IoT Based Traffic Management System

A Project report submitted in partial fulfilment of the requirements for the degree of B.E in Computer Science and Engineering

By

R.V.SRISAILANHARITHA(513221104316)

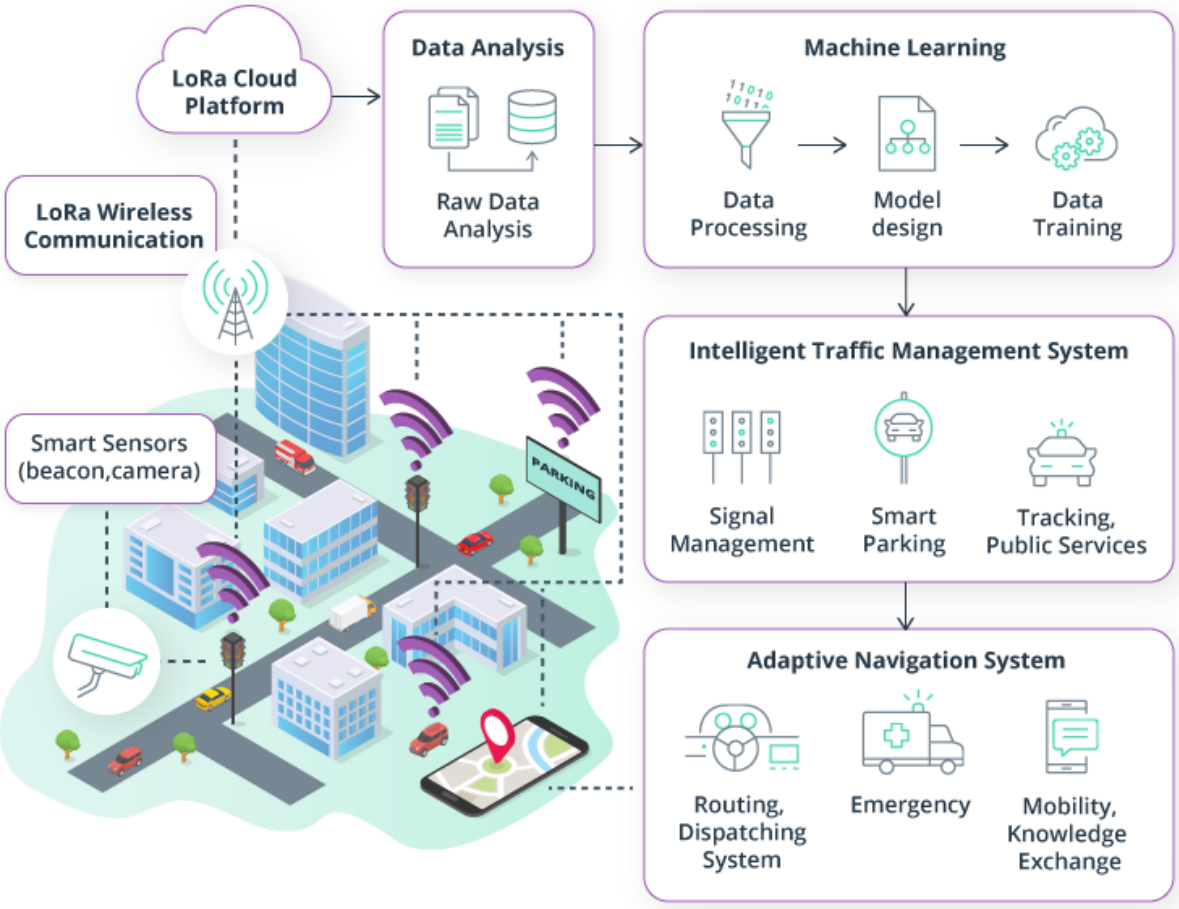
Under the supervision of

Professor & HOD

Department of Computer Science and Engieering

INTRODUCTION

With the increase in urbanization, many cities around the world are experiencing a very rapid growth in the number of vehicles which lead to serious traffic congestion problems. This places a greater demand on operating roadway systems with maximum efficiency.One major factor that affects the traffic flow is the management of the traffic at road intersections.



