Processor Hardware/Software Interface
EECS 113

Assignment 4 - Raspberry Pi

University of California, Irvine

Assigned: May, 13, 2021

Due: May, 20, 2021 by 11:55PM

1 Raspberry Pi Programming

1.1 Warnings

- "Before connecting power, get into the habit of checking that there is nothing conductive in contact that could cause a short circuit with you Raspberry Pi. A quick check that there is nothing nearby could save you from damaging your Pi."
- "Electricity can kill! Only experiment with low voltage and currents, and never work with mains. If you are ever in doubt you should check with someone suitably qualified."
- "Be extremely careful when working with circuits (especially 5V) that connect to the GPIO pins as they are not protected on the Raspberry Pi and the external power supply."

1.2 Objective

Interrupt-driven Input/Output on Raspberry Pi with pushbuttons (inputs) and LEDs (outputs).

1.3 Hardware Components

Raspberry Pi, Pushbuttons (green, red, yellow, blue), LEDs (preferably same color as pushbuttons), Resistors, Breadboard, Jumper Wires, Ribbon Cable, Raspberry Pi GPIO Extension Board.

1.4 Main Idea

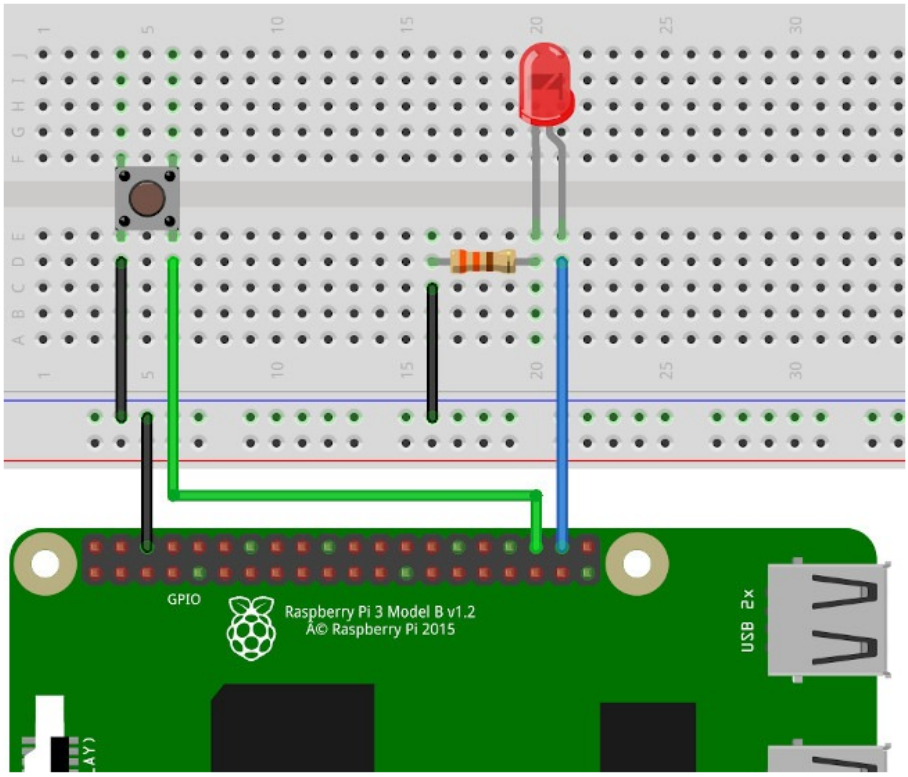
In this assignment, you are going to wire up four input pins to pushbuttons, and four output pins to light-emitting diodes (LEDs), and then write a python script to blink the LEDs when the special button combination is detected and exit from blinking mode when some conditions are met.

1. Please complete the Raspberry Pi setup by following the setup guide document which is provided for you already (check week 6 module in Canvas).

2. Wire up four inputs to pushbuttons, and four outputs to light-emitting diodes (LEDs).

- Inputs: Pushbutton style switches connect two points in a circuit when you press them and they can be straddled across the center divider of the breadboard.

