Raspberry Pi LEGO® Christmas Tree LED Lights

Tutorial by Andrew Oakley - Public Domain
Nov 2017 www.cotswoldjam.org

LEGO® is a trademark of the LEGO Group of companies which does not sponsor, authorise nor endorse this tutorial nor Cotswold Raspberry Jam.

Components

A Light Emitting Diode (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). You'll need one red, one blue and one yellow LED.

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.

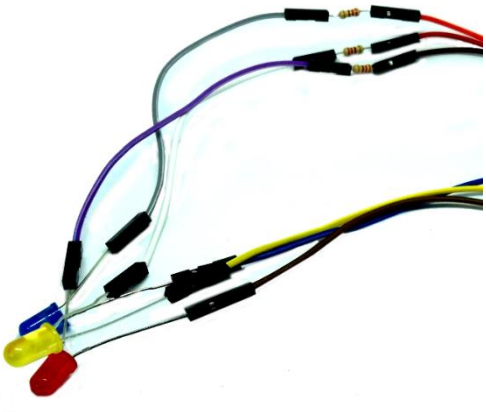
The resistor can be connected any way around. Lower value (lower ohm) resistors will allow more current through and will make the LED brighter; higher values will let less current through and make it dimmer. We're using a 270 ohm resistor but anything between 220-470 ohms will work. You'll need three resistors.

You will also need six 10cm female-to-female jumper wires, and three 20cm female-to-female jumper wires.

Connecting the LEDs and resistors

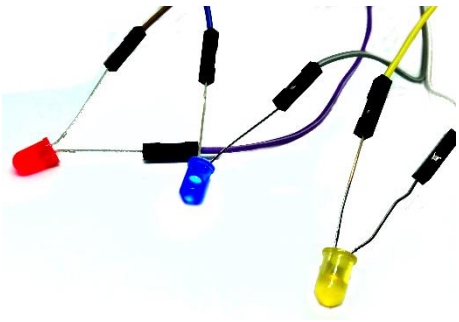
Take two short (10cm) female-to-female wires and place a resistor between them.

Do this three times so you have three long wires with a resistor in the middle.

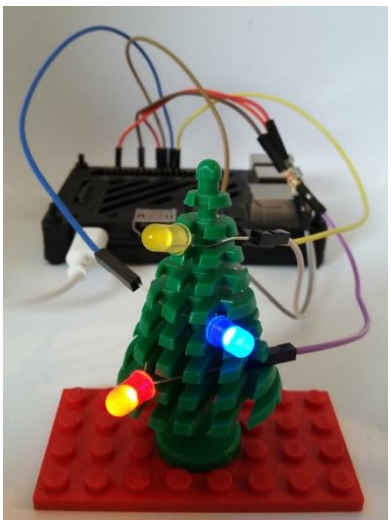


Next, attach the short resistor wires that you just made, to the negative (short) leg of each LED.

Finally, attach the long (20cm) female-female wires to the positive (long) leg of each LED.

