

Week 1: Basic electronic circuit 1

Tools in the box

Multimeter (*aka volt-ohm-milliammeter*)



- A measuring instrument that can measure multiple electrical properties, for example :
 - Voltage
 - Resistance
 - Current
- You can test open or closed/short circuit by using multimeter, fyi, open circuit = disconnected (*The multimeter value will be close to zero*) whereas closed/short circuit = connected (*The multimeter value will be OL*).

Raspberry Pi



- A small and affordable board
- There are 3 series of Raspberry Pi :
 - Raspberry Pi
 - Raspberry Pi Zero
 - Raspberry Pi Pico

- Most of the Raspberry Pi has **40 GPIO pins** which includes:
 - Eight (8) Ground (GND)
 - Two (2) 3.3V pins
 - Two (2) 5V pins
 - Twenty-eight (28) GPIO pins



Explanation for each GPIO pins

Ground (GND)

- Can be used to measure voltage and complete a circuit
- Zero voltage
- Mainly for safety purpose :
 - Lighting rod - It will provide low-resistance path to ground
 - Get rid of electric shock

3.3V

- Offers a stable 3.3V supply to power and test the LEDs

5V

- Give direct access to the 5v supply coming from your mains adaptor, less power than used by the Raspberry Pi itself.

I2C - Inter-Integrated Circuit

More info can found on the reference link

SPI - Serial Peripheral Interface

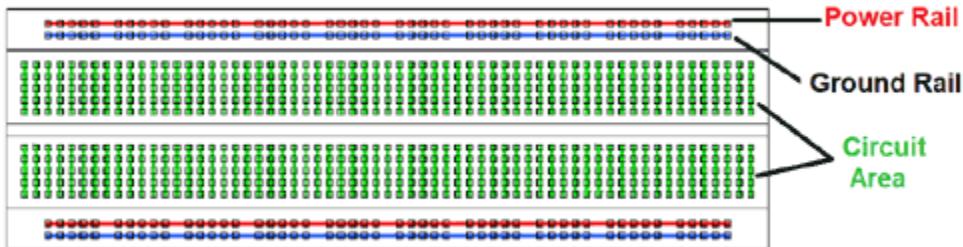
More info can found on the reference link

UART - Universal Asynchronous Receiver / Transmitter

More info can found on the reference link

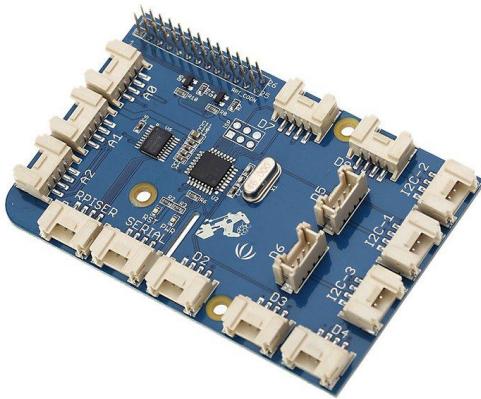
Reference link: <https://www.tomshardware.com/reviews/raspberry-pi-gpio-pinout,6122.html>

Breadboard



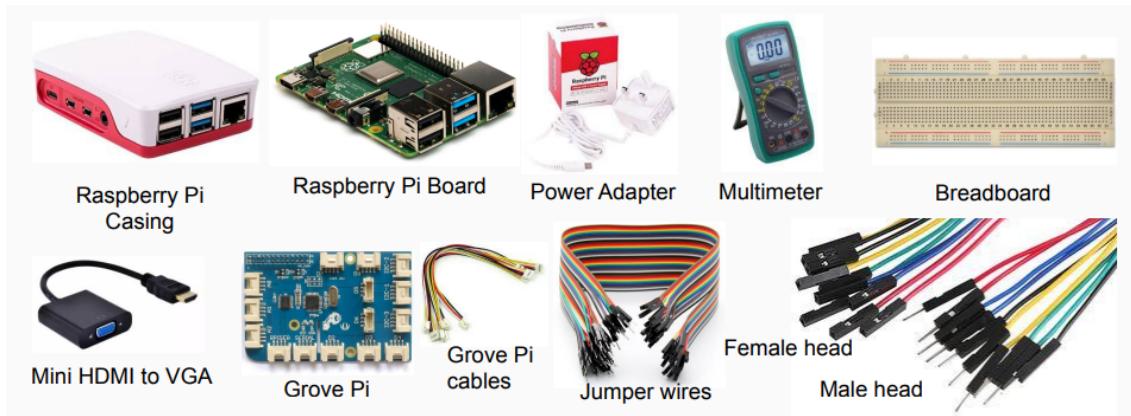
- A thin plastic board used to hold electronic components (transistors, resistors, chips, etc.) that are wired together
- A construction base for prototyping of electronics

Grove Pi Board

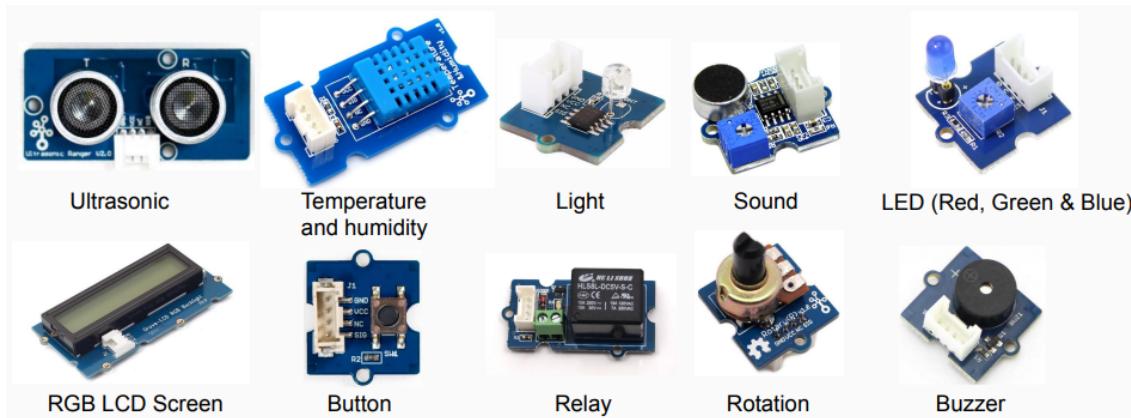


- Electronics board that can connect to hundreds of different sensors in which we can program them to monitor, control, and automate devices.

Other things in the box



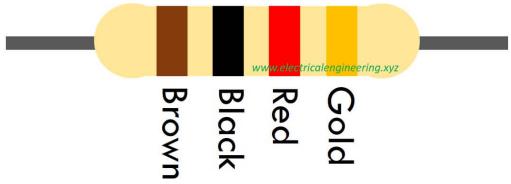
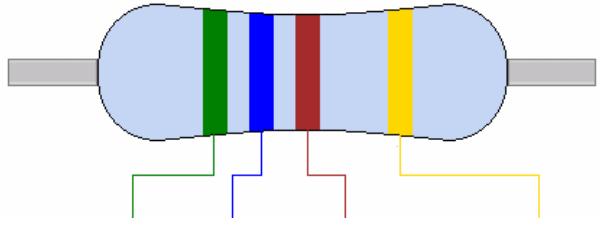
Sensor



Electronic components

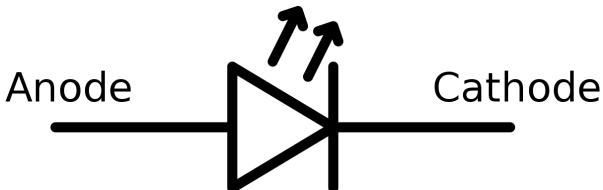
Resistor

- Resists the current flow

Common types of resistor	Electronic Symbol
<p>Brown-black-red-gold : $1\text{K}\Omega \pm 5\%$</p> 	
<p>Green-blue-brown-gold : $560\Omega \pm 5\%$</p> 	
<p>brown-black-orange -gold : $10\text{K}\Omega \pm 5\%$</p> <p>10K-ohm resistor</p> 	

LED

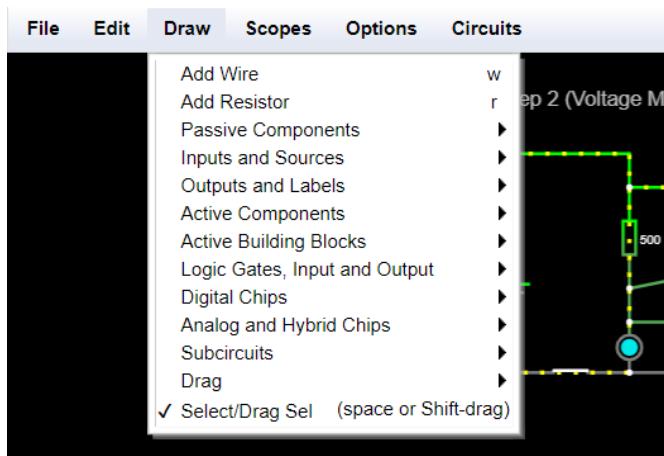
- Semiconductor light source that emits light when current flows through it

LEDs	Electronic Symbol
	

Falstah (Electronic Circuit Simulator)

- A simple circuit can be constructed and demo using Falstah

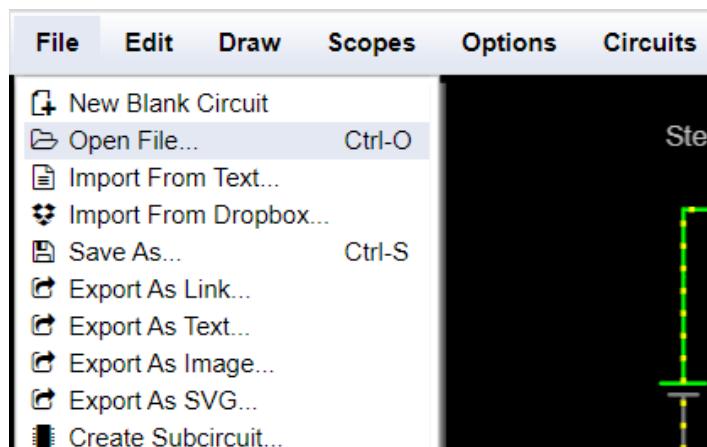
1. You can insert electronic components, wire, passive components, etc in the [Draw](#)



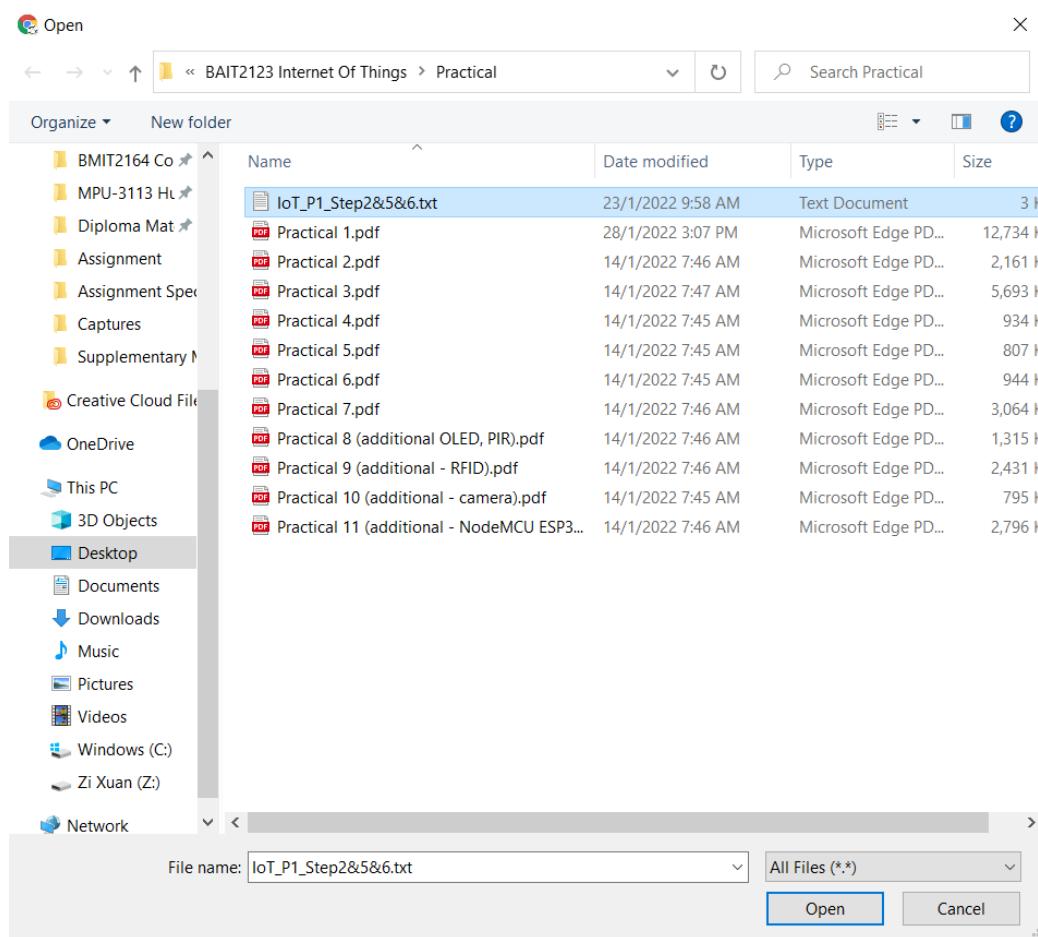
*** Alternatively, you can insert new electronic components by enter short form, for example, 'w' to add a wire into the circuit

2. You can open a text file that is saved from Falstah to your local PC. In my case, I open a txt file called "IoT_P1_Step2&5&6.txt" to open a pre-saved file.

