Maina Collins-639890

Kamanzi Emery-637013

Traffic Lights System- A project report

Digital Electronics- APT 2030

Dr Sylvester Namuye

USIU- Africa

Spring 2016

ABSTRACT

The project involved creating a traffic lights system by using one set of traffic lights and a push button, and learn how to connect Raspberry Pi to lights and switches, and also learn some simple concepts of the Python programming language. The result of the project was the LEDs lighting in a sequence like a set of real traffic lights: green, amber, red, red & amber together, and then green again.

Contents

INTRODUCTION

The raspberry pi us being used in this project to manipulate the LEDs and light them in turn. Each GPIO port can be configured as either an input or output. The GPIO ports can only provide a small current. When connecting an LED then it is necessary to limit the current to prevent damage to the Raspberry Pi. We therefore used 220Ω resistors along with the LEDs to limit the current through the ports. We also used a resistor for the input switch. The switch would not need a resistor as it is an input, but one needs to be included in case the output is configured as an output with the switch pressed. The circuit diagram is shown below:

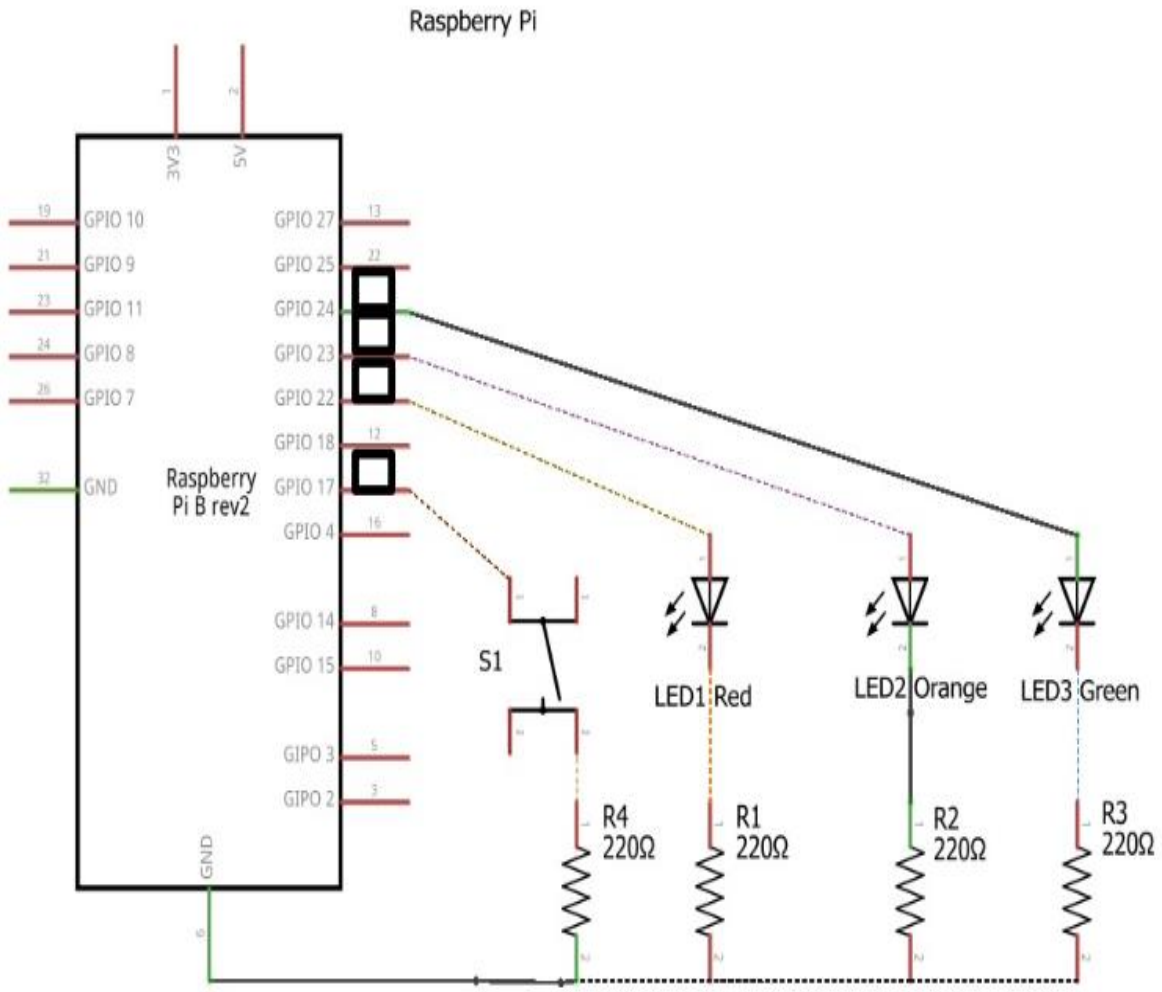

Figure 1 Circuit Diagram

OBJECTIVE

The objective of the project was to light up the LEDs in sequence of green, amber, red, red & amber together, and then green again.

REQUIREMENTS

- A Raspberry Pi with a recent version of Raspbian OS
- An internet connection to install the GPIO Zero Python library.
- 3 LED lights, preferably red, amber and green.
- A small breadboard.
- Some female – male jumper wires to connect to your Pi.
- Some male-male jumper wires to connect things on your breadboard to each other.
- 3 x 270Ω resistors.
- A push button switch

Below is an implementation diagram

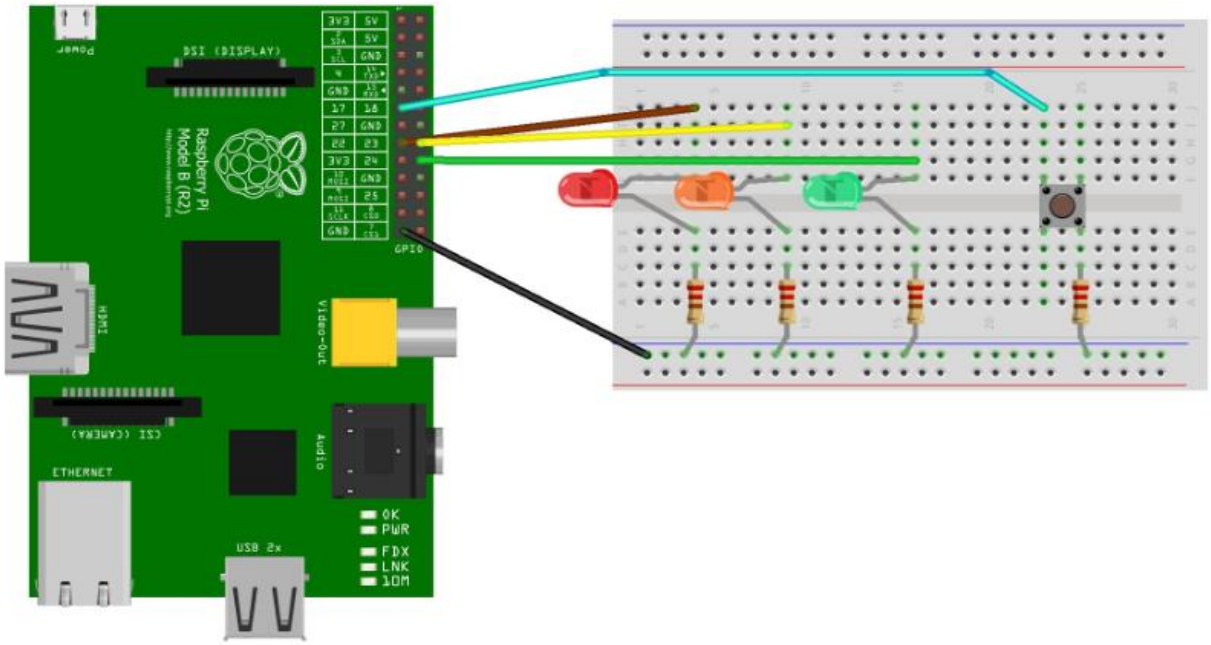

Figure 2 Implementation Diagram

This diagram shows the components plugged into a breadboard with jumper leads to connect to the Raspberry Pi. If you don't have a breadboard then it is also possible to create a circuit using crocodile clips to hold the wires together, or using female jumper cables. You should have already identified how the numbering of the GPIO connector works. As a reminder looking at the diagram below it is numbered left to right and then top to bottom. So pin 1 is top left, pin 2 is top right, pin 25 is bottom left and pin 26 is bottom right.

