



SOFE 4610U:
Internet of Things

Lab #2: Connecting IoT devices using BT wireless and Arduino Programming

Table of Contents

1. Objectives.....	3
2. Important Notes	3
3. Background information	3
4. Lab Activity	7
4.1 Pair HC-5 and Raspberry pi using Bluetooth communication.....	7
4.2 Pair the Bluetooth dongle with your windows machine	Error! Bookmark not defined.
4.3 Communication with RFCOMM (optional)	10
4.4 Pair any bluetooth device with the Raspberry Pi, please include the steps you perform.	12
4.5 Arduino Programming.....	12
5. Lab Deliverables.....	12
6. Reference	13

1. Objectives

- Raspberry Pi 4 model B has come with built-in Wi-Fi and Bluetooth 4.1. However, it didn't work as coming out of the box.
- **Bluetooth Low Energy (BLE)** is basically a wireless network technology everywhere in our life. In this experiment, you will explore in detailed how it works with Raspberry Pi and perform **Python** script
- To setup the Bluetooth on Raspberry Pi 4 can be fuzzy. It could be hit and miss. Basically Bluetooth in **Linux**.

2. Important Notes

- Work in groups of Four students
- All reports must be submitted as a PDF on blackboard, if source code is included submit your report as PDF and source files as an archive (e.g. zip, tar.gz)
- Save the submission as <lab_number>_<first student's id> (e.g. lab1_100123456.pdf)
- If you cannot submit the document on blackboard then please contact the TA with your submission.

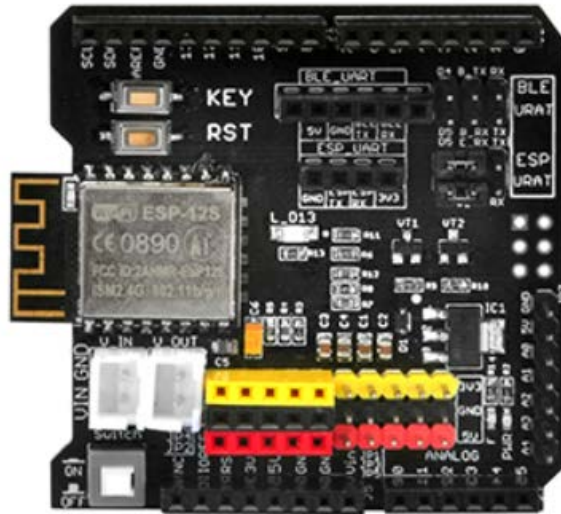
3. Background information

Hardware introduction

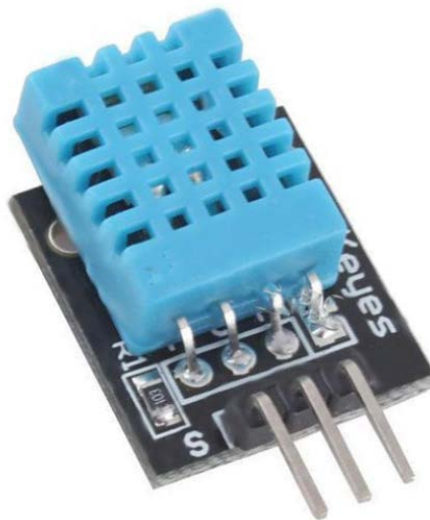
The Osoyoo Wifi IoT learning Kit is provided to you as the lab kit, that includes many hardware pieces. The major components we are focusing on in the labs are as follows, Arduino, Wifi shield and DHT11 temperature sensor.