Step 1: Open file

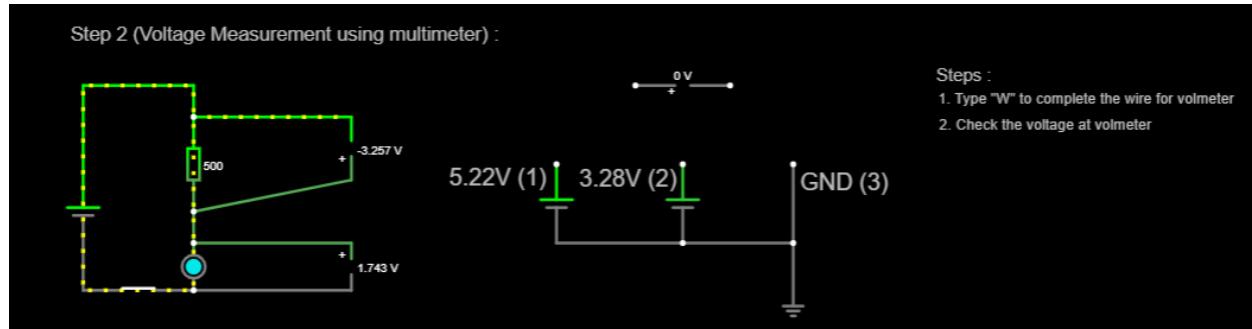


Step 2: Select the text file and click open

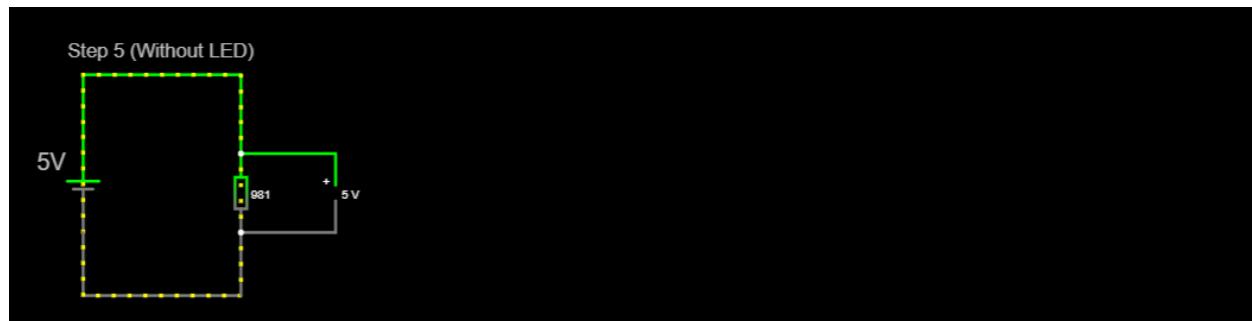


Overview of Practical 1 Simulation

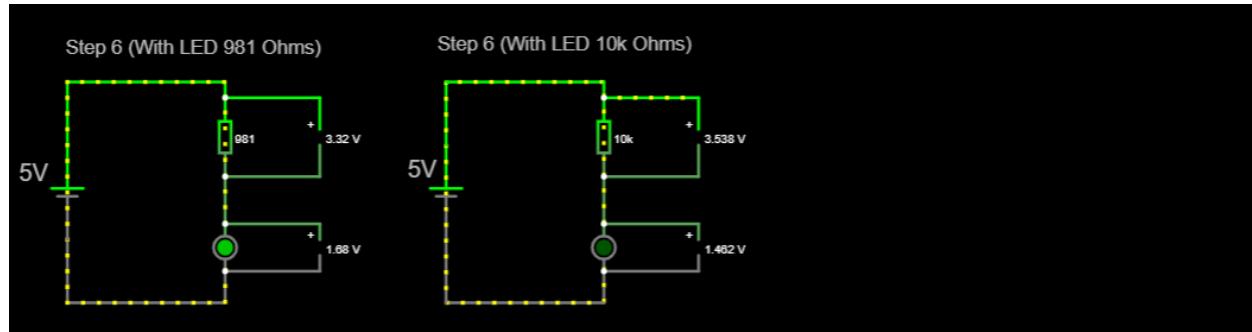
Step 2 Learn to measure voltage using a multimeter



Step 5 Construct a resistor circuit without LED

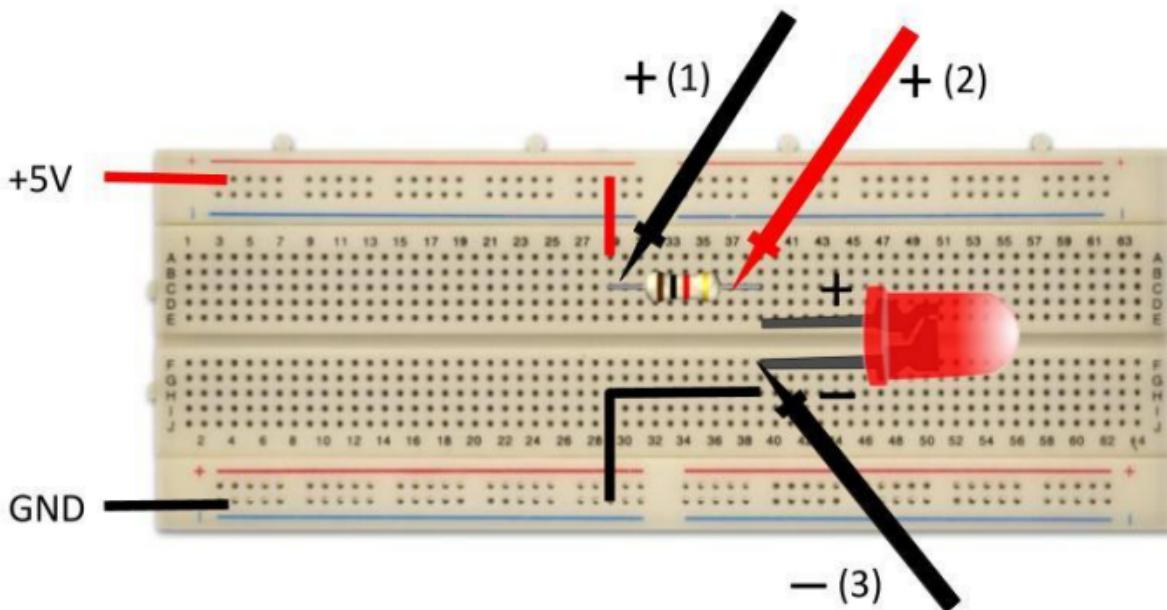


Step 6 Construct a resistor circuit with LED

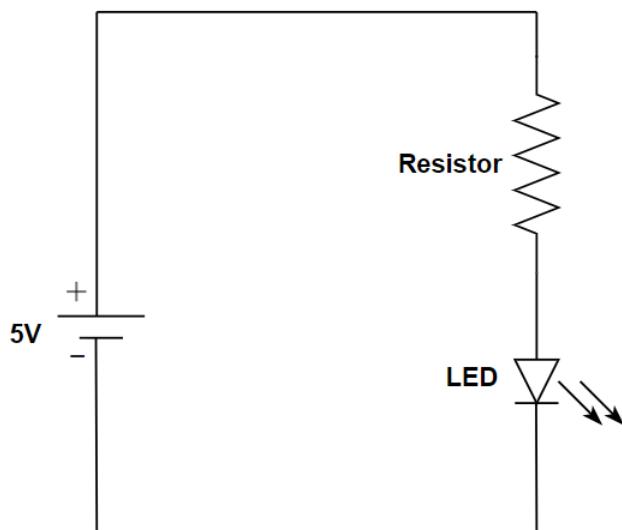


Practical 1 Quiz

Question: Draw the resistor with LED circuit using standard electronic symbols



Electronic circuit diagram



Week 2: Basic electronic circuit 2

2.1 Transistor ([SMALL SIGNAL NPN TRANSISTOR](#))

- Incompatible voltage between Raspberry Pi GPIO (i.e 3.3V) and voltage required by the relay, we will use the transistor as driver to control GPIO pin
- Insufficient current to energize the relay's coil, we will use the driver which is a transistor
- It can act as conductor or insulator
- It amplifies low input current to high output current (and hence power)

Transistor	Electronic Symbol
 **bipolar junction transistor (BJT)	INTERNAL SCHEMATIC DIAGRAM DS10130

1 - Emitter : The outlet for electrical supply (connected to GND)

2 - Base : The gate controller device for the largest electrical supply

3 - Collector : Larger electrical supply

How do transistors Act as Conductors or Insulators?

Important equation for transistor:

$I_C = I_B * h_{FE}$, true iff $V_{CE} > 0V$ // V_{CE} is the voltage measured between the collector and emitter

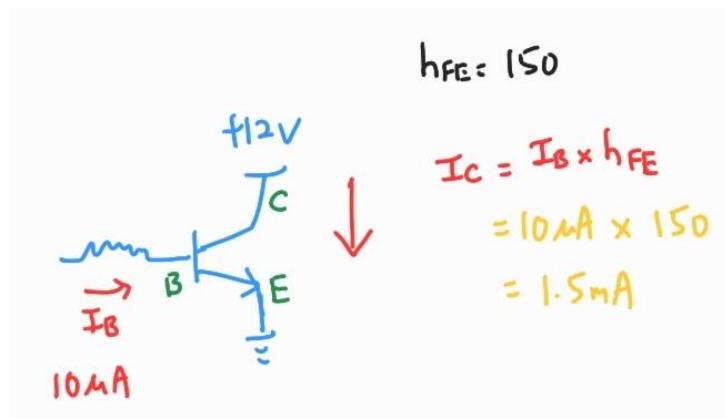
$h_{FE} = 150$ // It is fixed at this time

1. Transistor act as a *conductor* - It will amplify the current in the collector (C) by 150 times

$$I_B = 10\mu A$$

$$\begin{aligned}I_C &= 10\mu A * 150 \\&= 1.5mA\end{aligned}$$

Diagram Illustration:



∴ 10 μ A input can amplify to 1.5mA

2. Transistor act as an *insulator*

When $I_B = 0$, the I_C will also 0 (no current flow)

∴ It will become an insulator and is totally off

2.2 Relay ([Electromechanical or Electrical Relay » Electronics Notes](#))

Relays are electric switches that use electromagnetism to convert small electrical stimuli into larger currents. Relays are highly versatile components that are just as effective in complex circuits as in simple ones.

Switching Level of Current

- One of the most common situations that require the use of a relay occurs when an application needs to switch from high to low current (or vice versa) within the same circuit.
- For example, the temperature sensors that power HVAC units require levels of amperage that vastly exceed the capacity of their wiring.
- Relays provide the necessary amplification to convert a small current into a larger one.

Benefits

- Even when an application doesn't specifically require a relay, using one can prove beneficial.
- Relays can reduce the need for high-amperage wiring and switches, which are expensive and take up space.
- Therefore, switching to relays in your electronic systems can reduce the size or weight of a casing, for instance, or allow manufacturers to fit more functionality into a space of the same size.

Relay [1.1]	Electronic Symbol
	 <p style="text-align: center;"> Relay Pin/Terminal Numbering </p>

Understand the relay connectivity and specification

**You can view the relay diagram[1.1] at above to gain necessary information

Output Control Rating Type	Voltage (V)	Current (I)	Pmax = VI
1	250VAC	10A	2.5kW
2	125VAC	10A	1.25kW
3	30VDC	10A	0.3kW
4	28VDC	10A	0.28kW

The control voltage/coil voltage is **5VDC** which means the voltage required to energize the coil is 5V, then it will generate a magnetic field, pull the switch metal and make a contact. Once it makes contact, it will be a closed circuit and allows current to flow through.

** Extra notes: 15mA 250V can kill a person.

** Extra notes: Contact switching voltage is safe voltage that the contact can switch

Question included in Practical 2

1. Voltage required from a relay to control the household light bulb

Malaysia : 250V

America / Japan : 110 - 120V

2. To control the household light bulb (e.g., 20W one light bulb), what is the maximum number of light bulbs that the relay can control?

Calculation:

$$P = VI$$

$$20 = 250(I)$$

$$I = 20/250$$

$$= 0.08A$$

$$= 80mA$$

$$\therefore I_{max}/I_{bulb} = 10A/0.08A = 125 \text{ light bulbs}$$

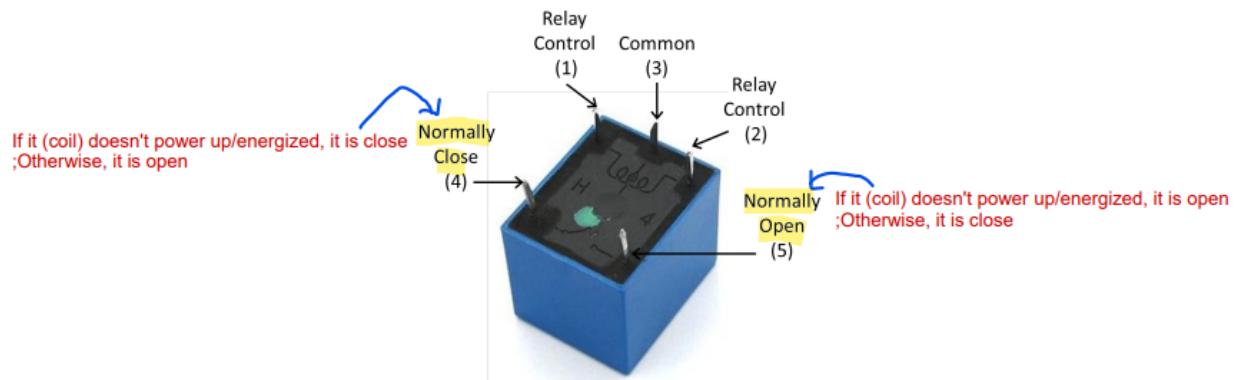
How to get I_{max} to substitute into the equation?

We can look at the AC (normally the household connected plug is AC device)

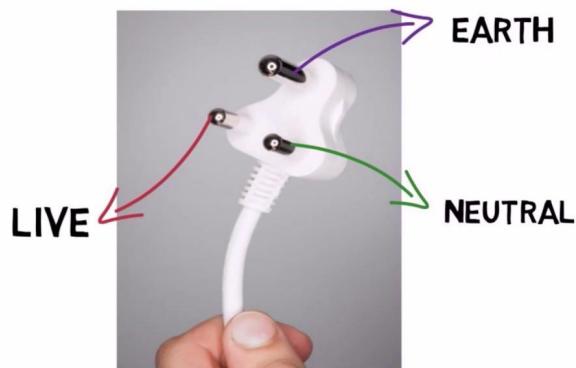
- All the figures in the relay [1.1] is the maximum voltage or current, which cannot exceed it, else it will malfunction.
- For 250AC, 10A is the maximum and if exceed it, it might not function properly (i.e. burn or cause fire).

