Raspberry Pi Python GPIO Zero Buzz Wire Game

Tutorial by David Pride - Public Domain 26 Oct 2016 - www.cotswoldjam.org

This tutorial will cover building your own fairground style ‘buzz wire’ game! You will need speakers or headphones to use this tutorial.

DO NOT TURN ON THE RASPBERRY PI UNTIL YOU HAVE FINISHED YOUR BUILD AND HAD IT CHECKED BY A TUTOR!!

The Kit

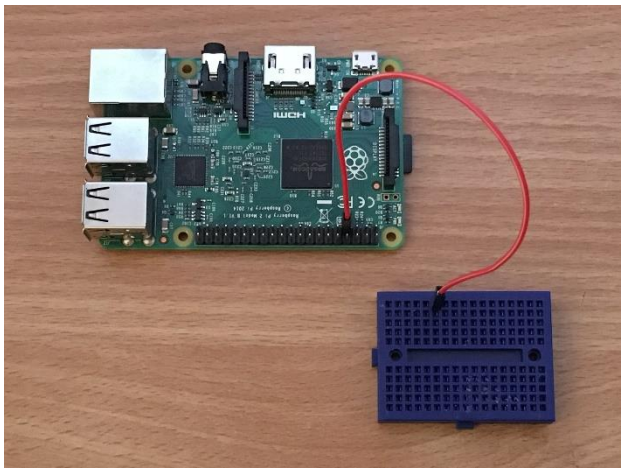
In your bag you will find the following components:

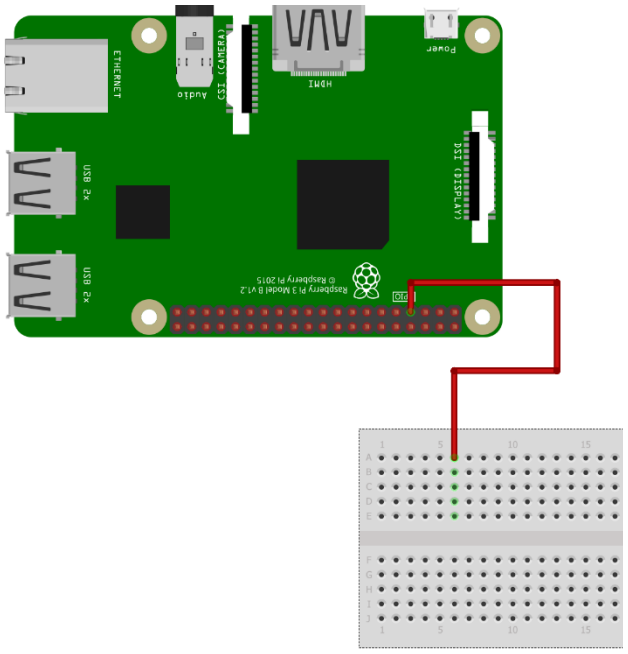
5 x M2F Jumper leads
1 x 220R Resistor
1 x LED
1 x 30cm bell wire
1 x 10cm bell wire
1 x Mini Breadboard.

