# Parking Sensor Project

**Objective:** The objective of this resource is to learn about Ohm's law and basic circuitry which can be applied to a real life setting through reverse park sensors found in most cars today.

## Ohm's Law Theory

Today you will learn some basic theories like Ohm's law and together we will be building a simple parking sensor which as you gain experience can be upgraded to include items like DC motors.

Ohm's law states that the current between two points is proportional to the voltage across the two points.

| Quantity | Ohm's law symbol | Unit of measurement |
|---|---|---|
| Voltage | V or E | Volt (V) |
| Current | I | Amps (A) |
| Resistance | R | Ohm (Ω) |

The formula is E=IR. E is the voltage. I is the current. R is the resistance. The formula can be rearranged to find the current and the resistance as seen below.

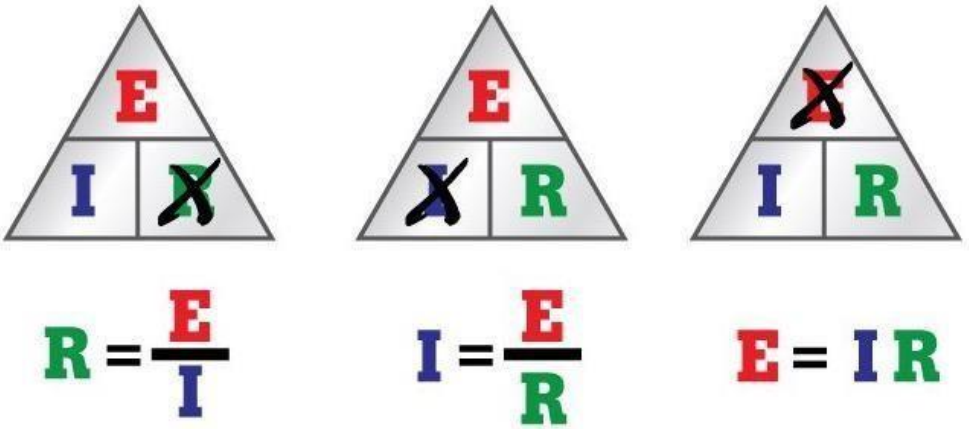



*Figure 1 – Source: Fluke,2020*

So, you might be thinking what we can use this for. Well that's simple. Ohm's Law can be used to validate the values of circuit components, current levels, voltage supplies, and voltage drops. If, for example, a test instrument detects a higher than normal current measurement, it could mean that resistance has decreased or that voltage has increased, causing a high-voltage situation. This could indicate a supply or circuit issue.

## Project Construction

**Equipment Needed:**

- 3 LED's (1 red, 1 yellow and 1 green)
- 3x 220 Ohm's Resistors
- 1x 1k Ohm Resistors
- 1x 2k Ohm Resistors
- Ultrasonic Sensor

- Breakout Board
- Piezo Buzzer
- Breadboard
- Male to Male wires

A circuit is a complete path around which electricity can flow. It must include a source of electricity, such as a battery. Materials that allow electric current to pass through them easily, called conductors, can be used to link the positive and negative ends of a battery, creating a circuit.

The main function of resistors in a circuit is to control the flow of current to other components. Take an LED for example. If too much current flows through an LED it is destroyed. So, a resistor is used to limit the current.

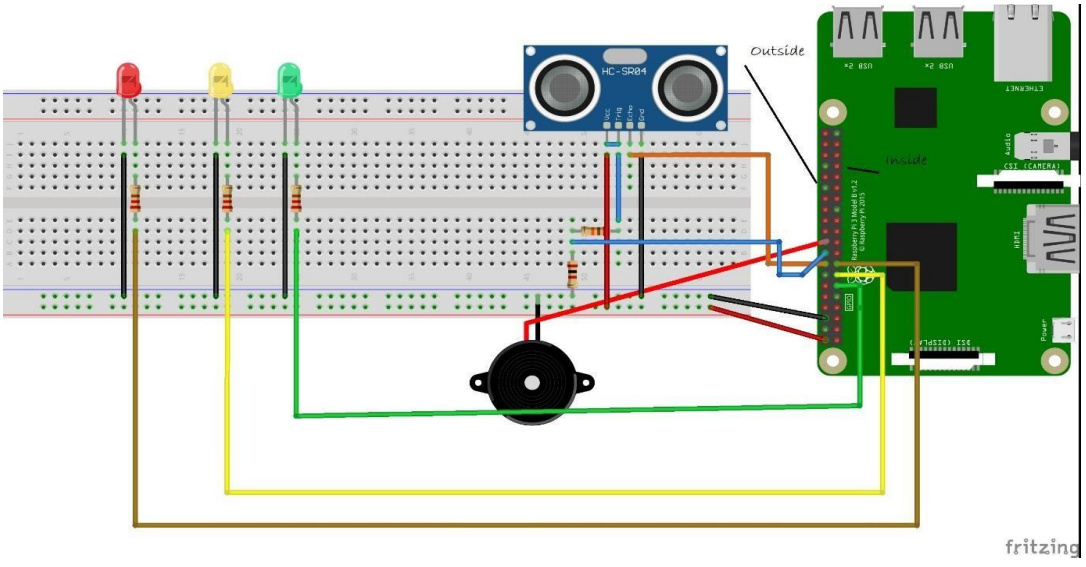
A piezo buzzer is a buzzer in which when electricity flows through it, it vibrates the piezo making a noise (Sam,2016).

An ultrasonic sensor "is an instrument that measures the distance to an object using ultrasonic sound waves. An ultrasonic sensor uses a transducer to send and receive ultrasonic pulses that relay back information about an object's proximity (Burnett,2019)." A breakout board allows for larger circuits to be prototyped easily. The four key components of an ultrasonic sensor are:

| Component | Purpose |
|---|---|
| VCC | Power supply pin – usually connected to 5v. |
| ECHO | This pin remains high for short period based on the time taken by the ultrasonic waves to bounce back to the receiving end. |
| TRIG | It plays a vital role to initialize measurement for sending ultrasonic waves. It should be kept high for 10us for triggering the measurement. |
| GND | This pin is connected to ground. |

Source: Aqeel,2018

## Circuit Schematics:



As you can see in the circuit diagram, everything is plugged into the breadboard directly, but today we will be using a breakout board to connect to our Raspberry Pi. Breakout board was not available on Fritzing when I developed this schematic.

