IDENTIFICATION OF VEHICLES USING UNIQUE ID AND TRAFFIC AUTOMATION

A Project Report Submitted in
Partial Fulfillment of the
Requirements for the award of
the Degree

Of

Bachelor of Engineering
in **Electronics and Communication Engineering**
Of
DR. A.P.J ABDUL KALAM UNIVERSITY, INDORE (M.P)

By

Dilruba Aleema (011ECE19GT008)

ELECTRONICS & COMMUNICATION ENGINEERING
COLLEGE OF ENGINEERING, DR APJ ABDUL KALAM UNIVERSITY, INDORE (M.P)

DECLARATION

I declare that the work presented in this project titled "***Identification of vehicles using unique id and traffic automation***"**,** submitted to the Electronics and Communication Engineering Department, College of Engineering (COE) for the award of the ***Bachelor of Engineering*** degree in ***Electronics and Communication Engineering***, is my original work. I have not plagiarized or submitted the same work for the award of any other degree. In case this declaration is found incorrect, I accept that my degree may be unconditionally withdrawn

.

Date: Dilruba Aleema (011ECE19GT008)

Place: Indore

CERTIFICATE

This is to certify that **Dilruba Aleema** has undergone project entitled **“Identification of vehicles using unique id and traffic automation”** towards the partial fulfillment of her four years bachelor’s degree in electronics and communication engineering successfully. She has carried out this project with full sincerity and dedication and the work is original and genuine.

PROJECT GUIDE

Mrs. Bindewasini Bageshwari
Assistant Professor

……………………………… ……………………………..

INTERNAL SIGNATURE EXTERNAL SIGNATURE

PRINCIPAL
College of Engineering, Indore

ACKNOWLEDGEMENT

I express my deep sense of gratitude to the almighty God for his blessings. The satisfaction and euphoria that accompany the successful completion of a task would be incomplete without the mention of the people who made it possible and whose constant guidance crowns all the efforts with success.

I also owe my sincere thanks to Mr. Amol Khumbare, Head of the Electronics Department, for the help rendered in carrying out this project work.

My special thanks to my guide, Mrs. Bindewasini Bageshwari, Assistant Professor, Dept. of Electronics and Communication Engineering, for the help and guidance given throughout the preparation and conduction of this project.

I would like to thank the teachers in the Dept. of Electronics and Communication Engineering, COE for their support and guidance given for the completion of the project, and to my parents for their continuous trust and support for the completion of my endeavor.

Date: Dilruba Aleema (011ECE19GT008)

Place: Indore

ABSTRACT

Traffic control is found to be a major challenging task in urban areas. In the existing traffic control methods green signal is given for a given period even for an empty road. To overcome this, we propose an idea of vehicles with a unique id, for tracking and recording data of vehicles passing through a junction with GPS receiver. It receives signals from GPS satellites and determines locations, with these collected data traffic signals are controlled using C++ to develop custom traffic signal, traffic signal algorithms. It can be used to analyze traffic flow and develop optimized signal timing based on different traffic scenarios.

The collected location data and information are transmitted to central server or monitoring stations. The data is uploaded to the server and stored in SQL database. The current method of vehicle identification in a hit and run solely depends on the number plates of vehicles. Since these number plates can be easily replaced there are high possibilities for crime. The idea of vehicles with unique id can also be used to reduce crime by collecting data and transmitting it to central server.

CONTENTS

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1 Overview

The project focuses on the identification of vehicles using a unique ID and its integration into a traffic automation system. In modern urban environments, traffic congestion has become a significant challenge, leading to increased travel times, fuel consumption, and environmental pollution. Therefore, there is a growing need for efficient traffic management systems that can optimize traffic flow and reduce congestion.
The core concept of this project revolves around assigning a unique identification (ID) to each vehicle using a combination of technologies.

The identification of vehicles using a unique ID plays a crucial role in developing an intelligent traffic automation system. By assigning a unique identifier to each vehicle, it becomes possible to track and monitor its movement throughout the road network. This identification can be achieved through various means, such as unique token on gps data, unique id of a hardware inside the vehicle. Once the vehicles are identified, the collected data can be processed and integrated into a traffic automation system. The system can leverage advanced technologies like machine learning, data analytics, and real-time processing to analyze the traffic data and make informed decisions. To ensure accurate and reliable identification, these features will be associated with a unique ID assigned to each vehicle. This ID can be stored in a centralized database and linked to relevant information such as vehicle ownership, registration details, insurance records, and any outstanding violations. Such a system will enable authorities to quickly identify vehicles, track their movements, and enforce traffic regulations more effectively.

The primary objective of this project is to develop a comprehensive solution that combines vehicle identification with traffic automation. By integrating the unique ID information of vehicles into the traffic automation system, it becomes possible to optimize traffic signal timings, dynamically adjust lane priorities, and coordinate the flow of vehicles more efficiently. Moreover, this project also aims to incorporate traffic automation to optimize the flow of vehicles and reduce congestion. By integrating the identified vehicles with smart traffic control systems, real-time data analysis, and predictive modeling, the system can dynamically adjust traffic signals, manage lane allocations, and provide alternate routes based on traffic conditions. This automation will help alleviate bottlenecks, improve traffic flow, and enhance overall safety on the roads.

The project will involve the design and implementation of algorithms and models to process the collected vehicle identification data. These algorithms will enable the system to recognize traffic patterns, predict congestion, and adapt traffic control strategies in real-time. The automation system will leverage the collected data to optimize traffic signal timings, prioritize lanes based on traffic volume, and provide optimal routes to drivers. The benefits of implementing this project are manifold. Authorities can effectively enforce traffic regulations, monitor vehicle movements, and detect any suspicious activities or stolen vehicles promptly. Commuters will experience reduced travel times, smoother traffic flow, and improved safety. Additionally, the availability of accurate and comprehensive data will enable policymakers to make informed decisions and plan infrastructure upgrades to meet the growing demands of urban mobility.

The successful implementation of this project will have several benefits. Firstly, it will help in reducing traffic congestion by optimizing traffic flow and minimizing delays. Secondly, it will enhance road safety by intelligently coordinating traffic movements and reducing the likelihood of accidents. Additionally, it will contribute to a more sustainable and eco-friendly transportation system by reducing fuel consumption and emissions. The project "Identification of Vehicles Using Unique ID and Traffic Automation" seeks to revolutionize traffic management by leveraging advanced technologies. By employing unique ID assignment, and traffic automation, this system will enhance the efficiency, safety, and overall effectiveness of traffic management, ultimately leading to a more streamlined and sustainable transportation ecosystem.

CHAPTER 2
LITERATURE SURVEY

The proposed system named Dynamic Traffic Signal Processing incorporates dynamic traffic prediction and timing scheme using image processing techniques that has self-learning ability. It effectively manages traffic flow even for a huge number of vehicles in the junction. It also develops a system that does not require qualified mechanical personnel to enhance and preserve the system. [1]

The project is aimed at designing a density based dynamic traffic signal system where the timing of signal will change automatically on sensing the traffic density at any junction. Traffic congestion is a severe problem in most cities across the world and therefore it is time to shift from more manual mode or fixed timer mode to an automated system with decision making capabilities. Present day traffic signaling system is fixed time based which may render inefficient if one lane is operational than the others. To optimize this problem, we have made a framework for an intelligent traffic control system. Sometimes higher traffic density at one side of the junction demands longer green time as compared to standard allotted time We, therefore propose here a mechanism in which the period of green light and red light is assigned on the basis ofthe density of the traffic present at that time. This is achieved by using PIR (proximity Infrared sensors). Once the density is calculated, the glowing time of green light is assigned by the help of the microcontroller (Arduino). The sensors which are present on the sides of the road will detect the presence of the vehicles and sends the information to the microcontroller where it will decide how long a flank will be open or when to change over the signal lights. In subsequent sections, we have elaborated the procedure of this framework. [2]

In order to overcome the vital issues of vehicle congestion and time consumption in traffic signals, a new method using IR Sensor technology is proposed. It can serve as the best solution to the manually operated traffic signal. The object detection sensor deployed at the traffic signal path measures the density of the vehicle in the road. Depending on the density it provides automatic signal timing accordingly. This makes the existing system still easier. Due to long queues in traffic signal, it is not comfortable to pass an ambulance immediately. We haveproposed a system which removes this problem using RF Transmitter and Receiver. The path signal will change to blue along with green signal, indicating the ambulance arrival. Thus, it reduces time & saves the lives of many people due to existing traffic system limitations. Also stolen vehicle can be easily detected with the help of RFID [3]

In the widely used traditional control system, traffic coordination and control between regions are neglected. Regional boundary Silas traffic control effect has direct influence on the input traffic flow control and output traffic flow control of the regions. In this paper, a traffic coordination and control model of regional boundary based on fuzzy control is proposed. Two fuzzy controllers are designed to coordinate and control the regional boundary when traffic congestion happens. After that, a simulation network with 20 intersections and 45 links were put into VISSIM 4.2 to test the method proposed. Finally, after comparison, it concludes that the proposed method could improve traffic condition of each region to some extent and alleviates traffic congestion [4]

Now a day's traffic problems are increasing day by day and transport traffic is becoming a serious problem in the world. Various traffic monitoring systems have been developed. This paper presents research on controlling real-time traffic using various image processing techniques in which the pictures are captured using the webcam of various lanes of roads where traffic takes place. The counting of transport vehicles in each picture is computed using image processing techniques in the MATLAB tool and the timer is allocated to lane based on several vehicles counted in the specific picture of the lane for showing the green signal to pass the vehicle. In this model LEDs are used to show the green and red signals and seven segments are used to show the decrementing timer of signal green [5]

Detection and identification of vehicles in traffic surveillance videos is very important to automate the surveillance system and to build an intelligent transportation system. In this paper a robust method to detect and identify vehicles is proposed which deals with problems like changes in illumination. Background subtraction is done using both Gaussian Mixture Model and Visual Background Extractor and supports dynamic changes in background. Vehicles are detected by finding contours in the image frame. Vehicles are tracked by assigning unique ID for each vehicle. Distance between the centroid of detected vehicle and existing vehicles is calculated. If the distance is greater than threshold value, then vehicle is considered to be arrived newly and a unique ID is assigned for further tracking. Otherwise, it is the vehicle will get the same ID as in previous frame. Detected vehicle is classified using Support Vector Machine in each frame and final decision is taken when the vehicle is about to exit from the scene [6]

In this paper, a real-time radio frequency identification (RFID)-GSM-based vehicle identification system compatible with IEEE802.15.4 is designed and implemented at 2.4 GHz, which provides a full automation of highway scanning far away from the monitoring station. Once the RFID reader broadcasts “auto-highway-scanning” RF signal, each tagged vehicle within the RF field of 80 m from the reader runs a collision-avoidance scheme involving two strategies. First, the unique tag's CC2530F256 SoC MAC address is used to generate a fixed waiting time. The second strategy utilizes tag's CC2530F256 pseudorandom number generator to add a random value to the first strategy output. This strategy makes the proposed scheme dynamic and secure, because it prevents the attackers from accessing the tag ID by estimating fixed waiting time of the first strategy. Such a collision-avoidance scheme overcomes the conventional methods constraints, such as tag population estimation latency in Aloha-based methods and time-consuming lengthy queries in Tree-based protocols. Simulation results show that the collision is avoided by using the carrier sensing capability and the identification efficiency of 63% is achieved by the proposed scheme, which is more efficient than conventional tag anti-collision schemes, such as ISO/IEC 18000-7, CSMA non persistent, CSMA p-persistent, QT, CT, and QWT. Besides, experimental results prove that the above-mentioned scheme works properly.[7]

CHAPTER 3
COMPONENTS DESCRIPTION

3.1 Hardware Components

3.1.1 UBLOX NEO 6M GPS MODULE

The NEO-6M GPS module is a robust GPS receiver featuring a built-in ceramic antenna measuring 25 x 25 x 4mm, enhancing its satellite search capabilities. The power and signal indicators provide real-time module status monitoring.

