Charith Perera (Eds.)
PhD, MBA

Internet of Things

Systems Design

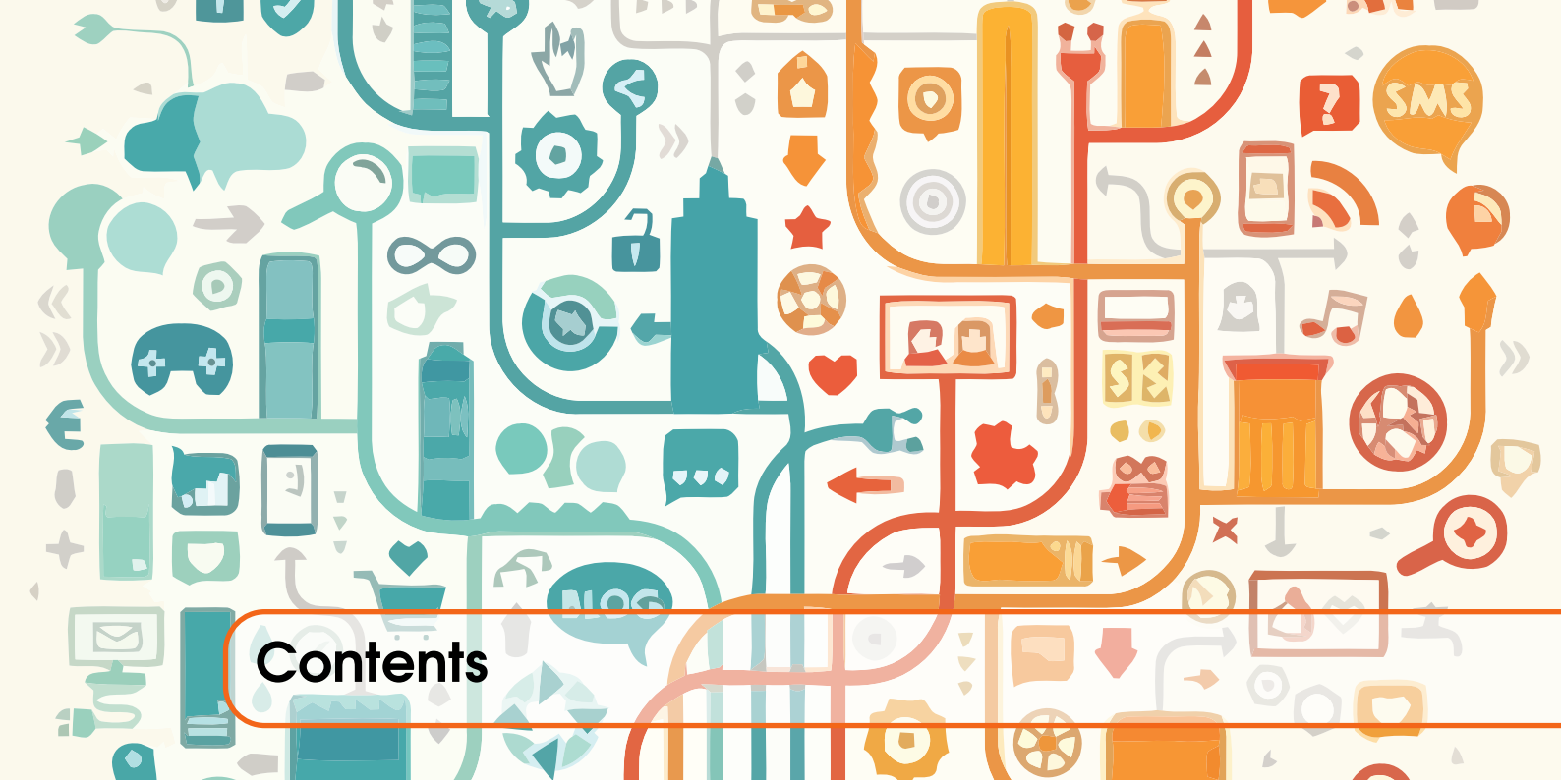
Lab Book

PUBLISHED BY IOT GARAGE

How to cite this book
Charith Perera (Eds.), Internet of Things: Systems Design Lab Book, IOT Garage, 2023

Contributing Authors (alphabetical order)
Hakan Kayan, Michal Malecki, Yasar Majib

Version 2.2 (Latest), July 2024
Version 2.1, March 2024
Version 2.0, February 2024
Version 1.9, January 2024
Version 1.8, November 2023
Version 1.7, September 2023
Version 1.6, March 2023
Version 1.5, September 2022
Version 1.4, March 2022
Version 1.3, February 2022
Version 1.2, January 2022
Version 1.1, January 2021
Version 1.0, January 2020



Contents

Preface

This IOT LAB BOOK is primarily compiled to support the university courses on *'Internet of Things: Systems Design'* at both undergraduate and postgraduate levels. If you are taking either of these modules, please make sure you follow this lab book. It is compulsory to complete labs marked ● for both undergraduate and postgraduate level modules. The labs marked ● are compulsory for the postgraduate level.

This lab book guides you through a series of labs. Each lab has its objectives. It is expected that you should be able to complete each lab session within two hours (most of the time, much less). This lab book does assume that you have some amount of networking knowledge. Further, it is important to mention that IoT, by nature, is a broad subject. Therefore, in a few lab sessions, we cannot teach you all the topics in-depth. For example, Arduino programming use C/C++ programming languages. However, we do not expect you to be an expert on C/C++ to follow the lab session. However, if you have some background, you will find some known concepts in action and feel comfortable. If you have never seen C/C++ before, you will, of course, feel nervous and sometimes will feel lost.

Throughout the lab book, we provided explanations, external links, and references to reading material. Especially if you do not understand certain programming tasks such as C/C++ it may be worth reading those links. Further, these links will guide you to explore the universe of IoT on your own, beyond the labs we have provided here. Finally, we want to emphasise that this is not a programming course. Therefore, we do not try to teach you a particular programming language (though we try to provide as many links and references to develop your skills). It is up to you to develop the gaps in your knowledge by referring to the links we provided.

— Further information, links and references. We will use this structure throughout this lab book to provide you with additional information and references. They are marked in orange colour bar on the left-hand side. If you are not interested in exploring further, you are more than welcome to skip these small sections.

IoT Kit for BSc and MSc Labs and Coursework

In order to follow all the labs in this lab book, you need the following components. The component requirement for each lab is presented at the beginning of each chapter. You can also see the lab numbers that we use for the items on figure captions.

Raspberry Pi The Raspberry Pi is a series of small single-board computers. We have tested the labs with both Raspberry Pi3B+ and Raspberry Pi 4.

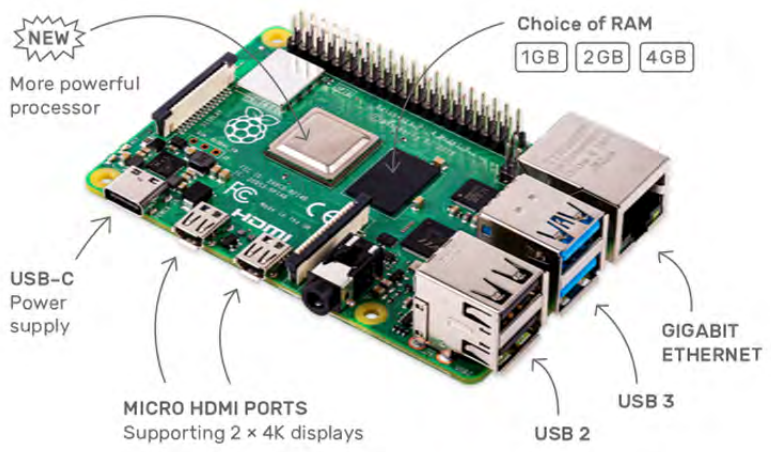


Figure 1: Raspberry Pi - Labs (2, 3, 4, 5, 6)

Notes We have provided the Raspberry Pi 4 version with 4GB RAM

- Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
- 1GB, 2GB or 4GB LPDDR4-3200 SDRAM (depending on model)
- 2.4 GHz and 5.0 GHz IEEE 802.11ac wireless, Bluetooth 5.0, BLE
- Gigabit Ethernet
- 2 USB 3.0 ports; 2 USB 2.0 ports.
- Raspberry Pi standard 40 pin GPIO header (fully backwards compatible with previous boards)
- 2 x micro-HDMI ports (up to 4kp60 supported)
- 2-lane MIPI DSI display port
- 2-lane MIPI CSI camera port
- 4-pole stereo audio and composite video port
- H.265 (4kp60 decode), H264 (1080p60 decode, 1080p30 encode)
- OpenGL ES 3.0 graphics
- Micro-SD card slot for loading operating system and data storage
- 5V DC via USB-C connector (minimum 3A*)
- 5V DC via GPIO header (minimum 3A*)
- Power over Ethernet (PoE) enabled (requires separate PoE HAT)
- Operating temperature: 0 – 50 degrees C ambient

GrovePi Plus GrovePi+ is an add-on board with 15 Grove 4-pin interfaces that brings Grove sensors to the Raspberry Pi. GrovePi+ is an easy-to-use and modular system for hardware hacking with the Raspberry Pi, no need for soldering or breadboards: plug in your Grove sensors and start programming directly. Grove is an easy-to-use collection of more than 100 inexpensive plug-and-play modules that sense and control the physical world. Connecting Grove Sensors to Raspberry Pi empowers your Pi in the physical world. With hundreds of sensors to choose from, Grove families, the possibilities for interaction are endless.

— Grove Ecosystem. Most of the Grove components can be connected to Raspberry Pi through GrovePi+ and can be connected Arduino micro-controller through Base Shield for Arduino. Most components are interchangeable between two platforms (i.e.,g Raspberry Pi and Arduino).

- **Sensors:** `http://wiki.seeedstudio.com/Sensor/`
- **Actuators:** `http://wiki.seeedstudio.com/Actuator/`
- **Displays:** `http://wiki.seeedstudio.com/Display/`
- **Communications:** `http://wiki.seeedstudio.com/Communication/`

Figure 2: GrovePi+ - Labs (2, 3, 4, 6)

Arduino Expansion Shield for Raspberry Pi B+ (V2.0) With the Arduino Adapter For Raspberry Pi, there's a way for the Raspberry Pi GPIO interface to adapt to Arduino pinouts, it is now possible to use the Pi together with vast Arduino shields and hardware/software resources: (i) Raspberry Pi GPIO interface to adapt to Arduino pinout, (ii) Compatible with Arduino UNO, Leonardo, easy to connect with various Arduino shields, (iii) XBee connector for connecting various XBee modules, and (iv) Sensor interface for connecting various sensors.

Figure 3: Arduino - Labs (1, 5, 6)

Base Shield for Arduino Arduino Uno is the most popular Arduino board so far; however, it is sometimes frustrating when your project requires many sensors or LEDs, and your jumper wires are in a mess. The Base Shield's purpose is to help you eliminate breadboard and jumper wires. With the rich grove connectors on the base board, you can conveniently add all the grove modules to the Arduino Uno! The pinout of Base Shield V2 is the same as Arduino Uno R3.

Figure 4: Base Shield - Labs (1, 5, 6)

Loudness (Sound) Sensor Sound Sensors can detect the sound intensity of the environment. The main component of the module is a simple microphone based on the LM386 amplifier and an electret microphone. This module's output is analogue and can be easily sampled and tested by a Seeeduino.

Figure 5: Loudness Sensor - Labs (6)

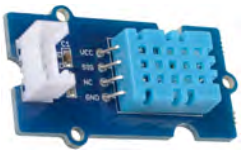
Light Sensor The light sensor integrates a photo-resistor (light-dependent resistor) to detect light intensity. The resistance of the photo-resistor decreases when the intensity of light increases. A dual OpAmp chip LM358 on board produces a voltage corresponding to light intensity (i.e. based on resistance value). The output signal is an analogue value. The brighter the light is, the larger the value. This module can be used to build a light-controlled switch, i.e. switch off lights during daytime and switch on lights during nighttime.

Figure 6: Light Sensor - Labs (1, 6)

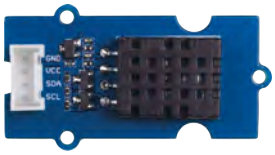
PIR Motion Sensor This sensor allows you to sense motion, usually human movement in its range. Simply connect it to Grove-Base shield and program it. When anyone moves in its detecting range, the sensor will output HIGH on its SIG (signal) pin.

Figure 7: PIR - Labs (1, 5, 6)

Temperature and Humidity This is a powerful sister version of our Grove - Temperature and Humidity Sensor Pro in figure 8a. It has a complete and more accurate performance than the basic version. The detecting range of this sensor is 5% RH - 99% RH, and -40C-80C. And its accuracy reaches up to 2% RH and 0.5C. A professional choice for applications that have relatively strict requirements.
The new Grove - Temperature & Humidity Sensor in figure 8b is based on the DHT20 sensor. The DHT20 is an upgraded version of the DHT11, compared with the previous version, the temperature and humidity measurement accuracy are higher, and the measurement range is larger. It features I2C output which means it is easier to use.



(a) DHT11



(b) DHT20

Figure 8: Temperature & Humidity - Labs (2, 3)

Servo Motor Servo is a DC motor with a gearing and feedback system. It is used in the driving mechanism of robots. The module is a bonus product for Grove lovers. We regulated the three-wire servo into a Grove standard connector. You can plug and play it as a typical Grove module now, without jumper wires clutter.

Figure 9: Servo Motor - Labs (1, 6)

LED Button LED Button is composed of Grove-Yellow Button, Grove-Blue LED Button and Grove-Red LED Button. This button is stable and reliable with a 100 000 times long life. With the built-in LED, you can apply it to many interesting projects. It is really useful to use the LED to show the status of the button. We use a high-quality N-Channel MOSFET to control the LED, ensuring high switching speed and low consumption.

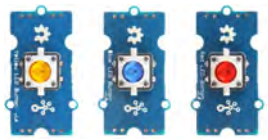


Figure 10: LED Button - Labs (2)

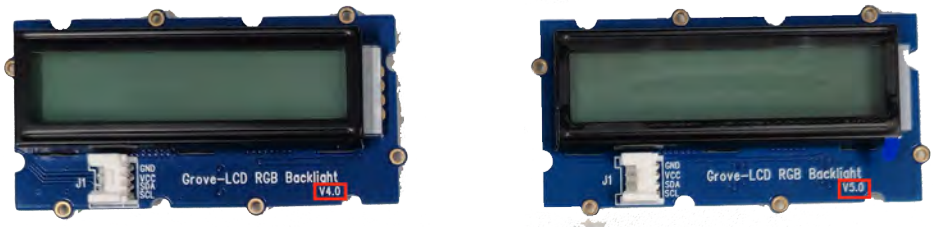
Ultrasonic Ranger The Ultrasonic Distance Sensor is an ultrasonic transducer that utilizes ultrasonic waves to measure distance. It can measure from 3cm to 350cm with an accuracy of up to 2mm. It is a perfect ultrasonic module for distance measurement, proximity sensors, and ultrasonic detectors. This module has an ultrasonic transmitter and an ultrasonic receiver, so you can consider it an ultrasonic transceiver. Familiar with sonar, when the 40KHz ultrasonic wave generated by the transmitter encounters the object, the sound wave will be emitted back, and the receiver can receive the reflected ultrasonic wave. It is only necessary to calculate the time from the transmission to the reception and then multiply the speed of the sound in the air (340 m/s) to calculate the distance from the sensor to the object.

Figure 11: Ultrasonic Ranger - Labs (2, 6)

Buzzer The buzzer module has a piezo buzzer as the main component. The piezo can be connected to digital outputs and emit a tone when the output is HIGH. Alternatively, it can be connected to an analogue pulse-width modulation output to generate various tones and effects.

Figure 12: Buzzer - Labs (2, 3, 4, 6)

LCD RGB Backlight Display This Grove enables you to set the colour to whatever you like via the simple and concise Grove interface. It takes I2C as a communication method with your microcontroller. So the number of pins required for data exchange and backlight control shrinks from 10 to 2, relieving IOs for other challenging tasks. Besides, Grove - LCD RGB Backlight supports user-defined characters. Want to get a love heart or some other foreign characters? Just take advantage of this feature and design it.



(a) LCD version 4.0 (b) LCD verion 5.0

Figure 13: LCD - Labs (2, 6)

Serial Bluetooth v3.0 Serial Bluetooth is an easy-to-use module compatible with the existing Grove Base Shield and designed for a transparent wireless serial connection setup. The serial port Bluetooth module is fully qualified Bluetooth V2.0+EDR(Enhanced Data Rate) 2Mbps Modulation with a complete 2.4GHz radio transceiver and baseband. It uses a CSR Bluecore 04-External single chip Bluetooth system with CMOS technology and AFH(Adaptive Frequency Hopping Feature). It has the smallest footprint of 12.7mm x 27mm. Hope it will simplify your overall design/development cycle.

Figure 14: Serial Bluetooth - Labs (5)

Raspberry Pi Camera v2 The Raspberry Pi Camera Module 2 can take hide definition videos and photographs. It allows connecting attachments such as lenses. While we can attach different cameras to Raspberry Pi, if case is not very specific, this camera should be preferred as there are libraries already available. We can do live image processing on Raspberry Pi via this camera as well.

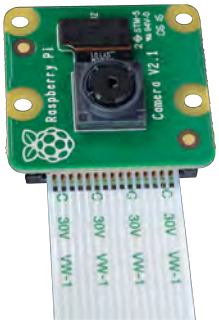


Figure 15: Camera - Labs (16)

SD Card Sandisk Class 10 Micro SD card reinstalled with the Raspberry Pi operating system.

Figure 16: SD Card - Labs (2, 3, 4, 5, 6)

Micro HDMI to HDMI Cable Connect a device that has a micro HDMI port to an HDMI compatible TV or monitor to share music, video or images up to 1080p resolution. Connect Monitor to Raspberry Pi.

Figure 17: Cable - Labs (2, 3, 4, 5, 6)

USB Keyboard and Mouse The keyboard has three in-built USB 2.0 type-A ports for powering other peripherals (such as the official Raspberry Pi mouse).

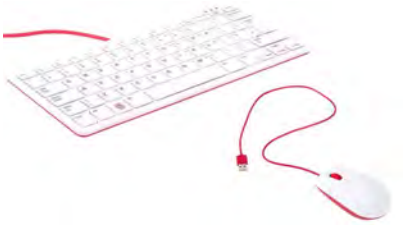


Figure 18: Keyboard - Labs (2, 3, 4, 5, 6)

Software Requirements

Arduino IDE The Arduino integrated development environment (IDE) is a cross-platform application (for Windows, macOS, and Linux) that is written in the programming language Java. It is used to write and upload programs to Arduino-compatible boards and, with the help of 3rd party cores, other vendor development boards (`https://www.arduino.cc/en/main/software`).

Figure 19: Arduino - Labs (1, 5, 6)

Node-RED Node-RED is an open source flow-based development tool for visual programming developed originally by IBM for wiring together hardware devices, APIs and online services as part of the Internet of Things. Node-RED provides a web browser-based flow editor, which can be used to create JavaScript functions. Elements of applications can be saved or shared for re-use. The runtime is built on Node.js. The flows created in Node-RED are stored using JSON. Node-RED can be used both on edge on in the cloud. We will use Node-RED as the edge IoT platform in our labs. (`https://nodered.org/`).

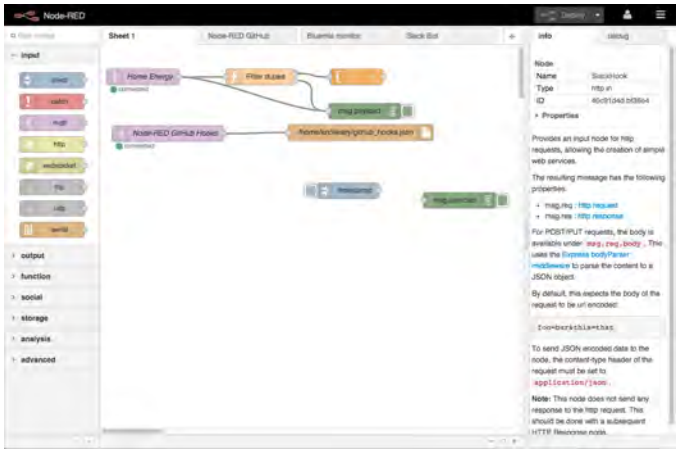


Figure 20: Node-RED - Labs (2, 3, 4, 5, 6)

Thingsboard ThingsBoard is an open-source IoT platform for device management, data collection, processing and visualization for your IoT projects. ThingsBoard can be used both on edge on in the cloud. In our labs, we will be using Thingsboard as the cloud IoT platform (`https://thingsboard.io/`).

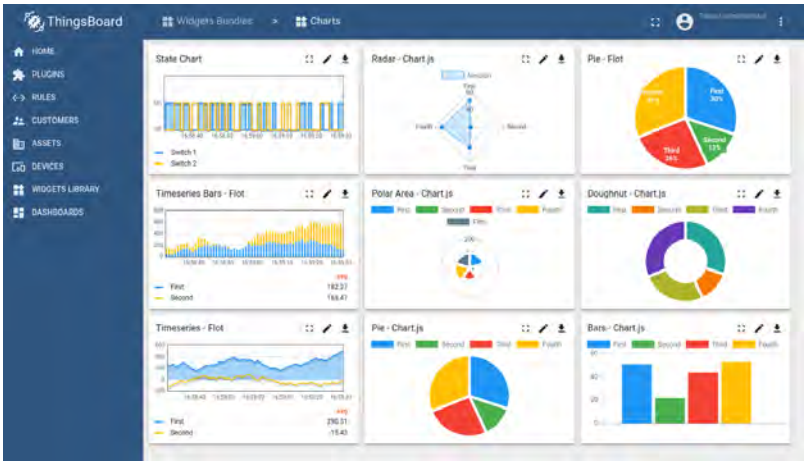


Figure 21: Thingsboard - Labs (3, 4, 6)

To Work From Home

Monitor You will need an external monitor to connect your Raspberry Pi to complete the labs. If you are an experienced user, you can SSH into your Pi and do the labs. According to your monitor's input, you may need a micro-HDMI to HDMI, DVI, or VGA (for older monitors) cable.

IoT Extension Pack for MSc Only Labs

LED Bar LED Bar is a 10-segment LED gauge bar with a built-in LED controlling chip. We use LED bars when we want to demonstrate a level of something. For example, we can show the remaining battery capacity, temperature level, distance, sound level etc. The bar colours range from red to green, while red indicates the highest level.

Figure 22: LED bar - Labs (8, 9, 14)

Rotary Angle Sensor Rotary Angle Sensor v1.2 is a potentiometer. We can set the resistance up to 10k Ohm. Thus, according to the input voltage potentiometer will generate an output. We can use potentiometers when we want to divide voltage to a certain degree. Thus we can use sensors that require 5V and 3.3V within the same setup.

Figure 23: Rotary Angle - Labs (8)

EMG Sensor We use electromyography sensors (EMG) to measure signals generated by our muscles. They are useful for detecting any behavioural anomalies in our muscles. Also, they are used to model the muscle behaviour of professionals such as athletes and footballers.

Figure 24: EMG - Labs (14)

Speaker Grove Speaker has a built-in potentiometer where we can set the sound level thus it offers power amplification. We can also set the frequency to generate a different tone. Speaker are useful when building alarm systems or distance indicators.

Figure 25: Speaker - Labs (12)

LED Socket Kit / LED Grove - LED Socket Kit allows us to control brightness of the LEDs via the built-in potentiometer. We might like to set the LED brightness in several cases such as reading a book, driving a car, or using a phone. Light-emitting diodes (**LED**) are one of the most efficient light sources. We currently have them in our phones, TVs, monitors, and simply anywhere we have a screen.

Figure 26: LED Socket Kit - Labs (10, 11)

RFID Reader RFID Reader is a module used to read uem4100 RFID card information with two output formats: Uart and Wiegand. It has a sensitivity with maximum 7cm sensing distance.

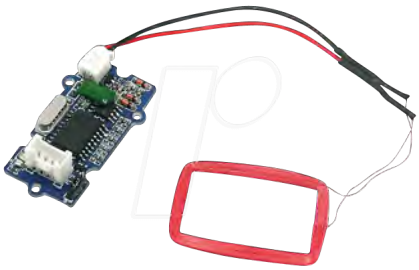


Figure 27: RFID Reader - Labs (11)

NFC Reader NFC Tag is a highly integrated Near Field Communication Tag module,this module is I2C interface,which base on M24LR64E-R,M24LR64E-R have a 64-bit unique identifier and 64 -Kbit EEPROM.Grove - NFC Tag attach an independent PCB antenna which can easily stretch out of any enclosure you use, leaving more room for you to design the exterior of your project.

Figure 28: NFC Reader - Labs (11)

RDIF Tags Combo A set of 5 RFID tags, consisting of 3 key rings and 2 cards . Each chip has a unique 64-bit code. Identifiers work a short distance from the device. The carrier frequency is 125 kHz.

Figure 29: RDIF Tags Combo (11)

NFC Tags NFC Round Cards NFC 215 Card Tag Compatible with TagMo and Amiibo, 504 Bytes Memory Fully Programmable for NFC-Enabled Devices

Figure 30: NFC Reader - Labs (11)

Additional Components for BSc and MSc Coursework

Thumb Joystick Joystick modules generate analogue signals that simulate the movements of the cartesian coordinate system. It also has a push button that we can use as input. Its output range is smaller than common joysticks. Thus, most of the time, we map the output to a certain range.

Figure 31: Thumb Joystick - Labs (12)

Gesture Sensor for Arduino Gesture is based on PAJ7620U2 that integrates gesture recognition function with general I2C interface into a single chip. It can recognize 9 gestures including moving up, moving down, moving left, moving right, etc with a simple swipe by your hand.

Figure 32: Gesture Sensor

UART Wifi UART WiFi is a serial transceiver module featuring the ubiquitous ESP8285 IoT SoC. With integrated TCP/IP protocol stack, this module lets your micro-controller interact with WiFi networks with only a few lines of code. Each ESP8285 module comes pre-programmed with an AT command set firmware, meaning you can send simple text commands to control the device. The SoC features integrated WEP, WPA/WPA2, TKIP, AES, and WAPI engines, can act as an access point with DHCP, can join existing WiFi networks and has configurable MAC and IP addresses.

Figure 33: UART WiFi

Triple Colour E-Ink Display Triple Color E-Ink Display 2.13” is a screen that can still be displayed after power off, we call it E-Paper(electronic paper) or E-Ink. The display is a TFT active matrix electrophoretic display, with interface and a reference system design.The 2.13 inch active area contains 212x104 pixels, and has 1-bit white/black and 1-bit red full display capabilities.