## Connections from breadboard to Raspberry Pi:

### Ultrasonic Sensor:

| Ultrasonic Sensor Port | Raspberry Pi/Breakout Board Port Connection |
|---|---|
| GND | Ground via negative railing |
| Echo | GPIO24 via a 1K and 2K Ohm resistor |
| Trig | GPIO23 |
| VCC | 5 Volts via positive railings |

### Piezo Buzzer:

| Piezo Buzzer Wiring | Raspberry Pi/Breakout Board Port Connection |
|---|---|
| Positive (Red) | GPIO25 |
| Negative (Black) | Ground via negative railings |

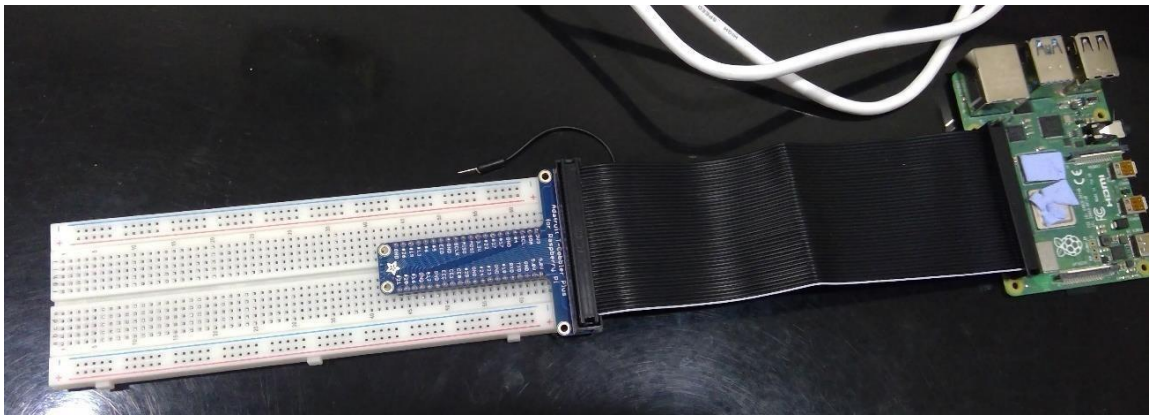
### LED:

| LED Pins | Raspberry Pi/Breakout Board Port Connection |
|---|---|
| LED (Red) – Positive | GPIO22 |
| LED (Yellow) – Positive | GPIO27 |
| LED (Green) – Positive | GPIO17 |
| LED (Red) – Negative | Via resistor into negative railing |
| LED (Yellow) – Negative | Via resistor into negative railing |
| LED (Green) – Negative | Via resistor into negative railing |

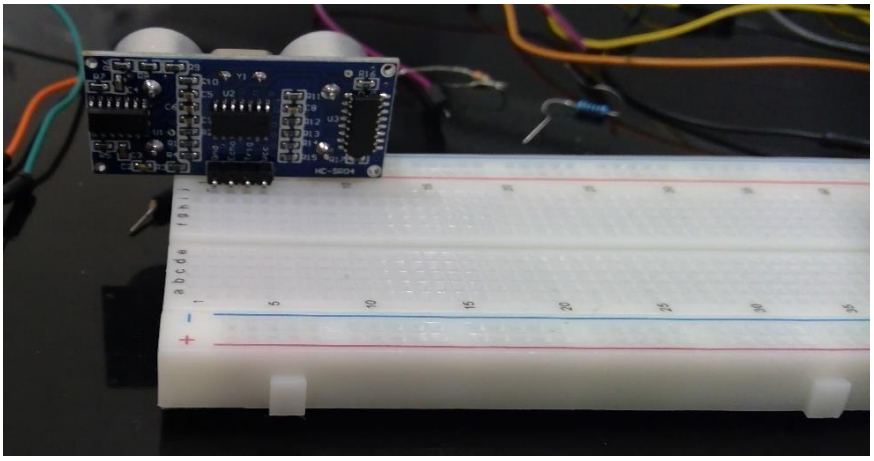
*Refer to Appendix 1 for diagram of the pins on Raspberry Pi.*

## Steps for Physical Connections:

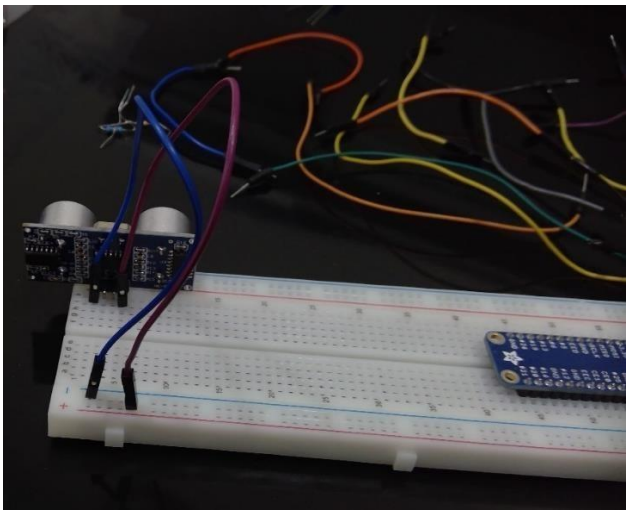
1. Place the breakout board onto the breadboard ensuring that it is fully flat. Connect the other pin onto the Raspberry Pi pins.