It is important that the LEDs are connected the correct way around, otherwise they will not work. The longer lead is normally the positive side of the LED (anode), or if there is a flat part (on a standard round LED) then that is normally the negative side (cathode). The resistors can be wired either way around – they should be labeled red (2), red (2), brown (x10), the 4th band is the tolerance (there is sometimes a 5th stripe). There are different switches that can be used. The one shown is a single push switch with 4 terminals. The two left hand pins are connected together as are the two right hand pins. Any push-to-make switch can be used as long as one side connects to the appropriate GPIO pin and the other to the resistor.

PROCEDURE

To access the GPIO ports then we need to run with root (superuser) permissions. This is easiest by running IDLE as root from a terminal shell.

```
sudo idle
```

IDLE starts in the shell mode where you can enter the following commands directly to test the circuit. First import the GPIO module and setup to use the GPIO port numbering scheme.

```
>>> import RPi.GPIO as GPIO
>>> GPIO.setmode(GPIO.BCM)
```

Each GPIO port needs to be setup as an output (GPIO.OUT) or an input as appropriate (GPIO.IN). The following will test the first LED by setting it as an output, turning the output on (set to true) and then turn it off again (set to false).

```
>>> GPIO.setup(22, GPIO.OUT)
>>> GPIO.output(22, True)
>>> GPIO.output(22, False)
```

Repeat this for all 3 LEDs. To test the input we need to set the pin up as an output. We also need to enable the internal pull-up resistors otherwise the input will be floating when the button is not pressed.

```
>>> GPIO.setup(17, GPIO.IN, pull_up_down=GPIO.PUD_UP)
>>> GPIO.input(17)
1
>>> GPIO.input(17)
0
```

The first time the input command is run is without the button pressed, which results in a 1 thanks to the internal pull-up resistor. The second time is when the switch is pressed which pulls the input down to a 0. Finally, the cleanup should be run to release the GPIO ports, otherwise there will be a warning when the program is next run.

The following screenshot shows these commands as run in the shell. Note that only one LED was tested in this instance, but you should test all 3 LEDs can be turned on and off using the appropriate GPIO port numbers.