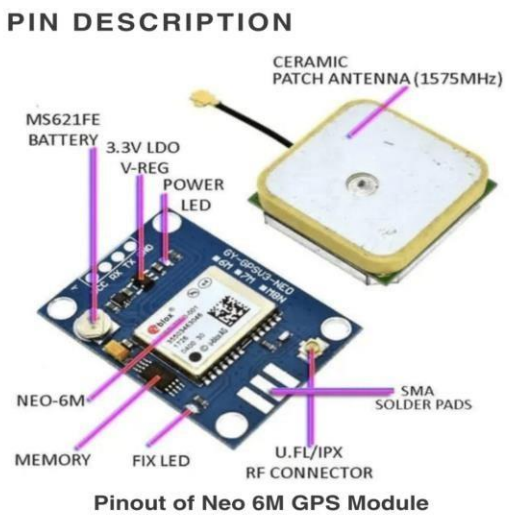


Fig 3.1 Ublox Neo 6M

- This is Neo 6M GPS Module
- This Neo 6M GPS Module with 56 channels for precise position updates at 10Hz.
- Protective molded plastic case suitable for aircraft and quadcopter applications.
- Seamless integration with Arduino for accurate GPS data.
- External GPS antenna and UART TTL connections for reliable connectivity
- Rechargeable li-ion battery enables quick hot starts and faster GPS lock acquisition.
- Excellent sensitivity, data backup battery, and compatibility with Ardupilot Mega v2 flight controller.
- VCC-Supply Voltage
- GND-Ground pin
- TX and RX-These 2 pins act as a UART interface for communication

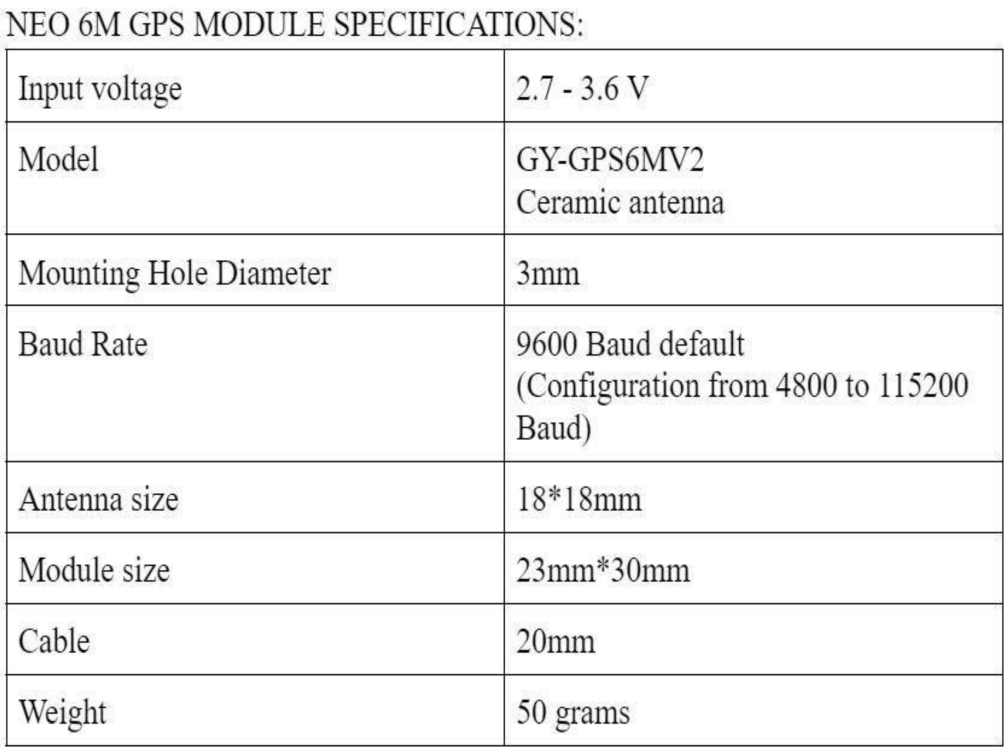
FEATURES:

- 5Hz position update rate
- Operating temperature range: -40 TO 85°CUART TTL socket
- EEPROM to save configuration settings Rechargeable battery for Backup
- The cold start time of 38 s and Hot start time of 1 s
- Supply voltage: 3.3 V Configurable from 4800 Baud to 115200
- Baud rates. (default 9600) SuperSense® Indoor GPS: -162 dBm
- tracking sensitivity Support SBAS (WAAS, EGNOS, MSAS, GAGAN)
- Separated 18X18mm GPS antenna

APPLICATIONS:

Connectivity between

- Machines
- Industrial tanks
- Agricultural irrigation systems
- Manufacturing robotics



NEO 6M GPS MODULE SPECIFICATIONS:

Input voltage	2.7 - 3.6 V
Model	GY-GPS6MV2 Ceramic antenna
Mounting Hole Diameter	3mm
Baud Rate	9600 Baud default (Configuration from 4800 to 115200 Baud)
Antenna size	18*18mm
Module size	23mm*30mm
Cable	20mm
Weight	50 grams

Fig 3.2 Neo 6M specification

3.1.2 NODE MCU

The NodeMCU (Node Microcontroller Unit) is an open-source software and hardware development environment built around an inexpensive System-on-a-Chip (SoC) called the ESP8266. The ESP8266, designed and manufactured by Espressif Systems, contains the crucial elements of a computer: CPU, RAM, networking (Wi-Fi), and even a modern operating system and SDK. That makes it an excellent choice for Internet of Things (IoT) projects of all kinds.

NodeMCU Specifications

The NodeMCU is available in various package styles. Common to all the designs is the base ESP8266 core. Designs based on the architecture have maintained the standard 30-pin layout. Some designs use the more common narrow (0.9″) footprint, while others use a wide (1.1″) footprint – an important consideration to be aware of.

The most common models of the NodeMCU are the Amica (based on the standard narrow pin-spacing) and the LoLin which has wider pin spacing and larger board. The open-source design of the base ESP8266 enables the market to design new variants of the NodeMCU continually.

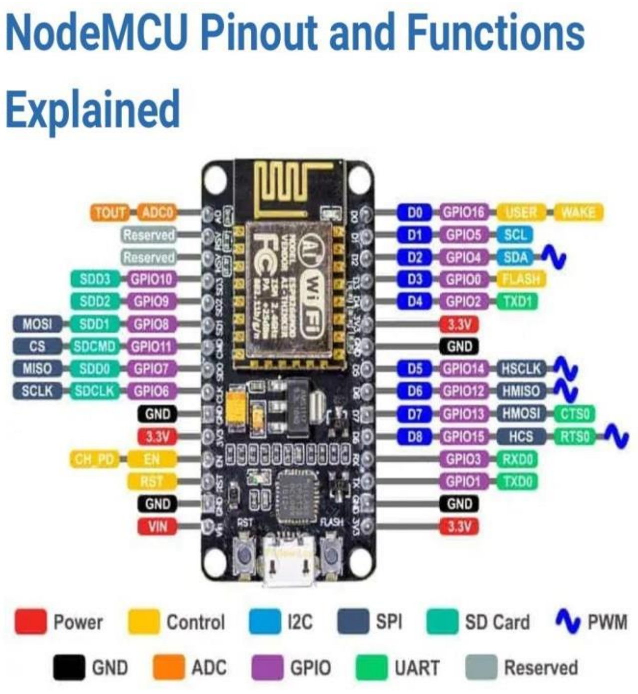


Fig 3.3 NodeMCU Pinout

Power Pins

There are four power pins. VIN pin and three 3.3V pins.

- VIN can be used to directly supply the NodeMCU/ESP8266 and its peripherals. Power delivered on VIN is regulated through the onboard regulator on the NodeMCU module – you can also supply 5V regulated to the VIN pin 3.3V pins are the output of the onboard voltage regulator and can be used to supply power to external components.
- GND are the ground pins of NodeMCU/ESP8266
- I2C Pins are used to connect I2C sensors and peripherals. Both I2C Master and I2C Slave are supported. I2C interface functionality can be realized programmatically, and the clock frequency is 100 kHz at a maximum. It should be noted that I2C clock frequency should be higher than the slowest clock frequency of the slave device.
- GPIO Pins NodeMCU/ESP8266 has 17 GPIO pins which can be assigned to functions such as I2C, I2S, UART, PWM, IR Remote Control, LED Light and Button programmatically. Each digital enabled GPIO can be configured to internal pull-up or pull-down or set to high impedance. When configured as an input, it can also be set to edge-trigger or level-trigger to generate CPU interruptions.
- ADC Channel the NodeMCU is embedded with a 10-bit precision SAR ADC. The two functions can be implemented using ADC. Testing power supply voltage of VDD3P3 pin and testing input voltage of TOUT pin. However, they cannot be implemented at the same time.
- UART Pins NodeMCU/ESP8266 has 2 UART interfaces (UART0 and UART1) which provide asynchronous communication (RS232 and RS485) and can communicate at up to 4.5 Mbps. UART0 (TXD0, RXD0, RST0 & CTS0 pins) can be used for communication. However, UART1 (TXD1 pin) features only data transmit signal so, it is usually used for printing logs.
- SPI Pins NodeMCU/ESP8266 features two SPIs (SPI and HSPI) in slave and master modes. These SPIs also support the following general-purpose SPI features:

4 timing modes of the SPI format transfer

- Up to 80 MHz and the divided clocks of 80 MHz
- Up to 64-Byte FIFO
- SDIO Pins NodeMCU/ESP8266 features Secure Digital Input/Output Interface (SDIO) which is used to directly interface SD cards. 4-bit 25 MHz SDIO v1.1 and 4-bit 50 MHz SDIO v2.0 are supported.

- PWM Pins The board has 4 channels of Pulse Width Modulation (PWM). The PWM output can be implemented programmatically and used for driving digital motors and LEDs. PWM frequency range is adjustable from 1000 μs to 10000 μs (100 Hz and 1 kHz).
- Control Pins are used to control the NodeMCU/ESP8266. These pins include Chip Enable pin (EN), Reset pin (RST) and WAKE pin.
- EN: The ESP8266 chip is enabled when EN pin is pulled HIGH. When pulled LOW the chip works at minimum power.
- RST: RST pin is used to reset the ESP8266 chip.
- WAKE: Wake pin is used to wake the chip from deep sleep.
- Tiny Sine WaveControl Pins are used to control the NodeMCU/ESP8266. These pins include Chip Enable pin (EN), Reset pin (RST) and WAKE pin.
- EN: The ESP8266 chip is enabled when EN pin is pulled HIGH. When pulled LOW the chip works at minimum power.
- RST: RST pin is used to reset the ESP8266 chip.
- WAKE: Wake pin is used to wake the chip from deep sleep.

NodeMCU Compatibility with Arduino IDE

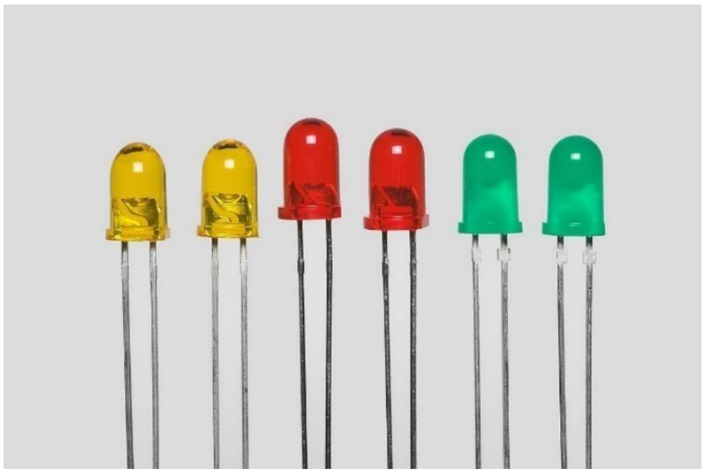
NodeMCU Compatibility with NodeMCU

NodeMCU - Arduino Open-Source Community

The NodeMCU offers a variety of development environments, including compatibility with the Arduino IDE (Integrated Development Environment). The NodeMCU/ESP8266 community took the IDE selection a step further by creating an Arduino add-on. If you’re just getting started programming the ESP8266 or even an established developer, this is the highly recommended environment. Visit our dedicated page on setting up and configuring the Arduino IDE for a NodeMCU ESP8266.

3.1.3 LED

A light-emitting diode (LED) is a semiconductor device that emits light when current flows through it. Electrons in the semiconductor recombine with electron holes, releasing energy in the form of photons. The color of the light (corresponding to the energy of the photons) is determined by the energy required for electrons to cross the band gap of the semiconductor. White light is obtained by using multiple semiconductors or a layer of light-emitting phosphor on the semiconductor device



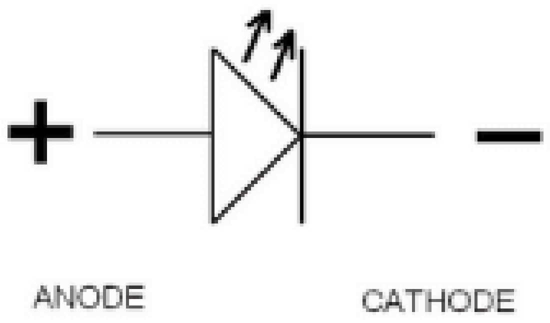


Fig 3.4 Led

3.2 Software Components

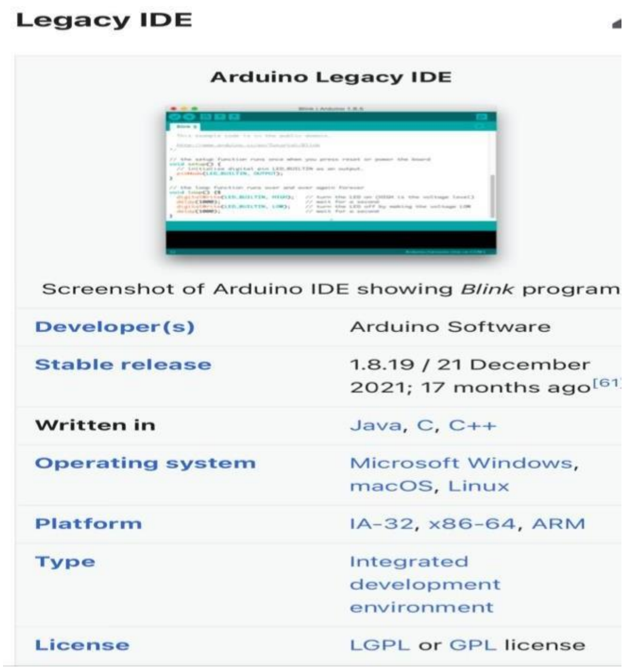
3.2.1 ARDUINO IDE

Arduino is an open-source hardware and software company, project, and user community that designs and manufactures single-board microcontrollers and microcontroller kits for building digital devices. Its hardware products are licensed under a CC BY-SA license, while the software is licensed under the GNU Lesser General Public License (LGPL) or the GNU General Public License (GPL), permitting the manufacture of Arduino boards and software distribution by anyone. Arduino boards are available commercially from the official website or through authorized distributors.

A program for Arduino hardware may be written in any programming language with compilers that produce binary machine code for the target processor. Atmel provides a development environment for their 8-bit AVR and 32-bit ARM Cortex-M based microcontrollers: AVR Studio (older) and Atmel Studio (newer).

The Arduino integrated development environment (IDE) is a cross-platform application (for Microsoft Windows, macOS, and Linux) that is written in the Java programming language. It originated from the IDE for the languages Processing and Wiring. It includes a code editor with features such as text cutting and pasting, searching and replacing text, automatic indenting, brace matching, and syntax highlighting, and provides simple one-click mechanisms to compile and upload programs to an Arduino board. It also contains a message area, a text console, a toolbar with buttons for common functions and a hierarchy of operation menus. The source code for the IDE is released under the GNU General Public License, version 2.