Figure 34: Triple Colour E-Ink Display

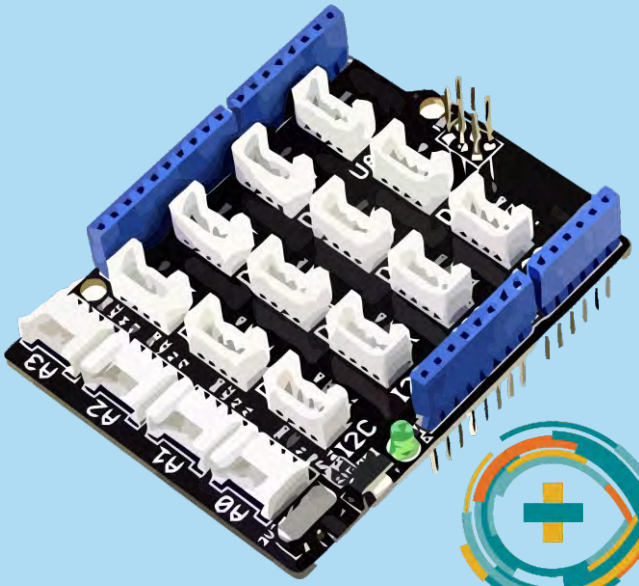
Blueseeed (BLE) Blueseeed is a cost-effective, low-power, true system-on-chip (SoC) for Bluetooth low energy applications. It enables robust BLE master or slave nodes to be built with very low total bill-of-material costs. It is based on TI CC2540 chip, which has AT command support.

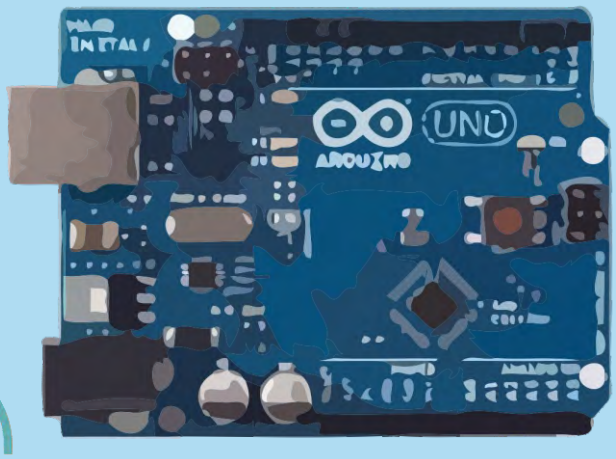
Figure 35: Blueseeed

Hardware Requirements Summary

Hardware Components	Lab 1	Lab 2	Lab 3	Lab 4	Lab 5	Lab 6	Lab 7	Lab 8	Lab 9	Lab 10	Lab 11	Lab 12	Lab 13	Lab 14	Lab 15	Lab 16
Raspberry Pi (3B+, 4)		✓	✓	✓	✓	✓	✓		✓							✓
GrovePi Plus		✓	✓	✓		✓			✓							
Arduino Expansion Shield for RPi	✓				✓	✓		✓		✓	✓	✓		✓		
Base Shield for Arduino	✓				✓	✓		✓		✓	✓	✓		✓		
Loudness (Sound) Sensor						✓			✓	✓						
Light Sensor	✓					✓		✓								
PIR Motion Sensor	✓				✓	✓										
Temperature/Humidity (V11,V20)		✓	✓	✓		✓										
Servo Motor	✓					✓										
LED Button (Yellow/Blue/Red)	✓					✓										
Ultrasonic Ranger		✓				✓		✓	✓	✓						
Buzzer		✓	✓	✓		✓			✓							
LCD Backlight Display (V4,V5)		✓				✓				✓	✓	✓				
Serial Bluetooth v3.0					✓	✓										
Raspberry Pi Camera v2																✓
SD Card		✓	✓	✓		✓	✓		✓							✓
Micro HDMI to HDMI Cable					✓		✓	✓	✓							✓
USB Keyboard and Mouse	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Monitor	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
LED Bar								✓	✓					✓		
Rotary Angle Sensor								✓								
EMG Sensor														✓		
Speaker												✓				
LED Socket Kit / LED										✓	✓					
RFID Reader											✓					
NFC Reader											✓					
RFID Tags Combo											✓					
NFC Tags											✓					
Thumb Joystick												✓				
Blueseeed (BLE)										✓						
Gesture Sensor for Arduino																
UART WiFi																
Triple Colour E-Ink Display																

Table 1: Required Hardware for Each Lab Tutorial





ARDUINO
OPEN-SOURCE
COMMUNITY

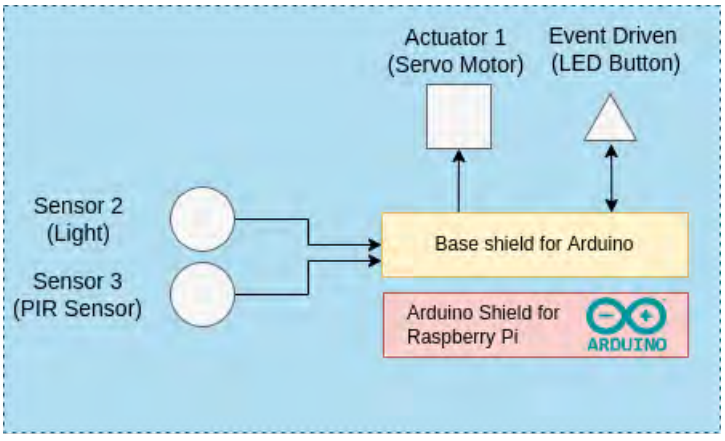
1. Micro-Controller Programming

Objective

- Learn how to program a Micro-controller.
- Learn how to read data from different types of sensors.
- Learn how to handle events.

Lab Plan

In this lab, you will learn how to connect three different sensors into a microcontroller (i.e., Arduino) through a Grove compatible base shield.



Required Hardware Components

- Microcontroller board (e.g., Arduino)
- Grove HAT (Base Shield for Arduino)
- Sensor 2 (Light Sensor)
- Sensor 3 (Passive Infrared (PIR) Motion)
- Actuator 1 (Servo Motor)
- Event-Driven (LED Button)
- USB Cable (Arduino to PC)

Tasks

We are going to use Arduino IDE for the first time to program the Arduino, at the end of this lab you'll be able to read data and produce outputs from several sensors.

1. First, open **Arduino IDE** on the lab. machine. All lab. machines have Arduino IDE already installed. If you are using your own computer you will need in install it by yourself. Instructions on how to install Arduino IDE on you computer is out of scope of this lab document. However, you may find following link useful on how to install Arduino IDE on you computer.

   — How to Install the Arduino Desktop IDE on your own computer. The Arduino Software (IDE) allows you to write programs and upload them to your board. At the Arduino website: "`https://www.arduino.cc/en/Main/Software`", you will find two options:

   Online IDE If you have a reliable Internet connection, you should use the online IDE (Arduino Web Editor). It will allow you to save your sketches in the cloud. Also, you will always have the most up-to-date version of the IDE.

   Desktop IDE If you would rather work offline, you should use the latest version of the desktop IDE.

2. Open Arduino IDE. Connect your Arduino to the computer and wait for few seconds. Then in Arduino IDE click **Tools** → **Board** → **Arduino Leonardo** as shown in Figure 14.1. Finally set the correct **PORT** from the same section as shown in Figure 14.2

3. Now place **Base Shield for Arduino** on top of Arduino microcontroller.

4. Then we can start connecting sensors and actuator to our Arduino. Plug them in accordingly:
   - **LED Button** → **D3 port**
   - **Motion Sensor (PIR)** → **D2 port**
   - **Light Sensor** → **A3 port**

5. Before starting coding in Arduino IDE, upload the empty sketch (by clicking the top left arrow icon ) to verify that you set up the Arduino correctly.

6. Get the related sketch from `https://gitlab.com/IOTGarage/iot-lab-book.git`. The code is pretty simple and comes with explanations. Please do not copy paste but type the code by yourself and read it carefully to understand how does it work.

7. Now **verify** (the top left tick icon ) and **upload** the given code. After you upload it the LED button should start blinking. When you press and hold the button, the LED should stop blinking. Your final setup should be as in Figure 1.3.

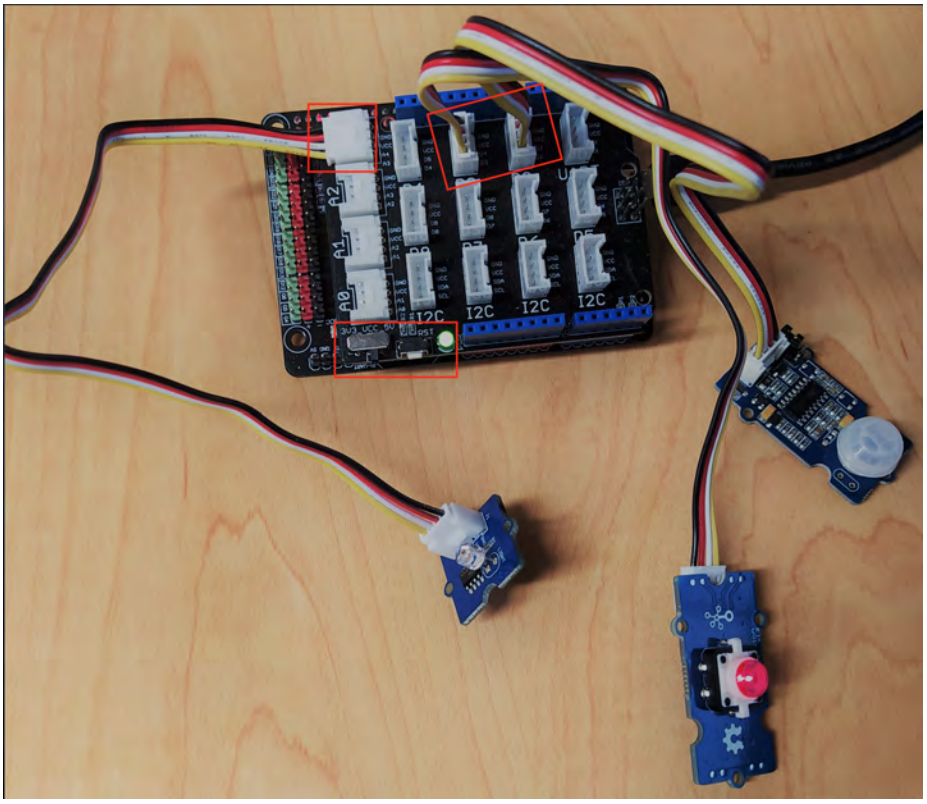


Figure 1.3: Final Setup

8. Then open your **Serial Monitor** from the top right the magnifying glass icon  . The serial monitor is the tether between the computer and Arduino.

   — No Output in the serial monitor?. If you are able to open "Serial Monitor" but there is no output, make sure the Base Shield is properly.

   Check the "Green" light The LED on base shield must be illuminated in green colour like shown in figure 1.3

   Turn on "Base Shield" If there is no illuminated green light on base shield then switch the black button found on the bottom left of base shield as shown in figure 1.3.

9. According to the code, when the button is pressed the **motion sensor** is working, otherwise the **light sensor** is working. So, you can switch between sensors by pressing the button.

10. Now push the button to get different outputs. If you have done everything correctly, you should have a similar output in your serial monitor as in Figure 1.4.

11. Congratulations, now we should have working Arduino operating in two different modes with button indicating in what mode or program is running.

12. Now we will learn how to run Grove Analog Servo.

13. First connect your servo motor to **D5 port**. Then get the code from `https://gitlab.com/IOTGarage/iot-lab-book.git`. Upload the code and push the button to run the servo. Try to understand what the code does.

14. **Use Case Scenario (Optional)**: Think this as it is your **FINAL TASK**. Imagine that servo

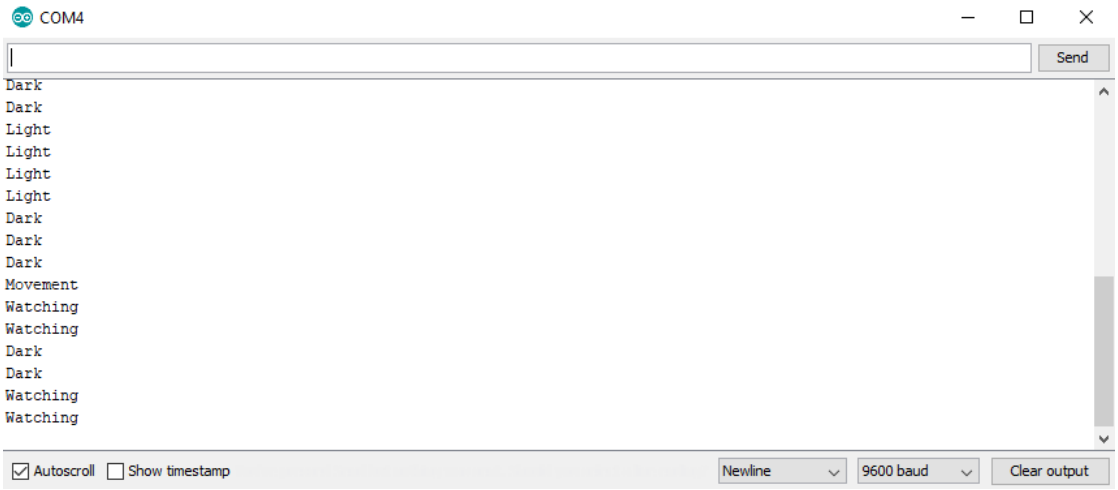
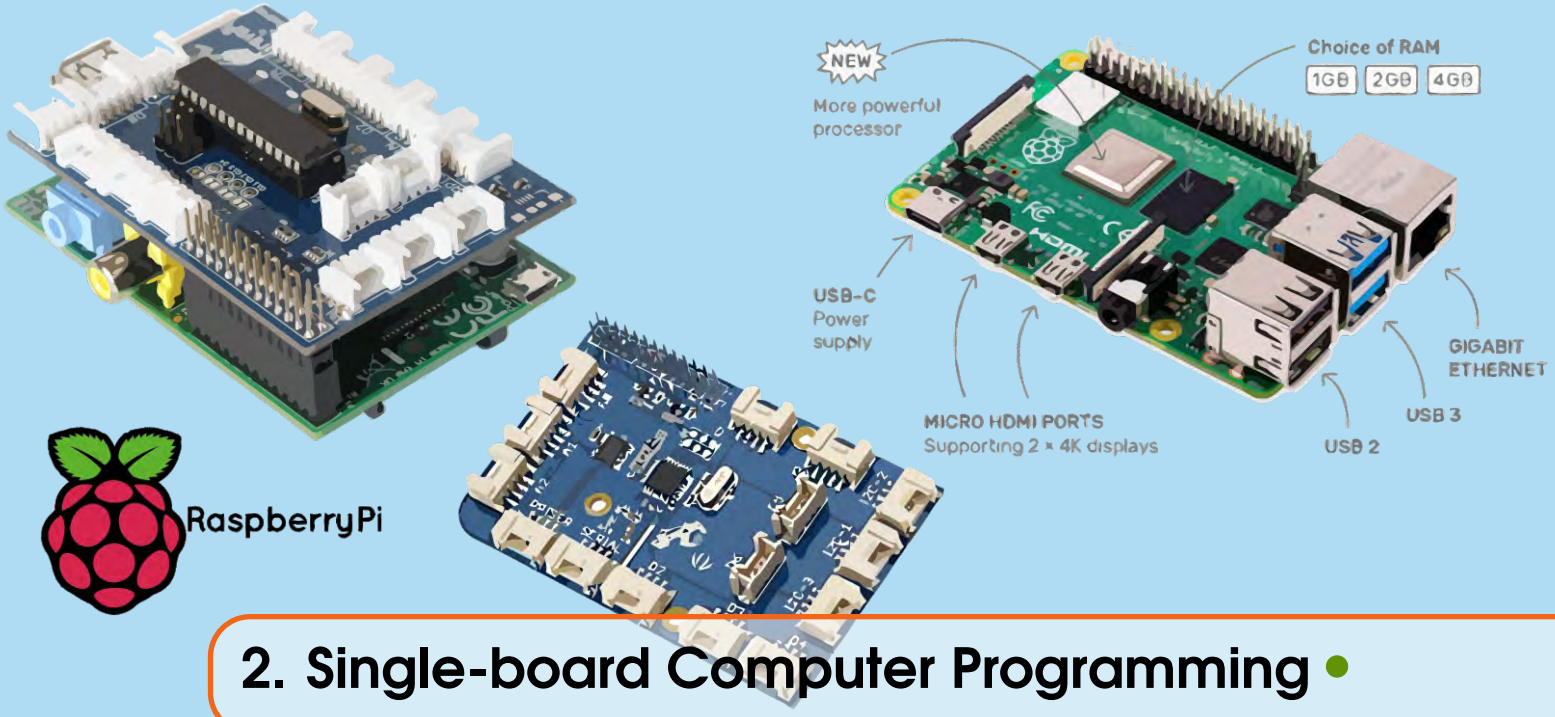


Figure 1.4: Serial Monitor

motor is controlling your door of your private room. You want to close the door when there is **dark** or there is any **movement**. The servo will close your door when the shaft of the servo moves back and forth across **180** degrees. You will write your own code.

— Further Reading.

- To learn more about Grove Sensors visit `http://wiki.seeedstudio.com/Sensor/`
- To learn more about Arduno Programming visit `https://www.arduino.cc/en/Tutorial/HomePage?from=Main.Tutorials`

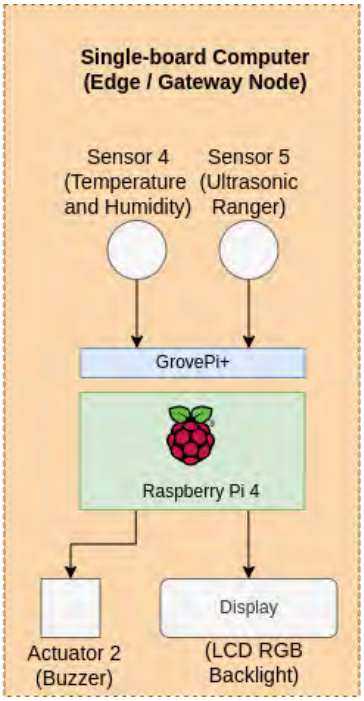


2. Single-board Computer Programming

Objective

- Learn how to program a Single-board Computer (Raspberry Pi model 4B with 4GB RAM)
- Learn how to read data from few different sensors
- Learn how to program actuators
- Learn how to program a display

Lab Plan



Required Hardware Components

- Raspberry Pi
- Grove Pi+
- Grove sensors: Ultrasonic Ranger, Temperature and Humidity sensor
- Buzzer
- LCD Backlight display
- SD Card with OS installed on it (we'll work on the Raspberry Pi OS)
- Display and HDMI cable
- Keyboard and mouse
- Power supply

Setup of the RaspberryPi

1. Start by plotting the **SD card** into the **SD card slot**.

2. Next plug in peripherals into the USB ports and connect Raspberry Pi to the monitor using the necessary (i.e., micro HDMI - DVI) cable. **DO NOT** power up the Pi.

3. Connect your Grove Pi+ to your Raspberry Pi.

4. Now we can connect our sensors to Grove Pi+. Connect sensors as shown in below:
   - **LCD Backlight → I2C-2 port**
   - **Temperature & Humidity Sensor (DHT) → D4 port**
   - **Ultrasonic Ranger → D3 port**
   - **Buzzer → D8 port**

5. Power up your Raspberry Pi. You should see the red light turn on. As the device is booting you'll see raspberries displayed on your monitor.

6. After the OS configured connect to the **Wi-Fi** network.

7. Check if Raspberry Pi shows the correct time and date. If not, set the correct date and time (check Chapter 16).

8. Check if **I2C** is enabled by clicking on the Raspberry icon(top left corner ). Then from **Preferences** choose **Raspberry Pi Configuration** and then **Interfaces**.

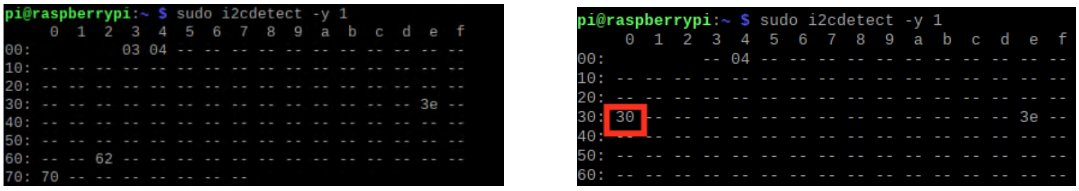
9. The Raspberry Pi is already **preconfigured** to run Node-RED without any issues. Run the below code to confirm that your Pi works fine:

```
sudo i2cdetect -y 1
```

   If you can see **04** in your output as shown in Figure 2.1, it means that Raspberry Pi is able to detect GrovePi+ which means you can move to the next section (**Programming with Node-RED**). It is also important to note that there are two versions of LCD available in the lab-kit, as discussed in Preface IoT Kit section. Figure 2.1a shows output for LCD version 4.0 whereas figure 2.1b shows output in case you have LCD version 5.0.

10. If you **DON'T** see **04** please reconnect your GrovePi+ and try again. If you still do not see it, run the following codes and try again. This step might take around 10 minutes.

```
bash <(curl -sL https://raw.githubusercontent.com/node-red/linux-
    installers/master/deb/update-nodejs-and-nodered)
curl -kL dexterindustries.com/update_grovepi | bash
```



(a) LCD v4.0 (b) LCD v5.0

Figure 2.1: Verifying GrovePi+ Connection for different LCD versions

```
sudo reboot
```

Programming with Node-RED

11. Download the `dht.py` python script to your **/home/pi** from GitLab.

12. Open your terminal (Ctrl + Alt + T) and run the following to make your python script executable.

```
sudo chmod 755 dht.py
```

> **— Different Sensor?.** Do you have DHT20 Sensor (black-colour)?
> **Plug-in I2C Port** Plug in the DHT20 Sensor (black-colour) to an I2C port (I2C-1) instead of digital port (D4).
> **Download** Download dht20.py python script to your **/home/pi** from GitLab.
> **Change Access Control** Replace the file name to "dht20.py" in the above command.

(a) DHT11 (b) DHT20

13. Check if npm version is 6+, if not, redo the above steps:

```
npm -v
```

14. Everything should be ready to start programming with **Node-RED**. Open the applications menu by clicking on the Raspberry icon in top left corner. Then from programming tools choose **Node-RED** as shown in Figure 2.3.

15. You can ignore any warning messages.

16. New terminal named **Node-RED console** should be opened. Now open the browser and type in the address `http://localhost:1880` to access the Node-RED local server.

17. Delete if there are any nodes on the workspace, delete them all and click **on deploy** once.

18. Click on the top right corner ☰. Select **Manage Palette** as shown in Figure 2.4. Then from the **install** section, install the following package **node-red-node-daemon** if it is already **NOT** installed. This may take more than a minute.

19. Drag the **daemon** node from the sidebar to the workspace area. Double click on it, this will open the properties window, **configure** it as you see in Figure 2.5.

Figure 2.3: Starting the Node-RED service

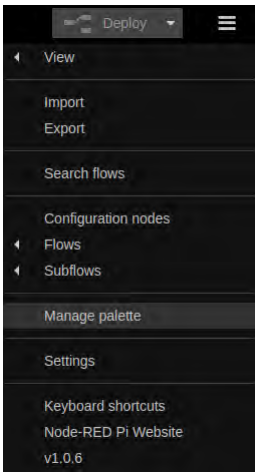


Figure 2.4: Manage Palette

20. Now let's see if our sensor works correctly. Choose output node called **debug** from the sidebar. This node is primarily used to check outputs of flows. Connect **debug node** to the **daemon node** (if you configure it as given above, name should be changed into python) as shown in Figure 2.6.

21. We've created the most basic flow, to make it work now click **Deploy** from top right corner. Then in the right window click on the **debug icon** to open debug window. If everything setup correctly it should look like as in Figure 2.7.

22. After you observe your sensor data in Node-RED, close it by **removing the connection** between nodes, then deploy again.

Creating thermometer with display

23. This time fetch `lcd.py` from GitLab to your **/home/pi**.

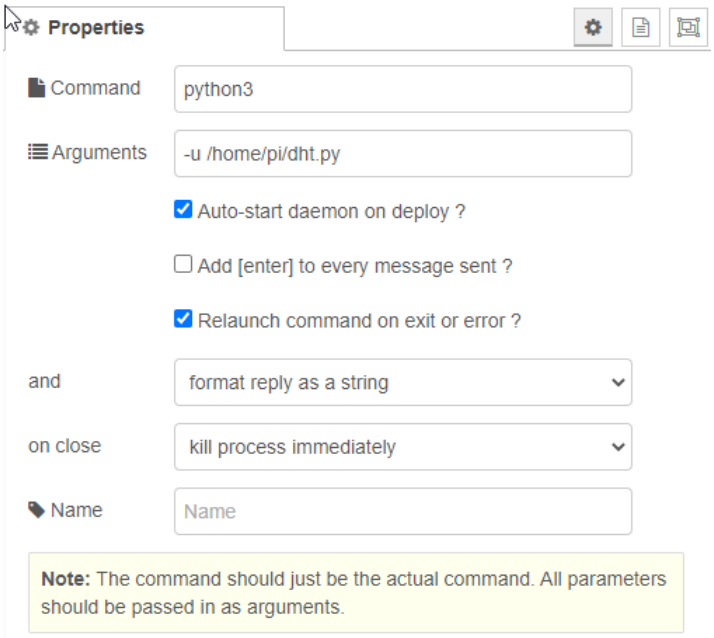


Figure 2.5: Node Settings

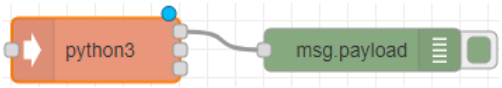


Figure 2.6: Simple Flow Example

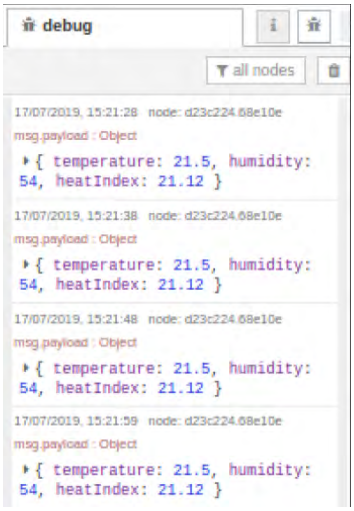


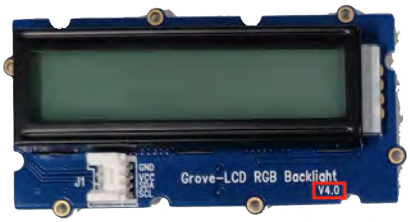
Figure 2.7: Debug window that shows temperature and humidity values

24. Open your terminal (Ctrl + Alt + T) and run the following to make python script executable.

```
sudo chmod 755 lcd.py
```

— No output on screen or error reading temperature?. Do you have a different screen, Grove-LCD RGB Backlight V5.0 instead of V4.0? Or do you have DHT 20 instead of DHT11?
What Happened? The lcd.py file was developed for DHT11 on LCD v4.0. Follow the

steps below if you have different hardware.
Download Download dht#_lcd#.py python script to your **/home/pi** from GitLab.
Change Access Control Replace the file name to "dht#_lcd#.py" in the above command.
Example For LCD v5.0 with DHT20, use "dht20_lcd5.py".