After calculating, we will know the safe number of lightbulbs to use so that we won't reach the limit. Otherwise it might burn and melt the metal inside the relay as it generates a lot of heat then burns the relay plastics as well as the rest of the system which will cause fire.

Relay Explanation



** Extra Notes for socket [1.2]



- Earth (yellow green) - Protection so that we won't get electric shock, safe
Live (Brown) - You can feel the voltage, it is dangerous
Neutral (Blue) - Connected to earth so it is safe (*Sometimes it might be connected to live, so have to be careful)

What should plug in the relay pin?

- Input pin - Pin 1 & 2 - Either one put 5V and another put 0V as GND
Output pin - Pin 3, 4, 5

When connect to socket[1.2], we should connect high voltage to live - pin (4) or pin (5) while neutral connect to common pin (3)

Relay control coil resistance

$$\text{Relay control coil resistance } (\Omega) = 72.3\Omega$$

$$\begin{aligned} V &= IR \\ 5 &= I(72.3) \\ I &= 5/72.3 \\ &= 0.069A \\ &= 69mA \end{aligned}$$

∴ The estimated current required to control the relay is 69mA

69mA is considered big to the microcontroller which normally provides up to 25mA or slightly bigger, hence it cannot drive the coil as it does not have enough power. It will then not make a contact or another case, it will make contact but we will hear the tik tak tik tak sound (it means connected then disconnected continuously). It will destroy the pin as well because it cannot

source enough current but we're pulling that much current, it will burn the relay inside then it will spoil.

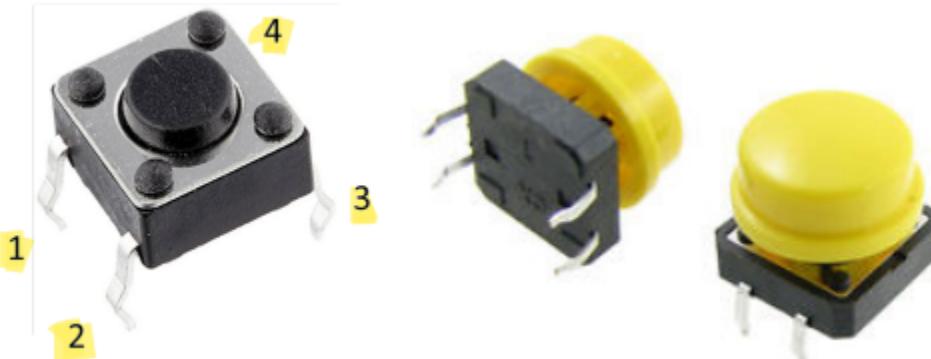
Hence, the relay will need a driver (transistor) to turn the low current input to high current output.

Connectivity among Relay's lead

Lead	Lead	Short (Just tick "✓")	Open (Just tick "✓")
3	4	✓	
3	5		✓
4	5		✓

(When coil is energized, normally close (4) will become open circuit while normally open (5) will become short circuit)

2.3 Push Button Switch

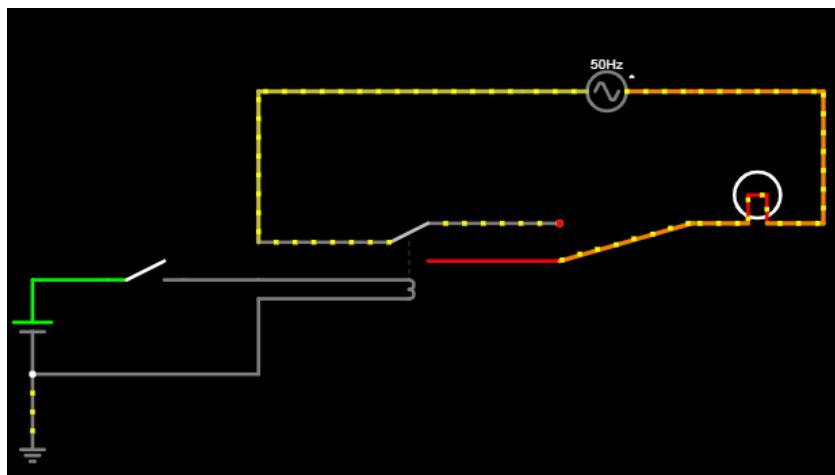


Push Button Lead Testing (When didn't press the push button switch)

Lead	Lead	Short Circuit	Open Circuit
1	2	✓	
2	3		✓
3	4	✓	
4	1		✓
1	3		✓
4	1		✓

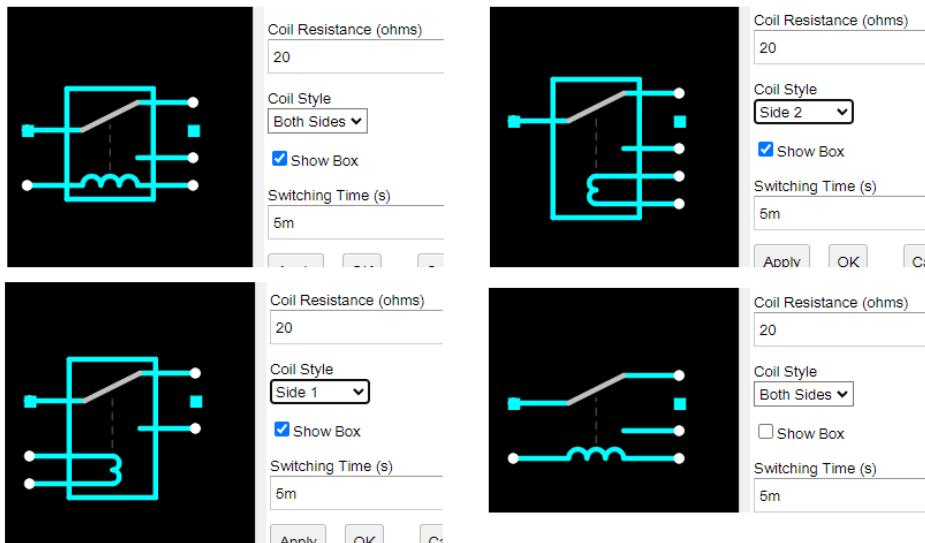
Push Button Lead Testing (When press the push button switch) All leads will be connected.

Falstad Simulation for Relay (Practical 2 Part 3 Video)



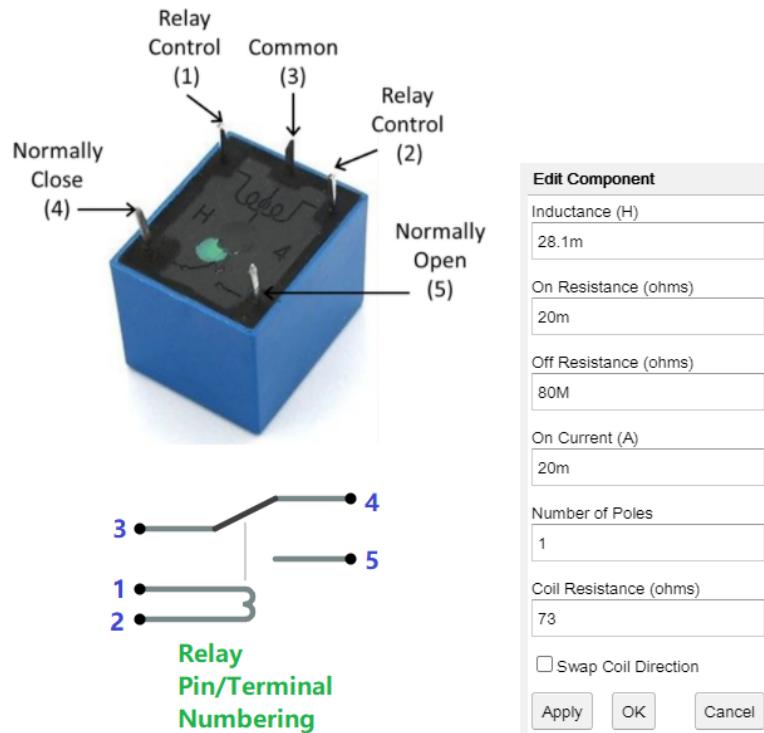
Settings for Relay	Settings for A/C sources (2 terminal)
<div style="border: 1px solid #ccc; padding: 5px;"> Edit Component Inductance (H) <input type="text" value="28.1m"/> On Resistance (ohms) <input type="text" value="20m"/> Off Resistance (ohms) <input type="text" value="80M"/> On Current (A) <input type="text" value="20m"/> Off Current (A) <input type="text" value="15m"/> Number of Poles <input type="text" value="1"/> Coil Resistance (ohms) <input type="text" value="73"/> Coil Style <input type="button" value="Both Sides ▾"/> <input checked="" type="checkbox"/> Show Box Switching Time (s) <input type="text" value="5m"/> <input type="button" value="Apply"/> <input type="button" value="OK"/> <input type="button" value="Cancel"/> </div>	<div style="border: 1px solid #ccc; padding: 5px;"> Edit Component Max Voltage <input type="text" value="250"/> Waveform <input type="button" value="A/C ▾"/> DC Offset (V) <input type="text" value="0"/> Frequency (Hz) <input type="text" value="50"/> Phase Offset (degrees) <input type="text" value="0"/> <input type="button" value="Apply"/> <input type="button" value="OK"/> <input type="button" value="Cancel"/> </div>

The schematic relay symbol may appear differently. This is because its appearance can be edited. Below is the many different appearances:



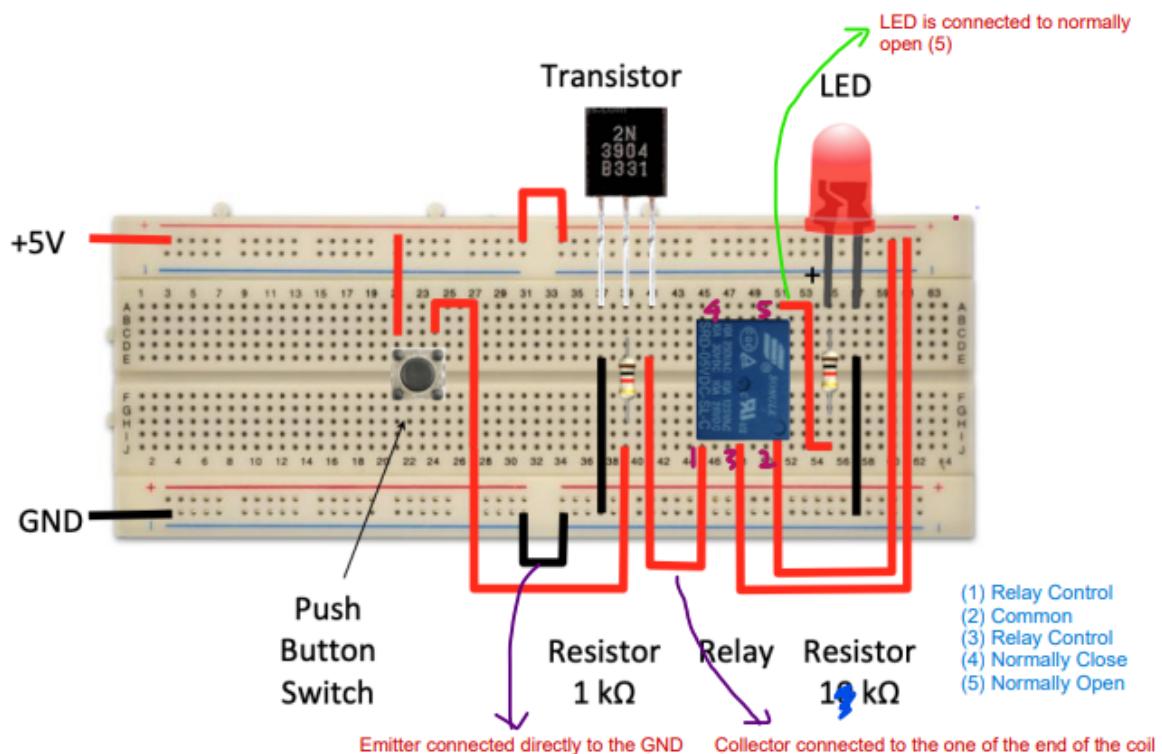
Appearance changes do not affect its electrical properties.

Below is the pin numbering of the relay to help you sketch the schematic diagram for answering Practical Quiz 2

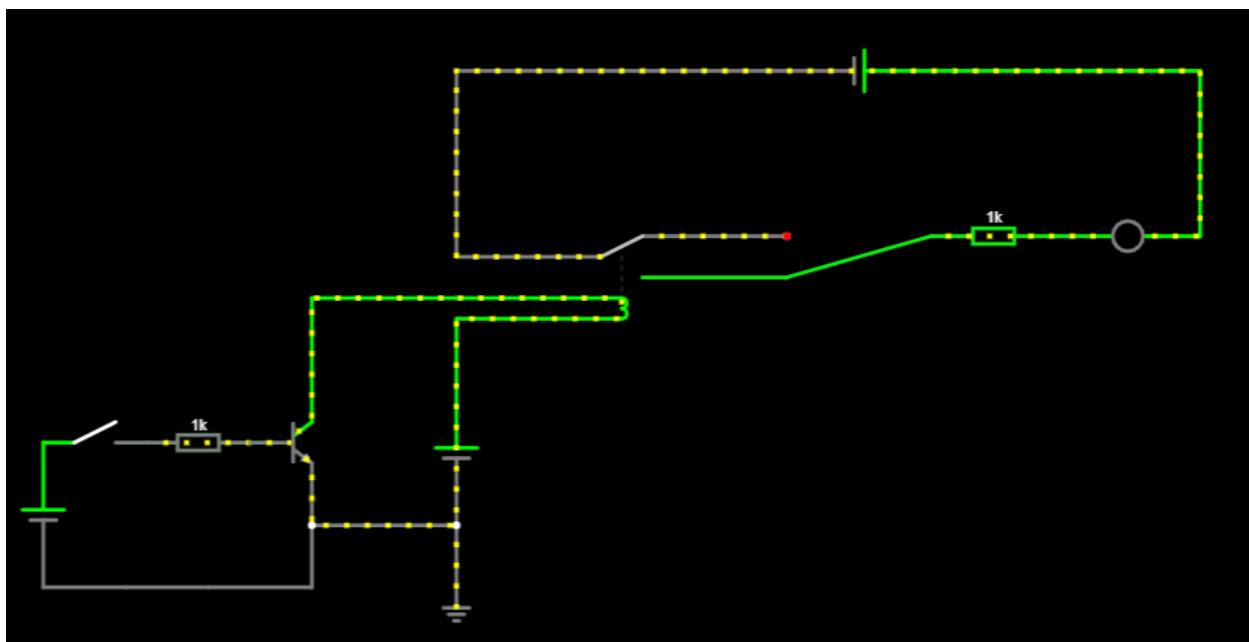


Practical 2 Quiz

Question : Draw the electronic circuit diagram and explain the simple working principle of the circuit.