The Arduino IDE supports the languages C and C++ using special rules of code structuring. The Arduino IDE supplies a software library from the Wiring project, which provides many common input and output procedures. User-written code only requires two basic functions, for starting the sketch and the main program loop, that are compiled and linked with a program stub main() into an executable cyclic executive program with the GNU toolchain, also included with the IDE distribution. The Arduino IDE employs the program avrdude to convert the executable code into a text file in hexadecimal encoding that is loaded into the Arduino board by a loader program in the board's firmware.



Legacy IDE

Arduino Legacy IDE

Screenshot of Arduino IDE showing *Blink* program

Developer(s)	Arduino Software
Stable release	1.8.19 / 21 December 2021; 17 months ago[61]
Written in	Java, C, C++
Operating system	Microsoft Windows, macOS, Linux
Platform	IA-32, x86-64, ARM
Type	Integrated development environment
License	LGPL or GPL license

Fig 3.5 Arduino ide

3.2.2 XAMPP

The goal of XAMPP is to build an easy to install distribution for developers to get into the world of Apache. To make it convenient for developers, XAMPP is configured with all features turned on.

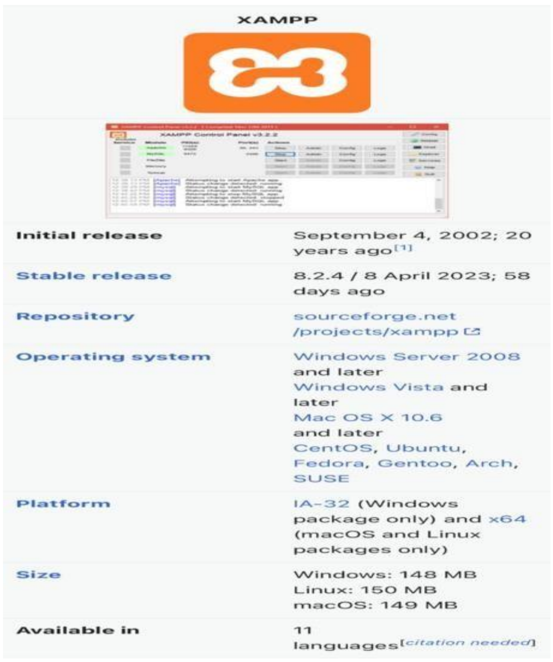
XAMPP is a free and open-source cross-platform web server solution stack package developed by Apache Friends, consisting mainly of the Apache HTTP Server, MariaDB database, and interpreters for scripts written in the PHP and Perl programming languages. Since most actual web server deployments use the same components as XAMPP, it makes transitioning from a local test server to a live server possible.

Features

XAMPP is regularly updated to the latest releases of Apache, MariaDB, PHP and Perl. It also comes with several other modules, including OpenSSL, phpMyAdmin, MediaWiki, Joomla, WordPress and more. Self-contained, multiple instances of XAMPP can exist on a single computer, and any given instance can be copied from one computer to another. XAMPP is offered in both a full and a standard version (Smaller version).

Prerequisites

XAMPP requires only one zip, tar, 7z, or exe file to be downloaded and run, and little or no configuration of the various components that make up the web server is required. The Windows version of XAMPP requires Microsoft Visual C++ 2017 Redistributable.



XAMPP

Initial release	September 4, 2002; 20 years ago[1]
Stable release	8.2.4 / 8 April 2023; 58 days ago
Repository	sourceforge.net /projects/xampp
Operating system	Windows Server 2008 and later Windows Vista and later Mac OS X 10.6 and later CentOS, Ubuntu, Fedora, Gentoo, Arch, SUSE
Platform	IA-32 (Windows package only) and x64 (macOS and Linux packages only)
Size	Windows: 148 MB Linux: 150 MB macOS: 149 MB
Available in	11 languages[citation needed]

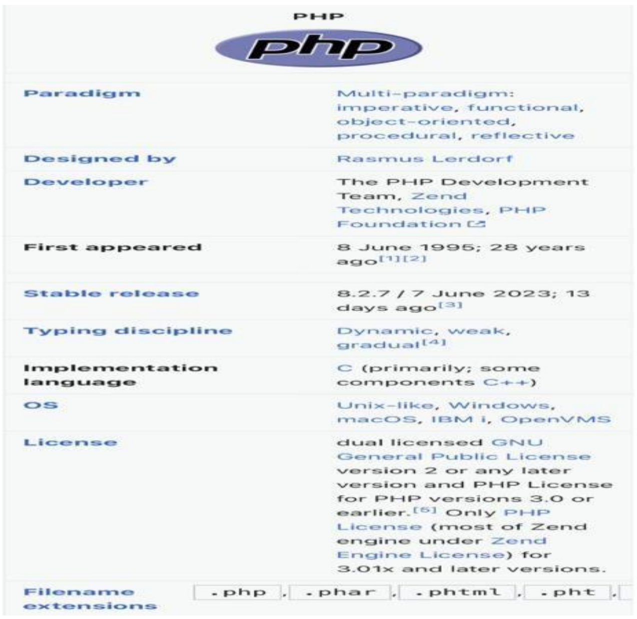
Fig 3.6 Xampp

3.3 Languages

3.3.1 PHP

PHP is a general-purpose scripting language geared towards web development. It was originally created by Danish-Canadian programmer Rasmus Lerdorf in 1993 and released in 1995.The PHP reference implementation is now produced by the PHP Group. PHP was originally an abbreviation of Personal Home Page, but it now stands for the recursive initialism PHP:Hypertext Preprocessor.

PHP code is usually processed on a web server by a PHP interpreter implemented as a module, a daemon or a Common Gateway Interface (CGI) executable. On a web server, the result of the interpreted and executed PHP code—which may be any type of data, such as generated HTML or binary image data—would form the whole or part of an HTTP response. Various web template systems, web content management systems, and web frameworks exist that can be employed to orchestrate or facilitate the generation of that response. Additionally, PHP can be used for many programming tasks outside the web context, such as standalone graphical applications and robotic drone control. PHP code can also be directly executed from the command line.



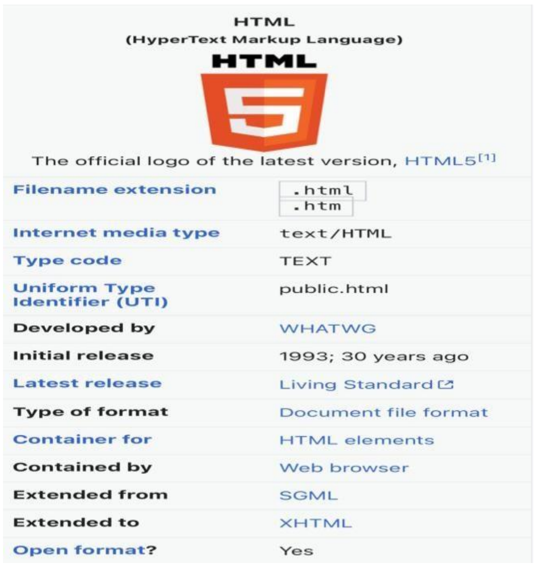
PHP

Paradigm	Multi-paradigm: imperative, functional, object-oriented, procedural, reflective
Designed by	Rasmus Lerdorf
Developer	The PHP Development Team, Zend Technologies, PHP Foundation
First appeared	8 June 1995; 28 years ago[1][2]
Stable release	8.2.7 / 7 June 2023; 13 days ago[3]
Typing discipline	Dynamic, weak, gradual[4]
Implementation language	C (primarily; some components C++)
OS	Unix-like, Windows, macOS, IBM i, OpenVMS
License	dual licensed GNU General Public License version 2 or any later version and PHP License for PHP versions 3.0 or earlier.[5] Only PHP License (most of Zend engine under Zend Engine License) for 3.01x and later versions.
Filename extensions	.php, .phar, .phtml, .pht,

Fig 3.7 Php

3.3.2 HTML

The HyperText Markup Language or HTML is the standard markup language for documents designed to be displayed in a web browser. It is often assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript.



HTML
(HyperText Markup Language)

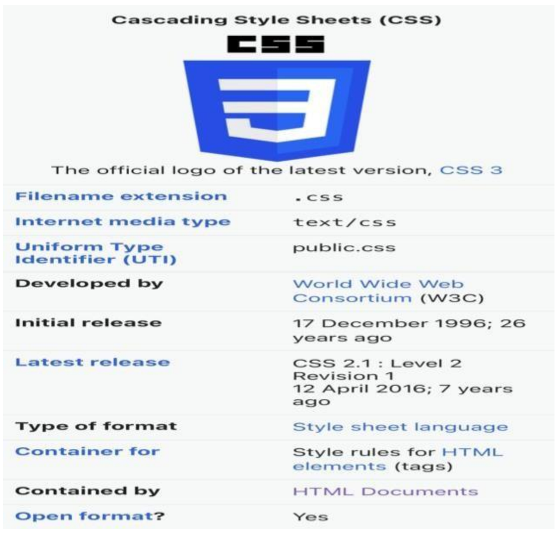
The official logo of the latest version, HTML5[1]

Filename extension	.html .htm
Internet media type	text/HTML
Type code	TEXT
Uniform Type Identifier (UTI)	public.html
Developed by	WHATWG
Initial release	1993; 30 years ago
Latest release	Living Standard
Type of format	Document file format
Container for	HTML elements
Contained by	Web browser
Extended from	SGML
Extended to	XHTML
Open format?	Yes

Fig 3.8 Html

3.3.3 CSS

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language such as HTML or XML (including XML dialects such as SVG, MathML or XHTML).CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.



Cascading Style Sheets (CSS)

CSS

The official logo of the latest version, CSS 3

Filename extension	.css
Internet media type	text/css
Uniform Type Identifier (UTI)	public.css
Developed by	World Wide Web Consortium (W3C)
Initial release	17 December 1996; 26 years ago
Latest release	CSS 2.1 : Level 2 Revision 1 12 April 2016; 7 years ago
Type of format	Style sheet language
Container for	Style rules for HTML elements (tags)
Contained by	HTML Documents
Open format?	Yes

Fig 3.9 Css

3.3.4 MYSQL

MySQL is an open-source relational database management system (RDBMS) that provides a powerful and scalable platform for storing, managing, and retrieving structured data. It is widely used in various applications, ranging from small personal projects to large-scale enterprise systems.

MySQL language and its key features:

Structured Query Language (SQL): MySQL uses the SQL language for interacting with the database. SQL provides a standardized way to perform operations such as creating, retrieving, updating, and deleting data from the database. MySQL supports a wide range of SQL statements and features, making it highly versatile for data manipulation.

Data Definition Language (DDL): MySQL allows you to define and manage the structure of your database using DDL statements. With DDL, you can create and modify database schemas, tables, indexes, constraints, and views. This allows for the organization and optimization of data storage and retrieval.

Data Manipulation Language (DML): DML statements in MySQL enable you to insert, update, delete, and retrieve data from tables. You can use SELECT statements to query the database and retrieve specific data based on conditions. Additionally, you can modify data using INSERT, UPDATE, and DELETE statements.

Stored Procedures and Functions: MySQL supports the creation of stored procedures and functions, which are reusable blocks of code that can be stored in the database and executed when needed. These stored routines help in encapsulating complex logic and can improve performance by reducing network traffic.

Triggers: Triggers in MySQL allow you to define actions that automatically execute in response to specific database events, such as inserting, updating, or deleting records in a table. Triggers provide a way to enforce data integrity and implement business rules at the database level.

Transactions and Concurrency Control: MySQL supports transactions, which are sequences of database operations that are executed as a single logical unit. Transactions ensure data consistency and integrity by providing ACID (Atomicity, Consistency, Isolation, Durability) properties. Additionally, MySQL offers various concurrency control mechanisms to handle multiple simultaneous transactions efficiently.

Scalability and Performance: MySQL is designed to handle large volumes of data and high traffic loads. It provides features such as indexing, query optimization, caching mechanisms, and replication to improve performance and scalability. MySQL also offers support for partitioning, which allows for distributing data across multiple physical storage devices.

Security: MySQL offers robust security features to protect the database and its data. It supports user authentication and authorization, allowing you to define user roles and permissions at various levels. MySQL also provides encryption options for securing data in transit and at rest.

Compatibility and Extensibility: MySQL is widely compatible with different operating systems, programming languages, and development frameworks. It provides connectors and APIs for popular programming languages like PHP, Python, Java, and .NET. MySQL can also be extended through plugins and user-defined functions to add custom functionality.

Community and Support: MySQL has a large and active community of users and developers, providing extensive documentation, forums, and resources for assistance. As an open-source project, MySQL benefits from continuous development, bug fixes, and feature enhancements from the community.

These features make MySQL a versatile and reliable choice for managing relational databases, offering developers and businesses a robust platform for storing, manipulating, and accessing structured data efficiently.

3.3.5 C++

C++ is a high-level, general-purpose programming language created by Danish computer scientist Bjarne Stroustrup. First released in 1985 as an extension of the C programming language, it has since expanded significantly over time; modern C++ currently has object-oriented, generic, and functional features, in addition to facilities for low-level memory manipulation. It is almost always implemented as a compiled language, and many vendors provide C++ compilers, including the Free Software Foundation, LLVM, Microsoft, Intel, Embarcadero, Oracle, and IBM.

C++ Core Guidelines

The C++ Core Guidelines [85] are an initiative led by Bjarne Stroustrup, the inventor of C++, and Herb Sutter, the convener and chair of the C++ ISO Working Group, to help programmers write 'Modern C++' by using best practices for the language standards C++11 and newer, and to help developers of compilers and static checking tools to create rules for catching bad programming practices.

The main aim is to efficiently and consistently write type and resource safe C++.

