

CamJam EduKit Sensors Worksheet Two

Project Controlling LEDs and a Buzzer with Python

Overview This project will demonstrate how to connect and control Light Emitting Diodes (LEDs) and a buzzer using a Raspberry Pi. These components will be used in later worksheets too.

Equipment Required

For this worksheet, you will need:

- Your Raspberry Pi
- A breadboard
- One red LED and one Blue LED
- One buzzer
- One M/M jumper wires
- Four M/F jumper wires
- Two 330Ω Resistors

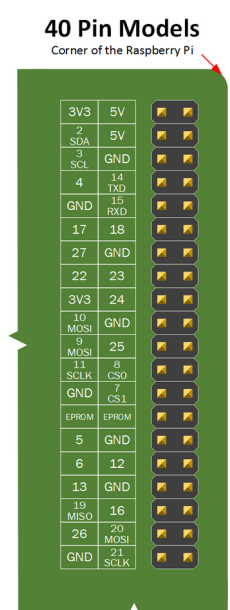
The Parts

In this initial circuit, you will connect two LEDs and a buzzer to the GPIO header of your Raspberry Pi, utilising Python to control the LEDs and activate the buzzer.

It is essential to review this section thoroughly to ensure a clear understanding of the Raspberry Pi GPIO pins, the connections within the breadboard, and the pin configurations of the LEDs and buzzer.

Before assembling the circuit, let us examine the components involved.

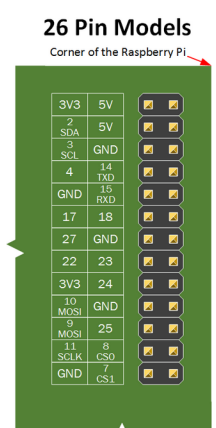
Raspberry Pi General Purpose Input/Output (GPIO) Pins



First, let us examine the Raspberry Pi's GPIO pins. GPIO stands for General Purpose Input/Output. These pins enable the Raspberry Pi to interface with and monitor external electronic components. Using GPIO, the Raspberry Pi can control devices such as LEDs, motors, and more. It can also detect input signals, such as a pressed switch, temperature readings, or light levels. In this CamJam EduKit, you will learn how to control LEDs and a buzzer, as well as how to detect button presses.

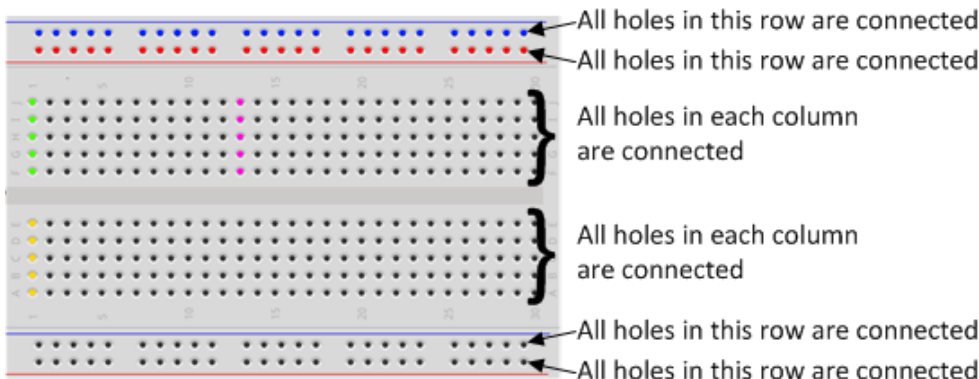
The diagram on the left illustrates the pin configuration for all Raspberry Pi models manufactured in recent years, featuring 40 GPIO pins. The original Raspberry Pi models A and B, depicted on the right, include 26 pins and share the same layout as the top 13 rows of current models.

It is important to note that some pins serve different functions. These include pins that supply power at 5 volts and 3.3 volts, ground pins (0 volts), input/output pins, and others that interface with external circuits in more complex ways. For the purposes of these worksheets, only the basic GPIO, 3.3V, and Ground pins will be used.



The Breadboard

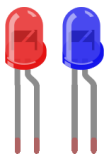
A breadboard provides a temporary platform for connecting electronic components without the need for soldering. It is commonly used for prototyping and testing circuit designs prior to designing and manufacturing a printed circuit board (PCB).



