EX NO.:	SENSING DOOR'S STATE USING RASPBERRY PI AND SHOW OUTPUT IN WEBPAGE
Date:	

AIM:

To write a program for develop an application using raspberry pi and flask.

COMPONENTS REQUIRED:

components	nos
Rasepberry pi	1
Ultrasonic sensor	1
SD card	1
Resistor(10k)	2

PROCEDURE:

Step 1:Download the Raspberry pi imager software from the official Raspberry Pi website

Step 2:install the raspberry pi Imager software and run it.

Step 3:place the raspberry pi board and all the components in the workspace

step 4:connect the raspberry pi with the ultrasonic sensor

Step 5:connect the vcc pin to the board 2nd pin

Step 6:connect the gnd pin to the ground pin

step 7:connect the trig pin to the board 16th pin with the 1 kilo ohm resistor

Step 8:connect the echo pin to the board 18th pin with the 2 kilo ohm resistor

step 9:open the python

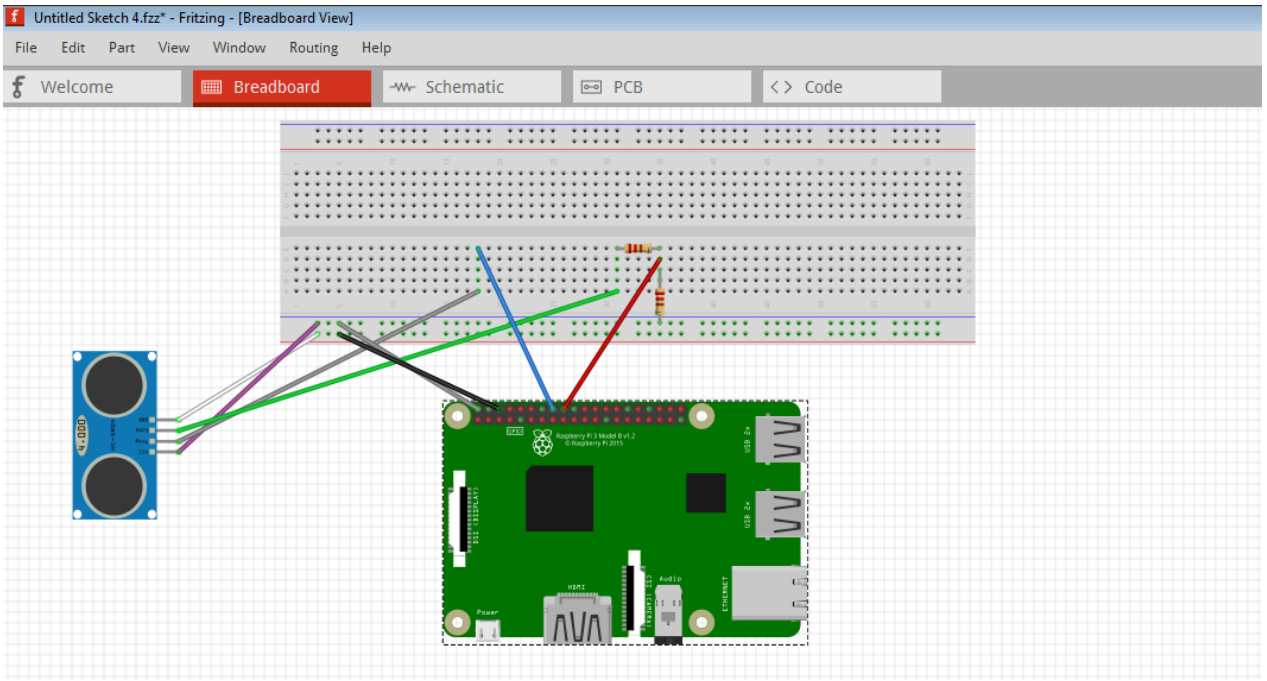
Step 10:initialize the program in a setup and type the program as a python program

And also type the html program for webpage

Step 11:Run the program

Step 12:click the link in the output and you should see the webpage on the web browser according to the figure

Schematic diagram:



Program:

Distance.py

```
import RPi.GPIO as GPIO
import time
from flask import Flask, render_template
import threading

# Set up GPIO mode and pins
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

TRIG = 23
ECHO = 24

# Flask web app setup
app = Flask(__name__)

# Global variable to store the distance and door status
distance = 0
door_status = ""

# Set up the GPIO pins
GPIO.setup(TRIG, GPIO.OUT)
GPIO.setup(ECHO, GPIO.IN)
GPIO.output(TRIG, False)

# Allow the sensor to settle
```

```
print("Waiting For Sensor To Settle")
time.sleep(2)

def measure_distance():
    global distance, door_status
    while True:
        # Send the pulse to the TRIG pin
GPIO.output(TRIG, True)
time.sleep(0.0001)  # Trigger pulse duration (10us)
GPIO.output(TRIG, False)

        # Wait for the ECHO pin to go HIGH
        while GPIO.input(ECHO) == 0:
pulse_start = time.time()

        # Wait for the ECHO pin to go LOW
        while GPIO.input(ECHO) == 1:
pulse_end = time.time()

        # Calculate the pulse duration
pulse_duration = pulse_end - pulse_start

        # Calculate the distance (speed of sound = 34300 cm/s)
        distance = pulse_duration * 17150  # Distance in cm
        distance = round(distance, 2)

        # Update the door status based on the distance
```

```
        if distance < 10:
door_status = "Door Closed"
        else:
door_status = "Door Opened"

        # Print the measured distance (for debugging)
print("Distance:", distance, "cm")
        print(door_status)

        # Add a short delay before the next measurement
time.sleep(1)

@app.route('/')
def index():
    # Serve the HTML page with the current distance and door status
    return render_template('home.html', distance=distance, door_status=door_status)

if __name__ == "__main__":
    # Start the distance measurement in a separate thread to run in the background
    thread = threading.Thread(target=measure_distance)
thread.daemon = True  # Daemonize the thread to automatically exit when the main program ends
thread.start()

    # Start the Flask web server
app.run(host='0.0.0.0', port=5000)
```

Home.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Distance Measurement</title>
<style>
        body {
            font-family: Arial, sans-serif;
            text-align: center;
            margin-top: 50px;
        }
        h1 {
            color: #4CAF50;}
        .status {
            font-size: 24px;
            color: #FF5733;
        }
</style>
</head>
<body>
<h1>Distance Measurement</h1>
<p>Current Distance: {{ distance }} cm</p>
<p class="status">{{ door_status }}</p>
</body>
</html>
```

Output:

Result:

Thus the above program was successfully executed and the output is verified.