The Core Guidelines were announced in the opening keynote at CPPC on 2015.

The Guidelines are accompanied by the Guideline Support Library (GSL), a header only library of types and functions to implement the Core Guidelines and static checker tools for enforcing Guideline rules.

The developers release minor updates of the MySQL Server approximately every two months. The sources can be obtained from MySQL's website or from MySQL's GitHub repository, both under the GPL license.

A program for Arduino hardware may be written in any programming language with compilers that produce binary machine code for the target processor. Atmel provides a development environment for their 8-bit AVR and 32-bit ARM Cortex-M based microcontrollers: AVR Studio (older) and Atmel Studio (newer).

CHAPTER 4

FLOW CHART AND CIRCUIT DIAGRAM

ALGORITHM

Initialization:

Set up the NodeMCU ESP12 with the necessary libraries and connect it to the u-blox 6M GPS module.

Configure the NodeMCU ESP12 to assign a unique token to the GPS module.

Establish a connection to the server.

GPS Data Acquisition:

Read GPS data (latitude and longitude) from the u-blox 6M GPS module using the NodeMCU ESP12.

Package the GPS data along with the unique token.

Send the data to the server using an HTTP request.

Server-side Data Handling:

Implement a PHP file on the server to receive the GPS data.

Extract the unique token and GPS coordinates from the received data.

Store the data in a MySQL database, associating it with the corresponding GPS module's unique ID.

Traffic Density Calculation:

Implement another PHP file on the server to calculate the traffic density for each lane of the junction.

Retrieve the GPS data from the database based on the unique ID.

Determine the lane for each GPS module based on the GPS coordinates and the junction map data stored on the server.

Increment the car count for the corresponding lane in the junction.

Repeat the process for all lanes in a loop.

Store the updated car count in a file on the server.

Traffic Light Control:

Configure a NodeMCU present at the junction to periodically request the traffic density file from the server.

Retrieve the file containing traffic density data from the server using an HTTP request.

Calculate the time duration for each lane's green light based on the traffic density.

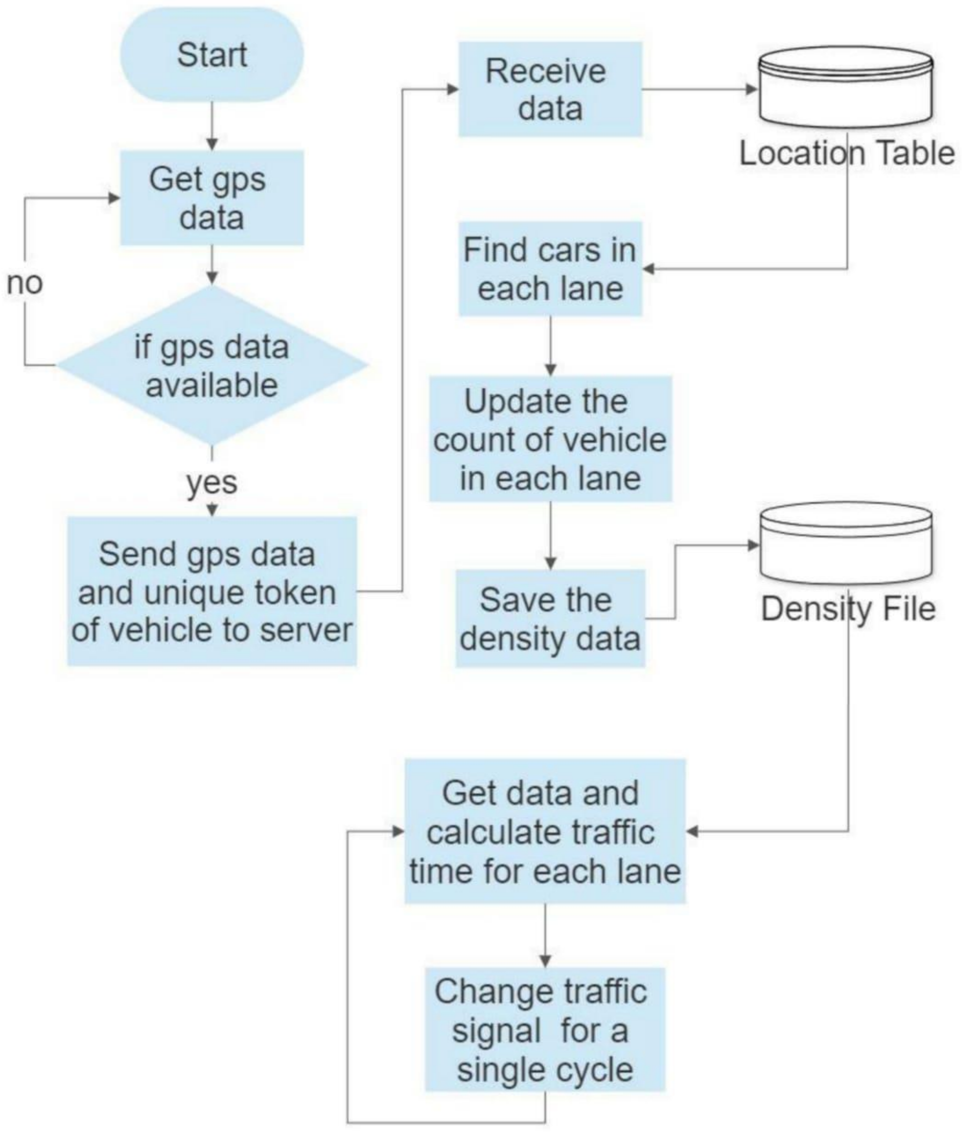
Turn on/off the green and red lights of each lane, accordingly, based on the calculated time duration.

Additional Functionality:

Implement server pages to add cars to the database.

Implement server pages to add and update the junction map used for traffic density calculation

4.1 FLOW CHART


Start
Get gps data
no
if gps data available
yes
Send gps data and unique token of vehicle to server
Receive data
Location Table
Find cars in each lane
Update the count of vehicle in each lane
Save the density data
Density File
Get data and calculate traffic time for each lane
Change traffic signal for a single cycle

4.2 PIN DIAGRAM

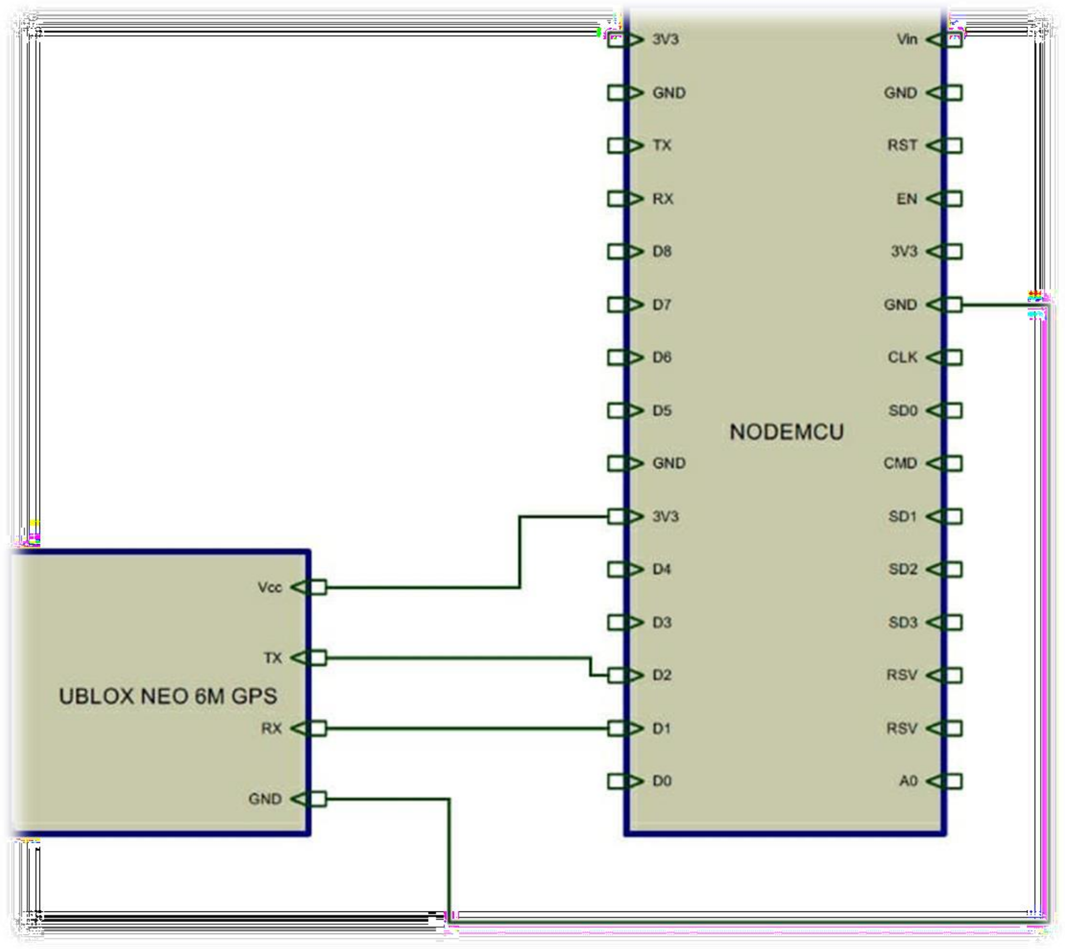


Fig 4.1 Pin Diagram

Connect vcc of Ublox neo 6m gps to 3V3 of nodemcu.

Connect transmission line of Ublox neo 6m gps to D2 pin of nodemcu.

Connect receiving line of Ublox neo 6m gps to D1 pin of nodemcu .

Connect ground pin of Ublox neo 6m gps to ground pin of nodemcu.