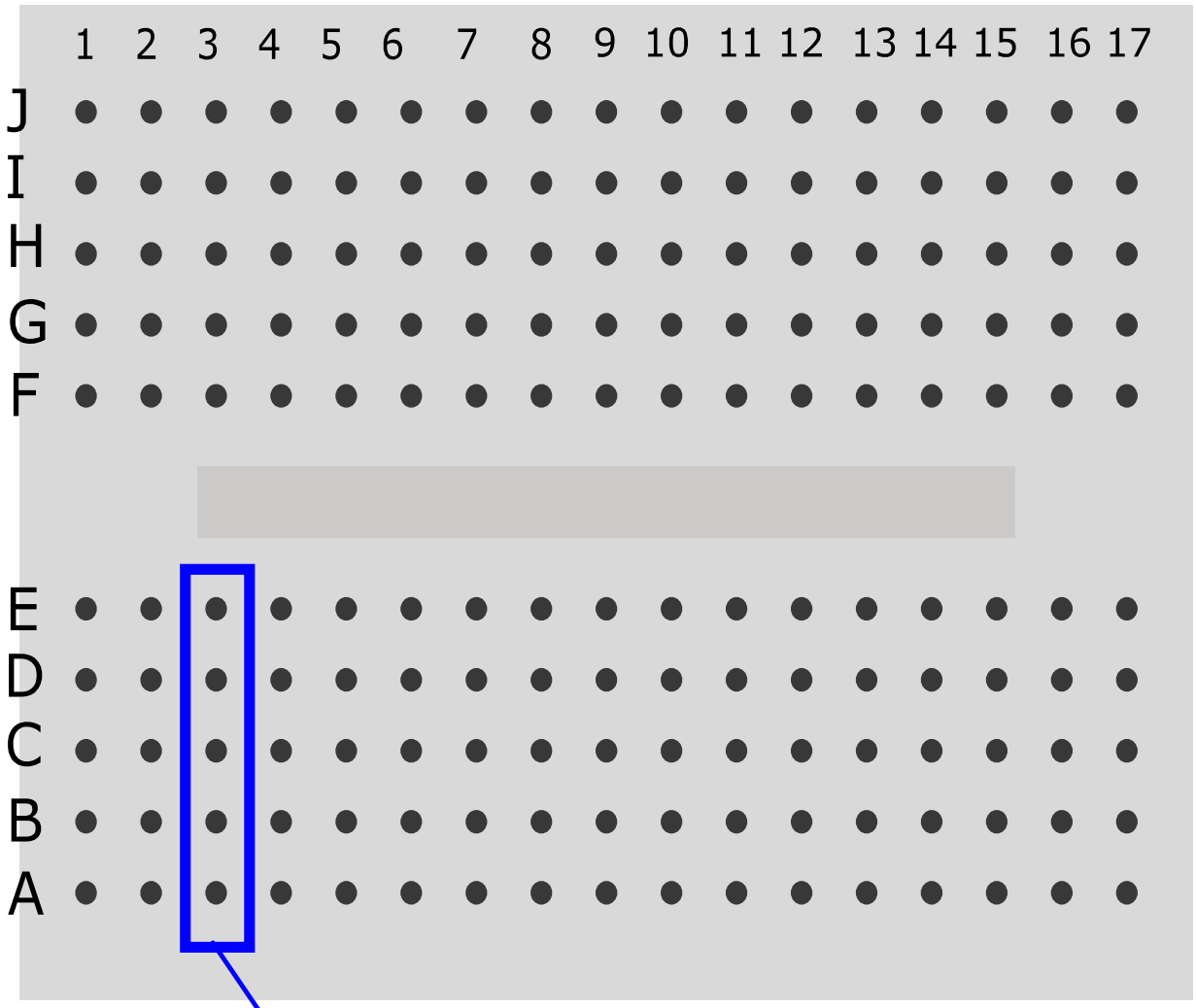
Putting it together

The colour of the wires is not important – where you put them is VERY important! Each of the holes on the breadboard is numbered – make sure everything goes in the correct hole.

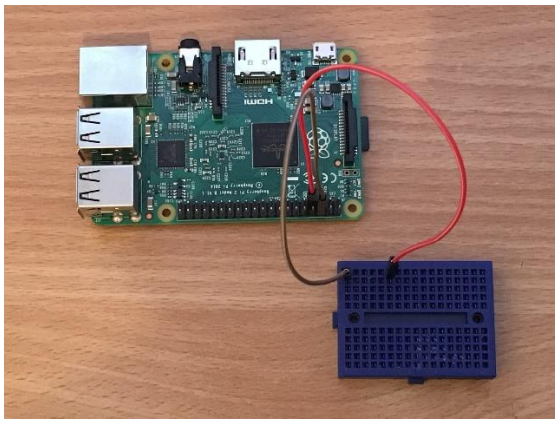
Step 1: Take 1 jumper lead and attach to J6 on breadboard and GPIO 4 on the Pi.

