Internet of Things (ECE:5550)
Spring 2024
Lab 03

Start Date: Wednesday, March 27

Due Date: Sunday, April 7 by 11:59pm (ICON submission)

Lab Teams: same as previous labs.

Overview

The objective of this lab is to gain experience using **Bluetooth Low Energy (BLE)** with the Arduino and the Raspberry Pi. To do this, you will use the **node-ble** node.js library on the Pi and the **ArduinoBLE** library on the Arduino. This lab will extend Lab 1 and Lab 2 by using Bluetooth LE to communicate temperature data from the Arduino to the Raspberry Pi. The Raspberry Pi will then publish the temperature data to Firebase. In addition, the interval that the temperature (via Arduino) and humidity (via the Pi) are collected will be controlled remotely from Firebase.

Getting Started

Getting Started on the Pi

Before embarking on our journey to learn about BLE, we need to install the **bluez** BLE stack on the Raspberry Pi, along with the node-ble library that will allow us to communicate with the BLE HCI. You can install these packages by entering the following commands on your Pi:

```
sudo apt-get update
```

```
sudo apt-get install bluetooth bluez libbluetooth-dev libudev-dev
```

Now, create and change directories to a new folder for this lab (e.g., ~/iot/lab3).

Once here, run:

```
npm install node-ble
```

Now that we have the proper libraries installed, the first thing we will eventually do is gain some familiarity with the functionality of Bluetooth LE, the Arduino, and the node-ble Bluetooth LE libraries. We will do this by setting up a simple **bi-directional Bluetooth link** between the Arduino and the Pi using the **Nordic UART service profile**.

Recall from lecture that Bluetooth LE uses the concept of UUIDs and GATT service profiles to define and advertise the types of services implemented by a Bluetooth device. *Services* expose *Characteristics*, which are the endpoints for communicating different types of data required by the service. Services and their characteristics are identified by unique UUIDs. The **Nordic UART Service** has the following UUID:

```
6E400001-B5A3-F393-E0A9-E50E24DCCA9E
```

The Nordic UART service **emulates a simple bi-directional serial port**. This **service** has two **characteristics**:

```
TX Characteristic (UUID: 6E400002-B5A3-F393-E0A9-E50E24DCCA9E)
```

```
RX Characteristic (UUID: 6E400003-B5A3-F393-E0A9-E50E24DCCA9E)
```

Please see this link for more info on this UART service. With the libraries installed on the Pi, you'll now get the Arduino ready to roll. Once that's done, we'll wrap up the example on the Pi.

Getting Started on the Arduino

To get started on the Arduino side of things, you'll need the ArduinoBLE library to interface with the Bluetooth LE module on the board. To see if the library is installed, open the Arduino editor. In the menu bar, highlight Sketch > Include Library. In the list, if you see "ArduinoBLE" then you should be ready to go. Otherwise, go to Tools > Manage Libraries…, search for ArduinoBLE and then install the library from the Library Manager. You should now be able to see the library listed in the Sketch > Include Library dropdown menu.

Once you have the library installed, you should now be able to write Arduino code that interfaces with the Bluetooth module. To test and see an example of this, please download the "1_UART-Example.zip" file from ICON. This simple example shows how to set up and use the Nordic UART service on the Arduino via the ArduinoBLE library. Later, we will see how to add a new service profile. Please review the code in the ArduinoBLE-UART file to see how the BLE device, services, and characteristics are initialized and used.

Compile and upload the ArduinoBLE-UART example to the Arduino. You can then open the Serial Monitor window by going to Tools > Serial Monitor in the Arduino editor. Important: if you do not see any output in the Arduino serial monitor, make sure that the window's settings are No line ending and 9600 baud. If you continue to have issues, you may find it helpful at times to close the Serial Monitor window, click the Restart button on your Arduino, and then reopen the Serial Monitor window. You should now see a "Bluetooth device (XX:..) active, waiting for connections…" printout.