CHAPTER 5
WORKING AND BLOCK DIAGRAM

5.1 BLOCK DIAGRAM

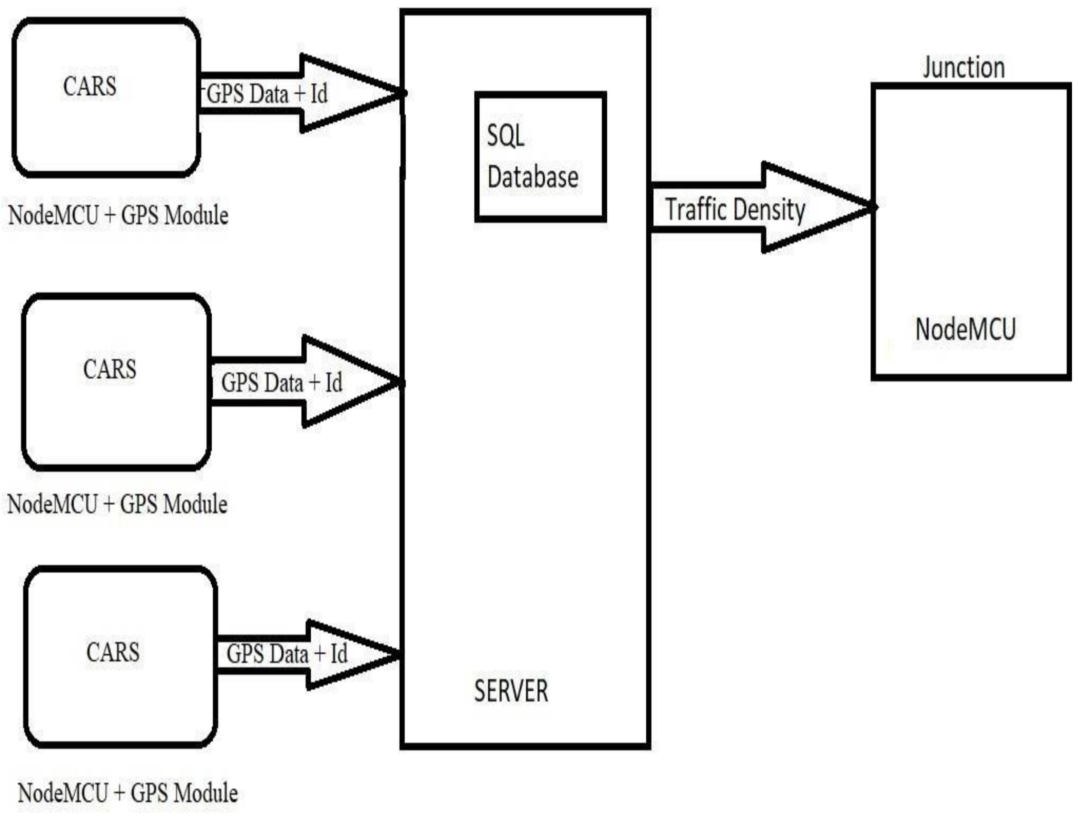


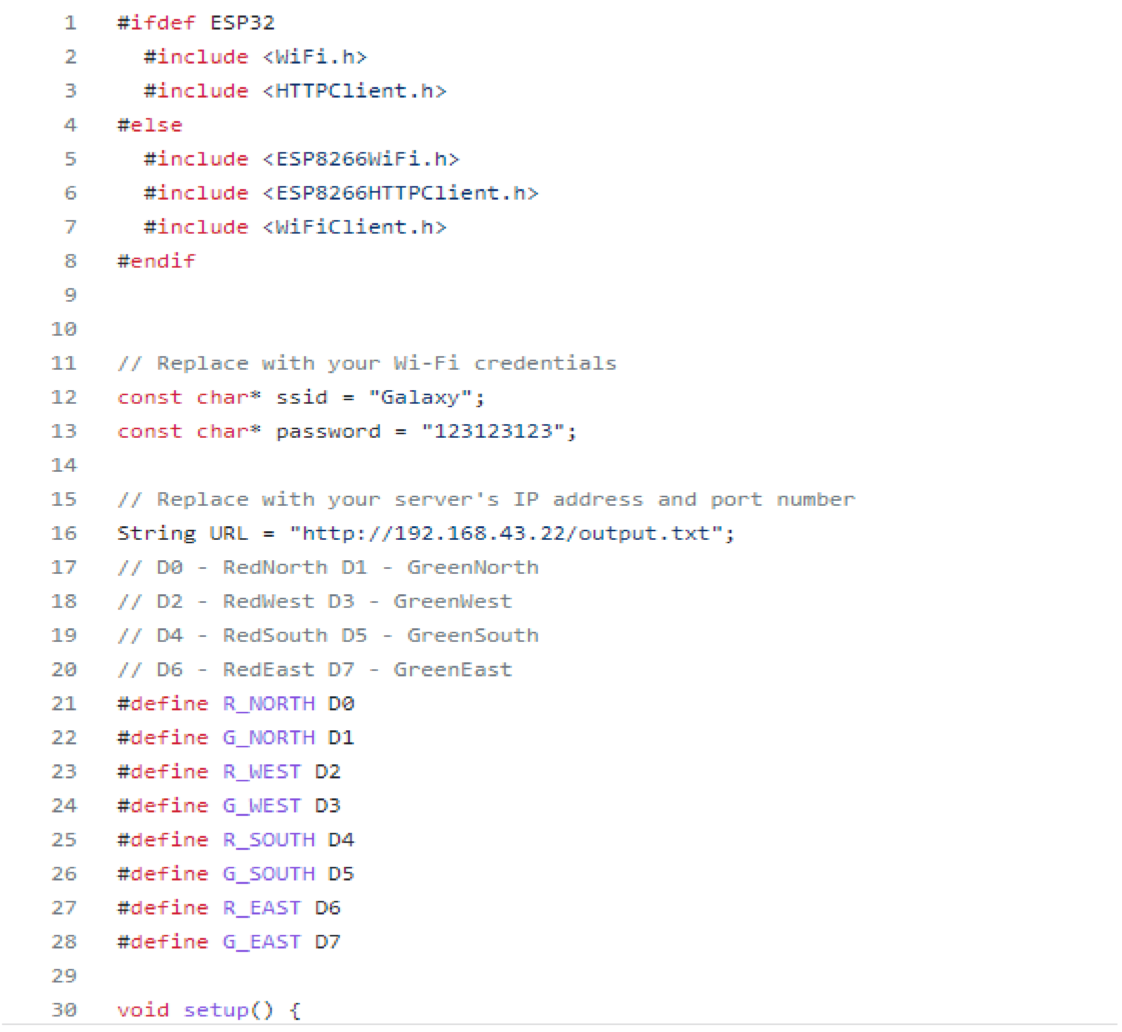
Fig 5.1 Block Diagram

5.2 WORKING

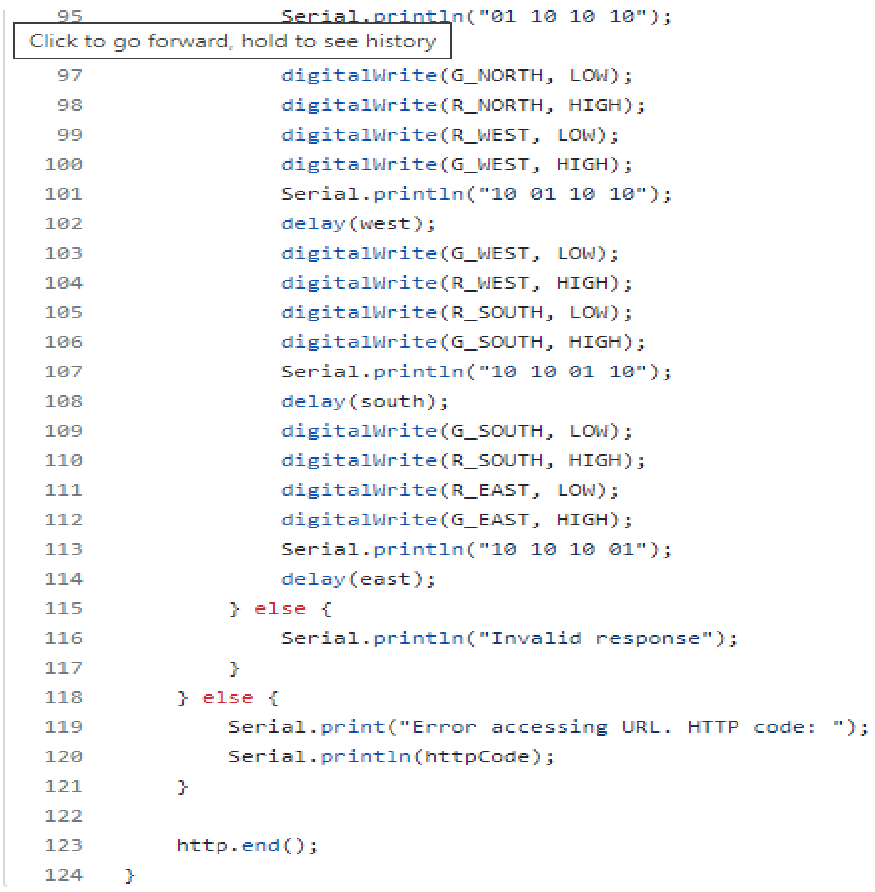
Gps module and nodemcu is connected together

Nodemcu code:

Here first we initialize a initial token value which can be considered as id of the module.Then we connect the esp module to the wifi. Once the wifi is connected, we check if we are getting any gps data from the gps module i.e. if the gps modul gets any signal from the satellite. If it gets any signal from the satellite, we get the longitude and latitude value of its location. Once a valid data is received this data and the unique token(id) is sent to our server through POST method. If the server receives the request correctly we get a valid response from the server. This collecting ofgps data and the sending to server is done every 5 second in a loop.



```
#ifdef ESP32
  #include <WiFi.h>
  #include <HTTPClient.h>
#else
  #include <ESP8266WiFi.h>
  #include <ESP8266HTTPClient.h>
  #include <WiFiClient.h>
#endif


// Replace with your Wi-Fi credentials
const char* ssid = "Galaxy";
const char* password = "123123123";

// Replace with your server's IP address and port number
String URL = "http://192.168.43.22/output.txt";
// D0 - RedNorth D1 - GreenNorth
// D2 - RedWest D3 - GreenWest
// D4 - RedSouth D5 - GreenSouth
// D6 - RedEast D7 - GreenEast
#define R_NORTH D0
#define G_NORTH D1
#define R_WEST D2
#define G_WEST D3
#define R_SOUTH D4
#define G_SOUTH D5
#define R_EAST D6
#define G_EAST D7

void setup() {
```



```
                Serial.println("01 10 10 10");
                digitalWrite(G_NORTH, LOW);
                digitalWrite(R_NORTH, HIGH);
                digitalWrite(R_WEST, LOW);
                digitalWrite(G_WEST, HIGH);
                Serial.println("10 01 10 10");
                delay(west);
                digitalWrite(G_WEST, LOW);
                digitalWrite(R_WEST, HIGH);
                digitalWrite(R_SOUTH, LOW);
                digitalWrite(G_SOUTH, HIGH);
                Serial.println("10 10 01 10");
                delay(south);
                digitalWrite(G_SOUTH, LOW);
                digitalWrite(R_SOUTH, HIGH);
                digitalWrite(R_EAST, LOW);
                digitalWrite(G_EAST, HIGH);
                Serial.println("10 10 10 01");
                delay(east);
            } else {
                Serial.println("Invalid response");
            }
        } else {
            Serial.print("Error accessing URL. HTTP code: ");
            Serial.println(httpCode);
        }

        http.end();
    }
```

Server-side code:

The server has 4 pages. Each page has a different code, but everything is working under a single database and storage.

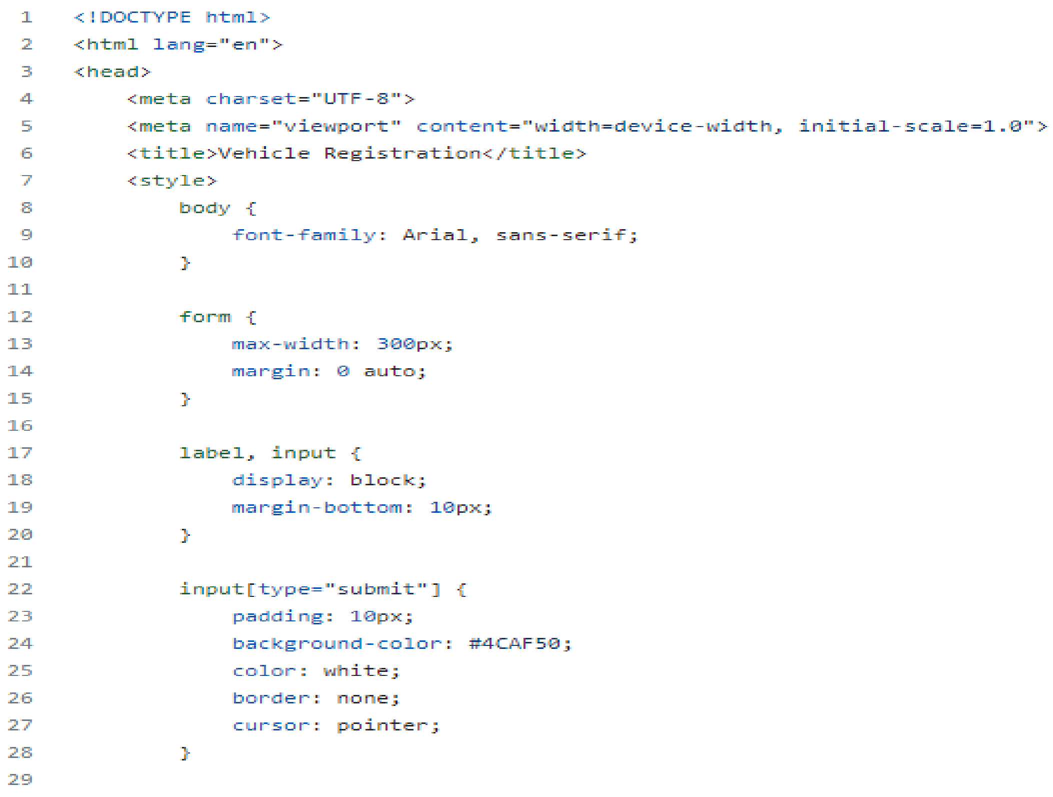
The first page is to collect the gps data and tokens sent by the nodemcu.

receive_data.php code explained:

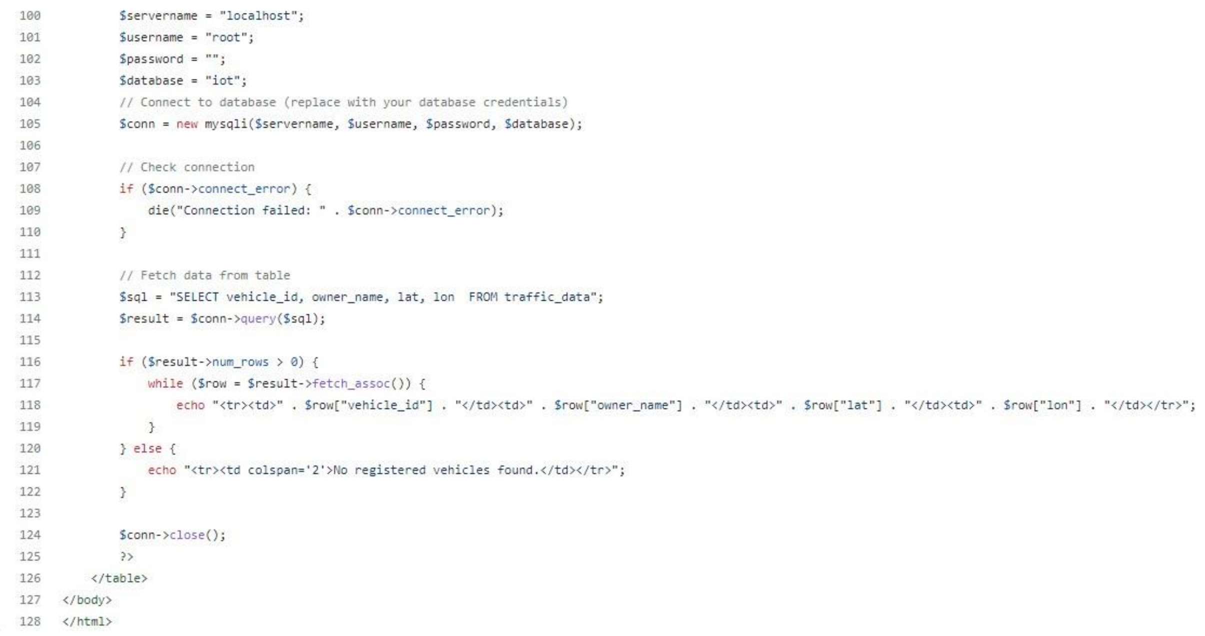
First the page connects to a table in MySQL. This table contains the vehicle details, and we also use it to store the gps data. This page listens for a POST request and when received save the value longitude, latitude, id into different variables. And the lon and lat value is updated into the table where the id received is equal. This means in the sql the query finds a column whose id is same as the is received through the request and if its same then the lan and lon value is updated for that column. By doing so the table contains the vehicle details and token and the latest longitude and latitude value of that module.

index.php code:

This is a simple page to collect the owner’s name and token from user. This page is used to add an new vehicle into the database. As we seen in receive_data.php only vehicles already present in the database gets updated. So, we need a method to insert the vehicle details and the token given to it into the server. This page has the owner name as the vehicle detail. This page contains two text boxes to enter this detail. Once the texts are entered it is inserted into the table traffic_data. This same table is used in receive_data.php. Index.php also shows the list of current vehicle details in the database.



```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Vehicle Registration</title>
    <style>
        body {
            font-family: Arial, sans-serif;
        }

        form {
            max-width: 300px;
            margin: 0 auto;
        }

        label, input {
            display: block;
            margin-bottom: 10px;
        }

        input[type="submit"] {
            padding: 10px;
            background-color: #4CAF50;
            color: white;
            border: none;
            cursor: pointer;
        }

