Phase3:Development part1

Project title	**IOT Traffic Management System**
Name	**R.Supraja**
Team id	**5237**
Team Name	**Proj_204181_Team_2**
College code name	**9238- Mangayarkarasi college of Engineering**
Group	**5**
Github respository link	**https://github.com/Supraja1508/Ibm-Naanmudhalvan-IOT.git**

IOT-TRAFFIC MANAGEMENT SYSTEM

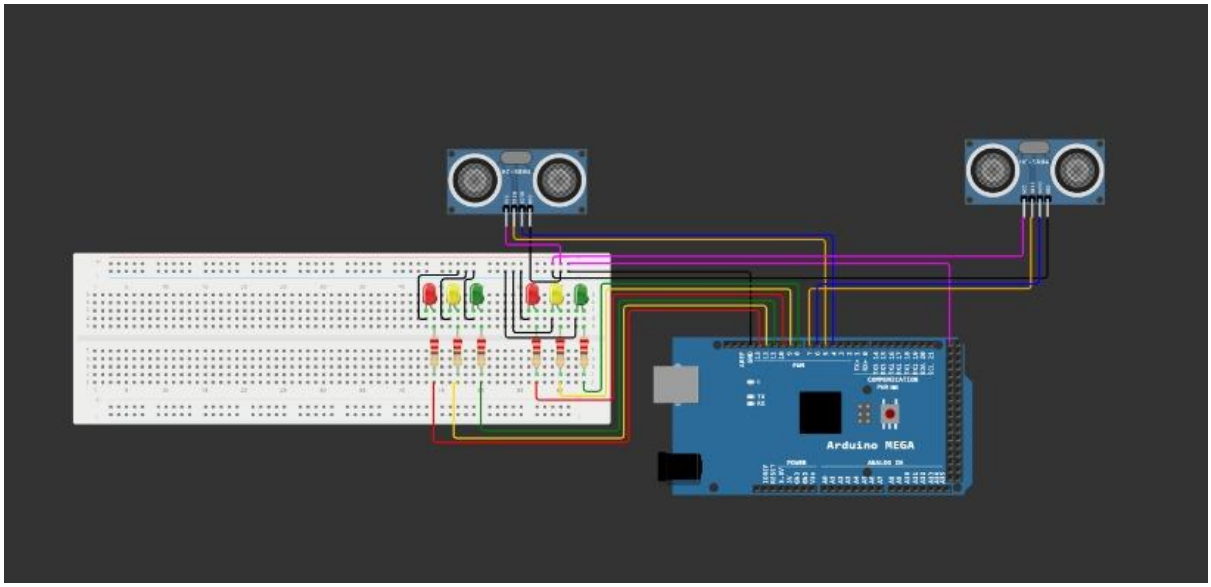
1.ABSTRACT:

Traffic management systems play a critical role in addressing the growing challenges of urban congestion, safety, and sustainability. This abstract provides an overview of key aspects and innovations in traffic management.Effective traffic management involves the integration of technologies such as smart traffic lights, real-time data analysis, and to optimize traffic flow and reduce traffic congestion.This IoT-based Traffic Management System (IoT-TMS) leverages real-time data acquisition and analysis from various sensors and devices deployed across urban road networks. These sensors include cameras, GPS trackers, vehicle detectors, and environmental sensors. The collected data is transmitted to a central control system through wireless communication protocols.The central control system processes this data to monitor traffic conditions, detect congestion, and optimize traffic signal timings in real time. Additionally, IoT-TMS provides valuable insights for urban planners and policymakers by analyzing historical traffic patterns and suggesting infrastructure improvements.

2.Proposal of the Project:

Development and implementation of Traffic Management System Deploy a network of sensors and devices, including cameras, GPS trackers, vehicle detectors, and environmental sensors, across the road network.Develop user-friendly mobile apps to provide real-time traffic information, navigation assistance, and alerts to drivers.