(a) LCD version 4.0

(b) LCD verion 5.0

25. Then configure your **daemon node** by changing `dht.py` to `lcd.py`
26. Then `deploy` again.
27. You should be able to see the temperature value on your LCD screen.

Programming Buzzer and Ranger

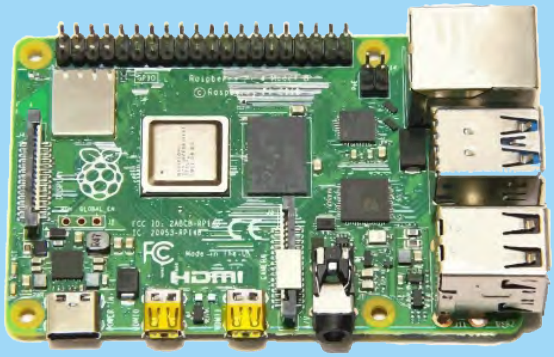
28. In this section, we will learn how to program buzzer and ultrasonic ranger. It is a simple device that we can use to respond to the sensor readings.
29. Fetch `buzzerRanger.py` from GitLab to your **/home/pi**.
30. Open your terminal (Ctrl + Alt + T) and run the following command to make your python script executable.

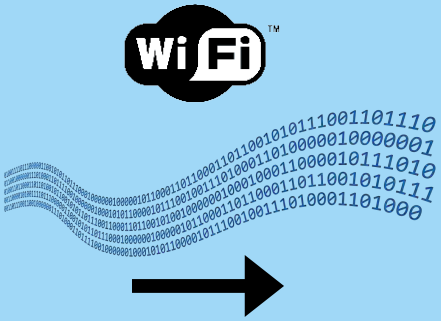
```
sudo chmod 755 buzzerRanger.py
```

31. Then configure your **daemon node** by changing `lcd.py` to `buzzerRanger.py`
32. Then `deploy` again.
33. You should be able to see distance on Node-RED debug space, if the distance is below 10 cm the buzzer will ring.
34. **Use Case Scenario (Optional)**: Think this as it is your **FINAL TASK**. Imagine you are cooking some dinner in your kitchen and you also have a cat (a really curious one). You want to leave the kitchen but also know if the cat gets closer to the oven. Change your setup accordingly.
Hint: The buzzing sound should indicate how far is your cat. Discrete means far, continous means close. Also display the temperature and the distance on your LCD. You need to write your own python script to complete this task.

— Further Reading. .

- To learn more about Node-RED see: `https://nodered.org/docs/`
- To learn more about RaspberryPi visit `https://www.raspberrypi.org/`





RaspberryPi

ThingsBoard

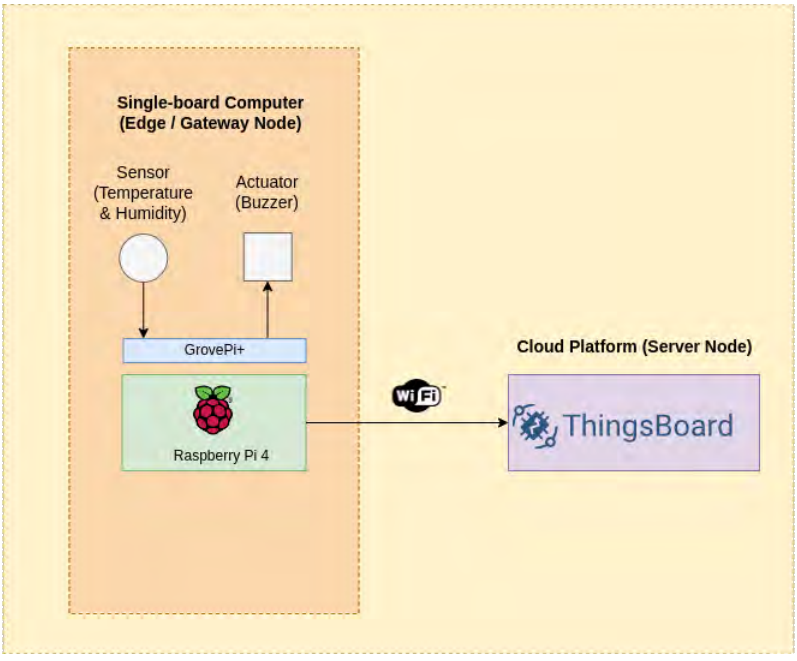
3. Posting Data to an IoT Cloud Platform

Objective

- Learn how to post IoT data on a cloud IoT platform

Lab Plan

This lab explains how to configure Thingsboard so that we can send data from Raspberry Pi via the MQTT node that we install to Node-RED. We will observe live data on Thingsboard and create customised widgets on the Thingsboard dashboard.



Required Hardware Components

- SD Card with OS installed on it (we'll work on the Raspberry Pi OS)
- Display and HDMI cable
- Keyboard and mouse
- Power supply

Additionally we'll use:

- Grove Pi+
- Grove sensor: Temperature and Humidity sensor
- LCD Backlight display

Posting Data to Thingsboard Dashboard

1. **To do this lab. you need to be at the end of 20th step of lab. 2 first**. However, this time you only need the connect the **temperature&humidity sensor**. Please do all the required steps and get data from your sensor in **Node-RED before starting this lab**.

2. In this lab. we will once again use Node-RED. But this time we will use both Watson IoT node and MQTT node. **MQTT** is a lightweight messaging protocol which enables fast and reliable communication between different devices. This is why it's so vastly popular in IoT solutions.

3. To begin we have to configure our Thingsboard. Go to Thingsboard website **https://thingsboard.cs.cf.ac.uk/login** and login via the given group username and password.

4. Now we have to add a **new device**. Go to the **Devices** page by selecting from left the panel . Click the plus icon from top right and select **Add new device**.

5. Then, configure your device as gateway as seen in Figure 3.1.

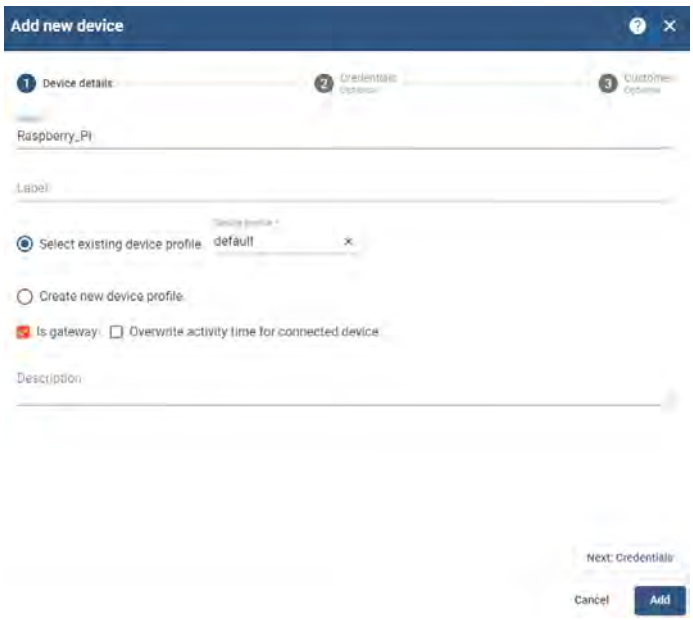


Figure 3.1: Adding New Device

6. Then click on **Next: Credentials** button to provide access token.

7. Provide your own random access token as shown in Figure 3.2

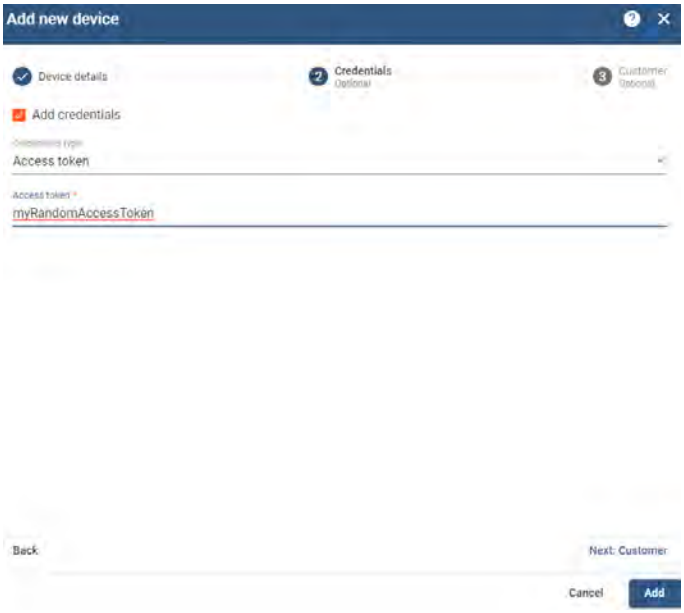


Figure 3.2: Random Access Token

8. Now select (click on the name) your device and copy the access token as show in Figure 3.3. Please **note** this token to somewhere as we will need it later on.

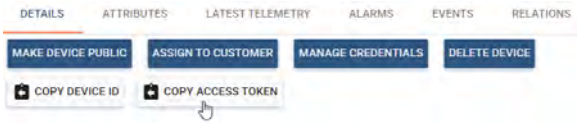


Figure 3.3: Copy Access Token

Node-RED Programming

9. Drag the **mqtt out** node (see Figure 3.4) to the workspace. Then, connect the **daemon** node to the **MQTT** node as in Figure 3.5.

Figure 3.4: MQTT out node.

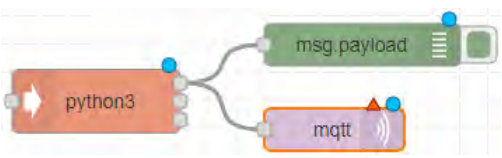


Figure 3.5: Node-RED setup.

10. Now we will configure the **mqtt** node.

11. First, double-click on **mqtt** node.

12. Then, click on the **pen icon** which is next to the server input.

13. Configure it as it is shown in Figure 3.6.

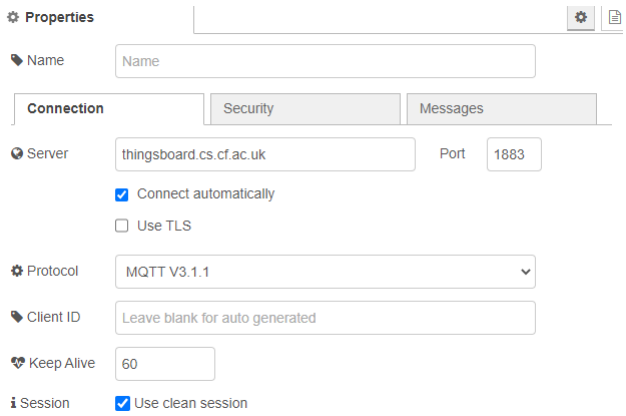


Figure 3.6: MQTT Configuration

14. Click on the security tab and provide the **access token** of the device as username that you have copied from the ThingsBoard website as in Figure 3.7. Click **Update** in top right corner of window.

Figure 3.7: Providing Access Token

15. Lastly in **Topic** window, enter **v1/devices/me/telemetry** as in Figure 3.8. This will provide Thingsboard with where data sent from Raspberry Pi has to exactly go. Click **Done** in top right corner of window and deploy your project/flow via top right **Deploy** button.

Figure 3.8: MQTT Topic

Observing Live Data in ThingsBoard

16. Now, we need to create a dashboard to use built-in ThingsBoard widgets to observe our live data. In the left panel select **Dashboards** . Then create a new dashboard in a similar way that creating a device. Name your dashboard as **RaspPiDashboard**.

17. After creating your dashboard, select your created device. Then click on **Latest telemetry**. Tick both temperature and humidity variables as shown in Figure 3.9.

Figure 3.9: Latest Telemetry

18. Then, click on **Show on widget**. Here, we will leave the widget type as default (which should be Cards). Then click on **Add to dashboard** configure as shown in Figure 3.10.

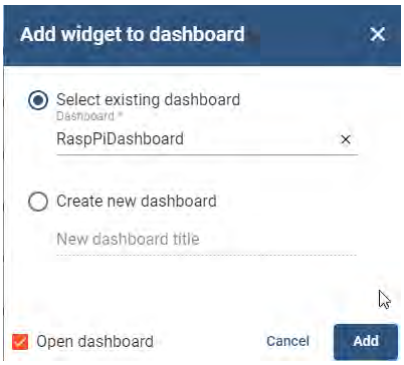


Figure 3.10: Dashboard Settings

19. Finally, click on **Add** to add that widget to your dashboard.

20. Now, you should be able to observe your live sensor data in ThingsBoard as it is shown in Figure 3.11. We have selected the default widget, you might want to try other widgets and select the one that you desire.

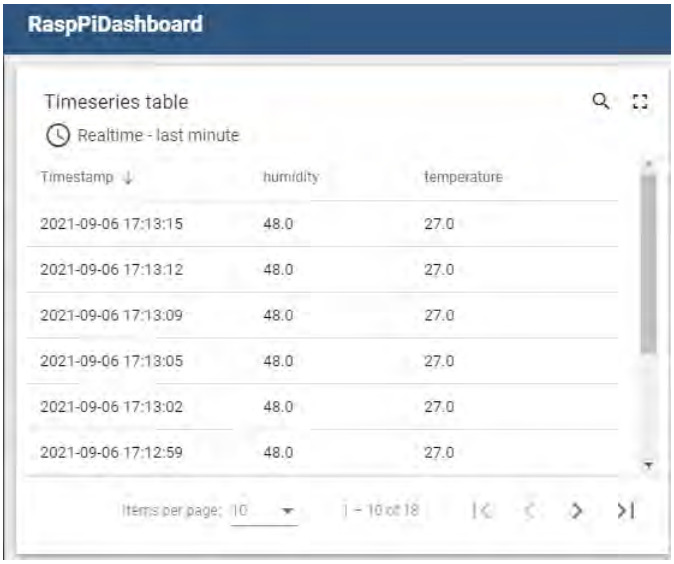


Figure 3.11: Observing Live Data

21. If you would like to test yourself further, **go for the final task**. Otherwise, remove all nodes from the workspace and deploy one more time. Then, in the terminal type the following:

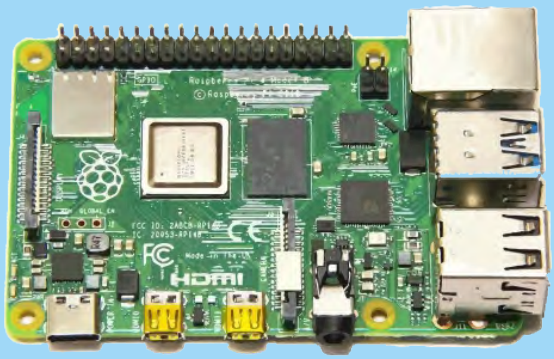
```
node-red-stop
sudo shutdown now
```

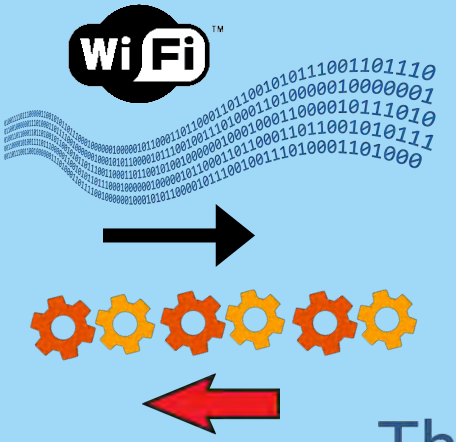
    Now, you can safely unplug your Pi after 5 seconds, if you will not do the final task.

22. **Use Case Scenario (Optional)**: Think this as it is your **FINAL TASK**. Imagine a meteor is approaching to the Earth. You would like the measure how it will effect the temperature of the earth. If temperature rises above a certain degree the buzzer will ring. You will visualize all the process via ThingsBoard.
    Hint: Your one hand will be a meteor, while the other one will hold the temperature sensor to increase the temperature value.

— Further Reading.

- `https://thingsboard.io/iot-use-cases/`





RaspberryPi

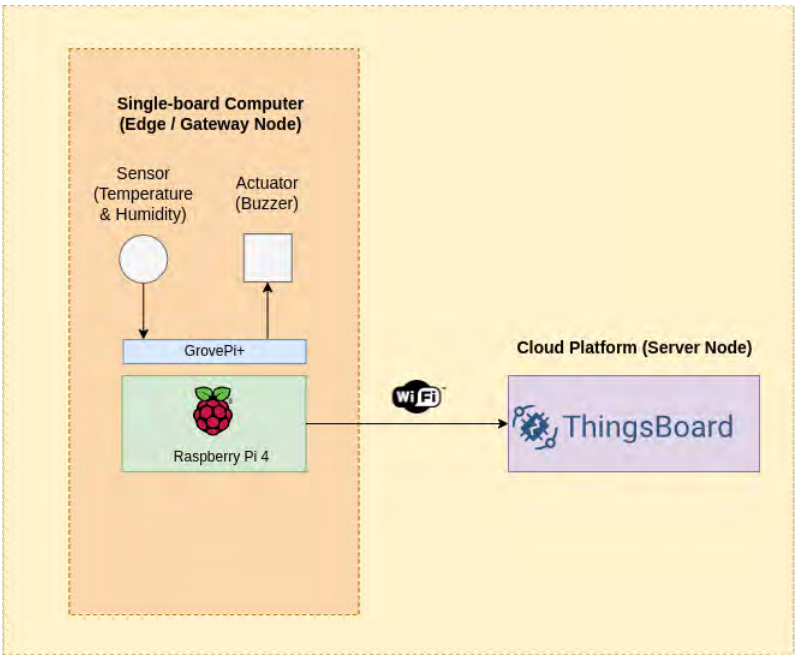
ThingsBoard

4. Connecting an IoT Gateway to an IoT Cloud

Objective

- Learn how to configure an IoT cloud platform
- Learn how to connect a gateway (Raspberry Pi) to IoT cloud platform (ThingsBoard)
- Learn how to send data from edge gateway to cloud
- Lean how to send commands/data from cloud to edge gateway

Lab Plan



Required Hardware Components

- Raspberry Pi 4
- SD Card with OS installed on it (we'll work on the Raspberry Pi OS)
- Display and HDMI cable
- Keyboard and mouse
- Power supply

Additionally we'll use:

- GrovePi+
- Grove sensors: Temperature&Humidity Sensor (DHT11)
- Buzzer

Connecting to ThingsBoard Cloud Platform via Python Script

1. We learnt how to send telemetry data to cloud, now, in this lab, we will establish a dual-way communication with the cloud.

2. We will be sending temperature and humidity data to ThingsBoard while controlling our buzzer from there.

3. To do this lab, you **MUST** have completed the **lab 3**. So, please ensure you can send your temperature and humidity data to the **ThingsBoard** without any issues.

4. Check if your sensors/actuators are connected to the correct ports.
   - **Temperature & Humidity Sensor (DHT) → D4 port**
   - **Buzzer → D8 port**

5. After testing it you can close Node-RED, because we will be using a **Python 3 script**. If Node-RED is running, stop it via running the following in the Pi's terminal:

```
node-red-stop
```

6. Now, get the Python 3 script named **cloud.py** from GitLab and place it to **/home/pi**.

7. Now, run the following to provide required permissions:

```
sudo chmod 755 cloud.py
```

8. We need a Python MQTT package to establish a communication with ThingsBoard. Hence, install one via running the following:

```
pip3 install paho-mqtt
```

9. Now, go to **https://thingsboard.cs.cf.ac.uk/**, create a new device and name it as **Buzzer Demo Device**.This time, do **NOT** declare your device as gateway. Do not provide any access token, this token will be generated automatically.

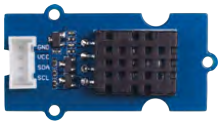
```
ACCESS_TOKEN = 'KV8ua9VXNu9cOQ8Op4DS' # <== Insert your own access token
    here.
```

10. Now run the code, via entering the following to Pi's terminal:

```
    python3 cloud.py
```

— Which DHT sensor are you using?. Do you have DHT 20 instead of DHT11?
Download Download cloud_dht20.py python script to your **/home/pi** from GitLab.
Example Run the above command with file name "cloud_dht20.py".

(a) DHT11



(b) DHT20

11. You should see the following output that is shown in Figure 4.2

```
pi@raspberrypi:~ $ python3 cloud.py
Connected with result code {'session present': 0}
Temperature: 24°C, Humidity: 64%
Success
Success
```

Figure 4.2: Cloud.py Output

12. The `{'session present': 0}` means that communication is established without any problem. The two `Success` messages mean that you successfully have sent the temperature, humidity, and state of the buzzer (either true (on) or false (off)).

13. Observe these data on the ThingsBoard as well. You data should be similar as in Figure 4.3.

Details | Attributes | Latest telemetry | Alarms | Events | Relations | Audit Logs

Latest telemetry

Last update time	Key	Value
2021-09-09 13:37:56	humidity	61.0
2021-09-09 13:37:56	State	false
2021-09-09 13:37:56	temperature	25.0

Figure 4.3: Cloud.py Output

14. Add a widget for the temperature and humidity data as you have done in the previous lab. This time, add a time-series chart.

15. Now, we will control our buzzer via ThingsBoard. First, please terminate (CTRL + C) the Python Script.

16. Then, go to your dashboard, from click-on the **pen icon** on the bottom right. Then, click on the **plus icon**. Finally, select the **file icon** to create a new widget. We will send Remote

Procedure Calls (RPC) from server side (ThingsBoard in this case) to our client (Raspberry Pi) to control the buzzer.

17. Control widgets are the widgets that can send RPC. So, select **control widgets** as shown in Figure 4.4.

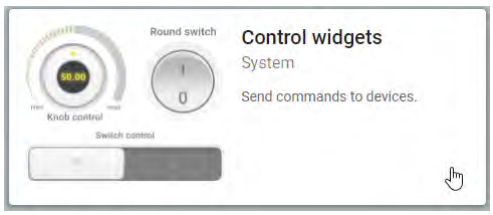


Figure 4.4: Control Widget

18. Now add the **switch control widget** to your dashboard as shown in Figure 4.5.

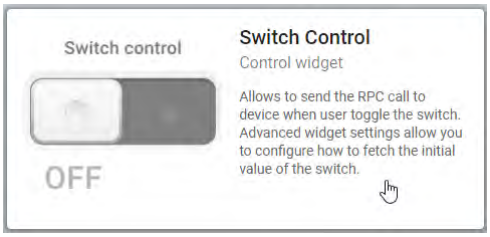


Figure 4.5: Switch Control

19. Select the **Buzzer Demo Device** as your target device. If you can't find that, add yourself. The type should be set to **Device Type**. Then you can add your widget. After adding, it can generate a **Request Timeout** error which you can ignore.

20. If you have done everything correct, your dashboard should look like as in Figure 4.6.

Figure 4.6: Dashboard

21. Now, we are ready to run our script. Run the `cloud.py` again.

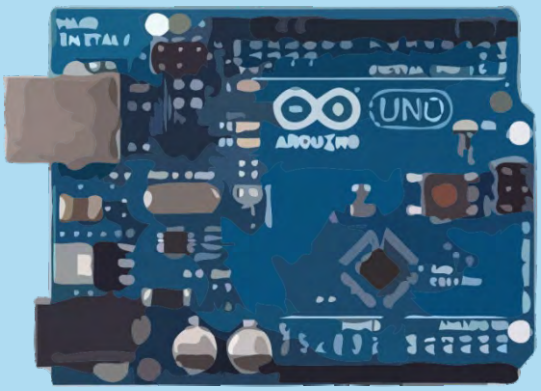
22. Use the switch, to turn on and off your buzzer. If you can hear the BEEP! congratulations! You can control your buzzer from ThingsBoard.

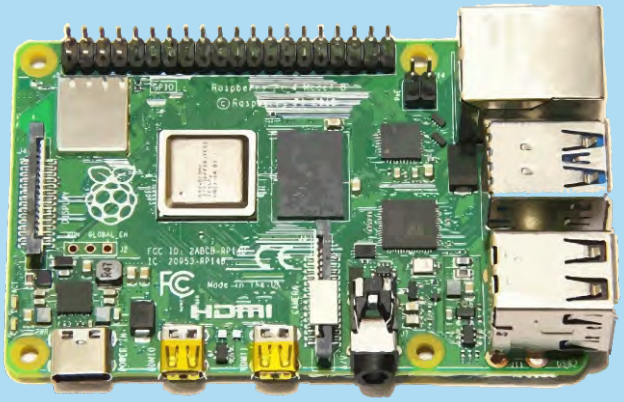
23. **Use Case Scenario(Optional):** Think this as it is your **FINAL TASK**. You are curious about

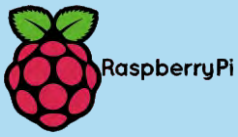
the humidity of your house due to previous mold problems. However you can't check it due to being super busy. So, deploy your application to cloud, print humidity to your LCD, and ask your friend to check the humidity of the house via another PC. Control the backlight of the LCD via switch widget on ThingsBoard.

— Further Reading.

- To learn more about IoT Cloud: `https://thingsboard.io/docs/user-guide/rpc/`.





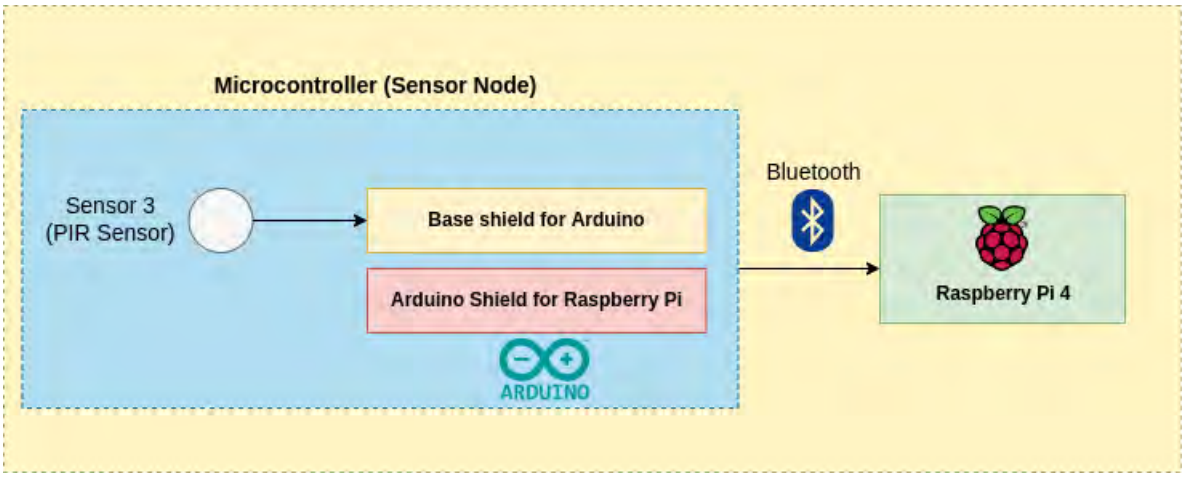


5. Connecting a Sensor Node to IoT Gateway

Objective

- Learn how to connect a sensor node (microcontroller-based) to an edge gateway node (single board computer)
- Learn how to send data from the sensor node to the edge gateway
- Learn how to use Bluetooth for short-range communication

Lab Plan



Required Hardware Components

- SD Card with the Raspberry Pi OS installed
- Microcontroller board (e.g., Arduino)
- Grove HAT (Base Shield for Arduino)
- Display and HDMI cable
- Keyboard and mouse
- Power supply

Additionally we'll use:

- Grove sensor: PIR (Motion) sensor
- Grove - Serial Bluetooth v3.0 or Serial Bluetooth v3.01

Raspberry Pi Setup

1. Raspberry Pi already has a connection module that enables us to make Wi-Fi and Bluetooth connections. However, as the Arduino is quite simple device we're going to use **Grove Bluetooth** module that will enable us to create a Bluetooth connection. In our case, Arduino will act as a **slave** while the Pi is **master**.