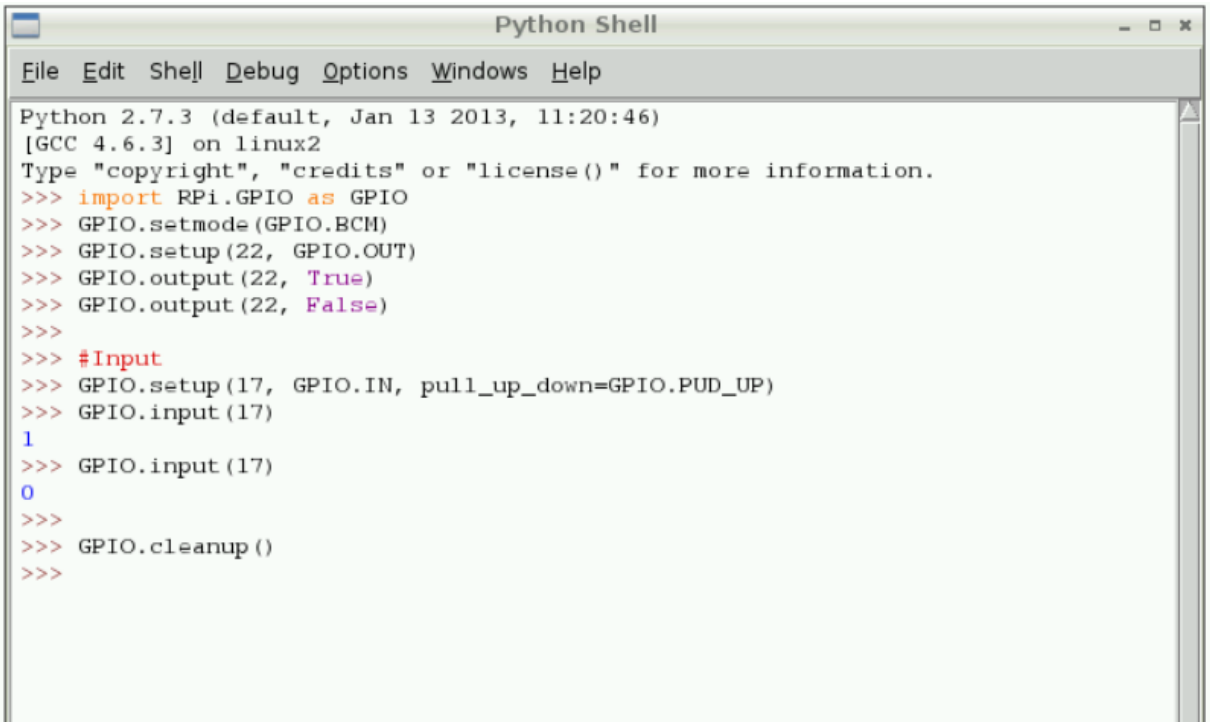

Figure 3 Commands screenshot

OBSERVATION AND RESULTS

The observation was the LEDs were able to light up in sequence. The result was the LEDs lit in sequence of green, amber, red, red & amber together, and then green again.

RECCOMMENDATIONS

The circuit will work better with more LEDs for a better sequencing pattern of the lights.

APPENDIX

Program Code

trafficlight.py

```
# GPIO traffic light program

# pin numbers
GPIO_RED = 22
GPIO_AMBER = 23
GPIO_GREEN = 24
GPIO_SWITCH = 17
```

```
# time delay in secs
TIME_CHANGE = 1
TIME_RED = 5
TIME_GREEN = 5     #only req if not using switch

# import RPi.GPIO module
Import RPi.GPIO as GPIO
# import time module used for sleep
import time

# Use GPIO pin numbering disable in-use warnings
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

# Setup relevant pins
GPIO.setup(GPIO_RED, GPIO.OUT)
GPIO.setup(GPIO_AMBER, GPIO.OUT)
GPIO.setup(GPIO_GREEN, GPIO.OUT)
GPIO.setup(GPIO_SWITCH, GPIO.IN, pull_up_down=GPIO.PUD_UP)

# start with red light on
GPIO.output(GPIO_RED, True)
GPIO.output(GPIO_AMBER, False)
GPIO.output(GPIO_GREEN, False)

# Loop keeps running
While 1:
        # red light on - sleep for set period of time
        time.sleep (TIME_RED)

        # Red and amber (puffin crossing - so don't flash)
        GPIO.output(GPIO_AMBER, True)
        # sleep for change time
        time.sleep (TIME_CHANGE)

        # Change to green
        GPIO.output(GPIO_RED, False)
        GPIO.output(GPIO_AMBER, False)
        GPIO.output(GPIO_GREEN, True)

        # sleep for button to press (or replace following code with a sleep if not
        using switch)

        #loops until GPIO becomes “Flase” ie switch pressed
        while GPIO.input(GPIO_SWITCH):
                #sleep used to reduce processor usage
```

```
        #sleep for ¼ sec between checking switch
        time.sleep (0.25)

#Change to amber
GPIO.output(GPIO_AMBER, True)
GPIO.output(GPIO_GREEN, False)

#sleep for change time
time.sleep (TIME_CHANGE)

#change to red
GPIO.output(GPIO_RED, True)
GPIO.output(GPIO_AMBER, False)
```