3.Hardware setup:



Hardware Components:

1.Aurdino:

- Aurdino serves as the main controllers and runs the python program

2. Ultrasonic Sensors:

- Ultrasonic sensors (e.g., HC-SR04) are used to measure distances and determine vehicles. You have configured two ultrasonic sensors in the program.

3. Wi-Fi Module:

- While not explicitly mentioned in the code, built-in Wi-Fi module is used to connect the aurdino to the Wi-Fi network.

4. Jumper Wires:

- Jumper wires are used to establish connections between the ESP32's GPIO pins and the sensors.

5.Breadboard:

- Breadboard is used to connect the sensors and aurdino.

Hardware components:

Ultrasonic Sensors (HC-SR04):

- Each ultrasonic sensor (e.g., HC-SR04) requires four connections:
- VCC (Voltage): Connect to a 5V pin on the ESP32 for power.
- GND (Ground): Connect to a ground (GND) pin on the ESP32.

- Trig (Trigger): Connect to the GPIO pins defined in trig_pins (pins 2, 18, and 22 in your code).
- Echo: Connect to the GPIO pins defined in echo_pins (pins 4, 19, and 23 in your code).

Wi-Fi Module:

- You should ensure that your Aurdino is connected to a Wi-Fi network, either using built-in Wi-Fi or an external Wi-Fi module. The program relies on this network connection to send data to Firebase.

Power Supply:

- The Aurdino and sensors should be powered appropriately. The ESP32 can be powered through a USB power supply, and sensors may need a separate 5V supply. Ensure that all components share a common ground.

