Lab 1: Using GPIO as Input and Output

---

1 Learning Objective

In this lab, you are going to learn how to implement human and hardware interactions, read some digital data, and output digital data for actuation through Raspberry Pi's GPIO interface. Input data can be manipulated inside the Raspberry Pi before writing the outputs.

Here is some helpful reading about GPIO:
https://www.raspberrypi.org/documentation/computers/os.html#gpio-and-the-40-pin-header

2 Components Needed

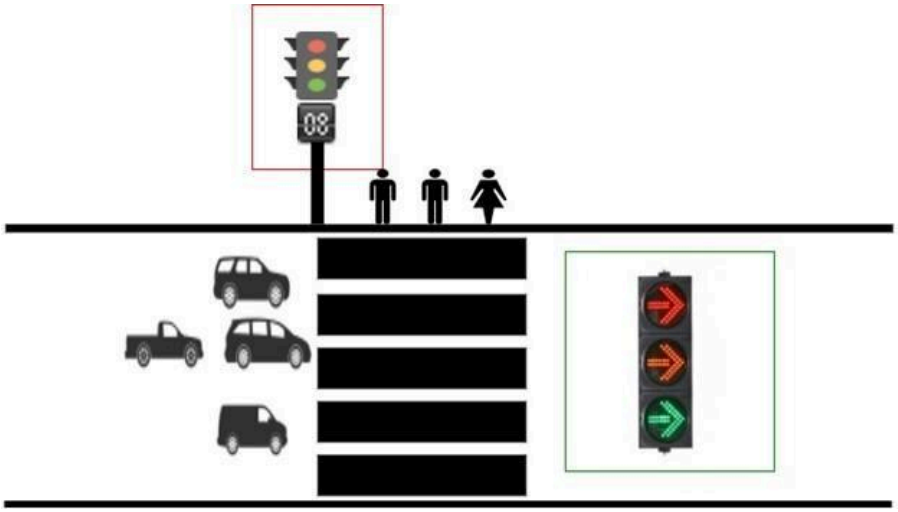
- Raspberry Pi 3
- 2x RGB LED(Blue for traffic light)
- Push button
- 7-segment display
- Resistor
- Appropriate wires

3 Lab Activities

In this lab activity, you are going to design a stoplight system using the GPIO on the Raspberry Pi 3. This system will be designed using polling and interrupt techniques.

3.1 System Requirement

1. **Introduction:** Imagine a scenario like the image below. You have a heavy traffic street where people are walking across the street while cars are driving. In this case, an appropriate traffic light system is needed for this street to ensure everyone's safety. Your
   goal is to design a control system for **traffic light 1 (in the red box)** & **traffic light 2 (in green box)**



2. **Components for traffic light 1:** RGB LED, countdown panel (7-segment display) and a push button (not shown in the figure). One RGB LED should work as it changes color and only one of the three colors is needed at any time.
3. **Components for traffic light 2:** RGB LED.
4. **System requirements:**
    a. When the button has not been pressed, traffic light 2 stays green
    b. When the button is pressed, traffic light 2 turns to blue, blinks 3 times, then turns red.
    c. When Traffic light 2 turns red, traffic light 1 becomes green and the countdown panel begins to count down from 9 to 0, in seconds. (In the real world it would be longer)
    d. When countdown reaches 4, traffic light 1 flashes with blue light until time 0.
    e. When countdown reaches 0, traffic light 1 becomes red, traffic light 2 becomes green.
    f. When the button is pressed once there will be a 20 seconds cooldown to be able to make another valid press.
5. **Implementation requirements: implement the system in both two ways as below**
    a. Read press button state using the polling method
    b. Read press button state using the interrupt method