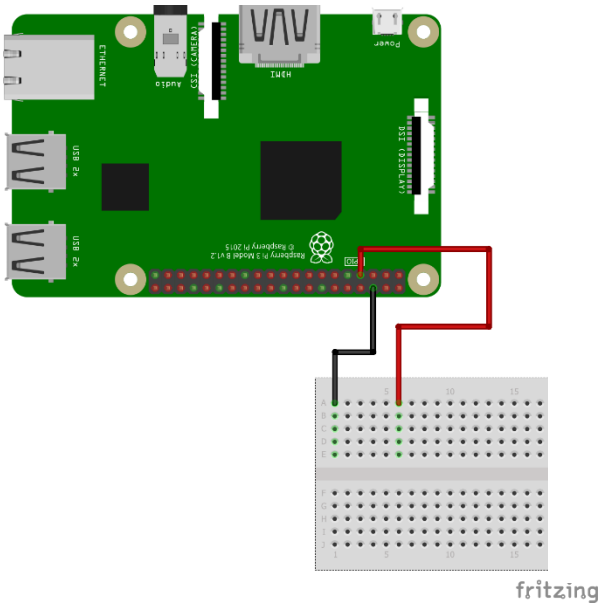




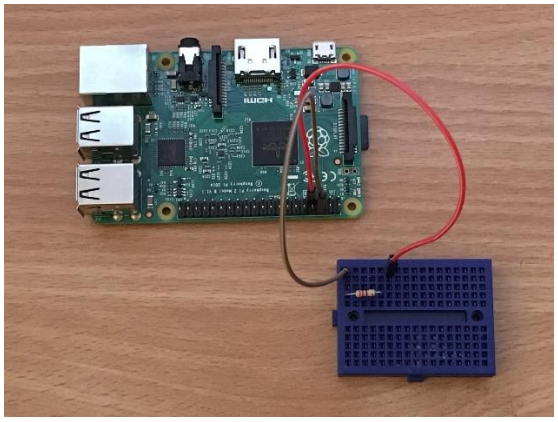

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
J
I
H
G
F
E
D
C
B
A
To connect two or more wires together, just plug them into the same column of five sockets

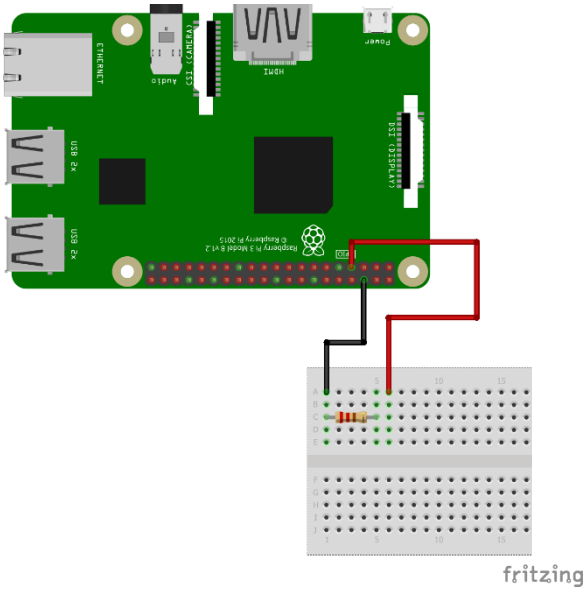
Step 2: Take 1 jumper lead and attach to J1 on breadboard and Pin 6 (GND) on the Pi.