Hardware Components

1. **Microcontroller (Arduino Mega 2560):** The Arduino Mega 2560 is a microcontroller board based on the Atmega 2560. It has 54 digital input/output pins (of which 15 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The Mega 2560 board is compatible with most shields designed for the Uno and the former boards Duemilanove or Diecimila.

Figure 1: Arduino Mega 2560

2. **Microcontroller (Arduino Uno):** The Arduino UNO is an open-source microcontroller board based on the Microchip

ATmega328Pmicrocontroller and developed by Arduino.cc. The board is equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards (shields) and other circuits. The board has 14 Digital pins, 6 Analog pins, and programmable with the Arduino IDE (Integrated Development Environment) via a type B USB cable.

Figure 2: Arduino Uno

3. **LEDs:** LEDs are used for the purpose of signaling according to the traffic condition.

Figure 3: LED for Traffic Lights.

4. **IR Sensor:** IR Sensor is used to count the vehicles on the road.

Figure 4: IR Sensors.

5. **Jumper Wires:** It is used to connect the components to each other.

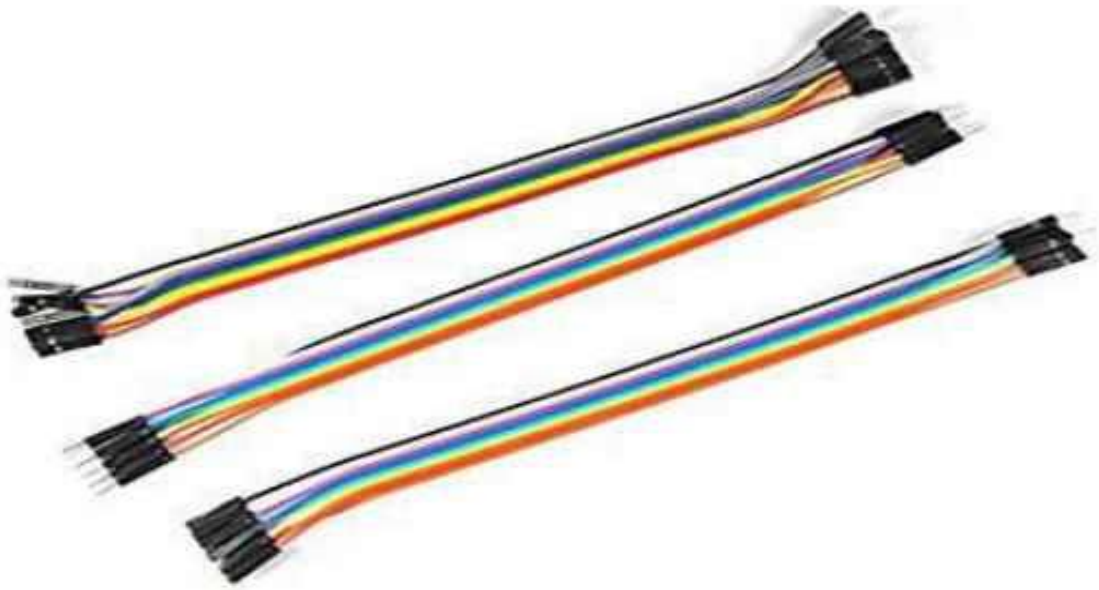


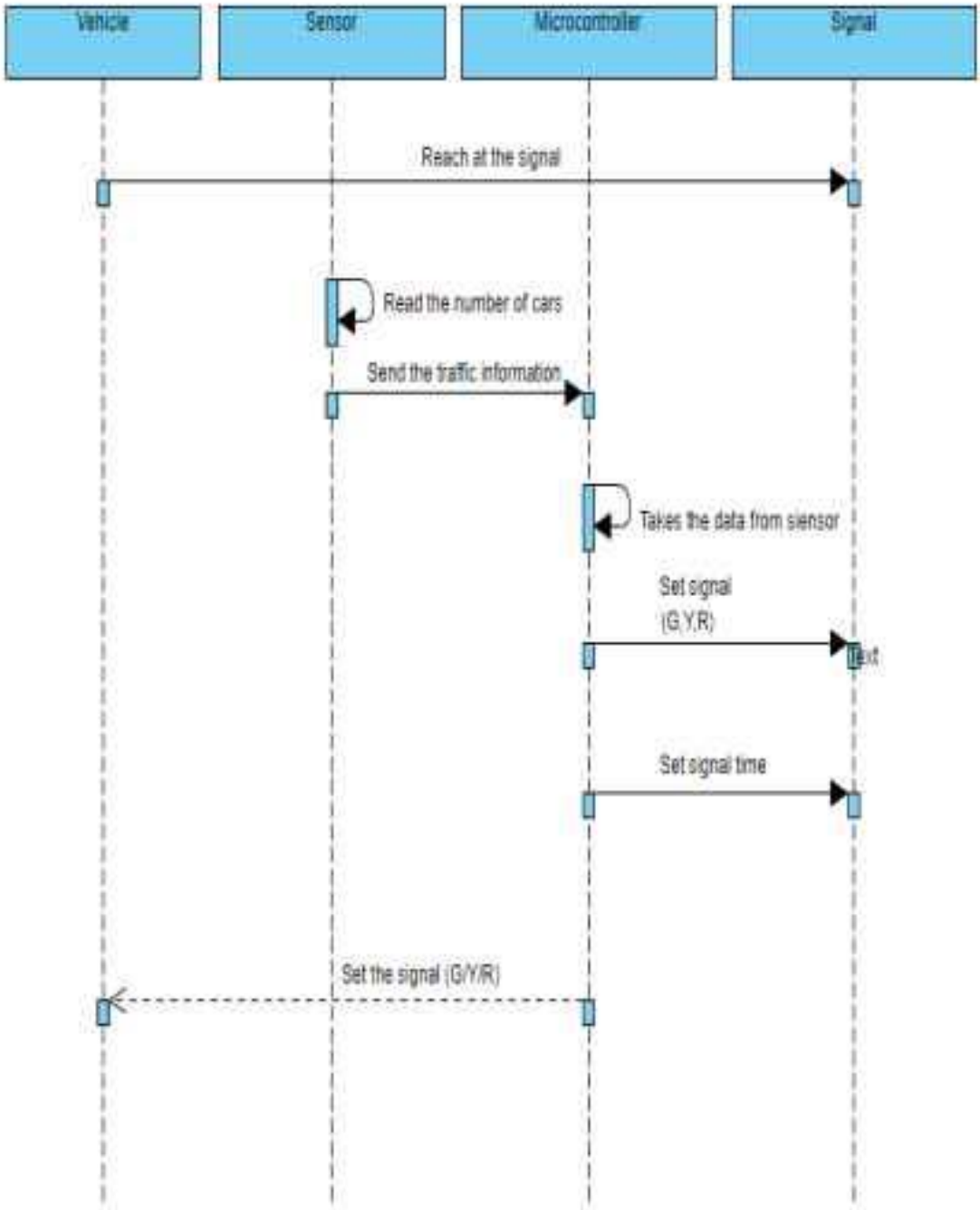
Figure 5: Jumper Wires.

Software Requirement

1. Arduino IDE: The Arduino integrated development environment (IDE) is a cross-platform application (for Windows, MacOS, Linux) that is written in the programming language Java. It is used to write and upload programs to Arduino board.The source code for the IDE is released under the GNU General Public License, version 2.

2. Proteus Design Suite: The Proteus Design Suite is a proprietary software tool suite used primarily for electronic design automation. The software is used mainly by electronic design engineers and technicians to create schematics and electronic prints for manufacturing printed circuit boards.

Sequence Diagram



Algorithms

Vehicle Counter Algorithm

Assuming the objects detected by the IR Sensors to be vehicles,
int counter = 0;
int hitObject = false;
int val ;

Step 1: Read value from sensor (val). Sensor gives output 0 if car is detected and 1 if no car is detected.