Electronic circuit diagram



Simple working principle of the circuit

When the push button switch is pressed, it completes the circuit. Since there is not enough power to drive the relay's coil, a driver called the transistor needed to amplify the low current input to high current output. To illustrate, the transistor will act as a conductor to amplify the current in the collector (C) and energize the relay's coil. When there is enough current, the relay will generate a magnetic field, pull the switch metal and make a contact. Once it makes contact, it will become a closed circuit and allow the current to flow through. At this point, the LED light is on.

Question for this Quiz 2

Question

Actually we have to combine them as one circuit or remain like this and each of them have different Voltage sources to power them up?

Answer

By right for a relay, the 2 sides do not share the same power sources. The reason for using a relay is because the 2 sides have incompatible power sources. But in the practical worksheet, the experiment is a very simple one and so happens that they can use the same power source. However, if the power sources are different, then you **MUST** change it because the result would be different.

Week 3: Programming with Python

Python Library Reference link: [Python Library Documentation - Dexter Industries](#)

How to run in the Thorny Python

Test on LED Blink: *led_blink.py* (*test01.py*)

Python Command	Explanation
import time	To delay some action in order to see the effect (i.e turn on, delay for a while, vice versa)
from grovepi import *	Import everything inside Grovepi
led = 4	The led is connected to D4 (Digital 4) port
pinMode(led, "OUTPUT")	Configure the pin number 4 (D4) as an output
while True:	Loop forever when it is true
try:	Try function
time.sleep(1)	Sleep for 1 second
digitalWrite(led, 1)	Write high to the LED to turn on it
time.sleep(1)	Sleep for another 1 second
digitalWrite(led, 0)	Shut off the LED
except KeyboardInterrupt:	Same like catch function Catch the keyboard interrupt, we can use it to escape the loop (i.e ctrl c, ctrl z)
digitalWrite(led, 0)	Shut off the LED
break	Break out of this while loop and back to main function and then stop the program
except IOError:	The raspberry pi communicates with GrovePi to do the action (i.e write the led to 1, write the led to 0, etc), it communicates using i2c. Sometimes, the communication fails, when it fails, it will generate this error which indicates raspberry pi failed to communicate with the GrovePi board. One of the reasons might be you unplug/detached the GrovePi board from the raspberry pi which causes them to not communicate.

print("I/O error occurs")	Display I/O error message
break	Break out of this while loop and back to main function and then stop the program

Task 1: Find a way to allow the program code to run and two ways to stop the program.

1. Run code by pressing F5 OR Thonny > Run current script (arrow right with green bg)
2. Stop the program by pressing ctrl + F2 OR Thonny > Stop/Restart backend (red by with stop text)

Task 2: Modify the program code for LED to blink faster and slower

1. For LED blink faster - Change the time.sleep(1) to time.sleep(0.5) or shorter time
2. For LED blink slower- Change the time.sleep(1) to time.sleep(1.5) or longer time

How to run in the console

1. Click “Tools”
2. Click “Open System Shell”
3. In the shell, type python3 ./led_blink.py
4. Press enter, the led will blinking
5. Press ctrl c, the led will turn off (due to keyboard interruption)

Diagram Illustration

The screenshot shows a terminal window titled "pi@raspberrypi: ~/Desktop". The window contains the following text:

```

pi@raspberrypi:~/Desktop
File Edit Tabs Help
-----
Some Python commands in the PATH of this session:
- python    -> /usr/bin/python2.7
- python3   -> /usr/bin/python3.7
- python3.7 == /usr/bin/python3.7
- pip       == /usr/bin/pip
- pip3     == /usr/bin/pip3

pi@raspberrypi:~/Desktop $ python3 ./led_blinky.py

```

The terminal window has a dark background and light-colored text. The title bar and menu bar are visible at the top.

Test LED Fade: (*test02.py*)

Python Command	Explanation
import time	To delay some action in order to see the effect (i.e turn on, delay for a while, vice versa)
import grovepi	Import grovepi library
rotaryangle = 2	Connect rotaryangle sensor to port 2 of analogy which is A2
led = 5	Connect the LED to D5 which is digital pin 5
grovepi.pinMode(rotaryangle, "INPUT")	Set the pin mode of the rotaryangle as input because the rotaryangle is reading the signal
grovepi.pinMode(led, "OUTPUT")	Configure the pin number 4 (D4) as an output
val = 0	This val will carry the value read from rotaryangle
while True:	Loop forever when it is true
try:	Start try block
time.sleep(1)	Sleep for 1 second
val = grovepi.analogRead(rotaryangle)	Read the analog from the rotaryangle through the grovepi library and store it in val
print(val)	Print out the val so that we can see the value from the console
grovepi.analogWrite(led, val//4)	Set the LED but this setting is different from what we did before. In the previous program, we will digital write the LED to high or low. This analog is a special function which will take in the value from 0 to 255, and then will generate a square wave. The LED will switch on and off very quickly but at different duty cycles, and therefore produce varying intensity. <i>Notes: analogRead values go from 0 to 1023, analogWrite values from 0 to 255</i>
except KeyboardInterrupt:	Same like catch function Catch the keyboard interrupt, we can use it to escape the loop (i.e ctrl c, ctrl z)
digitalWrite(led, 0)	Shut off the LED

break	Break out of this while loop and back to main function and then stop the program
except IOError:	The raspberry pi communicates with GrovePi to do the action (i.e write the led to 1, write the led to 0, etc), it communicates using i2c. Sometimes, the communication fails, when it fails, it will generate this error which indicates raspberry pi failed to communicate with the GrovePi board. One of the reasons might be you unplug/detached the GrovePi board from the raspberry pi which causes them to not communicate.
print("I/O error occurs")	Display I/O error message
break	Break out of this while loop and back to main function and then stop the program

test03.py

Python Command	Explanation
import time	To delay some action in order to see the effect (i.e turn on, delay for a while, vice versa)
import math	Import math library
from grovepi import *	Import all the functions inside Grovepi
buzzer = 2	Connect the buzzer to D2
button = 4	Connect the button to D4
pinMode(buzzer, "OUTPUT")	Set the pin mode of buzzer to output as we gonna buzz the buzzer
pinMode(button, "INPUT")	Set the pin mode of button to input as we need to read the button statement whether it is low (0V) or high (3.3V)
while True:	Loop forever when it is true
try:	Start try block
time.sleep(0.25)	Sleep for 0.25 seconds
bStatus = digitalRead(button)	Read the button using digitalRead and place the value inside the bStatus (0=>not pressed, 1=>pressed)

if(bStatus):	If the button has been pressed, enter inside that statement
digitalWrite(buzzer, 1)	Write 1 to buzzer (Turn on the buzzer)
else:	If the button hasn't been pressed
digitalWrite(buzzer, 0)	Write 0 to buzzer (Turn off the buzzer)
except KeyboardInterrupt:	Same like catch function Catch the keyboard interrupt, we can use it to escape the loop (i.e ctrl c, ctrl z)
digitalWrite(buzzer, 0)	Turn off the buzzer
break	Break out of this while loop and back to main function and then stop the program
except IOError:	The raspberry pi communicates with GrovePi to do the action (i.e write the led to 1, write the led to 0, etc), it communicates using i2c. Sometimes, the communication fails, when it fails, it will generate this error which indicates raspberry pi failed to communicate with the GrovePi board. One of the reasons might be you unplug/detached the GrovePi board from the raspberry pi which causes them to not communicate.
print("I/O error occurs")	Display I/O error message
break	Break out of this while loop and back to main function and then stop the program

Task 1: Modify the code to make the buzzer sound like a warning siren / ambulance.

Buzzer.py

Python Command	Explanation
import time	To delay some action in order to see the effect (i.e turn on, delay for a while, vice versa)
from grovepi import *	Import all the functions inside Grovepi
buzzer = 3	Connect the buzzer to D3
pinMode(buzzer, "OUTPUT")	Set the pin mode of buzzer to output as we gonna buzz the buzzer
while True:	Loop forever when it is true
val = input("Enter a value between 0 - 255 and assign to val")	User input for a value between 0 - 255 and assign to val
val = int(val)	Convert the val to integer
if val > 255:	If the val value is greater than 255
val = 255	The value of val will be 255
elif val < 0:	Else if the val value is lesser than 0
val = 0	The value of val will be 0
analogWrite(buzzer, val)	Write the val value to the D3 pin <i>*In this case, buzzer is 3 which is D3</i>

Result:

255
50
255
50

can make the buzzer sound like a warning siren / ambulance

** D3, D5 and D6 pin allows us to write 8 bits value which is from 0 - 255 using analogWrite **
** A0, A1 and A2 pin allows us to read 10 bits value which is from 0 - 1023 using analogRead **

Practical 3 Quiz

Why do we need `i//4` at line 16? *

4/4

```
test02.py ✘
1 import time
2 import grovepi
3
4 rotaryangle = 14
5 led = 6
6
7 grovepi.pinMode(rotaryangle, "INPUT")
8 grovepi.pinMode(led, "OUTPUT")
9 i = 0
10
11 while True:
12     try:
13         time.sleep(1)
14         i = grovepi.analogRead(rotaryangle)
15         print(i)
16         grovepi.analogWrite(led, i//4)
17     except IOError:
18         print("IO Error occurs")
```

- To ensure the `i` value in return of 4 decimal points from LED.
- To ensure the value assigned to LED (output) is divided by 4.
- To ensure led brightness act according to the rotary angle value.
- To ensure the analog read in return of 4 decimal points from rotary angle.
- To ensure 10 bits values converted to 8 bits value.
- To ensure 8 bits value converted to 10 bits value.
- To ensure the correct value is assigned to the LED (output).

Feedback

Well done.

To let the LED blink faster (on and off time consistently), we should modify the 4/4 code to become: *

```
test01.py *»
1 import time
2 from grovepi import *
3
4 led = 4
5 pinMode(led, "OUTPUT")
6 while True:
7     try:
8         time.sleep(1)
9         digitalWrite(led, 1)
10        time.sleep(1)
11        digitalWrite(led, 0)
12    except KeyboardInterrupt:
13        digitalWrite(led, 0)
14        break
15    except IOError:
16        print("IO Error occurs")
17
```

- Line 8 : time.sleep(1.5) Line 10: time sleep (1.5)
- Line 8 : time.sleep(0.5) Line 10: time sleep (1.5)
- Line 10: time.sleep(0.5)
- Line 8: time sleep (1.5)
- Line 8 : time.sleep(0.5) Line 10: time sleep (0.5)
- Line 10: time sleep (1.5)
- Line 8 : time.sleep(1.5) Line 10: time sleep (0.5)
- Line 8 : time.sleep(0.5)

Feedback

Modify the sleep time to 0.5 (shorter sleeping time) consistently for on and off time, to blink faster.

What is the purpose of having time.sleep(1) at line 8 and 10 ? *

4/4

```
test01.py *  
1 import time  
2 from grovepi import *  
3  
4 led = 4  
5 pinMode(led, "OUTPUT")  
6 while True:  
7     try:  
8         time.sleep(1)  
9         digitalWrite(led, 1)  
10        time.sleep(1)  
11        digitalWrite(led, 0)  
12    except KeyboardInterrupt:  
13        digitalWrite(led, 0)  
14        break  
15    except IOError:  
16        print("IO Error occurs")  
17
```

- To have sufficient time (one second) to rest and clear memory before the next task.
- To hold on for one second and follow by turning on / off LED for the blinking effect.
- To have one second for initializing the LED light before it turns on / off.

Feedback

For human to sense the one second blinking time.

Week 4: Programming with Python

Step 1: Test Ultrasonic and Relay

ultrasonic_with_relay.py

Python Command	Explanation
from time import *	Import all the functions in the time library
from grovepi import *	Import all the functions inside Grovepi
ultrasonic_port = 3	Set the ultrasonic port to D3
relay_port = 2	Set the relay port to D2
pinMode(relay_port, "OUTPUT")	Set the relay port as output *Important as if you didn't specify it as output, you will not be able to control it *While for the ultrasonic port, it is handled by library, so we don't have to specify it as input or output
while True:	Loop forever
try:	Try block
sleep(0.5)	Sleep for 0.5 seconds
distance = ultrasonicRead(ultrasonic_port)	Read from the ultrasonic D3 port and return the distance Notes: ultrasonicRead(pin) is a grove specific function that used to Read the distance reading from the Grove Ultrasonic ranger
print('Distance : ', distance, 'cm')	Print the distance on the terminal
if distance <= 10:	If the distance is less than 10
digitalWrite(relay_port, 1)	Turn on the relay / Write the HIGH value to relay_port (D2 pin)
else:	otherwise;
digitalWrite(relay_port, 0)	Turn off the relay / Write the LOW value to relay_port (D2 pin)
except KeyboardInterrupt:	Same like catch function Catch the keyboard interrupt, we can use it to escape the loop (i.e ctrl c, ctrl z)
digitalWrite(relay_port, 0)	Turn off the relay

break	Break out of this while loop and back to main function and then stop the program
except TypeInterrupt:	None of the function being called within this try block, this error will occur
print("Type error occurs")	Display Type error message
break	Break out of this while loop and back to main function and then stop the program
except IOError:	The raspberry pi communicates with GrovePi to do the action (i.e write the led to 1, write the led to 0, etc), it communicates using i2c. Sometimes, the communication fails, when it fails, it will generate this error which indicates raspberry pi failed to communicate with the GrovePi board. One of the reasons might be you unplug/detached the GrovePi board from the raspberry pi which causes them to not communicate.
print("I/O error occurs")	Display I/O error message
break	Break out of this while loop and back to main function and then stop the program

