

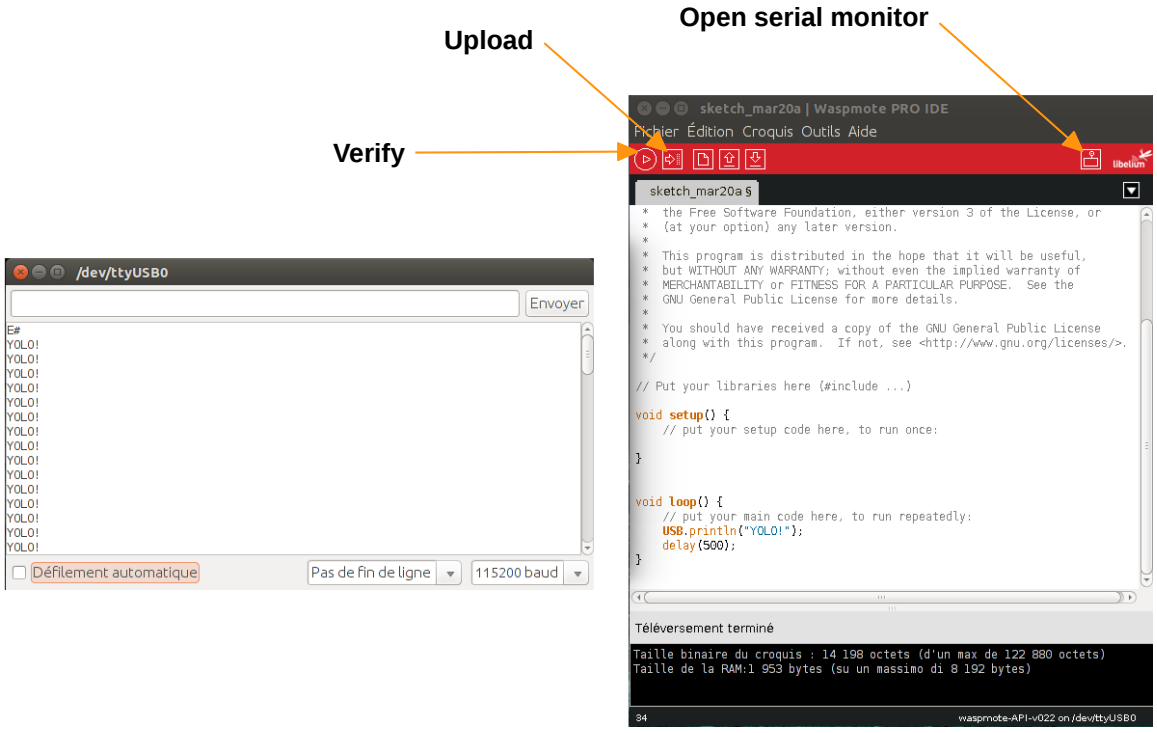
The environment allows you to develop C program in order to control the BLE module. A program is mainly composed of two functions : setup() and loop().

The setup() function is used to initialize the Waspmote, ie. enabling the USB UART or turning on the BLE module, whereas the loop() function is intended to perform a continuous work, such as scanning devices and collecting data periodically.

— Using the USB.println() function, display a message on the UART console. The delay() method can be used to temporarily setup the processor in sleep mode.

— In the menu "Outils / Types de carte", select the Waspmote device and in "Port série", select the serial link attached to the board.
— Click on "Vérifier" to compile and check for syntax errors.
— Click on "Téléverser" to download the binary file to the processor.

The button "Moniteur série" allows you to visualize the serial output.



Note for MAC OS users : An error may occur while downloading the binary to the Waspmote. If you have this problem you must press the reset button on the Waspmote and release it exactly when the console display the first information `Taille binaire du croquis : xxxx Taille de la RAM: xxxx`.

2.3 BLE devices scanning

Lets attempt to get the first contact between your Libelium board and the SensorTag. Turn on the SensorTag. A led should be blinking, it indicates it is advertising.

— Back to the Waspmote IDE, open the device scan example from the menu and have a look on the code :
"Fichier/Exemples/03 - Communication/BLE/BLE_01_Normal_scan"

— Verify and upload the application. Then in the serial console it will print the observed devices, check if your device appears in the inquiry list.