Arduino UNO R3

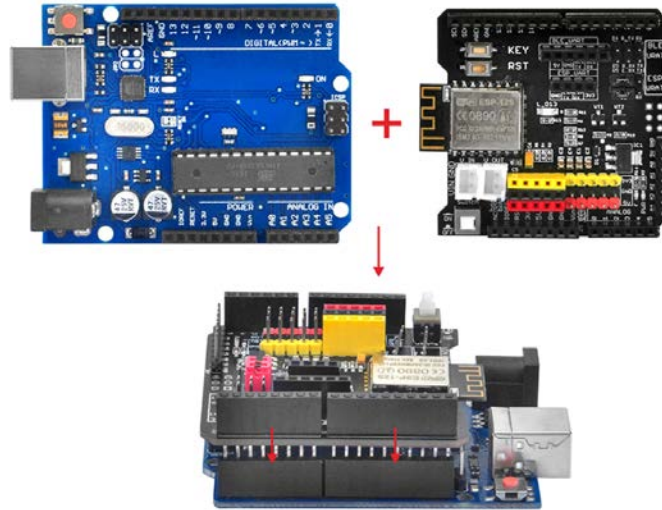


ESP8266 WIFI Shield

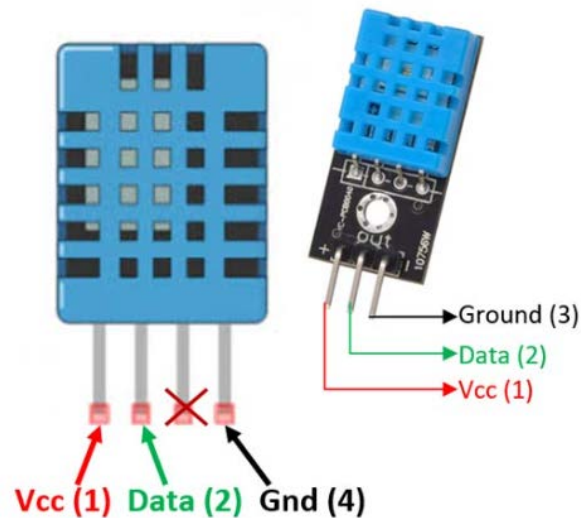


DHT11 Temperature and Humidity Sensor

The WIFI shield could be easily mounted on top of the Arduino as shown



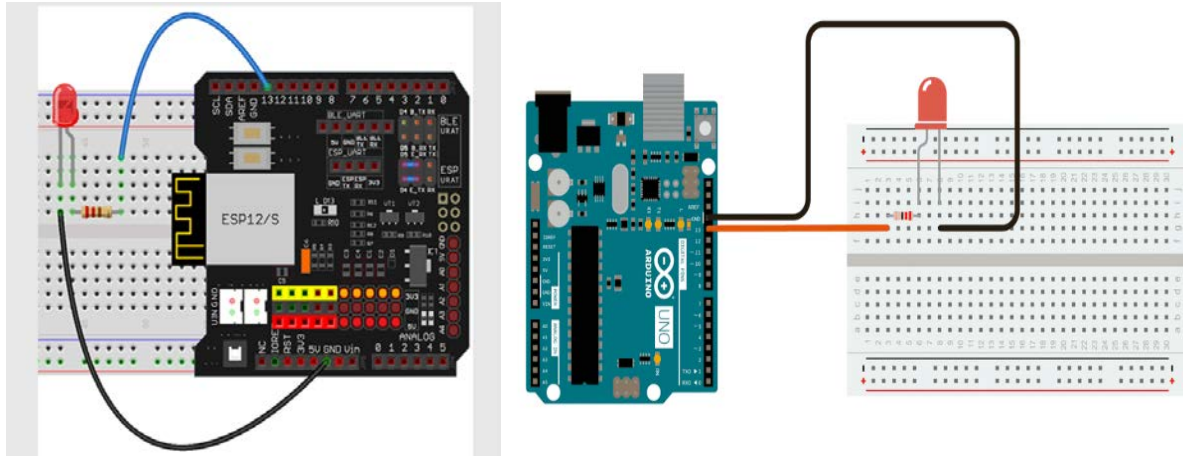
In order to connect the sensor with Arduino, please utilize the given jumper wires in the kit. The following is the pin configuration for the sensor, how to connect to Arduino with proper Vcc and ground is self-explanatory. The data pin can be used any of the unused pins from Arduino such as pin 4.