Step 2: Test RGB LCD Display and DHT (Digital Humidity and Temperature)

dht_with_lcd.py

Python Command	Explanation
from time import *	Import all the functions in the time library
from grovepi import *	Import all the functions inside Grovepi
from grove_rgb_lcd import *	Import all the functions to control the lcd
DHT11 = 0	Specify DHT11 as DHT module type that we want to communicate
dht_sensor = 3	Connect dht sensor to D3
pinMode(dht_sensor, "INPUT")	Configure the pin mode of DHT sensor as input
setRGB(5, 15, 10)	Set the rgb to color cyan
while True:	Loop forever

try:	Try block
sleep(0.5)	Sleep for 0.5 seconds
[temp, hum] = dht(dht_sensor, DHT11)	Read from the dht sensor which will return 2 parameters/array of 2 floats (temperature and humidity)
print('Temp: ', temp, '\u00b0', 'Hum: ', hum, '%')	Print the temperature and humidity on the console
t = str(temp)	Convert the temperature from integer to string * When we want to display text on LCD, it must be in string format
h = str(hum)	Convert the humidity from integer to string
setText('Temp: ', t, '\u00b0', 'Hum: ', h, '%')	Print the text on lcd
except KeyboardInterrupt:	Same like catch function Catch the keyboard interrupt, we can use it to escape the loop (i.e ctrl c, ctrl z)
setText("Program Exited")	Set the text of lcd to "Program Exited"
break	Break out of this while loop and back to main function and then stop the program
except TypeError:	None of the function being called within this try block, this error will occur
print("Type error occurs")	Display Type error message
break	Break out of this while loop and back to main function and then stop the program
except IOError:	The raspberry pi communicates with GrovePi to do the action (i.e write the led to 1, write the led to 0, etc), it communicates using i2c. Sometimes, the communication fails, when it fails, it will generate this error which indicates raspberry pi failed to communicate with the GrovePi board. One of the reasons might be you unplug/detached the GrovePi board from the raspberry pi which causes them to not communicate.

print("I/O error occurs")	Display I/O error message
break	Break out of this while loop and back to main function and then stop the program

Test RGB Color

rgb.py

Python Command	Explanation
from grove_rgb_lcd import *	Import all the functions to control the lcd
while True:	Loop forever
val = input("Enter RGB values: ")	Get user input and store inside the val
rgb = val.split(',')	Split the val using comma(,)
num = len(rgb)	Return the length of the list (e.g 1, 2, 3 ...)
if num < 3 or num > 3:	If the length of the list is more than or less than 3
print('Three values needed, but you gave {}, format(len(rgb)))	It will print the message to alert user to input exact 3 values
continue	The continue statement rejects all the remaining statements in the current iteration of the loop and moves the control back to the top of the loop
setRGB(int(rgb[0]), int(rgb[1]), int(rgb[2])))	Set the rgb based on the user input
text = "R:{}G:{}B:{}".format(rgb[0], rgb[1], rgb[2])	The rgb value will be stored on text variable
setText(text)	Print out the text inside the text variable

Sample Output from LCD

Red



Blue



Cyan



Cyan



Pink



Yellow





Question for Practical 4 Worksheet

1. Can the fake device library emulate a relay?

There is no virtual relay. But you can see visually the state of the pin (connected to the relay). If the pin is set to high (then by right the relay should turn on). Similarly, if the pin is set to low (then the relay should be off). You have to imagine that the state of the pin you see is the state of the relay you are controlling.

2. How to emulate a buzzer controlled from a relay?

You can emulate a buzzer by configuring the launcher script to beep when a pin is set to high. Although there is no relay in between the pin and the buzzer, the effect is the same with the physical setup. Just that you will not be hearing the click-clack sound generated by the relay.

3. Can we use analogWrite(.) instead of digitalWrite(.) to sound the buzzer?

Yes, by right both can generate sound. The sound generated by digitalWrite(.) is only a single tone. If you use analogWrite(.), you can produce different, though limited, numbers of tones. You can watch me demonstrate this in this 2-minute [video](#). The tones are not the best tones you get from a perfect tone generator. This is true only when you use real Grove Pi. But, in the fake device, I did not emulate this. So at the moment you *cannot* use analogWrite(.) under the fake device library.

4. Can we use analogWrite(.) to control the sound volume of a buzzer?

No, the analogWrite(.) works differently from the actual analog output. Grove Pi generates a high frequency square wave on the output pin to create the illusion that controls the brightness of an LED connected to the pin. It works on the LED. But it does not work the same for a speaker or buzzer.

Practical 4 Quiz

How to set the LCD display to have Red background? *

4 points

```
test05.py*
1 from time import *
2 from grovepi import *
3 from grove_rgb_lcd import *
4
5 dhtsensor = 7
6
7 pinMode(dhtsensor, "INPUT")
8
9 while True:
10     try:
11         sleep(0.5)
12         [temp, hum] = dht(dhtsensor, 0)
13         print("Temp = ", temp, '\u00b0C', " Hum = ", hum, " %")
14         t = str(temp)
15         h = str(hum)
16         setRGB(0, 255, 0)
17         setText("Temp = " + t + '\u00b0C' + "Hum = " + h + " %")
18     except KeyboardInterrupt:
19         setText("Program Exited")
20         break
21     except TypeError:
22         print("Type Error occurs")
23     except IOError:
24         print("IO Error occurs")
25
```

- At line 16, change to setRGB(0, 255, 255)
- At line 16, change to setRGB(255, 0, 255)
- At line 16, change to setRGB(255, 255, 0)
- At line 16, change to setRGB(255, 0, 0)
- At line 16, change to setRGB(0, 0, 255)

The maximum and minimum GrovePi Ultrasonic sensors value is... *

4 points

```
test04.py ✘
1 from time import *
2 from grovepi import *
3
4 ultrasonic = 3
5 relay = 2
6
7 pinMode(ultrasonic, "INPUT")
8 pinMode(relay, "OUTPUT")
9
10 while True:
11     try:
12         sleep(0.5)
13         distance = ultrasonicRead(ultrasonic)
14         print(distance, 'cm')
15         if distance <= 10:
16             digitalWrite(relay, 1)
17         else:
18             digitalWrite(relay, 0)
19     except KeyboardInterrupt:
20         digitalWrite(relay, 0)
21         break
22     except TypeError:
23         print("Type Error occurs")
24     except IOError:
25         print("IO Error occurs")
26
```

- max = 50, min = 10
- max = 100, min = 10
- max = 65535, min 5
- max = 500, min = 0
- max = 255, min = 10

Notes: This prototype GrovePi Ultrasonic sensor able to measure distance from 0 mm to 500 mm (some says 0 - 350mm), depends on the quality / specification of ultrasonic sensor

Referring to the following code, what are the steps to set the buzzer sounds like a warning siren ? * Answers (in steps) are indicated continuously. *

4 points

```
10 while True:  
11     try:  
12         time.sleep(0.25)  
13         bStatus = digitalRead(button)  
14         if bStatus:  
15             digitalWrite(buzzer,1)  
16         else:  
17             digitalWrite(buzzer,0)  
18     except KeyboardInterrupt:  
19         digitalWrite(buzzer, 0)  
20         break  
21     except IOError:  
22         print("IO Error occurs")  
23
```

- Replace the function from digitalWrite to analogRead at line 15, 17, 19.
- Replace the function from digitalWrite to analogWrite at line 15, 17, 19.
- Change the button pinMode to "OUTPUT" at line 8
- Change the assigned value from 1 to 0, at line 15.
- Change the assigned value from 1 to 255, at line 15.
- Increase the time.sleep from 0.25 to 5 at line 12.
- Add a new line: time.sleep(0.25) in between 18 and 19.
- Add a new line: time.sleep(0.25) in between 14 and 15.
- Add two new lines: time.sleep(0.25) ; analogWrite(buzzer, 100) in between 21 and 22.
- Add two new lines: time.sleep(0.25) ; analogWrite(buzzer, 100) in between 15 and 16.

Improved Code to set the buzzer sounds like a warning siren:

```
while True:  
    try:  
        time.sleep(0.25)  
        bStatus = digitalRead(button)  
        if(bStatus):  
            time.sleep(0.25)  
            analogWrite(buzzer, 255)  
            time.sleep(0.25)  
            analogWrite(buzzer, 100)  
    except KeyboardInterrupt:  
        analogWrite(buzzer, 0)  
        break  
    except IOError:  
        print("IO Error occurs")
```

Notes: D3, D5 and D6 pin allows us to write 8 bits value which is from 0 - 255 using analogWrite

Notes: A0, A1 and A2 pin (aka D14, D15, D16) allows us to read 10 bits value which is from 0 - 1023 using analogRead

Week 5 Working with Firebase

Step 1: Register an account at <https://firebase.google.com/>, refer to *Setup - Firebase.pdf*

- * *Do not use TAR UC student gmail account as it doesn't come with Google Firebase features.*
- * *Recommend to create a new Gmail account to be used among team members (gmail password is used as part of log in procedure to be shown in your python code)*

Reference:  [Setup - Firebase.pdf](#)

Step 2: There are many ways to establish connection to Google Firebase.

We use Pyrebase, that is a simple python wrapper for the Firebase API, pre-installed in our Raspberry Pi module

- * If you are using your own Raspberry Pi, do install Pyrebase by typing:
sudo pip3 install Pyrebase
- * Reference: <https://github.com/thisbejim/Pyrebase>

Reference: [Pyrebase - A simple python wrapper for the Firebase API.](#)

Step 3: Open Thonny Python (IDE) - open a new file - save as test06.py - code the following
test06.py ([test06.py](#))

```
from time import *
from grovepi import *
from grove_rgb_lcd import *
from pyrebase import pyrebase

dhtsensor = 7
pinMode(dhtsensor, "INPUT")

config = {
    "apiKey": "AIzaSyBkGe8rwXaekDyGa68SN5BndPlN-6XYY-g",
    "authDomain": "bait2123-iot-12d48.firebaseio.com",
    "databaseURL": "https://bait2123-iot-12d48-default-rtdb.firebaseio.com/",
    "storageBucket": "gs://bait2123-iot-12d48.appspot.com"
}

firebase = pyrebase.initialize_app(config)
auth = firebase.auth()
user = auth.sign_in_with_email_and_password("zixuan2001711@gmail.com",
"loozx123")
db = firebase.database()

while True:
    try:
        sleep(1.5)
        [temp, hum] = dht(dhtsensor, 0)
        print("Temp = ", temp, '\u00b0C', " Hum = ", "%")
```

```

t = str(temp)
h = str(hum)
setRGB(0,255,0)
setText("Temp = " + t + '\337' + "C Hum = "+ h + "%")
data1 = {"temperature":t}
data2 = {"humidity":h}
results = db.child("PI_001").push(data1, user['idToken'])
results = db.child("PI_001").push(data2, user['idToken'])
except KeyboardInterrupt:
    setText("Program Exited")
    break
except TypeError:
    print("Type error occurs")
    break
except IOError:
    print("I/O error occurs")
    break

```

**You have to replace the “text” in red color in order to suit your case (based on configuration that you had done in your Firebase)*

Step 4: Open a new file - save as *test06_launcher.py* - code the following.

Note that **you will run the code using this python file (*test06_launcher.py*) not *test06.py*.**

test06_launcher.py ([test06_launcher.py](#))

```

from FakeDevices import *
import logging

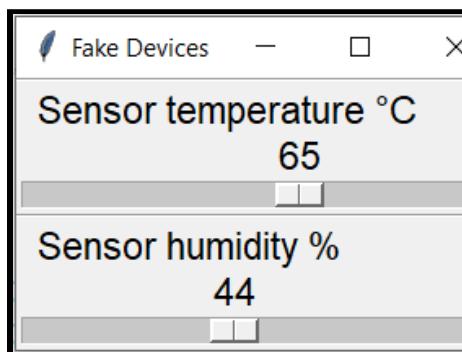
log = logging.getLogger(__name__)

try:
    gui = Gui()
    gui.add(DHT(7, 'Sensor'))

    from test06 import *
except:
    log.exception('-----Log-----')

```

Step 5: Run the *test06.py* and adjust the fake device’s sensor value.



Adjust the value by swipe right or left

```
Temp = 65 °C Hum = 44%
R:0, G:255, B:0
Temp = 65°C Hum
= 44%
```

Console Output in Thonny Python

You can see the temperature and humidity that was set in the console.

Now, let's go into **Firebase - Realtime Database**. Here, you can see the data (Temp = 65°C Hum = 44%) is being inserted into the real time database.

The screenshot shows the Firebase Realtime Database interface. At the top, the URL is https://bait2123-iot-12d48-default-rtdb.firebaseio.com/. Below the URL are standard browser controls for zooming and navigating. The main area displays a tree structure of database nodes. One node, "Sensor Value", has two children: "-Mw7HE2Y8VN-sJL2tJS6" and "-Mw7HE6SYqSqdH0hXOjk". The first child has a "temperature" child with value "65" and a "humidity" child with value "44". The second child also has "temperature" and "humidity" children, both with value "65". At the bottom left, there is a note: "Database location: United States (us-central1)".