2. In this lab. you will need two computers, one is for Raspberry Pi and the other one is to program Arduino.

3. First, connect your Raspberry Pi to your monitor then power up the Pi. Then connect to the **eduroam or CU-PSK** Wi-Fi (in case the Raspberry Pi is NOT already connected to either of the networks).

4. Next, we need to make sure the Node-red is ready to use for serial communication. Type to following commands:

5. Raspberry Pi comes with an already installed Bluetooth package. The Bluetooth should be active by default. Check the status by typing the following in Pi's terminal:

```
systemctl status bluetooth
```

You should see that the service is active just as in Figure 5.1.

```
pi@raspberrypi:~ $ systemctl status bluetooth
● bluetooth.service - Bluetooth service
   Loaded: loaded (/lib/systemd/system/bluetooth.service; enabled; vendor preset: enabled)
   Active: active (running) since Tue 2021-09-07 11:37:32 BST; 1h 38min ago
     Docs: man:bluetoothd(8)
 Main PID: 751 (bluetoothd)
   Status: "Running"
    Tasks: 1 (limit: 4915)
   CGroup: /system.slice/bluetooth.service
           └─751 /usr/lib/bluetooth/bluetoothd
```

Figure 5.1: Bluetooth Status

6. Now we need to program **Arduino** so we can pair it with **Raspberry Pi** and establish a Bluetooth connection.

Arduino Programming

7. Connect your sensors to **Arduino** according to the following:

- **Grove Bluetooth Module v3.0 → D8 port**
- **PIR → D2 port**

8. Now plug your Arduino to PC and open Arduino IDE. Select correct board and port from **Tools** tab. Upload the empty code to verify Arduino is working without any issues.

9. Then get to the required code to program your Arduino from here.

10. Change the **Slave** to something else so you distinguish your Bluetooth id from others:

```
blueToothSerial.print("AT+NAMESlave");
```

11. Finally upload your code to Arduino. Open the **Serial Monitor**. Your Bluetooth module LED should blink slowly which means it is waiting for connection.

12. Now we will pair and connect to our module from Raspberry Pi.

Pairing Raspberry Pi to Arduino

13. Go back to Raspberry Pi and type to following command to find your Bluetooth's 48 bit MAC address as shown in Figure 5.2:

```
hcitool scan
```

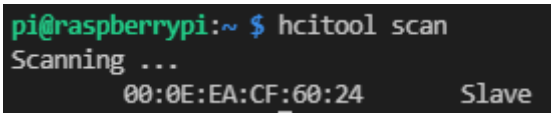


Figure 5.2: Finding the MAC Address of the Bluetooth

Note that address to somewhere. From now on whenever you see **MAC ADDRESS HERE** you will write that address.

14. Now type to following in order. When you type **pair** it may ask for the pin code. It is either **0000** or **1234**:

```
bluetoothctl
agent on
default-agent
scan on     //wait until you see your MAC address
scan off
pair MAC ADDRESS HERE
trust MAC ADDRESS HERE
quit
```

If you have done all the steps successfully you should have a similar output as shown in Figure 5.3.

15. Now that we have paired, we can connect to the Arduino from Pi. Enter the following to your terminal:

```
sudo rfcomm connect hci0 MAC ADDRESS HERE
```

Then, it would be a similar output as in Figure 5.4. Also, your Bluetooth module should give continuous green light.

```
[bluetooth]# pair 00:0E:EA:CF:60:24
Attempting to pair with 00:0E:EA:CF:60:24
[CHG] Device 00:0E:EA:CF:60:24 Connected: yes
Request PIN code
[agent] Enter PIN code: 0000
[CHG] Device 00:0E:EA:CF:60:24 UUIDs: 00001101-0000-1000-8000-00805f9b34fb
[CHG] Device 00:0E:EA:CF:60:24 ServicesResolved: yes
[CHG] Device 00:0E:EA:CF:60:24 Paired: yes
Pairing successful
[CHG] Device 00:0E:EA:CF:60:24 ServicesResolved: no
[CHG] Device 00:0E:EA:CF:60:24 Connected: no
[bluetooth]# trust 00:0E:EA:CF:60:24
[CHG] Device 00:0E:EA:CF:60:24 Trusted: yes
Changing 00:0E:EA:CF:60:24 trust succeeded
```

Figure 5.3: Pairing Bluetooth

```
pi@raspberrypi:~ $ sudo rfcomm connect hci0 00:0E:EA:CF:60:24
Connected /dev/rfcomm0 to 00:0E:EA:CF:60:24 on channel 1
Press CTRL-C for hangup
```

Figure 5.4: Connecting Bluetooth

16. You can also check your connection from the top right Bluetooth icon . Now, we will read the data that Arduino sends via Node-RED. If you have any problems remove your Bluetooth device using bluetoothctl command and redo everything.

Node-RED Programming

17. Open Node-RED.

18. Drag down the **serial in** node to the workspace. Then add **debug** node and connect it to the Serial Node.

— Can not find serial in node?. Could you check in manage palette (you will see this option by clicking on three parallel lines on the top-right corner) if Node-Red has serial-port plug-in?

Download Download repair_nodered.sh bash script from LAB - General Solutions in GitLab to **/home/pi/Downloads/** directory.

Example Open Terminal, cd into Downloads and run command "sh Downloads repair_nodered.sh" in terminal.

```
cd Downloads
sh repair_nodered.sh

```

What will happen? The script you downloaded will run the following commands to remove the old version of node-red and install a new one.

```
    node-red-stop
    wget https://nodejs.org/download/release/latest-v19.x/node
-v19.9.0-linux-armv7l.tar.gz
    tar -xzf node-v19.9.0-linux-armv7l.tar.gz
    sudo cp -R node-v19.9.0-linux-armv7l/* /usr/local/
    sudo ln -s /usr/local/bin/node /usr/bin/node
```

```
        sudo ln -s /usr/local/bin/npm /usr/bin/npm
        sudo npm uninstall -g node-red
        sudo npm install -g --unsafe-perm node-red
        sudo npm install node-red-node-serialport
```

What next? Run node-red again and check if the serial in node is available, if not found then install by accessing "manage palette", then Install tab.

19. Before deploying the project, we must configure the Serial node. Double-click on the node and configure it as in Figure 5.5.

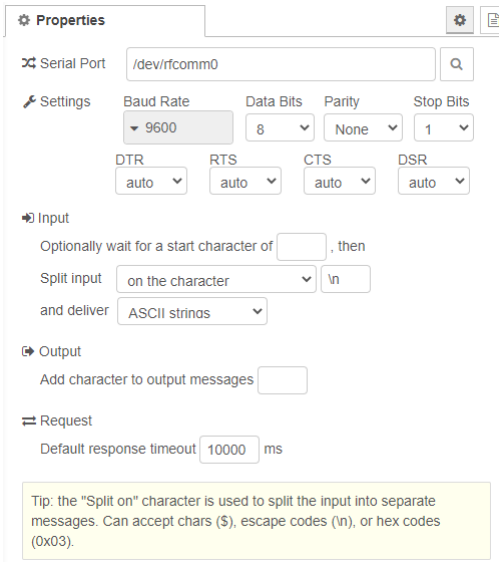


Figure 5.5: Serial Node Settings

Item: Be sure that the **Serial Monitor** of **Arduino** is open. Then, deploy the project in Node-RED. Check your debug messages. You should receive data from the Arduino's PIR(motion sensor). Congratulations!

20. If you have followed all the steps successfully, you should get the final output as in Figure 5.6.

21. **Use Case Scenario (Optional):** Think this as it is your **FINAL TASK**. A thief entered your house at night. You want to detect it via motion and light sensors. Configure your system according to this use case scenario.
Hint: When the light density is below a certain level and there is a movement, you should get an output in Node-RED as "There is a thief!".

— Further Reading.

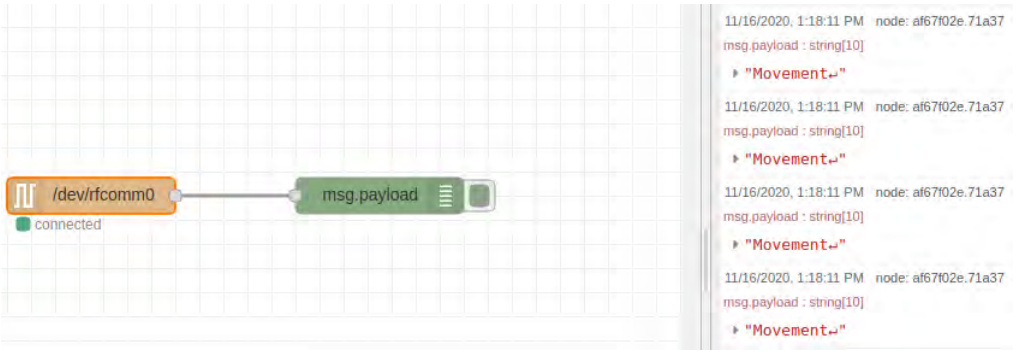
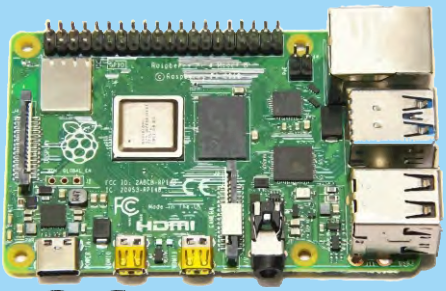
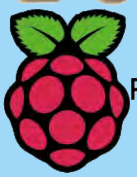


Figure 5.6: Final Output

- To learn more about software serial on Arduino, see: `https://www.arduino.cc/en/Reference/softwareSerial`



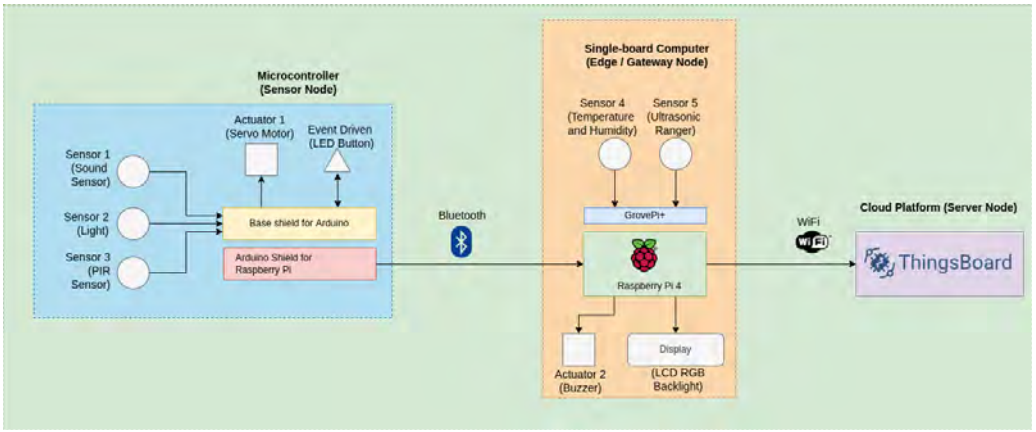
6. End to End Full Stack IoT Development

Objective

- Learn how to connect the sensor node to an edge gateway node
- Learn how to send data from the sensor node to Edge / Gateway
- Learn how to develop an end-to-end IoT stack

Lab Plan

This lab explains how to develop an end-to-end IoT stack that contains edge, fog, and cloud nodes. You will combine what you have learned during the previous labs. The data from the edge-to-fog will be sent over Bluetooth Serial. Then, Raspberry Pi will send the data to Thingsboard cloud where you will observe via dashboard.



Required Hardware Components

- All Hardware components except **Grove-Servo** required in Lab 1, 2, 3, and 4.

End to End Full Stack IoT Development

1. In previous labs. we've learned how to program Arduino and Raspberry Pi, then how to create an Edge Gateway with Raspberry Pi and finally how to communicate Arduino and Pi using Bluetooth. We've learned all of this steps so we can finally create end-to-end IoT stack. In this lab. we will focus on applying knowledge that we've gained previously by creating solution that with small changes could be implemented into the real life.

2. To start you may want to perform previous labs first. However, you do not need to use Grove - Servo motor.

3. First task will be creating temperature monitoring system. Using DHT sensor, Buzzer and LCD, create system that will monitor and display Temperature and Humidity. If the temperature will get too high i.e. 25°C, the system will display new, warning message on the LCD. Additionally the buzzer will work as an alarm after the humidity has reached level of 80 percent. If both the temperature and humidity are above said levels both display and buzzer should react to that. Ensure that the data is sent to the **ThingsBoard** using the required node so later it can be processed.

4. Second task will be creating Home Security System. Using PIR and Sound (loudness) sensors create system that will respond to any suspicious activities i.e. movement or loud noises. Use the LED Button connected to the Arduino to program two modes of the system. First one will be waiting, second - armed just as in regular security systems which we arm after we leave the building. Waiting mode should just display message about it's current state using LCD Back-light display for example: "Waiting". Armed mode will wait for any sensor readings that could mean breaking in. This would activate the Buzzer and would send the message to the Thingsboard. Sensors that could be used in this project are: PIR, Sound Sensor, Ultrasonic Ranger - be creative! The extra feature that you could create would be the uploading Node-RED flow to cloud to disable the system after it's been activated and would stop the buzzer. That way you could remotely check the state of your system and disable it.

> **— Further Reading.**
>
> - `https://www.arm.com/glossary/iot-cloud`

Wireshark

7. Introduction to Wireshark on Raspberry Pi

What is Wireshark?

Wireshark is one of the most common network protocol analyzers. It presents all the captured data that goes through the network in detail in real time. It is an open-source software available on most of the platforms including Linux, Window and IOS etc. It is even used by big companies such as Verizon and Boeing.

Some of the main features that we will be using are mentioned in their website (`https://www.wireshark.org/docs/wsug_html_chunked/ChapterIntroduction.html#ChIntroWhatIs`) summarised below:

- Capture live packet data from a network interface.
- Open files containing packet data captured with tcpdump/WinDump, Wireshark, and many other packet capture programs.
- Import packets from text files containing hex dumps of packet data.
- Display packets with very detailed protocol information.
- Save and export packet data captured.
- Filter and search for the packets depending on your criteria.
- Create various statistics.

In this guide, you will be learning how to use Wireshark on the Raspberry Pi OS which is based on Debian in detail.

Installing Wireshark

There are 2 common ways of installing a package to Raspberry Pi (I will just refer as RPi from now on):

- Installing the package from the source.
- Installing the package from repository.

Installing the package from the repository should be your first option. This prevents lots of possible problems such as version incompatibility or lack of dependencies etc. However sometimes, the the repository package may not be updated. Then you may choose to install newer version from

the source.
To install the package from the repository, first we should check if the package is available there or not. Before doing this, we need check which repository that our RPi is based on. Do the following in order:

- Open your terminal (**ctrl+alt+T**). Type the following:

```
sudo nano /etc/apt/sources.list
```

This will give you the repository name that you are looking for. The text written in orange is your repository name. In this case, it is **buster** as you can see in Fig. 7.1.

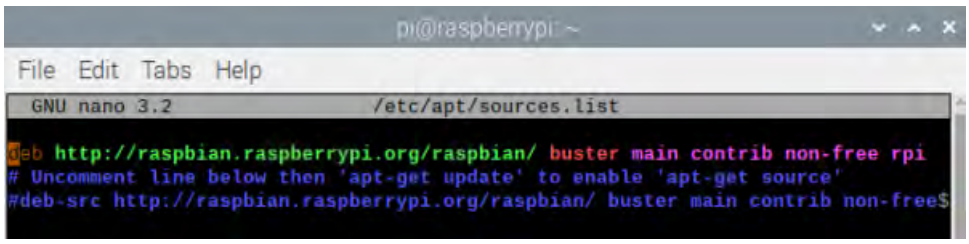


Figure 7.1: Repository Name

- Then, go to the following website `http://archive.raspbian.org/raspbian/dists/buster/main/binary-armhf/` and download the file named **Packages**. Open the file with the any text editor you like such as Geany. Then search for **package: wireshark**. You will see a detailed documentation if the package is available. In this case, package is available so we can find the package details as in Fig. 7.2.

```
1519928  Package: wireshark
1519929  Version: 2.6.8-1.1
1519930  Architecture: armhf
1519931  Maintainer: Balint Reczey <rbalint@ubuntu.com>
1519932  Installed-Size: 63
1519933  Depends: wireshark-qt | wireshark-gtk
1519934  Conflicts: ethereal (<< 1.0.0-3)
1519935  Replaces: ethereal (<< 1.0.0-3)
1519936  Homepage: http://www.wireshark.org/
1519937  Priority: optional
1519938  Section: net
1519939  Filename: pool/main/w/wireshark/wireshark_2.6.8-1.1_armhf.deb
1519940  Size: 50188
1519941  SHA256: e3db3211624aa5cf5c48c9dcc03392885084836b717b1a39220f36fa277d95ae
1519942  SHA1: 97c5fd31918191a962d7d4711ae147f3fcb47ee5
1519943  MD5sum: 5a9123586e1a650cdbbc7ea4c91f06cb
1519944  Description: network traffic analyzer - meta-package
1519945   Wireshark is a network "sniffer" - a tool that captures and analyzes
1519946   packets off the wire. Wireshark can decode too many protocols to list
1519947   here.
1519948   .
1519949   This is a meta-package for Wireshark.
```

Figure 7.2: Wireshark Package

- This means we can install the **wireshark** package via the following line:

```
sudo apt install wireshark
```

It will ask for permission to install dependencies, just enter **y** then continue. It may also ask a permission for non-super users. You may just select **YES** and wait for installation to finish.

- After the installation is finished, you may run the Wireshark from the Internet section as you see in Fig. 7.3.

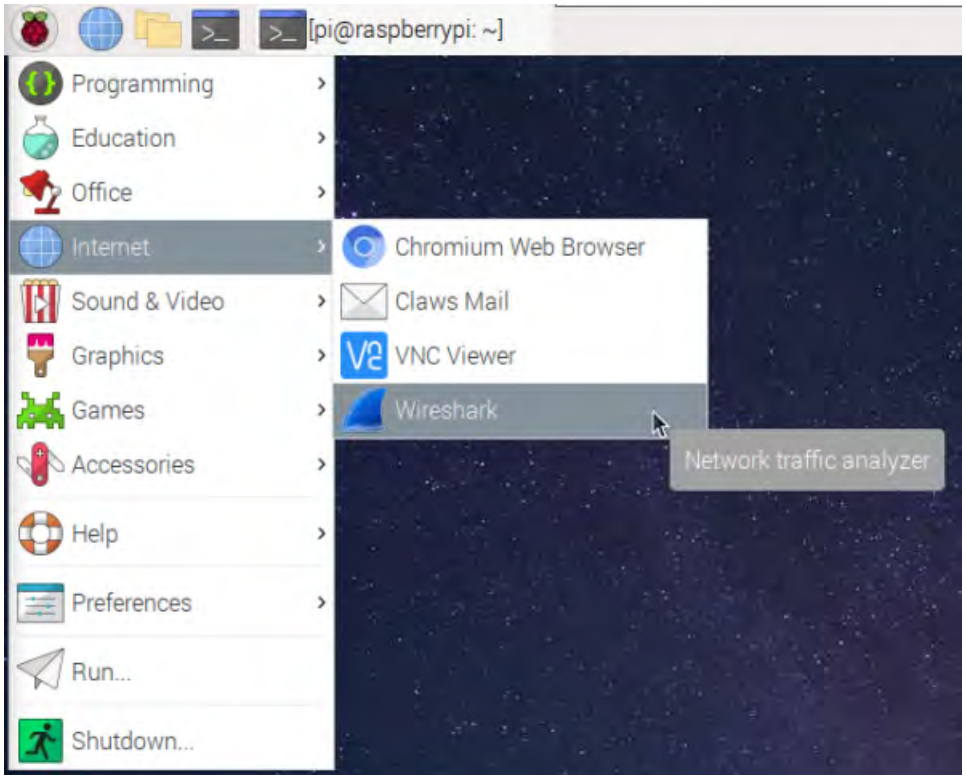


Figure 7.3: Running Wireshark

- Another option is running Wireshark from terminal. Running applications from terminal should be more preferable, because if you get an error when you use the application, you may see the error log on your terminal. Type the following to run the application via terminal:

```
sudo wireshark
```

  Normally you do not need give admin privileges by writing **sudo** to run the applications. However, as we are working in a safe environment, running with **sudo** makes more sense to prevent warnings related to privileges.
  Also if you use terminals you may select different options to run Wireshark. To see all the options type the following to the another terminal:

```
wireshark -h
```

Capturing and Analysing Packets

We have successfully installed Wireshark. Now we will run it. When you run the Wireshark, you will see the main window as you can see in Fig. 7.4.

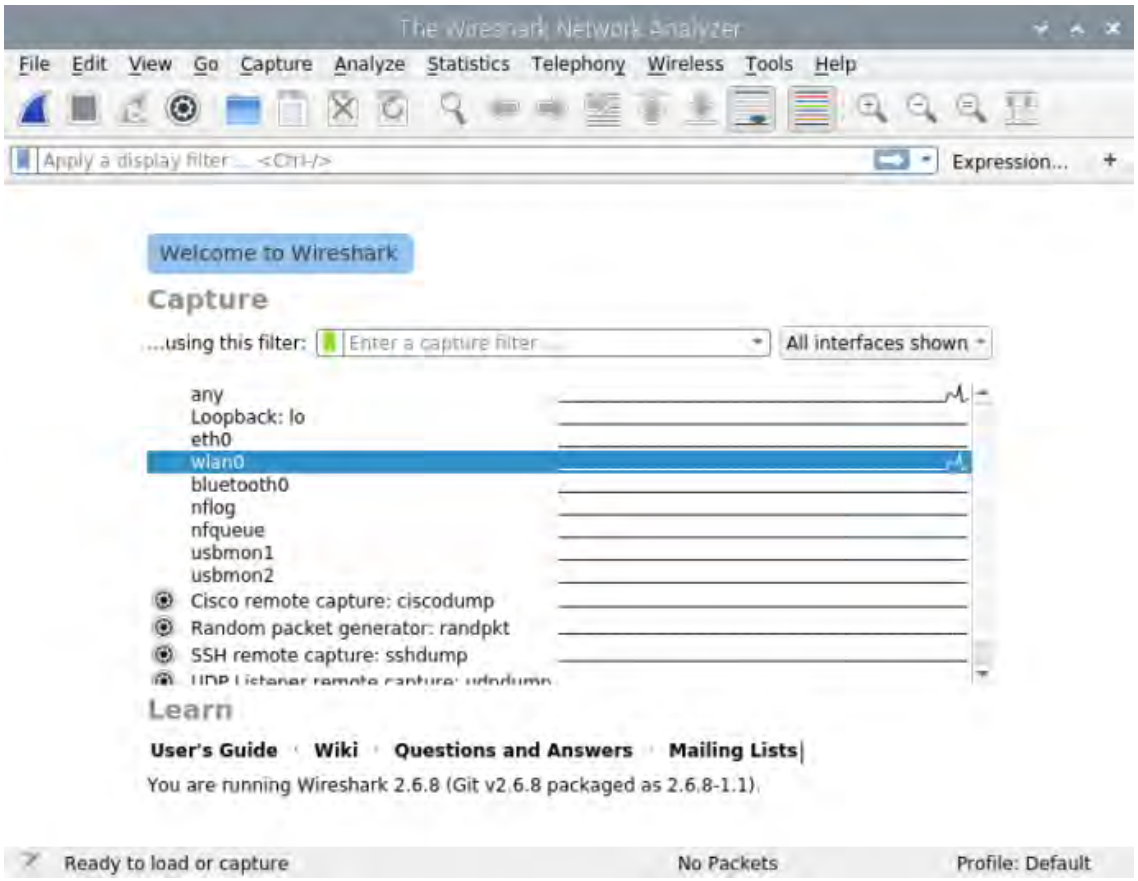


Figure 7.4: Wireshark Main Window

You can see there are different interfaces such as **eth0** and **wlan0**. You also may realize from the graphic there is data going through the **wlan0**.

Now, to capture the network traffic in real time, click on the blue shark fin icon as you see in Fig. 7.5.

Figure 7.5: Capture Icon

Now you will see a live feed of captured packets belonging **wlan0** interface as in Figure 7.6. The GUI of the wireshark interface is summarized below:

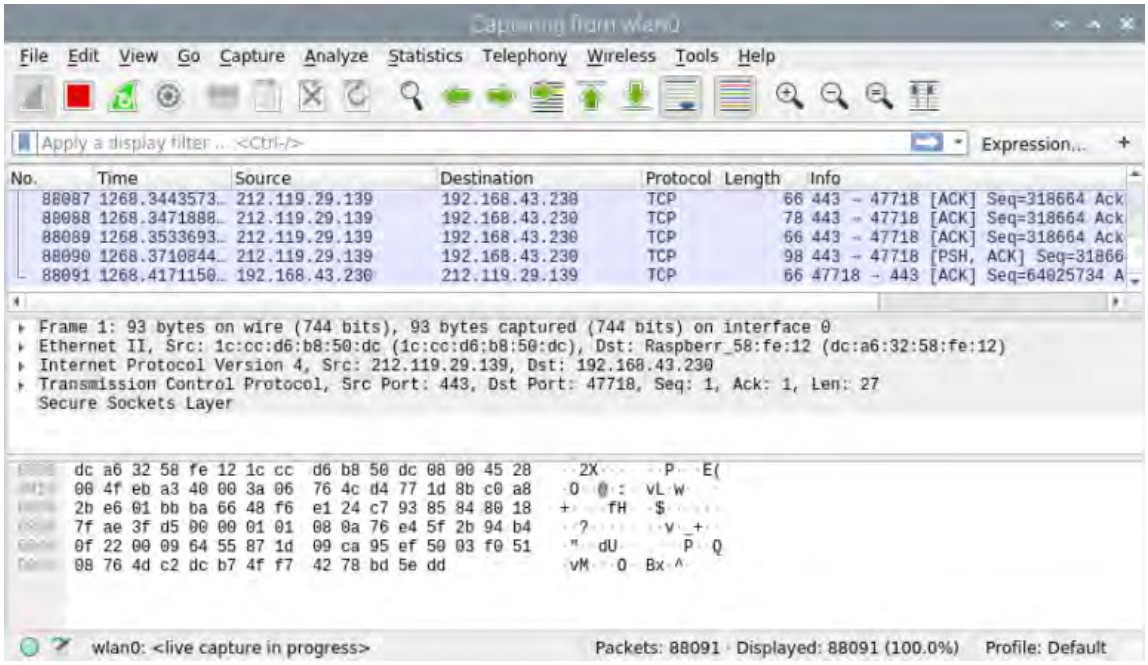


Figure 7.6: The Wireshark GUI

- 1st bar is the window name that indicates which network is captured. In this case it is **wlan0**.
- 2nd bar is the **menu** that is used to start actions.
- 3th bar is **main toolbar** provides quick access to some commonly used functions in the menu.
- 4th bar is the **filter toolbar** that lets you filter the displayed packets.
- 5th part is the **packet list pane** that shows the summary of the captured packet.
- 6th part is the **packet bytes pane** shows the data of the selected packet in a hexdump style.
- 7th bar is the **status bar**.

