**Learn Interfacing Distance Sensor. [Lab 2]**

**OBJECTIVE:**

The idea of this park sensor is to show green when you have plenty of room to pull your car forward in parking lot, and then turn yellow as you approach the fully forward position, and then red when you should stop. Further, we improve it by adding a buzzer.

### **COMPONENTS:**

1) RPi 3

2) Breadboard

3) 330 Ω Resistor x 5

4) LED x 3 (Red, Yellow, Green)

5) HC-SR04 Ultrasonic Distance Sensor x 1

6) Active Buzzer x 1

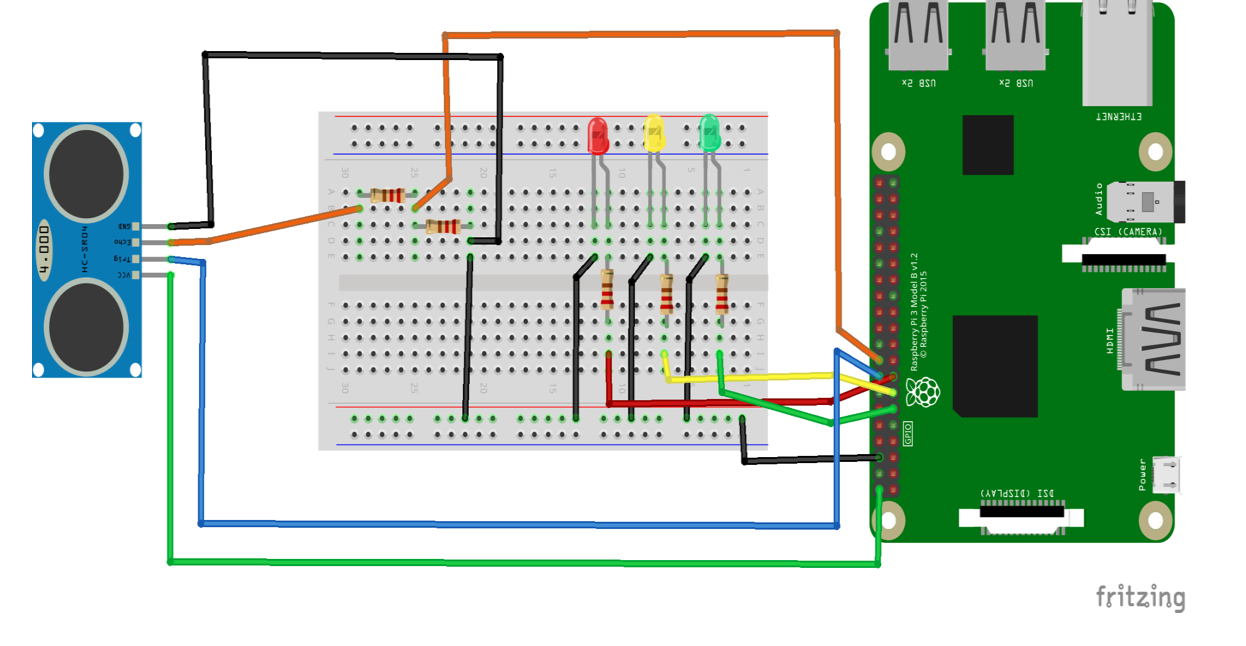
7) Connecting Wires

**Explanation**

Ultrasonic Sensor: HC-SR04 is a commonly used module for non-contact distance measurement for distances from 2cm to 400cm. This sensor uses ultrasonic sound to measure distance just like bats and dolphins do. Ultrasonic sound has such a high pitch that humans cannot hear it. This particular sensor sends out an ultrasonic sound that has a frequency of about 40 kHz. The sensor has two main parts: a transducer that creates an ultrasonic sound and another that listens for its echo. The time difference between transmission and reception of ultrasonic signals is calculated. Using the speed of sound and ‘Speed = Distance/Time' equation, the distance between the source and target can be easily calculated.

Distance Calculation: Distance = Speed \* Time/2 The "2" is in the formula because the sound has to travel back and forth. First the sound travels away from the sensor, and then it bounces off of a surface and returns back. we need only half of this. Therefore time is taken as time/2. Speed of sound at sea level = 340 m/s

Thus, Distance = 17150 \* Time (unit cm) For accurate distance readings, the output can be calibrated using a ruler. In the below program a calibration of 0.5 cm is added.



