

# TRAFFIC MANAGEMENT SYSTEM PHASE 5

# PROJECT DEFINITION

- Internet of Things(IOT) enabled intelligent traffic management system can solve pertinent issues by leveraging technologies like wireless connectivity and intelligent sensors.
- With increase in the number of cars queues at toll booths on highways have become common place and while automatic tools using RFID(radio frequency identification) tags have reduced waiting times further improvements are possible using IOT technology.

# OBJECTIVES

- Estimation of traffic bulkiness performed using real time video feed.
- Vehicle recognition in order to differentiate between emergency and non emergency vehicles.
- It explore the role of IOT in traffic management ,the challenges it can solve and essential technologies to develop an intelligent system and it also explain the how a city government can implement it to offer a good citizen experience

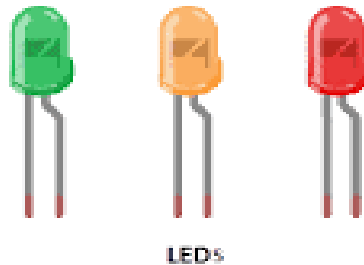
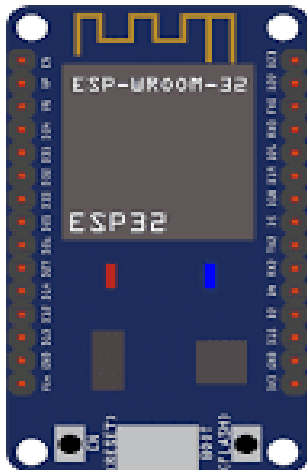
# ULTRASONIC DISTANCE SENSOR –HC-SR04(5V)

- This is the HC-SR04 ultrasonic distance sensor. This economical sensor provides 2cm to 400cm of non-contact measurement functionality with a ranging accuracy that can reach up to 3mm. Each HC-SR04 module includes an ultrasonic transmitter, a receiver and a control circuit.
- There are only four pins that you need to worry about on the HC-SR04: VCC (Power), Trig (Trigger), Echo (Receive), and GND (Ground). You will find this sensor very easy to set up and use for your next range-finding project!
- This sensor has additional control circuitry that can prevent inconsistent "bouncy" data depending on the application.

