CamJam EduKit Worksheet Six

Project Morse code SOS using a Buzzer

Description In this project, you will learn how to wire and program a buzzer, and use it to produce Morse code.

You will be using 'user-defined functions'.

NOTE: This worksheet can use a Raspberry Pi Model A, B or B+. The first 26 pins on the B+ (when looking at the B+ with the pins in the top left) are exactly the same as the Model A and Model B.

Equipment Required

The circuit built in CamJam EduKit Worksheet Five, plus the following:

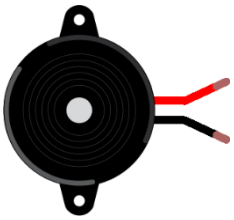
□ Buzzer □ 1 x M/F Jumper Wires □ 1 x M/M Jumper Wire

Additional Parts

You will be adding a buzzer to the LED and switch circuit that you made in the CamJam EduKit Workshop Five. Let us look at the additional components.

Do not skip this section as you need to know how to connect up the buzzer.

Buzzer



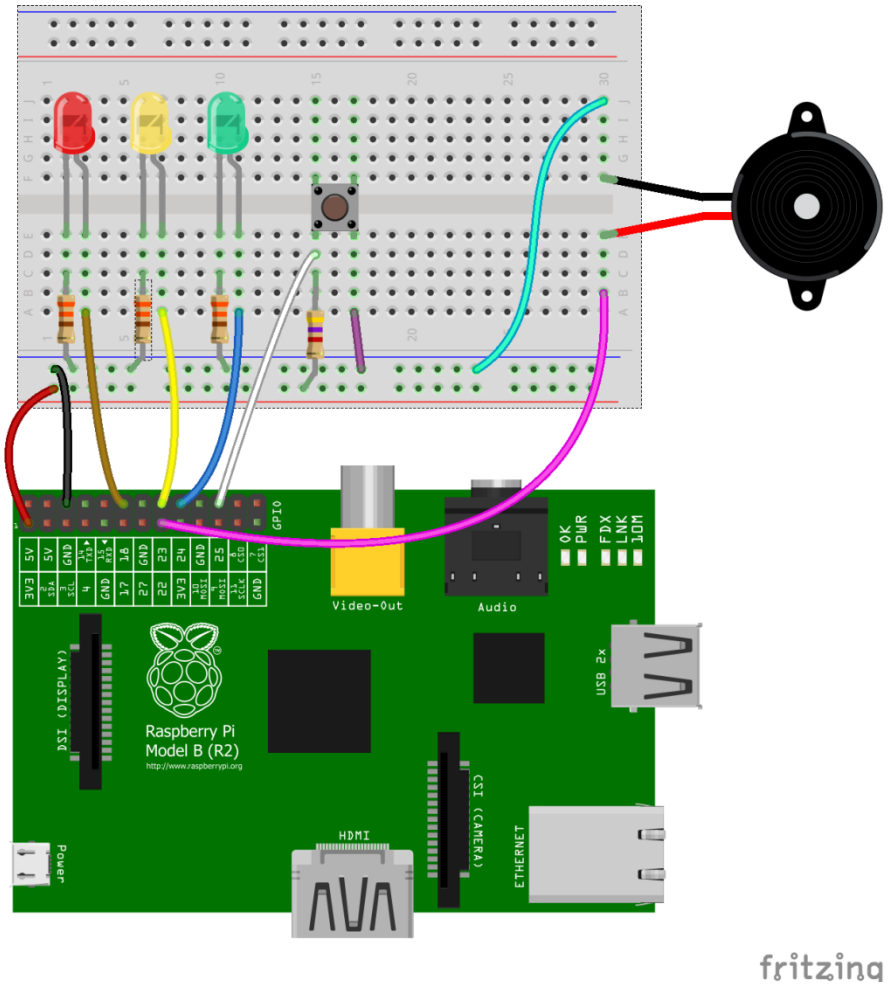
The buzzer supplied in the EduKit is an 'active' buzzer, which means that it only needs an electric current to make a noise. In this case, you are using the Raspberry Pi to supply that current.