- Outputs: LEDs are suitable for use with the 220Ω resistors at +3.3V, at a reasonable brightness. Wiring should be straightforward. The following figure shows the circuit schematic connecting one pushbutton and one LED to the breadboard (black wires are connected to the GND pin).



3. To make the connections to the RPi3 GPIO, use a Pi Cobbler which is a ribbon cable, and a Raspberry Pi GPIO Extension Board (it is used to lead out pins of Raspberry Pi to breadboard to avoid GPIO damage caused by frequent plugging in or out. Check Appendix section for the pin mapping) that connects to a solderless breadboard, where you can add the required hardware components to build the circuit.

4. You can use the following port selection (Appendix section shows the GPIO pin layout for raspberry Pi 3B+): Different port selections may be used based on your Raspberry Pi GPIO borad information.

- Pin #22 / GPIO 25: green pushbutton
- Pin #12 / GPIO 18: red pushbutton
- Pin #13 / GPIO 27: yellow pushbutton
- Pin #15 / GPIO 22: blue pushbutton
- Pin #29 / GPIO 5: green LED
- Pin #31 / GPIO 6: red LED
- Pin #32 / GPIO 12: yellow LED
- Pin #33 / GPIO 13: blue LED

5. After wiring up all the pushbuttons and LEDs and resistors to the GPIO pins, write a python script using RPi.GPIO library (check "GPIO library in Python" in Appendix section) in which:

- When both **Yellow** and **Blue** buttons are pressed simultaneously, the script will enter the blink mode and toggle all of the outputs (LEDs) in each iteration, at a rate controllable by **Red** and **Green** buttons (Red and Green buttons control speed of blinking (slower and faster) for LEDs).
- All of the outputs (LEDs) stop blinking (exit blink mode) when both **Yellow** and **Blue** buttons are pressed simultaneously, or an activated timer reaches a specific threshold (example: 5s to 10s delay).
  Note: you need to oscillate LEDs from on to off to simulate the blinking mode and for this you can consider an LED to be turned off for a time delay and be turned on for the same time delay (example: 0.3s delay)).

6. Create a directory with the name of **gpio** in your home directory (**/home/pi/gpio**) and save your Python script with the name of **assignment4.py** in this directory.

7. Make a copy of the provided script (check the Assignment 4 folder in "Canvas" to access the file) and save it in the directory (**/home/pi/gpio/**) with the name of **gpint** (without any extension).
Note: You need to make sure that your **gpint** file is an executable file.

8. Execute the following commands in terminal:

- **chmod a+x /home/pi/gpio/assignment4.py**
  Note: There is a space between chmod and a+x. A space between a+x and /home/pi/gpio/assignment4.py
- **sudo ln -s /home/pi/gpio/gpint /etc/init.d**
  Note: There is a space between sudo and ln. A space between ln and -s. A space between -s and /home/pi/gpio/gpint. A space between /home/pi/gpio/gpint and /etc/init.d
- **sudo /etc/init.d/gpint start**
  Note: There is a space between sudo and /etc/init.d/gpint. A space between /etc/init.d/gpint and start
  - This command starts running your Python script in background.
  - For stopping your script from running in background, you can execute the following command:
    * **sudo /etc/init.d/gpint stop**
- **sudo ln -s /etc/init.d/gpint /etc/rc5.d/S01gpint**
  Note: There is a space between sudo and ln. A space between ln and -s. A space between sudo and /etc/init.d/gpint. A space between /etc/init.d/gpint and /etc/rc5.d/S01gpint

9. After following step 8, the bash script (**gpint** file) runs your Python script in background continuously and listens for button events to light up the LEDs. You can test your code by pressing pushbuttons (which triggers an interrupt) to see if your script outputs the desired result.

Note: You can use your own method to implement this assignment. If you are not comfortable using the provided bash script (which runs your Python code in background), you can develop your code in a way that doesn't need working with the bash script.