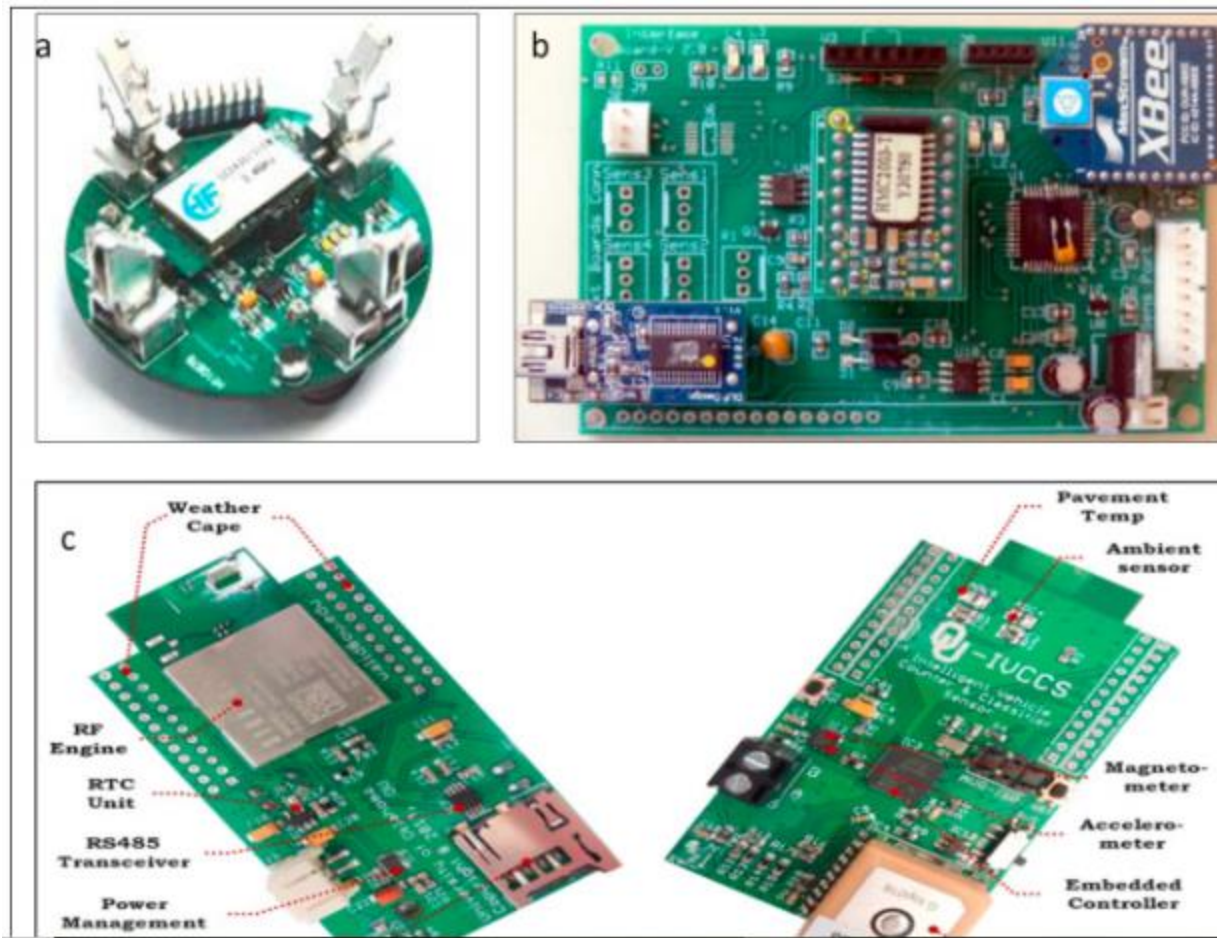


# ESP32 MICROCONTROLLER

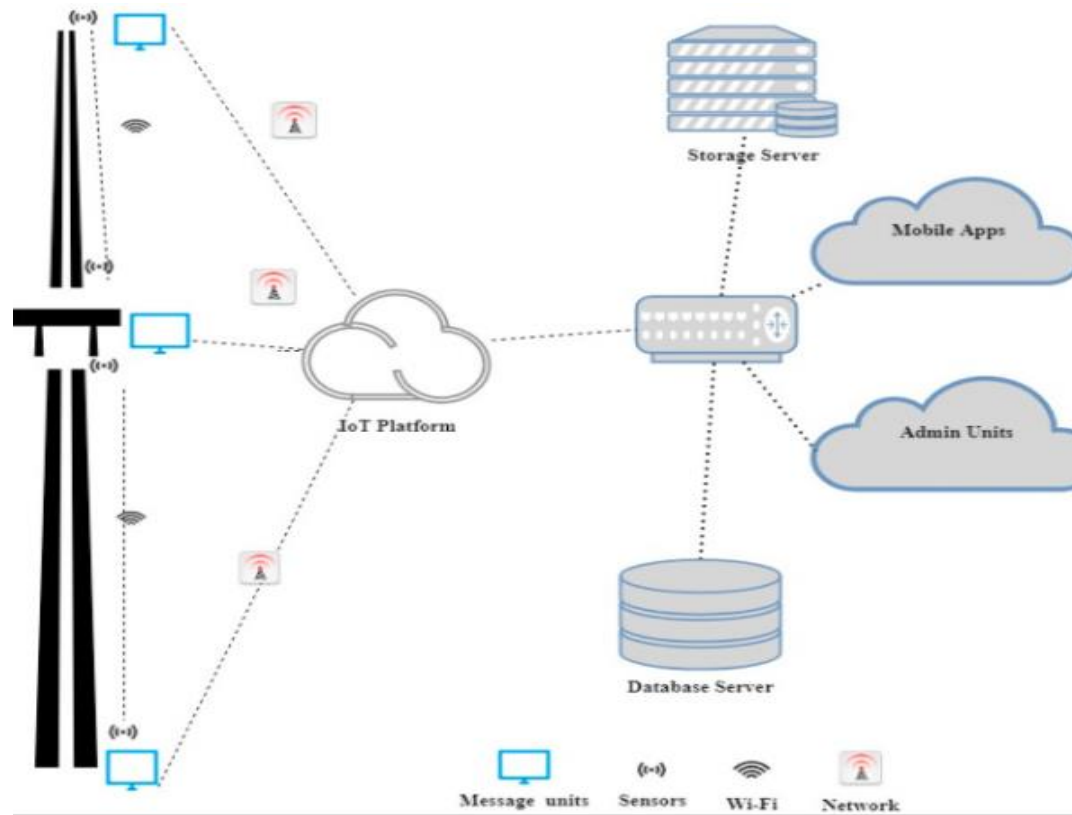
The ESP32 is a low-cost, low-power microcontroller with built-in Wi-Fi and Bluetooth capabilities. It is a popular choice for IoT projects and is commonly used for a variety of applications such as home automation, wireless control, and sensor data logging. The ESP32 features a dual-core processor, a rich set of peripherals, and support for a wide range of protocols. It can be programmed using the Arduino IDE and various other programming languages such as C, C++, and MicroPython