Program:

```
#include&lt;TimerOne.h&gt;
Int signal1[] = {23, 25, 27};
Int signal2[] = {46, 48, 50};
Int signal3[] = {13, 12, 11};
Int signal4[] = {10, 9, 8};
Int redDelay = 5000;
Int yellowDelay = 2000;
Volatile int triggerpin1 = 31;
Volatile int echopin1 = 29;
Volatile int triggerpin2 = 44;
Volatile int echopin2 = 42;
Volatile int triggerpin3 = 7;
Volatile int echopin3 = 6;
Volatile int triggerpin4 = 5;
Volatile int echopin4 = 4;
Volatile long time; // Variable for storing the time traveled
Volatile int S1, S2, S3, S4; // Variables for storing the distance covered
Int t = 5; // distance under which it will look for vehicles.
Void setup(){
Serial.begin(115200);
Timer1.initialize(100000); //Begin using the timer. This function must be called first.
“microseconds” is
the period of time the timer takes.
Timer1.attachInterrupt(softInterr); //Run a function each time the timer period finishes.
// Declaring LED pins as output
For(int i=0; i&lt;3; i++){
pinMode(signal1[i], OUTPUT);
pinMode(signal2[i], OUTPUT);
pinMode(signal3[i], OUTPUT);
pinMode(signal4[i], OUTPUT);

}
// Declaring ultrasonic sensor pins as output
```

```
pinMode(triggerpin1, OUTPUT);
pinMode(echopin1, INPUT);
pinMode(triggerpin2, OUTPUT);
pinMode(echopin2, INPUT);
pinMode(triggerpin3, OUTPUT);
pinMode(echopin3, INPUT);
pinMode(triggerpin4, OUTPUT);
pinMode(echopin4, INPUT);
}
Void loop()
{
// If there are vehicles at signal 1
If(S1&lt;t)
{
Signal1Function();
}
// If there are vehicles at signal 2
If(S2&lt;t)
{
Signal2Function();
}
// If there are vehicles at signal 3
If(S3&lt;t)
{
Signal3Function();
}
// If there are vehicles at signal 4

If(S4&lt;t)
{
Signal4Function();
}
}
// This is interrupt function and it will run each time the timer period finishes. The timer
period is set at
100 milli seconds.
Void softInterr()
{
// Reading from first ultrasonic sensor
digitalWrite(triggerpin1, LOW);
delayMicroseconds(2);
digitalWrite(triggerpin1, HIGH);
delayMicroseconds(10);
digitalWrite(triggerpin1, LOW);
time = pulseIn(echopin1, HIGH);
S1= time*0.034/2;
// Reading from second ultrasonic sensor
digitalWrite(triggerpin2, LOW);
```

```
delayMicroseconds(2);
digitalWrite(triggerpin2, HIGH);
delayMicroseconds(10);
digitalWrite(triggerpin2, LOW);
time = pulseIn(echopin2, HIGH);
S2= time*0.034/2;
// Reading from third ultrasonic sensor
digitalWrite(triggerpin3, LOW);
delayMicroseconds(2);
digitalWrite(triggerpin3, HIGH);

delayMicroseconds(10);
digitalWrite(triggerpin3, LOW);
time = pulseIn(echopin3, HIGH);
S3= time*0.034/2;
// Reading from fourth ultrasonic sensor
digitalWrite(triggerpin4, LOW);
delayMicroseconds(2);
digitalWrite(triggerpin4, HIGH);
delayMicroseconds(10);
digitalWrite(triggerpin4, LOW);
time = pulseIn(echopin4, HIGH);
S4= time*0.034/2;
// Print distance values on serial monitor for debugging
Serial.print("S1: ");
Serial.print(S1);
Serial.print(" S2: ");
Serial.print(S2);
Serial.print(" S3: ");
Serial.print(S3);
Serial.print(" S4: ");
Serial.println(S4);
}
Void signal1Function()
{
Serial.println("1");
Low();
// Make RED LED LOW and make Green HIGH for 5 seconds
digitalWrite(signal1[0], LOW);
digitalWrite(signal1[2], HIGH);

delay(redDelay);
// if there are vehicels at other signals
If(S2&lt;t || S3&lt;t || S4&lt;t)
{
// Make Green LED LOW and make yellow LED HIGH for 2 seconds
digitalWrite(signal1[2], LOW);
digitalWrite(signal1[1], HIGH);
```

```
delay(yellowDelay);
}
}
Void signal2Function()
{
Serial.println(“2”);
Low();
digitalWrite(signal2[0], LOW);
digitalWrite(signal2[2], HIGH);
delay(redDelay);

if(S1&lt;t || S3&lt;t || S4&lt;t)
{
digitalWrite(signal2[2], LOW);
digitalWrite(signal2[1], HIGH);
delay(yellowDelay);
}
}
Void signal3Function()
{
Serial.println(“3”);
Low();

digitalWrite(signal3[0], LOW);
digitalWrite(signal3[2], HIGH);
delay(redDelay);
if(S1&lt;t || S2&lt;t || S4&lt;t)
{
digitalWrite(signal3[2], LOW);
digitalWrite(signal3[1], HIGH);
delay(yellowDelay);
}
}
Void signal4Function()
{
Serial.println(“4”);
Low();
digitalWrite(signal4[0], LOW);
digitalWrite(signal4[2], HIGH);
delay(redDelay);
if(S1&lt;t || S2&lt;t || S3&lt;t)
{
digitalWrite(signal4[2], LOW);
digitalWrite(signal4[1], HIGH);
delay(yellowDelay);
}
}
// Function to make all LED’s LOW except RED one’s.
```

```
Void low()
{
For(int i=1; i&lt;3; i++)
{

digitalWrite(signal1[i], LOW);
digitalWrite(signal2[i], LOW);
digitalWrite(signal3[i], LOW);
digitalWrite(signal4[i], LOW);
}
For(int i=0; i&lt;1; i++)
{
digitalWrite(signal1[i], HIGH);
digitalWrite(signal2[i], HIGH);
digitalWrite(signal3[i], HIGH);
digitalWrite(signal4[i], HIGH);
}
}
```

This Arduino program appears to control a traffic signal system using ultrasonic sensors to detect the presence of vehicles at four different signal intersections (signal1, signal2, signal3, and signal4). Here's a brief description of the program's functionality:

Library Inclusion: The code includes the "TimerOne" library to manage time intervals.