My Firebase link

https://console.firebaseio.google.com/u/0/project/bait2123-iot-12d48/database/bait2123-iot-12d48-default-rtdb/data/~2FPI_001

Extra Task for Practical 5

Task 1: Test your temperature and humidity sensor by facing it with a few breaths and show it in your firebase's database (Since I am using the fake device, thus I can't demonstrate this part, but basically the humidity and temperature will increase when you breathe through the dht sensor). Modify the code to display the values without adding a new "child" (updating the same child values for every 2 seconds).

test06.py

```
from time import *
from grovepi import *
from grove_rgb_lcd import *
from pyrebase import pyrebase

dhtsensor = 7
pinMode(dhtsensor, "INPUT")

config = {
    "apiKey": "AIzaSyBkGe8rwXaekDyGa68SN5BndPlN-6XYg",
    "authDomain": "bait2123-iot-12d48.firebaseio.com",
    "databaseURL": "https://bait2123-iot-12d48-default-rtdb.firebaseio.com/",
    "storageBucket": "gs://bait2123-iot-12d48.appspot.com"
}

firebase = pyrebase.initialize_app(config)
auth = firebase.auth()
user = auth.sign_in_with_email_and_password("zixuan2001711@gmail.com",
"loozx123")
db = firebase.database()

while True:
    try:
        sleep(2)
        [temp, hum] = dht(dhtsensor, 0)
        print("Temp = ", temp, '\u00b0C', " Hum = ", "%")
        t = str(temp)
        h = str(hum)
        setRGB(0,255,0)
        setText("Temp = "+ t + '\u00b0C' + " Hum = " + h + "%")
        data1 = {"temperature":t}
        data2 = {"humidity":h}

        # Updating same child values for every 2 seconds
        results = db.child("PI_001").set({
            'Sensor Values': {
                'temperature': data1,
                'humidity': data2
            }
        })
    except KeyboardInterrupt:
        setText("Program Exited")
        break
```

```
except TypeError:  
    print("Type error occurs")  
    break  
except IOError:  
    print("I/O error occurs")  
    break
```

test06_launcher.py

```
from FakeDevices import *  
import logging  
  
log = logging.getLogger(__name__)  
  
try:  
    gui = Gui()  
    gui.add(DHT(7, 'Sensor'))  
  
    from test06 import *  
except:  
    log.exception('-----Log-----')
```

Useful link for firebase

https://firebase.google.com/docs/database/admin/save-data#python_2
[Pyrebase/README.md at master](#)

Practical 5 Quiz

At line 25 and 29, what is the indication for '\u00b0' and '\337' ? Why do we need to have line 26 and 27 ? * 4/4

```
test06.py
1 from time import *
2 from grovepi import *
3 from grove_rgb_lcd import *
4 from pyrebase import pyrebase
5
6 dhtsensor = 7
7 pinMode(dhtsensor, "INPUT")
8
9 config = {
10     "apiKey": "[API_KEY]",
11     "authDomain": "[PROJECT_ID].firebaseapp.com",
12     "databaseURL": "https://[DATABASE_NAME].firebaseio.com",
13     "storageBucket": "[PROJECT_ID].appspot.com"
14 }
15 firebase = pyrebase.initialize_app(config)
16 auth = firebase.auth()
17 user = auth.sign_in_with_email_and_password([EMAIL_USERNAME], [EMAIL_PASSWORD])
18 db = firebase.database()
19
20 while True:
21     try:
22         # adjust the sleep time if you have successfully push data to your firebase
23         sleep(5)
24         [temp, hum] = dht(dhtsensor, 0)
25         print("Temp = ", temp, '\u00b0C', " Hum = ", hum, "%")
26         t = str(temp)
27         h = str(hum)
28         setRGB(0, 255, 0)
29         setText("Temp = " + t + "\337" + "C    Hum = " + h + "%")
30         data1 = {"temperature":t}
31         data2 = {"humidity":h}
32         results = db.child("PI_001").push(data1, user['idToken'])
33         results = db.child("PI_001").push(data2, user['idToken'])
34         # remove the break if you have successfully push data to your firebase
35         break
36     except KeyboardInterrupt:
37         setText("Program Exited")
38         break
39     except TypeError:
40         print("Type Error occurs")
41     except IOError:
42         print("IO Error occurs")
43
```

- Secret code for reading temperature. Change the temp and hum variable to machine readable text.
- Symbol for certain character. Convert temp and hum variable to string data type.
- "Celcius" symbol indicator. Casting for temp and hum variable to integer data type.

Feedback

Yes. \u00b0 and \337 are Unicode and Octal value (LCD character table) for (Degree, °) symbol. str(xxx) is to convert the object xxx to string data type.

```
test06.py 3C
1 from time import *
2 from grovepi import *
3 from grove_rgb_lcd import *
4 from pyrebase import pyrebase
5
6 dhtsensor = 7
7 pinMode(dhtsensor, "INPUT")
8
9 config = {
10     "apiKey": "[API_KEY]",
11     "authDomain": "[PROJECT_ID].firebaseapp.com",
12     "databaseURL": "https://[DATABASE_NAME].firebaseio.com",
13     "storageBucket": "[PROJECT_ID].appspot.com"
14 }
15 firebase = pyrebase.initialize_app(config)
16 auth = firebase.auth()
17 user = auth.sign_in_with_email_and_password([EMAIL_USERNAME], [EMAIL_PASSWORD])
18 db = firebase.database()
19
20 while True:
21     try:
22         # adjust the sleep time if you have successfully push data to your firebase
23         sleep(5)
24         [temp, hum] = dht(dhtsensor, 0)
25         print("Temp = ", temp, '\u00b0C', " Hum = ", hum, "%")
26         t = str(temp)
27         h = str(hum)
28         setRGB(0, 255, 0)
29         setText("Temp = " + t + '\u00b0C' + "Hum = " + h + "%")
30         data1 = {"temperature":t}
31         data2 = {"humidity":h}
32         results = db.child("PI_001").push(data1, user['idToken'])
33         results = db.child("PI_001").push(data2, user['idToken'])
34         # remove the break if you have successfully push data to your firebase
35         break
36     except KeyboardInterrupt:
37         setText("Program Exited")
38         break
39     except TypeError:
40         print("Type Error occurs")
41     except IOError:
42         print("IO Error occurs")
43
```

- Program will pause until user press "space bar" to continue the program flow.
- Program will exit the while loop
- Program will break (branch out) and perform exception case (run the code in except KeyboardInterrupt)
- Program will pause and continue with setText("Program Exited")

Feedback

I'm sure you listen to the practical classes. Yes, we don't need this line if we have confirmed our coding "push" data properly to the firebase, to ensure it runs continuously.

```

test06.py ✎
1  from time import *
2  from grovepi import *
3  from grove_rgb_lcd import *
4  from pyrebase import pyrebase
5
6  dhtsensor = 7
7  pinMode(dhtsensor, "INPUT")
8
9  config = {
10    "apiKey": "[API_KEY]",
11    "authDomain": "[PROJECT_ID].firebaseapp.com",
12    "databaseURL": "https://[DATABASE_NAME].firebaseio.com",
13    "storageBucket": "[PROJECT_ID].appspot.com"
14  }
15  firebase = pyrebase.initialize_app(config)
16  auth = firebase.auth()
17  user = auth.sign_in_with_email_and_password([EMAIL_USERNAME], [EMAIL_PASSWORD])
18  db = firebase.database()
19
20 while True:
21   try:
22     # adjust the sleep time if you have successfully push data to your firebase
23     sleep(5)
24     [temp, hum] = dht(dhtsensor, 0)
25     print("Temp = ", temp, '\u00b0C', " Hum = ", hum, "%")
26     t = str(temp)
27     h = str(hum)
28     setRGB(0, 255, 0)
29     setText("Temp = " + t + '\u00b0C' + "Hum = " + h + "%")
30     data1 = {"temperature":t}
31     data2 = {"humidity":h}
32     results = db.child("PI_001").push(data1, user['idToken'])
33     results = db.child("PI_001").push(data2, user['idToken'])
34     # remove the break if you have successfully push data to your firebase
35     break
36   except KeyboardInterrupt:
37     setText("Program Exited")
38     break
39   except TypeError:
40     print("Type Error occurs")
41   except IOError:
42     print("IO Error occurs")
43

```

- Set up the connection between python code and Google Firebase.
- Authentication for any user to access the Google Firebase.
- Application layer for updating or retrieving data to / from the Google Firebase.
- Retrieve Google services like Google Maps, Gmail, etc.
- Access to Google Drive for updating and retrieving sensors data.
- Access to TAR UC servers for updating and retrieving sensors data.

Week 6 Working with Thinkspeaks & Beebotte

PART 1: Connecting to the cloud (internet) - thinkspeaks.com

Step 1 : Setup your thingspeak by following these steps in the pdf ([Setup - Thingspeak.pdf](#))

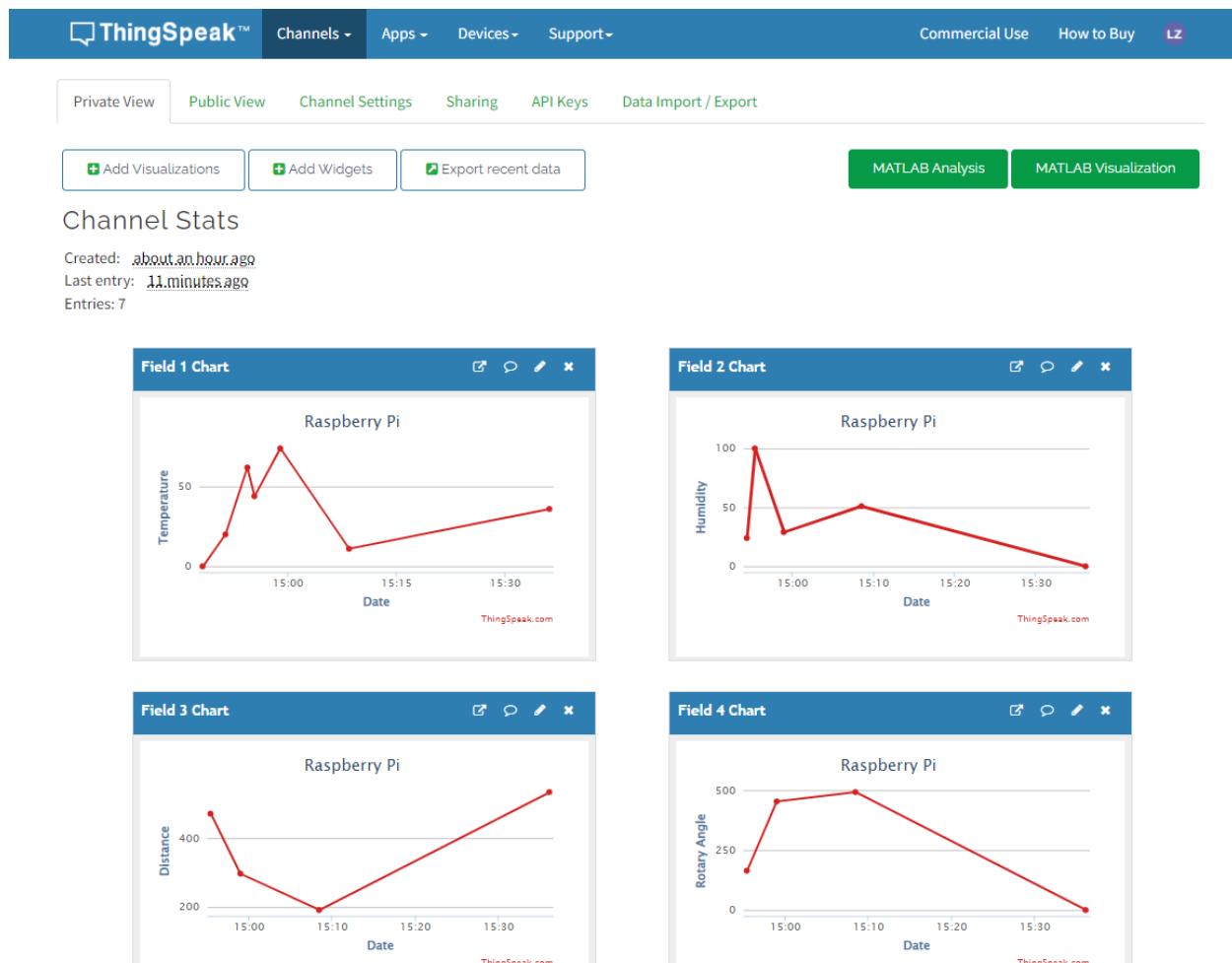
Step 2 : Create new python file called *test07.py* & *test07_launcher.py* and retype the codes given below

- [test07.py](#)
- [test07_launcher.py](#)

*Note that you have to use your **own Write API key** in order to push the data onto your thingspeak successfully.

Step 3 : Run the code in *test07_launcher.py*

Step 4 : You can view the data if you have successfully push data to your thingspeak



Example the data that have been successfully push onto thingspeak

PART 2: Connecting to the cloud (internet) - Beebotte.com

Step 1 : Setup your thingspeak by following these steps in the pdf ([Setup - Beebotte.pdf](#))

Step 2 : Create new python file called *test08.py* & *test08_launcher.py* and retype the codes given below

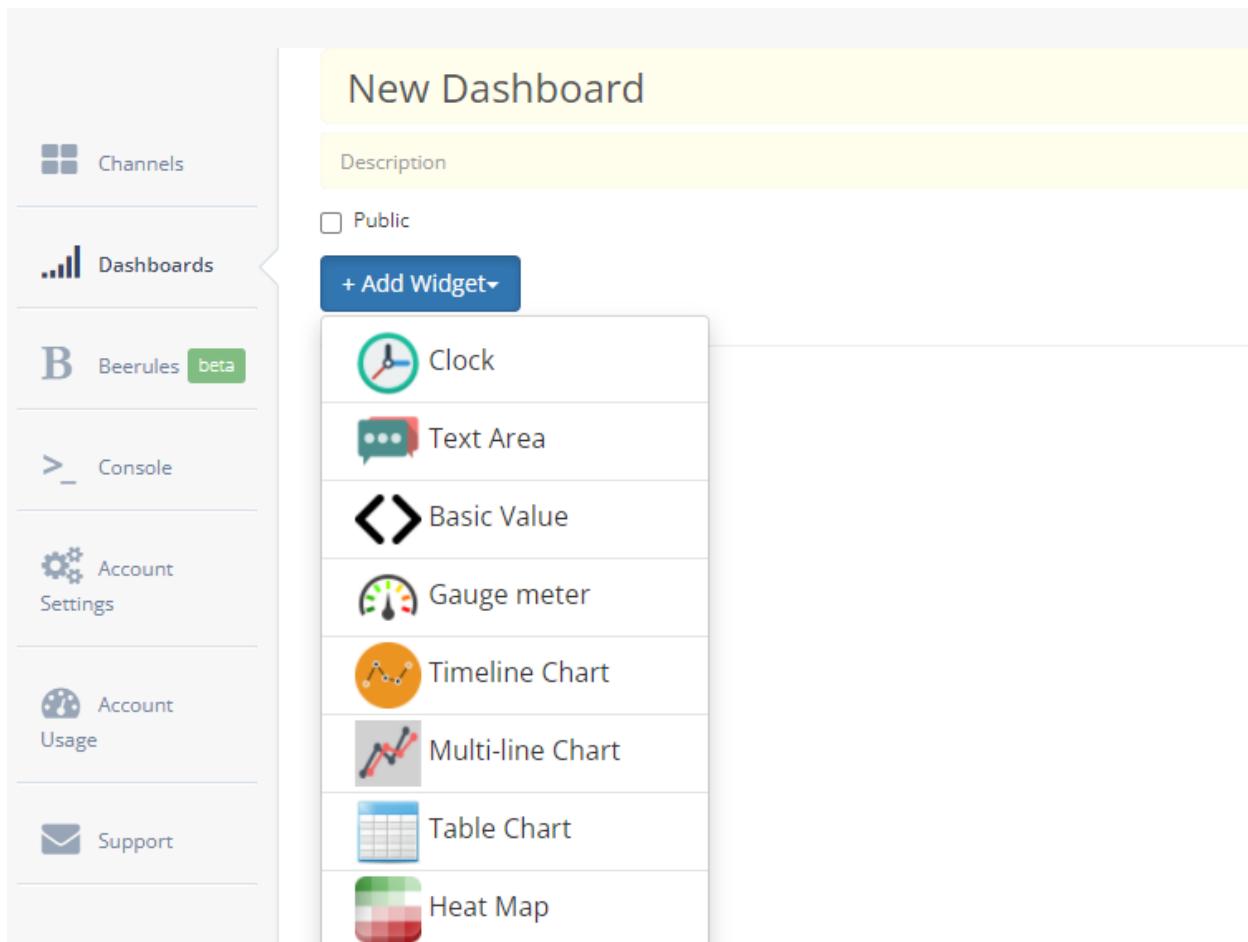
- [test08.py](#)
- [test08_launcher.py](#)

*Note that you have to use your **own API key and secret key** in order to push the data onto your thingspeak successfully.