**CIRCUIT DESCRIPTION:**

1. BCM PIN CONFIGURATION IS USED.
2. VCC of the ultrasonic sensor is connected to 5v pin of RPi
3. Trigger to pin 23
4. Echo is connected to 330 ohm resistor and to pin 24.
5. 330 ohm resistors are for the leds and they are connecting to the positive leg of the leds and then GPIO.

### **CREATING THE PROGRAM:**

1. Go to TERMINAL
2. Navigate to workspace folder (created in lab 1) by typing the command: $ cd workspace
3. Open this folder in your file system
4. Right click the mouse and choose new file and name it as lab3.py
5. Right click on the file and open with IDLE3. Make it default program.
6. Type the below code and save the file.

**Code:**

**import RPi.GPIO as GPIO**

**import time**

**trigger\_pin = 23**

**echo\_pin = 24**

**red\_pin = 22**

**yellow\_pin = 27**

**green\_pin = 17**

**def setup(): #method to setup pins and make led's low initially**

**GPIO.setmode(GPIO.BCM)**

**GPIO.setup(echo\_pin, GPIO.IN)**

**GPIO.setup(trigger\_pin, GPIO.OUT)**

**GPIO.setup(red\_pin, GPIO.OUT)**

**GPIO.setup(yellow\_pin, GPIO.OUT)**

**GPIO.setup(green\_pin, GPIO.OUT)**

**GPIO.output(green\_pin, GPIO.LOW)**

**GPIO.output(yellow\_pin, GPIO.LOW)**

**GPIO.output(red\_pin, GPIO.LOW)**

**try:**

**print('calling loop')**

**calculate\_distance() #calling distance calculation method**

**except KeyboardInterrupt: #cleaning pins on program exit**

**print('cleaning up gpio pins')**

**GPIO.cleanup()**

**def red():**

**GPIO.output(red\_pin, GPIO.HIGH)**

**GPIO.output(green\_pin, GPIO.LOW)**

**GPIO.output(yellow\_pin, GPIO.LOW)**

**def yellow():**

**GPIO.output(yellow\_pin, GPIO.HIGH)**

**GPIO.output(red\_pin, GPIO.LOW)**

**GPIO.output(green\_pin, GPIO.LOW)**

**def green():**

**GPIO.output(green\_pin, GPIO.HIGH)**

**GPIO.output(yellow\_pin, GPIO.LOW)**

**GPIO.output(red\_pin, GPIO.LOW)**

**def calculate\_distance():**

**while True:**

**GPIO.output(trigger\_pin, False) #set trigger\_pin as LOW**

**print ("Giving delay between distance measuring cycle")**

