Practical 3

PART 1: Getting Know the the GrovePi Starter Kit

The starter kit bundles the most popular sensors for education and hobbyists, and lets you start playing and prototyping hardware with Raspberry Pi. The GrovePi Starter Kit package includes:

- GrovePi+ Board
- Grove cables for connecting the sensors and modules to the GrovePi board.
- Grove sensors and modules (Please request from lab assistant when it is required)

Sensors and modules include


Grove pi+ × 1


Grove - sound Sensor × 1


Grove - Temperature&Humidity × 1


Grove - light sensor × 1


Grove -Relay × 1


Grove - Button × 1


Grove - Ultrasonic Ranger × 1


Grove - Rotary Angle Sensor × 1


Printing limited text - for diagnostic and message displaying purpose
Grove-LCD RGB Backlight × 1


Grove - red led × 1


Grove Buzzer × 1


Grove blue led × 1


Detect the rain drop
Water (raindrop) sensor x 1


Grove green led × 1


Press into the soil to detect soil moisture
Moisture sensor x 1


Passive InfraRed (PIR) sensor x 1

Detect motion - Anything generates heat will be detected


Touch sensor x 1


Radio Frequency IDentification (RFID) Reader (125kHz) x 1


OLED display x 1

and more… please consult lab assistant for sensor's availability

Getting started with the GrovePi for the first time is easy:

- Check out our Quickstart guide to GrovePi here.
- We recently published the Starter Kit guide for the GrovePi+ in collaboration with Seeedstudio. You can download it from here:

Connected GrovePi on the Raspberry Pi

First, identify the mounted GrovePi (blue board) on the Raspberry Pi (green board). The GrovePi has a black plastic piece on the bottom, which fits perfectly with the metal pins sticking out of the Raspberry Pi. Slide the GrovePi board onto the pins on the Raspberry Pi as shown in the pictures below. The GrovePi fits both the Raspberry Pi A, B, B+, and Raspberry Pi 2.

Ensure that the pins are properly aligned when stacking the GrovePi, and pushed down until they go in all the way.

INCORRECT installation will cause the Raspberry Pi / GrovePi spoil.

Step 4: Connected GrovePi on the Raspberry Pi

Step 5: In a terminal, run Update Grove Pi library

```
curl -kL dexterindustries.com/update_grovepi | bash
```

* If GrovePi software has not been installed.

PART 2: Updating the Grove Pi Firmware (Flashing the memory in GrovePi board)
*** If the GrovePi is functioning normal, please skip this PART 2.**

1. Run a Firmware update by using the following command: Notes: We will use bash to execute firmware update script

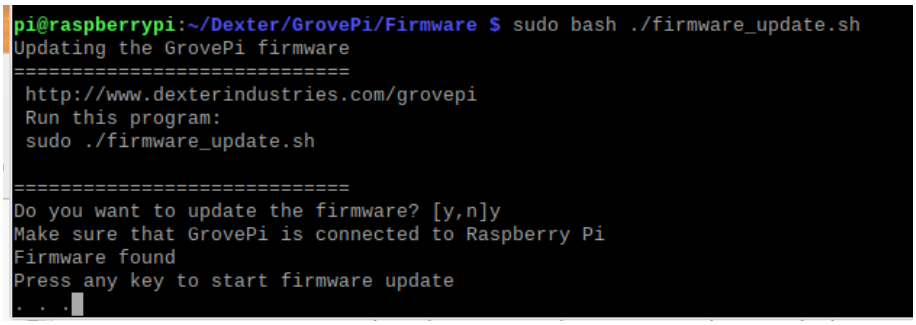
```
cd /home/pi/Dexter/GrovePi/Firmware/
sudo bash ./firmware_update.sh
```