Pin Declarations: Defines various pins for LEDs (signal1, signal2, signal3, signal4), ultrasonic sensors (triggerpin1, echopin1, triggerpin2, echopin2, triggerpin3, echopin3, triggerpin4, echopin4), and other variables like delay times.

Setup Function: This function runs once at the beginning and initializes pins, sets up the timer, and opens a serial connection for debugging.

Loop Function: This is the main program loop that repeatedly checks if there are vehicles at each signal and calls the appropriate function to control the traffic signal based on the detected vehicle presence.

Interrupt Function (**softInterr**): This function is triggered by a timer interrupt. It reads data from four ultrasonic sensors, calculates the distance to nearby objects, and stores this information in S1, S2, S3, and S4. It also prints these values to the serial monitor for debugging.

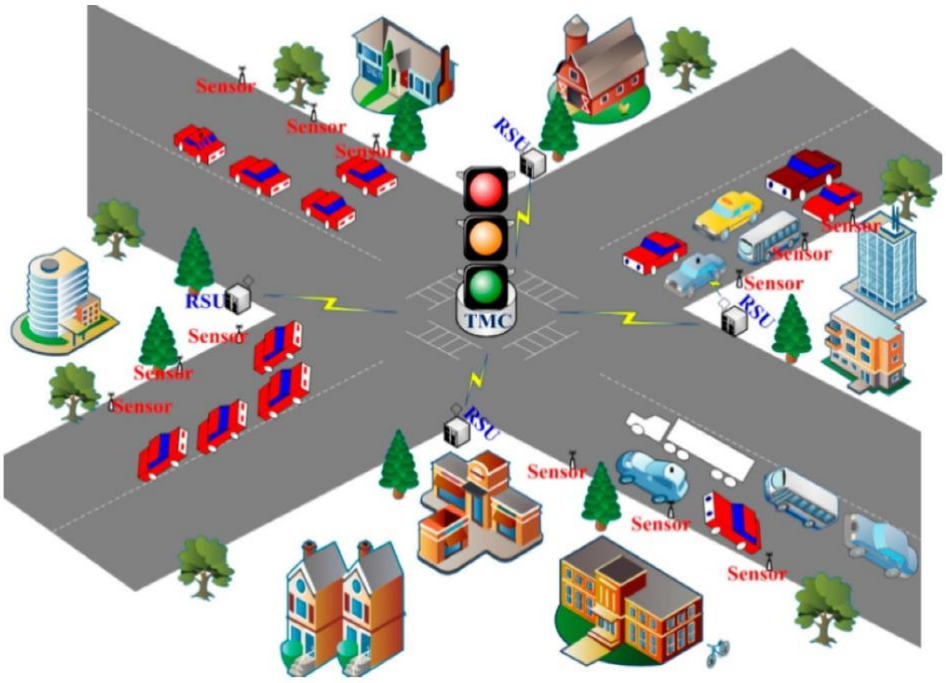
Signal Functions (**signal1Function**, **signal2Function**, **signal3Function**, **signal4Function**): These functions control the traffic signal behavior for each signal. They change the LED states based on whether vehicles are detected and follow a typical traffic light pattern (red, green, yellow).

Low Function: This function turns off all LEDs except the red ones for all signals.

The program essentially simulates a traffic signal system that adapts its timing based on the presence of vehicles detected by ultrasonic sensors. If there are vehicles waiting at a signal, it will give them a green light, otherwise, it follows a standard traffic light sequence with red and yellow lights. It also prints information about the detected distances for debugging purposes.

Please note that to operate this code successfully, you need to have the necessary hardware components and wiring set up, as the code relies on the interaction between the Arduino and the ultrasonic sensors and LEDs., triggerpin2, echopin2, triggerpin3, echopin3, triggerpin4, echopin4), and other variables.

Architecture of the project:



As per our design we plan to place the sensors in the side of the road with low noise which detect the vehicle Coming is the vehicle reach the sensor by 50 cm then the sensor will share the data thorough The micro-controller to the firebase that give the Signals to the vehicles if the road is empty in opposite direction.