Now click on the red stop icon next the blue shark fin to stop live capture so we can examine it in details.

In the **packet list panel** each line belongs to a one packet. The items in this pane are summarised below:

- **No :** Packets are numbered when the live capture is started. You can see the order of the packets.
- **Time :** The timestamp of the packet.
- **Source :** The address where the packet is coming from.
- **Destination :** The address where the packet is going to.
- **Protocol :** The network protocol name.
- **Length :** The length of the packet in terms of **bytes**.
- **Info :** Detailed information about the packet.

When you select an packet you will see an symbol in the **No** section. In Fig. 7.7, you can see there is a **tick on packet 3**. Which means that, the 4th packet acknowledges the 3th packet. To see all the symbols and their meanings, please visit `https://www.wireshark.org/docs/wsug_html_chunked/ChUsePacketListPaneSection.html`.

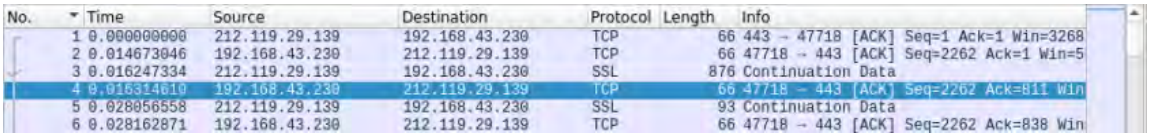


Figure 7.7: Related Packet Symbol

You can save your captured network by clicking **File** then **Save As...** in many formats including **pcap** as you see in Fig. 7.8. This helps you to analyse your network as you want. You can simulate your network traffic by replaying those files. In case of unusual traffic such as cyberattacks, these pcap files are examined in detail to understand the behavior of the attack.

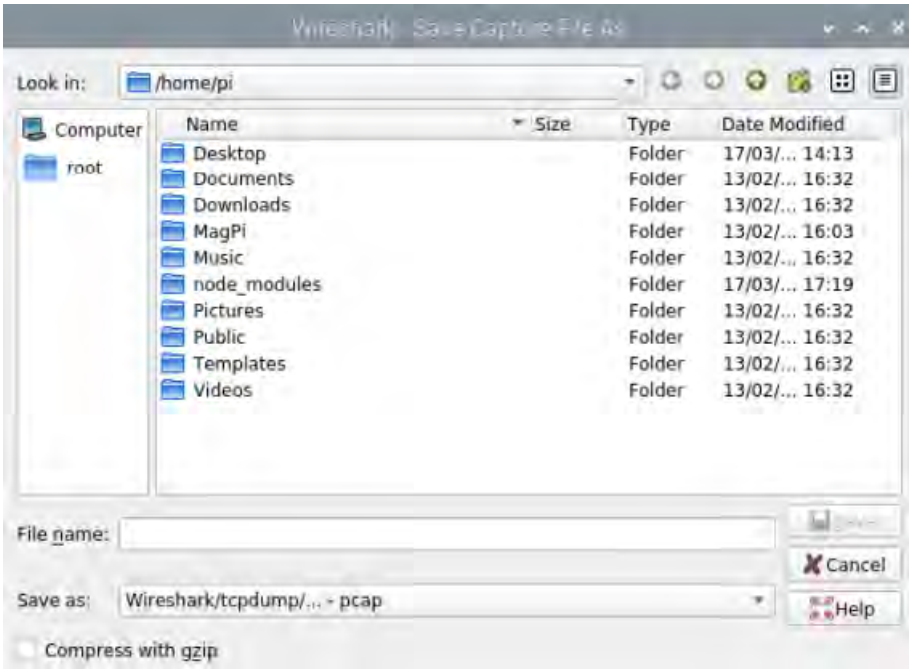


Figure 7.8: Saving Pcap File

You can also open and analyse any pcap files that you have from the same section.

You can filter the captured traffic via filter bar. Let's say you want to see DNS traffic. You can type **dns** to the filter and see all the packets belongs to that traffic. The filter bar has its own syntax. If you type something wrong, the bar will turn to a red. It should be green as in Fig. 7.9 to apply a filter. You can apply the filter by the arrow at the end of the filter bar.

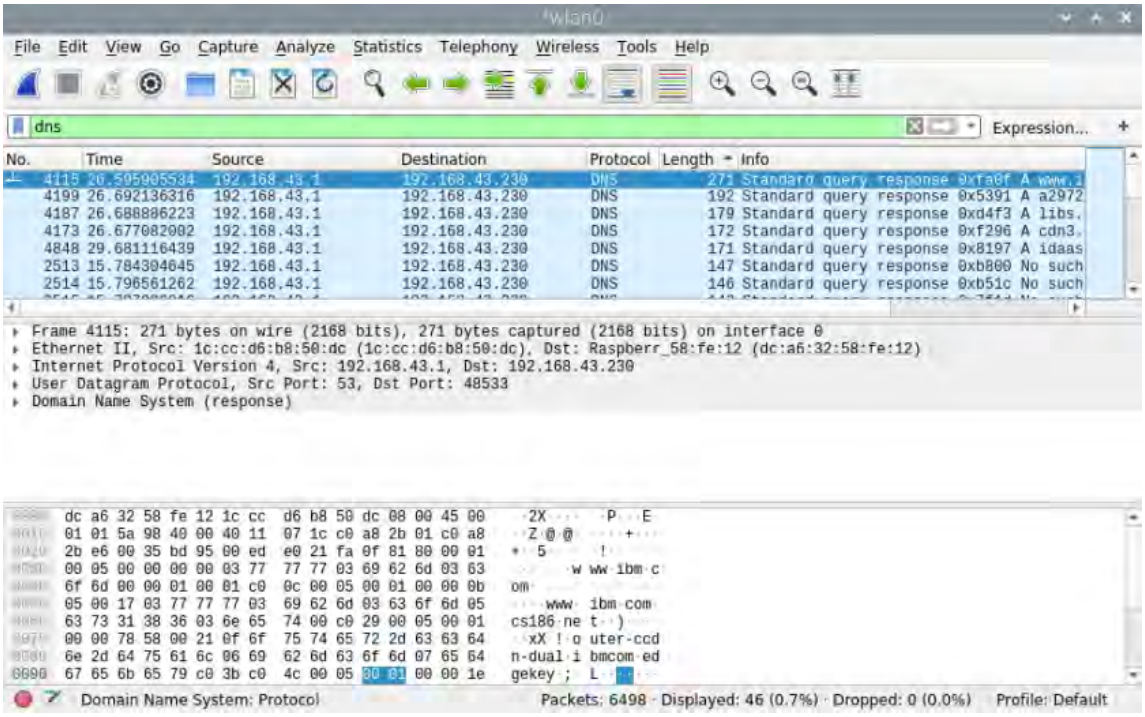


Figure 7.9: Applying Filter

As you see in above figure, only DNS packets are shown which means the filter is applied successfully. Let's say you want to observe the traffic that includes specific IP address. In this case you need to type the following to the filter:

```
ip.addr == 194.13.12.12
```

In this case Wireshark will display all the packets that includes **194.13.12.12** as an source or destination address. You can reach the whole syntax list from the following address: `https://www.wireshark.org/docs/dfref/`.

The timestamp shows how many seconds are passed since the capture is began. However, this information is moslty useless. You can set the timestamp from **View** then **Time Display Format** as you want.

Wireshark has **packet colorization**. This is a very useful mechanism when observing a live traffic as in Fig. 7.10. If there are some unexpected colored packets you can immediately react to an event.

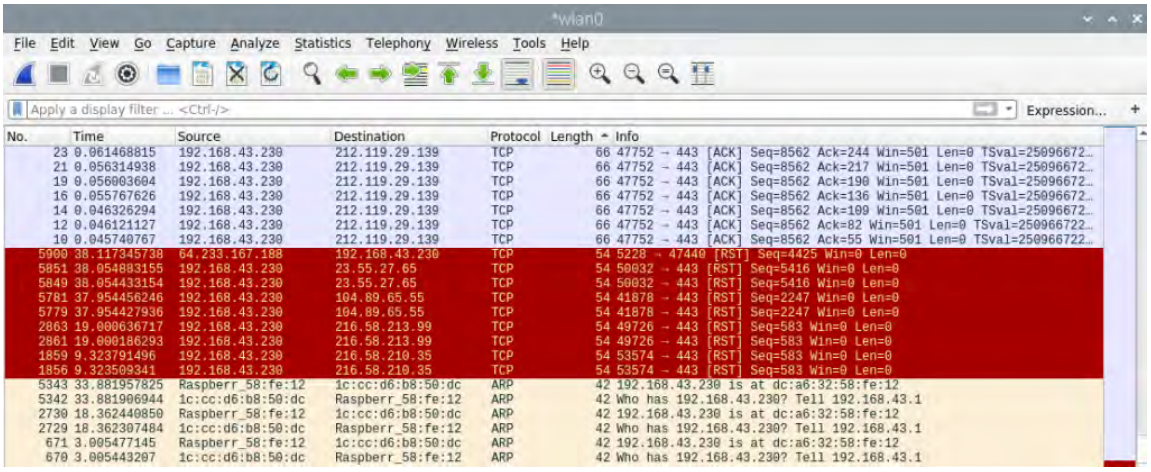


Figure 7.10: Packet Colorization

Coloring rules are also customisable. Click on **View** and then **Coloring Rules**. You will see a similar window as in Fig. 7.11. You can add a new color rule by clicking the **+** icon on bottom left.

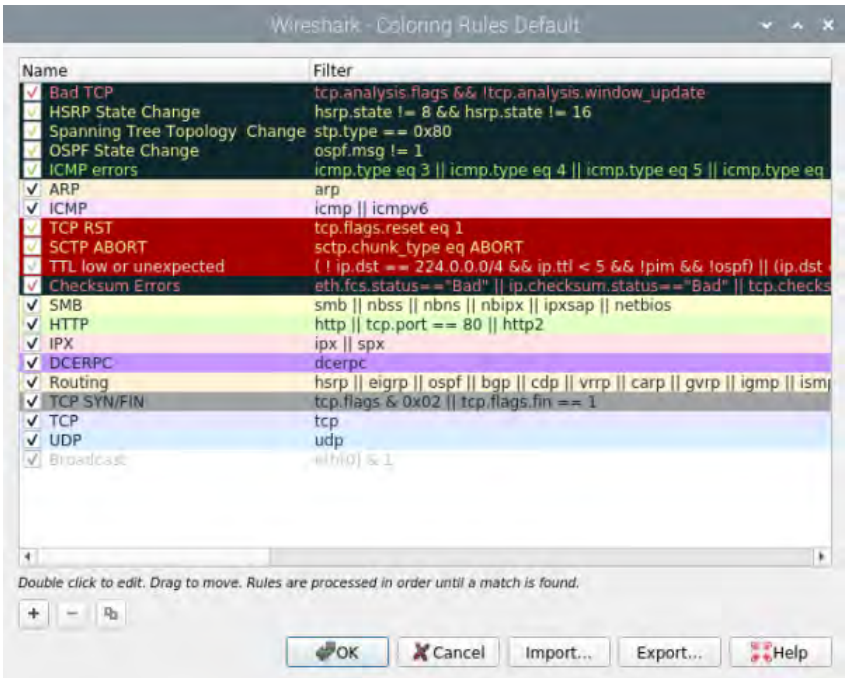


Figure 7.11: Color Rules

Another useful feature of Wireshark is **Statistics** tool in the menu. Here you can generate graphs and get many detailed information about your captured file. You can select the first option to see the main statistics of the pcap file as shown in Fig. 7.12.

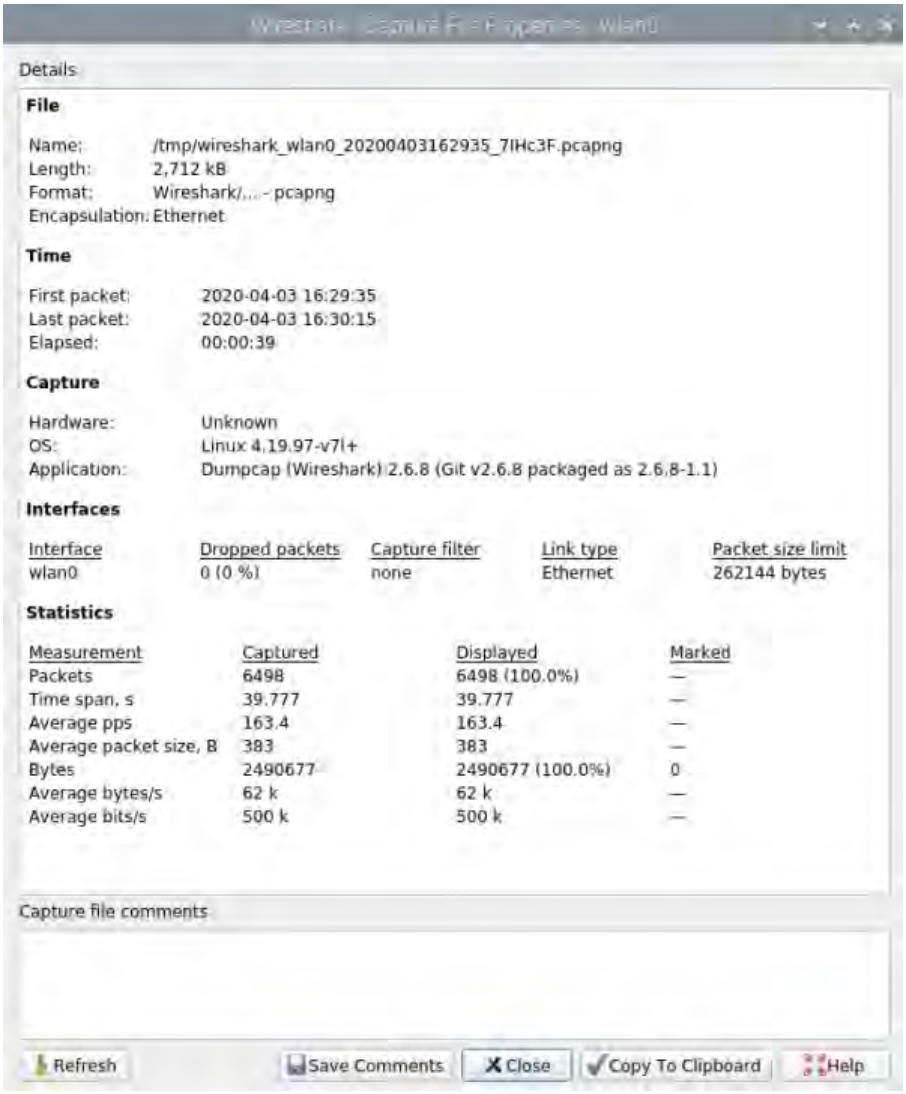
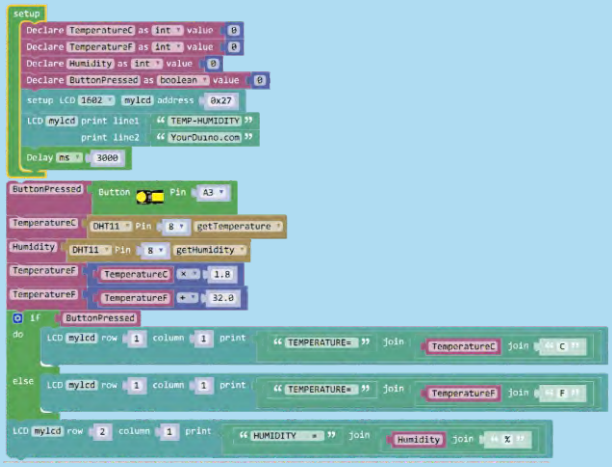


Figure 7.12: Statistics

— Further Reading.

- To learn more about Wireshark: `https://www.wireshark.org/docs/`



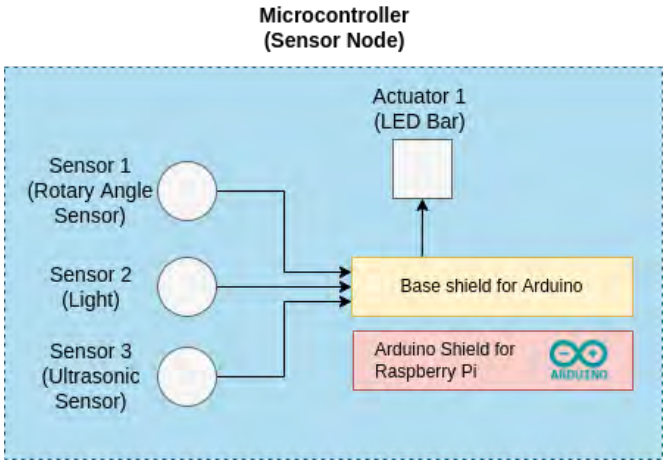
8. Programming Arduino with Blockly

Objective

- Learn how to program an Arduino via Blockly
- Learn how to read data from various sensors
- Learn how to use a LED Bar

Lab Plan

This lab explains how to program Arduino via Blockly. We will control LED bar via rotary angle sensor while using drag and drop nodes which are built-in in Blockly.



Required Hardware Components

- Grove - Rotary Angle Sensor v1.2
- Grove - LED Bar v2.1

- Grove - Light Sensor v1.2
- Grove - Ultrasonic Ranger v2.0
- Arduino Expansion Shield for Raspberry Pi B+ (V2.0)
- Grove - Base Shield

Setting Up Codecraft

1. Navigate to the `www.tinkergen.com`.
2. Then from **Codecraft** select **Programming Online** from the header bar.
3. You will see lots of different hardware options.
4. Select **Arduino(Uno/Mega/BeginnerKit)**.
5. Now, you should see a workspace as shown in Figure 8.1.

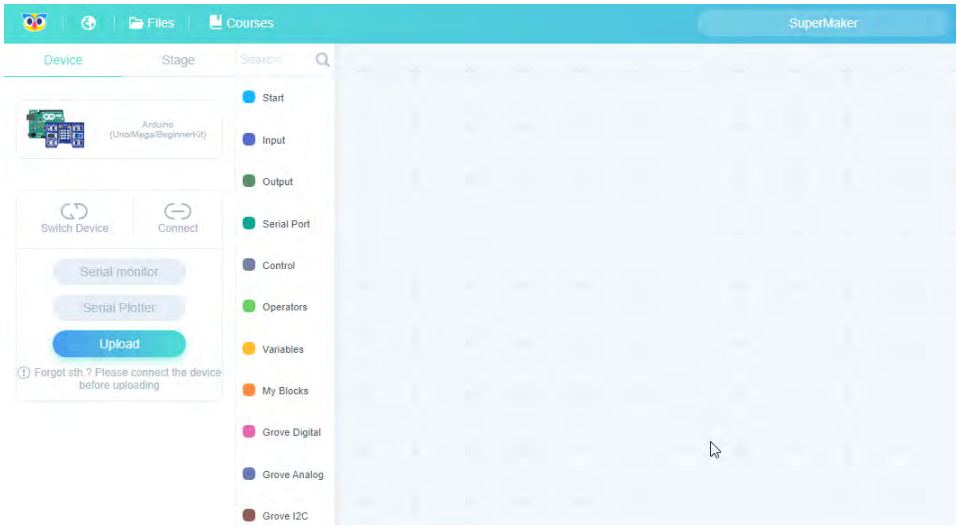


Figure 8.1: Codecraft Workspace

6. Now, we will setup our Arduino Leonardo.

Setting Up Arduino

7. Connect your Grove Base Shield on top of Arduino.
8. Then connect the sensors accordingly:
   - **LED Bar → D4 port**
   - **Rotary Angle Sensor → A3 port**
   - **Ultrasonic Ranger → D3 port**
   - **Light Sensor → A0 port**
9. On your base shield, there is a voltage switch. There are 2 options: **3V3**, and **5V**. Make sure **5V** is set.
10. Now, connect your Arduino to PC.

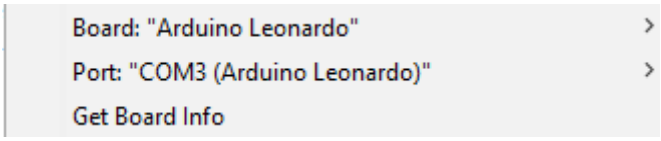


Figure 8.2: Arduino Settings

11. Run Arduino IDE.
12. From, **File** select **New**, then save the file.
13. From **Tools**, set your **Board** to *Arduino Leonardo* and select the correct port.
14. Your **Tools** settings should be similar as shown in Figure 8.2.
15. First we will run ultrasonic ranger.
16. Download the required library from our GitLab. The library file is named as:
    Seeed_Arduino_UltrasonicRanger-master.zip.
17. From **Sketch**, select **Include Library**, then select **Add .ZIP Library...**, and upload the library then you have downloaded.
18. Now we will generate our code in Codecraft.

Programming Ultrasonic Ranger via Codecraft

19. Set up your workspace as it is shown in Figure 8.3. You can easily find the correct nodes via looking at their colors.

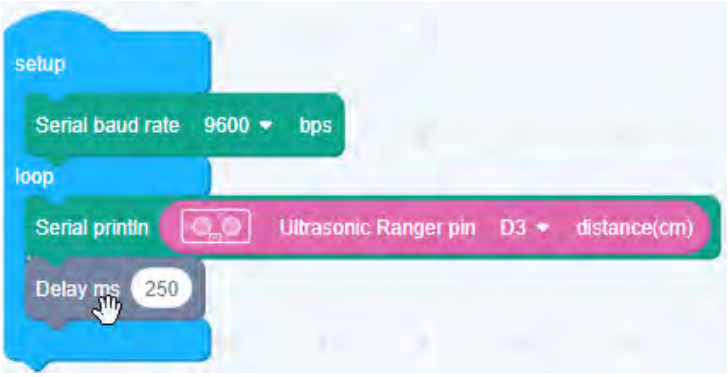


Figure 8.3: Codecraft Ultrasonic Ranger Setup

20. Then from top right, click on the workspace switch icon that looks as in Figure 8.4.

Figure 8.4: Workspace Switch

21. Now, copy the code from the workspace to Arduino IDE. Upload the code, and open the serial

monitor. You should be seeing the distance generated by ultrasonic ranger in centimeters.

22. Now, we will get light data via light sensor.

Programming Light Sensor via Codecraft

23. Set up the Codecraft workspace as shown in Figure 8.5.

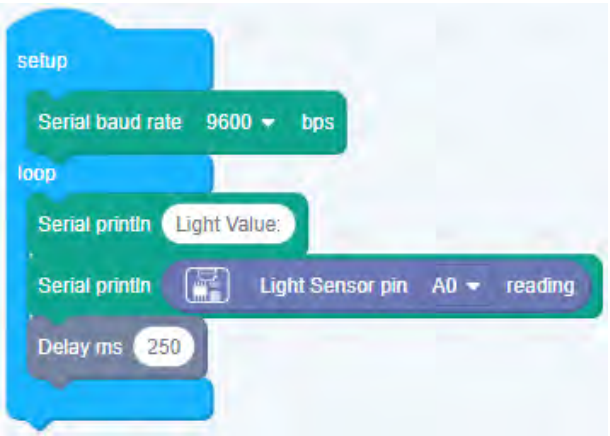


Figure 8.5: Codecraft Light Sensor Setup

24. Then again switch the workspace, copy the code to your Arduino IDE, and upload.

25. Open the serial monitor to observe the light data. If you hold the sensor in your hand, you will see that, the light density will decrease.

26. Now, we will program the rotary angle sensor.

Programming Rotary Angle Sensor via Codecraft

27. Set up the Codecraft workspace as shown in Figure 8.6.

28. Then again, as you have done before, copy the code to your Arduino IDE. However, now, we need to do a slight change in the code to make it properly working. Remove the outer parenthesises when defining the voltage as shown below:

```
voltage = analogRead(A3) * 5 / 1023;
```

29. Now, we can upload the code to our Arduino.

30. Open serial monitor and rotate the actuator on the rotary angle sensor to observe the change in the angle value.

31. Now, we will learn how to program the LED Bar.

Programming Grove LED Bar via Codecraft

32. We already have learnt how to use Codecraft to generate Arduino code. Now we will do the reverse.

Figure 8.6: Codecraft Rotary Angle Sensor Setup

33. First download and install the library named "Grove_LED_Bar-master.zip" from our GitLab.

34. Then again copy the code from **groveLEDBar.ino** to your own Arduino script and upload the code. That file is also located in our GitLab. Then observe the behavior of the LED Bar.

35. Now, you will develop a similar code via using Codecraft.

36. You can find the LED bar node in **Grove Digital** section. The required node is shown in Figure 8.7.

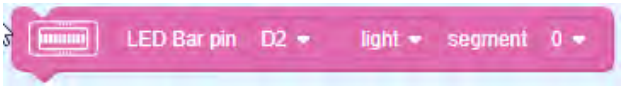


Figure 8.7: Codecraft LED Bar Node

37. Now build your own setup in Codecraft that provides the similar behavior of the previously deployed Arduino code.

38. **Use Case Scenario (Optional)**: Think this as it is your **FINAL TASK**. Imagine that you would like to build a smart window. The LED Bar indicates the level of the window. When the amount of light is above certain level, you will close the window via using rotary angle sensor. You will do the opposite as well.

> **— Further Reading.** .
>
> - To learn more about Codecraft see official documentation `https://www.yuque.com/tinkergen-help-en/codecraft?language=en-us`
> - To learn more about Arduino visit `https://www.arduino.cc/`

RaspberryPi

python™

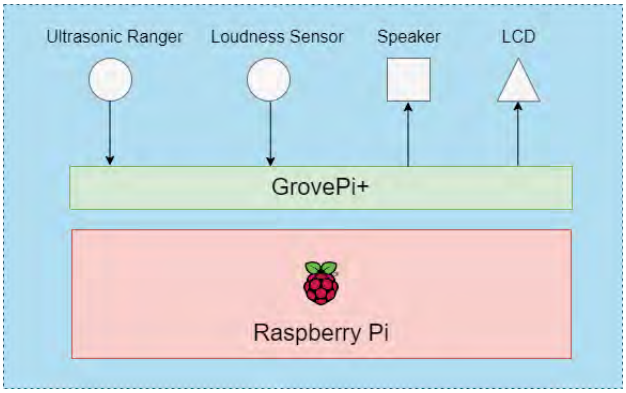
9. Programming Raspberry Pi with Python

Objective

- Learn how to execute Python scripts on Raspberry Pi
- Learn how to read data via Python scripts
- Learn how to utilize a speaker and LCD display together

Lab Plan

This lab explains how use grove sensors via Python scripts. Python comes pre-installed with Raspberry Pi OS. We will connect Grove sensors to Raspberry Pi via GrovePi+. Then we will run Python scripts and program several sensors including ultrasonic ranger.



