CamJam EduKit - Worksheet Two

Project Controlling LEDs with Python

Description In this project, you will learn how to connect and control LEDs (Light Emitting Diode) with the Raspberry Pi.

Equipment Required

For this worksheet, you will require:

- Your Raspberry Pi
- 400 Point Breadboard
- 1 x Red LED
- 1 x Yellow LED
- 1 x Green LED
- 3 x 330Ω Resistors
- 4 x M/F jumper wires

The Parts

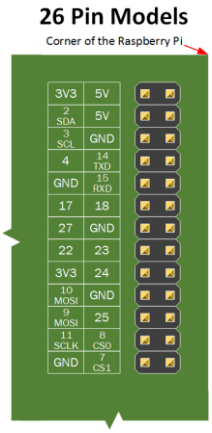
In this first circuit, you will be connecting three LEDs to the GPIO header of your Raspberry Pi and using Python to turn the LEDs on and off.

It is important that you read this section, as you need to understand the Raspberry Pi GPIO pins, how the holes in the breadboard are connected together, and which leg of the LED is which.

Before you build the circuit, let us look at the parts you are going to use.

Raspberry Pi GPIO Pins

First, lets look at the Raspberry Pi's '**GPIO**' pins. GPIO stands for **General Purpose Input Output**. It is a way the Raspberry Pi can control and monitor the outside world when connected to electronic circuits. The Pi is able to control LEDs, turning them on or off, or motors, or many other things. It is also able to detect whether a switch has been pressed, or what the temperature is, or whether there is light. In this CamJam EduKit you will learn to control LEDs and a buzzer, and detect when a button has been pressed.

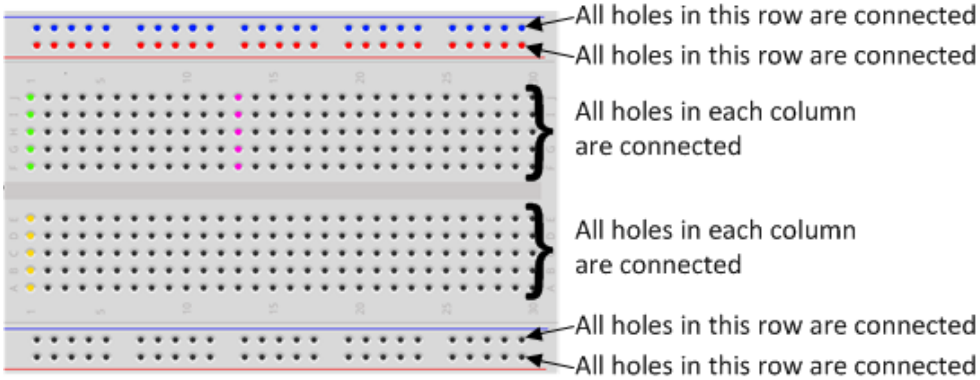


The diagram on the left shows the pin layout for all the Raspberry Pi Models built for the last few years; they have 40 GPIO pins. The original Raspberry Pi models A and B only had 26 pins, as shown in the diagram on the right; they are layed out exactly the same as the top 13 rows on the current Pi's.

Some pins have different functions. There are pins that provide power at 5 volts and 3.3 volts, ground pins (0 volts), input/output pins and some pins that interface to external circuits in more complex ways. You are just going to use the basic GPIO, 3.3v and Ground pins in these worksheets.

The Breadboard

The breadboard is a way of connecting electronic components to each other without having to solder them together. They are often used to test a circuit design before creating a Printed Circuit Board (PCB).

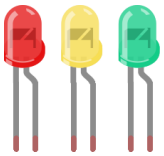


The holes on the breadboard are connected in a pattern.

With the breadboard in the CamJam EduKit, the top row of holes are all connected together – marked with blue dots. And so are the second row of holes – marked with red dots. The same goes for the two rows of holes at the bottom of the breadboard.

In the middle, the columns of holes are connected together with a break in the middle. So, for example, all the green holes marked are connected together, but they are not connected to the yellow holes, nor the purple ones. Therefore, any wire you poke into the green holes will be connected to other wires poked into the other green holes.

The LEDs



Three LEDs are supplied in the EduKit – one red, one yellow, and one green. LED stands for Light Emitting Diode. An LED glows when electricity is passed through it.

When you pick up the LED, you will notice that one leg is longer than the other. The longer leg (known as the 'anode'), is always connected to the positive supply of the power supply. The shorter leg (known as the 'cathode') is connected to the negative side of the power supply, known as 'ground'.

LEDs will only work if power is supplied the correct way round (i.e. if the 'polarity' is correct). You will not break the LEDs if you connect them the wrong way round – they will just not light. If you find that they do not light in your circuit, it may be because they have been connected the wrong way round.