3.2 GPIO map

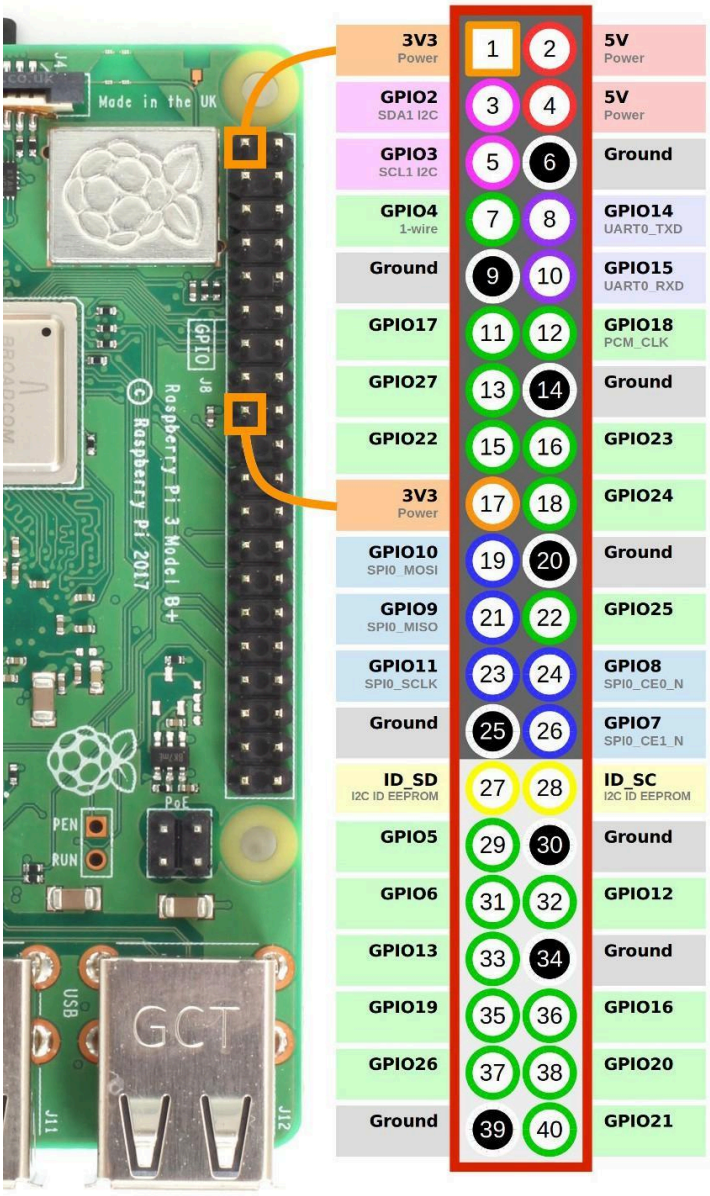
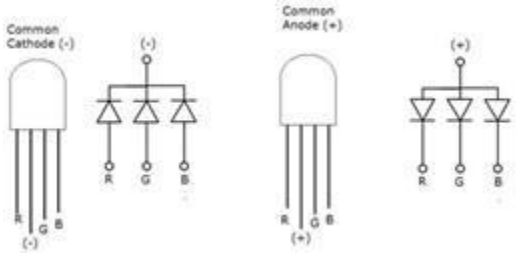


Image Source:
http://www.raspberrypi-spy.co.uk/2012/06/simple-guide-to-the-rpi-gpio-header- and-pins/

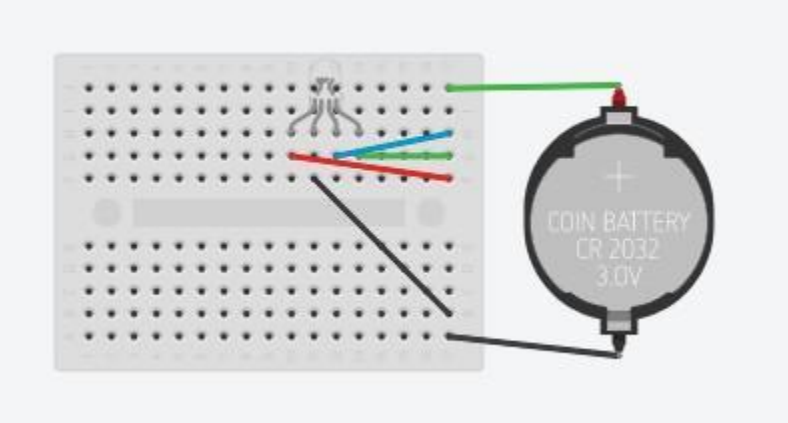
3.3 RGB LED

An RGB (Red, Green, Blue) LED produces three different colors of light when you connect the power source to the different pins. We provide a RGB LED that has the same hardware schema as the image below. We have a common Cathode (-), which means, the second node should connect to the ground while the rest of them connect to the output pins on the Raspberry Pi. A positive voltage on the output pin will cause the corresponding LED to light up.



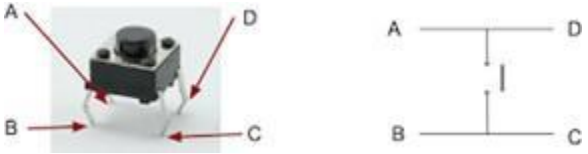
Always remember to connect a resistor to the ground node! The voltage forward each pin is around 2V(R),3.2V(GB). The red pin will get burned immediately if you connect it directly into a pin. The blue and green pins can both stay on for several seconds before they will generate lots of heat and get burned (if connected directly to the pins).

Circuit Design: A tentative schematic is shown below. You will determine appropriate pins, resistors, power supplies for your design.