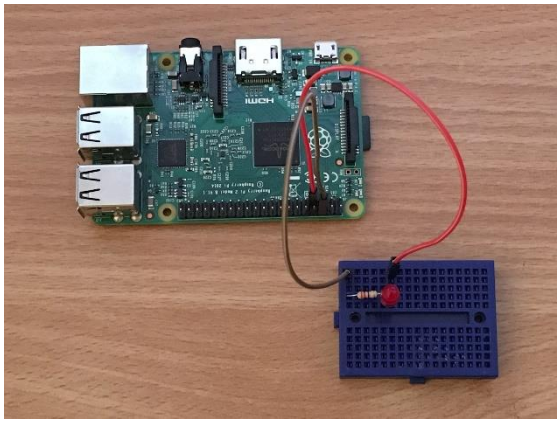


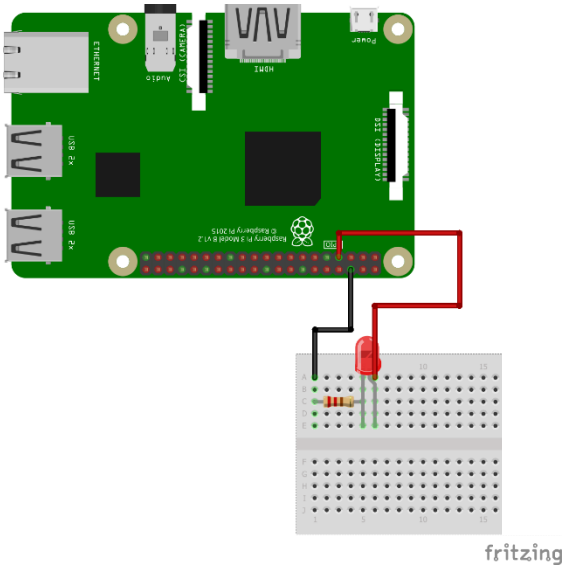
Step 3: Take the resistor and insert into holes H1 and H5 on the breadboard – it doesn't matter which way round this goes. LEDs are really greedy and will take all the power they can, which could damage the Pi. We use this resistor to limit the amount of power the LED can draw from the Pi.



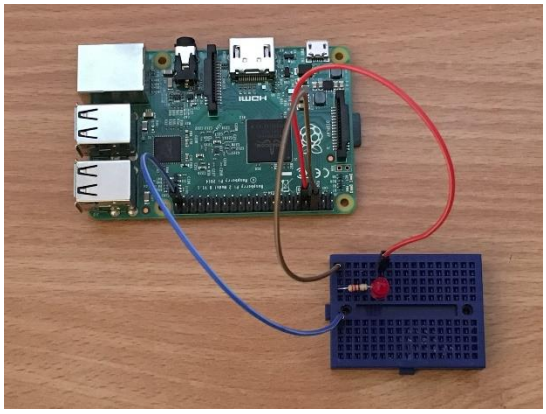


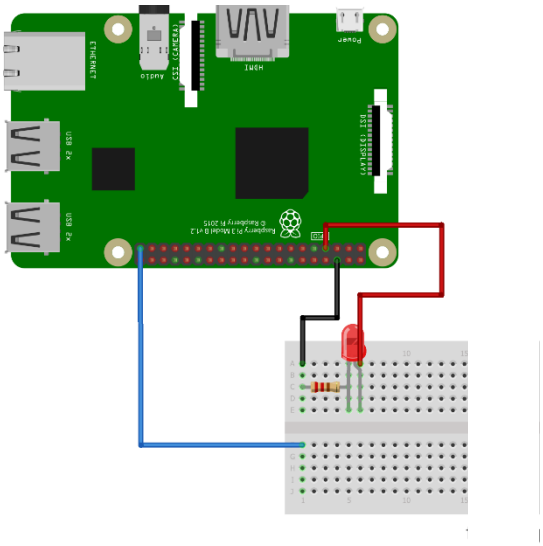
Step 4: Take the LED and insert into holes F5 and F6 on breadboard. **MAKE SURE THE LONG LEG OF THE LED GOES INTO F6.**



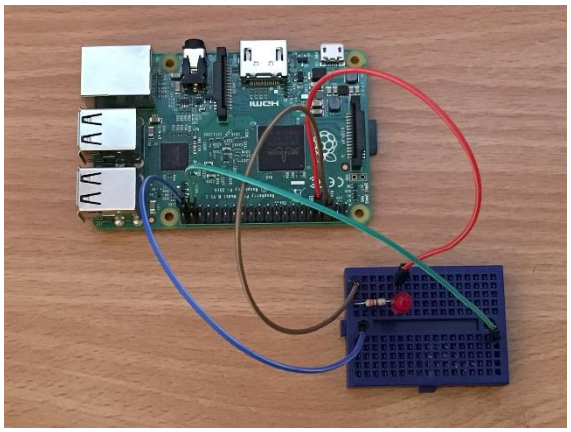


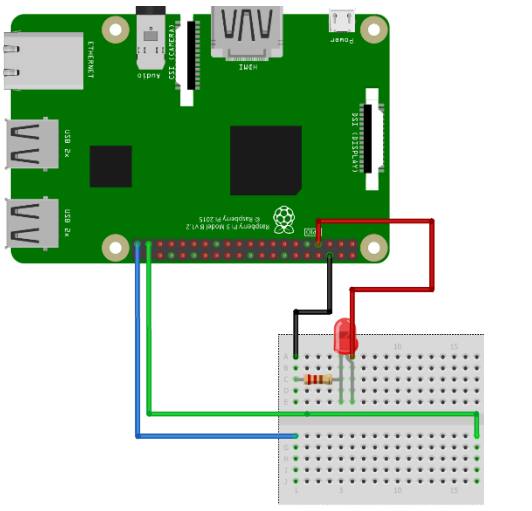
Step 5: Take 1 jumper lead and attach to E1 on the breadboard and Pin 39 (GND) on the Pi.



