Raspberry Pi LEGO® Traffic Lights

Tutorial by Andrew Oakley
Public Domain 26 Sept 2015
www.cotswoldjam.org

Introduction

This tutorial shows you how to build a set of traffic lights based around LEDs (lights) and a Raspberry Pi computer. The lights can be placed inside a LEGO® Technic brick and used with other LEGO® models.

Getting started

Conventions

```
Words you will see on the screen are highlighted like this
Words you need to type in are highlighted and underlined
```

At the end of each line, you will usually have to press the Enter key.

Your tutor should have prepared your Raspberry Pi for you. If not, see the section “Preparation” at the end.

The electronics

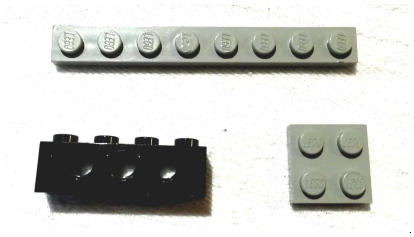
Components

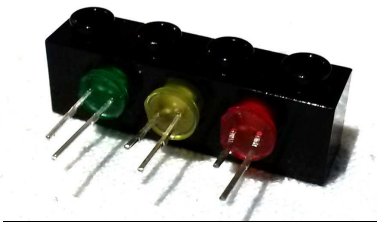
An LED has a short leg and a long leg. If you feel around the rim, you'll also find a flat edge. The short leg and flat edge always connect to negative (ground). We use 5mm LEDs because they fit into LEGO® Technic holes.

LEDs always need to be connected with a resistor. If you connect them without a resistor, they will burn out and probably won't work again.

The resistor can be connected any way around. Lower value (lower ohm) resistors will allow more current through and will make the LED brighter; higher values will let less current through and make it dimmer. We're using a 270 ohm resistor but anything between 220-470 ohms will work fine. We will use one resistor connected to ground to cover all 3 LEDs.

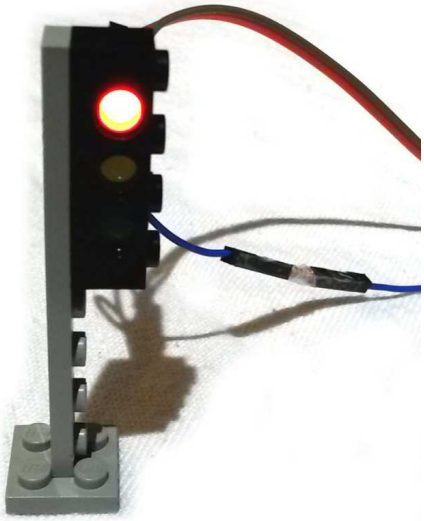
These three LEGO® bricks can be made into a traffic light, and the 5mm LEDs fit inside the holes of the Technic brick.





The part numbers for these bricks – which can be purchased second-hand on sites such as eBay or Bricklink.com – are:

3701 – Technic, Brick, 1x4 with holes, black
3460 – Plate, 1x8, grey
3022 – Plate, 2x2, grey



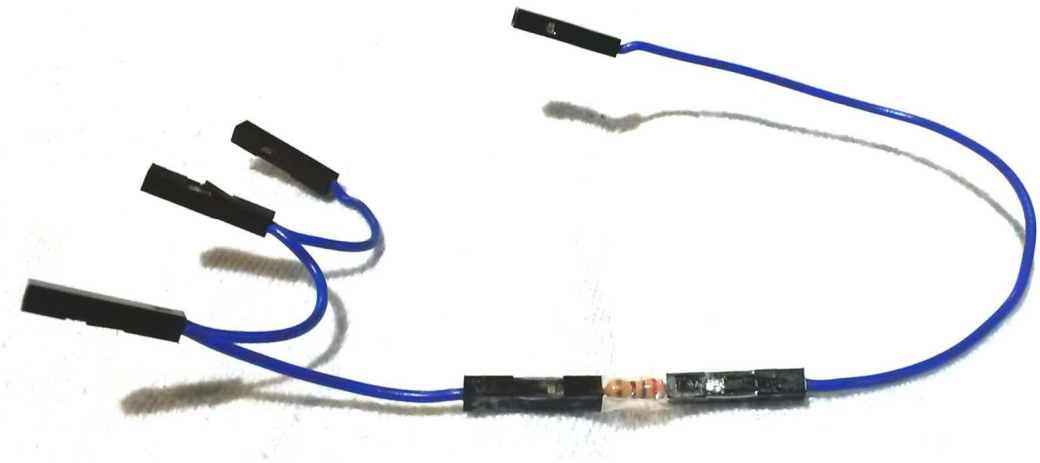
Push the LEDs into the Technic brick so that the short legs of each LED are at the bottom.