Required Hardware Components

- Raspberry Pi 4
- SD Card with OS installed on it (we'll work on the Raspberry Pi OS)
- Display and HDMI cable

- Keyboard and mouse
- Power supply

Additionally we'll use:

- GrovePi+
- Grove Loudness Sensor v0.9b
- Grove - LED Bar v2.1
- Grove Ultrasonic Ranger V2.0
- Grove - Buzzer v1.2

Setting Up Pi

1. Connect your Raspberry Pi to GrovePi+.
2. Then, connect the sensors/modules accordingly to your GrovePi+.
   - **Buzzer → D8 port**
   - **LED Bar → D5 port**
   - **Loudness Sensor → A2 port**
   - **Ultrasonic Ranger → D4 port**
3. Now power up your Raspberry Pi.
4. Then open your terminal and enter the following:

```
sudo i2cdetect -y 1
```

5. If you can see **04** in your output, it means that Raspberry Pi is able to detect GrovePi+.

Programming Buzzer

6. Buzzer is an audio signaling device that beeps. Hence, we can utilize it as an indicator or alarm. Now, we will run our Buzzer via Python script.
7. First create **buzzer.py** in your home directory from your terminal (CTRL + ALT + T):

```
cd ~
touch buzzer.py
```

8. Then from our GitLab, copy the content of **buzzer.py** to your own script. To do this, you can double-click to file and paste it. Then save the file and close it.
9. Now we will run the file via:

```
python3 buzzer.py
```

10. After hearing them most beautiful beep in the world, you can terminate the script via **CTRL + C**.

Programming LED Bar

11. Now we will program our LED bar in Python.
12. This time we will use built-in Python IDE named Thonny.
13. Open Thonny Python IDE from the main Rasspery Pi panel as shown in Figure 9.1.

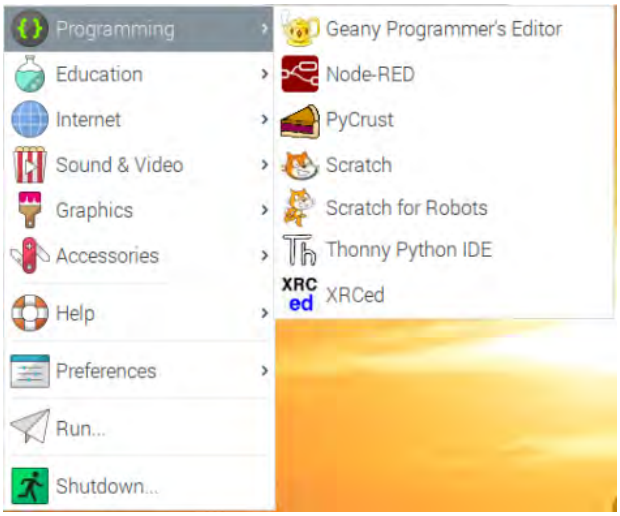


Figure 9.1: Opening Thonny Python IDE

14. Then, create a new script by clicking on the + icon as shown in Figure 9.2.

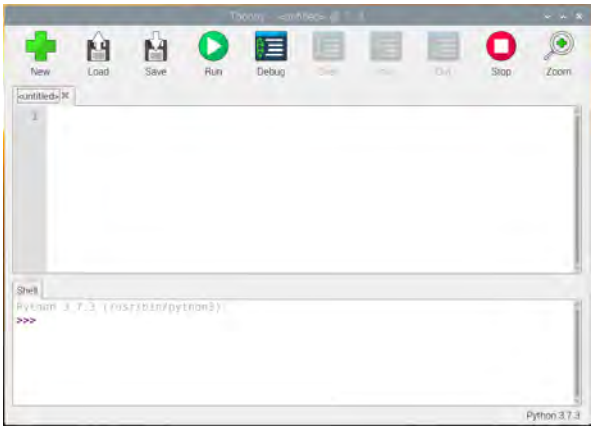


Figure 9.2: Creating Thonny Python IDE Script

15. Copy the content of an **ledBar.py** from our GitLab and paste it to your own script.
16. Then, save the script, name as **ledBar.py**.
17. Now, run the script by clicking on a **Run** button.
18. You will observe 18 different behaviours on the LED bar, each behaviour will be printed on shell as well as shown in Figure 9.3.

Figure 9.3: Thonny Python IDE Shell Output

19. Then click on the stop icon, to terminate the script.

20. Now you know two different ways to create and run a Python script on Raspberry Pi. For the upcoming two examples, you can pick the way you like.

Programming Loudness Sensor

21. Loudness shows you the magnitude of sound.

22. You can find the required code to read data via loudness sensor in our GitLab.

23. Analog sensors tend to be sensitive, so feel free to try, other analog ports if you are not able to read any data. Don't forget to edit your code accordingly.

24. Clap your hands, or make noise in some other way to observe the change in the

Programming Ultrasonic Ranger

25. Ultrasonic rangers show the distance.

26. You can find the required code to read data from ultrasonic ranger in our GitLab.

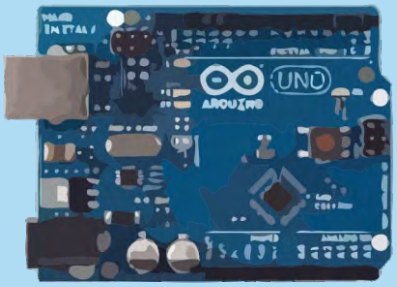
27. Move your hand closer to the sensor and observe the data generated by the sensor.

28. **Use Case Scenario (Optional)**: Think this as it is your **FINAL TASK**. Imagine that you would like a house intrusion detection system. The LED Bar indicates the level of sound. Ultrasonic ranger is a detector that observes the house door. Buzzer will beep if there is a sudden change in the sound. Based on your imagination, write your own script and show it to your friends.

> **— Further Reading.** .
>
> - You can find more examples in the following GitHub link. `https://github.com/DexterInd/GrovePi/tree/master/Software/Python`

Wireless devices streaming rich content like data, video, and audio
(device pairing required)

Sensor devices sending small bits of data, using very little energy
(device pairing not required)



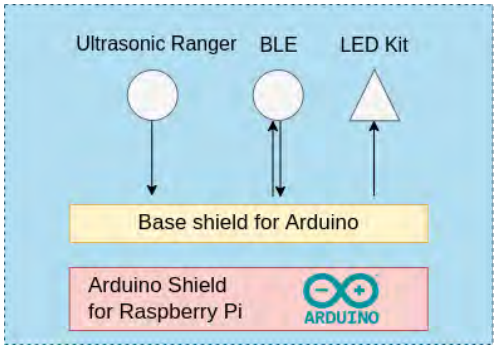
10. Bluetooth Low Energy (BLE) Based Systems

Objective

- Learn how to configure a BLE module.
- Learn how to use BLE application
- Learn how to transmit data over BLE
- Learn how to display received data

Lab Plan

This lab explains how to establish a Bluetooth Low Energy (BLE) connection between the edge development board and mobile phone via application. We will utilize AT commands to communicate.



Required Hardware Components

- Arduino Shield for Raspberry Pi
- Grove Base Shield
- Grove BLE v1.0
- Grove Ultrasonic Ranger V2.0
- Grove LED Socket Kit v1.5
- Loudness Sensor

Setting Up Arduino Shield for Raspberry Pi

1. Arduino Shield for Raspberry Pi has an Arduino Leonardo chip, hence we can treat it as an Arduino. I will mention as **Arduino** from now on.
2. First, connect your Grove Base Shield to Arduino.
3. Then, connect your LED to Grove LED Socket Kit.
4. Finally, connect the sensors below accordingly to your shield.
   - **BLE → UART**
   - **LED Kit → D5 port**
   - **Loudness Sensor → A2 port**
   - **Ultrasonic Ranger → D4 port**
5. Now, connect your Arduino to PC.
6. Check if the port and board **(Leonardo)** is correct.
7. Upload an empty sketch to make sure everything is fine.
8. If you do not get any errors, you may proceed to next part.

Understanding the BLE Code

9. Arduino Leonardo has two serial ports. Hence, we can utilize the Universal asynchronous receiver-transmitter (UART) port of base shield. You can find more detailed information in the following link: `https://www.arduino.cc/reference/en/language/functions/communication/serial/`.
10. Instead of master and slave devices, we have central and peripheral in BLE connections. While peripheral device acts like a server, central acts like a client.
11. In this case, the Arduino is a peripheral device while the mobile phone is a central device.
12. Copy the code from our GitLab and paste it to your sketch.
13. Upload the code.
14. In this code, we have the following section:

```
while(!Serial);
```

    Thanks to this code, our sketch will hang the device until we open the serial monitor. This is useful during development. However, if you use Arduino Leonardo for production, you would like to delete this part.

15. Now, open the serial monitor. As soon as you open, you should see the following.

```
OKOK+Set:0OK+Set:Ratata
```

16. This is the reply coming from the BLE module. At the end of the code, we have the following section:

```
  Serial1.print("AT");
  delay(400);
  Serial1.print("AT+ROLE0"); // set the role as peripheral.
```

```
4   delay(400);
5   Serial1.print("AT+NAMERatata") // set the name as Ratata.
```

17. Whatever we print to **Serial1** are sent to BLE module. These are called as **Hayes command set.** It is the most common way to talk to these modules.

18. Set your **communication format** to **No line ending** as shown in Figure 10.1.

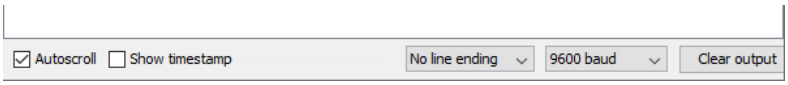


Figure 10.1: Communication Format

19. Now type **AT** and send. You should get **OK** in return from the module which means that your communication with the module is healthy.

20. Each **AT command set** per module is different. You should check the source document to find out what are the commands that you can use. The source document of our module is available in the following link: `https://files.seeedstudio.com/wiki/Grove-BLE_v1/res/Bluetooth4_en.pdf`.

21. Now we will communicate with the module via mobile phone.

Connecting to BLE Module via Mobile Phone

22. Download the Serial Bluetooth Terminal application from Google Play Store.

23. Install the app and provide the required permissions.

24. Open the app and click on the adapter icon on the top right to **turn on** your Bluetooth if closed.

25. Then from the hamburger menu (an icon with 3 straight lines) select **Devices**.

26. You will see two options: **Bluetooth Classic** and **Bluetooth LE**.

27. As we are using a BLE module you need to select **Bluetooth LE**.

28. Now before scanning for BLE devices I would recommend changing the name of your module to something unique. Edit to following line in the code to change the name into something else:

```
1   Serial1.print("AT+NAMERatata") // set the name as Ratata.
```

29. Now, click on the **SCAN** button. You will see lots of other devices as shown in Figure 10.2.

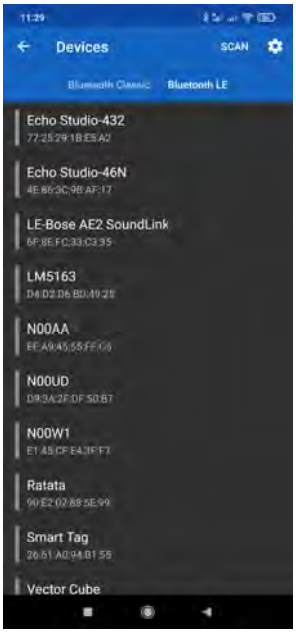


Figure 10.2: Scanned Device List

30. Find your own device in that list and click on it to connect. After you connect, the green LED on BLE module will stop blinking and be on all the time which means it is connected.

31. Now, you should be able to send data via your phone. Type "Hello World!" and click on the **arrow icon** to send your data. You should see "Hello World!" on your serial monitor.

32. Now, lets utilize the sensors.

Sending Sensor Data Over BLE

33. Now we will simulate an alarm system that warn us when there is a storm.

34. The LED will be **on** if the storm is very close, will **blink** if the storm is coming, will be off if there are no signs regarding the storm.

35. Create a new Arduino sketch. Copy the code from our GitLab and paste it to your sketch.

36. Then, upload the code and open serial monitor. You should see the following output as shown in Figure 10.3:

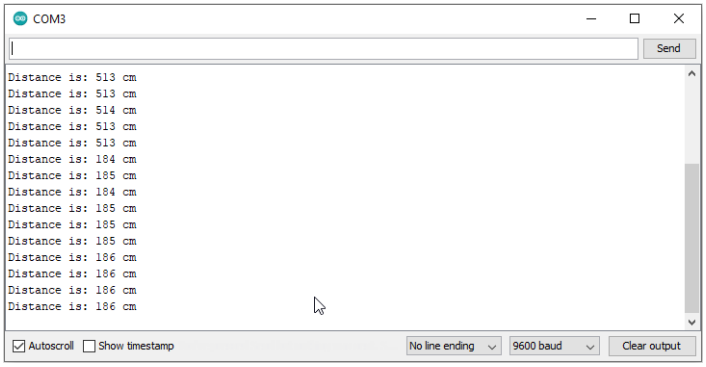


Figure 10.3: Distance on Serial Monitor

37. Now connect to your BLE via mobile phone again.

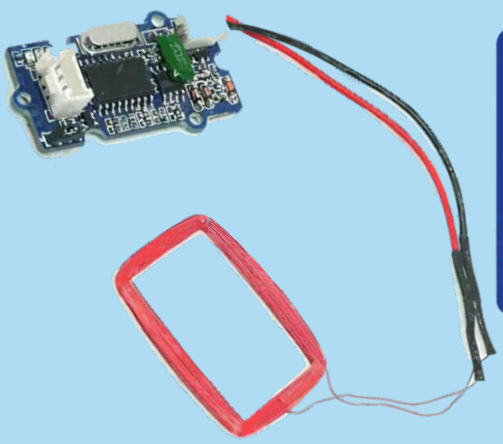
38. Move your hand accordingly to change the distance read by the ultrasonic ranger. Check what kind of messages your are getting in your app. These messages are sent over BLE by BLE module. You should see similar readings as shown in Figure 10.4.

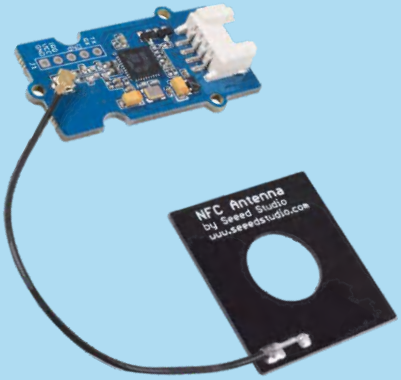
Figure 10.4: Reading Data via App

39. Congratulations, now you have learnt how to send data over BLE.

40. **Use Case Scenario (Optional)**: Think this as it is your **FINAL TASK**. Imagine that we need a more detailed representation of the storm warning system. The blinking rate of the LED will indicate how close the storm is. You will observe the distance via your app. You will print 5 different warning messages on serial monitor based on the location of storm.

> **— Further Reading.** .
>
> - You can learn more about BLE on the following link: `https://www.oreilly.com/library/view/getting-started-with/9781491900550/ch01.html`
> - More learn about the Hayes command set: `https://en.wikipedia.org/wiki/Hayes_command_set`





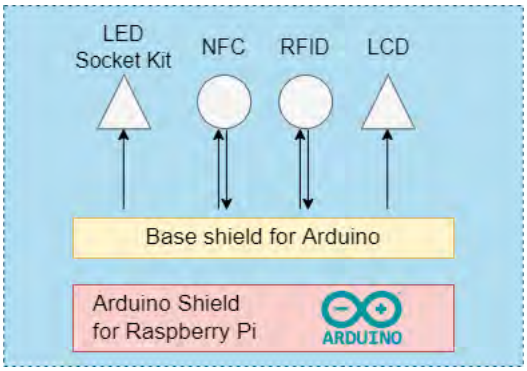
11. RFID and NFC Based Tracking

Objective

- Learn how to establish NFC communication
- Learn how to use NFC tags
- Learn how to use RFID reader

Lab Plan

This lab explains how to utilize NFC and RFID sensors. We will learn how to connect both RFID and NFC modules via UART port. We will turn on the LED in case of there is a successfull communication.



Required Hardware Components

- Arduino Shield for Raspberry Pi
- Grove Base Shield
- Grove-LCD RGB Backlight V4.0
- Grove LED Socket Kit v1.5

- LED
- Grove - NFC Tag v1.1
- Grove - 125 KHz RFID Reader
- NFC Tags

Setting Up Connections and Testing LED Kit

1. Arduino Shield for Raspberry Pi has an Arduino Leonardo chip, hence we can treat it as an Arduino. I will mention as **Arduino** from now on.

2. First, connect your Grove Base Shield to Arduino. Make sure that the power switch on base shield is set to 5V.

3. The longer leg is the positive side of LED, hence connect your LED to your LED kit in a way that the same sides match.

4. Finally, connect the sensors below accordingly to your shield.
   - **LCD → I2C Port**
   - **LED Kit → D4 port**
   - **NFC → UART Port**
   - **RFID → UART Port** - This will be done, after completing NFC part.

5. Now you can connect your Arduino to your PC.

6. Let's first test the **LED kit** to confirm it works fine.

7. Run **Arduino IDE**.

8. Open a new sketch and copy paste the following code:

```
#define LED 5 //we connect LED kit to D5 of Grove Shield
void setup() {
    // initialize the digital pin2 as an output.
    pinMode(LED, OUTPUT);
}
void loop() {
    digitalWrite(LED, HIGH);
    delay(500);
    digitalWrite(LED, LOW);
    delay(500);
}
```

9. You can also download the sketch from our GitLab as well.

10. The LED should blink in certain intervals. If it does not blink, make sure that you connected the LED tightly. Also check the polarity.

11. Now, let's confirm if the LCD works fine.

Setting Up LCD

12. Now we will check if our LCD works fine.

13. First download the LCD library from our GitLab and install it.

14. Download the *labTest.ino* Arduino script from our GitLab and upload it to Arduino.

15. You should see that the screen color of LCD will change between **green, red, and blue** while acting as a timer.

16. Now we will learn how to

Setting Up Grove NFC Module

17. First, download the "Seeed_Arduino_NFC-master.zip" from our GitLab.GitLab.

18. Run Arduino IDE.

19. Install the library via **Sketch -> Include Library -> Add .ZIP Library**.

20. Then either download the **nfc.ino** or copy the contents of the file to your own Arduino script. The code is available in our GitLab repository.

21. Before running the code let's learn what it does. First we include the required libraries via the following:

```
#include <NfcAdapter.h>
#include <PN532/PN532_HSU/PN532_HSU.h>
#include <rgb_lcd.h>
```

22. Then we define our initial parameters:

```
PN532_HSU pn532hsu(Serial1);
NfcAdapter nfc(pn532hsu);
rgb_lcd lcd;
// connect led kit to D4
#define LED  4
```

Here we can use **Serial1** because our device is Arduino Leonardo which has an additional serial port. Check the following link for further information: `https://www.arduino.cc/reference/en/language/functions/communication/serial/`.

23. Then, in setup we have the following line:

```
while(!Serial);
```

This line hangs the code until the serial monitor is open. This is very useful for developing while should be avoided for production.

24. Then, in the loop we have the following condition:

```
if (nfc.tagPresent()) {
    Serial.print("NFC TAG FOUND");
    digitalWrite(LED, HIGH);
```

This means that if NFC tag is detected by the Grove - NFC Tag module, the LED on the LED kit will turn on.

25. Then we write a message ("Hello World") to our NFC tag and verify that the process is completed successfully:

```
// write a message to tag
NdefMessage message = NdefMessage();
message.addUriRecord("Hello World");
```

```
bool success = nfc.write(message);
if (success) {
    Serial.println("Success.");
} else {
    SERIAL.println("Write failed.");
}
```

26. Finally, we read the message and other parameters and print them on serial monitor, while printing the UID to LCD screen:

```
// read the message from the tag
NfcTag tag = nfc.read();
// print UID to screen
String UID = tag.getUidString();
String firstHalf = UID.substring(0, UID.length()/2);
String secondHalf = UID.substring(UID.length()/2);
lcd.clear();
lcd.setCursor(0,0);
lcd.print("UID:  ");
lcd.print(firstHalf);
lcd.setCursor(0,1);
lcd.print(secondHalf);
// show tag parameters
tag.print();
```

27. Now, run the code and put the NFC tag on the reader of the NFC module as shown in Figure 11.2.

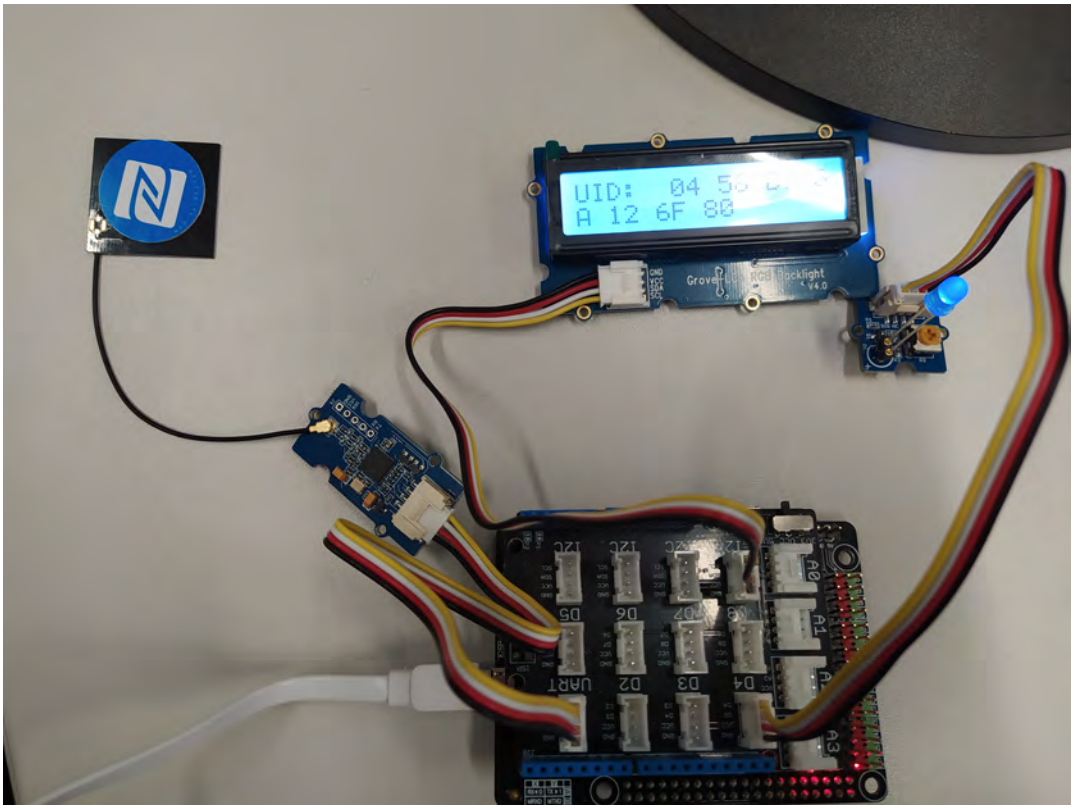


Figure 11.1: The NFC Setup

28. Then open your serial monitor. You should have a similar output as shown in Figure **??**.

```
11:22:38.127 -> NFC TAG FOUNDSuccess.
11:22:38.305 -> NFC Tag - NFC Forum Type 2
11:22:38.305 -> UID 04 58 B7 6A 12 6F 80
11:22:38.305 ->
11:22:38.305 -> NDEF Message 1 record, 16 bytes
11:22:38.305 ->   NDEF Record
11:22:38.305 ->     TNF 0x1 Well Known
11:22:38.305 ->     Type Length 0x1 1
11:22:38.305 ->     Payload Length 0xC 12
11:22:38.305 ->     Type 55  U
11:22:38.305 ->     Payload 00 48 65 6C 6C 6F 20 57 6F 72 6C 64  .Hello World
11:22:38.305 ->     Record is 16 bytes
```

Figure 11.2: The NFC Setup

You can see our message on the **Payload** line.

Setting Up RFID Reader

29. Now remove the NFC module from UART port, and plug Grove - RFID reader.
30. As this is a reader module, we will only read data from the card.
31. RFID cards are heavily used in control access as each card has an unique tag.
32. Now copy the code from our GitLab repository, and upload to Arduino.
33. Open the serial monitor.
34. Now bring your RFID card to closer (around 2.5cm) to your RFID reader.
35. The tag of the card will be printed to serial monitor as shown in Figure 11.3.

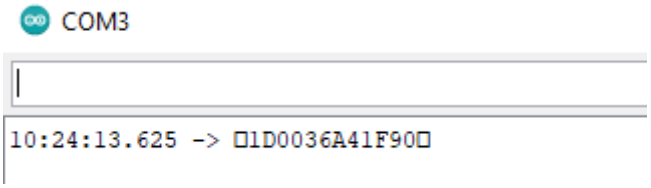


Figure 11.3: The RFID Tag

36. Swap cards with other groups to see if the cards have an unique identifier.

> **— Further Reading.** .
>
> - To learn more about NFC `https://nfc-forum.org/`
> - To learn more about RFID `https://en.wikipedia.org/wiki/Radio-frequency_identification`

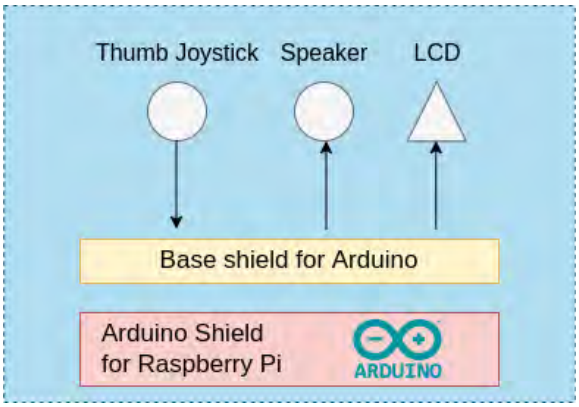
12. Multimedia Communication

Objective

- Learn how to display data on LCD
- Learn how to program and control joystick
- Learn how to generate sound on speaker
- Learn how to display received data

Lab Plan

This lab explains how to utilize thumb joystick and speaker. We will see how thumb joystick generates coordinate as we see in game controllers. We will control the colours of LCD via joystick.