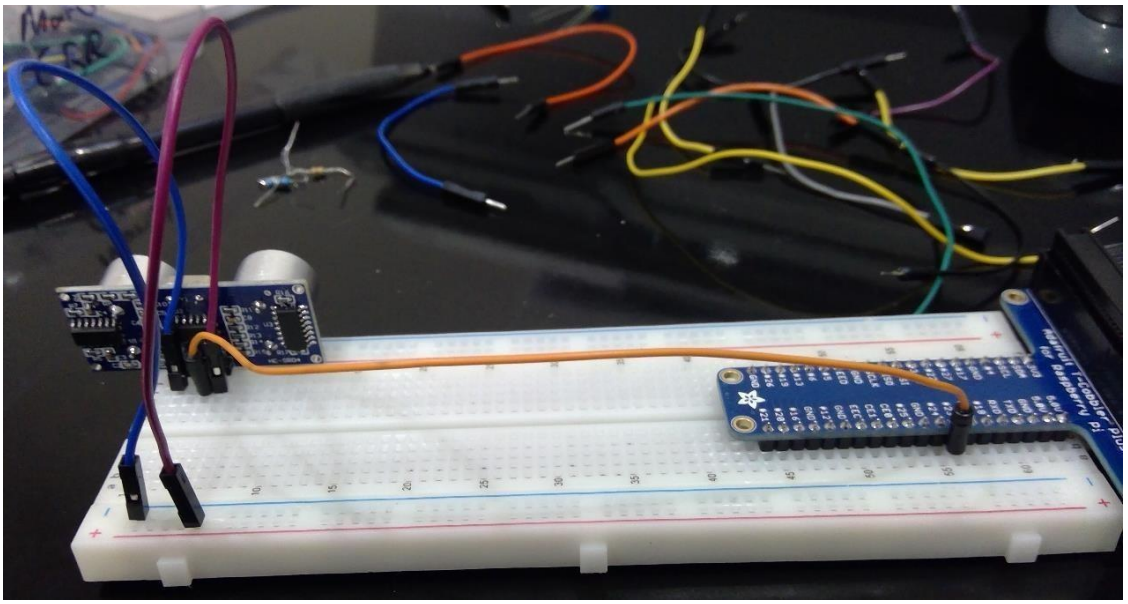
2. Plug the ultrasonic sensor into the far-left side of the breadboard.



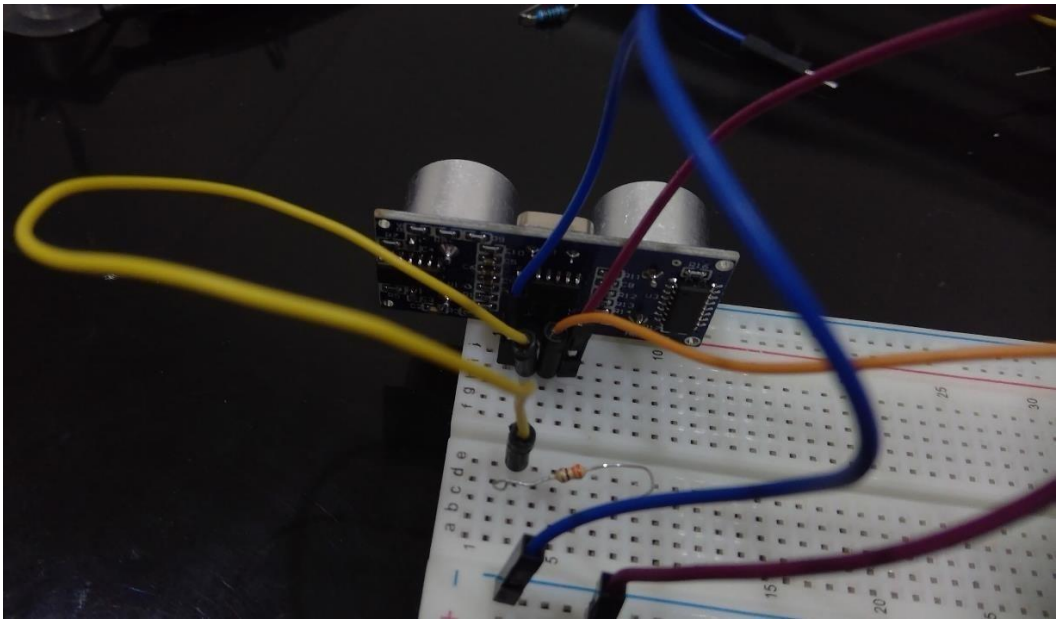
3. Plug ground into the negative railing of the breadboard and plug VCC into the positive railing of the breadboard.



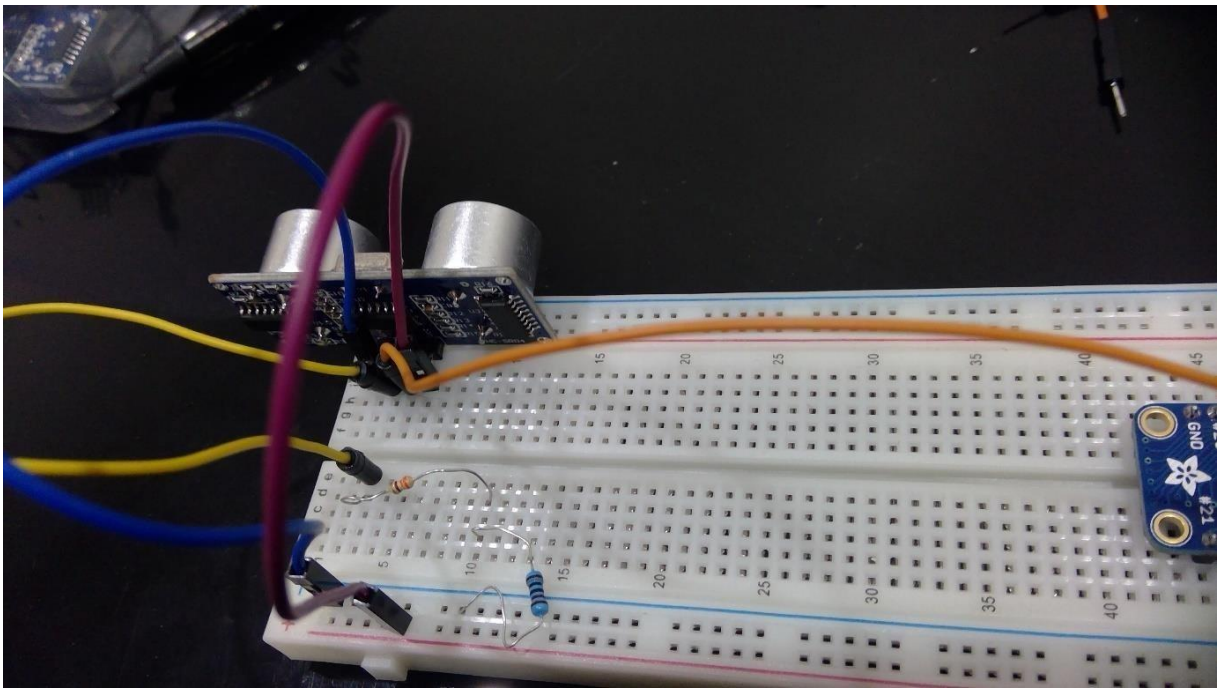
4. Plug TRIG directly into GPIO23 on the breakout board.