**time.sleep(0.5)**

**GPIO.output(trigger\_pin, True) #set trigger\_pin as HIGH**

**time.sleep(0.00001)**

**GPIO.output(trigger\_pin, False)**

**while GPIO.input(echo\_pin)==0: #check if echo\_pin is LOW**

**pulse\_start = time.time() #saves the last known time of LOW pulse**

**while GPIO.input(echo\_pin)==1: #Check if echo\_pin is HIGH**

**pulse\_end = time.time() #saves the last known time of HIGH pulse**

**pulse\_duration = pulse\_end - pulse\_start #get pulse duration to a variable**

**distance = pulse\_duration \* 17150 #multiply pulse duration by 17150 to get distance**

**distance = round(distance, 2) #round to two decimal points**

**print ("Distance:",distance - 0.5,"cm") #print distance with 0.5 cm calibration**

**if distance <= 5: #call red method if distance is less than or equal to 5cm**

**red()**

**elif 6 <= distance < 20: #call yellow method if distance is greater than 6cm and less than 20cm**

**yellow()**

**elif 21 <= distance < 150: #call green method if distance is greater than 21cm and less than 150cm**

**green()**

**else:**

**print("Out of Range")**

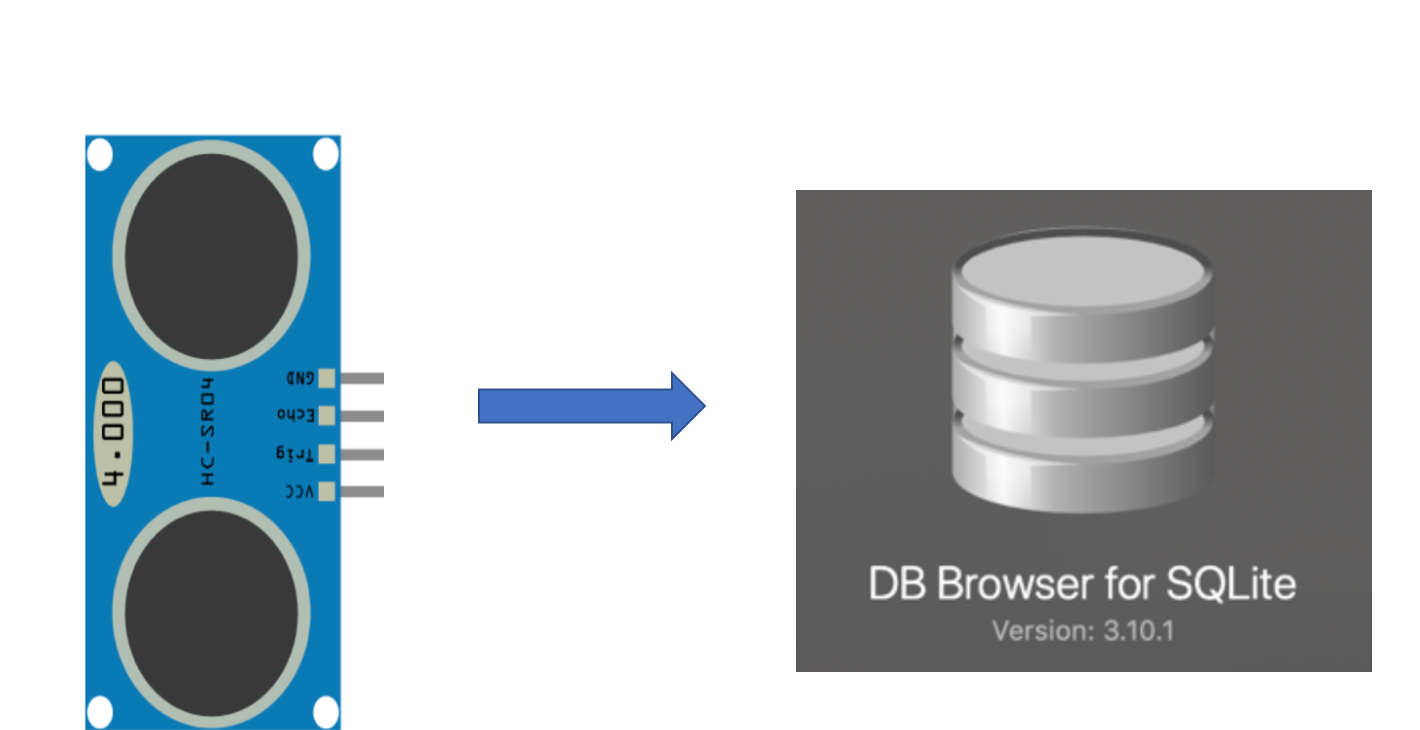
**setup()**

## **EXPLANATION:**

With reference to the above code:

If the distance is greater than or equal to 5 cm, we show a red light. If it's between 6 and 20 cm, we'll turn yellow, and then we'll turn green for greater than or equal to 21 cm and less than 150cm.

# Distance Sensor Data Logging on a Database using SQLITE3 [Lab 2A]



So far, we learned how to interface Distance Sensor with Raspberry pi let us include a database. In this experiment we try to log the sensor value to the database with time when it was recorded. Please make sure to follow or connect the pins correctly in this example I am using trigger pin GPIO 17 and echo as GPIO as 27

Code:

import RPi.GPIO as GPIO

import time

import datetime

#GPIO Mode (BOARD / BCM)

GPIO.setmode(GPIO.BCM)

#set GPIO Pins

GPIO\_TRIGGER = 23

GPIO\_ECHO = 24

#set GPIO direction (IN / OUT)

GPIO.setup(GPIO\_TRIGGER, GPIO.OUT)