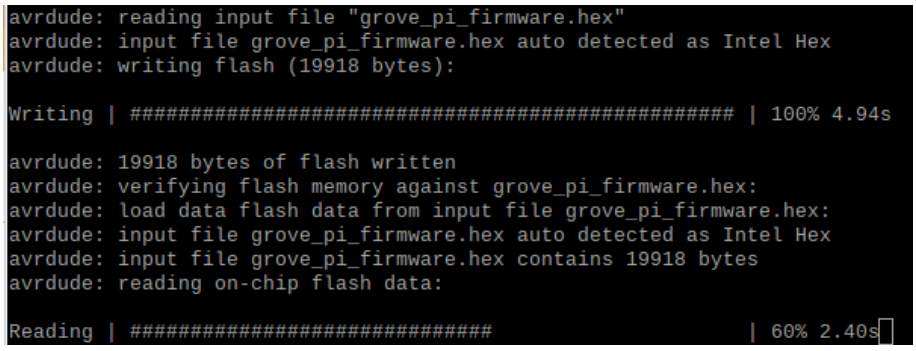
Press "Y" for yes, followed by any key to start the firmware update.

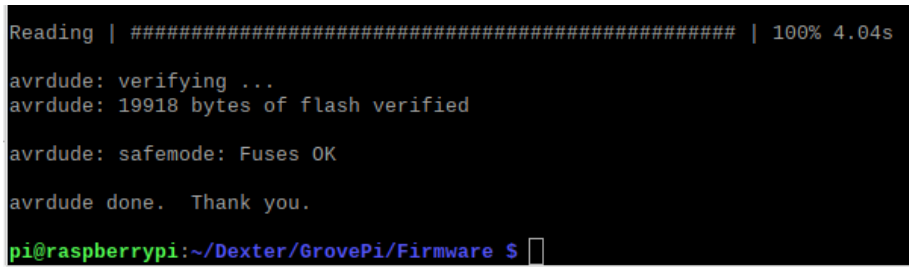
Understand what is the meaning of these commands:

cd stands for "change directory".
sudo stands for "superuser do".
bash to command the computer (cpu) to action, i.e., "run". bash is a command-line interpreter for Unix OS

In Linux platform, directory separator is defined as **"/"."."/"** is the current folder.
. means "current", / means "folder"



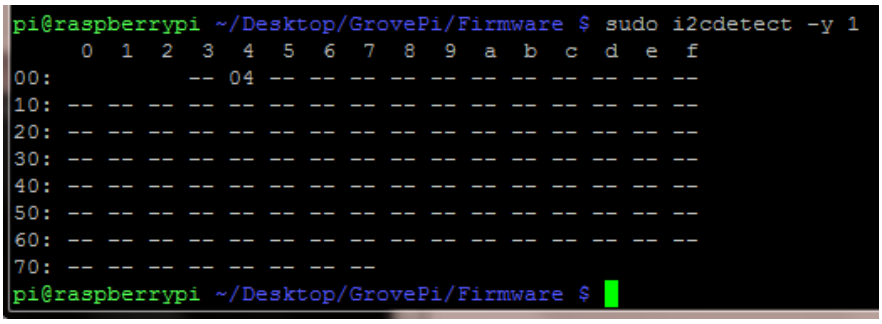




[Ignore this step 2 if you are using the latest updated Raspberry Pi]

2. Now to check that the script was correctly installed. We will check that the Raspberry Pi is able to detect the Grove pi: run **i2cdetect** using this command:

   sudo i2cdetect -y 1



   If "**04**" is in the output **4**, means the Raspberry Pi is able to detect the GrovePi.

**** THIS IS AN IMPORTANT STEP FOR STUDENT TO CHECK THE FUNCTIONALITY OF GROVE PI, when you receive Error response from any execution, please remember to repeat this PART 2 accordingly.***

**** Latest update for Raspberry Pi may not able to support the command***
Sudo i2cdetect -y 1

PART 3: Create your own project to read data from sensors

Step 1: Create a folder under Linux Debian Operating system.

1. Launch the terminal and key in the following command line.

```
sudo mkdir /home/pi/Desktop/[YOUR_NAME]/   // Create a folder
```

2. Set the permission of the test folder to be accessible and writable by anyone.

**** Important step when you want to create / save files in any folder.***

```
sudo chmod a+w /home/pi/Desktop/[YOUR_NAME]/
```

In Raspbian, we commonly use *sudo* (superuser do) to execute commands, especially deal with file system modification.