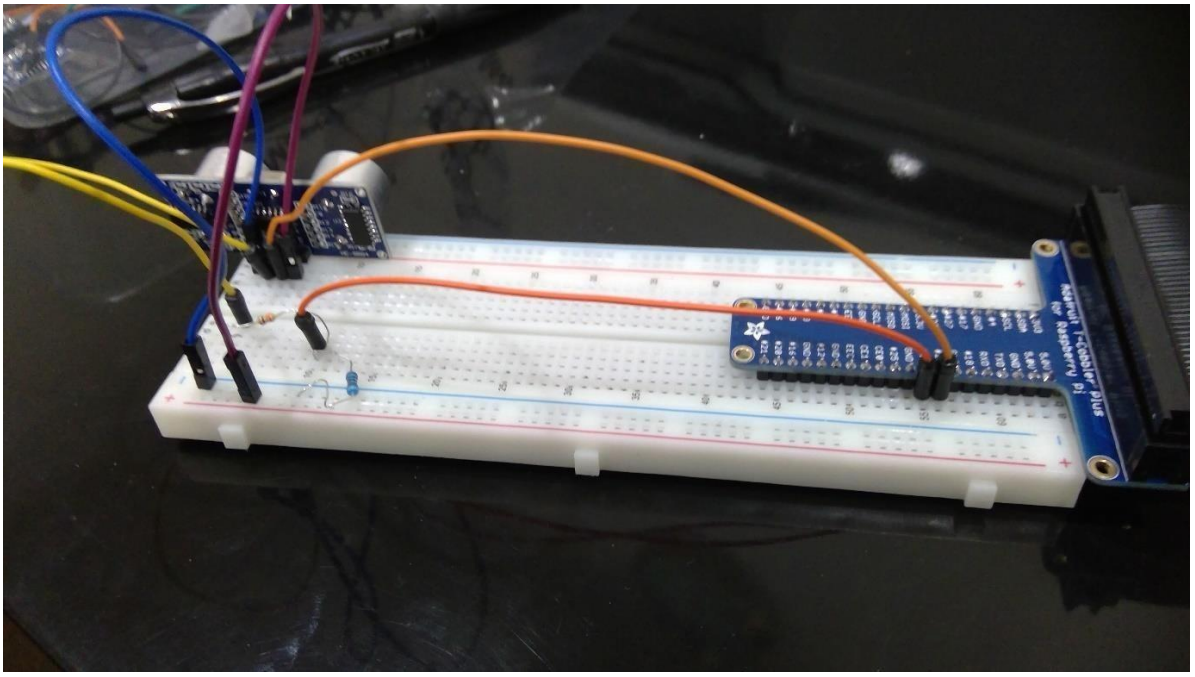
5. Plug a male to male wire from ECHO to a blank railing and place a 1K Ohm Resistor in a free rail.



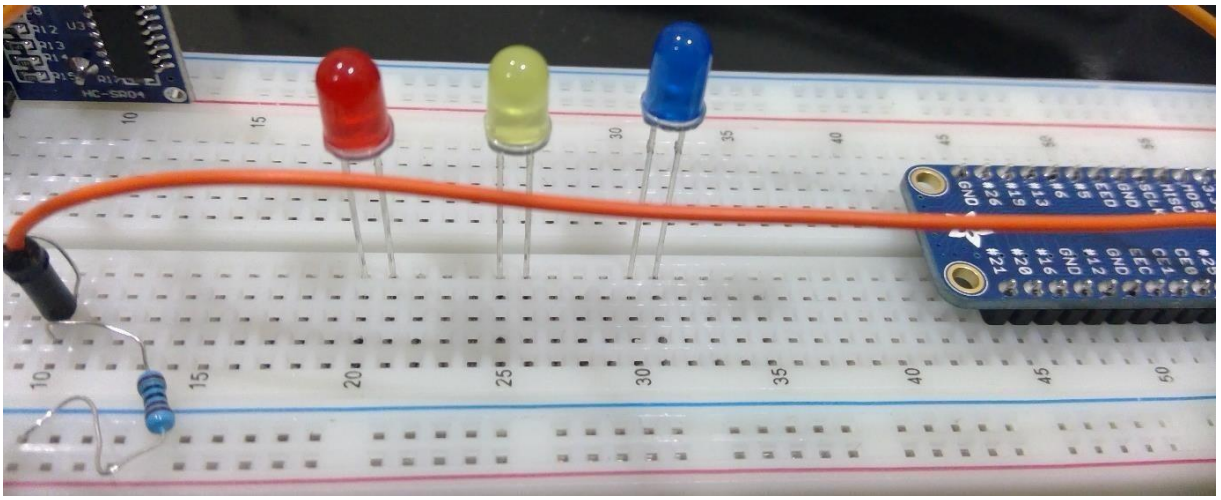
6. Link the 1K Ohm Resistor using a 2K Ohm resistor into the negative railing and ensure there is a space between these two resistors.