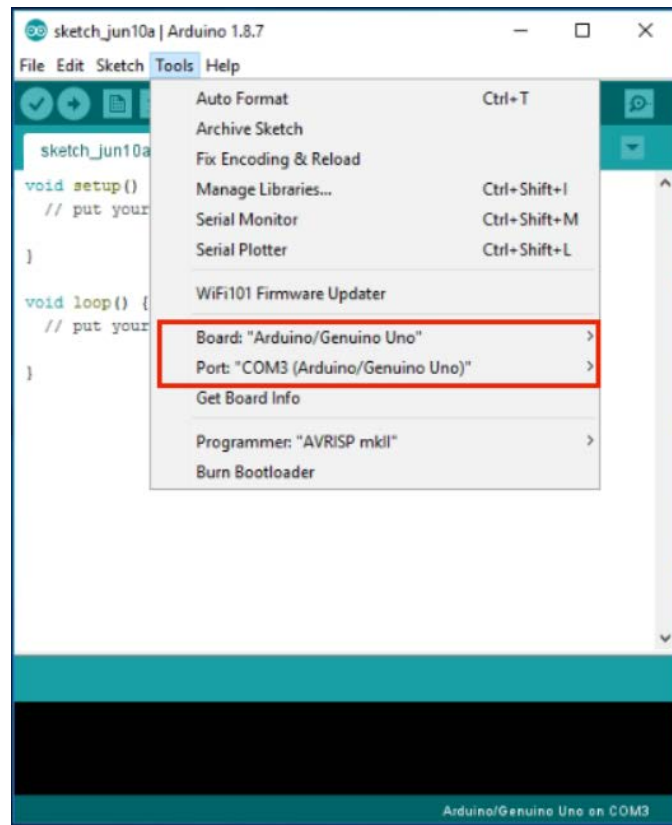
Arduino programming introduction

As many of you are already familiar with, Arduino programming is essential programming skills you should have at this stage of undergraduate study. The Arduino programming tool is the Arduino IDE, which you can download from the web.

After install the IDE, you could perform some basic programming hardware components such as a LED with a 200 Ω resistor per say as follows with either the shield or Arduino directly



Connect the Arduino with your host computer, open the IDE and choose the corresponding board and port type for your project. It can be looked like as shown below, but keep in mind the com port configuration could be varied from your own computer.



Practice with the following code to flash the LED every second.

```

void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(1000);                     // wait for a second
  digitalWrite(LED_BUILTIN, LOW);  // turn the LED off by making the voltage LOW
  delay(1000);                     // wait for a second
}

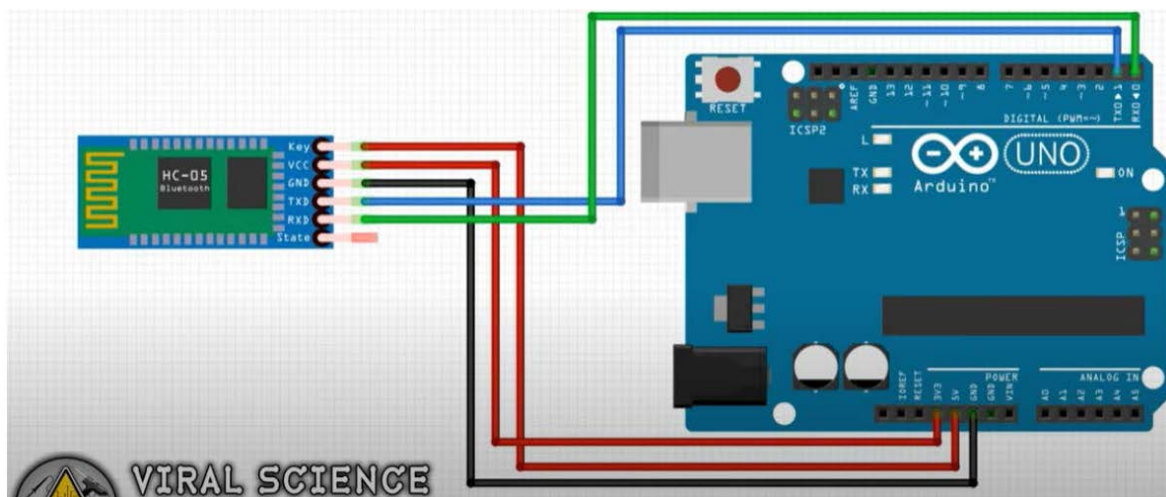
```

4. Lab Activity

As previously practice, please login into the Raspberry Pi in the console window. In the console terminal, we really don't require the GUI for our lab purpose. You will pair CC2650 wireless MCU with Raspberry pi using Bluetooth communication.