Step 3 : Run the code in *test08_launcher.py*

Step 4 : You can view the data if you have successfully push data to your beebotte

Before you can view the data, you have to enter beebotte website and click on the [Dashboard](#) > [Create Dashboard](#) > [Add Widget](#)



Then, select the widget and resource you want to be displayed and click Done.

Practical 6

Temperature and Humidity

 zixuan  Private  Created: February 25th 2022  Views: 4

Temperature		
110		
30 minutes ago		

Temperature		
RESOURCE	DATA	WHEN
bait2123.temperature	110	30 minutes ago
bait2123.temperature	17	30 minutes ago
bait2123.temperature	110	34 minutes ago
bait2123.temperature	9	34 minutes ago
bait2123.temperature	25	34 minutes ago
bait2123.temperature	54	38 minutes ago
bait2123.temperature	1	40 minutes ago
bait2123.temperature	8	41 minutes ago

Showing 1 to 8 of 8 entries

Previous  Next

Temperature	
150	Period: Latest

Humidity	
50	30 minutes ago

Example the widget data that have been successfully push onto beebotte

If you exit the dashboard accidentally, you can click on [Dashboard](#) at the sidebar > [title](#) (i.e *Practical 6* in my case) to view back your widget data.

Channels

Dashboards

Create and manage your dashboards

Beerules 

[Create Dashboard](#)

Search:

TITLE	DESCRIPTION	CREATED ON	SCOPE	VIEWS
Practical 6	Temperature and Humidity	February 25th 2022	Private	4

Showing 1 to 1 of 1 entries

Previous  Next

Practical 6 Quiz

In Beebotte cloud service, updated data are show in the cloud database. 4/4

Which coding line act as authenticating connection to beebotte cloud service and updating the code respectively ? *

```
test08.py * %
1 from time import *
2 from grovepi import *
3 from grove_rgb_lcd import *
4 from random import *
5 from urllib.request import *
6 from beebotte import *
7
8 led = 3
9 soundsensor = 14
10 lightsensor = 15
11 dhtsensor = 16
12 pinMode(led, "OUTPUT")
13 pinMode(soundsensor, "INPUT")
14 pinMode(lightsensor, "INPUT")
15 pinMode(dhtsensor, "INPUT")
16 apikey = [YOUR API KEY]
17 secretkey = [YOUR SECRET KEY]
18 bclient = BBT(apikey, secretkey)
19
20 while True:
21     try:
22         # adjust the sleep time if you have successfully push data to your beebotte
23         sleep(5)
24         [temp, hum] = dht(dhtsensor, 0)
25         light = analogRead(lightsensor)
26         sound = analogRead(soundsensor)
27         print("Temp = %.2f, hum = %d, light = %d, sound = %d" %(temp, hum, light, sound))
28         setText("Temp = " + t + '\337' + "C Hum = " + h + "%")
29         bclient.write('test', 'temperature', temp)
30         bclient.write('test', 'humidity', hum)
31         # remove the break if you have successfully push data to your beebotte
32         break
33     except KeyboardInterrupt:
34         setText("Program Exited")
35         break
36     except TypeError:
37         print("Type Error occurs")
38     except IOError:
39         print("IO Error occurs")
40
```

- Line 16 and 17; Line 29 and 30
- Line 18; Line 28
- Line 18; Line 29 and 30
- Line 6 ; Line 29 and 30
- Line 6 and 16; Line 28

```
test07.py
1 from time import *
2 from grovepi import *
3 from grove_rgb_lcd import *
4 from random import *
5 from urllib.request import *
6
7 led = 3
8 soundsensor = 14
9 lightsensor = 15
10 dhtsensor = 16
11 pinMode(led, "OUTPUT")
12 pinMode(soundsensor, "INPUT")
13 pinMode(lightsensor, "INPUT")
14 pinMode(dhtsensor, "INPUT")
15 apikey = [YOUR_API_KEY]
16
17 while True:
18     try:
19         # adjust the sleep time if you have successfully push data to your thingspeak
20         sleep(5)
21         [temp, hum] = dht(dhtsensor, 0)
22         light = analogRead(lightsensor)
23         sound = analogRead(soundsensor)
24         print("Temp = %.2f, hum = %d, light = %d, sound = %d" %(temp, hum, light, sound))
25         t = str(temp)
26         h = str(hum)
27         setText("Temp = " + t + '\337' + "C    Hum = " + h + "%")
28         r = randint(0, 255)
29         g = randint(0, 255)
30         b = randint(0, 255)
31         setRGB(r, g, b)
32         content = urlopen("https://api.thingspeak.com/update?api_key=" + apikey + "&field1=" + t).read()
33         if (content):
34             print("Updated Thingspeak")
35             # remove the break if you have successfully push data to your thingspeak
36             break
37     except KeyboardInterrupt:
38         setText("Program Exited")
39         break
40     except TypeError:
41         print("Type Error occurs")
42     except IOError:
43         print("IO Error occurs")
```

- light sensor gives us a value of 0 and 1
- sound sensor gives us a value more than two levels
- digitalRead will not be able to read the value from light and sound sensor.
- digitalRead will give us an inappropriate value from light and sound sensor.
- analogRead will give us a value from 0 to 255
- analogRead will give us a value from 0 to 1023
- analogRead will give us a value from 0 to 2047

To update the cloud storage / service with multiple type of sensor data at the same time, we can ... *

```
test07.pyX
1 from time import *
2 from grovepi import *
3 from grove_rgb_lcd import *
4 from random import *
5 from urllib.request import *
6
7 led = 3
8 soundsensor = 14
9 lightsensor = 15
10 dhtsensor = 16
11 pinMode(led, "OUTPUT")
12 pinMode(soundsensor, "INPUT")
13 pinMode(lightsensor, "INPUT")
14 pinMode(dhtsensor, "INPUT")
15 apikey = [YOUR_API_KEY]
16
17 while True:
18     try:
19         # adjust the sleep time if you have successfully push data to your thingspeak
20         sleep(5)
21         [temp, hum] = dht(dhtsensor, 0)
22         light = analogRead(lightsensor)
23         sound = analogRead(soundsensor)
24         print("Temp = %.2f, hum = %d, light = %d, sound = %d" %(temp, hum, light, sound))
25         t = str(temp)
26         h = str(hum)
27         setText("Temp = " + t + '\337' + "C    Hum = " + h + "%")
28         r = randint(0, 255)
29         g = randint(0, 255)
30         b = randint(0, 255)
31         setRGB(r, g, b)
32         content = urlopen("https://api.thingspeak.com/update?api_key=" + apikey + "&field1=" + t).read()
33         if (content):
34             print("Updated Thingspeak")
35             # remove the break if you have successfully push data to your thingspeak
36             break
37     except KeyboardInterrupt:
38         setText("Program Exited")
39         break
40     except TypeError:
41         print("Type Error occurs")
42     except IOError:
43         print("IO Error occurs")
44
```

- Modify the url string at line 32
- Add lines in between line 32 and 33 to call more urlopen functions for updating multiple data.
- Add more if else condition statement to check multiple data updates at line 33
- Remove break at line 36 to allow multiple updates on sensor data.

Feedback

Yes. We can add on the URL string by putting more parsing parameters and value (e.g., ...&field2=data2&field3=data3...)

Week 7 Mosquitto MQTT on Windows 10

Notes from Dr Poh Tze Ven

For practical 7, the MQTT on Windows 10 document ([document](#)) shows how to *install* and *configure* Mosquitto MQTT on Windows 10. It is complicated because Windows 10 enforces IP access security by default, so we have to disarm the security on Mosquitto MQTT. Otherwise, any client trying to access the broker through TCP/IP will be blocked by the Windows OS. Note that if the broker is not properly set up, the MQTT client Python program (using the Paho library) will not work. To run the program given in Practical 7, you need to install the Paho library:

```
pip3 install paho-mqtt
```

If you want to know the *return code* by the MQTT client, it can be found in the last page of this [document](#). Please watch the [video](#) (Practical 7: Part 2) on why you might need to know the *return code* by the MQTT. In the video, I encountered a connection issue with one of the MQTT brokers on the Internet because the network was temporarily down.

There is a new section in Practical 7 regarding **NodeRED**. Please try that on your own. At the moment, I don't have a video covering that topic.

For those who are trying the **MQTT on the Raspberry Pi OS running on the real Raspberry Pi** (or under a VirtualBox in Windows), you can change the MQTT broker's configuration by editing the `mosquitto.conf` file. For example you might want to change the MQTT broker to listen on a specific IP address and port of your Raspberry Pi, do the following:

```
$ sudo thonny /etc/mosquitto/mosquitto.conf &
```

Add the following lines and save:

```
listener 1883 [YOUR_IP_ADDRESS]
listener 1883 127.0.0.1
allow_anonymous true #This line is needed for "Bullseye" OS to work correctly
```

Restart the MQTT broker:

```
$ sudo systemctl stop mosquitto
$ sudo systemctl start mosquitto
```

Or simply just:

```
$ sudo systemctl restart mosquitto
```

On the "Bullseye" version of Raspberry Pi OS, you will need the `allow_anonymous true` command (see above). Otherwise, Connection Refused, not authorized error (error code 5) will be returned.

Self-note-taking for Practical 7

Part 1 Test the MQTT process using cmd

First, you have to follow this [document](#) to set up a Mosquitto MQTT on Windows 10 .

- Note that I was currently learning from home in which I don't have access to the Raspberry Pi to develop the MQTT program, hence I'm using this [document written by Dr Poh Tze Ven](#) to set up a Mosquitto MQTT in my laptop.
- If you follow the [document](#) properly, it will be no issue for you to demonstrate the MQTT process by setting up a Mosquitto MQTT and demonstrating it solely using cmd.

Special Notes for above section

If you are still facing this error even after extending MQTT service to accept external requests -

Error: No connection could be made because the target machine actively refused it, it might be you accidentally closed the broker cmd [1]. Note that you cannot close this cmd as it acts as a BROKER that enables MQTT clients to communicate.

There are 3 command prompts needed in order to make the MQTT process successful.

[1] **1st cmd** - Start the MQTT broker service using the configurations in the mosquito.conf file

mosquitto -v -c mosquito.conf

```
C:\Windows\System32\cmd.exe - mosquitto -v -c mosquito.conf
```

```
C:\Program Files\mosquitto>
C:\Program Files\mosquitto>
C:\Program Files\mosquitto>mosquitto -v -c mosquito.conf
1646455597: mosquitto version 2.0.14 starting
1646455597: Config loaded from mosquito.conf.
1646455597: Opening ipv4 listen socket on port 1883.
1646455597: mosquitto version 2.0.14 running
```

2nd cmd - Act as a Subscriber

mosquitto_sub -h 192.168.1.37 -t sensor/temp/device1 -d

```
C:\Windows\System32\cmd.exe - mosquitto_sub -h 192.168.1.37 -t sensor/temp/device1 -d
```

```
Microsoft Windows [Version 10.0.19044.1526]
(c) Microsoft Corporation. All rights reserved.
```

```
C:\Program Files\mosquitto>
C:\Program Files\mosquitto>mosquitto_sub -h 192.168.1.37 -t sensor/temp/device1 -d
Client null sending CONNECT
Client null received CONNACK (0)
Client null sending SUBSCRIBE (Mid: 1, Topic: sensor/temp/device1, QoS: 0, Options: 0x00)
Client null received SUBACK
Subscribed (mid: 1): 0
Client null received PUBLISH (d0, q0, r0, m0, 'sensor/temp/device1', ... (6 bytes))
hello!
```

3rd cmd - Act as a Publisher

```
mosquitto_pub -h 192.168.1.37 -t sensor/temp/device1 -m "hello!" -d
```

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19044.1526]
(c) Microsoft Corporation. All rights reserved.