Required Hardware Components

- Arduino Shield for Raspberry Pi
- Grove Base Shield
- Grove - LCD RGB Backlight

- Grove - Thumb Joystick v1.1
- Grove - Speaker v1.1

Setting Up Arduino

1. First, connect your base shield to Arduino.
2. Then, connect your modules accordingly:
   - **LCD → I2C port**
   - **Thumb Joystick → A0 port**
   - **Speaker → D4 port**
3. Now we can connect our Arduino to PC.
4. Run Arduino IDE.
5. Install the following library if not installed.
   - Grove LCD RGB library.
6. Ensure the voltage switch on the base shield is set to 5V.
7. Select the correct **port** and **board** from **Tools** section.
8. Upload an empty sketch. If everything works well, move to the next section.

Programming the Thumb Joystick

9. Copy the code from our GitLab and paste it to your sketch.
10. Upload the code.
11. Now open the serial monitor and play with the joystick. You should see **X** and **Y** values change as you play with the joystick as shown in Figure 12.1.

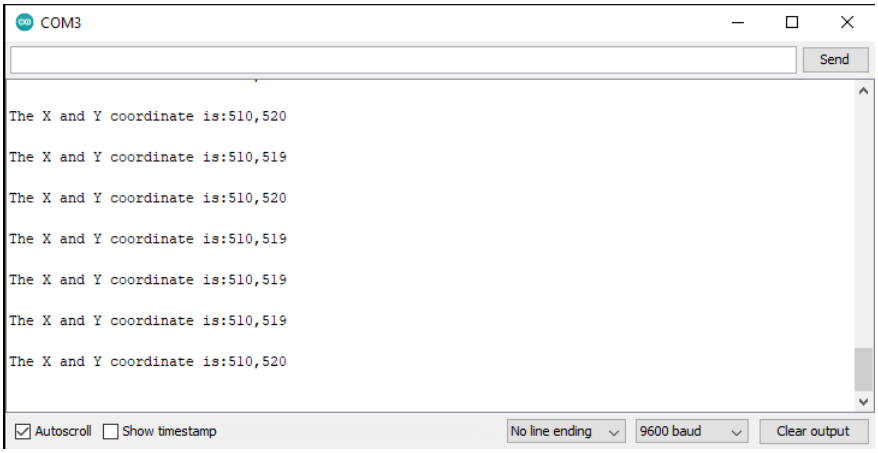


Figure 12.1: Arduino Settings

Programming the Speaker

12. The speaker is an actuators that generates a sound.
13. Create a sketch and copy the code from our GitLab and paste it to your sketch.
14. Upload the sketch and observe the behaviour of the speaker.
15. Change the variables of following section to generate different sound.

```
int BassTab[]={1911,1702,1516,1431,1275,1136,1012};//bass 1~7
```

Combining Modules

16. In this section we will combine these three modules and make a mini application.
17. Create a sketch and copy the code from our GitLab and paste it to your sketch.
18. Do not upload the code as it will make a noise. Let's first breakdown the code a little bit.
19. These are the **initial** values that determine the bass level and the color of the backlight. Feel free to change and observe their behaviours.

```
int colorR = 123;
int colorG = 123;
int colorB = 123;
int bass = 1000;
```

20. Joystick generates data in **IDLE** position as well. In our case default X is 509, and default Y is 510. We print the default values via the following:

```
int sensorValue1 = analogRead(A0);
int sensorValue2 = analogRead(A1);
Serial.print("The X and Y coordinate is:");
Serial.print(sensorValue1, DEC);
Serial.print(",");
Serial.println(sensorValue2, DEC);
Serial.println(" ");
```

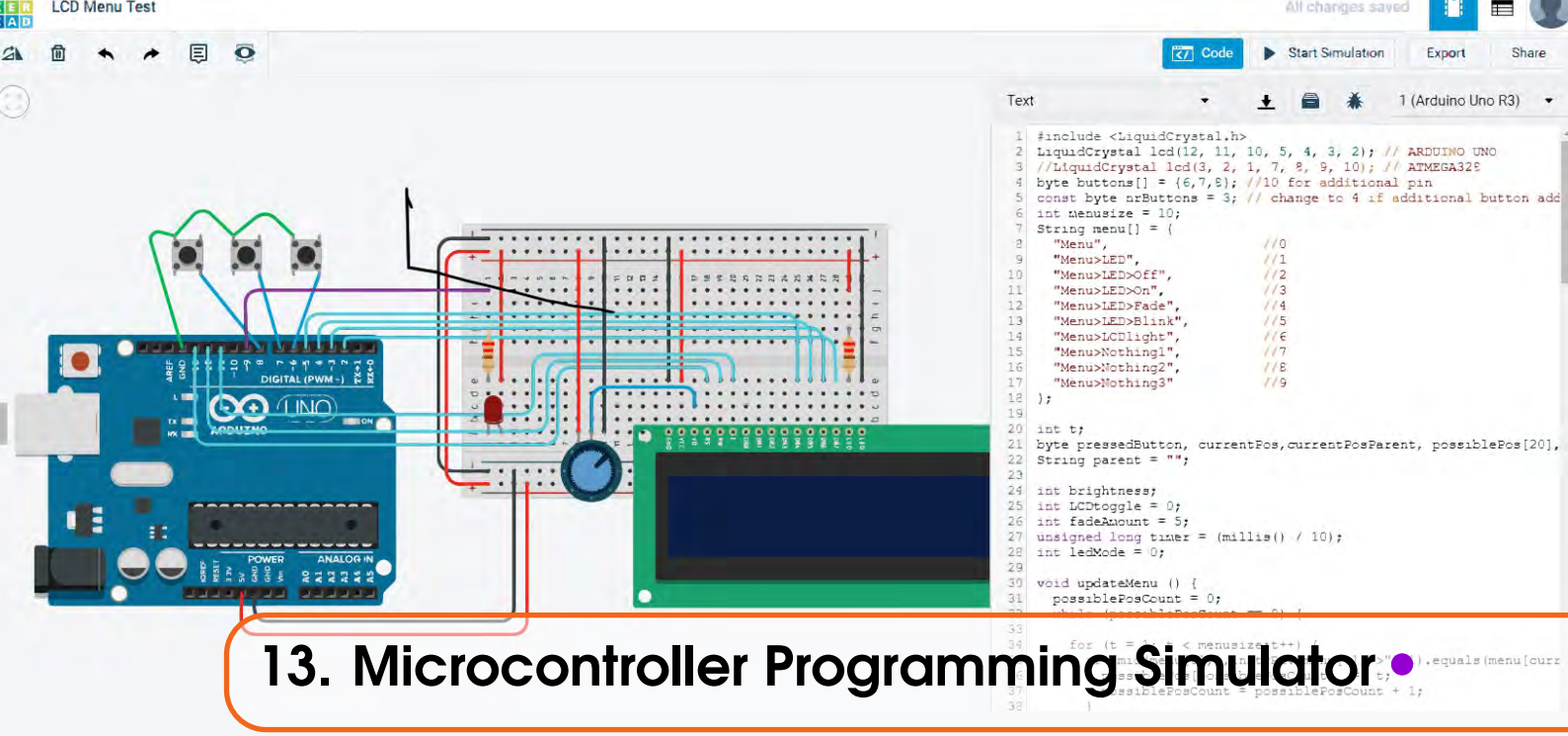
21. Hence, we define the conditions based on these default values. We add an additional condition to prevent any unwanted changes from happening while the joystick is **IDLE**.

```
  if(sensorValue1 < joystickXDef && sensorValue1 != 510){ //module not
    calibrated, hits 510 randomly.
    //bass down
    bass = bass - 200;
  } else if (sensorValue1 > joystickXDef && sensorValue1 != 510){
    //bass up
    bass = bass + 200;
  }
  if (sensorValue2 > joystickYDef && sensorValue2 != 520) { //module not
    calibrated, hits 520 randomly
    //color up
    colorR = 255;
    colorG = 0;
    colorB = 0;
  } else if (sensorValue2 < joystickYDef && sensorValue2 != 520) {
    colorR = 0;
```

```
15      colorG = 255;
16      colorB = 0;
17    }
```

22. What is happening under which condition is pretty clear.

23. **Use Case Scenario (Optional):** Think this as it is your **FINAL TASK**. We would like to match color level with the bass level. As the level of bass increases, the color gets darker. Use joystick to increase the bass. Also the display should only be on when there is a sound. Configure your script accordingly.



13. Microcontroller Programming Simulator

Objective

- Learn how to use a simulator.
- Learn how to design a circuit.
- Get familiar with Tinkercad circuits interface.

Required Hardware Components

- PC with internet connection

Setting up an Initial Circuit with Tinkercad

1. Go to `www.tinkercad.com`.
2. Click on **Start Tinkering** button.
3. Login to your Tinkercad account. Create one if you do not have any.
4. After that select **Circuits** from left and click on **Create new Circuit** as shown in Figure 13.1.
5. Then from right side change component selection to all as shown in Figure 13.2.
6. Now add three components which are Arduino Uno R3, Breadboard Small, and LED via dragging.
7. Connect them as shown in Figure 13.3 via left clicking with your mouse. You can set the cable color from top left. We use **red** for positive and **black** for ground. Then from top right click on the **Start Simulation**.
8. Hover your mouse over LED and examine the warning.

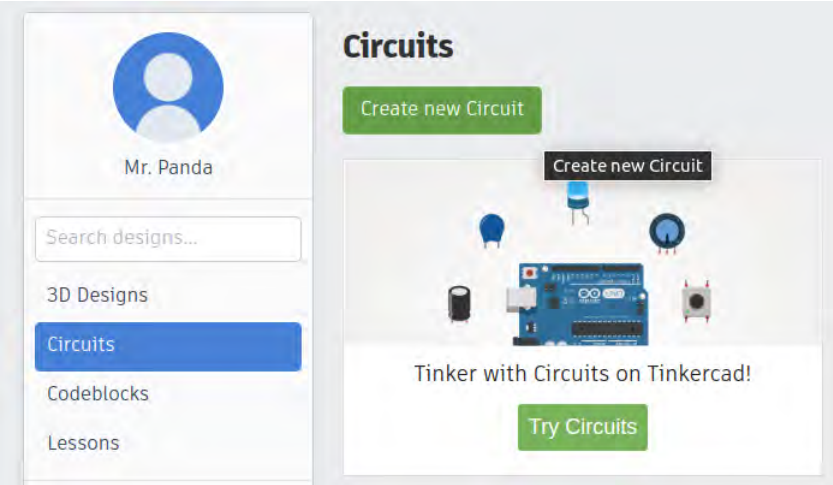


Figure 13.1: Create New Circuit

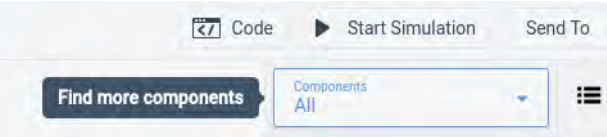


Figure 13.2: Component Selection

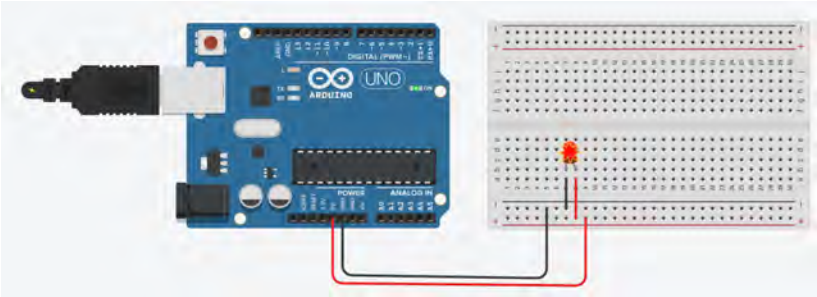


Figure 13.3: Initial Setup

9. Basically, LED is drawing more current than it should do. In real world, we would burn the LED if we had the exact same setup.

10. Very simple solution to that is adding a **resistor** to system. By this way, we allow some of the potential difference to be used on resistor.

11. Add a resistor of 1kΩ (which is the default). Connect them as shown in Figure 13.4. Then, start simulation. Now you should have a healthy working LED.

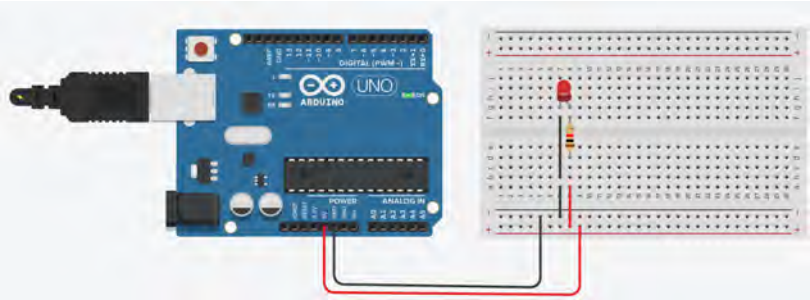


Figure 13.4: Adding a Resistor

Using the Code Editor

12. When you run the previous setup, you should see that the built-in LED of Arduino is **blinking**.

13. Click on the **Code** editor which is next to the **Start Simulation**. You should see a similar setup as shown in Figure 13.5.

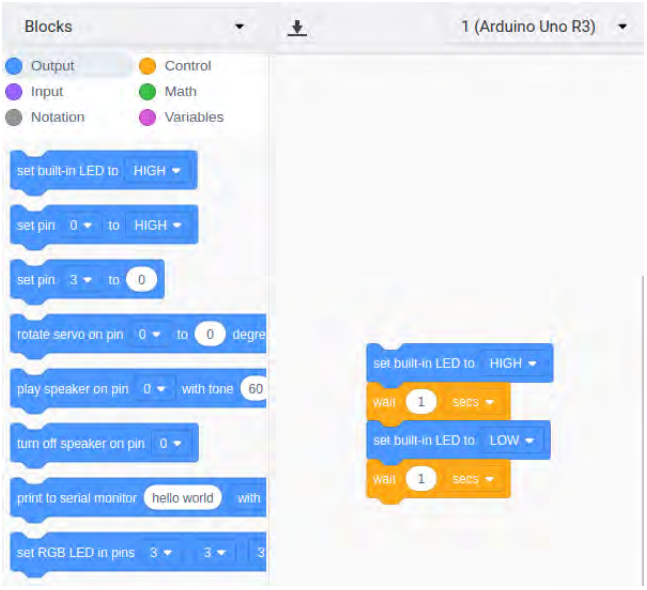


Figure 13.5: Code Editor

14. On the left, we see the blocks that we can use. On the right we see the workspace.

15. Change the delay and run the simulation. Then observe the behaviour of the built-in LED.

Printing Temperature to the LCD

16. Now we will do a little bit more complex setup. This time we will print the temperature value that we get from analog temperature sensor (TMP36).

17. Set your workspace as you shown in Figure 13.6. The resistor value is 220Ω. Be careful about the connection locations.

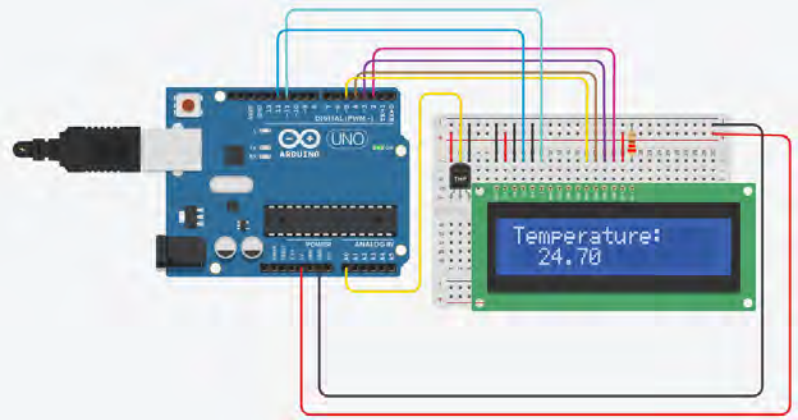


Figure 13.6: Printing Temperature to LCD

18. Now we can either use blocks or we can write our code from the scratch. This time we will write from scratch. Open the **code editor** and select **text** from drop-down menu.

19. Copy paste the following code to your sketch.

```
#include<LiquidCrystal.h>
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
void setup()
{
 lcd.begin(16, 2);
}
void loop()
{
  int reading = analogRead(0);
  float voltage = reading * 5.0;
  voltage /= 1024.0;
  float temperatureC = (voltage - 0.5) * 100 ;
  lcd.setCursor(0,0);
  lcd.print("Temperature: ");
  lcd.setCursor(2,1);
  lcd.print(String(temperatureC));
  delay(1000);
}
```

20. Now run the simulation. If you have done everything correct, you should see the temperature value on LCD.

> **— Further Reading.** .
>
> - You can learn more about how we are calculating the temperature value from the following link: `https://learn.adafruit.com/tmp36-temperature-sensor/using-a-temp-sensor`

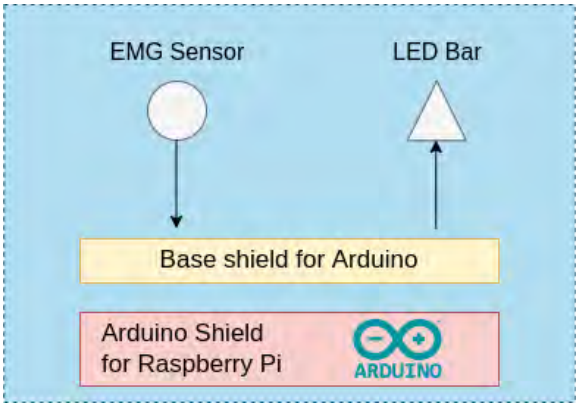
14. Advance Sensors, Actuators, Components

Objective

- Learn how to map one sensor to another.
- Learn how to use advanced sensors.
- Get familiar with EMG Sensor.

Lab Plan

This lab explains to utilize Electromyography Sensor (EMG) Sensor via Arduino. We will generate signals via moving our arm muscles.



Required Hardware Components

- PC with internet connection
- Arduino Shield for Raspberry Pi
- Grove Base Shield
- Grove LED Bar v2.1
- Grove EMG Sensor v1.1

Setting up an Arduino

1. First, open **Arduino IDE** on the lab. machine. All lab. machines have Arduino IDE already installed. If you are using your own computer you will need in install it by yourself. Instructions on how to install Arduino IDE on you computer is out of scope of this lab document. However, you may find following link useful on how to install Arduino IDE on you computer.

2. Open Arduino IDE. Connect your Arduino to the computer and wait for few seconds. Then in Arduino IDE click **Tools** → **Board** → **Arduino Leonardo** as shown in Figure 14.1. Finally set the correct **PORT** from the same section as shown in Figure 14.2

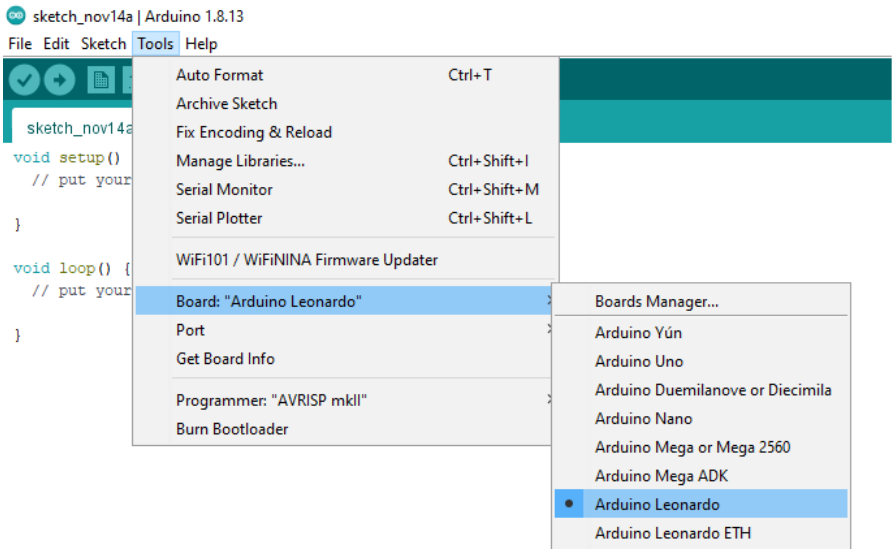


Figure 14.1: Configuring Arduino IDE Board

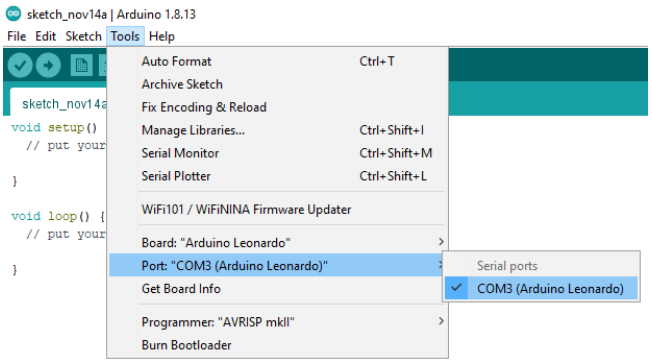


Figure 14.2: Setting The Port

3. Now place **Base Shield for Arduino** on top of Arduino microcontroller.

4. Then we can start connecting sensors and actuator to our Arduino. Plug them in accordingly:
   - **LED Bar → D8 port**
   - **EMG Detector → A0 port**

5. Before starting coding in Arduino IDE, upload the empty sketch (by clicking the top left arrow icon) to verify that you set up the Arduino correctly.

Setting up the LED Bar

6. The LED bar is a good indicator which can be used in many different scenarios such as indicating temperature, proximity, humidity, sound level etc.

7. First add the LED bar library to your setup. You can download the library from our GitLab repository.

8. To add the library, click on **Sketch**, then **Include Library**, and finally **Add .Zip Library...**. Select the correct library and wait for Arduino to install it.

9. From **File** tab, select **Examples**. Then select **Bounce**.

10. Set the following line as below as we are using the **D8 port**:

```
Grove_LED_Bar bar(9, 8, 0, LED_BAR_10);
```

11. Observe the behaviour of LED bar. You can several parameters to understand how the code works.

Setting up the EMG Detector

12. Our muscles generate EMG signals. We use EMG detectors to interpret those signals.

13. Do not remove covers of electrodes yet.

14. Get the required code from our GitLab repository.

15. Upload the code to your Arduino.

16. Now remove the cover of your electrodes and stick them to your arm.

17. First, let your arm **relax** and observe the LED bar.

18. Then move your muscles. When your muscles move they generate EMG signals. The way we code our Arduino is that the level of LED bar indicates the magnitude of those signals.

19. The example experiment is shown in Figure 14.3.

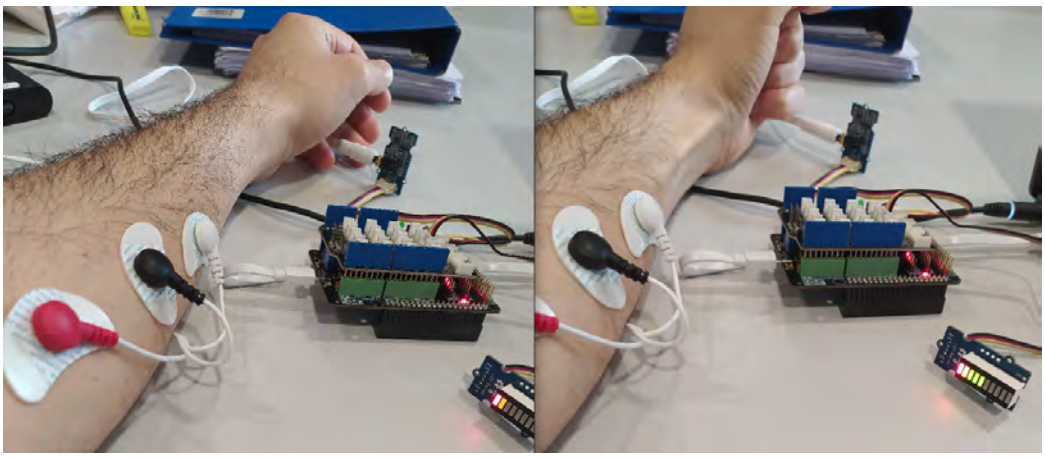
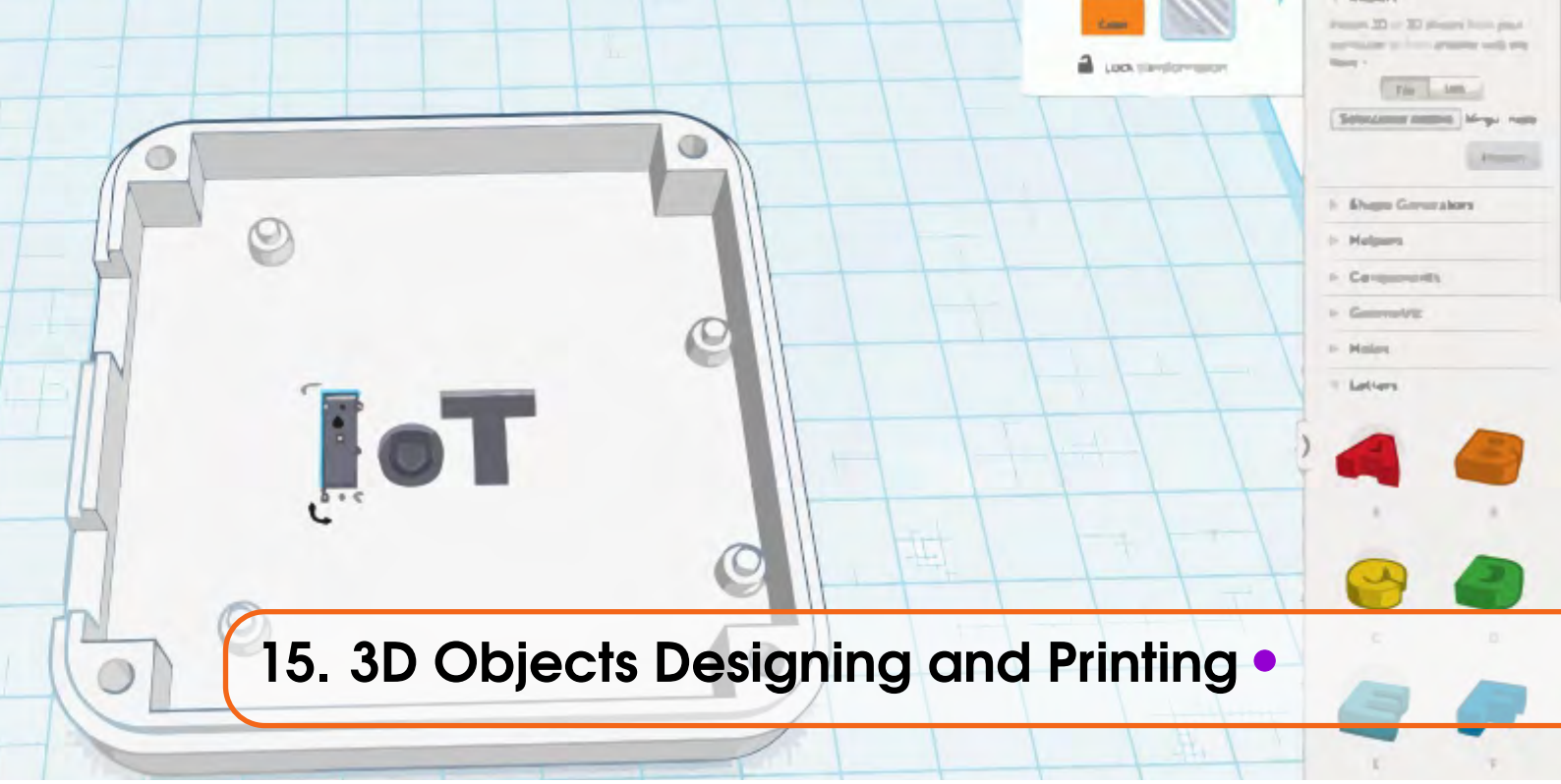


Figure 14.3: Example Experiment.

