

TRAFFIC MANAGEMENT SYSTEM PHASE 5

PYTHON PROGRAM FOR TRAFFIC PREDICTION

```
import machine
import utime

# GPIO pins for the HC-SR04 sensor
trigger_pin = machine.Pin(2, machine.Pin.OUT) # Connect to the sensor's trigger pin
echo_pin = machine.Pin(3, machine.Pin.IN) # Connect to the sensor's echo pin

# Traffic light control pins (simulated)
red_light = machine.Pin(10, machine.Pin.OUT)
yellow_light = machine.Pin(11, machine.Pin.OUT)
green_light = machine.Pin(12, machine.Pin.OUT)

# Function to measure distance using the HC-SR04 sensor
def measure_distance():
    trigger_pin.value(0)
    utime.sleep_us(2)
    trigger_pin.value(1)
    utime.sleep_us(10)
    trigger_pin.value(0)
```

```
while echo_pin.value() == 0:
    pulse_start = utime.ticks_us()

while echo_pin.value() == 1:
    pulse_end = utime.ticks_us()

pulse_duration = utime.ticks_diff(pulse_end, pulse_start)
distance = (pulse_duration * 0.0343) / 2 # Speed of sound is approximately 343 meters per second

return distance
```

Traffic light control function

```
def control_traffic_lights(distance):
    if distance < 20: # If a vehicle is very close
        red_light.value(0)
        yellow_light.value(1)
        green_light.value(0)
    elif 20 <= distance < 40: # If a vehicle is moderately close
        red_light.value(1)
        yellow_light.value(0)
        green_light.value(0)
    else: # If no vehicle is detected
        red_light.value(0)
        yellow_light.value(0)
        green_light.value(1)
```

```
while True:
```

```
    distance = measure_distance()
```

```
    # Control traffic lights based on the distance measurements
```

```
    control_traffic_lights(distance)
```

```
    # For simulation purposes, print the distance and the traffic light state
```

```
    print("Distance: {:.2f} cm".format(distance))
```

```
    utime.sleep(2) # Wait for a few seconds before taking the next measurement
```

OUTPUT FOR TRAFFIC PREDICTION

The screenshot displays a Raspberry Pi simulation interface. On the left, a code editor shows a Python script named `main.py` that configures GPIO pins for an HC-SR04 sensor and traffic lights. The code includes comments for pin connections and a function to calculate distance using the speed of sound. The right side of the interface features a 'Simulation' tab with a visual representation of the hardware. A breadboard circuit diagram shows the HC-SR04 sensor connected to a Raspberry Pi via jumper wires. Below the diagram, a terminal window displays a series of distance measurements in centimeters, ranging from 403.32 to 403.73. The top right corner of the simulation window shows a timer at 00:46.432 and a 100% CPU usage indicator.

```
main.py diagram.json
1 import machine
2 import utime
3
4 # GPIO pins for the HC-SR04 sensor
5 trigger_pin = machine.Pin(2, machine.Pin.OUT) # Connect to the sensor's trigger pin
6 echo_pin = machine.Pin(3, machine.Pin.IN)      # Connect to the sensor's echo pin
7
8 # Traffic light control pins (simulated)
9 red_light = machine.Pin(10, machine.Pin.OUT)
10 yellow_light = machine.Pin(11, machine.Pin.OUT)
11 green_light = machine.Pin(12, machine.Pin.OUT)
12
13 # Function to measure distance using the HC-SR04 sensor
14 def measure_distance():
15     trigger_pin.value(0)
16     utime.sleep_us(2)
17     trigger_pin.value(1)
18     utime.sleep_us(10)
19     trigger_pin.value(0)
20
21     while echo_pin.value() == 0:
22         pulse_start = utime.ticks_us()
23
24     while echo_pin.value() == 1:
25         pulse_end = utime.ticks_us()
26
27     pulse_duration = utime.ticks_diff(pulse_end, pulse_start)
28     distance = (pulse_duration * 0.0343) / 2 # Speed of sound is approximately 343 meters per second
29
30     return distance
31
32 # Traffic light control function
33 def control_traffic_lights(distance):
34     if distance < 20: # If a vehicle is very close
35         red_light.value(0)
36         yellow_light.value(1)
```

Simulation

00:46.432 100%

Distance: 403.52 cm
Distance: 403.32 cm
Distance: 403.71 cm
Distance: 403.51 cm
Distance: 403.49 cm
Distance: 403.52 cm
Distance: 403.69 cm
Distance: 403.30 cm
Distance: 403.32 cm
Distance: 403.69 cm
Distance: 403.73 cm