The Resistors

Resistors are a way of limiting the amount of electricity going through a circuit; specifically, they limit the amount of 'current' that is allowed to flow. The measure of resistance is called the Ohm (Ω), and the larger the resistance, the more it limits the current. The value of a resistor is marked with coloured bands along the length of the resistor body.

The EduKit is supplied with two sets of resistors. There are three 330Ω resistors and one 4.7kΩ (or 4700Ω) resistor. In the LED circuit, you will be using the three 330Ω resistors. You can identify the 330Ω resistors by the colour bands along the body. The colour coding will depend on how many bands are on the resistors supplied:

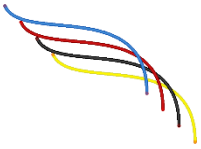
- If there are four colour bands, they will be Orange, Orange, Brown, and then Gold.
- If there are five bands, then the colours will be Orange, Orange, Black, Black, Brown.

You have to use resistors to connect LEDs up to the GPIO pins of the Raspberry Pi. The Raspberry Pi can only supply a small current (about 60mA). The LEDs will want to draw more, and if allowed to they will burn out

the Raspberry Pi. Therefore, putting the resistors in the circuit will ensure that only this small current will flow and the Pi will not be damaged.

It does not matter which way round you connect the resistors. Current flows in both ways through them.

The Jumper Wires



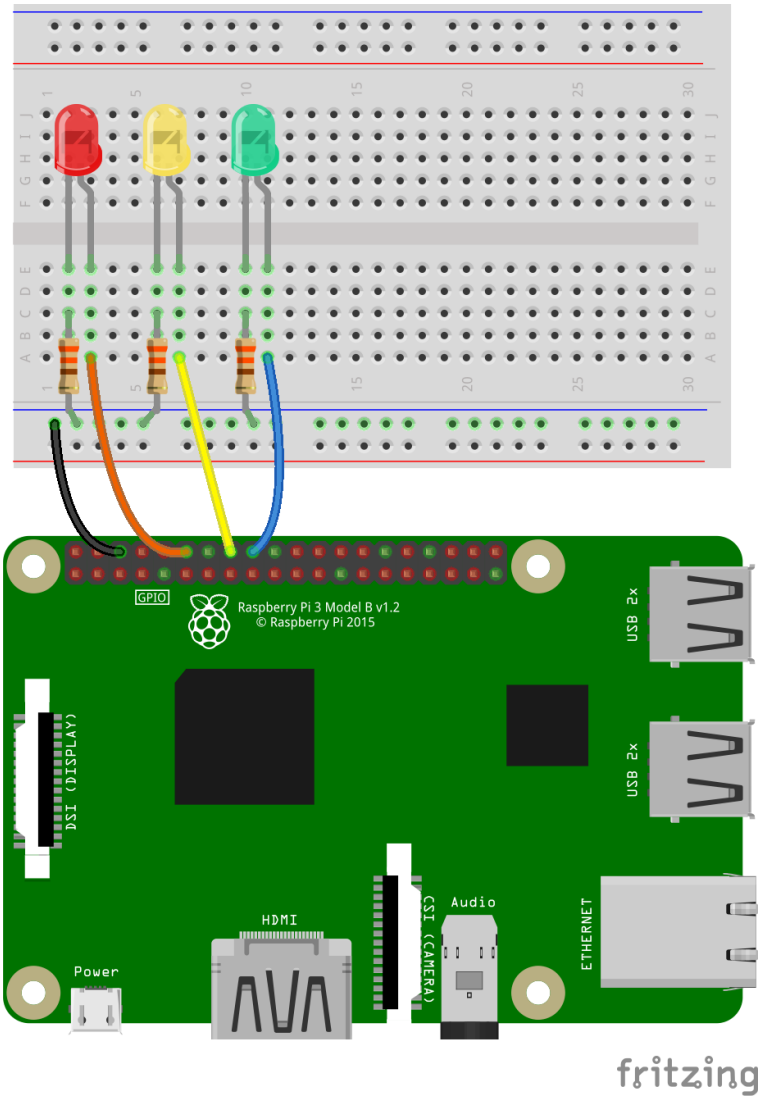
Jumper wires are used on breadboards to 'jump' from one connection to another. The ones you will be using in this circuit have different connectors on each end. Then end with the 'pin' will go into the breadboard, and is known as the 'male' end. The end with the piece of plastic with a hole in it will go onto the Raspberry Pi's GPIO pins. This is the 'female' end.

The colour of jumper wires supplied in the EduKit will vary, and are unlikely to match the colours used in the diagrams.

Building the Circuit

While you can build the circuit with the Pi turned on, it is best to turn it off at this stage.

You will be using one of the 'ground' (GND) pins to act like the 'negative' or 0 volt end of a battery.



The 'positive' ends of the battery will be provided by three of the other GPIO pins, one for each of the three LEDs. You will be using the pins marked 18, 23 and 24 for the Red, Yellow and Green LEDs respectively.