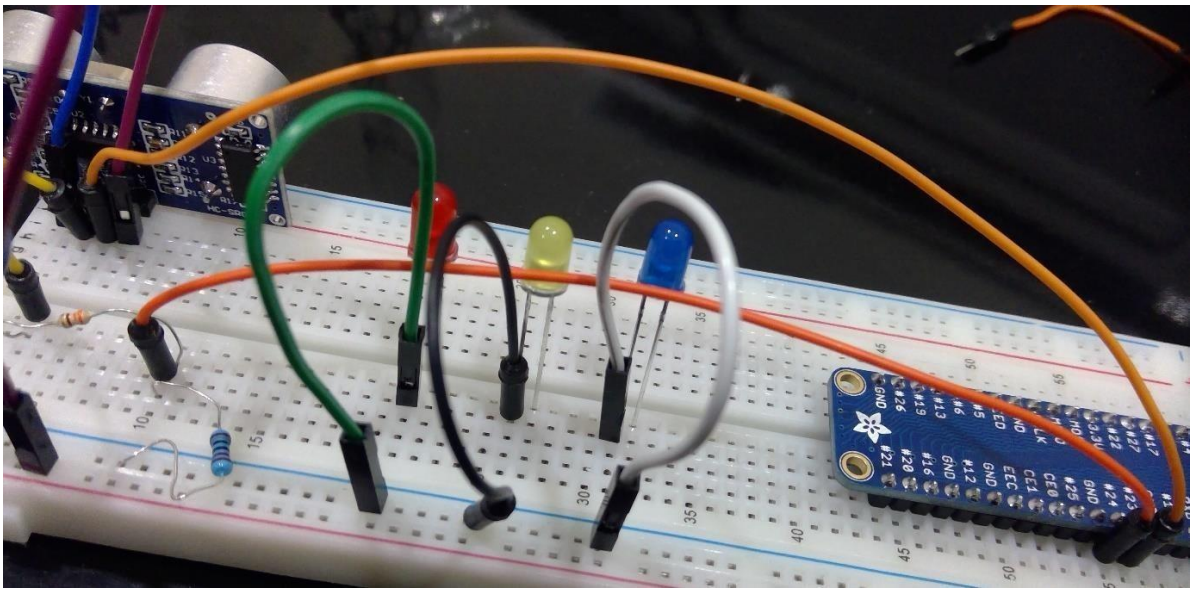

7. Plug a male to male wire in that gap and connect it to GPIO24.



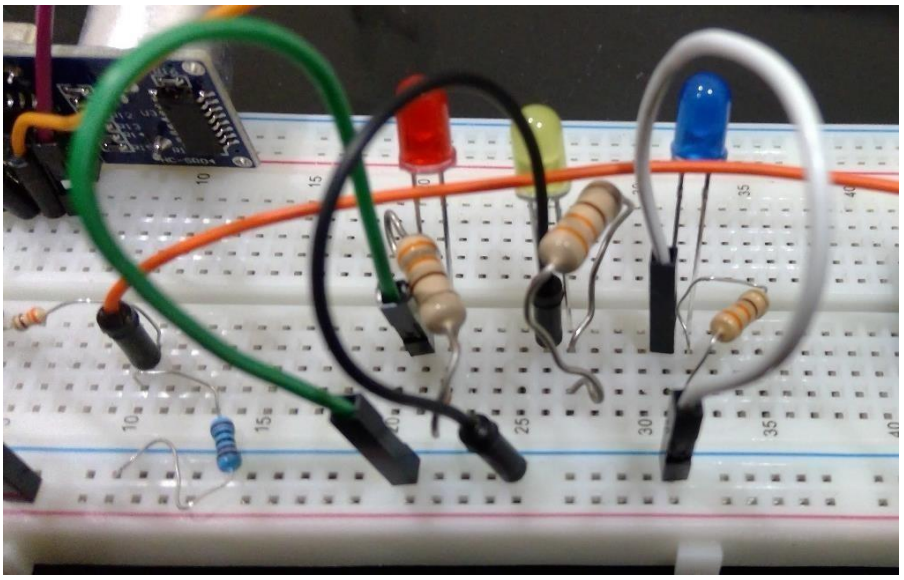
8. Next place the 3 LED's onto the breadboard and spread them out by 3 spaces.



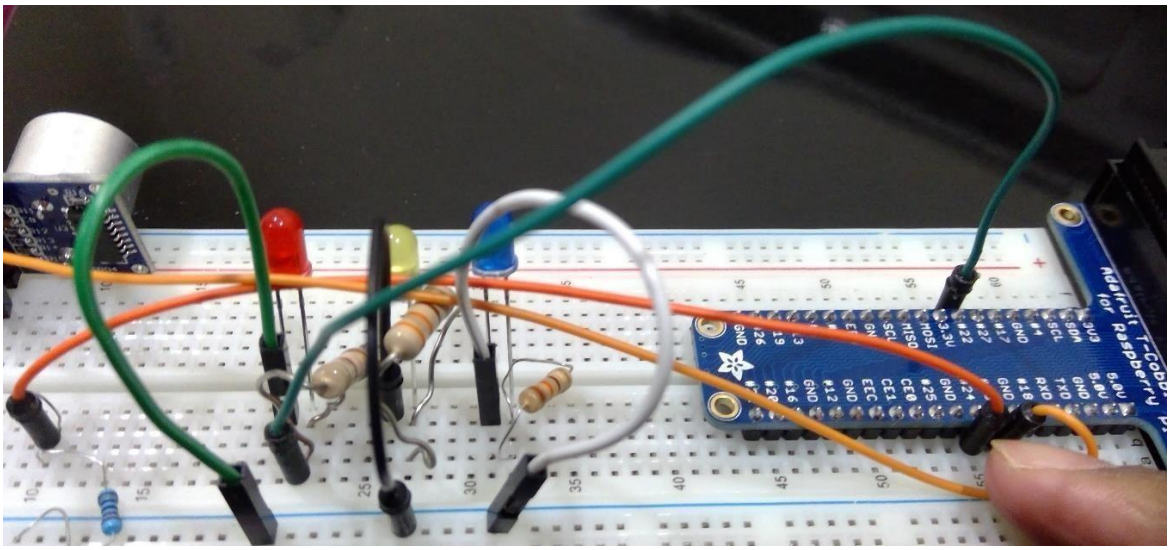
9. From the negative (short leg) plug it into the negative railing for the three LED's.