You will need to connect the short legs to a single resistor and then to one of the Raspberry Pi's ground pins. Your tutor may have made up a special "pigtail" cable to do this, or you may use a breadboard.

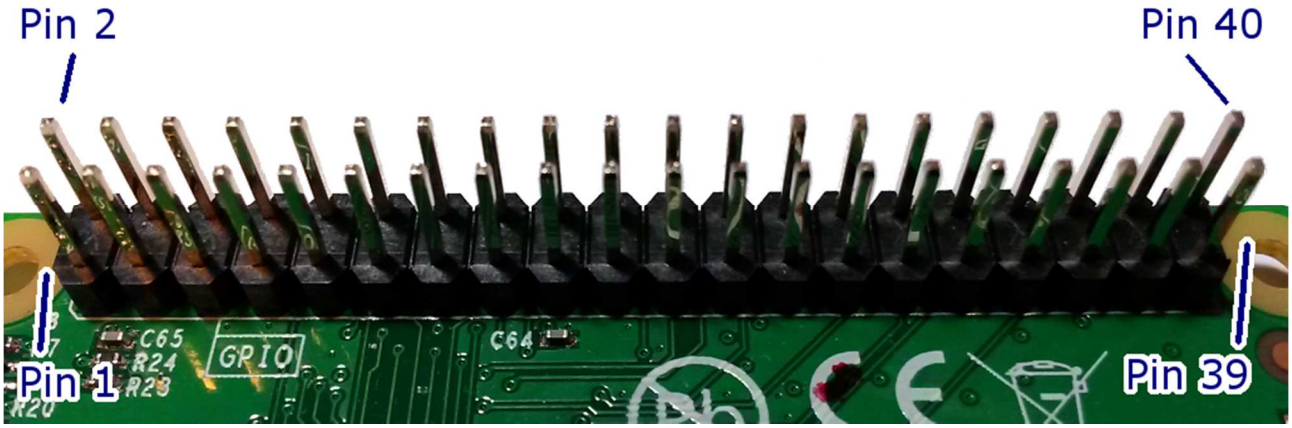
Ground (negative) wires

Pigtail cable

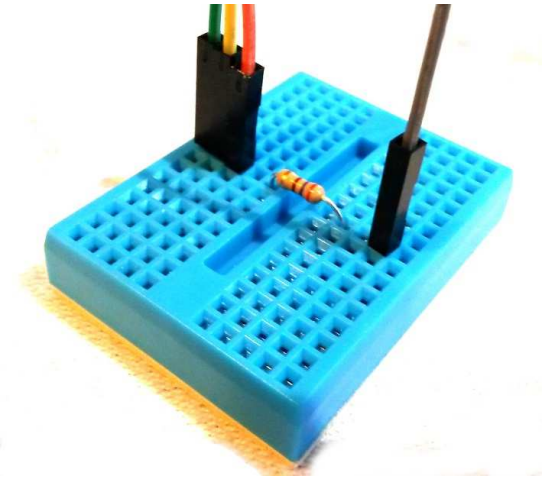
Your tutor may have made up this cable using a Dupont crimper.



It has three female ends on one side, which plug into the short legs of the LEDs, a resistor in the middle, and a single female end on the other side which plugs into pin 9 (ground) of the Raspberry Pi. A short bit of clear sticky tape wraps around the resistor to hold it in place.



Breadboard



If you don't have a specially-made pigtail cable, you can use a breadboard to connect the three LEDs' short legs to one resistor and ground.

You will need four male-to-female jumper wires, plus the resistor.

Do not worry about the colours of the jumper wires. They do not have to match the LED colours. The author of this document is heavily colour-blind, so the colours will be random anyway.

The three jumper wires at the top of the breadboard go to the short legs of the LEDs. The single jumper wire at the bottom goes to ground (pin 9) on the Raspberry Pi.

