

Name: E.Santhosh  
reg: 950321104043

# **IOT\_Phase\_4**

## **Overview**

In the fourth phase of our project, we continued to develop our Traffic Management System, focusing on enhancing vehicle detection and congestion analysis. We made significant progress by implementing Ultrasonic sensors, Arduino boards, and simulated traffic lights to improve our system's functionality and efficiency.

## **Vehicle Detection**

Our Traffic Management System employs Ultrasonic sensors to detect the presence of vehicles on the road. The principle behind this detection method is straightforward: when a vehicle passes by, the distance reading from the sensor decreases. In contrast, when no vehicles are nearby, the distance reading remains high (at its maximum). This way, we determine the degree of congestion.

## **Pivots**

We decided to pivot and utilize the Wokwi platform due to our familiarity with it. Wokwi simplified the development process for us and made it more accessible. This transition allowed us to streamline the integration of our components and data flow.

In a move towards simplicity and cost-effectiveness, we shifted from our initial idea of using Raspberry Pi to utilizing Arduino boards. This change allowed us to maintain a cost-effective approach without compromising functionality. Additionally, we introduced LED simulations of traffic lights, which enhanced our system's capabilities. We felt that it will be efficient to control traffic lights near our board instead of connecting to cloud and again controlling the lights just a few distance away from our arduino board.

## **Congestion Density**

While our system counts the number of vehicles, it's important to note that the readings don't provide a precise count of nearby vehicles. Instead, they offer an abstract score of congestion density, which serves as an indicator of traffic conditions in the area.

## **Three Key Components**

Our project comprises three main components:

### **1. IoT Component**

This component consists of Arduino boards, Ultrasonic sensors, and simulated traffic lights. It's responsible for vehicle detection and counting.

### **2. Cloud Integration**

We integrated Wokwi as our cloud platform to send and receive data. The cloud component receives data on congestion levels, processes it, and stores the information. This data is then visualized, allowing us to gain insights into congestion patterns.