10. Following is the suggested structure you may use for your code:

```python
#!/usr/bin/python
# Assignment 4 #

### import Python Modules ###
import threading
import RPi.GPIO as GPIO
import time     # for time delay and threshold

### Pin Numbering Declaration (setup channel mode of the Pi to Board values) ###


GPIO.setwarnings(False)    # to disable warnings

### Set GPIO pins (for inputs and outputs) and all setups needed based on assignment description ###


# This fucntion takes care of blinking and is called by the blinking thread (check line 33)
def blink_thread():

# This function catches interrupt and spawn the blink_thread() thread to handle the interrupt
def handle(pin):

    # light corresponding LED when pushbutton of same color is pressed

    t = None
    # when yellow and blue pressed simultaneously, enter blink mode
    # You may add some extra info to this part based on your implementation
    if (pin == BTN_Y or pin == BTN_B):
            # print("starting thread")

            # entering thread
            t = threading.Thread(target=blink_thread)
            t.daemon = True
            t.start()     # start threading


### Event listener (Tell GPIO library to look out for an event on each pushbutton and pass handle function)
### function to be run for each pushbutton detection ###


# endless loop with delay to wait for event detections
while True:
    time.sleep(1e6)
```

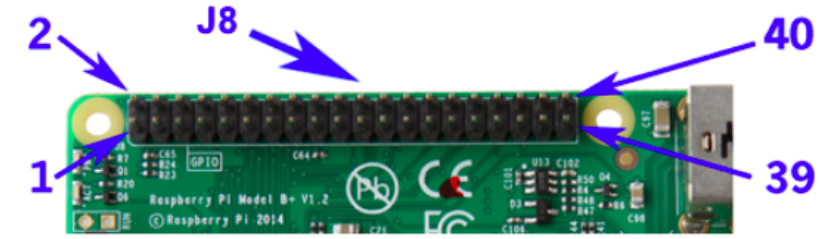
2 Assignment Deliverable

- Create and submit a 2 to 3 minute video (not more than 5-minute) of running and testing your code based on the assignment description.
  - Note that you need to state your name and student ID in the video. You also have the option of appearing in the video but are not required to do so.
  - You need to explain your code and your approach.
- Submit your Python code and your video in **Canvas** before deadline in a **.zip** file (your code needs to be commented extensively).

Appendix

GPIO (General Purpose I/O):

The Raspberry Pi 3B+ board contains a single 40-pin expansion header labeled as 'J8' providing access to 26 GPIO pins (pins 1, 2, 39 40 are also labeled below):



J8 Pinout (40-pin Header)

The following figure shows the GPIO on Raspberry Pi's 40-pin J8 header:

Alternate Function						Alternate Function
	3.3V PWR	1	2	5V PWR		
I2C1 SDA	GPIO 2	3	4	5V PWR		
I2C1 SCL	GPIO 3	5	6	GND		
	GPIO 4	7	8	UART0 TX		
	GND	9	10	UART0 RX		
	GPIO 17	11	12	GPIO 18		
	GPIO 27	13	14	GND		
	GPIO 22	15	16	GPIO 23		
	3.3V PWR	17	18	GPIO 24		
SPI0 MOSI	GPIO 10	19	20	GND		
SPI0 MISO	GPIO 9	21	22	GPIO 25		
SPI0 SCLK	GPIO 11	23	24	GPIO 8		SPI0 CS0
	GND	25	26	GPIO 7		SPI0 CS1
	Reserved	27	28	Reserved		
	GPIO 5	29	30	GND		
	GPIO 6	31	32	GPIO 12		
	GPIO 13	33	34	GND		
SPI1 MISO	GPIO 19	35	36	GPIO 16		SPI1 CS0
	GPIO 26	37	38	GPIO 20		SPI1 MOSI
	GND	39	40	GPIO 21		SPI1 SCLK

GPIO Library in Python:

The GPIO is an easy and convenient means of providing simple button input and LED output on a headless Raspberry Pi. Here are two links which provide you some information about RPI.GPIO Python module in Raspbian which supports interrupts:

http://tieske.github.io/rpi-gpio/modules/GPIO.html
https://learn.sparkfun.com/tutorials/raspberry-gpio/python-rpigpio-api