9 – Output Devices: Buzzers

Project

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

Components

In addition to your Raspberry Pi, you will need:

- A breadboard
- Two LEDs with associated current limiting resistors
- Four female-to-male jumper wires to connect your breadboard to the Raspberry Pi’s GPIO
- One male-to-male jumper wires for making connections on the breadboard

Buzzer

The buzzer supplied 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 one the images above and has positive and negative legs. The longer/red leg is positive, the shorter/black the negative.
Making the connections
Please note that this experiment references the GPIO pins on the Raspberry Pi differently than we have done in previous experiments. Previously, we have identified GPIO pins by their physical number; pin 1 being in the top left corner and pin 26 being in the bottom right corner of the GPIO connector.

This experiment references the pins by their ‘label’; GPIO 18 (physical pin 12), GPIO 22 (physical pin 15) and GPIO 24 (physical pin 18) – see diagram below for clarification.

![Diagram of GPIO pins on Raspberry Pi]

We have introduced this method of identifying the GPIO pins because articles you may read in future may use either method and you should be aware of both.

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

First, let’s 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 this experiment you will learn to control LEDs and a buzzer, and detect light.

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 GPIO18, GPIO24 and GPIO22 for the LEDs 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 resistor 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 on the Raspberry Pi. You are going to use the top row of the breadboard.
Remember that the holes on the top and two bottom rows are all connected together? So, connect one of the Jumper wires from ground pin on the Pi (physical pin 9) to the top row of the breadboard, as shown by the long grey wire in the diagram.

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 grey 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 grey 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 GPIO18 (physical pin 12) and GPIO24 (physical pin 18) to the right hand legs of the LEDs, and pin GPIO22 (physical pin 18) 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 (please see next page).
The code
Create a new program called 2-ledbuzz.py and type the following code in to it...

```python
#Load 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()
```

Now run your program by typing...

```
sudo python 2-ledbuzz.py
```
### How the code works

So, what is happening in the code? Let’s go through it a section at a time...

```python
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.

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

There are different ways of referring to the GPIO pins on a Raspberry Pi; either by their physical pin number, or their label.

Previously, we have used the physical pin number, but in this example we’re referring to the pins by their label, hence the ‘BCM’ in this instruction as opposed to the ‘BOARD’ we have used previously.

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

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

```python
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’.

```python
print "Lights and sound on"
```

This line prints some information to the terminal.

```python
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.

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

Pauses the running of the code for one second.

```python
print "Lights and sound off"
```

This line prints some more information to the terminal.
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.

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.

**Acknowledgments**
This worksheet is a modified version of one that was first published on the Cambridge Raspberry Jam
web site¹.