C:\Program Files\mosquitto>mosquitto_pub -h 192.168.1.37 -t sensor/temp/device1 -m "hello!" -d
Client null sending CONNECT
Client null received CONNACK (0)
Client null sending PUBLISH (d0, q0, r0, m1, 'sensor/temp/device1', ... (6 bytes))
Client null sending DISCONNECT

C:\Program Files\mosquitto>
```

Part 2 Test the MQTT process using Thonny Python (ID)

Step 1

In **Thonny Python (ID)**, click “New” to create a new python file and Save As “test09.py”. Type the following codes for subscriber([test10.py](#) & [test10_launcher.py](#)) and publisher([test09.py](#)).

Step 2

As Thonny Python (ID) restricts only one program running at one time, in this case, we will use **Tools > Open System Shell (1)** and **Run current script (2)** in Thonny Python to demonstrate the MQTT process.

(1)

Issue “python test10_launcher.py” to run this program - it will act as subscriber

```
C:\WINDOWS\system32\cmd.exe - python test10_launcher.py
*****
Some Python commands in the PATH of this session:
- python    == C:\Users\zixua\AppData\Local\Programs\Thonny\python.exe
- pip        == C:\Users\zixua\AppData\Local\Programs\Thonny\Scripts\pip.bat
- pip3      == C:\Users\zixua\AppData\Local\Programs\Thonny\Scripts\pip3.bat
- pip3.7    == C:\Users\zixua\AppData\Local\Programs\Thonny\Scripts\pip3.7.bat
*****
C:\Users\zixua\OneDrive\Desktop\RSD - Y2S3\BAIT2123 Internet Of Things\Practical\Practical 7>python test10_launcher.py
**config: pin 3 has been set to OUTPUT**
Connected with result code 0
```

(2)

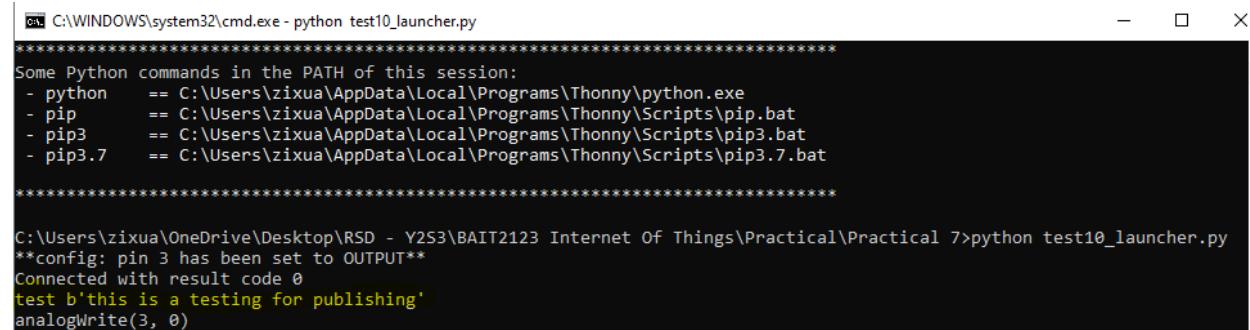
Run test09.py by clicking [Run current script](#) or [F5](#). - it will act as publisher



A screenshot of the Thonny Python IDE interface. The title bar shows 'Run current script (F5)'. The code editor contains the following Python script:

```
1 from time import *
2 from grovepi import *
3 from paho.mqtt import publish
4
5 MQTT_BROKER = "192.168.1.37"
6 #MQTT_BROKER = "broker.emqx.io" #using public mqtt broker to act as publisher
7 MQTT_TOPIC = "test"
8
9 publish.single(MQTT_TOPIC, "this is a testing for publishing", hostname=MQTT_BROKER)
```

Next, you can see the text “this is a testing for publishing” appear on the system shell (1) that acts as the subscriber.



A screenshot of a Windows Command Prompt window titled 'C:\WINDOWS\system32\cmd.exe - python test10_launcher.py'. The output shows the following text:

```
*****
Some Python commands in the PATH of this session:
- python == C:\Users\zixua\AppData\Local\Programs\Thonny\python.exe
- pip == C:\Users\zixua\AppData\Local\Programs\Thonny\Scripts\pip.bat
- pip3 == C:\Users\zixua\AppData\Local\Programs\Thonny\Scripts\pip3.bat
- pip3.7 == C:\Users\zixua\AppData\Local\Programs\Thonny\Scripts\pip3.7.bat
*****
C:\Users\zixua\OneDrive\Desktop\RSD - Y2S3\BAIT2123 Internet Of Things\Practical\Practical 7>python test10_launcher.py
**config: pin 3 has been set to OUTPUT**
Connected with result code 0
test b'this is a testing for publishing'
analogWrite(3, 0)
```

A simple MQTT process demonstrated.

Extra

Except using your [own IP address], you can also modify the value of the MQTT_BROKER variable (i.e `MQTT_BROKER = "broker.emqx.io"`) to one of the public MQTT brokers as what I comment in the `test09.py` and `test10.py`.

One of the example for public MQTT broker: [Free Public MQTT 5 Broker Server | EMQ](#)

∴ You can search more public MQTT Broker via online

Testing MQTT Service (with Username and Password)

After setting up username and password (last part of the [document](#)),

Save it and restart the mosquitto MQTT broker using the following command (Note: Ensure the previous running MQTT broker has been terminated first):

```
mosquitto -v -c mosquitto.conf
```

```
C:\Windows\System32\cmd.exe - mosquitto -v -c mosquitto.conf
Microsoft Windows [Version 10.0.19044.1526]
(c) Microsoft Corporation. All rights reserved.

C:\Program Files\mosquitto>mosquitto -v -c mosquitto.conf
1646458593: mosquitto version 2.0.14 starting
1646458593: Config loaded from mosquitto.conf.
1646458593: Opening ipv4 listen socket on port 1883.
1646458593: mosquitto version 2.0.14 running
```

For subscriber,

```
mosquitto_sub -h [YOUR_IP_ADDRESS] -t [TOPIC] -u [USERNAME] -P [PASSWORD]
```

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19044.1526]
(c) Microsoft Corporation. All rights reserved.

C:\Program Files\mosquitto>mosquitto_pub -h 192.168.1.37 -t test -m testing123 -u zixuan -P zixuan2001711
```

For publisher,

```
mosquitto_pub -h [YOUR_IP_ADDRESS] -t [TOPIC] -m [DATA] -u [USERNAME] -P [PASSWORD]
```

```
C:\Windows\System32\cmd.exe - mosquitto_sub -h 192.168.1.37 -t test -u zixuan -P zixuan2001711
Microsoft Windows [Version 10.0.19044.1526]
(c) Microsoft Corporation. All rights reserved.

C:\Program Files\mosquitto>mosquitto_sub -h 192.168.1.37 -t test -u zixuan -P zixuan2001711
testing123
```

If either or both username and password is/are wrong or were not given, the broker will refuse connection.

Part 3

Task 1 - Modify the code to cross control other sensors / devices with another raspberry pi.

I had modified the code to cross control the buzzer to on or off by using digitalWrite(buzzer,0) / digitalWrite(buzzer,1).

Code can be found on my Google Drive:

- [test09_task1.py](#)
- [test10_task1.py](#)
- [test10_task1_launcher.py](#)

Practical 7

In MQTT Protocol, the important element to distinguish between clients who send a message to who act on the message sent by the client, is ... * 4/4

- Client IP address
- Message length
- Topic
- Client networking port

Feedback

We define topic to be send by publisher and received by subscriber.

In MQTT protocol, Broker serves two type of clients, which are: *

4/4

```
test11.py*  
1 from time import *  
2 from grovepi import *  
3 from paho.mqtt.client import *  
4  
5 buzzer = 6  
6 pinMode(buzzer, "OUTPUT")  
7  
8 MQTT_SERVER = "localhost"  
9 MQTT_PATH = "buzzer"  
10  
11 def on_connect(client, userdata, flags, rc):  
12     print("Connected with result code " + str(rc))  
13     client.subscribe(MQTT_PATH)  
14  
15 def on_message(client, userdata, msg):  
16     print(msg.topic + " " + str(msg.payload))  
17     i = int(msg.payload)  
18     if i > 0 & i < 256:  
19         analogWrite(buzzer,i)  
20     else:  
21         analogWrite(buzzer,0)  
22  
23 client = Client()  
24 client.on_connect = on_connect  
25 client.on_message = on_message  
26  
27 client.connect(MQTT_SERVER, 1883, 60)  
28  
29 client.loop_forever()  
  
from time import *  
from grovepi import *  
from paho.mqtt import publish  
  
MQTT_BROKER = "[PUT YOUR PI IP ADDRESS HERE]" # Change later  
MQTT_TOPIC = "buzzer"  
  
def on_connect(client, userdata, flags, rc):  
    print("Connected with result code " + str(rc))  
    client.subscribe(MQTT_PATH)  
  
def on_message(client, userdata, msg):  
    print(msg.topic + " " + str(msg.payload))  
  
publish.single(MQTT_TOPIC, "120", hostname=MQTT_BROKER)
```

- Sender and Receiver
- Publisher and Subscriber
- Transmitter and Receiver
- Server and Client

In this MQTT communication test code, the left hand side serves as receiver 4/4 to perform action, and the right hand side serves as sender to give commands. What is the different between MQTT_SERVER and MQTT_BROKER ? *

```
test11.pyx
1 from time import *
2 from grovepi import *
3 from paho.mqtt.client import *
4
5 buzzer = 6
6 pinMode(buzzer, "OUTPUT")
7
8 MQTT_SERVER = "localhost"
9 MQTT_PATH = "buzzer"
10
11 def on_connect(client, userdata, flags, rc):
12     print("Connected with result code " + str(rc))
13     client.subscribe(MQTT_PATH)
14
15 def on_message(client, userdata, msg):
16     print(msg.topic + " " + str(msg.payload))
17     i = int(msg.payload)
18     if i > 0 & i < 256:
19         analogWrite(buzzer,i)
20     else:
21         analogWrite(buzzer,0)
22
23 client = Client()
24 client.on_connect = on_connect
25 client.on_message = on_message
26
27 client.connect(MQTT_SERVER, 1883, 60)
28
29 client.loop_forever()
```



```
from time import *
from grovepi import *
from paho.mqtt import publish
MQTT_BROKER = "[PUT YOUR PI IP ADDRESS HERE]" # Change later
MQTT_TOPIC = "buzzer"
def on_connect(client, userdata, flags, rc):
    print("Connected with result code " + str(rc))
    client.subscribe(MQTT_PATH)
def on_message(client, userdata, msg):
    print(msg.topic + " " + str(msg.payload))
publish.single(MQTT_TOPIC, "120", hostname=MQTT_BROKER)
```

- No difference.
- MQTT_BROKER is for our own IP address; MQTT_SERVER is for a server IP address.
- MQTT_BROKER is for broker IP address; MQTT_SERVER is for a server IP address.
- MQTT_BROKER is for our own IP address; MQTT_SERVER is for localhost IP address.

Practical 8

Get the code from my Github - <https://github.com/loozixuan/Degree-RSD-Materials>

In the practical, we use python coding and include luma-oled library with SH1106 driver to communicate with Grove - OLED Display 1.12" module. Select the following parameters that required for establishing this communication: *

- Using Serial Peripheral Interface protocol
- Using Universal Asynchronous Receiver / Transmitter protocol
- Using Inter-Integrated Circuit protocol
- With module address of 0x04
- With module address of 0x3C
- With module address of 0x01

OLED stands for *

1/1

- Organic Light-Emitting Diode
- Original Light-Emitting Diode
- Original Liquid Crystal Display
- Operational Light-Emitting Diode

By using Python Image Library (PIL), we can resize an image with the following 2/2 codes / functions (assume that there is a "testing.jpg" file in "C:\". *** Only CHECK the necessary lines, without these CHECKED lines, program will prompt error(s). *** *

- import matplotlib.image as mpimg
- from PIL import Image
- img = Image.open(r"C:\testing.jpg")
- print(img.format)
- newsize=(64, 64)
- print(newsize)
- img = img.resize(newsize)
- img = img.save(r"C:\testing_resized.jpg")

The following(s) are CORRECT description for Grove - Passive InfraRed (PIR) motion sensor usage: *

- The motion detector is triggered and the sensor outputs as HIGH on its SIG (signal) pin.
- The motion detector is triggered and the sensor outputs as LOW on its SIG (signal) pin.
- The response speed is from 0.3s - 25s, and max 6 meters of detecting range.
- Read the motion sensor value by checking i = grovepi.digitalWrite(pir_sensor, 1)
- Read the motion sensor value by checking i = grovepi.analogWrite(pir_sensor, 255)
- Read the motion sensor value by checking i = grovepi.digitalRead(pir_sensor)
- If i = 128, motion is triggered
- If i = 0, no motion is triggered
- If i = 255, no motion is triggered

Practical 9

Get the code from my Github - <https://github.com/loozixuan/Degree-RSD-Materials>

RFID stands for *

1/1

- Radio Frequency IDentification
- Response Fault IDentifier
- Reduced Fraud IDentification
- Randomise Frequency IDentification
- Radio Fidelity IDentification

The following are essential elements for a complete and function-able RFID system: *

4/4

- A screen protector, to protect the RFID card surface from scratch
- A card holder, that can hold the card when it is read by embedded system
- A reader, that is connected to (or integrated with) embedded system
- An antenna, that sends out / receives in a radio signal
- A printer, that can print a label on the RFID card surface.
- A tag (or transponder) that returns the signal with information added
- An algorithm that is coded / programmed in an embedded system for specific features.
- A ring, to hold / tagged the RFID card with a keychain (with your name)

Which of the following statement is CORRECT ? *

3/3

- Our Malaysia Identity Card RFID feature is running on 13.56MHz frequency.
- Our Bank credit card RFID feature is running on 125kHz frequency.
- RFID Scanner (13.56MHz) requires Grove Pi module
- RFID Scanner (125kHz) requires Grove Pi module
- RFID that run under 125kHz transmit only 14 bytes of data.
- RFID that run under 13.56MHz transmit more than 14 bytes of data.
- RFID that run under 125kHz is more secured than 13.56MHz.
- RFID that run under 13.56MHz is less secured than 125kHz.

RFID with 13.56MHz Mifare

- Have higher frequency
- Include smart processor
- Offer more security
- Can read and write data to the device

RFID with 125KHz

- Read-only technology with little or no encryption
- Less security

Practical 10

By using Raspberry Pi, in order to ensure the Camera device (either USB or Pi 0/4 Camera) function-able, we MUST do take the following step(s) *

-  Ensure the camera device is properly connected to Pi.
-  Ensure the audio jack is connected properly.
-  Ensure the Pi casing is closed.
-  picamera library is installed for accessing USB webcam in Python programming
-  fswebcam library is installed for accessing Pi camera in Python programming
-  Turn on Raspberry Pi power.

Feedback

Check the correct library for correct camera type, casing and audio jack does not reflecting the functionality of camera.

⚠ The circle in green color should be the correct answer.

To stream a video camera, these elements are important, EXCEPT: *

2/2

- Network connectivity
- IP address of sender
- IP address of receiver
- Port for video stream at sender's IP

We can stream the input video to multiple viewers via network using http (web browser). *

2/2

- True
- False