GPIO.setup(GPIO\_ECHO, GPIO.IN)

def get\_time():

t = datetime.datetime.now()

my\_time ='{}:{}:{}'.format(t.hour,t.minute,t.second)

return my\_time

def distance():

# set Trigger to HIGH

GPIO.output(GPIO\_TRIGGER, True)

# set Trigger after 0.01ms to LOW

time.sleep(0.00001)

GPIO.output(GPIO\_TRIGGER, False)

StartTime = time.time()

StopTime = time.time()

# save StartTime

while GPIO.input(GPIO\_ECHO) == 0:

StartTime = time.time()

# save time of arrival

while GPIO.input(GPIO\_ECHO) == 1:

StopTime = time.time()

# time difference between start and arrival

TimeElapsed = StopTime - StartTime

# multiply with the sonic speed (34300 cm/s)

# and divide by 2, because there and back

distance = (TimeElapsed \* 34300) / 2

my\_time = get\_time()

my\_database(distance, my\_time)

return distance

def my\_database(distance,my\_time):

"""

:return: Nothing

"""

# define the connection

conn = sqlite3.connect("distance.db")

#define the cursor

cursor = conn.cursor()

# create Tables

cursor.execute("""

CREATE TABLE IF NOT EXISTS my\_table (id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,

distance TEXT,

my\_time TEXT)""")

# Add data to database

cursor.execute("""

INSERT INTO my\_table (distance,my\_time) VALUES (?, ?)""",(distance, my\_time))

# perform 3C

#COMMIT, CLOSE CLOSE

conn.commit()

cursor.close()

conn.close()

if \_\_name\_\_ == '\_\_main\_\_':

try:

while True:

dist = distance()

print ("Measured Distance = %.1f cm" % dist)

time.sleep(1)

# Reset by pressing CTRL + C

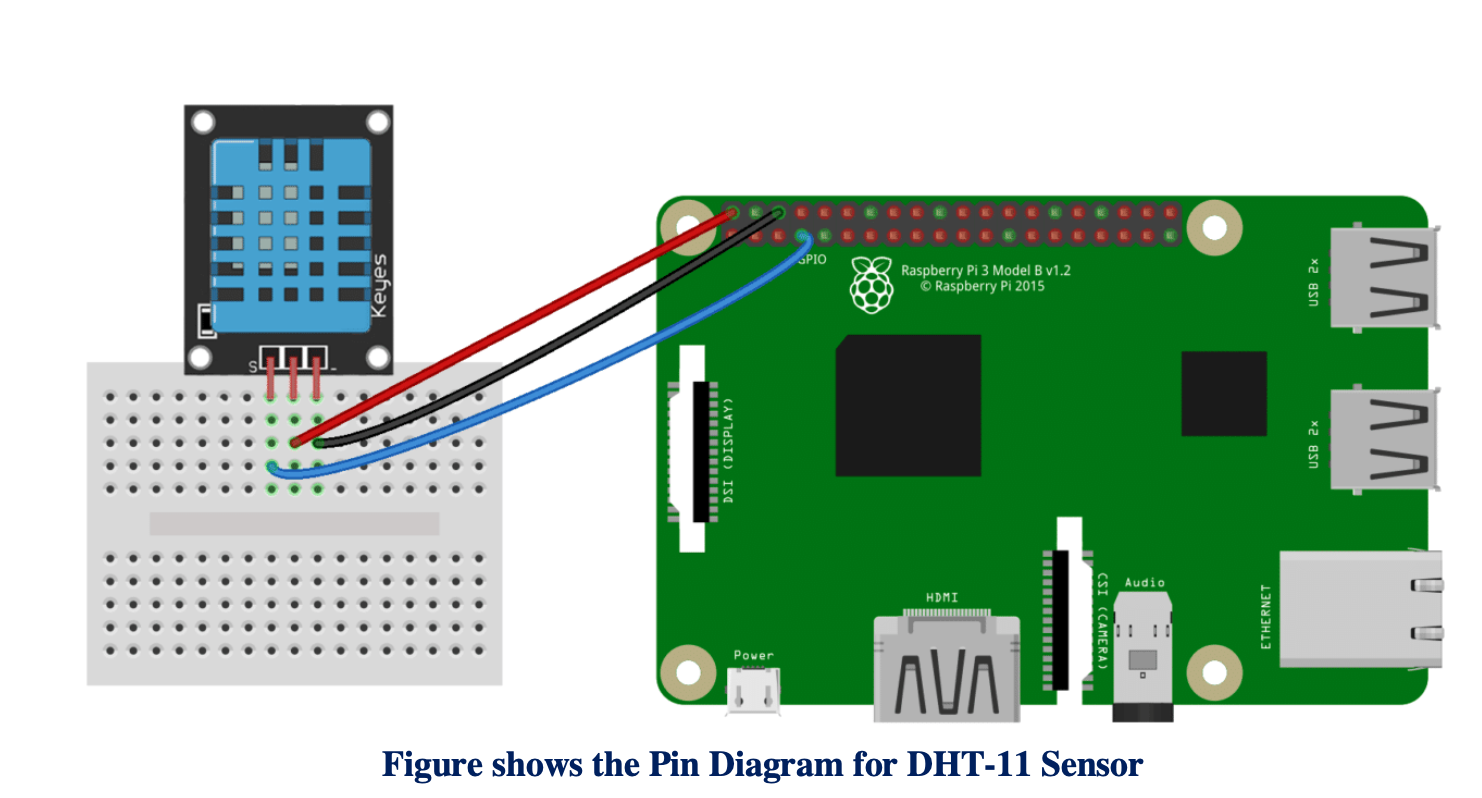
except KeyboardInterrupt:

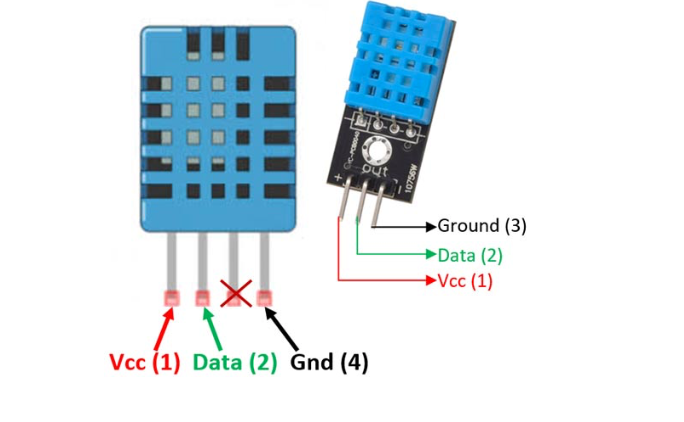
print("Measurement stopped by User")

GPIO.cleanup()

# Interfacing DHT-11/ DHT-22 Sensor with Raspberry Pi [Lab 2B]

**Objective:** To learn how to interface DHT-11/DHT-22 Sensor with Raspberry pi. Later we shall add SQLITE Database and try to upload the code on Cloud.





# Please follow complete Steps on how to install the Library on following link

# <https://github.com/adafruit/Adafruit_Python_DHT>

**Installation Steps: -**

Python library to read the DHT series of humidity and temperature sensors on a Raspberry Pi or Beaglebone Black. Designed specifically to work with the Adafruit DHT series sensors ----> <https://www.adafruit.com/products/385> Currently the library is tested with Python 2.6, 2.7, 3.3 and 3.4. It should work with Python greater than 3.4, too Dependencies

For all platforms (Raspberry Pi and Beaglebone Black) make sure your system is able to compile and download Python extensions with pip:

On Raspbian or Beaglebone Black's Debian/Ubuntu image you can ensure your system is ready by running one or two of the following sets of commands

Python 2: sudo apt-get update sudo apt-get install python-pip sudo python -m pip install --upgrade pip setuptools wheel

Python 3: sudo apt-get update sudo apt-get install python3-pip sudo python3 -m pip install --upgrade pip setuptools wheel

Install with pip Use pip to install from PyPI.

