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
```

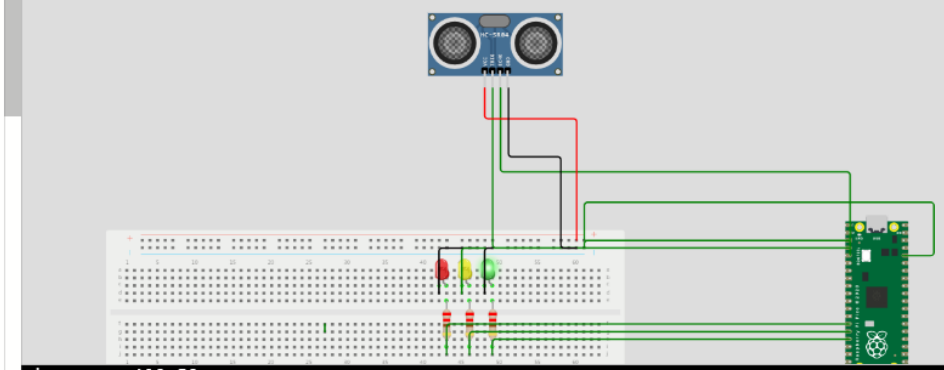
OUTPUT FOR TRAFFIC PREDICTION

main.py | diagram.json

```python
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 pe

    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)
```

Simulation

00:46.432 100%



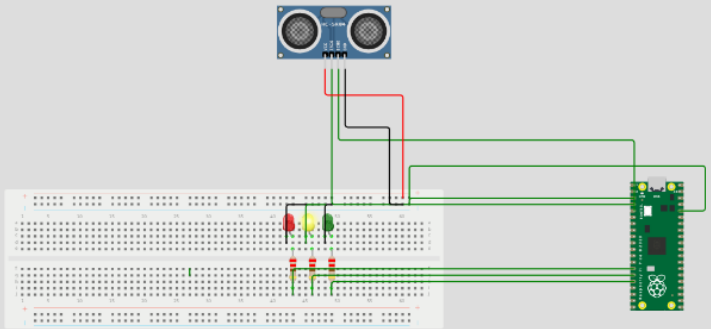
```
Distance: 403.52 cm
Distance: 403.32 cm
Distance: 403.71 cm
Distance: 403.51 cm
Distance: 403.49 cm
Distance: 403.52 cm
Distance: 403.69 cm
Distance: 403.30 cm
Distance: 403.32 cm
Distance: 403.69 cm
Distance: 403.73 cm
```

main.py | diagram.json

```python
# 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
```

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
```

WOKWi SAVE SHARE Docs SIGN UP

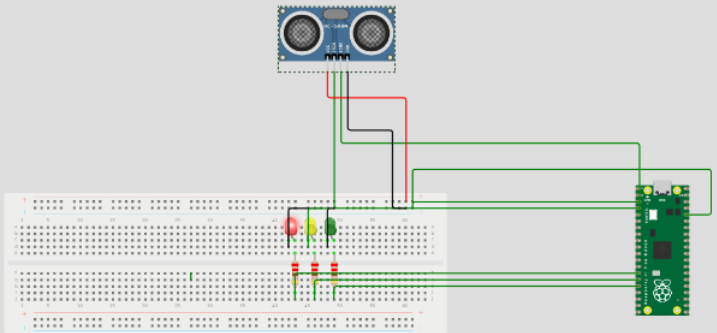
main.py diagram.json

```python
# 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
```

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
```