20. **Use Case Scenario (Optional)**: Think this as it is your **FINAL TASK**. Try to print EMG values to RGB LCD. Set the color of RGB as same as your LED bar.

> **— Further Reading.** .
>
> - You can learn more about the EMG detector from the following link: `https://wiki.seeedstudio.com/Grove-EMG_Detector/`



15. 3D Objects Designing and Printing

Objective

- Learn how to design 3D models
- Learn how to use Tinkercad
- Learn how to Generate STL files.

Required Hardware Components

- PC with internet connection

Setting up Tinkercad

1. Go to `www.tinkercad.com`.
2. Click on *Start Tinkering* button.
3. You can create an Tinkercad account with your google account.
4. If you have done everything correct, you should have a similar workspace as shown in Figure 15.1.
5. Now we can start Tinkering!
6. Click on create new design that you will see under **My recent designs**.
7. Now you should see a 2D workplace as in Figure 15.2.
8. Use your mouse wheel and right click to control the workplane.

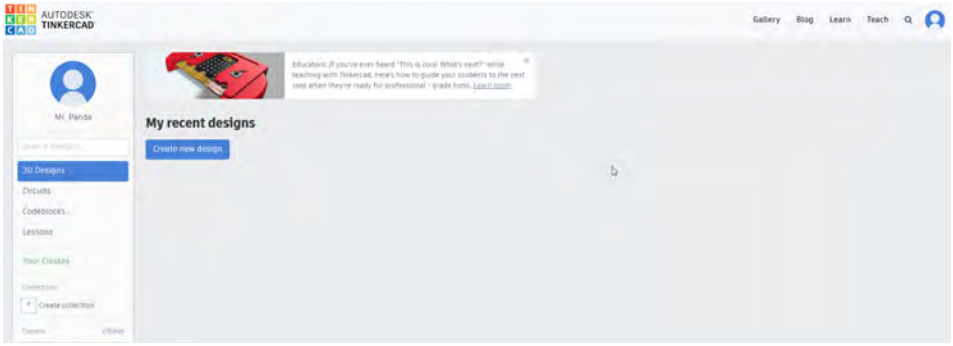


Figure 15.1: Tinkercad Workspace

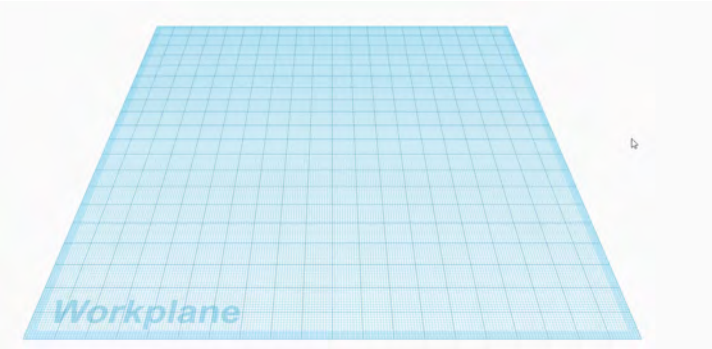


Figure 15.2: Tinkercad Workplane

Creating Objects in Tinkercad

9. Let's create our first object.
10. Drag and drop a box from the right object panel.
11. Set its height and length to **80**, and set its width to **200.**
12. Now, you should see a box as shown in Figure 15.3.

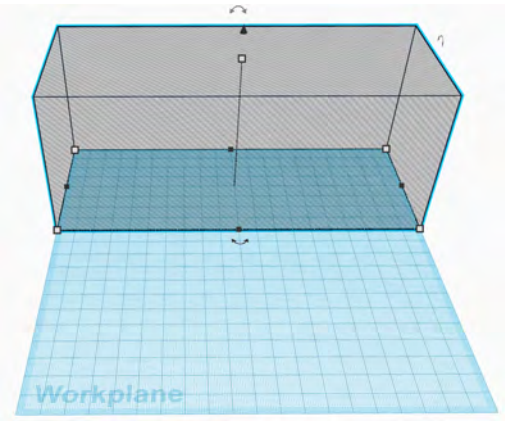


Figure 15.3: Tinkercad Box

13. Now create another box.

14. Set its height and length to **75**, and set its width to **120**.

15. Then place it inside the other box. To do that, put the box to far left corner.

16. Press **right arrow** 5 times to move 5mm right, then place **down arrow** 2 times, and finally press **CTRL + UP arrow** 5 times to set 5mm above the workplane.

17. If you look at from left, you should see a similar setup as shown in Figure 15.4.

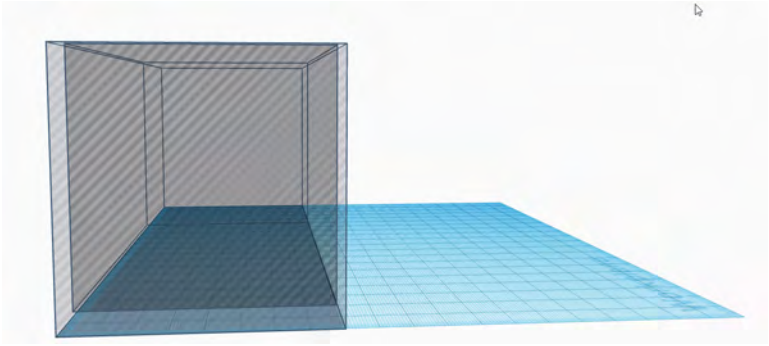


Figure 15.4: Tinkercad Box Placement

18. Imagine that is a place to store our staff.

19. Now drag and drop 1 cylinder.

20. Set its width and length to **55mm**, set its height to **75mm**.

21. Place it next to the inner box and set it 5mm above from the plane. Make outer box **solid.** Now your setup should be similar to as shown in Figure 15.5.

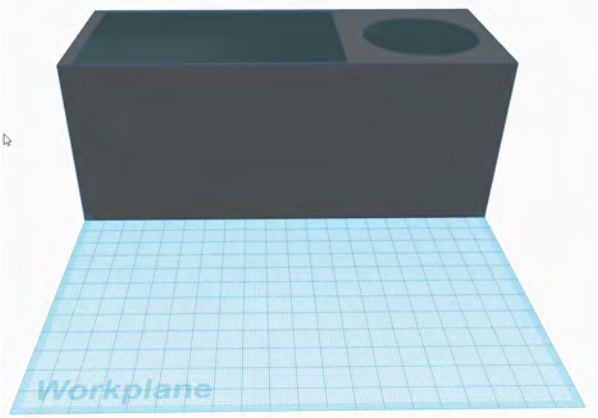


Figure 15.5: Tinkercad Cylinder Placement

22. Imagine we will use that to store pens/pencils.

23. Oh no! We do not have any space to store our IoT components.

24. Now that you now how to create and shape objects, create something as shown in Figure 15.6 by yourself.

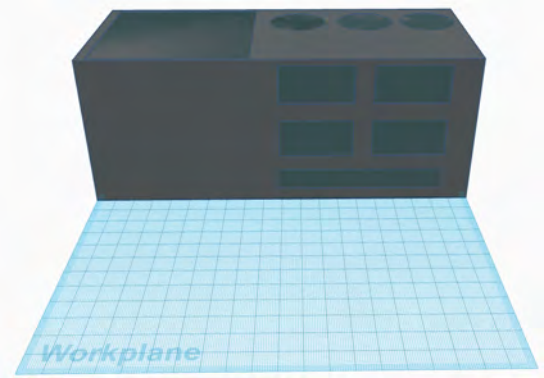


Figure 15.6: Tinkercad Final Setup

25. Now we have a well built IoT box. We can store Pi and Arduino on left side, while we stone smaller Grove components (e.g., temperature and humidity sensor) on 6 small boxes. we can store our cables at the bottom long box, while we can store our pens/pencils at 3 cylinder boxes.

26. We can export our model in STL format. **STL means Standard Triangle Language** which is the common file format of CAD software.

27. To export the 3D design as **STL**, click on **Export** from top right.

28. Then you should see a similar output as shown in Figure 15.7.

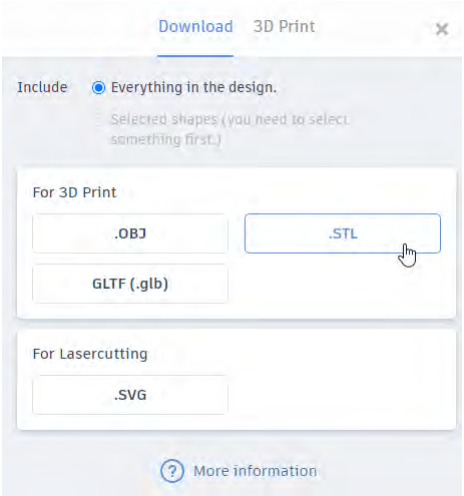
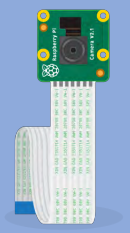


Figure 15.7: Exporting as STL

— Further Reading. .

- You can learn more about Tinkercad on the following link: `https://www.tinkercad.com/learn/designs`


16. Getting Started with Raspberry Pi Camera

Objective

- Learn how to connect a camera to Raspberry Pi
- Learn how to view camera's output using Python
- Learn how to save images and video from camera using command line
- Learn how to save images and video from camera in python
- Learn how to apply basic effects on camera source

Lab Plan

This lab explains how to connect a Raspberry Pi camera to Raspberry Pi to take images and record video. We will also learn how to apply some basic image filters.

Figure 16.1: Raspberry Pi Camera connected to a Raspberry Pi

Required Hardware Components

- SD Card with OS installed on it (we'll work on the Raspberry Pi OS)
- Display and HDMI cable
- Raspberry Pi Camera V2

- Keyboard and mouse
- Power supply

Getting Started with Raspberry Pi Camera

Connecting the Camera with Raspberry Pi

All current models of Raspberry Pi have a port for connecting the Camera Module. **Note:** If you want to use a Raspberry Pi Zero, you need a Camera Module ribbon cable that fits the Raspberry Pi Zero's smaller Camera Module port.

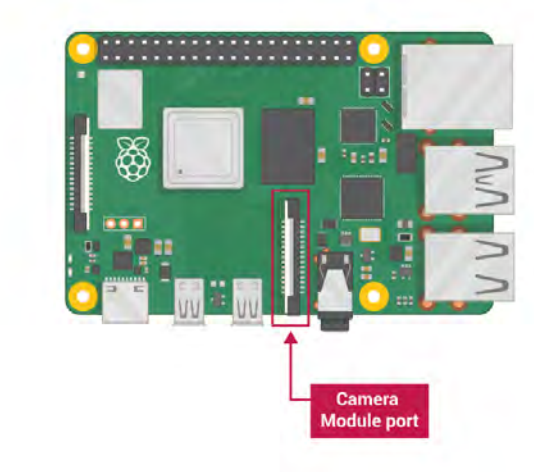


Figure 16.2: Raspberry Pi Camera Module Connector

Figure 16.3: Raspberry Pi Camera Module

There are two versions of the Camera Module:

1. The standard version, which is designed to take pictures in normal light
2. The NoIR version, which does not have an infrared filter, so you can use it together with an infrared light source to take pictures in the dark

Connect the Camera Module

Ensure your Raspberry Pi is turned off.

1. Locate the Camera Module port as shows in Figure **??**
2. Gently pull up on the edges of the port's plastic clip
3. Insert the Camera Module ribbon cable; make sure the connectors at the bottom of the ribbon cable are facing the contacts in the port.
4. Push the plastic clip back into place

[itemsep=10pt]

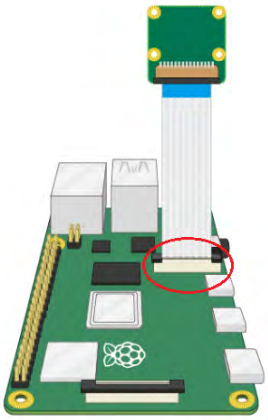


Figure 16.4: Raspberry Pi Camera Connection

Enable Camera Interface

After connecting the cameras, you need to enable camera interface in Raspberry Pi Configuration tool. Follow these steps to complete this operation.

1. Start up your Raspberry Pi
2. Go to the main menu and open the **Raspberry Pi Configuration** tool as shows in Figure 16.5.

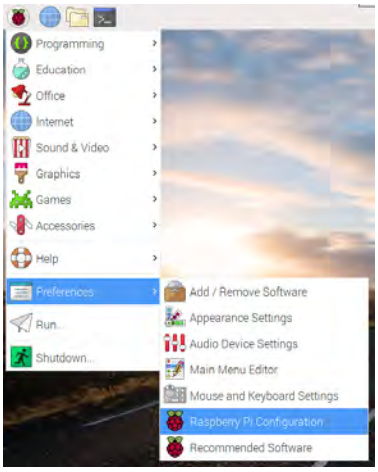


Figure 16.5: Raspberry Pi Configuration Menu

3. Select the **Interfaces** tab and ensure that the camera is **enabled** as shown in 16.6.

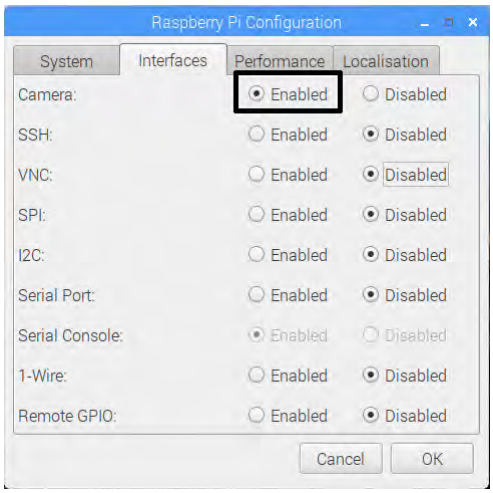


Figure 16.6: Raspberry Pi Interface Configuration

4. Reboot your Raspberry Pi

[itemsep=10pt]

How to control Camera Module via the command line

Now your Camera Module is connected and the software is enabled, try out the command line tools **raspistill** and **raspivid**.

- Open terminal window by clicking on terminal icon in the taskbar or by selecting from menu
- Type "*raspistill -o Desktop/image.jpg*" in terminal window without "" and press Enter key.

When the command runs, you can see the camera preview open for five seconds before a still picture is taken.

- Look for the picture file icon on the Desktop, and double-click the file icon to open the picture.

By adding different options, you can set the size and look of the image the **raspistill** command takes.

- For example, add **-h** and **-w** to change the height and width of the image
- Type "*raspistill -o Desktop/image-small.jpg -w 640 -h 480*" in terminal window without "" and press Enter key.
- Now record a video with the Camera Module by using the following **raspivid** command
- Type "*raspivid -o Desktop/video.h264*" in terminal window without "" and press Enter key.
- In order to play the video file, double-click the **video.h264** file icon on the Desktop to open it in VLC Media Player.

For more information and other options you can use with these commands, read the documentation for *raspistill* and the documentation for *raspivid*.

How to control the Camera Module with Python code

The Python *picamera* library allows you to control your Camera Module and create amazing projects.

- Open a Python 3 editor, such as **Thonny Python IDE**
- Open a new file and save it as **camera.py** (DO NOT SAVE THE FILE AS **picamera.py**)
- Enter the following code

```
from picamera import PiCamera
from time import sleep
camera = PiCamera()
camera.start_preview()
sleep(5)
camera.stop_preview()
```

- Save and run your program. The camera preview should be shown for five seconds and then close again

Note: the camera preview only works when a monitor is connected to your Raspberry Pi. If you are using remote access (such as SSH or VNC), you won't' see the camera preview.

- If your preview is upside-down, you can rotate it by 180 degrees with the following code

```
camera = PiCamera()
camera.rotation = 180
```

You can rotate the image by **90**, **180**, or **270** degrees. To reset the image, set **rotation** to **0** degrees.

- Make the camera preview see-through by setting an **alpha** level

```
camera.start_preview(alpha=200)
```

The **alpha** value can be any number between **0** and **255**.

Take still pictures with Python code

Now use the Camera Module and Python to take some still pictures.

- Amend your code to add a **camera.capture()** line

```
camera.start_preview()
sleep(5)
camera.capture('/home/pi/Desktop/image.jpg')
camera.stop_preview()
```

Note: it's important to **sleep** for at least two seconds before capturing an image, because this gives the camera's sensor time to sense the light levels.

- Run the code.

You should see the camera preview open for five seconds, and then a still picture should be captured. As the picture is being taken, you can see the preview briefly adjust to a different resolution. Your new image should be saved to the Desktop.

- Now add a loop to take five pictures in a row

```
camera.start_preview()
for i in range(5):
    sleep(5)
    camera.capture('/home/pi/Desktop/image%s.jpg' % i)
camera.stop_preview()
```

The variable **i** counts how many times the loop has run, from **0** to **4**. Therefore, the images get saved as **image0.jpg**, **image1.jpg**, and so on.

- Run the code again and hold the Camera Module in position.

The camera should take one picture every five seconds. Once the fifth picture is taken, the preview closes.

- Look at your Desktop to find the five new pictures.

Recording video with Python code

- Amend your code to remove **capture()** and instead add **start_recording()** and **stop_recording()**
  Your code should look like this now:

```
camera.start_preview()
camera.start_recording('/home/pi/Desktop/video.h264')
sleep(5)
camera.stop_recording()
camera.stop_preview()
```

- Run the code.

Your Raspberry Pi should open a preview, record 5 seconds of video, and then close the preview.

How to change the image settings and add image effects

The Python *picamera* software provides a number of effects and configurations to change how your images look.
Note: some settings only affect the preview and not the captured image, some affect only the captured image, and many others affect both.

Set the image resolution

You can change the **resolution** of the image that the Camera Module takes.
By default, the image resolution is set to the resolution of your monitor. The maximum resolution is 2592×1944 for still photos, and 1920×1080 for video recording.

- Use the following code to set the **resolution** to maximum and take a picture.
  Note: you also need to set the frame rate to **15** to enable this maximum resolution.
  The minimum resolution is 64×64.

```
camera.start_preview()
camera.start_recording('/home/pi/Desktop/video.h264')
sleep(5)
camera.stop_recording()
camera.stop_preview()
```

- Try taking a picture with the minimum resolution.

Add text to your image

You can add text to your image using the command **annotate_text**.

- Run this code to try it

```
camera.start_preview()
camera.annotate_text = "Hello world!"
sleep(5)
camera.capture('/home/pi/Desktop/text.jpg')
camera.stop_preview()
```

Change the look of the added text

- Set the text size with the following code

```
camera.annotate_text_size = 50
```

You can set the text size to anything between **6** to **160**. The default size is **32**.
It's also possible to change the text colour.

- First of all, add **Color** to your **import** line at the top of the program

```
from picamera import PiCamera, Color

```

- Then below the **import** line, amend the rest of your code so it looks like this

```
    camera.start_preview()
    camera.annotate_background = Color('blue')
    camera.annotate_foreground = Color('yellow')
    camera.annotate_text = " Hello world "
    sleep(5)
    camera.stop_preview()
```

Change the brightness of the preview

You can change how bright the preview appears. The default brightness is **50**, and you can set it to any value between **0** and **100**.

- Run the following code to try this out

```
    camera.start_preview()
    camera.brightness = 70
    sleep(5)
    camera.capture('/home/pi/Desktop/bright.jpg')
    camera.stop_preview()
```

- The following loop adjusts the brightness and also adds text to display the current brightness level

```
    camera.start_preview()
    for i in range(100):
        camera.annotate_text = "Brightness: %s" % i
        camera.brightness = i
        sleep(0.1)
    camera.stop_preview()
```

Change the contrast of the preview

Similarly to the preview brightness, you can change the contrast of the preview.

- Run the following code to try this out:

```
    camera.start_preview()
    for i in range(100):
        camera.annotate_text = "Contrast: %s" % i
        camera.contrast = i
        sleep(0.1)
    camera.stop_preview()
```

Add cool image effects

You can use **camera.image_effect** to apply a particular image effect.
The image effect options are:

- none
- negative
- solarize
- sketch
- denoise
- emboss
- oilpaint
- hatch

- gpen
- pastel
- watercolor
- film
- blur
- saturation
- colorswap
- washedout
- posterise
- colorpoint
- colorbalance
- cartoon
- deinterlace1
- deinterlace2

The default effect is **none**.

- Pick an image effect and try it out

```
camera.start_preview()
camera.image_effect = 'colorswap'
sleep(5)
camera.capture('/home/pi/Desktop/colorswap.jpg')
camera.stop_preview()
```

- Run this code to loop over **all** the image effects with **camera.IMAGE_EFFECTS**

```
camera.start_preview()
for effect in camera.IMAGE_EFFECTS:
    camera.image_effect = effect
    camera.annotate_text = "Effect: %s" % effect
    sleep(5)
camera.stop_preview()
```

Set the image exposure mode

You can use **camera.exposure_mode** to set the exposure to a particular mode. The exposure mode options are:

- off
- auto
- night
- nightpreview
- backlight

Figure 16.7: Raspberry Pi Interface Configuration

- spotlight
- sports
- snow
- beach
- verylong
- fixedfps
- antishake
- fireworks

The default mode is **auto**.

- Pick an exposure mode and try it out:

```
camera.start_preview()
camera.exposure_mode = 'beach'
sleep(5)
camera.capture('/home/pi/Desktop/beach.jpg')
camera.stop_preview()
```

- You can loop over all the exposure modes with **camera.EXPOSURE_MODES**, like you did for the image effects.

Change the image white balance

You can use **camera.awb_mode** to set the auto white balance to a preset mode. The available auto white balance modes are:

- off
- auto
- sunlight
- cloudy

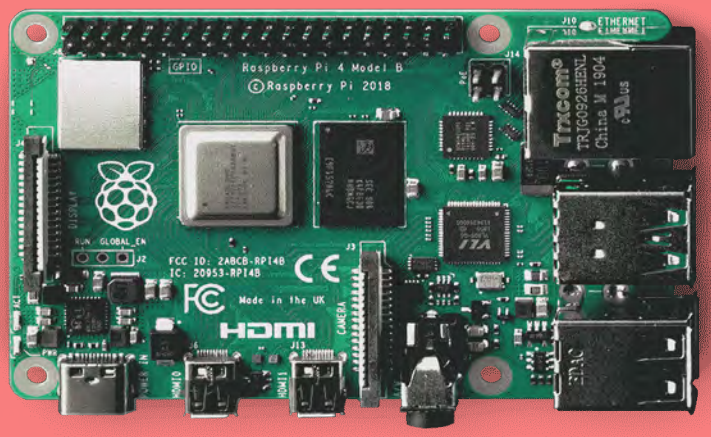
- shade
- tungsten
- fluorescent
- incandescent
- flash
- horizon

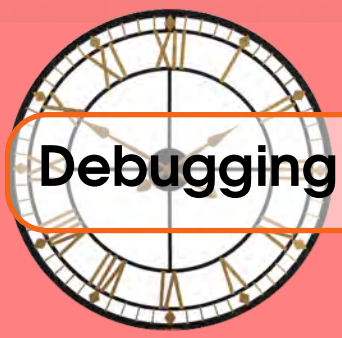
The default is **auto**.

- Pick an auto white balance mode and try it out:

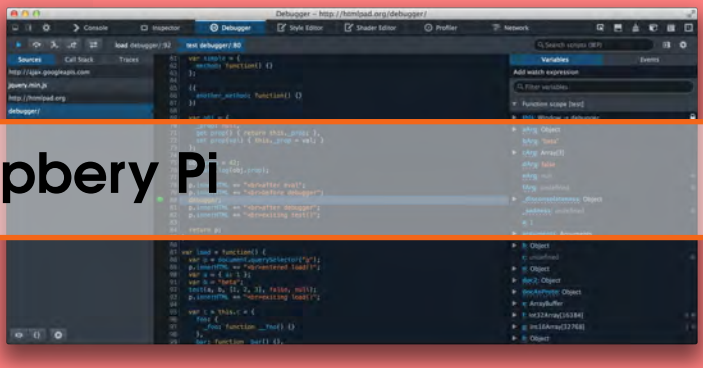
```
    camera.start_preview()
    camera.awb_mode = 'sunlight'
    sleep(5)
    camera.capture('/home/pi/Desktop/sunlight.jpg')
    camera.stop_preview()
```

- You can loop over all the auto white balance modes with **camera.AWB_MODES**, like you did for the image effects.





Debugging The Raspbery Pi



Setting Correct Time and Date

One of the known issues of Raspberry Pi OS (previously called Raspbian) is the incorrect time and date. You need to set them right to continue to work on Raspberry Pi. Follow the below steps to set the correct time and date:

- Open the Terminal (CTRL-ALT-T).
- Type the following by changing the values with current time and date to manually set:

```
sudo date -s "19/09/2020 11:00"
```

- Then go into settings via:

```
sudo raspi-config
```

and select the **Localization Options** then **Change Time Zone** to set the correct date and time as seen in Figure 16.8.

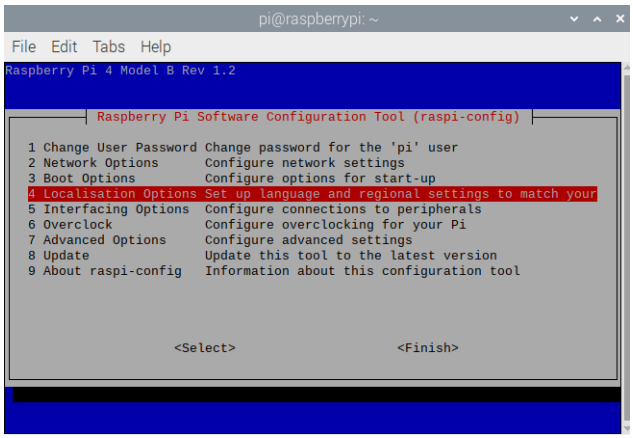


Figure 16.8: Set The Time Zone

Now your Raspberry Pi should show the exact time and date.

Updating The Raspberry Pi

You may encounter variety of errors due to utilizing old packages. Then updating the Pi may fix your error. However **ONLY APPLY THIS** if you encounter an error during the labs and **CLOSE** all other applications while updating the system:

```
sudo apt update && sudo apt full-upgrade
sudo reboot
```

Know The Difference: While *update* command updates the package listings the *full-upgrade* installs the latest version of the available packages. This is why these two are usually run in order. Then you need to reboot the Pi in order for the changes to take effect.