The holes on the breadboard are connected in specific patterns. The top row of holes (indicated by blue dots) are connected together, as are the second row of holes (indicated by red dots). The same connection pattern applies to the two rows of holes at the bottom of the breadboard. In

the central area, columns of holes are connected vertically, with a break in the middle. For example, all the green-marked holes are connected to each other, but are not connected to the yellow or purple holes. Therefore, any wire inserted into the green holes will be electrically connected to other wires inserted into the green holes.

The LEDs



Two large (10mm) LEDs are supplied in this EduKit – one red, one blue. LED stands for Light Emitting Diode, and 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 circuit. 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 around (i.e. if the 'polarity' is correct). You will not break the LEDs if you connect them the wrong way around – 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 around.

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.

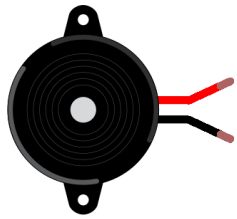
The EduKit is supplied with two sets of resistors. There are two 330Ω resistors and one $4.7k\Omega$ (or 4700Ω) resistor. In the LED circuit, you will be using the two 330Ω resistors. You can identify the 330Ω resistors by the colour bands along the body. The colour coding will depend on how many bands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

Buzzer

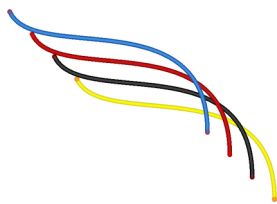


The buzzer supplied in the EduKit is an 'active' buzzer, which means that it only needs an electric current to make a noise. In this case, you are using the Raspberry Pi to supply that current.

The buzzer looks like the image on the right. It has positive and negative legs. The longer leg is positive (shown in red in the diagram), the shorter is negative (coloured black in the diagram).



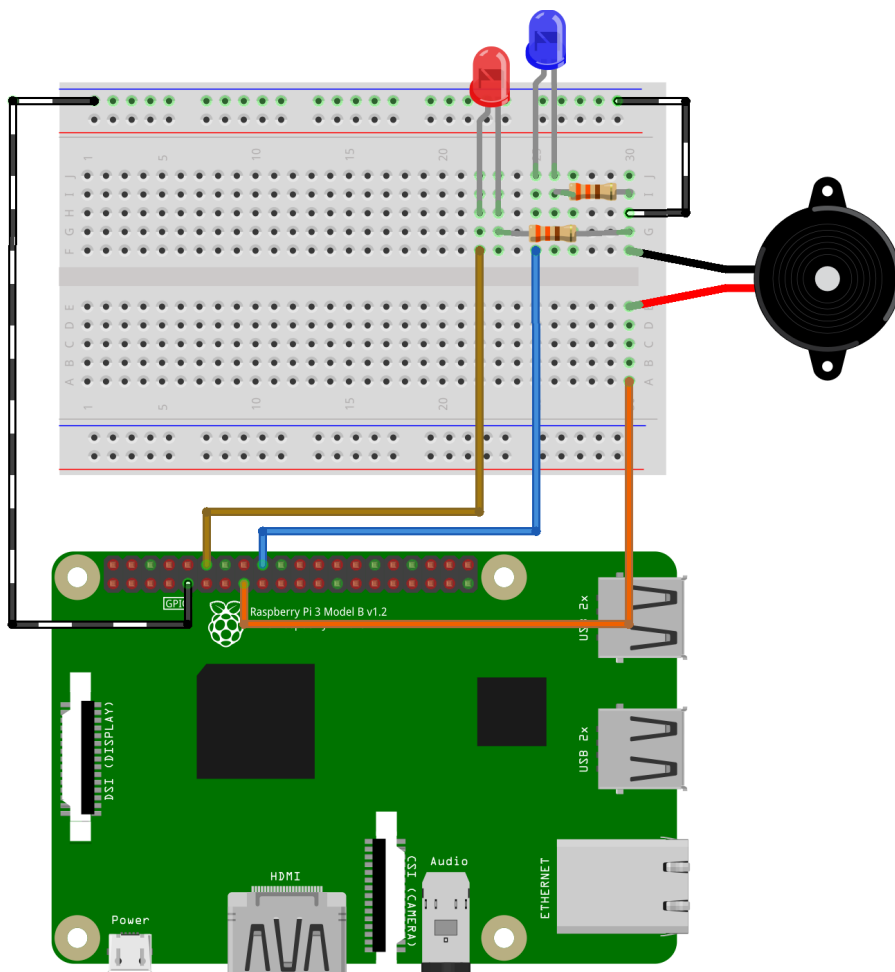
Jumper Wires



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. The end with the 'pin' will go into the Breadboard. The end with the piece of plastic with a hole in it will go onto the Raspberry Pi's GPIO pins. You will also use a jumper wire that has a pin on each end to connect the buzzer to the 'ground rail'.