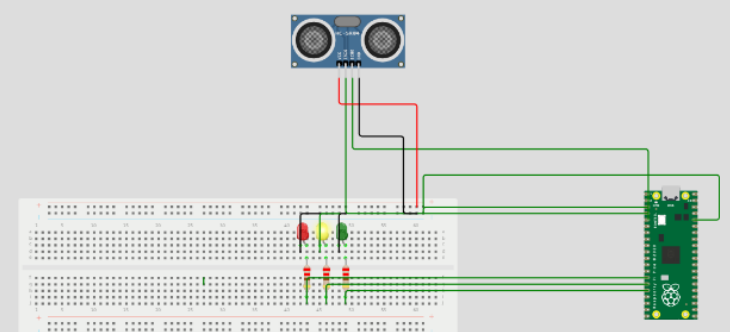
main.pydiagram.json

```
31
32 # Traffic light control function
33 def control_traffic_lights(distance):
34     if distance < 20: # If a vehicle is very close
35         red_light.value(0)
36         yellow_light.value(1)
37         green_light.value(0)
38     elif 20 <= distance < 40: # If a vehicle is moderately close
39         red_light.value(1)
40         yellow_light.value(0)
41         green_light.value(0)
42     else: # If no vehicle is detected
43         red_light.value(0)
44         yellow_light.value(0)
45         green_light.value(1)
46
47 while True:
48     distance = measure_distance()
49
50     # Control traffic lights based on the distance measurements
51     control_traffic_lights(distance)
52
53     # For simulation purposes, print the distance and the traffic light state
54     print("Distance: {:.2f} cm".format(distance))
55     utime.sleep(2) # Wait for a few seconds before taking the next measurement
```

Simulation

⏮⏪⏸

⌚ 06:56.074 🔋 99%



```
Distance: 2.11 cm
Distance: 1.90 cm
Distance: 1.90 cm
Distance: 1.90 cm
Distance: 1.90 cm
Distance: 1.90 cm
Distance: 1.90 cm
Distance: 2.11 cm
Distance: 2.11 cm
Distance: 2.11 cm
Distance: 2.11 cm
Distance: 2.11 cm
```

main.py

diagram.json

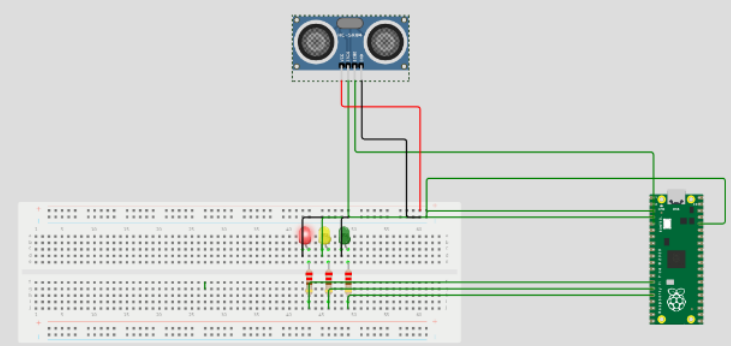
```
31
32 # Traffic light control function
33 def control_traffic_lights(distance):
34     if distance < 20: # If a vehicle is very close
35         red_light.value(0)
36         yellow_light.value(1)
37         green_light.value(0)
38     elif 20 <= distance < 40: # If a vehicle is moderately close
39         red_light.value(1)
40         yellow_light.value(0)
41         green_light.value(0)
42     else: # If no vehicle is detected
43         red_light.value(0)
44         yellow_light.value(0)
45         green_light.value(1)
46
47 while True:
48     distance = measure_distance()
49
50     # Control traffic lights based on the distance measurements
51     control_traffic_lights(distance)
52
53     # For simulation purposes, print the distance and the traffic light state
54     print("Distance: {:.2f} cm".format(distance))
55     time.sleep(2) # Wait for a few seconds before taking the next measurement
```

Simulation

05:22.186 99%

Editing Ultrasonic Distance Sensor

Distance: 31cm



Distance: 31.56 cm
Distance: 31.28 cm
Distance: 31.28 cm
Distance: 31.25 cm
Distance: 31.30 cm
Distance: 31.26 cm
Distance: 31.52 cm
Distance: 31.28 cm
Distance: 31.08 cm
Distance: 31.25 cm
Distance: 31.08 cm