4.1 Pair HC-5 and Raspberry pi using Bluetooth communication.

Make up power up the HC module as following



4.1.1 Command line

1. You will need to login in a root or super user to update the pi. However, you will have to make sure the Pi has internet connection.
2. First login as the root. To get the latest update and packages

```

1. $sudo su
2. $apt-get update
3. $apt-get upgrade

```

3. Restart the pi after upgrade

```
$init 6 or reboot // restart the raspberry Pi
```

4. Install pi Bluetooth

```
$apt-get install pi-bluetooth
```

5. To verify if the Bluetooth has been installed on the system,

```
$dpkg -l bluez
```

```
$dpkg -l bluez-firmware
```

As previously indicated, Raspberry Pi has installed those packages for us.

6. Bluetooth command terminal

```
pi@raspberrypi:~ $ bluetoothctl
[NEW] Controller B8:27:EB:26:0D:2E raspberrypi [default]
[NEW] Device D4:25:8B:80:EB:97 UOSL17-4MXBQH2
[bluetooth]#
```

```
$bluetoothctl //this will invoke a Bluetooth shell as shown above
```

```
[Bluetooth] help
```

```
[Bluetooth] list
```

It will list all Bluetooth controllers. You may experience network lose issue using putty working from home, network connection dropped, the sudden slowing down.

```
[Bluetooth] agent on
```

```
[Bluetooth] default-agent
```

```
[Bluetooth] scan on
```

```
pi@raspberrypi:~ $ sudo hcitool lescan
LE Scan ...
68:FC:8A:46:0D:3E (unknown)
00:0C:BF:02:7A:3D HC-02
00:0C:BF:02:7A:3D HC-02
7D:8B:1C:05:BA:AC (unknown)
7D:8B:1C:05:BA:AC (unknown)
70:99:59:23:BD:88 (unknown)
70:99:59:23:BD:88 (unknown)
70:99:59:23:BD:88 (unknown)
```

Please note: the screen shot above as HC-02. For your lab kit, it should show as HC-05 with different mac address.

```
[Bluetooth] pair 00:0C:BF:02:7A:3D
```


Next **gatttool** is used to work with Bluetooth low energy devices, type `–help` for details

```
1. $gatttool -b 00:0C:BF:02:7A:3D -I
```

```
[blueetooth]# exit
pi@raspberrypi:~ $ gatttool -I
[LE]> connect 00:0C:BF:02:7A:3D
Attempting to connect to 00:0C:BF:02:7A:3D
Connection successful
[00:0C:BF:02:7A:3D][LE]> characteristics
handle: 0x0002, char properties: 0x02, char value handle: 0x0003, uuid: 00002a00-0000-1000-8000-00805f9b34fb
handle: 0x0004, char properties: 0x02, char value handle: 0x0005, uuid: 00002a01-0000-1000-8000-00805f9b34fb
handle: 0x0011, char properties: 0x02, char value handle: 0x0012, uuid: 00002a29-0000-1000-8000-00805f9b34fb
handle: 0x0013, char properties: 0x02, char value handle: 0x0014, uuid: 00002a24-0000-1000-8000-00805f9b34fb
handle: 0x0015, char properties: 0x02, char value handle: 0x0016, uuid: 00002a25-0000-1000-8000-00805f9b34fb
handle: 0x0017, char properties: 0x02, char value handle: 0x0018, uuid: 00002a27-0000-1000-8000-00805f9b34fb
handle: 0x0019, char properties: 0x02, char value handle: 0x001a, uuid: 00002a26-0000-1000-8000-00805f9b34fb
handle: 0x001b, char properties: 0x02, char value handle: 0x001c, uuid: 00002a28-0000-1000-8000-00805f9b34fb
handle: 0x001d, char properties: 0x02, char value handle: 0x001e, uuid: 00002a23-0000-1000-8000-00805f9b34fb
handle: 0x001f, char properties: 0x02, char value handle: 0x0020, uuid: 00002a2a-0000-1000-8000-00805f9b34fb
handle: 0x0022, char properties: 0x0a, char value handle: 0x0023, uuid: 49535343-6daa-4d02-abf6-19569aca69fe
handle: 0x0024, char properties: 0x0a, char value handle: 0x0025, uuid: 49535343-aca3-481c-91ec-d85e28a60318
handle: 0x0026, char properties: 0x10, char value handle: 0x0027, uuid: 49535343-1e4d-4bd9-ba61-23c647249616
handle: 0x0029, char properties: 0x0c, char value handle: 0x002a, uuid: 49535343-8841-43f4-a8d4-ecbe34729bb3
handle: 0x002b, char properties: 0x1a, char value handle: 0x002c, uuid: 49535343-aca3-481c-91ec-d85e28a60318
[00:0C:BF:02:7A:3D][LE]>
```

4.1.2 GUI option: install Bluetooth manager

```
1. $sudo apt-get install Bluetooth bluez blueman
```

Then go the desktop select Preferences, pair the device from there.