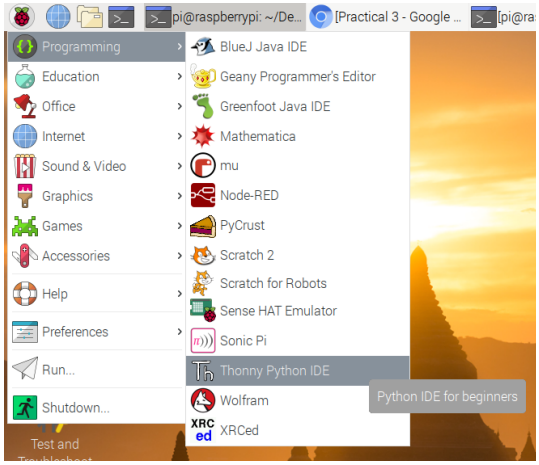
Notes: mdkir, chmod, i2cdetect are a few commands which require superuser privillege

Step 2: Create a new python file

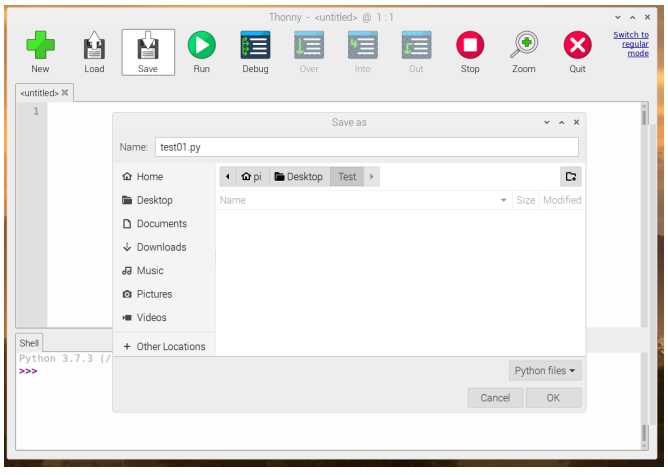
**** Python programming is very concerned with character spacing and case sensitive.***

**** There are minor errors on the provided codes throughout the practical notes as a part of students' assessment, that requires students to identify it based on logical programming skills.***

