CamJam EduKit Sensors Worksheet Two

Project Controlling LEDs and a Buzzer with Python

Description In this project, you will learn how to connect and control LEDs (Light Emitting Diode) and a buzzer with the Raspberry Pi. These two sets of devices will be used in later worksheets.

Equipment Required

For this worksheet you will require:

- Your Raspberry Pi
- 400 Point Breadboard
- 1 x Red LED and 1 x Blue LED
- 1 x Buzzer
- 1 x M/M jumper wires
- 4 x M/F jumper wires
- 2 x 330Ω Resistors

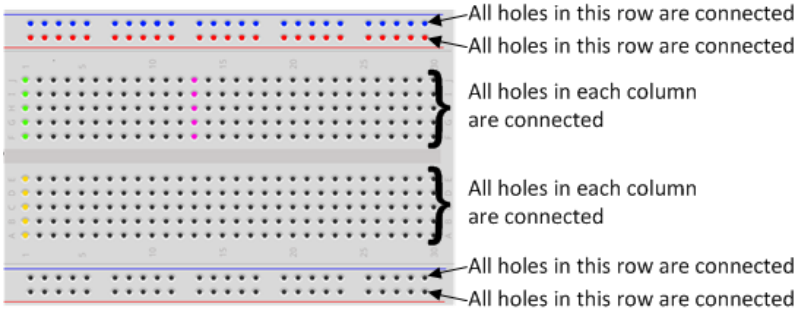
The Parts

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

It is important that you read this section, as you need to understand how the holes in the breadboard are connected together, and which legs of the LEDs and buzzer is which.

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

The Breadboard



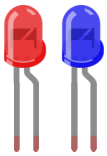
The breadboard is a way of connecting electronic components to each other without having to solder them together. They are often used to test a circuit design before creating a Printed Circuit Board (PCB).

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 red dots in the diagram and with a red line on the breadboard. And so are the second row of holes – marked with blue dots. The same goes for the two rows of holes at the bottom of the breadboard. These rows are usually used for ground (GND) and power (PWR), which means that the blue rows are connected to ground (0v) and red rows to the power supply (3.3v or 5v in this kit). In electronics, they are often referred to as the 'ground rail' and 'power rail'. These terms will be used in the EduKit worksheets. Components requiring power and ground are usually connected to these 'rails' and not directly to the power or ground on the Pi.

In the middle, the columns of wires are connected together with a break in the middle. Therefore, 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

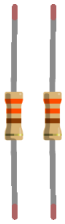


Two large (10mm) LEDs are supplied in this EduKit – one red, one blue. LED stands for Light Emitting Diode, and glows when electricity is passed through it.

When you pick up the LED, you will notice that one leg is longer than the other is. 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 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.

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.

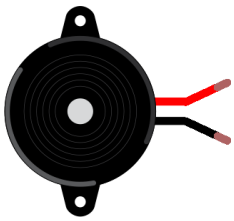
The EduKit is supplied with two sets of resistors. There are two 330Ω resistors and one 4.7kΩ (or 4700Ω) resistor. In the LED circuit, you will be using the two 330Ω resistors. You can identify the 330Ω resistors by the colour bands along the body. The colour coding will depend on how many bands are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