4.2 Communication with RFCOMM (optional)

If you have additional raspberry Pi, use 2 Raspberry Pi to run and run the code as following on each pi, to what will happen on each Raspberry Pi. Bluetooth programming in Python follows general socket programming mode. [1]

Make sure the following installations.

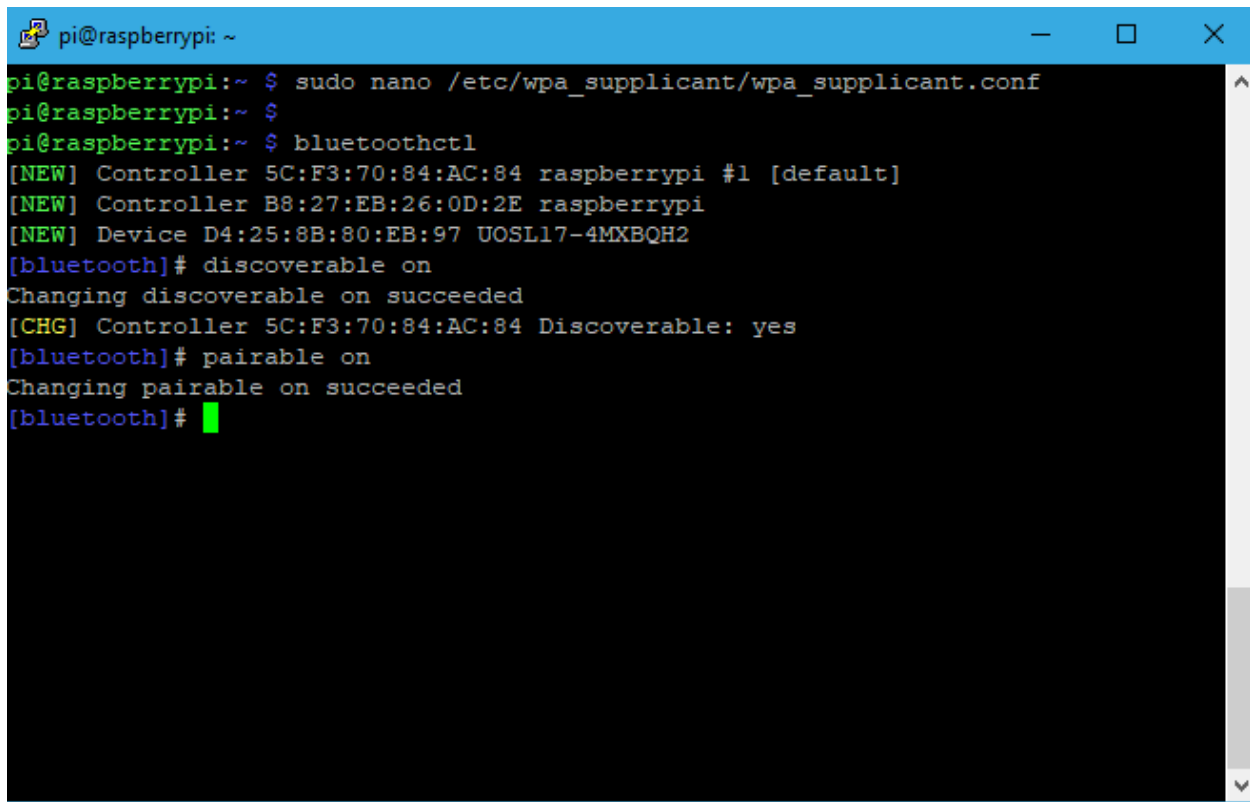
```
1. python3 -m pip install pybluez
```

if didn't work, so run the followings

```
1. $sudo apt-get install bluetooth libbluetooth-dev
2. $sudo python3 -m pip install pybluez
3. sudo apt-get install bluez
4. $ sudo apt-get install python-bluez
```

Still BluetoothSocket is not defined, adding bluetooth.BluetoothSocket in to the .py files.

```
1. $bluetoothctl
2. [Bluetooth]# discoverable on
3. [Bluetooth]# pairable on
```