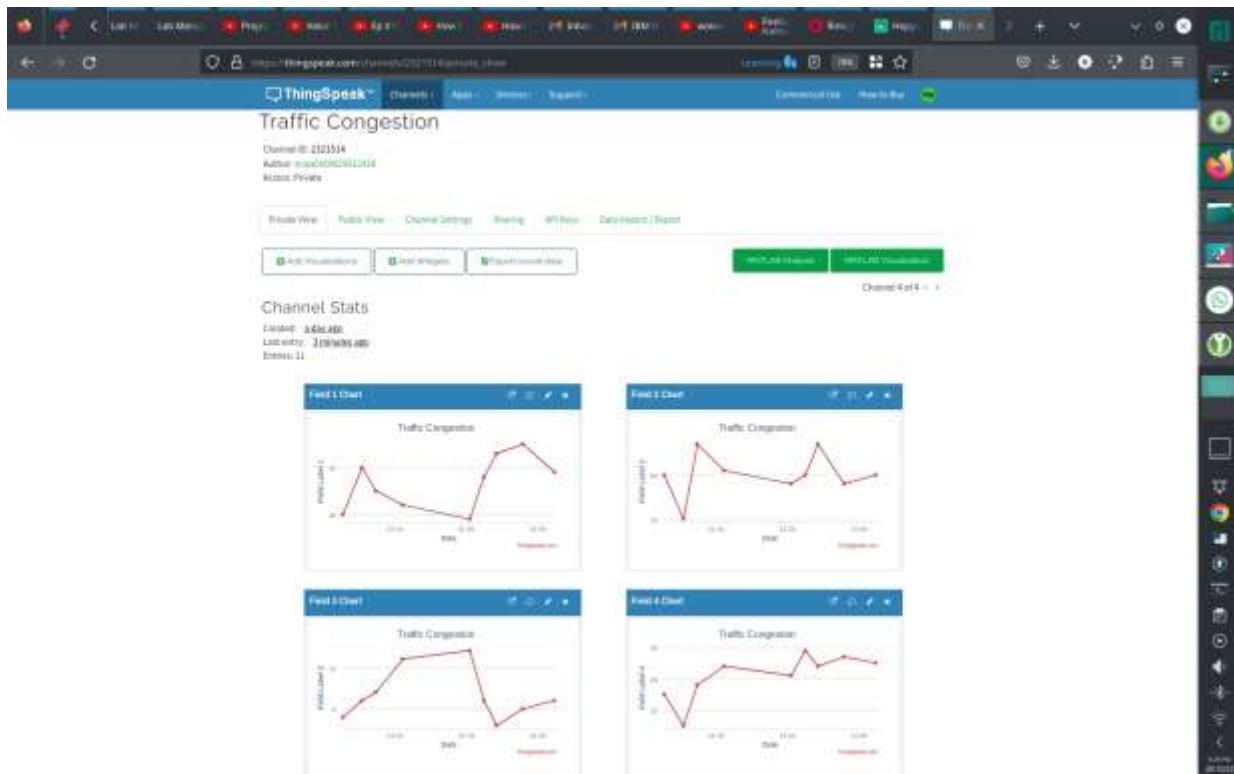
### **3. Client-Facing Application**

A Dart-based client application has been developed to provide end-users with access to congestion information. This application retrieves real-time congestion data from Wokwi based on the latest updates from our IoT component. Users can make informed decisions about their routes and travel plans based on this up-to-date congestion data.

## **Data Flow**

The IoT component detects and records vehicle numbers, which are then transmitted to the Wokwi platform. The cloud component processes this data and provides a clear visualization of congestion patterns. Simultaneously, the Dart application periodically queries Wokwi for the latest congestion data, enabling users to make well-informed decisions based on real-time information.

In this phase, we made substantial progress in enhancing our Traffic Management System by improving vehicle detection, transitioning to a more accessible platform, and ensuring the efficient analysis of congestion data. The incorporation of traffic light simulations and the client-facing application also contribute to the system's overall effectiveness.



The above Picture depicts vehicle congestion readings from wokwi platform.

```

1 #include <TimerOne.h>
2 int signal1[] = {23, 25, 27};
3 int signal2[] = {46, 48, 50};
4 int signal3[] = {13, 12, 11};
5 int signal4[] = {10, 9, 8};
6 int redDelay = 5000;
7 int yellowDelay = 2000;
8 volatile int triggerpin1 = 31;
9 volatile int echopin1 = 29;
10 volatile int triggerpin2 = 44;
11 volatile int echopin2 = 42;
12 volatile int triggerpin3 = 7;
13 volatile int echopin3 = 6;
14 volatile int triggerpin4 = 5;
15 volatile int echopin4 = 4;
16 volatile long time; // Variable for storing the time
17 volatile int S1, S2, S3, S4; // variables for storing the dist
18 int t = 5; // distance under which it will look for vehicles.
19 void setup() {
20   Serial.begin(115200);
21   Timer1.initialize(1000000); //Begin using the timer. This function must
22   Timer1.attachInterrupt(sofrinterr); //Has a function each time the time
23   // Declaring LED pins as output
24   for(int i=0; i<4; i++){
25     pinMode(signal1[i], OUTPUT);
26     pinMode(signal2[i], OUTPUT);
27     pinMode(signal3[i], OUTPUT);
28     pinMode(signal4[i], OUTPUT);
29   }
30   // Declaring ultrasonic sensor pins as output
31   pinMode(triggerpin1, OUTPUT);
32   pinMode(echopin1, INPUT);
33   pinMode(triggerpin2, OUTPUT);
34   pinMode(echopin2, INPUT);
35   pinMode(triggerpin3, OUTPUT);
36   pinMode(echopin3, INPUT);
37   pinMode(triggerpin4, OUTPUT);
38   pinMode(echopin4, INPUT);

```

The screenshot shows the Wokwi simulation environment. On the left, the Arduino IDE code editor displays the sketch code for the traffic congestion simulation. The code includes the TimerOne library and defines four ultrasonic sensors (trigger pins 31, 44, 7, 5 and echo pins 29, 42, 6, 4). The setup function initializes the serial port, timer, and pins. The main loop (not fully visible) would handle the sensor readings and data transmission. On the right, the simulation window shows a top-down view of the circuit board with four ultrasonic sensors connected to the Arduino Uno. Below the simulation, a terminal window displays the serial output, showing the sensor readings (S1, S2, S3, S4) as 2, 399, 399, 399.

This shows the wokwi simulation we created. It also sends data to thingspeak