```

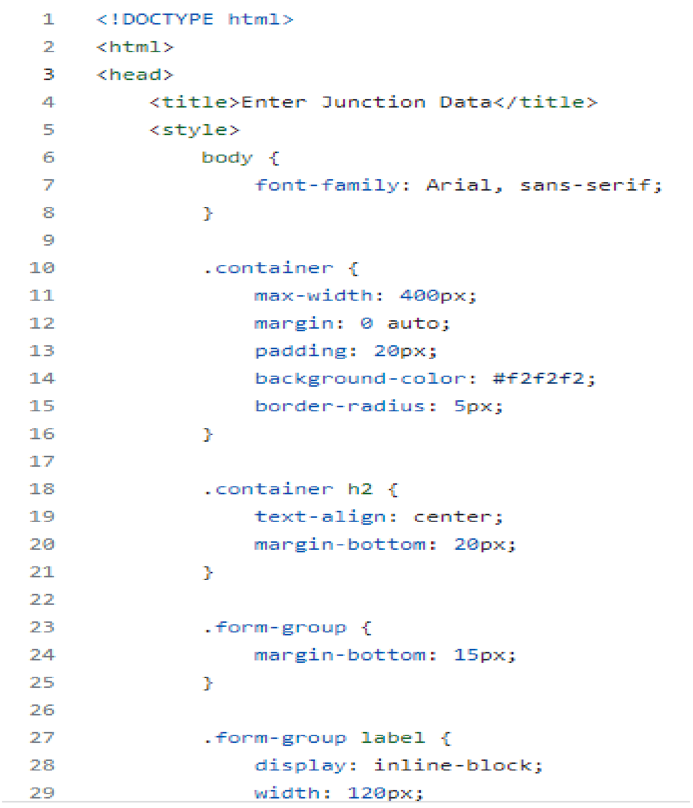


```
        $servername = "localhost";
        $username = "root";
        $password = "";
        $database = "iot";
        // Connect to database (replace with your database credentials)
        $conn = new mysqli($servername, $username, $password, $database);

        // Check connection
        if ($conn->connect_error) {
            die("Connection failed: " . $conn->connect_error);
        }

        // Fetch data from table
        $sql = "SELECT vehicle_id, owner_name, lat, lon  FROM traffic_data";
        $result = $conn->query($sql);

        if ($result->num_rows > 0) {
            while ($row = $result->fetch_assoc()) {
                echo "<tr><td>" . $row["vehicle_id"] . "</td><td>" . $row["owner_name"] . "</td><td>" . $row["lat"] . "</td><td>" . $row["lon"] . "</td></tr>";
            }
        } else {
            echo "<tr><td colspan='2'>No registered vehicles found.</td></tr>";
        }

        $conn->close();
        ?>
    </table>
</body>
</html>
```

junction.php code:

This is like the index.php. Instead of vehicle details it accepts the coordinates of a junction. It accepts latitude and longitude of 4 points of a lane like vertices of a rectangle, but the 4 points doesn'thave to make a rectangle any polygon shape will do. and order doesn't matter. The user has to enter each point of each lane manually.

junctionsimple.php is same as junction.php but only has a single text field where the points can be entered separated by a ",".

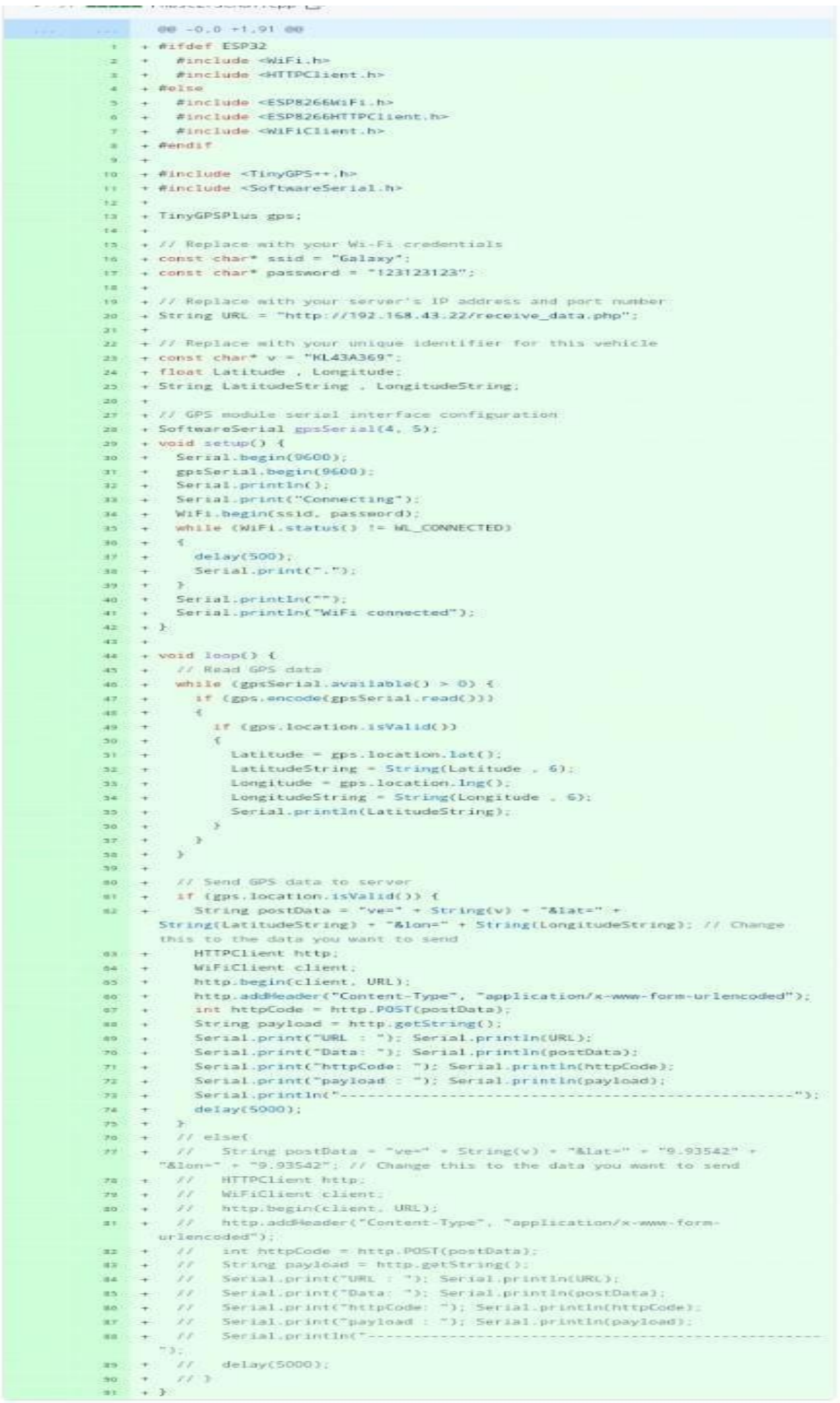


```
<!DOCTYPE html>
<html>
<head>
    <title>Enter Junction Data</title>
    <style>
        body {
            font-family: Arial, sans-serif;
        }

        .container {
            max-width: 400px;
            margin: 0 auto;
            padding: 20px;
            background-color: #f2f2f2;
            border-radius: 5px;
        }

        .container h2 {
            text-align: center;
            margin-bottom: 20px;
        }

        .form-group {
            margin-bottom: 15px;
        }

        .form-group label {
            display: inline-block;
            width: 120px;
```

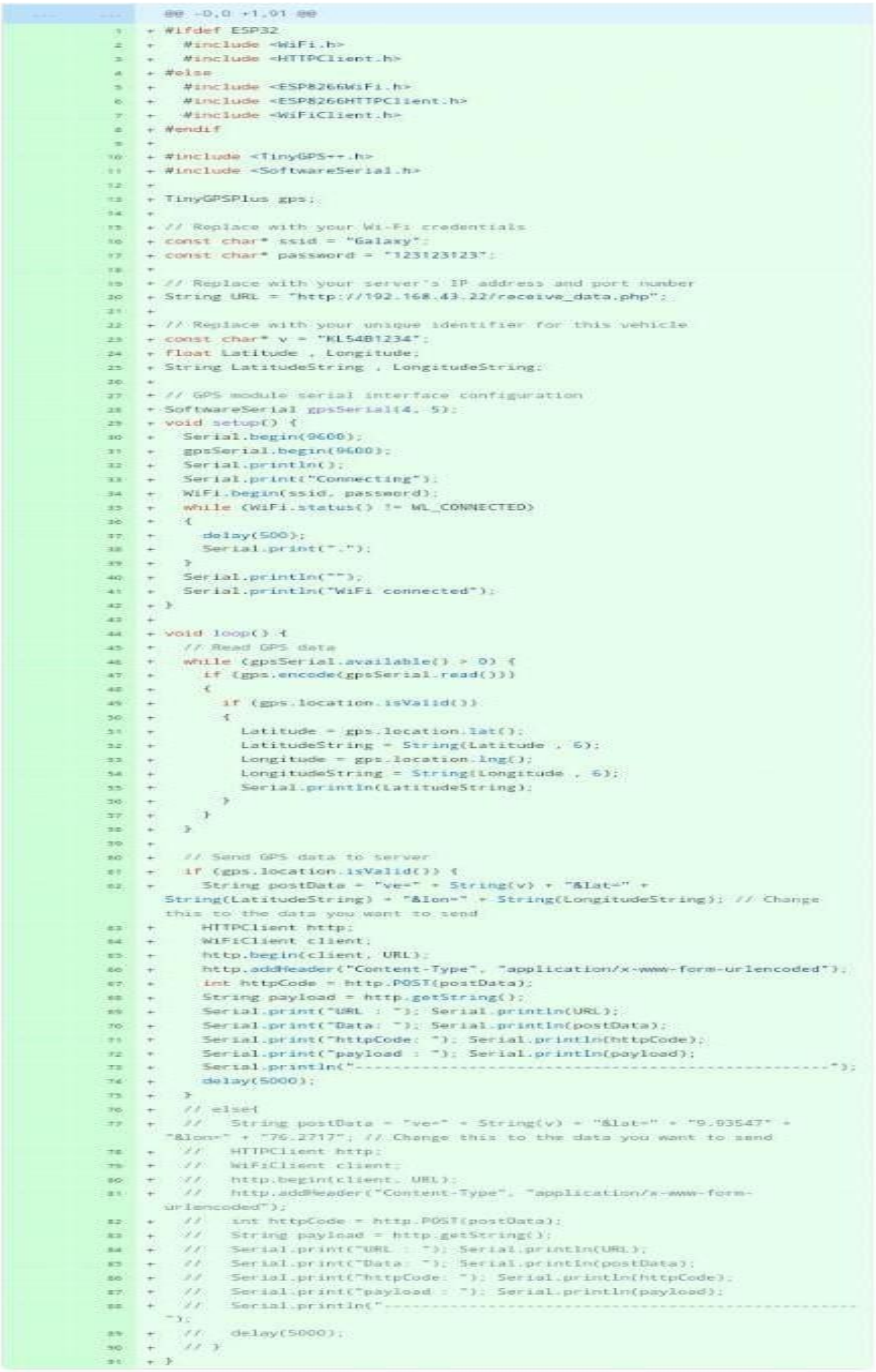
send.php code:

What it does is find vehicles that are inside the table traffic_data and check if there are any cars in the location of the lanes, we entered in the junction.php.

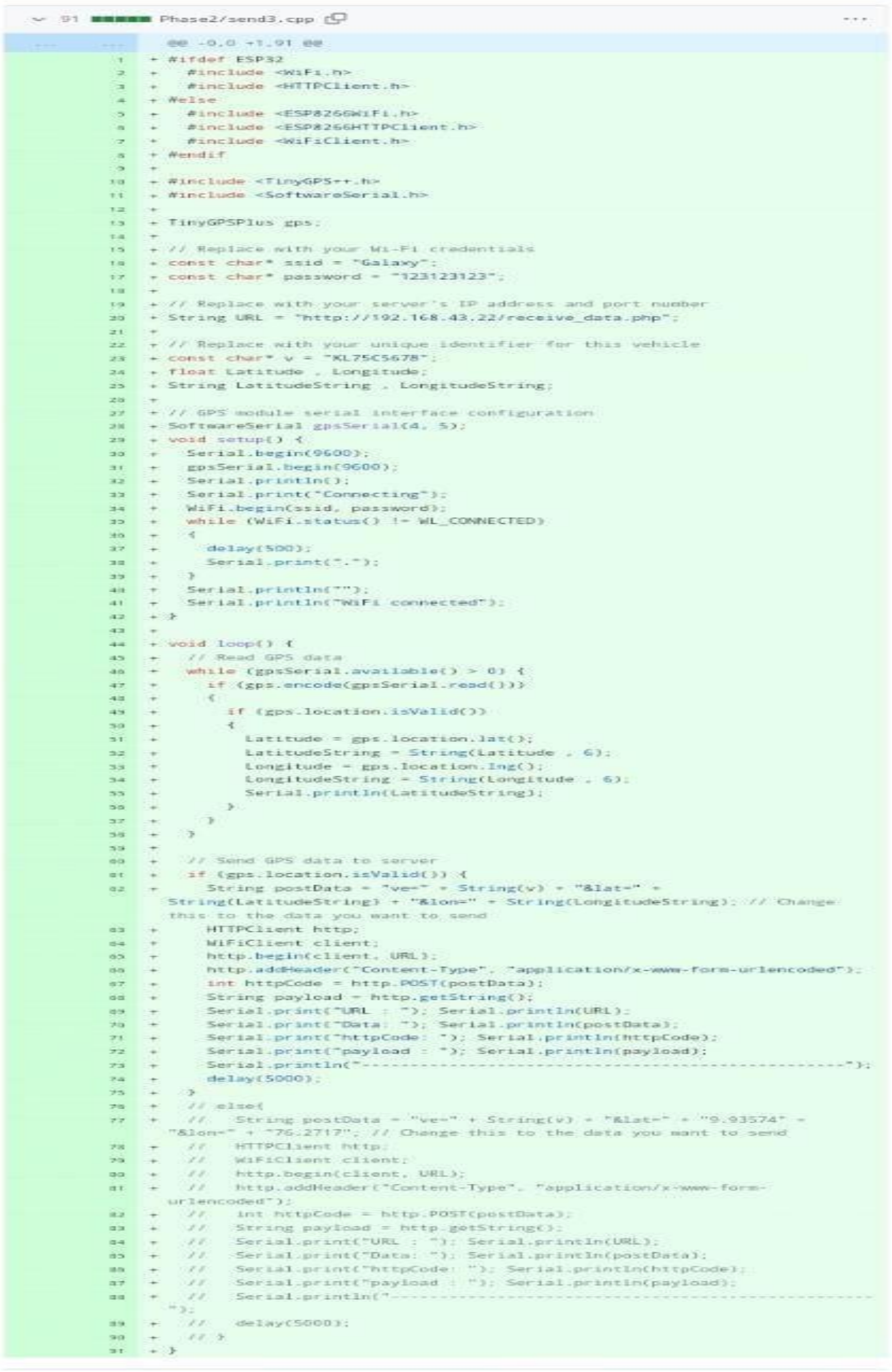
To do this we have a calculate and stop button. When the calculate button is pressed the page takes all the vehicle details in the traffic_data table in a loop and checks if it is inside any of the 4 points of a lane using a geometrical equation (equation to find if a point is inside a polygon). If the point is found to be inside a lane, then the count of vehicles inside the lane is increased. All the vehicles are checked this wayand in the end all the count values are stored in a txt file called output.txt. This text file contains the traffic density of each lane in a certain order.