# SENSORS IN TRAFFIC PREDICTION



# MOBILE APPS TO DISPLAY REAL-TIME TRAFFIC INFORMATION



# 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))
```

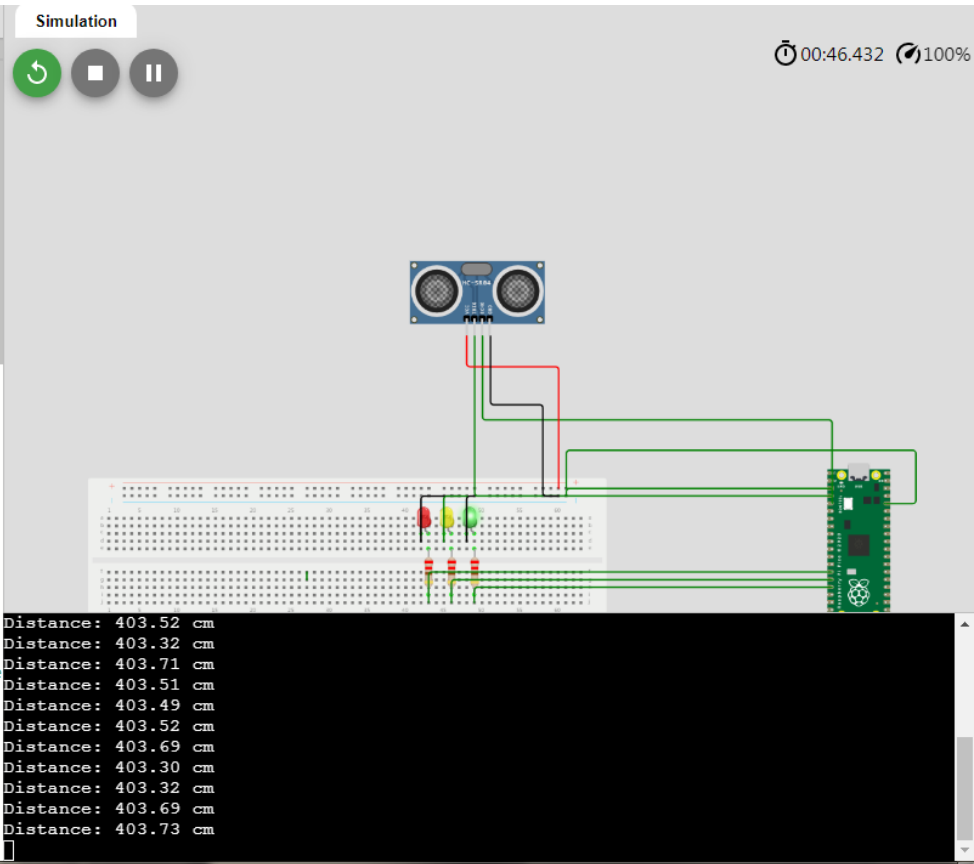
```
    utime.sleep(2) # Wait for a few seconds before taking the next measurement
```

# OUTPUT FOR TRAFFIC PREDICTION

```

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)

```



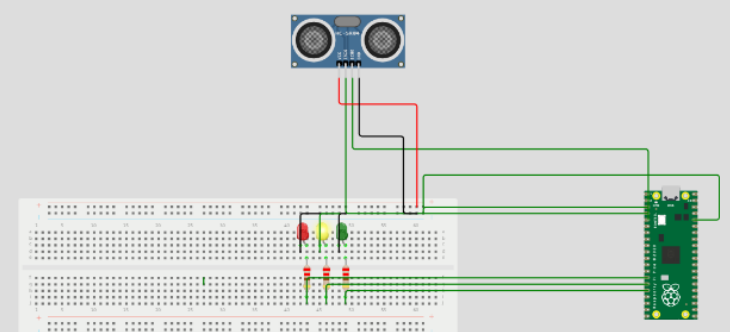
main.pydiagram.json

```
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
```

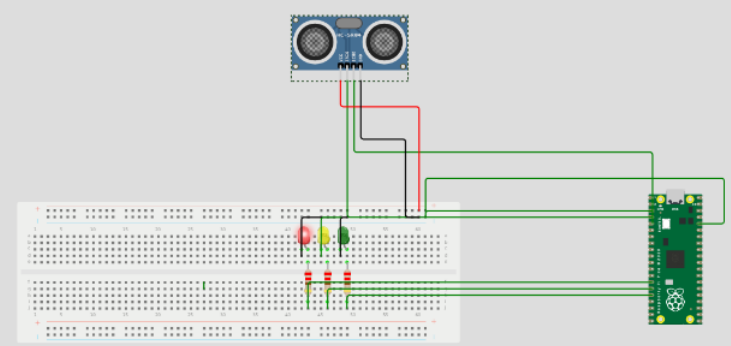
```
main.py diagram.json
31
32 # Traffic light control function
33 def control_traffic_lights(distance):
34     if distance < 20: # If a vehicle is very close
35         red_light.value(0)
36         yellow_light.value(1)
37         green_light.value(0)
38     elif 20 <= distance < 40: # If a vehicle 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
```

# CONCLUSION

- We designed a smart traffic control system using ESP32 to solve the problem of congestion at the intersection of the Doral Moalemen region, working to prevent traffic jam and reduce time, Using Arduino mega, ultrasonic sensor, and a camera esp32, the suggested technique analyses and manages everyday traffic at a three-line intersection. Humidity and temperature variations do not affect the system's accuracy.
- Furthermore, the suggested system achieves three-line intersection sync and implements a balance between the number of vehicles on each side and the green light. In the case of traffic violation, the camera will capture the car number and send it to the database by using telegram.