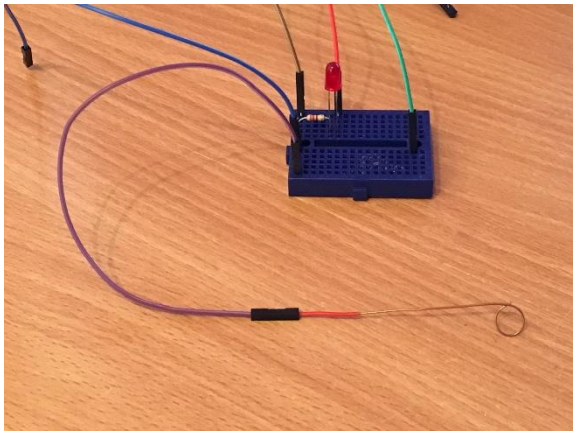


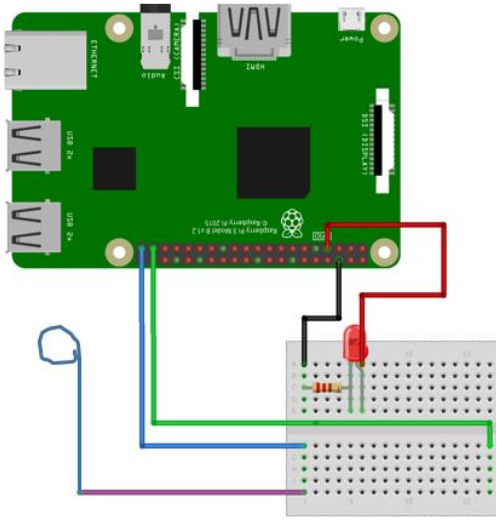
Step 6: Take 1 jumper lead and attach to E17 on the breadboard and GPIO 26 on the Pi.



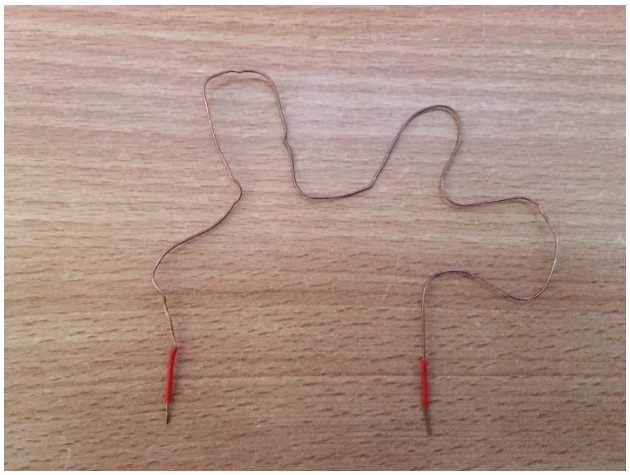


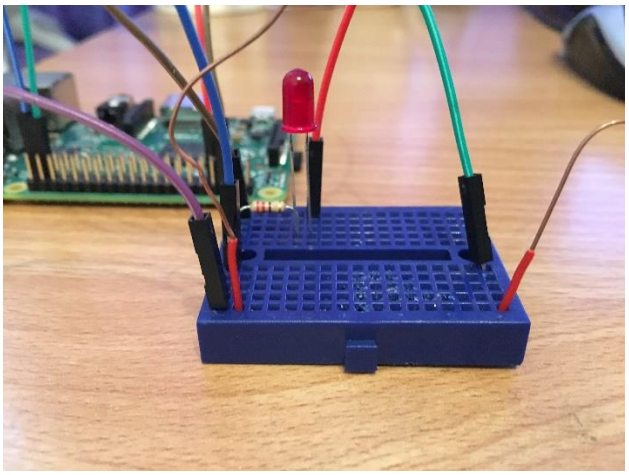
Step 7: Take 1 jumper lead and attach to A1 on the breadboard. Now take the short length of bellwire, bend the end carefully around your finger then insert the other end into the free end of the jumper lead you just attached to the breadboard.