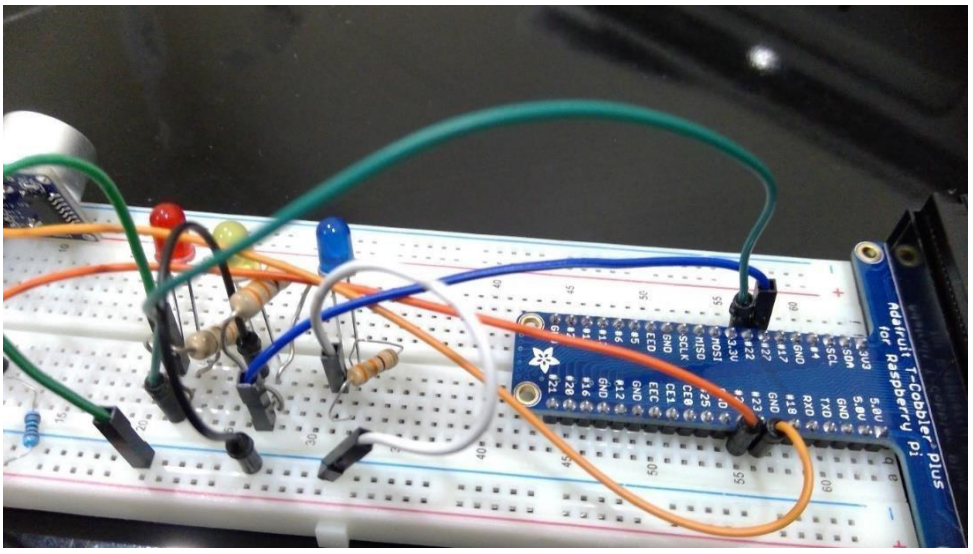
10. For the positive (long leg) plug a resistor in for all three LED's and connect to another blank rail in that row.



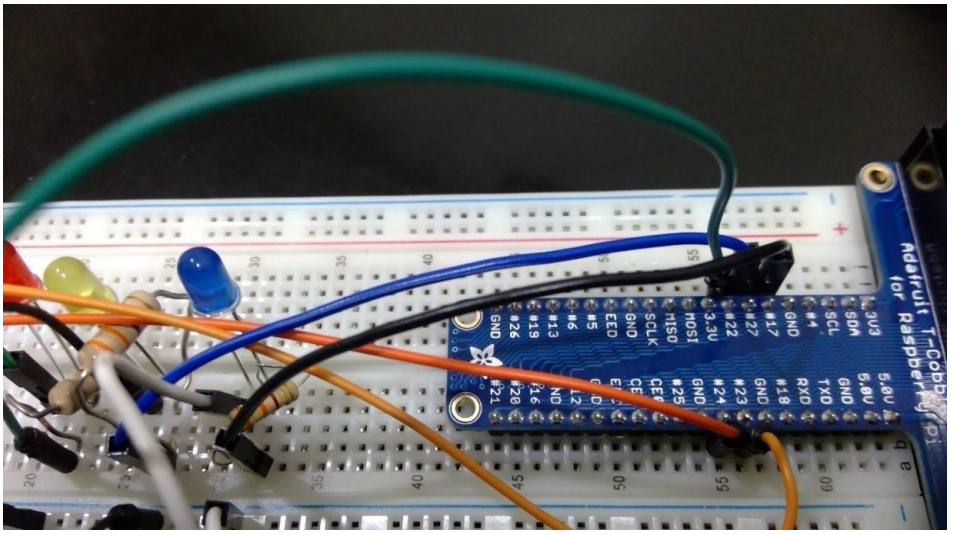
11. For the red light, connect the male to male wire from the resistor to GPIO22.



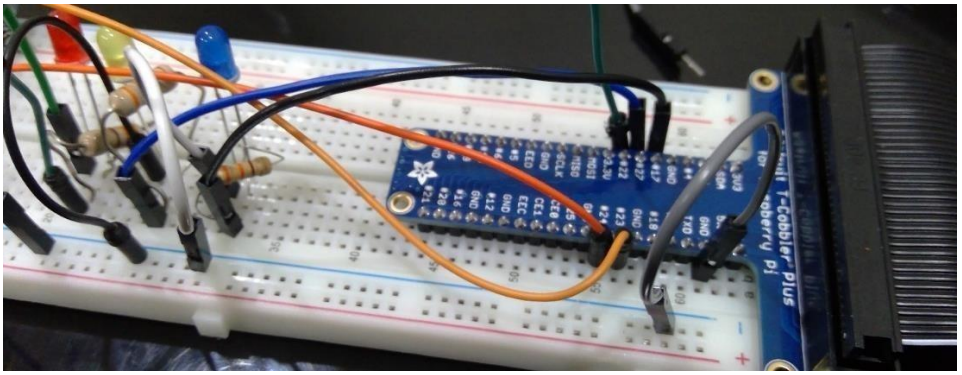
12. For the yellow light, connect the male to male wire from the resistor to GPIO27.