A terminal window titled 'pi@raspberrypi: ~' with standard window controls. The terminal shows the following commands and output:

```
pi@raspberrypi:~ $ sudo nano /etc/wpa_supplicant/wpa_supplicant.conf
pi@raspberrypi:~ $
pi@raspberrypi:~ $ bluetoothctl
[NEW] Controller 5C:F3:70:84:AC:84 raspberrypi #1 [default]
[NEW] Controller B8:27:EB:26:0D:2E raspberrypi
[NEW] Device D4:25:8B:80:EB:97 UOSL17-4MXBQH2
[bluetooth]# discoverable on
Changing discoverable on succeeded
[CHG] Controller 5C:F3:70:84:AC:84 Discoverable: yes
[bluetooth]# pairable on
Changing pairable on succeeded
[bluetooth]#
```

You should open a new terminal to run the server sider first, by executing

```
1. $python3 rfcomm-server.py
```

On the Bluetooth terminal, you should see the following

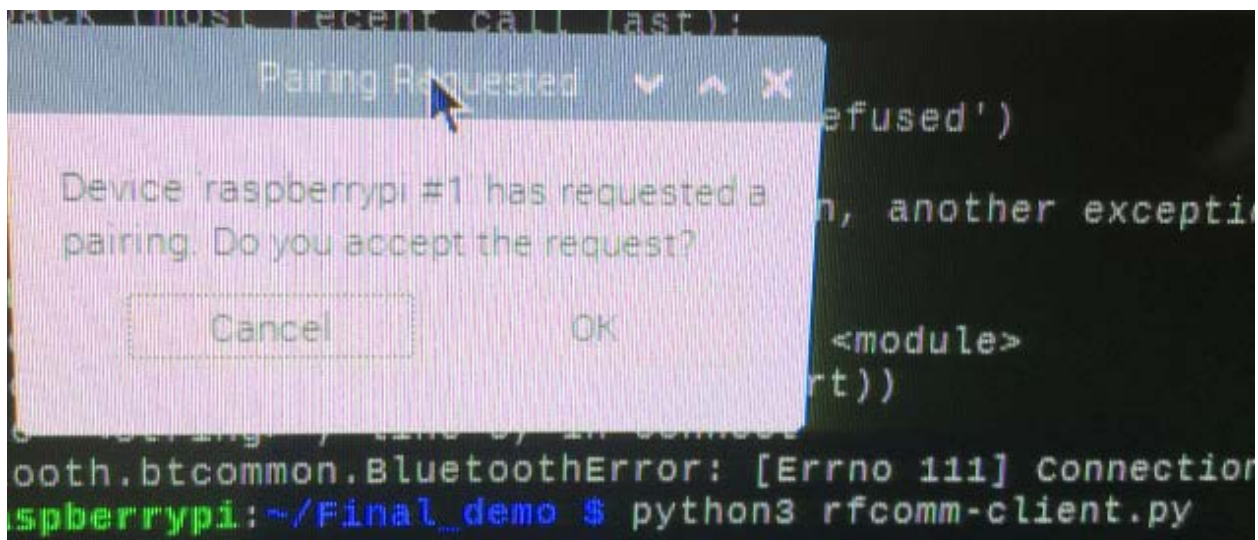
```
received:
pi@raspberrypi:~/bluetooth $ sudo python3 rfcomm-server.py
Accept connection from
received:
pi@raspberrypi:~/bluetooth $
```