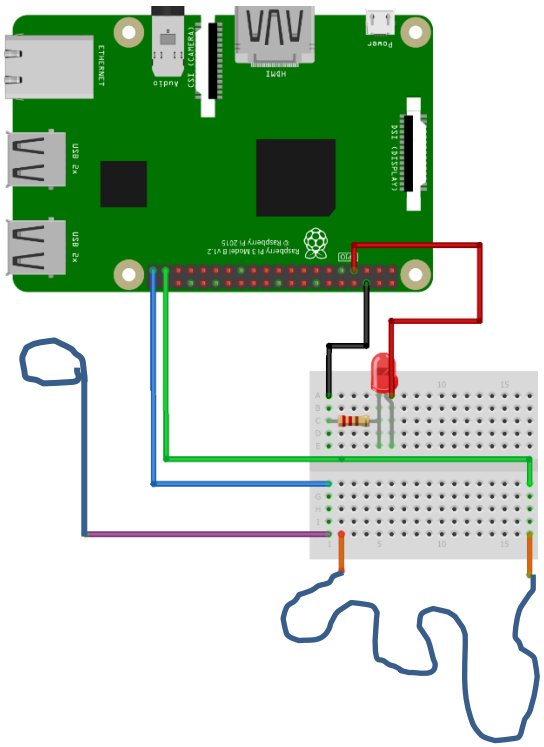




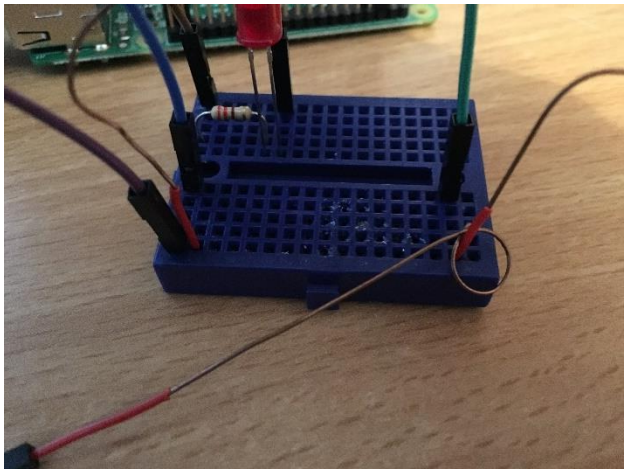
Step 8: Take the long piece of bell wire and bend it into an interesting shape. Remember – the more bends you add, the harder your game will be! Then insert the two ends of this piece of wire into A2 and A17 on the breadboard.







Step 9: Hook the loop you made in the short wire over one end of the bendy wire like this...



Your Buzz Wire game is now complete! – **Grab a tutor and ask them to check the connections now.**

The program

Power up your Raspberry Pi. From the desktop menu, select Programming - Python 3 (IDLE). Then use File, New Window to create a new program.

Type in the following program, or alternatively you can use File, Open to open the buzzwire.py program in the python/buzzwire folder.

Use File, Save to save this program as buzzwire.py **in the python/buzzwire folder** and then run it with Run menu, Run Module.

```
from time import sleep
from os import system
from gpiozero import Button, LED

led = LED(4)
button = Button(26)

while True:
    if button.is_pressed():
          print("BUZZZZZZ")
          led.on()
          system("aplay buzzer.wav &")
          sleep(1)
          led.off()
```

What does the program do?

```
from time import sleep
from os import system
from gpiozero import Button, LED
```

The first three lines tell the computer to learn about new things. Computers can learn from programs that other people have written; we call these other programs "libraries". Our instructions tell the computer to learn how to run other programs (such as the sound player) from the system library, and how electrical buttons work (two wires touching) from the GPIO Zero library.

We then create a Button object and an LED object. We say that the button is connected to GPIO 26 and the led to GPIO 4

```
led = LED(4)
button = Button(26)
```

This next line tells the program to run forever in a loop

```
while True:
```

Next the program runs until the wires touch. It then prints a message and plays the sound file.

```
  if button.is_pressed():
     print ("BUZZZZ")
     system("aplay buzzer.wav &")
```

Finally the program waits for one second, then turns off the led so play can continue.

```
     sleep(1)
     led.off()
```