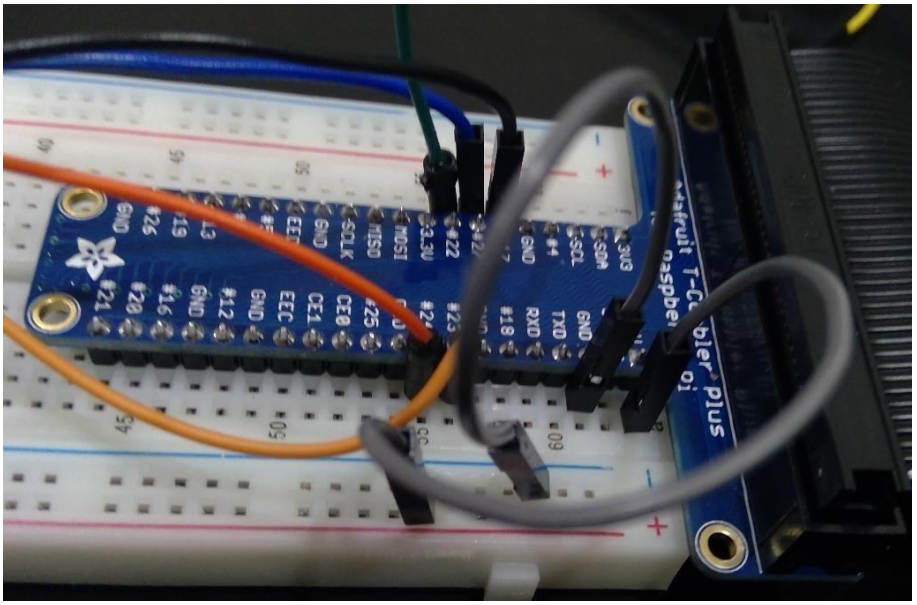

13. For the green/blue light, connect the male to male wire from the resistor to GPIO17.



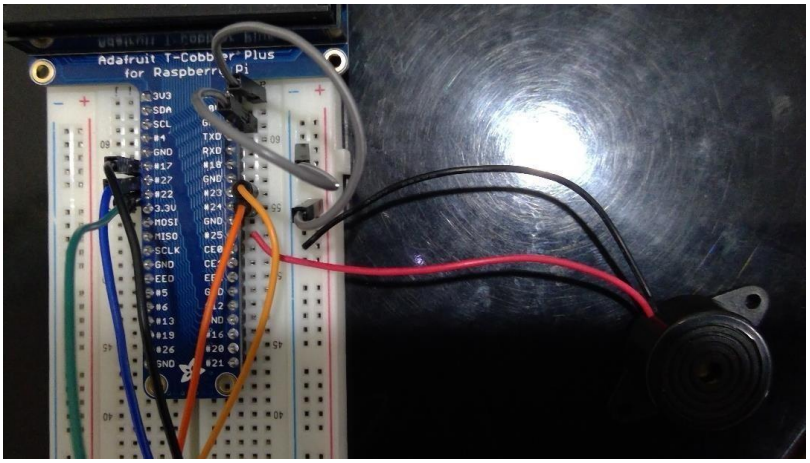

14. Connect a male to male wire from the negative railing to Ground on the breakout board.



15. Connect a male to male wire from the positive railing to 5V on the breakout board.



16. Plug the red wire(positive) on the piezo buzzer directly in line with GPIO25. And plug the black wire(negative) on the piezo buzzer directly into the negative railing.



17. We have completed the circuit build. The next step is to enter the code to make the program run. To do this, boot up the Raspberry Pi and open up Thonny Pythone IDE.

## Code:

Enter this code into Thonny and ensure you follow Python syntax, or your program will not run.

```
#code was sourced from https://www.instructables.com/id/Raspberry-Pi-Park-Sensor/ and modified to meet the needs of the task as there were errors found in the original source code.

# The code was also modified to add a piezo buzzer that would make noise when the object was too close to

sensor. import RPi.GPIO as GPIO  #import libraries import time            #import libraries

GPIO.setwarnings(False) #getting rid of any warnings that may block the code from running.


GPIO.setmode(GPIO.BCM) #setting the pinmode to GPIo and we tell it we are using the BCM GPIO numbers.

TRIG = 23             # We are setting the TRIG on the ultrasonic sensor to port 23

ECHO = 24             # We are setting the ECHO on the ultrasonic sensor to port 24

GREEN = 17            # We are setting the Green Light to port 17

YELLOW = 27           # We are setting the orange Light to port 27

RED = 22              # We are setting the Red Light to port 22

BUZZER = 25           # We are setting the Red Light to port 25

GPIO.setup(TRIG,GPIO.OUT) # We are setting up the TRIG as an output.

GPIO.setup(ECHO,GPIO.IN) # We are setting up the ECHO as an input.

GPIO.setup(GREEN,GPIO.OUT) # We are setting up the green LED as an output

GPIO.setup(YELLOW,GPIO.OUT) # We are setting up the yellow LED as an output

GPIO.setup(RED,GPIO.OUT) # We are setting up the red LED as an output

GPIO.setup(BUZZER,GPIO.OUT) # We are setting up the buzzer as an output start

= 1 #declaring a global variable


def no_light ():            #This is what the output should be when the function is called. This is saying that nothing will turn on.

GPIO.output(GREEN, GPIO.LOW) # The green light be  off

   GPIO.output(YELLOW, GPIO.LOW) # The yellow light be off.

   GPIO.output(RED, GPIO.LOW)     # The red light will be off.

   GPIO.output(BUZZER, GPIO.LOW) # The buzzer will be off.
```

```
def green_light():              #This is what the output should be when the function green light is to be run.
    GPIO.output(GREEN, GPIO.HIGH) # The green light will turn on.
    GPIO.output(YELLOW, GPIO.LOW) # The yellow light be off.
    GPIO.output(RED, GPIO.LOW)      # The red light will be off.
    GPIO.output(BUZZER, GPIO.LOW) # The buzzer will be off.


def yellow_light():             #This is what the output should be when the function yellow light is to be run.
    GPIO.output(GREEN, GPIO.LOW) # The green light be  off
    GPIO.output(YELLOW, GPIO.HIGH) # The yellow light will turn on.
    GPIO.output(RED, GPIO.LOW)      # The red light will be off.
    GPIO.output(BUZZER, GPIO.LOW)  # The buzzer will be off.


def red_light(): #This is what the output should be when the function red light is to be run.
    GPIO.output(GREEN, GPIO.LOW)     # The green light be  off
    GPIO.output(YELLOW, GPIO.LOW) #The yellow light be  off.
    GPIO.output(RED, GPIO.HIGH) # The red light will turn on.
    GPIO.output(BUZZER, GPIO.HIGH) # The buzzer will turn on and make noises.


def get_distance(): #We are defining a function here called get_distance
GPIO.output(TRIG, True) #Sets gpio output pin to a high or 1
time.sleep(0.00001) #sleeps for 0.00001
    GPIO.output(TRIG, False) #Sets gpio output pin to a low or 0     global start     #In Python, global keyword is used to create a
global variable and make changes to the variable in a local context.
while GPIO.input(ECHO) == False:
        start = time.time() #this lets the variable start equal time in seconds since epoch, its continuously reset while the echo = false
while GPIO.input(ECHO) == True:
        end = time.time() #this lets the variable end equal time in seconds since epoch, its continuously reset while the echo = true


     signal_time = end-start # basic maths variable end minus variable start equals new variable signal_time      distance =
signal_time / 0.000058  # basic maths variable signal_time divided by 0.000058 equals new variable distance      return
distance #A return statement is used to end the execution of the function call and “returns” the result
```

```
    while True: #This will happen if ECHO picks something up and it will loop until user ends it.
    distance = get_distance()     time.sleep(0.05)     print("distance")
#when it detects something it will print "distance"

    if distance >= 60: #if the distance is greater than the declared value then the function no_light will run. This means nothing will turn on.
        no_light()

    elif 90 > distance > 23: #if the distance is between than the declared value then the function no_light will run. This means that function
green_light will run.
        green_light()

    elif 22 > distance > 16: #if the distance is between than the declared value then the function no_light will run. This means that function
green_light will run.        yellow_light()

    elif distance <=15: #if the distance is between than the declared value then the function no_light will run. This means that function
green_light will run.
        red_light()

    else:            #if there is an issue with anything like sensor not working etc. it will print "error".
        print ("error")
```