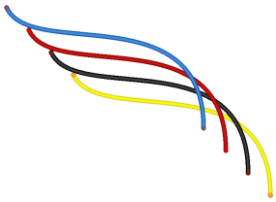
Buzzer



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

Building the Circuit

26 Pin Models

40 Pin Models

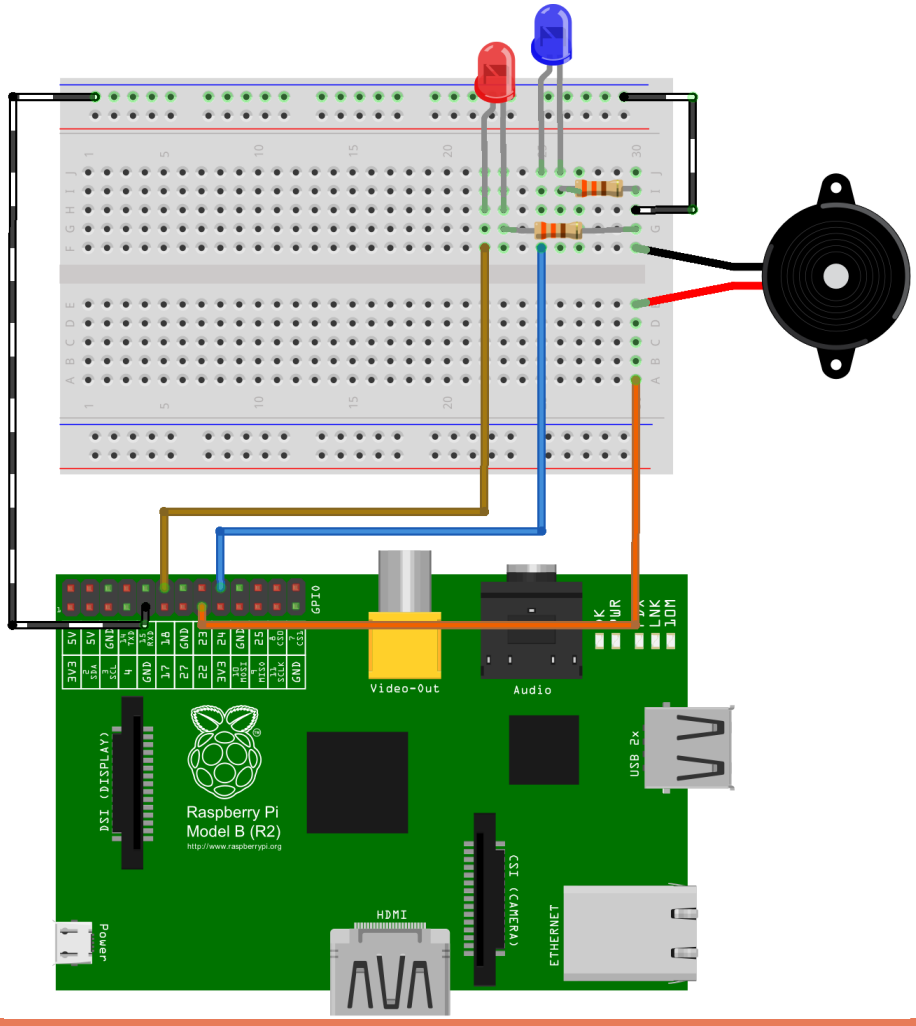
While you could build the circuit with the Pi turned on, it is much safer to turn it off.

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 by being 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 temperature, or light. In the CamJam Sensor EduKit you will learn to control LEDs and a buzzer, and detect temperature, light and movement.

The diagram on the left shows the pin layout for a Raspberry Pi Models A and B (Rev 2 - the original Rev 1 Pi is slightly different), looking at the Pi with the pins in the top right corner. The Raspberry Pi A+, B+ and Pi 2 share exactly the same layout of pins for the top 13 rows of GPIO pins. The pin layout for the A+, B+ and Pi2 is on the right.

Now take a look at the circuit diagram on the next page.

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 the pins marked 18, 24 and 22 for the Red LED, Blue LED and buzzer respectively.



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 resister and the Blue LED
- The buzzer

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 right, as shown 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 here 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.

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:

```
#CamJam Edukit 2 - Sensors
# Worksheet 2 – LEDs and Buzzer

# Import Python libraries
import RPi.GPIO as GPIO
import time

# Set the GPIO naming convention
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

# Set the three GPIO pins for Output
GPIO.setup(18, GPIO.OUT)
GPIO.setup(24, GPIO.OUT)
GPIO.setup(22, GPIO.OUT)

print "Lights and sound on"
GPIO.output(18, GPIO.HIGH)
GPIO.output(24, GPIO.HIGH)
GPIO.output(22, GPIO.HIGH)

# Pause for one second
time.sleep(1)

print "Lights and sound off"
GPIO.output(18, GPIO.LOW)
GPIO.output(24, GPIO.LOW)
GPIO.output(22, GPIO.LOW)
```

```
GPIO.cleanup()
```

Save the code as `2-LEDBuzz.py` into the `EduKitSensors` directory.

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

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

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

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.

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

```
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(24, GPIO.OUT)
GPIO.setup(22, GPIO.OUT)
```

These three lines are telling the Python interpreter that pins 18, 24 and 22 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 and sound on")
```

This line prints some information to the terminal.

```
GPIO.output(18, GPIO.HIGH)
GPIO.output(24, GPIO.HIGH)
GPIO.output(22, 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 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.

```
GPIO.output(18, GPIO.LOW)
GPIO.output(24, GPIO.LOW)
GPIO.output(22, GPIO.LOW)
```

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.

```
GPIO.cleanup()
```

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.

Note

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