Putting Everything Together

Back on the Pi, enter the following command:

`hciconfig`

If the response indicates that the Bluetooth interface is “DOWN” enter the following to activate:

`sudo hciconfig hci0 up`

Now type:

`sudo bluetoothctl`

You should see a prompt that looks like:

`[Bluetooth]#`

At the prompt, enter:

`scan on`

Bluetoothctl will begin to display a list of discovered devices. As soon as you see the discovery for “ArduinoBLE UART", you can stop scanning by typing:

`scan off` *(if you don’t see this advertisement for ArduinoBLE UART, try restarting the Arduino board by hitting its on-board button and then running the scan again)*

Copy the address for the ArduinoBLE device. When you are done, exit bluetoothctl by typing:

`quit`

Now, copy the file `nodeble-uart.js` from the zip file downloaded from the ICON site and paste it into a file on your Pi in this lab’s directory (e.g., `~/iot/lab03/).` This file contains a very simple example that shows how to interact with Bluetooth on the Pi to implement a simple Nordic UART communication service.

Edit the device address in Line 5 to the discovered address of your ArduinoBLE device.

Please review the code to see how the node-ble library is used to scan for Bluetooth peripherals, connect to your device, get references to the UART service and characteristics, and transmit and receive data using the characteristics. Now, save and then run the program by typing:

`sudo node nodeble-uart.js`

After connecting, you should be able to exchange data between your Pi terminal window and Arduino serial window. Try typing in your Pi terminal and hitting enter. Your message should send from the Pi to the Arduino serial window (important, ensure that the Arduino serial window's settings are "No line ending" and "9600 baud"). Similarly, you should be able to type a message into the Arduino serial window and hit the "Send" button; the Pi should print the received message content on its respective console. If things do not seem to be working, please take note of the console.log() or Serial.print() statements in the Pi and Arduino code to see where the issue may be occurring. Sometimes you may see a "Software caused connection abort" on the Pi's console. Typically, simply rerunning (ctrl+c to kill the node process, restart the program) can fix this. If you cannot connect after multiple attempts, please reset your Arduino. You can also try rescanning for devices on the Pi to ensure that your Arduino's Bluetooth address has not changed. If it has, please update the *ARDUINO_BLUETOOTH_ADDR* variable.

Exploring GATT

You are ready to work with a different GATT service. On ICON, download the file "2_ESS-Starting.zip" and extract it. Open the ArduinoBLE-Lab3 Sketch stored within. This extended example *simulates* a temperature sensor and periodically transmits a random temperature via Bluetooth LE using another standard GATT service profile called Environmental Temperature Sensing (we looked at this one in class). Its specification can be found here:

https://www.bluetooth.com/specifications/specs/environmental-sensing-service-1-0/

The Environmental Sensing Service has a mandatory characteristic called "Temperature." Its specification is listed under "Temperature" (pg. 222) here:

https://www.bluetooth.org/DocMan/handlers/DownloadDoc.ashx?doc_id=524815

Note that the Temperature characteristic has just one mandatory field which is a signed 16-bit integer containing the current temperature in Celsius. The resolution of the temperature is 0.01 degrees Celsius. To get this resolution in an integer, the decimal value must be multiplied by 100 before being converted to an integer and transmitted. Upon receipt, it will need to be divided by 100 to get the original decimal.

Examine the example Arduino sketch you got from ICON to see how ArduinoBLE is used to (1) set up the Environmental Sensing service and the Temperature characteristic and (2) how data is transmitted via the characteristic.

Receiving on the Pi

Modify the Pi code in the ArduinoBLE-Lab3 folder to support the temperature characteristic. Model this new value after the existing examples: uartService, txChar, and rxChar. You will have to identify and initialize the new temperature characteristic based on its UUID (found in the GATT specification, included in the example code), signify that you want notified of new data at this characteristic, and provide a callback to process said data.

Note that the data received by the callback function `<characteristicVar>.on()` method should be a two element Buffer. Element 0 of the buffer will contain the LSB of the temperature value and element 1 will contain the MSB of the temperature value. You should convert this data

back into a Celsius value with precision of 0.01 (per the GATT specification). Looking at how the Arduino is sending temperature values may help in this process. To simplify this conversion, you can assume the received temps will be positive. Also, remember to turn on notifications for incoming measurement data using `<characteristicVar>.startNotifications()` as in the nodeble-uart.js example.