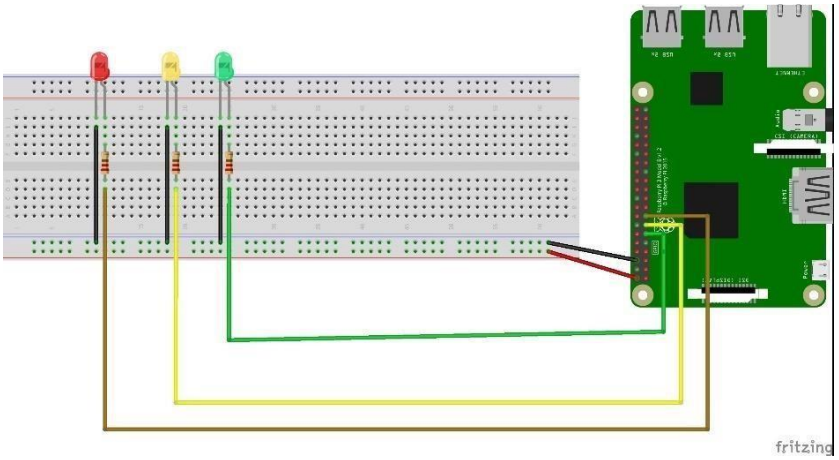
Once you have entered the code and ensured that there are no Python syntax errors, run the program and your parking sensor will work.

## Applications of Ohm's Law in our circuit:

We can apply Ohm's law in our circuit. For our example today we can find out how much current is flowing through aspects of our circuit.

To find the current flowing through our circuit we use the formula: $I = \frac{V}{R}$

Let us try to calculate the current flowing into the LED's in our circuit.




We know that our resistors are 220 Ohm. We also know that operating voltage given to GPIO pins is 3.3V.

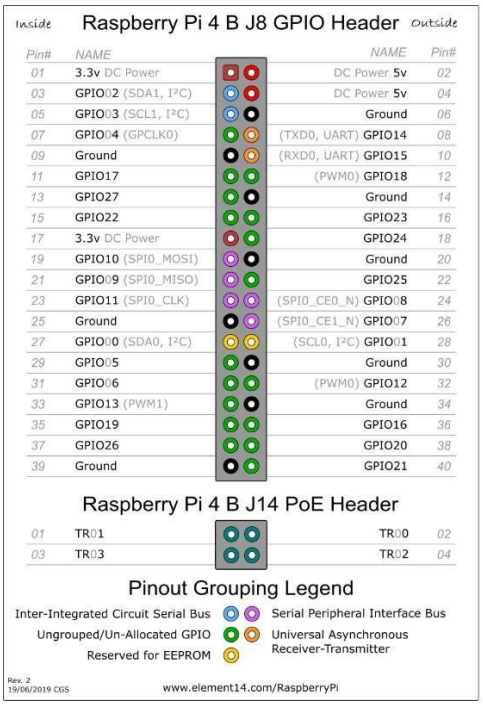
Let's input this data into our given equation.

$$I = \frac{3.3V}{220\ Ohm}$$

$$0.015 = \frac{3.3V}{220\ Ohm}$$

Thus, the current flowing into the LED is 0.015 Amps This can be converted to 15 milliamps which is the maximum amount we can safely draw power to light and LED (Pounder,2019).

# Appendix:




*Appendix 1*

# Reference List:


Aqeel, A 2018, Introduction to HC-SR04 (Ultrasonic Sensor), viewed 22 March 2020, <https://www.theengineeringprojects.com/2018/10/introduction-to-hc-sr04-ultrasonicsensor.html>.

Burnett, R 2019, Understanding How Ultrasonic Sensors Work, MaxBotix, viewed 21 March 2020, <https://www.maxbotix.com/articles/how-ultrasonic-sensors-work.htm>.

Fluke 2018, What is Ohm's Law?, viewed 19 March 2020, <https://www.fluke.com/en-au/learn/bestpractices/measurement-basics/electricity/what-is-ohms-law>.

Pounder, L 2019, Raspberry Pi GPIO Pinout: What Each Pin Does on Pi 4, Earlier Models, Tom's Hardware, viewed 21 March 2020, <https://www.tomshardware.com/reviews/raspberry-pi-gpiopinout,612>.

Sam, N 2016, Arduino: How to Use a Piezo Buzzer, online video, June 13, viewed 21 March 2020, <https://www.youtube.com/watch?v=K8AnlUT0ng0>.