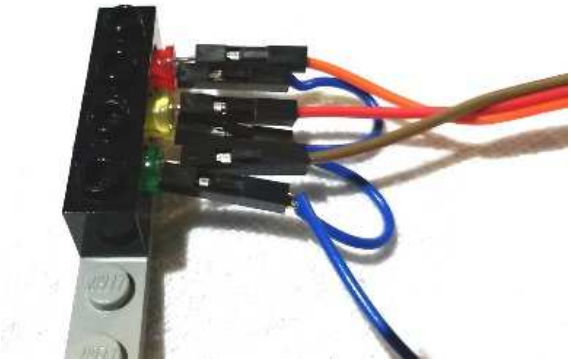
Signal (positive) wires

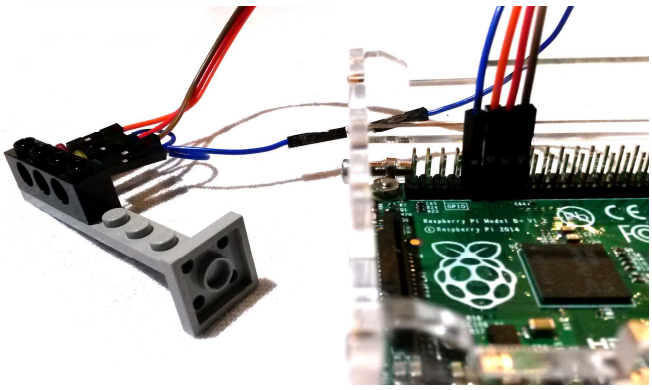
Once you've got the resistor and ground connected to the LEDs' short legs, you can test any LED by using a female-to-female jumper wire to connect the long leg to pin 1 (live 3.3 volts) on the Raspberry Pi (assuming the Raspberry Pi is powered on).

However we don't want the lights on all the time; we need them to be controlled by a program. Disconnect pin 1 once you're finished testing the lights.

To control the LEDs from our program, use female-to-female jumper wires to connect the long (positive) legs of the LEDs to the following pins on the Raspberry Pi:

Pin 11 – Red
Pin 13 - Yellow
Pin 15 – Green





The program

Power up your Raspberry Pi, log in and go to the desktop. If you need to log in, the default username is `pi` and the password is `raspberry` . If the desktop still does not appear, type `startx` and press the enter key.

From the menu, select Programming - Python 3. Then use File, New Window to create a new program.

Type in the following program, or alternatively you can use File, Open to open the traffic1.py program in the python/traffic folder.

```
import RPi.GPIO as GPIO, time
GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)

red=11
yellow=13
green=15

GPIO.setup(red, GPIO.OUT)
GPIO.setup(yellow, GPIO.OUT)
GPIO.setup(green, GPIO.OUT)
```

Use File, Save to save this program as traffic1.py .

Here's what this program is doing:

```
import RPi.GPIO as GPIO, time
```

> This command teaches the Raspberry Pi about its GPIO pins, so it can turn the LEDs on and off, and about time, so we can tell it to wait for a few seconds between each colour.

```
GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)
```

Here we're telling the computer to use call the pins by the numbers in the order they are laid out on the board (there is an alternative way of numbering them called BCM). Next we tell it not to warn us if the GPIO pins have already been used.

```
red=11
yellow=13
green=15
```

Now we're telling the Raspberry Pi which pins are used for which coloured LEDs. Note that we don't have to state that we're using pin 9 for ground; it doesn't matter.

```
GPIO.setup(red, GPIO.OUT)
GPIO.setup(yellow, GPIO.OUT)
GPIO.setup(green, GPIO.OUT)
```

Finally we tell the Raspberry Pi to prepare the red, yellow and green pins for output. If we were getting input, such as reading a switch, then this bit would look different.

This part of the program just gets everything ready. It doesn't actually turn any of the LEDs on. We'll do that next.

Use File, Save As to save the program as `traffic1.py` . Don't worry it it tells you that the program already exists – save it anyway.

Lighting up one LED

Let's start by lighting up the yellow LED. At the bottom of the program, add these lines:

```
GPIO.output(yellow, GPIO.HIGH)
time.sleep(2)
GPIO.output(yellow, GPIO.LOW)
```

HIGH turns a light on. LOW turns it off.

Save the program as `traffic2.py` .

Now leave the Python window open, and start a terminal session by going to the main desktop Menu - Accessories - Terminal. You should see a black window open.

In the terminal, type:

```
cd python/traffic
sudo python traffic2.py
```

You should see the yellow LED light up for two seconds, and then the LED will turn off and the program will end. You can repeat the program by entering `sudo python traffic2.py` again.

Try changing the program to light up the red or green LED.

Flashing an LED on and off repeatedly

Now let's change the program so that the last lines read:

```
while (True):
  GPIO.output(yellow, GPIO.HIGH)
  time.sleep(2)
  GPIO.output(yellow, GPIO.LOW)
  time.sleep(2)
```

Save this program as `traffic3.py` . Go back to the black terminal window and run it with:

```
sudo python traffic3.py
```

The yellow LED should flash on and off every 2 seconds.