Firebase Database:

Real time Database:

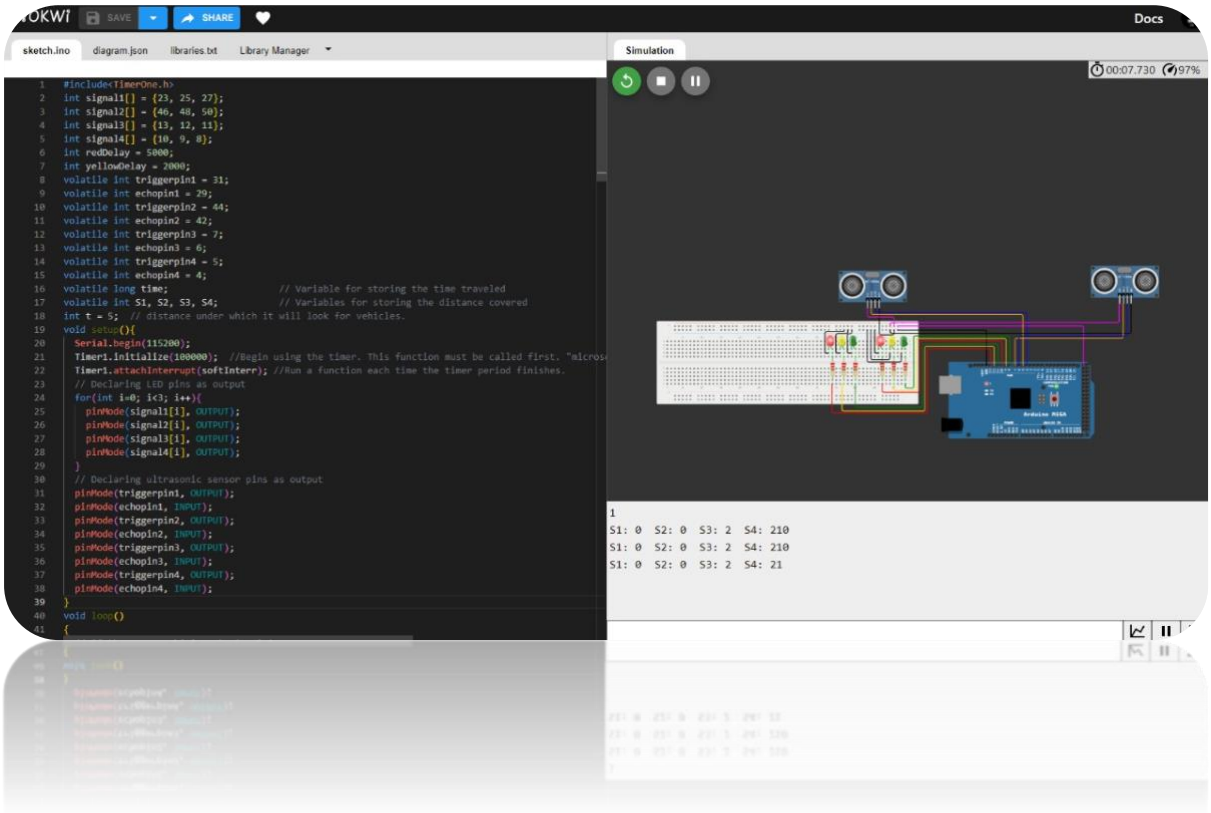
Firebase Realtime Database is a cloud-hosted NoSQL database provided by Firebase,

A mobile and web application development platform that is now part of Google's

Cloud offerings. Firebase Realtime Database is designed for real-time data

Synchronization and is commonly used in applications where you need to store,

Retrieve, and synchronize data across various clients and platforms.



Conclusion:

The provided Python program is designed for a Traffic Management system that utilizes various Sensors to monitor Traffic in the road and communicate this information to a Firebase Real-time Database.