When they are 'taken high', which means they output 3.3 volts, the LEDs will light.

Now take a look at the circuit diagram on the left.

The power for each LED will be provided by the Pi, from GPIO pins 18, 23 and 24. You can control them from Python, meaning you can make the GPIO pins supply either 0 volts (off) or 3.3 volts (on).

There are in fact three separate circuits in the diagram. Each one consists of the power supply (the Pi), an LED that lights when the power is applied, and a resistor to limit the current that can flow through the circuit.

Each circuit is going to share a 'common ground rail'. In other words, you will be connecting all of the circuits to the same 'ground' (0 volts) pin of the Raspberry Pi. You are going to use the second row up from the bottom of the breadboard. Remember that the holes on the two top and two bottom rows are all connected together? So, connect one of the Jumper wires from the third pin from the left on the top row of the Pi to the second row up of the breadboard, as shown in the diagram (the black wire).

Next push three LEDs legs into the breadboard, with the long leg on the right as shown in the circuit diagram.

Then connect the three 330Ω resistors between the 'common ground rail' and the left leg of the LEDs. You will need to bend the legs of each of the resistors to fit, but please make sure that the wires of each leg do not touch one another.

Lastly, using three Jumper wires, complete the circuit by connecting pins 18, 23 and 24 to the right-hand leg of each LED. These are shown here with the orange, yellow, and blue wires.

You are now ready to write some code to switch the LEDs on.

Code

Follow the instructions in Worksheet One to turn on your Pi and open the IDLE3 Python editor. Create a new file by going to the File menu item and selecting New File. Type in the following code:

```
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(18, GPIO.OUT)
GPIO.setup(23, GPIO.OUT)
GPIO.setup(24, GPIO.OUT)
print("Lights on")
GPIO.output(18, GPIO.HIGH)
GPIO.output(23, GPIO.HIGH)
GPIO.output(24, GPIO.HIGH)
```

Once you have typed all the code and checked it, save the file in the EduKit directory and call it `2-on.py`.

You now need to repeat most of the above to create the second file which will be used to turn the LEDs off. Save the file again, calling it `2-off.py`, and edit it to match the following code.

```
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(18, GPIO.OUT)
GPIO.setup(23, GPIO.OUT)
GPIO.setup(24, GPIO.OUT)
print("Lights off")
GPIO.output(18, GPIO.LOW)
GPIO.output(23, GPIO.LOW)
GPIO.output(24, GPIO.LOW)
GPIO.cleanup()
```

Save the file again.

So, what is happening in the code? Let's go through it a section at a time, taking "2-on.py" as an example:

```
import RPi.GPIO as GPIO
```

The first line tells the Python interpreter (the thing that runs the Python code) that it will be using a 'library' that will tell it how to work with the Raspberry Pi's GPIO pins. A 'library' gives a programming language extra commands that can be used to do something different that it previously did not know how to do. This is like adding a new channel to your TV so you can watch something different.

```
GPIO.setmode(GPIO.BCM)
```

Each pin on the Pi has several different names, so you need to tell the program which naming convention is to be used.

```
GPIO.setwarnings(False)
```

This tells Python not to print GPIO warning messages to the screen.

```
GPIO.setup(18, GPIO.OUT)
GPIO.setup(23, GPIO.OUT)
GPIO.setup(24, GPIO.OUT)
```

These three lines are telling the Python interpreter that pins 18, 23 and 24 are going to be used for outputting information, which means you are going to be able to turn the pins 'on' and 'off'.

```
print("Lights on")
```

This line prints some information to the terminal.

```
GPIO.output(18, GPIO.HIGH)
GPIO.output(23, GPIO.HIGH)
GPIO.output(24, GPIO.HIGH)
```

These three lines turn the GPIO pins 'on'. What this actually means is that these three pins are made to provide power of 3.3volts. This is enough to turn the LEDs in our circuit on.

To turn the LEDs off, you need to replace the GPIO.HIGH with GPIO.LOW. This will turn the pins off so that they no longer supply any voltage.

Then there's the extra line in 2-off.py. The `GPIO.cleanup()` command at the end is necessary to reset the status of any GPIO pins when you exit the program. If you don't use this, then the GPIO pins will remain at whatever state they were last set to. If you put this in `2-on.py` the the pins would immediately be turned off and you would not see them light.

Running the Code

You are now ready to run the code. Open `2-on.py` and run it by selecting the Run Module menu option, under the Run menu item. Alternatively, you can just press the F5 key. You should see your LEDs light.

To turn the LEDs off, open `2-off.py` and run the code.

If you find that the code does not run correctly there may be an error in the code you have typed. You should check and re-edit the code, save again and re-run it.

What you are doing here is telling the Python interpreter to run the commands in file "2-on.py" or "2-off.py".

Note

Do not disassemble this circuit as it will be used in the following worksheets.