The buzzer has positive and negative legs. The longer leg is positive (shown in red in the diagram), the shorter the negative (coloured black).

Building the Circuit

Before you connect additional components to your circuit, you should turn off your Pi.

Leave the LED and switch circuit from Worksheet 5 in place.



Place the buzzer on the breadboard straddling the middle divide. The longer leg should be connected via a jumper wire to pin 22.

The other leg should be connected to the ground rail.

Pin 22 will be an output pin, and when it is set on, the buzzer will sound.

Code

You are going to be using 'user-defined functions' in the code below. These are pieces of code that you may want to run more than once, but by using functions you only have to write them once. You then 'call' that function from within your code each time you want to run it.

To define a function, you first need to tell Python that you are writing a function. For this, use 'def' command followed by the name of the function (you choose that name), followed by brackets and a colon (:):

```
def hello():
```

Everything indented after this line will be included in the function, for example:

```
    Print "Hello World!"
```

To use the function, you must 'call' it by simply using then name of your new function:

```
hello()
```

Code

Now, every time Python sees 'hello ()' in you code, it will print 'Hello World!'

The code will be saved to the EduKit directory. Follow the instructions below.

1. Change to the EduKit directory:
```
cd ~/EduKit/
```
2. Create a new file by typing:
```
nano 6-morsecode.py
```
3. Type in the following code below exactly as seen, including the important indents.
```
# Import Libraries
import os
import time
import RPi.GPIO as GPIO

# Set the GPIO pin naming mode
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
PINBuzzer = 22 #Sets the buzzer pin 22

# Sets PINBuzzer as an output pin and initialise it to 'off'
GPIO.setup(PINBuzzer,GPIO.OUT,initial=0)

# Define some 'user-defined functions'
def dot(): #A single morse dot
    GPIO.output(PINBuzzer,GPIO.HIGH)
    time.sleep(.1)
    GPIO.output(PINBuzzer,GPIO.LOW)
    time.sleep(.1)

def dash(): #A single morse dash
    GPIO.output(PINBuzzer,GPIO.HIGH)
    time.sleep(.3)
    GPIO.output(PINBuzzer,GPIO.LOW)
    time.sleep(.1)

def letterSpace(): #The space between letters
    time.sleep(.2)

def wordSpace(): #The space between words
    time.sleep(.6)

def morseS(): #The Morse for S, ...
    dot()
    dot()
    dot()

def morseO(): #The Morse for O, ---
    dash()
```

Code

```
    dash()
    dash()

os.system('clear') #Clears the screen
print "Morse Code"
# Prompt
loop_count = input("How many times would you like SOS to loop?:
")

while loop_count > 0: #Loop around the chosen number of times
    loop_count = loop_count - 1
    morseS()
    letterSpace()
    morseO()
    letterSpace()
    morseS()
    wordSpace()

GPIO.cleanup()
```

Once complete use “Ctrl + x” then “y” then “enter” to save the file.

Running the Code

To run this code type:

```
sudo python 6-morsecode.py
```

You will be prompted for the number of times you want to repeat ‘SOS’.

Challenge

Using the above code as your template, write another program that will allow you to sound any Morse code you choose. Use the following rules:

- The length of a dot is one unit.
- The length of a dash is three units.
- The space between the parts of each letter is one unit.
- The space between letters is three units.
- The space between words is seven units.
- The letter and number codes are:

A	•–	J	•–––	S	•••	0	–––––	5	•••••
B	–•••	K	–•–	T	–	1	•––––	6	–••••
C	–•–•	L	•–••	U	••–	2	••–––	7	––•••
D	–••	M	––	V	•••–	3	•••––	8	–––••

Challenge

E	•	N	– •	W	• – –	4	• • • • –
F	• • – •	O	– – –	X	– • • –	9	– – – – •
G	– – •	P	• – – •	Y	– • – –		
H	• • • •	Q	– – • –	Z	– – • •		
I	• •	R	• – •				