Step 2: If val == 0 hitObject = false then increment the counter and set hitObject = true.

else if val == 1 hitObject = true
then set hitObject = false.

Step 3: Go to step 1

Traffic Control Algorithm

No. of sensors = 8 and are denoted by S1, S2, S3, S4, S5, S6, S7, S8
No. of cars in Lane 1 (N1) = S1 – S2
No. of cars in Lane 2 (N2) = S3 – S4
No. of cars in Lane 3 (N3) = S5 – S6
No. of cars in Lane 4 (N4) = S7 – S8
Li = (L1, L2, L3, L4), Ni = (N1, N2, N3, N4), Ti = (T1, T2, T3, T4)

Step 1: Start

Step 2: Sensors will read the no. of vehicles on each lane (i.e. L1, L2, L3, L4)

Step 3: if (Vehicle Count < Threshold) Then status = Normal traffic. Turn on the green signal for all the lanes one after another in a sequential manner (L1-L2-L3-L4). When signal is green for one lane,the others will remain red.

Step 4: else status = congestion.

Step 5: COMPARE (N1, N2 , N3, N4), Select the highest of the four (say Ni),turn on green signal for that lane (say Li) for time (Ti). When time Ti ends, turn on the red signal.

Step 6: COMPARE (N2, N3, N4), Select the highest of the three (say Ni), turn on green signal for that lane (say Li) for time (Ti). When time Ti ends, turn on the red signal.

Step 7: COMPARE (N3, N4), Select the highest of the two (say Ni), turn on green signal for that lane (say Li) for time (Ti). When time Ti ends, turn on the red signal.

Step 8: The last remaining lane automatically gets selected and it is given the green signal for time Ti.

Step 9: Jump to Step 3.

Displaying real-time traffic information involves multiple components, including HTML, CSS, and JavaScript for the front-end, as well as a back-end system.

Front-End (HTML, CSS, JavaScript):

1. **Design the User Interface (UI):** Create a user-friendly and intuitive design for your traffic information platform. Sketch out wireframes and design mockups before you start coding.

2. HTML Structure: Write the HTML structure for your platform. This should include the layout of your webpage, placeholders for real-time traffic data, and interactive elements.

Html

```
<!DOCTYPE html>
<html>
<head>
  <title>Real-time Traffic Information</title>
  <link rel="stylesheet" type="text/css" href="styles.css">
</head>
<body>
  <header>
    <h1>Real-time Traffic Information</h1>
  </header>
  <main>
    <section id="map"></section>
```

```
        <section id="traffic-info">
            <div id="traffic-list"></div>
        </section>
    </main>
    <script src="script.js"></script>
</body>
</html>
```

CSS Styling: Use CSS to style your platform, making it visually appealing and responsive. You may want to use a CSS framework like Bootstrap to expedite this process.

Css

```
body {
    font-family: Arial, sans-serif;
}
header {
    background-color: #333;
    color: white;
    padding: 10px;
    text-align: center;
}
main {
    display: flex;
}
{
    flex: 1;
  }
```

```
{
  flex: 1;
  padding: 20px;
}
```

JavaScript for Real-Time Data: Use JavaScript to fetch and display real-time traffic information. You can use APIs like Google Maps API or a service like Mapbox to display the map.

Javascript

```
// script.js
window.addEventListener('DOMContentLoaded', () => {
  // Code to fetch and display real-time traffic data goes here
});
```

Back-End (Server):

- **Data Source:** Identify a reliable source for real-time traffic data. This could be a traffic data provider or a publicly available API like the one provided by your local transportation department.
- **Server:** Set up a server to fetch and store real-time traffic data. You can use a back-end technology like Node.js, Python (with Flask or Django), or any other technology of your choice.
- **API:** Create an API on your server to expose the real-time traffic data to the front-end. This API should be able to provide data such as traffic congestion, road closures, and estimated travel times.
- **Caching:** To reduce the load on the server and improve response times, you can implement caching mechanisms to store and serve frequently requested traffic data.

- **Authentication:** Depending on your data source, you may need to implement authentication to access real-time traffic data securely.