3.4 Push Buttons

A push button is a very common component we used to control electronic devices. We usually use them as a switch to connect/disconnect circuits. First, you will need to understand the four pins on the push button:



When you're pressing the button, AD will be connected with BC, which will connect the circuit. **To protect your raspberry Pi, a resistor is suggested to be placed in the circuit.** Also, the input pin on the Pi will be floating when the button is not pressed (because there are no complete circuits attached to that pin). To deal with the pin's voltage floating, it may be beneficial to put a pull-down resistor attached to the input pin if the other side of the button is attached to positive voltage. If the button is attached to ground, then a pull-up resistor should be used. **Make sure the resistance is high enough that the input pin can go to the correct voltage when the button is pressed (preferably significantly higher than the resistor you placed in series with the switch).**

3.5 7-Segments Display

The data sheet can be found here depends on your model:
https://cdn-shop.adafruit.com/datasheets/2155datasheet.pdf https://www.es.co.th/Schemetic/PDF/TOS-5161AG-B.PDF
This lab activity only requires you to light up one number on the 7-segment display (countdown from 9 to 0). Read the data sheet carefully and understand how the 7-segment LED works.
Always connect ground with a resistor first before you connect the other pins.

3.6a Polling

In this implementation, we will light up LEDs by polling the pins using GPIO.input() in an infinite loop with 0.01 second delays between each iteration. Once the pin has been raised to a low or high voltage, the code written on the raspberry pi's should control the LEDs according to the system requirements.

3.6b Interrupt

The first type of implementation, when we light up LEDs by pressing the corresponding buttons, we used a polling method: calling GPIO.input() in an infinite loop with 0.01 second delay between each iteration, which is not very efficient and it will burn CPU power to continuously loop over and over again, even the buttons are rarely pushed. To improve this, we can use interrupts instead of polling.

An interrupt is a signal generated by a peripheral device that lets the CPU know that an action is happening on the device and the software will react depending on the action. The python RPI GPIO library provides different functions to help you describe what type of actions you are waiting for. In this case, instead of trapping in a loop you can add GPIO.add_event_detect() to tell the program what device action you are waiting for, and then GPIO.add_event_callback() to set up what action the program should take to handle the interrupt.

A quick tutorial:
https://medium.com/@rxseger/interrupt-driven-i-o-on-raspberry-pi-3-with-leds-and-pushbuttons-rising-falling-edge-detection-36c14e640fef

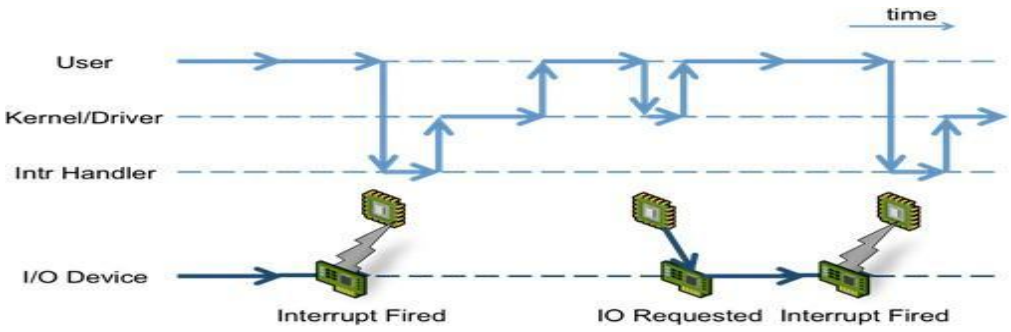


Image source: History of Interrupts: https://virtualirfan.com/history-of-interrupts

4 Questions to Explore

In your report, write down the discussion of the following questions (NOTE: You might want to search for the datasheet. All questions refer to the Raspberry Pi 3 Model b+ unless indicated otherwise):

1. What is RPI 3b+ CPU clock speed? Why is it important to know?
2. What is the range of voltage that represents logic low on Raspberry Pi?
3. What is the range of voltage that represents logic high on Raspberry Pi?
4. Compare the Raspberry Pi 3b+ with a standard PC, can you use Pi as a desktop PC? What's an application for RPi?
5. Discuss the difference(s) between polling and interrupt. What situation should we use polling and what situation should we use interrupt?
6. What is a hardware interrupt? What is a software interrupt?

5 What to Turn In

- Answer the questions in section 1.4 and submit it as a report (**pdf/word**) on Canvas(2 points).
  The report is an individual assignment. Should include:
  - Circuit Diagram
  - Code Included
  - Group Members
  - Lab Questions
- Include your code with the report **DO NOT ZIP** (0.5 point)
- Demo Traffic Light to TA. (2.5 points)
  - Circuit Review
  - Software Review
  - Demo Working Product
- The **submission policy** can be found in **the lab introduction file.**