Python 2: sudo pip install Adafruit\_DHT

Python 3: sudo pip3 install Adafruit\_DHT

Compile and install from the repository First download the library source code from the GitHub releases page, unzipping the archive, and execute: Python 2: cd Adafruit\_Python\_DHT sudo python setup.py install

Python 3: cd Adafruit\_Python\_DHT sudo python3 setup.py install

You may also git clone the repository if you want to test an unreleased version: git clone <https://github.com/adafruit/Adafruit_Python_DHT.git>

# Task 1: Read the sensor

# Code:

# import Adafruit\_DHT

# # Example using a Raspberry Pi with DHT sensor

# # connected to GPIO23.

# pin = 4

# sensor = Adafruit\_DHT.DHT22

# humidity, temperature = Adafruit\_DHT.read\_retry(sensor, pin)

# if humidity is not None and temperature is not None:

# print('Temp={0:0.1f}\*C Humidity={1:0.1f}%'.format(temperature, humidity))

# else:

# print('Failed to get reading. Try again!')

Run the above Code and if you get the Temperature and humidity congregations and let’s move ahead. Let’s try to read the sensor value continuously we just need to add a while loop

# Task 2: Read the sensor value continuously.[¶](http://localhost:8888/notebooks/Desktop/IoT%20Fall%202019/LAB%202/Ipython%20Notebook/Lab%202B.ipynb#Task-2:-Read-the-sensor-value-continuously.)

# Code:

# import Adafruit\_DHT

# # Example using a Raspberry Pi with DHT sensor

# # connected to GPIO23.

# pin = 4

# sensor = Adafruit\_DHT.DHT22

# while 1:

# humidity, temperature = Adafruit\_DHT.read\_retry(sensor, pin)

# if humidity is not None and temperature is not None:

# print('Temp={0:0.1f}\*C Humidity={1:0.1f}%'.format(temperature, humidity))

# if humidity >34:

# print("Alert ")

# else:

# print('Failed to get reading. Try again!')

# Task 3: Try to create a UI where you can display Temperature and Humidity continuously

# import tkinter as tk

# import numpy as np

# import random

# import time

# import datetime

# import threading

# import Adafruit\_DHT

# pin = 4

# sensor = Adafruit\_DHT.DHT22

# def tick():

# time2=time.strftime('%H:%M:%S')

# clock.config(text=time2)

# clock.after(200,tick)

# def get\_data():

# threading.Timer(5, get\_data).start()

# humidity, temperature = Adafruit\_DHT.read\_retry(sensor, pin)

# if humidity is not None and temperature is not None:

# print('Temp={0:0.1f}\*C Humidity={1:0.1f}%'.format(temperature, humidity))

# l\_display.config(text = temperature)

# else:

# print('Failed to get reading. Try again!')

# return temperature

# mainwindow = tk.Tk()

# mainwindow.geometry('640x340')

# mainwindow.title("Sensor Data Live Feed ")

# clock=tk.Label(mainwindow,font=("Arial",30), bg='green',fg="white")

# clock.grid(row=0, column=0, padx=10, pady=10, sticky="nsew")

# l\_m=tk.Label(mainwindow,text="Sensor Data ",font=("Arial",30),fg="Black")

# l\_m.grid(row=0,column=1, padx=10, pady=10, sticky="nsew")

# l\_t=tk.Label(mainwindow, text="Temperature C",font=("Arial",25))

# l\_t.grid(row=1,column=0, padx=10, pady=10, sticky="nsew")

# l\_display=tk.Label(mainwindow,font=("Arial",25),fg="red")

# l\_display.grid(row=1,column=1, padx=10, pady=10, sticky="nsew")

# tick()

# get\_data()

# mainwindow.mainloop()

# Task 4: Try to Log the value on Database[¶](http://localhost:8888/notebooks/Desktop/IoT%20Fall%202019/LAB%202/Ipython%20Notebook/Lab%202B.ipynb#Task-4:-Try-to-Log-the-value-on-Database)