1. Go to Menu > Programming and launch **Thonny Python (IDE)**

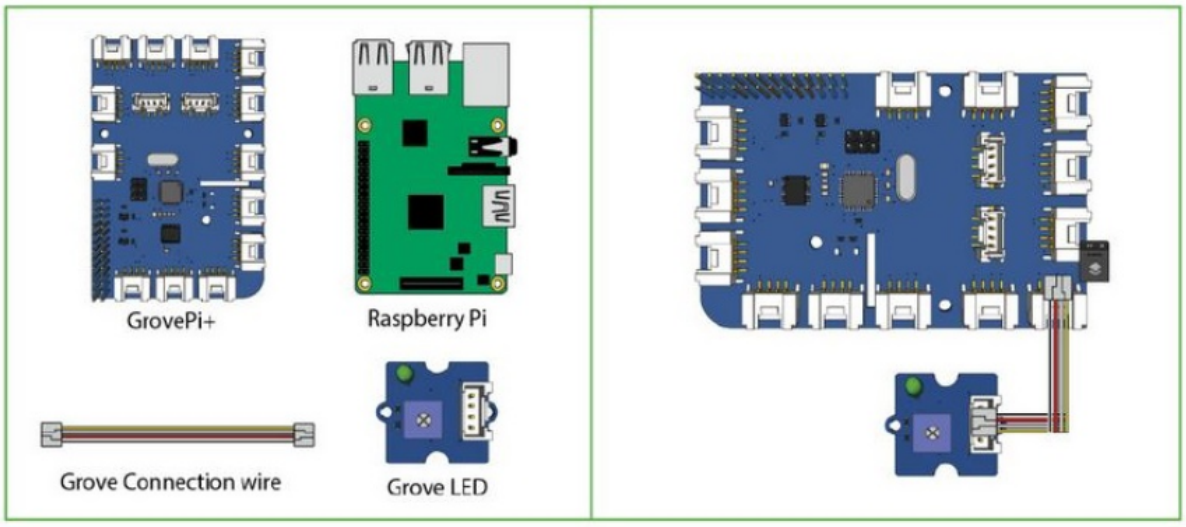


2. Save the untitled file to **/Pi/Desktop/[YOUR_NAME]/** to create a new python file with a new name: **test01.py**
   - If you cannot save the file, maybe you have missed the Step 1 for permission setting.



*** NOTE: Ctrl + C** or **Ctrl + Z** is always the best way to stop the execution by initiating KeyboardInterrupt.

Step 3: Test on LED Blink



1. Connect a LED to D4:

2. Write the following codes:

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) // Sleep one second
9          digitalWrite(led, 1) // turn on led 1 sec
10         time.sleep(1) // Sleep one second
11         digitalWrite(led, 0) // turn off led 1 sec
12     except KeyboardInterrupt:
13         digitalWrite(led, 0)
14         break
15     except IOError:
16         print("IO Error occurs")
17
```

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

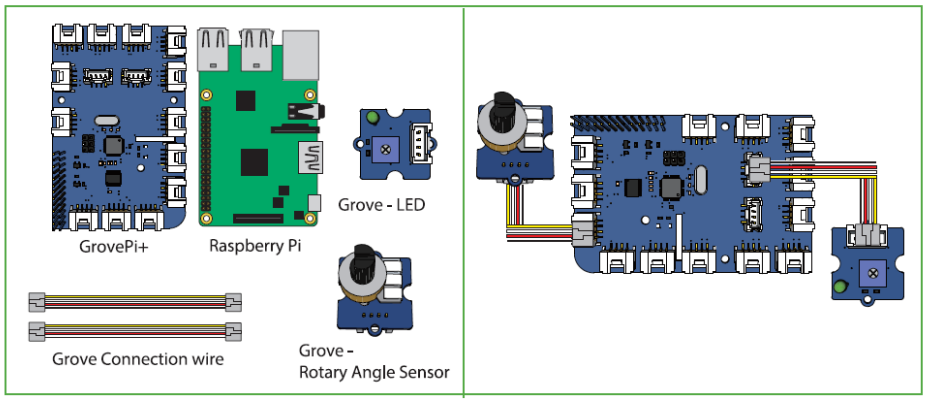
Run code by pressing F5 OR Thonny > Run current script (arrow right with green bg)
Stop the program by pressing ctrl + F2 OR Thonny > Stop/Restart backend (red by with stop text)

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

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

Step 4: Test LED Fade.

**** Picture is for illustration ONLY, please follow the steps correctly.***



change to A2

1. Connect a Grove LED to port D6 and the Rotary encoder to port A0.

2. In **Thonny Python (ID)**, click "New" to create a new python file and Save As "test02.py". Type the following codes: *A0, A1, A2 support analogRead from value 0 -1023

test02.py