Lab Assignment

For this lab, you will augment your Lab 2 solution with the ability to harvest the temperature data from the Arduino via Bluetooth LE. You will also add the ability to control the interval at which the temperature and humidity data are harvested. To do this, you will need to add one value to your Firebase Database: `Interval`. `Interval` will control the sampling rate of the simulated environmental sensor service on the Arduino and the humidity sensor on the Pi. The valid values of `Interval` are 1-10, corresponding to a sampling interval of one to ten seconds.

You will need to modify the example Arduino sketch so that it can vary the sampling interval in the range of one to ten seconds. Whenever the value of Interval changes in the Firebase database, the updated value should be pushed to your Node app on the Pi and should then be sent wirelessly to the Arduino **using the Bluetooth LE Nordic UART service**. The JavaScript app on the Pi should also change the sampling interval of the humidity to match the new value of `Interval`. This means that your Arduino sketch (and your JavaScript app) will **need to use both the Bluetooth LE Nordic UART and Environmental Sensing service profiles.** You will notice that the example Arduino sketch already has the UART service set up as well as some code to read Nordic UART data sent from the Pi.

Here are the specific requirements for your JavaScript app and Arduino sketch:

JavaScript App:

- Must still be able to update the sense-hat's light array like in Lab 2.
- Add value `Interval` to your Firebase database (valid values are 1-10 **seconds**).
- Set up your JavaScript app to receive a callback whenever the value of `Interval` changes in the Firebase database. This callback should be specifically on the Interval entry, not on the root of your database. Your callback function should **forward the new Interval value to the Arduino using the Bluetooth LE Nordic UART service profile**. It should also change the sampling rate of the humidity readings to reflect the new Interval value. The humidity should be read and pushed to Firebase at the rate specified by the new Interval value. Your app should print a message when the Interval value changes.
- Must send the updated Temperature value to Firebase whenever it is received from the Arduino via the **Temperature characteristic** of the **Environmental Sensing service**.
- Messages that your JavaScript App should print (at a minimum):
    - The temperature whenever it receives it from the Arduino.
    - The humidity after every reading at rate specified by Interval.
    - The new value of Interval whenever it changes on Firebase.

Arduino Sketch (modify the example Arduino sketch so that it can):

- Read the temperature from the HTS221 sensor used in Lab 1 and transmit it to the Pi via the **Temperature characteristic of the Environmental Sensing service** at the Interval specified by the Pi. DO NOT use the UART service to transmit temperature data.
- Receive the sampling interval from the Pi via the Bluetooth LE **Nordic UART service** profile and change the sampling interval of the temperature sensor accordingly.
- Print the following messages to the serial monitor (at a minimum):
    - The temperature after every Interval's reading.
    - The new Interval every time it is received from the Pi. This message should be printed within one second of the Pi sending the new interval.

Submission & Check-off (one submission per lab team)

Submit to the ICON dropbox screenshot(s) showing your Arduino Serial Monitor, Pi ssh terminal window, and the Firebase dashboard. The screenshot should be taken immediately after changing the value of Interval from the Dashboard and should show the appropriate message displayed by the Pi and Arduino in response to the change.

Please submit the source code files for your JavaScript app and Arduino Sketch.

In addition, please submit a brief 1-2 page lab report describing an overview of your completed lab, code, and how everything runs and is integrated together from an IoT perspective. Essentially, demonstrate your basic understanding of the provided code. Please also detail the various steps your team took towards completing the lab. This may include any issues you ran into along the way, how you overcame them, any observations you had, and, if necessary, a breakout of who completed which parts of the lab. Here you can include additional screenshots, code snippets, and insights you gained. This is not meant to be too formal or much work, just detail your process, what you learned, and an overview of your completed system.

The submission deadline is Sunday, April 7 by 11:59pm (ICON submission).

As before, you will need to demonstrate your completed lab to a TA and receive their check-off. The TAs will have a sign-up sheet available in the Teams Checkoff channel's files to allow teams to reserve slots for this check-off. **If you would like to meet with the TA outside of these times, please make these arrangements via email.**

If you have any issues, particularly with installing any of the libraries or dependencies, please contact a TA or Prof. Bell ASAP so we can help get you on the right track. Feel free to post a question on the course Teams Questions channel, send a DM in Teams, or send an email. We are here to help!