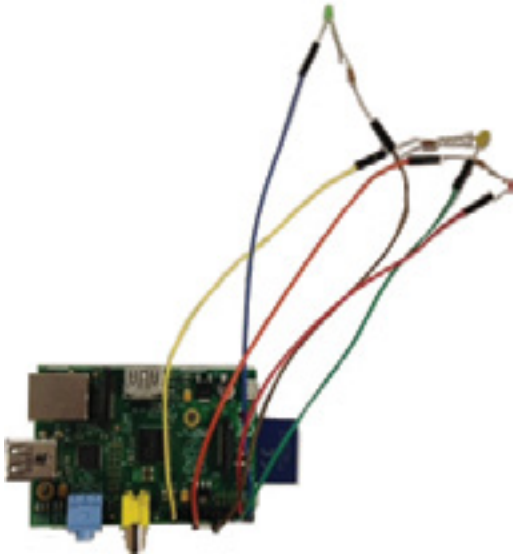
TRAFFIC LIGHTS LED RECIPE

A PHYSICAL COMPUTING PROJECT FOR THE RASPBERRY PI – NO SOLDERING, TOOLS OR INTERNET ACCESS REQUIRED!

Difficulty: Basic

This recipe will allow you to create a set of traffic lights by turning LEDs into output devices for your Raspberry Pi – we will guide you through writing a program to get them to light in the correct sequence.



Ingredients needed in addition to your Raspberry Pi:

3 x LEDs (red, yellow, green)

3 x 220Ω Resistors

6 x Jumper Wires (female to female)

A small rectangular piece of black card – with three holes for the LEDs

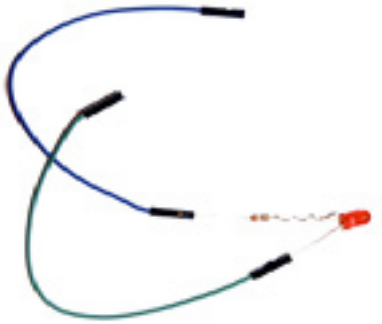
Method:

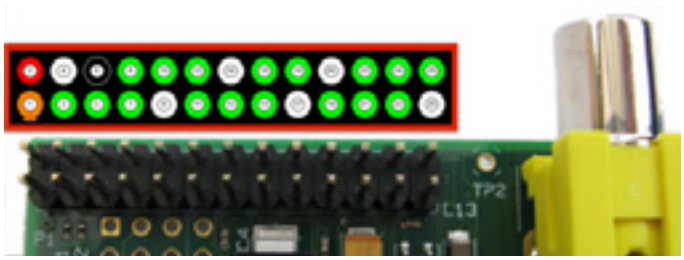
Turn the 3 x LEDs into outputs for your program

1. Take one end of the resistor and twist it around the cathode of the LED (nearest flat edge and the shorter lead) so that it forms a strong connection.

2. Push both the anode (longer lead) of the LED and the other end of the resistor into each of the jumper wires. Repeat this for all 3 LEDs.

3. For each LED take the end of the jumper lead connected to the cathode of the LED (flat edge, shorter wire) and push onto pins 17, 20 and 25 of the GPIO headers which are connected to ground.





Raspberry Pi GPIO header pins. The diagram above the pins shows the pin numbers. You will be using pins 3, 5, 7, 17, 20 and 25. ***Warning! You can damage your Raspberry Pi if you do not use the GPIO pins correctly!***

4. Then take the end of the other jumper lead and push onto pin 3 for the red LED, pin 5 for the yellow LED and pin 7 for the green LED of the General Purpose Input-Output (GPIO) header which is connected to the GPIO channels.

5. Push the LEDs through your black card in the correct order for traffic lights.

Congratulations! You have now attached the LEDs to your Raspberry Pi which can be used as an output in your programs.

. . . continued

WRITE A PROGRAM THAT LIGHTS UP THE LEDS IN SEQUENCE

1. Open a command line text editor

```
nano TrafficLED.py
```

2. Type in the code below (Pro Tip: Any lines beginning with a # symbol are comments so don't need to be included for the program to work - they will, however, help you to understand the code)

```
# First we need to import the libraries that
# we need

# Import the time library so that we can make
# the program pause for a fixed amount of time
import time

# Import the Raspberry Pi GPIO libraries that
# allow us to connect the Raspberry Pi to
# other physical devices via the General
# Purpose Input-Output (GPIO) pins
import RPi.GPIO as GPIO

# Now we need to set-up the General Purpose
# Input-Ouput (GPIO) pins

# Clear the current set-up so that we can
# start from scratch
GPIO.cleanup()

# Set up the GPIO library to use Raspberry Pi
# board pin numbers
GPIO.setmode(GPIO.BOARD)

# Set Pin 3 on the GPIO header to act as
# an output
GPIO.setup(3,GPIO.OUT)

# Set Pin 5 on the GPIO header to act as
# an output
GPIO.setup(5,GPIO.OUT)

# Set Pin 7 on the GPIO header to act as
# an output
GPIO.setup(7,GPIO.OUT)

# This loop runs forever and flashes the LED
while True:

  # Turn on the red LED
  GPIO.output(3,GPIO.HIGH)

  # Wait for 2 seconds
  time.sleep(2)

  # Turn on the yellow LED
  GPIO.output(5,GPIO.HIGH)

  # Wait for 2 seconds
  time.sleep(2)

  # Turn off the yellow LED
  GPIO.output(5,GPIO.LOW)

  # Turn off the red LED
  GPIO.output(3,GPIO.LOW)

  # Turn on the green LED
  GPIO.output(7,GPIO.HIGH)

  # Wait for 2 seconds
  time.sleep(2)

  # Turn off the green LED
  GPIO.output(7,GPIO.LOW)

  # Turn on the yellow LED
  GPIO.output(5,GPIO.HIGH)

  # Wait for 2 seconds
  time.sleep(2)
```

3. Run the program.

```
sudo python TrafficLED.py
```

Congratulations! Now when you run the program your traffic lights will work.

MAKE IT YOUR OWN!

There are many simple ways that you could create your own variations of this recipe, some examples of which are listed below:

- Change the rate at which the LEDs flash
- Require a user input to trigger the LEDs – so it becomes a pedestrian crossing!