Raspberry PI (RPI) 3 Model B

EEE3096S RPi LAB MANUAL

CHAPTER 3

Author: Do Yeou Ku

University of Cape Town

Revised for EEE3096S by S. Winberg and O. Konan, June 2018

Practical 3: RPI-3B GPIO (RPI-Python Basics)

The objective of the practical 3 is to write code for a simple LED brightness-control system. The system includes three switches and a LED.

If the *brighter* switch is pressed 10 times continuously, the LED will light up completely and after the 10th pressing, the LED should be off completely and start again.

If the *dimmer* switch is pressed 7 times continuously, the LED will be off and after the 7th pressing, the LED should be on completely and start again.

If the *on-off* switch is pressed the LED should switch between fully on and off.

1. Answer the following questions:
    a. What is the frequency that is appropriate for this system where the light should be visible? Justify your answer. (2)
    b. What is the increase/decrease in duty cycle after a *brighter/ dimmer* switch is pressed? Show your reasoning. (4)
    c. Draw a flowchart of above system (4)
2. Demonstration.
    a. *Brighter* (4)
    b. *Dimmer* (4)
    c. *On-Off* (2)

You might want to use a VNC if you have trouble using the terminal.

At the end of practical 3, you are to submit one .py file and one .pdf file including your answers on Vula. The pdf and python file must have the following format
prac_<prac_number>_<std_number>.<extension>

I.

I. RPI-Python Basics

A. Writing Python script (.py) to interact with GPIO pins (*Skip this section if you have used the image on Vula*)

We will be using Python 3. Raspbian, just like any Linux based OS, comes with both Python 2.7 and 3.5 by default. Since both versions are installed by default, it is important to know how to distinguish the two. Run following commands.

```
$ python -v

$ python2 -v

$ python3 -v
```

It may be important to install necessary missing modules, this may need the RPi connected to the internet to do so... which means you may need to connect to internet via Wi-Fi or to use the RPi in headed mode with the Ethernet cable plugged in to the network.

```
$ sudo apt-get update

$ sudo apt-get install python3-rpi.gpio
```

I recommend using APT since all modules will be updated by a single command line. However, not all Python packages are available in the Raspbian archives. These modules can be found by installing packages from the Python Package Index (PyPI).

```
$ sudo apt-get install python3-pip

$ pip3 install <module>
```

Finally, it is important to create a python script in which you can code. To create or access a python script in a directory/folder of your choice, type the command below. If there is no file of the given filename, a new script will be created.

```
    $ sudo nano firstpy.py
```

For any python script you create, it is important that you have `#!/usr/bin/python` as your first line to specify that it is a python script. To execute a python script, following command is used.

```
    $ sudo python filename.py
```

B. Digital Output

The Raspberry Pi has been around for almost a decade now and consequently there are numerous libraries available for you to use. Some libraries are specialized such that they allow control of a specific component or communication protocols. In this document, you will be introduced to some of the most basic libraries (that are probably also the most useful of libraries).

As with any programming, first thing to do is to create a new code file. Open a terminal whether or not you are using a headless Raspberry Pi.

```
#!/usr/bin/python


import RPi.GPIO as GPIO
import time


GPIO.setmode(GPIO.BCM)              # use GPIO pin numbering
# may change to GPIO.BOARD phy.pin if preferred
pinnumber = 22                      # set pinnumber to pin 22
second = 1
GPIO.setup(pinnumber, GPIO.OUT)     # set pin 'pinnumber' as digital output
GPIO.output(pinnumber, GPIO.LOW)    # set pin 'pinnumber' default low
GPIO.output(pinnumber, True)        # set pin 'pinnumber' output high
time.sleep(second)                  # delay for 'second'


GPIO.output (pinnumber, False)      # set pin 'pinnumber' output low
time.sleep(second)                  # delay for 'second'


GPIO.cleanup()                      # release GPIO pins from its opeartion
```

C. Digital Input

As with the case for configuring a microcontroller for digital input, a RPI pin needs to be configured either in pull-up or pull-down mode depending on the application.

For a refresher on pull-up/down resistors, read https://www.electronics-tutorials.ws/logic/pull-up-resistor.html

```
#!/usr/bin/python


import RPi.GPIO as GPIO


GPIO.setmode(GPIO.BCM)              # use GPIO pin numbering

# may change to GPIO.BOARD phy.pin if preferred

pinNumber = 22                      # set pinnumber to pin 22


# set pin 'pinNumber' as a digital input and use a pull-up resistor

GPIO.setup(pinNumber, GPIO.IN, pull_up_down=GPIO.PUD_UP)


# Alternatively, set pin 'pinNumber' as a digital input and use a pull-down
resistor

# NB!! ONLY USE ONE OF THESE COMMANDS.

GPIO.setup(pinNumber, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)


# Read digital input

# When 'pinNumber' is setup in pull-up mode, the default value will be true

# When 'pinNumber' is setup in pull-down mode, the default value will be false

Digital_input = GPIO.input(pinNumber)


GPIO.cleanup()                      # release GPIO pins from its opeartion
```

D. Analogue Output

On RPI, there is no pin to output true analogue, i.e., continuous, signal. It is merely simulation of analogue voltages using pulse width modulation (PWM). Since for a high frequency signal, the output voltage is the average voltage of a signal.

$$V_{analogue} = V_{avg} = \frac{1}{T}\int_0^T V_{out}\ dt$$

Since the maximum voltage outing of digital pins on RPI is 3V3. The analogue voltage output range is between 0 and 3V3.

```
#!/usr/bin/python


import RPi.GPIO as GPIO

import time


GPIO.setmode(GPIO.BCM)              # use GPIO pin numbering

# may change to GPIO.BOARD phy.pin if preferred

pinnumber = 22

GPIO.setup(pinnumber, GPIO.OUT)     # set pin 'pinnumber' as digital output


PWM = GPIO.PWM(pinnumber, f)        # f is the PWM frequency

PWM.start(dutycycle)                # 0 < dutycycle < 100

PWM.ChangeDutyCycle(newdutycycle)   # used to change the PWM duty cycle

PWM.ChangeFrequency(newfrequency)   # newfrequency [Hz]

PWM.stop()


GPIO.cleanup()
```

E. Analogue Input

On RPI, there is no pin to output Raspberry PI does not have inherent ADC therefore it is not possible to obtain analogue input reading without additional component. A component that I recommend is MCP3008.

MCP3008 is essentially an 8-channel ADC chip (allows 8 analogue inputs) and it uses SPI serial communication protocol to communicate with the mother-chip/board.