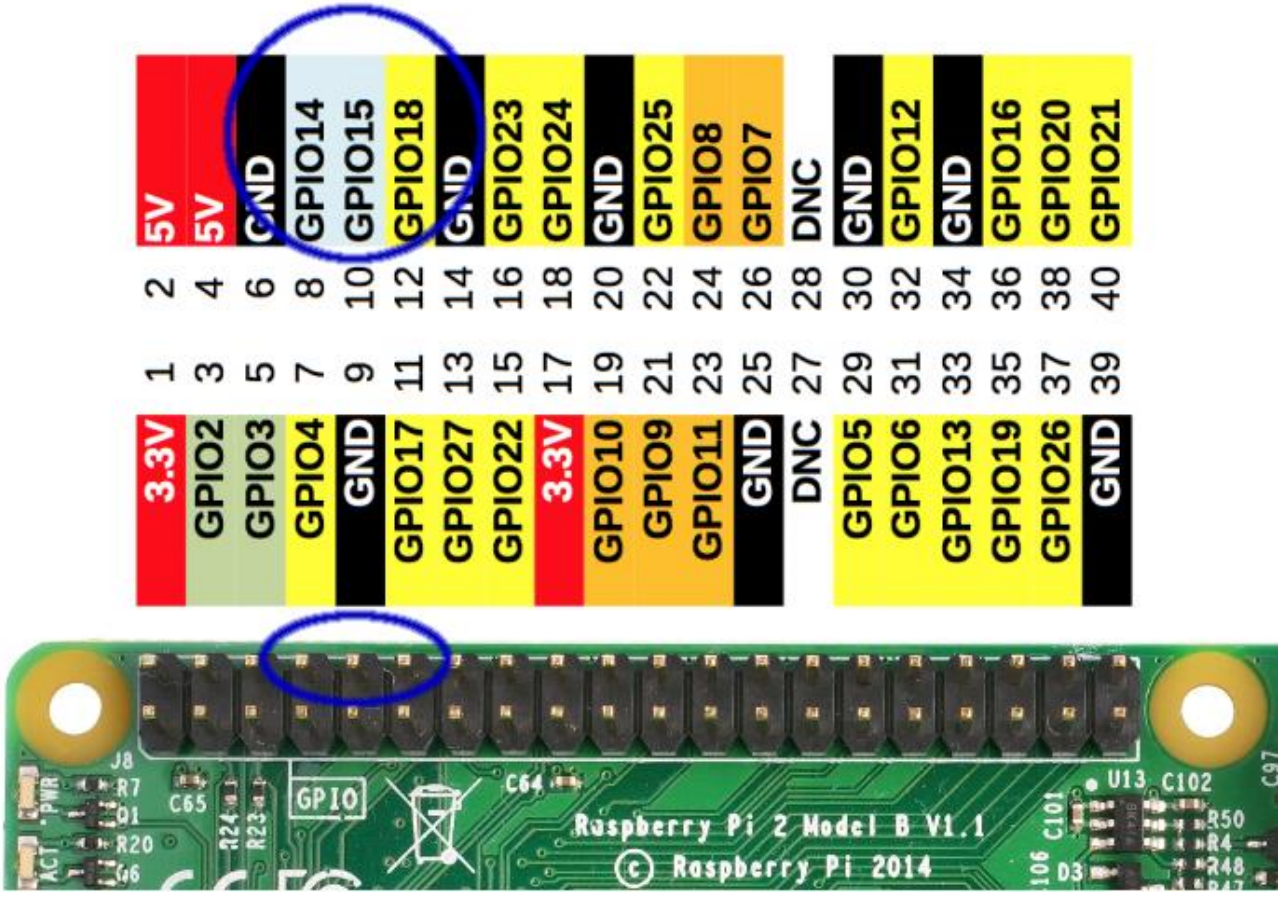


Bend the legs of the LEDs so that you can fit them around the LEGO fir tree.



Connecting to the Raspberry Pi

Connect the wires with resistors to any of the GND (Ground, Negative) pins. These are shown in black on the diagram below.



Next, connect the positive wires - the long wires without resistors - to the following pins. They are circled in blue in the diagram above.

Red LED - GPIO 14

Blue LED - GPIO 15

Yellow LED - GPIO 18

Don't worry about the colours of the wires. Concentrate on the colours of the LEDs.

GPIO stands for General Purpose Input Output. These pins have many uses.

The program

On your Raspberry Pi desktop, click the Raspberry menu (top right), then select **Programming** and click on **Python 3 (IDLE).**

Click **File** menu, **Open**. Double-click the **python** folder (you may need to scroll right to find it).

Double-click the **lego-xmas** folder.

Select **lego-xmas1.py** and click **Open**.

A new window will open. In the new window, select **Run** menu and click **Run Module**.

The red LED should light up briefly, and then turn off. Let's have a look at this program.

```
from gpiozero import LED
from time import sleep

red=LED(14)
red.on()
sleep(0.5)
```

What is this program doing?

```
from gpiozero import LED
from time import sleep
```

The computer doesn't know how to do everything itself; it has to learn. These two lines teach the computer about LEDs and waiting (sleep). The places where these lessons are kept, are called Libraries. We import the LED command from the gpiozero library, and the sleep command from the time library.

```
red=LED(14)
red.on()
```

We have to tell the computer which GPIO pin the red LED is attached to, and then we tell it to turn the LED on.

```
sleep(0.5)
```

This tells the computer to wait (sleep) for half a second.

When the program ends, it turns off the LEDs automatically. This auto-off feature is a part of the gpiozero library.

Lighting all the LEDs in sequence

Close the program by clicking on the **lego-xmas1.py** window, then select **File** menu, **Close**.

From the main window, open the **lego-xmas2.py** program, then run it with **Run** menu, **Run Module**.

This program lights all the LEDs one by one, waiting half a second between each.

Stop the program by holding down the **CTRL key** and pressing **C**

What is this program doing?

```
from gpiozero import LED
from time import sleep

red=LED(14)
blue=LED(15)
yellow=LED(18)

while True:
  blue.off()
  yellow.off()
  red.on()
  sleep(0.5)

  red.off()
  blue.on()
  sleep(0.5)

  blue.off()
  yellow.on()
  sleep(0.5)
```

Notice how we define the pin numbers for the red, blue and yellow LEDs, and we switch them on and off in order.

```
while True:
```

The **while** command repeats a section. We say **while True** to mean "forever".

Notice how the commands beneath the **while** command are indented by a couple of spaces from the left. This is called a block. The indentation lets the computer know which commands it should repeat as part of the **while** loop.

Also notice that **while True:** ends with a colon : . The colon lets the computer know that it should expect a block next.

Try changing the sequence. Swap some of the **on()** and **off()** commands. Change the **sleep** length to longer or shorter. Try adding extra **on()**, **off()** and **sleep** commands for different LEDs.

Make your own amazing Yuletide light sequence!

Advanced extras

The program **lego-xmas3.py** shows you how to use the **pulse** command. Notice how lego-xmas3.py uses the **PWMLED** library rather than just the simple **LED** library.

```
from gpiozero import PWMLED
```

The **pulse** command is more difficult to use, because you have to time the pulses of your different coloured LEDs so that they match up with your **sleep** time!

```
  red.pulse(0.3,0.3,4,True)
  blue.pulse(0.4,0.4,3,True)
  yellow.pulse(0.6,0.6,2,True)
  sleep(2.4)
```

The program contains some comments which explain how the pulse command works.

Teacher Preparation

Downloads available from http://www.cotswoldjam.org/downloads/2017-11

To add the lego-xmas*.py program files to Raspbian, go to the terminal:


```
cd
mkdir -p python
cd python
curl "http://www.cotswoldjam.org/downloads/2017-11/lego-xmas.zip" -O
unzip lego-xmas.zip
exit
```

Note that -O is the capital letter O, not zero.

More tutorials available from http://www.cotswoldjam.org/tutorials