Exit the program by holding down the CTRL key and tapping the C key. This may leave the yellow light on! Don't worry about that.

The full sequence

The full sequence for British traffic lights is:

Red (stop) – for a long time
Red and Yellow together (get ready to go) – for a short time
Green (go) – for a long time
Yellow on its own (get ready to stop) – for a short time

Let's change our program to do this full sequence:

```
while (True):

  GPIO.output(red, GPIO.HIGH)
  time.sleep(5)

  GPIO.output(yellow, GPIO.HIGH)
  time.sleep(2)

  GPIO.output(red, GPIO.LOW)
  GPIO.output(yellow, GPIO.LOW)
  GPIO.output(green, GPIO.HIGH)
  time.sleep(5)

  GPIO.output(green, GPIO.LOW)
  GPIO.output(yellow, GPIO.HIGH)
  time.sleep(2)

  GPIO.output(yellow, GPIO.LOW)
```

Notice how we leave the red light on whilst we turn on the yellow for the first time, so that we get red and yellow together.

That's it! Save this program as `traffic4.py` . Go back to the terminal window and run it with:

```
sudo python traffic4.py
```

Exit the program by holding down the CTRL key and tapping the C key.

You can use the `alloff.py` program to turn all the lights off after you've stopped the program. Also there is the `traffic5.py` program which uses some advanced techniques to turn off the lights when the program is stopped.

Further adventures

Here's some ideas for other things you could do:

- Run two sets of lights from one Raspberry Pi

You could use pins 12, 16 and 18 (plus pin 14 for ground) to control another set of lights.

- For a T-junction you need 3 sets of lights. For a crossroads, 4 sets.

On a Model B+ or Pi 2 you could use pins 36, 38 and 40 (plus pin 34 for ground) to control a third set of lights, or you could use two Raspberry Pis.

If you had more than one Raspberry Pi, how could you synchronise the lights? How could two Raspberry Pis communicate between each other?

- How could you detect a toy car waiting at the lights?

Does your strategy work if it is a plastic LEGO® car or a metal toy car, or both?

Does it matter if the car is heavy or light?

What about if it was a bicycle or a horse? How do traffic lights in the real world detect cycles and horses – do they have another strategy, or do they ignore that problem?

- How could you run your program automatically when you turn on your Raspberry Pi?

Try searching for information on the crontab @reboot command.

Preparation

The files for this tutorial can be found at:

http://www.cotswoldjam.org/downloads/2015-09/

Please create a `python` folder under the home/pi directory, then unzip the files into that `python` directory.

GPIO Numbers

Raspberry Pi B
Rev 1 P1 GPIO Header

	Pin No.		
3.3V	1	2	5V
GPIO0	3	4	5V
GPIO1	5	6	GND
GPIO4	7	8	GPIO14
GND	9	10	GPIO15
GPIO17	11	12	GPIO18
GPIO21	13	14	GND
GPIO22	15	16	GPIO23
3.3V	17	18	GPIO24
GPIO10	19	20	GND
GPIO9	21	22	GPIO25
GPIO11	23	24	GPIO8
GND	25	26	GPIO7

Raspberry Pi A/B
Rev 2 P1 GPIO Header

	Pin No.		
3.3V	1	2	5V
GPIO2	3	4	5V
GPIO3	5	6	GND
GPIO4	7	8	GPIO14
GND	9	10	GPIO15
GPIO17	11	12	GPIO18
GPIO27	13	14	GND
GPIO22	15	16	GPIO23
3.3V	17	18	GPIO24
GPIO10	19	20	GND
GPIO9	21	22	GPIO25
GPIO11	23	24	GPIO8
GND	25	26	GPIO7

Raspberry Pi B+
B+ J8 GPIO Header

	Pin No.		
3.3V	1	2	5V
GPIO2	3	4	5V
GPIO3	5	6	GND
GPIO4	7	8	GPIO14
GND	9	10	GPIO15
GPIO17	11	12	GPIO18
GPIO27	13	14	GND
GPIO22	15	16	GPIO23
3.3V	17	18	GPIO24
GPIO10	19	20	GND
GPIO9	21	22	GPIO25
GPIO11	23	24	GPIO8
GND	25	26	GPIO7
DNC	27	28	DNC
GPIO5	29	30	GND
GPIO6	31	32	GPIO12
GPIO13	33	34	GND
GPIO19	35	36	GPIO16
GPIO26	37	38	GPIO20
GND	39	40	GPIO21

Key

Power +	UART
GND	SPI
I²C	GPIO

This useful GPIO guide is courtesy of Alex Eames http://raspi.tv/rpi-gpio