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

Building the Circuit



While you could build the circuit with the Pi turned on, it is much safer to turn it off.

Look at the circuit diagram on the left.

Think of the Pi's power pins as a battery. You will be using one of the Pi's 'ground' (GND) pins to act like the 'negative' end of a battery, with the 'positive' end of the battery provided by three GPIO pins, one for each of the two LEDs and one for the buzzer. You will be using GPIO pins 18, 24 and 22 for the Red LED, Blue LED and buzzer respectively (see the GPIO pin diagram on page one).

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

There are in fact three separate circuits in the diagram:

- A resistor and the Red LED
- A resistor and the Blue LED
- The buzzer

fritzing

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 top row 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 a 'ground' pin to the top row of the breadboard, as shown by the long black/white wire in the diagram. Here we are using the fifth pin in on the bottom row.

Then connect the jumper wire that has a pin on each end between the top row and the last column of the breadboard, as shown by the short black/white wire in the diagram. This will be the 'ground' (0v) for the two LEDs and the buzzer.

Next, push the buzzer into the breadboard with the buzzer itself straddling the centre of the board. The buzzer is marked with a + for the positive leg – this leg needs to be in the bottom section of the board, with the negative leg in the same column as the black/white wire.

Push the LEDs legs into the breadboard, with the long leg on the left, as viewed in the circuit diagram.

Then connect the two 330Ω resistors between the 'ground' and the left leg of the LEDs; that is, the same column as the grey wire. 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 cross each other.

Lastly, using three Jumper wires, complete the circuit by connecting pins 18 and 24 to the right-hand legs of the LEDs, and pin 22 to the positive leg of the buzzer. These are shown in the diagram with the brown, blue and orange wires respectively.

You are now ready to write some code to switch the LEDs on and make the buzzer sound.

Note: When you connect the circuit up, you may find that the LEDs light up or that the buzzer sounds before running the program. This is usual, as the pins will be in a non-determined state before your program controls them.

Code

Follow the instructions in Worksheet One to turn on your Pi and open the Thonny Python editor. Remember to make sure it uses the EduKit2venv environment, as show in Worksheet 1.

Create a new file by going to the File menu item and selecting New File. Type in the following code:

```
# CamJam EduKit 2 - Sensors
# Worksheet 2 - LEDs and Buzzer

# Import Python libraries
import time
from gpiozero import LED, Buzzer

# Set up the LEDs and Buzzer
red = LED(18)
blue = LED(24)
buzzer = Buzzer(22)

print("Lights and sound on")
red.on()
blue.on()
buzzer.on()

# Pause for one second
time.sleep(1)
```

```
print("Lights and sound off")
red.off()
blue.off()
buzzer.off()
```

Save the code as `2-led-buzzer.py`.

Running the Code

You are now ready to run the code. Select the Run Module menu option, under the Run menu item. Alternatively, you can just press the F5 key.

How the Code Works

What is happening in the code? Let's go through it a section at a time:

```
from gpiozero import LED, Buzzer
import time
```

The first line tells the Python interpreter (the thing that runs the Python code) that it will be using a 'library' called `gpiozero` 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.

The 'time' library is used for time related commands.

```
# Set up the LEDs and Buzzer
red = LED(18)
blue = LED(24)
buzzer = Buzzer(22)
```

These three lines are telling the Python interpreter that pins 18, 24 and 22 are going to be used to control the red and blue LEDs, and the buzzer.

```
print("Lights and sound on")
```

This line prints some information to the terminal.

```
red.on()
blue.on()
buzzer.on()
```

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 on the LEDs and make the buzzer sound.

```
time.sleep(1)
```

Pauses the running of the code for one second.

```
print("Lights and sound off")
```

This line prints some more information to the terminal.

```
red.off()
blue.off()
buzzer.off()
```

This will turn the pins off so that they no longer supply any voltage.

Note

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