NOTE : If the Waspmote module has been reset while connected to the SensorTag, the latter could be invisible during the next scan because it is still in a connected state. To avoid this, you must keep the reset button of the Waspmote pressed during several seconds until the SensorTag comes back in advertisement mode (blinking LED).

2.4 Managing direct connection and service characteristics

In this exercise the goal is to access the services provided by the SensorTag. For doing so we need to point to the handlers described in the GATT table of the device.

The SensorTag services are listed at the following URL in the "Sensors and Services" dropdown menu :
`http://www.ti.com/ww/en/wireless_connectivity/sensortag2015/tearDown.html`

In order to access these services we will start by opening the example *BLE_07_connection_to_a_ble_device_as_master*. Take a look to this table and identify the services proposed by the GATT profile.

Take a look at this example and modify it in order to :

1. Connect to YOUR SensorTag module by using YOUR SensorTag MAC address. (line 35 of the example)
2. Read the "device name" of your device in the corresponding characteristic (line 70). You must indicate the handler of the value you want to read. From the GATT table, this value is 0x3 for the device name.
3. Now we want to read the IR Temperature Data. For doing so you must first write to the *IR Temperature Config* Handler (GATT handler 0x24) the value for start sampling. This can be done by modifying the example in line 95 of the example.
   Attribute values are expressed in Little Endian. Check that data have been written in the correct order by reading it back.
4. Once the sensor is activated you can read it's value by reading the respective handler (0x21)
5. Now you know how to activate services and read them. Up to you to do the same for the accelerometer, activate the measures, and read data.
   You have to take into consideration that after the activation, the accelerometer will take some time before providing valid data.

NOTE : In the example from the IDE, the function used to write an attribute is adapted to write string attributes. In our case, we have to write byte attributes, so you have to define an array of bytes and to use the attributeWrite() function which takes this array and the number of bytes to write as parameters :

```
unsigned char attributeData[2] = {0x00};      // Declare the byte array
attributeData[0] = 0xAB;                      // Setup the attribute value
// Write the attribute value (1 byte)
if (BLE.attributeWrite(BLE.connection_handle, 42, attributeData, 1) == 0) {
...
```

2.5 Handling notifications

An alternative use of BLE instead of periodically asking, is to configure notifications. The client can configure the server in order to send notifications periodically.

For doing so :

— From the GATT table, find the notification characteristic in the accelerometer service associated to the movement data.
— Open the example BLE_10_characteristic_notification_master and modify it in order to enable and display accelerometer notifications.

2.6 Activity recognition

Based on the previous exercises, implement a new application in which you will discriminate the activities performed by someone wearing the SensorTag module as a watch.

We want to discriminate three activities : Still, Walking and Running. The activity recognition will be based on the intensity of the movements performed by the user over a period of time.

This can be done by following the next steps :

1. Configure the accelerometer to get samples for the 3 axis during 2 seconds at a sampling frequency of 10 Hz. These samples can be even read as done in subsection 2.4 or notified by the SensorTag as done in 2.5.
2. Store these samples in an array.
3. Identify the value of the measured gravity in each of the 3 axis, and in both directions. Is it always the same ?
4. The gravity is an undesired measured effect when computing activity intensity. We want to remove it. The problem is that the orientation of the device can be completely independent from the activity so it cannot be just subtracted in one axe.
   For doing so, we can first compute the magnitude of the acceleration in the form of an Euclidean (or pseudo-Euclidean) vector.
   You can then perform the absolute value of the difference between the measured magnitude and the gravity.
   At the end, You got a quantification of the intensity of the movement. (See notes below)
5. The movement intensity on a single sample is meaningless. You need a sequence of them in order to make sense. You can for instance compute the average intensity during 2 seconds.
6. Then you can apply a very simple decision tree in order to discriminate activities by comparing the average intensity with a couple of threshold values. Up to you to find them !
7. Wait for 10 seconds... and restart a new capture.
8. The activity logs will be stored on a remote server through a LoRa interface... But this will be done in the next lab.

NOTE 1 : Be careful with operations overflow, verify computation intermediate results. You don't need high precision in your computation. Scale-down your values if required

NOTE 2 : Keep in mind you have a small 8 bit processor with reduced memory resources for performing computation. Avoid overestimating types, storing useless data, and using complex mathematical functions like square roots !