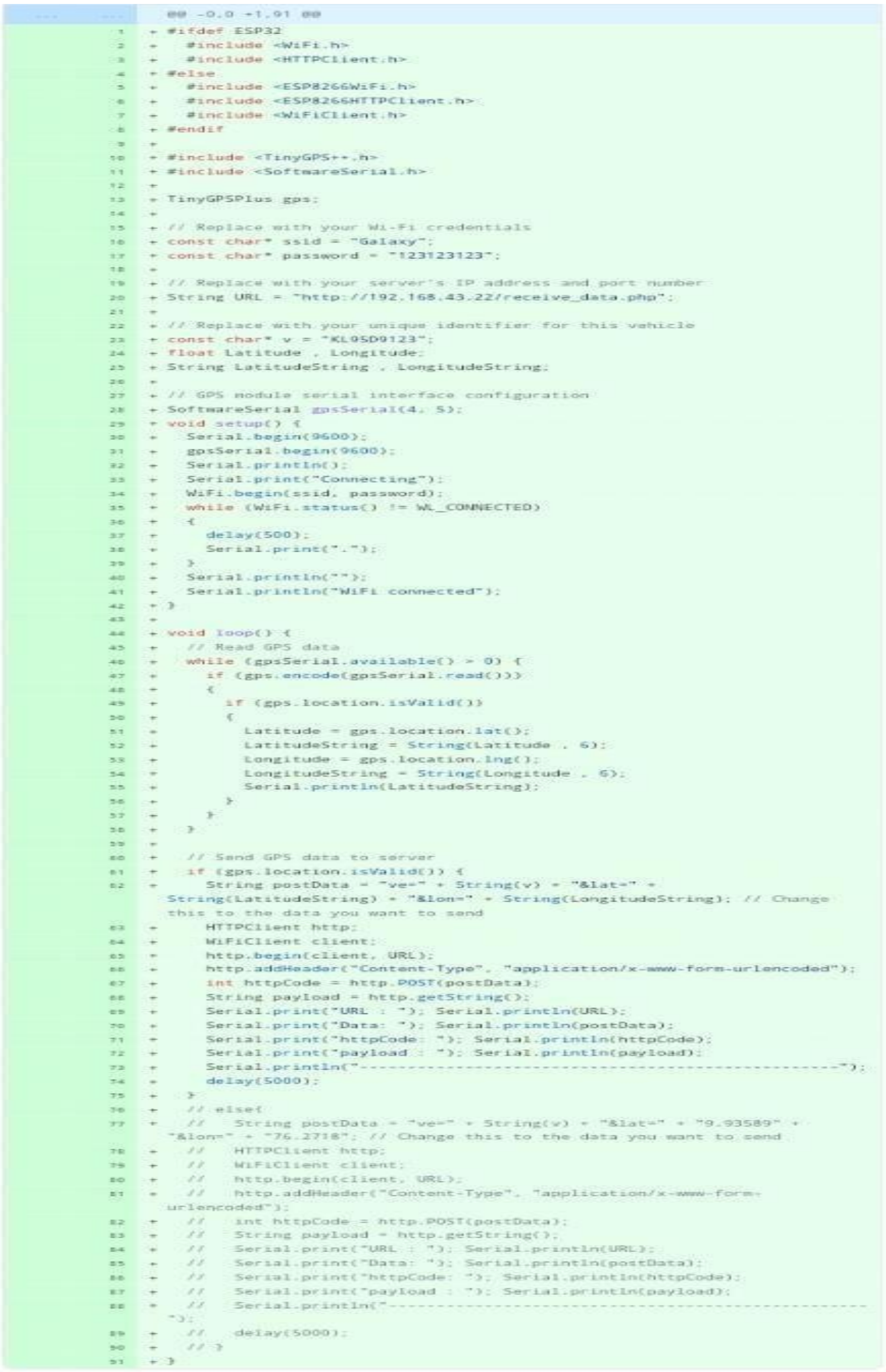
```
@@ -0,0 +1,91 @@
+ #ifdef ESP32
+   #include <WiFi.h>
+   #include <HTTPClient.h>
+ #else
+   #include <ESP8266WiFi.h>
+   #include <ESP8266HTTPClient.h>
+   #include <WiFiClient.h>
+ #endif
+
+ #include <TinyGPS++.h>
+ #include <SoftwareSerial.h>
+
+ TinyGPSPlus gps;
+
+ // Replace with your Wi-Fi credentials
+ const char* ssid = "Galaxy";
+ const char* password = "123123123";
+
+ // Replace with your server's IP address and port number
+ String URL = "http://192.168.43.22/receive_data.php";
+
+ // Replace with your unique identifier for this vehicle
+ const char* v = "KL43A369";
+ float Latitude , Longitude;
+ String LatitudeString , LongitudeString;
+
+ // GPS module serial interface configuration
+ SoftwareSerial gpsSerial(4, 5);
+ void setup() {
+   Serial.begin(9600);
+   gpsSerial.begin(9600);
+   Serial.println();
+   Serial.print("Connecting");
+   WiFi.begin(ssid, password);
+   while (WiFi.status() != WL_CONNECTED)
+   {
+     delay(500);
+     Serial.print(".");
+   }
+   Serial.println("");
+   Serial.println("WiFi connected");
+ }
+
+ void loop() {
+   // Read GPS data
+   while (gpsSerial.available() > 0) {
+     if (gps.encode(gpsSerial.read()))
+     {
+       if (gps.location.isValid())
+       {
+         Latitude = gps.location.lat();
+         LatitudeString = String(Latitude , 6);
+         Longitude = gps.location.lng();
+         LongitudeString = String(Longitude , 6);
+         Serial.println(LatitudeString);
+       }
+     }
+   }
+
+   // Send GPS data to server
+   if (gps.location.isValid()) {
+     String postData = "ve=" + String(v) + "&lat=" + String(LatitudeString) + "&lon=" + String(LongitudeString); // Change this to the data you want to send
+     HTTPClient http;
+     WiFiClient client;
+     http.begin(client, URL);
+     http.addHeader("Content-Type", "application/x-www-form-urlencoded");
+     int httpCode = http.POST(postData);
+     String payload = http.getString();
+     Serial.print("URL : "); Serial.println(URL);
+     Serial.print("Data: "); Serial.println(postData);
+     Serial.print("httpCode: "); Serial.println(httpCode);
+     Serial.print("payload : "); Serial.println(payload);
+     Serial.println("--------------------------------------------------");
+     delay(5000);
+   }
+   // else{
+   //   String postData = "ve=" + String(v) + "&lat=" + "9.93542" + "&lon=" + "9.93542"; // Change this to the data you want to send
+   //   HTTPClient http;
+   //   WiFiClient client;
+   //   http.begin(client, URL);
+   //   http.addHeader("Content-Type", "application/x-www-form-urlencoded");
+   //   int httpCode = http.POST(postData);
+   //   String payload = http.getString();
+   //   Serial.print("URL : "); Serial.println(URL);
+   //   Serial.print("Data: "); Serial.println(postData);
+   //   Serial.print("httpCode: "); Serial.println(httpCode);
+   //   Serial.print("payload : "); Serial.println(payload);
+   //   Serial.println("--------------------------------------------------");
+   //   delay(5000);
+   // }
+ }
```



```
@@ -0,0 +1,91 @@
+ #ifdef ESP32
+   #include <WiFi.h>
+   #include <HTTPClient.h>
+ #else
+   #include <ESP8266WiFi.h>
+   #include <ESP8266HTTPClient.h>
+   #include <WiFiClient.h>
+ #endif
+
+ #include <TinyGPS++.h>
+ #include <SoftwareSerial.h>
+
+ TinyGPSPlus gps;
+
+ // Replace with your Wi-Fi credentials
+ const char* ssid = "Galaxy";
+ const char* password = "123123123";
+
+ // Replace with your server's IP address and port number
+ String URL = "http://192.168.43.22/receive_data.php";
+
+ // Replace with your unique identifier for this vehicle
+ const char* v = "KL5481234";
+ float Latitude , Longitude;
+ String LatitudeString , LongitudeString;
+
+ // GPS module serial interface configuration
+ SoftwareSerial gpsSerial(4, 5);
+ void setup() {
+   Serial.begin(9600);
+   gpsSerial.begin(9600);
+   Serial.println();
+   Serial.print("Connecting");
+   WiFi.begin(ssid, password);
+   while (WiFi.status() != WL_CONNECTED)
+   {
+     delay(500);
+     Serial.print(".");
+   }
+   Serial.println("");
+   Serial.println("WiFi connected");
+ }
+
+ void loop() {
+   // Read GPS data
+   while (gpsSerial.available() > 0) {
+     if (gps.encode(gpsSerial.read()))
+     {
+       if (gps.location.isValid())
+       {
+         Latitude = gps.location.lat();
+         LatitudeString = String(Latitude , 6);
+         Longitude = gps.location.lng();
+         LongitudeString = String(Longitude , 6);
+         Serial.println(LatitudeString);
+       }
+     }
+   }
+
+   // Send GPS data to server
+   if (gps.location.isValid()) {
+     String postData = "ve=" + String(v) + "&lat=" +
String(LatitudeString) + "&lon=" + String(LongitudeString); // Change
this to the data you want to send
+     HTTPClient http;
+     WiFiClient client;
+     http.begin(client, URL);
+     http.addHeader("Content-Type", "application/x-www-form-urlencoded");
+     int httpCode = http.POST(postData);
+     String payload = http.getString();
+     Serial.print("URL : "); Serial.println(URL);
+     Serial.print("Data: "); Serial.println(postData);
+     Serial.print("httpCode: "); Serial.println(httpCode);
+     Serial.print("payload : "); Serial.println(payload);
+     Serial.println("--------------------------------------------------");
+     delay(5000);
+   }
+   // else{
+   //   String postData = "ve=" + String(v) + "&lat=" + "9.93547" +
"&lon=" + "76.2717"; // Change this to the data you want to send
+   //   HTTPClient http;
+   //   WiFiClient client;
+   //   http.begin(client, URL);
+   //   http.addHeader("Content-Type", "application/x-www-form-
urlencoded");
+   //   int httpCode = http.POST(postData);
+   //   String payload = http.getString();
+   //   Serial.print("URL : "); Serial.println(URL);
+   //   Serial.print("Data: "); Serial.println(postData);
+   //   Serial.print("httpCode: "); Serial.println(httpCode);
+   //   Serial.print("payload : "); Serial.println(payload);
+   //   Serial.println("-----------------------------------------------
");
+   //   delay(5000);
+   // }
+ }
```



```
91 Phase2/send3.cpp
@@ -0,0 +1,91 @@
+ #ifdef ESP32
+   #include <WiFi.h>
+   #include <HTTPClient.h>
+ #else
+   #include <ESP8266WiFi.h>
+   #include <ESP8266HTTPClient.h>
+   #include <WiFiClient.h>
+ #endif
+
+ #include <TinyGPS++.h>
+ #include <SoftwareSerial.h>
+
+ TinyGPSPlus gps;
+
+ // Replace with your Wi-Fi credentials
+ const char* ssid = "Galaxy";
+ const char* password = "123123123";
+
+ // Replace with your server's IP address and port number
+ String URL = "http://192.168.43.22/receive_data.php";
+
+ // Replace with your unique identifier for this vehicle
+ const char* v = "KL75C5678";
+ float Latitude , Longitude;
+ String LatitudeString , LongitudeString;
+
+ // GPS module serial interface configuration
+ SoftwareSerial gpsSerial(4, 5);
+ void setup() {
+   Serial.begin(9600);
+   gpsSerial.begin(9600);
+   Serial.println();
+   Serial.print("Connecting");
+   WiFi.begin(ssid, password);
+   while (WiFi.status() != WL_CONNECTED)
+   {
+     delay(500);
+     Serial.print(".");
+   }
+   Serial.println("");
+   Serial.println("WiFi connected");
+ }
+
+ void loop() {
+   // Read GPS data
+   while (gpsSerial.available() > 0) {
+     if (gps.encode(gpsSerial.read()))
+     {
+       if (gps.location.isValid())
+       {
+         Latitude = gps.location.lat();
+         LatitudeString = String(Latitude , 6);
+         Longitude = gps.location.lng();
+         LongitudeString = String(Longitude , 6);
+         Serial.println(LatitudeString);
+       }
+     }
+   }
+
+   // Send GPS data to server
+   if (gps.location.isValid()) {
+     String postData = "ve=" + String(v) + "&lat=" +
String(LatitudeString) + "&lon=" + String(LongitudeString); // Change
this to the data you want to send
+     HTTPClient http;
+     WiFiClient client;
+     http.begin(client, URL);
+     http.addHeader("Content-Type", "application/x-www-form-urlencoded");
+     int httpCode = http.POST(postData);
+     String payload = http.getString();
+     Serial.print("URL : "); Serial.println(URL);
+     Serial.print("Data: "); Serial.println(postData);
+     Serial.print("httpCode: "); Serial.println(httpCode);
+     Serial.print("payload : "); Serial.println(payload);
+     Serial.println("--------------------------------------------------");
+     delay(5000);
+   }
+   // else{
+   //   String postData = "ve=" + String(v) + "&lat=" + "9.93574" +
"&lon=" + "76.2717"; // Change this to the data you want to send
+   //   HTTPClient http;
+   //   WiFiClient client;
+   //   http.begin(client, URL);
+   //   http.addHeader("Content-Type", "application/x-www-form-
urlencoded");
+   //   int httpCode = http.POST(postData);
+   //   String payload = http.getString();
+   //   Serial.print("URL : "); Serial.println(URL);
+   //   Serial.print("Data: "); Serial.println(postData);
+   //   Serial.print("httpCode: "); Serial.println(httpCode);
+   //   Serial.print("payload : "); Serial.println(payload);
+   //   Serial.println("--------------------------------------------------
");
+   //   delay(5000);
+   // }
+ }
```