# import Adafruit\_DHT

# import sqlite3

# import threading

# import datetime

# def get\_time():

# my=datetime.datetime.now()

# data\_time = '{}:{}:{}'.format(my.hour,my.minute,my.second)

# data\_date='{}/{}/{}'.format(my.day,my.month,my.year)

# return data\_date,data\_time

# def my\_database(x, y,my\_time, my\_date):

# """

# :return: Nothing

# """

# # define the connection

# conn = sqlite3.connect("xyz.db")

# #define the cursor

# cursor = conn.cursor()

# # create Tables

# cursor.execute("""

# CREATE TABLE IF NOT EXISTS my\_table (id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,

# temperature TEXT,

# humidity TEXT,

# m\_date TEXT,

# m\_time TEXT)""")

# # Add data to database

# cursor.execute("""

# INSERT INTO my\_table (temperature,humidity,m\_date,m\_time) VALUES (?, ?, ?, ?)""",(x, y,my\_time, my\_date))

# # perform 3C

# #COMMIT, CLOSE CLOSE

# conn.commit()

# cursor.close()

# conn.close()

# def main():

# threading.Timer(10, main).start()

# pin = 4

# sensor = Adafruit\_DHT.DHT22

# humidity, temperature = Adafruit\_DHT.read\_retry(sensor, pin)

# if humidity is not None and temperature is not None:

# print('Temp={0:0.1f}\*C Humidity={1:0.1f}%'.format(temperature, humidity))

# data\_date,data\_time = get\_time()

# print('Date',data\_date)

# my\_database(temperature, humidity,data\_time,data\_date)

# print("data was written on database T{} H{}".format(temperature,humidity))

# else:

# print('Failed to get reading. Try again!')

# if \_\_name\_\_ == '\_\_main\_\_':

# while 1:

# main()

# Objective Learn to Interface Camera Module



# Code:

# import time

# from picamera import PiCamera

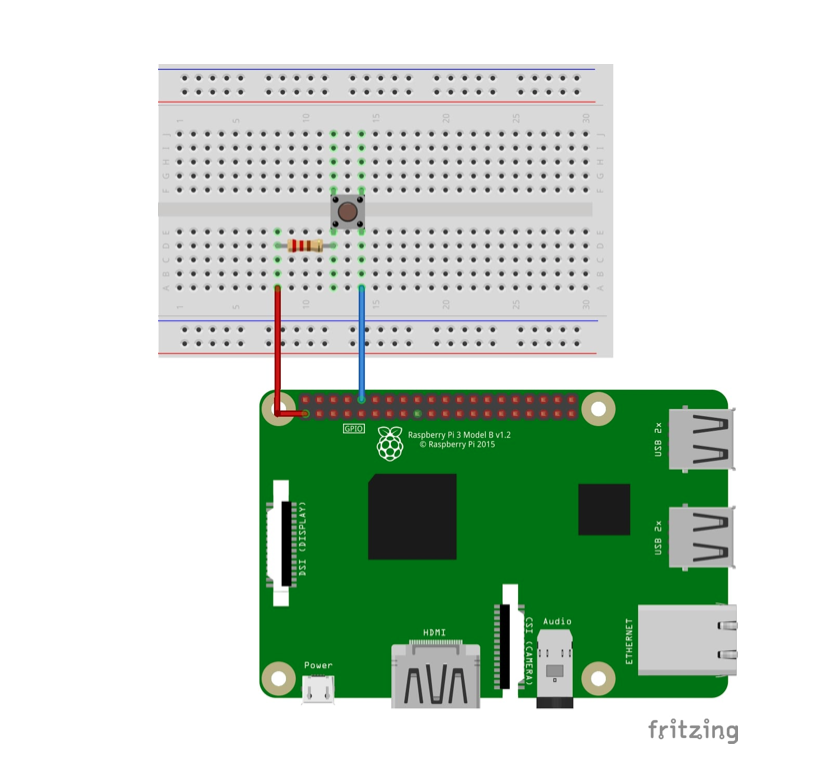
# camera=PiCamera()

# #camera.start\_preview()

# time.sleep(2)

# camera.capture("yo.jpg")

# Modify the Code to whenever you press button it takes Photo



# Code

# import time

# from picamera import PiCamera

# from time import sleep # Library will let us put in delays

# import RPi.GPIO as GPIO # Import the RPi Library for GPIO pin control

# button1\_pin=10 # Button 1 is connected to physical pin 12

# GPIO.setmode(GPIO.BOARD) # Use Physical Pin Numbering Scheme

# GPIO.setup(button1\_pin,GPIO.IN,pull\_up\_down=GPIO.PUD\_UP)

# # Make button1\_pin an input, Activate Pull UP Resistor

# while(1): # Create an infinite Loop

# input1=GPIO.input(button1\_pin)

# if input1==0:

# sleep(.1)

# print ('Button 1 Pressed')

# # Look for button 1 press # Delay

# # Notify User

# camera=PiCamera()

# #camera.start\_preview()

# time.sleep(2)

# camera.capture("soumil.jpg")