At the same time, you should see the following on the bluetooth terminal, you should see the following

```
[CHG] Device B8:27:EB:D5:89:80 UUIDs: 00001103-0000-1000-8000-00805f9
[CHG] Device B8:27:EB:D5:89:80 UUIDs: 00001106-0000-1000-8000-00805f9
[CHG] Device B8:27:EB:D5:89:80 UUIDs: 00001108-0000-1000-8000-00805f9
[CHG] Device B8:27:EB:D5:89:80 UUIDs: 0000110a-0000-1000-8000-00805f9
[CHG] Device B8:27:EB:D5:89:80 UUIDs: 0000110b-0000-1000-8000-00805f9
[CHG] Device B8:27:EB:D5:89:80 UUIDs: 0000110c-0000-1000-8000-00805f9
[CHG] Device B8:27:EB:D5:89:80 UUIDs: 0000110e-0000-1000-8000-00805f9
[CHG] Device B8:27:EB:D5:89:80 UUIDs: 00001112-0000-1000-8000-00805f9
[CHG] Device B8:27:EB:D5:89:80 UUIDs: 0000111f-0000-1000-8000-00805f9
[CHG] Device B8:27:EB:D5:89:80 UUIDs: 0000112f-0000-1000-8000-00805f9
[CHG] Device B8:27:EB:D5:89:80 UUIDs: 00001132-0000-1000-8000-00805f9
[CHG] Device B8:27:EB:D5:89:80 UUIDs: 00001133-0000-1000-8000-00805f9
[CHG] Device B8:27:EB:D5:89:80 UUIDs: 00001200-0000-1000-8000-00805f9
[CHG] Device B8:27:EB:D5:89:80 UUIDs: 00005005-0000-1000-8000-0002ee0
[CHG] Device B8:27:EB:D5:89:80 ServicesResolved: yes
[CHG] Device B8:27:EB:D5:89:80 Paired: yes
```

On the client side,

1. `$python3 rfcomm-client.py`

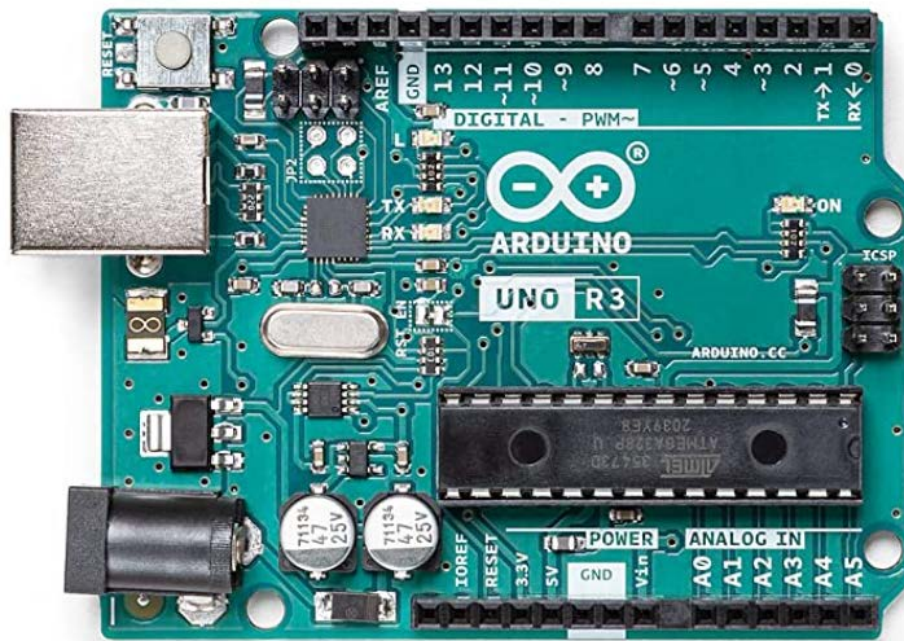


4.3 Pair any bluetooth device with the Raspberry Pi, please include the steps you perform.

4.4 Arduino Programming

Before you come to this experiment, you're assume that you have adequate programming experience in the past, especially in C or C++. Arduino code is written in C++ with an addition of special method and functions. When you develop code in Arduino, unlike in other programming environment, you create a sketch, the name given to Arduino code files.

The reason it's called Arduino programming is that the hardware piece Arduino is deeply involved with the coding as shown below.



All pin allocations are self-explanatory. For some specific external hardware, such as DHT11 temperature sensor, you could download the library using Arduino IDE.

Write a program, connect the DHT11 sensor, read the temperature and display it on the serial terminal from Arduino.

5. Lab Deliverables

1. Connecting the Bluetooth Module HC series to Raspberry Pi using above methods, and demonstrate during the lab session. Save the screen shots and record your steps in the lab report.
2. Extract the data such as temperature reading from DHT11 using in Arduino programming.

Hints: More details how to use the DHT library will be provided during the lab session, and make install.

3. Submit your Arduino development code snipping screen shots and how to read the DHT11 Sensor using the Arduino IDE.

Submit your lab report including all steps and screen shots.

6. Reference

[1] <https://people.csail.mit.edu/albert/bluez-intro/x232.html#rfcomm-server.py>