```
1  import time
2  import grovepi
3
4  rotaryangle = 14 // change to 2 (A2 - read value from 0 - 1023)
5  led = 6 // As it connected to D6
6
7  grovepi.pinMode(rotaryangle, "INPUT") configure rotaryangle (A2) as input pin
8  grovepi.pinMode(led, "OUTPUT") configure led(D6) as output pin
9  i = 0
10
11 while True:
12     try:
13         time.sleep(1)
14         i = grovepi.analogRead(rotaryangle) you can rotate the rotaryangle to change the value from 0 -1023
15         print(i)
16         grovepi.analogWrite(led, i//4) Divide it to ensure the value is within 0 - 255
17     except IOError:
18         print("IO Error occurs")
```

3. Observe the difference of including libraries between ***import*** and ***from X import Y***.
   Differences : pinMode(led, "OUTPUT") versus grovepi.pinMode("INPUT")
4. Run the code

Task 1: Add another exception case to ensure the led is off when keyboard interrupt occurs. Add KeyboardInterrupt exception

Task 2: Modify the code to make the LED light more / less sensitive to the rotary angle (also called a potentiometer). 1023 - i: fade inversely

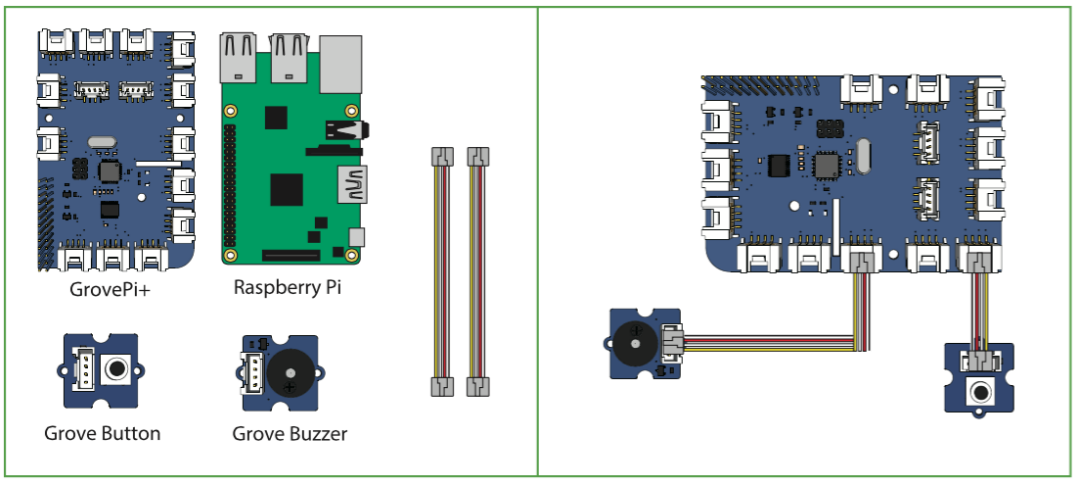
Task 3: Extend the module to 2 LEDs (Blue / Red / Green) and modify the code to make the LEDs fade *inversely* to each other with respect to the potentiometer value.

Notes: A potentiometer is a three-terminal resistor with a sliding or rotating contact that forms an adjustable voltage divider.

Step 5: Test Button and Buzzer.

**** Picture is for illustration ONLY, please follow the steps correctly.***

1. Connect a Grove Buzzer Module to port D2 and the Grove Button to port D4.



2. In **Thonny Python (ID)**, click "New" to create a new python file and Save As "test03.py". Type the following codes:

```
import time
import math
from grovepi import *

buzzer = 2   // It means buzzer has been connected to D2
button = 4   // It means button has been connected to D4
pinMode(buzzer,"OUTPUT")
pinMode(button,"INPUT")

while True:
    try:
        time.sleep(0.25)
        bStatus = digitalRead(button)
        if bStatus:
            digitalWrite(buzzer,1)
        else:
            digitalWrite(buzzer,0)
    except KeyboardInterrupt:
        digitalWrite(buzzer, 0)
        break
    except IOError:
        print("IO Error occurs")
```

TO ENSURE IF ANY ERROR, IT WILL ONLY HAPPENED IN EVERY 0.25 SEC.

3. Observe the difference of having / not having `time.sleep(0.25)` at line 12.

It will hold on 0.25 seconds before digital read the button status (whether 1 or 0) and write the buzzer to high or low

4. Run the code.

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

*** Clue: understand the difference between each port function and analogWrite / DigitalWrite.**

```
import time
from grovepi import *

buzzer = 3  // Buzzer is connected to D3
pinMode(buzzer, "OUTPUT")
while True:
    val = input("Enter a value between 0 - 255: ")
    val = int(val)
    if val > 255
        val = 255
    elif val < 0
        val = 0
    analogWrite(buzzer, val)
```

255
50
255
50
can make the buzzer sound like a warning siren / ambulance

* D3, D5, D6 allow us to write 8-bit values 0-255 with analogWrite()