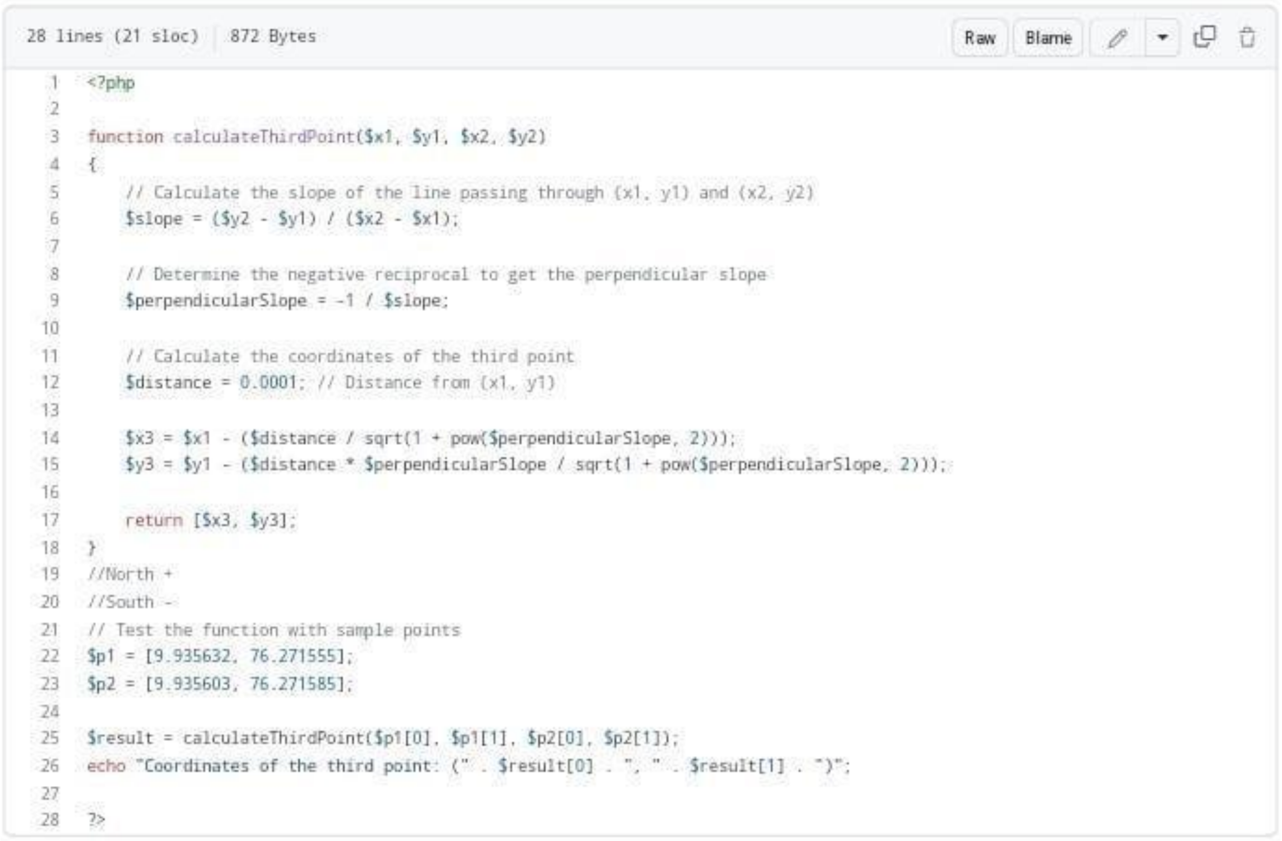


```
@@ -0,0 +1,91 @@
+ #ifdef ESP32
+   #include <WiFi.h>
+   #include <HTTPClient.h>
+ #else
+   #include <ESP8266WiFi.h>
+   #include <ESP8266HTTPClient.h>
+   #include <WiFiClient.h>
+ #endif
+
+ #include <TinyGPS++.h>
+ #include <SoftwareSerial.h>
+
+ TinyGPSPlus gps;
+
+ // Replace with your Wi-Fi credentials
+ const char* ssid = "Galaxy";
+ const char* password = "123123123";
+
+ // Replace with your server's IP address and port number
+ String URL = "http://192.168.43.22/receive_data.php";
+
+ // Replace with your unique identifier for this vehicle
+ const char* v = "KL05D9123";
+ float Latitude , Longitude;
+ String LatitudeString , LongitudeString;
+
+ // GPS module serial interface configuration
+ SoftwareSerial gpsSerial(4, 5);
+ void setup() {
+   Serial.begin(9600);
+   gpsSerial.begin(9600);
+   Serial.println();
+   Serial.print("Connecting");
+   WiFi.begin(ssid, password);
+   while (WiFi.status() != WL_CONNECTED)
+   {
+     delay(500);
+     Serial.print(".");
+   }
+   Serial.println("");
+   Serial.println("WiFi connected");
+ }
+
+ void loop() {
+   // Read GPS data
+   while (gpsSerial.available() > 0) {
+     if (gps.encode(gpsSerial.read()))
+     {
+       if (gps.location.isValid())
+       {
+         Latitude = gps.location.lat();
+         LatitudeString = String(Latitude , 6);
+         Longitude = gps.location.lng();
+         LongitudeString = String(Longitude , 6);
+         Serial.println(LatitudeString);
+       }
+     }
+   }
+
+   // Send GPS data to server
+   if (gps.location.isValid()) {
+     String postData = "ve=" + String(v) + "&lat=" +
String(LatitudeString) + "&lon=" + String(LongitudeString); // Change
this to the data you want to send
+     HTTPClient http;
+     WiFiClient client;
+     http.begin(client, URL);
+     http.addHeader("Content-Type", "application/x-www-form-urlencoded");
+     int httpCode = http.POST(postData);
+     String payload = http.getString();
+     Serial.print("URL : "); Serial.println(URL);
+     Serial.print("Data: "); Serial.println(postData);
+     Serial.print("httpCode: "); Serial.println(httpCode);
+     Serial.print("payload : "); Serial.println(payload);
+     Serial.println("--------------------------------------------------");
+     delay(5000);
+   }
+   // else{
+   //    String postData = "ve=" + String(v) + "&lat=" + "9.93589" +
"&lon=" + "76.2718"; // Change this to the data you want to send
+   //    HTTPClient http;
+   //    WiFiClient client;
+   //    http.begin(client, URL);
+   //    http.addHeader("Content-Type", "application/x-www-form-
urlencoded");
+   //    int httpCode = http.POST(postData);
+   //    String payload = http.getString();
+   //    Serial.print("URL : "); Serial.println(URL);
+   //    Serial.print("Data: "); Serial.println(postData);
+   //    Serial.print("httpCode: "); Serial.println(httpCode);
+   //    Serial.print("payload : "); Serial.println(payload);
+   //    Serial.println("---------------------------------------------
");
+   //    delay(5000);
+   // }
+ }
```

getDataFromServer.cpp code:

Here the Nodemcu tries to connect to the Wi-Fi. Once connected it sends a request to connect to the file output.txt in the server. The server replies with the content of the file. From the received content we get the traffic density of each lane. Usually in traffic signal junction what happens is there is time to circle around the lanes that is if the looping time is 120 seconds and there are four lanes then 40 seconds is given for each lane as green light time and its loops like this. Here what we do is based on the traffic density we divide the time according to the ratio. suppose if the ration of roads south, east, north, west in order is 2, 0, 1, 1 there the south has 2 vehicles and rest respectively then the 120 second is divided on the ration i.e. 60, 0, 30, 30 the east has 0 time given as there is vehicle there and hence doesn't need green time. The actual equation used is (looptime/sumOfDensity) *density[I].

in above south case it will be (120/4) *2 = 60. Once the time is found the nodemcu turns the lights on andoff according to the new time given for each lane.



```
<?php

function calculateThirdPoint($x1, $y1, $x2, $y2)
{
    // Calculate the slope of the line passing through (x1, y1) and (x2, y2)
    $slope = ($y2 - $y1) / ($x2 - $x1);

    // Determine the negative reciprocal to get the perpendicular slope
    $perpendicularSlope = -1 / $slope;

    // Calculate the coordinates of the third point
    $distance = 0.0001; // Distance from (x1, y1)

    $x3 = $x1 - ($distance / sqrt(1 + pow($perpendicularSlope, 2)));
    $y3 = $y1 - ($distance * $perpendicularSlope / sqrt(1 + pow($perpendicularSlope, 2)));

    return [$x3, $y3];
}
//North +
//South -
// Test the function with sample points
$p1 = [9.935632, 76.271555];
$p2 = [9.935603, 76.271585];

$result = calculateThirdPoint($p1[0], $p1[1], $p2[0], $p2[1]);
echo "Coordinates of the third point: (" . $result[0] . ", " . $result[1] . ")";

?>
```

CHAPTER 6
CONCLUSION

In conclusion, the project on the identification of vehicles using a unique ID and traffic automation, with the implementation of MySQL, XAMPP, PHP, and C++, showcases a comprehensive approach to solving the challenges associated with vehicle management and traffic control.

The utilization of MySQL as the database management system offers a reliable and efficient solution for storing and retrieving vehicle information. The unique ID assigned to each vehicle serves as a primary key, allowing for seamless tracking, management, and retrieval of specific vehicle data. By leveraging the power of MySQL, the project ensures the integrity and security of the stored information.

XAMPP, acting as a web server solution, provides the necessary infrastructure to host the project's web-based interface or application. It enables the integration of PHP, HTML, CSS, and other technologies to create a user-friendly interface for managing vehicles and automating traffic processes. Through XAMPP, the project becomes accessible to authorized users, facilitating easy interaction and control.

PHP serves as a scripting language within the project, enabling dynamic web page generation and interaction with the database. It enhances the functionality of the web interface by allowing seamless data exchange between the user and the database. PHP plays a pivotal role in implementing the logic behind vehicle identification, traffic automation, and the associated algorithms.

Additionally, the project incorporates C++, a powerful programming language, to develop the core functionalities of the system. C++ allows for efficient algorithm design and implementation, making it suitable for complex operations such as traffic signal control, route optimization, and data processing. By leveraging C++, the project achieves high performance and robustness in its automation processes.

By combining MySQL, XAMPP, PHP, and C++, the project successfully addresses the challenges of vehicle identification and traffic automation. The unique ID system enables accurate tracking and management of vehicles, contributing to enhanced security and improved traffic control. The automation processes, powered by efficient algorithms implemented in C++, optimize traffic flow, reduce congestion, and enhance overall road safety.

In conclusion, the project's utilization of MySQL, XAMPP, PHP, and C++ demonstrates a comprehensive and effective approach to vehicle identification and traffic automation. Through the integration of these technologies, the project provides a robust system for managing vehicles, optimizing traffic, and improving transportation efficiency and safety.

CHAPTER 7
REFERENCE

[1]. L.Sherly Puspa Annabel and Kripa Sekaran,”Automatic signal clearance system using density based traffic control”

[2]. Md.Zaved Parvez,Khondker Zakir Ahmed,Quazi Raguib Mahfuz and Md.Saifur Rahman,”A theoretical model of GSM network based vehicle tracking system”

[3]. Er.Faruk Bin Poyen’,Amit Kumar Bhakta,B.Durga Manohar,Imran Ali,ArgyaSantra and AwanishPratap Rao,”Density based traffic control”

[4] Yuan wang, zhaoshen yang, qing guan,” Traffic Coordination and Control Model of RegionalBoundary Based on Fuzzy Control”

[5] Muhammad Hanif tunio, Imran Memon,Ghulam Ali Mallah , Noor Ahmed Shaikh

[6] Lavanya herle, Poonam Sharma,” Vehicle detection and identification in an unconstrained environment”

[7] Fatemeh nafar, Hossein shamsi “Design and Implementation of an RFID-GSM-Based Vehicle Identification System on Highways”