INTERNET OF THINGS-GROUP 4
TRAFFIC MANAGEMENT SYSTEM

STUDENT NAME :S MALINI

REGISTER NUMBER: 822621106008

NM ID : au822621106008

COLLEGE CODE : 8226

COLLEGE NAME : ARIFA INSTITUTE OF TECHNOLOGY

EMAIL ID : malinivani03@gmail.com

INTRODUCTION

• Hypothesis

A smart traffic management system utilizing sensor data, communication and auto- mated algorithms is to be developed to keep traffic flowing more smoothly. The aim is to optimally control the duration of green or red light for a specific traffic lightat an intersection. The traffic signals should not flash the same stretch of green orred all the time, but should depend on the number of cars present. When traffic is heavy in one direction, the green lights should stay on longer; less traffic should mean the red lights should be on for longer time interval. This solution is expected to eliminate inefficiencies at intersections and minimize the cost of commuting and pollution.

• Motivation

In 2014, 54% of the total global population was urban residents. The prediction was a growth of nearly 2% each year until 2020 leading to more pressure on the trans- portation system of cities. Additionally, the high cost of accommodation in business districts lead to urban employees living far away from their place of work/education and therefore having to commute back and forth between their place of residenceand their place of work. More vehicles moving need to be accommodated over a

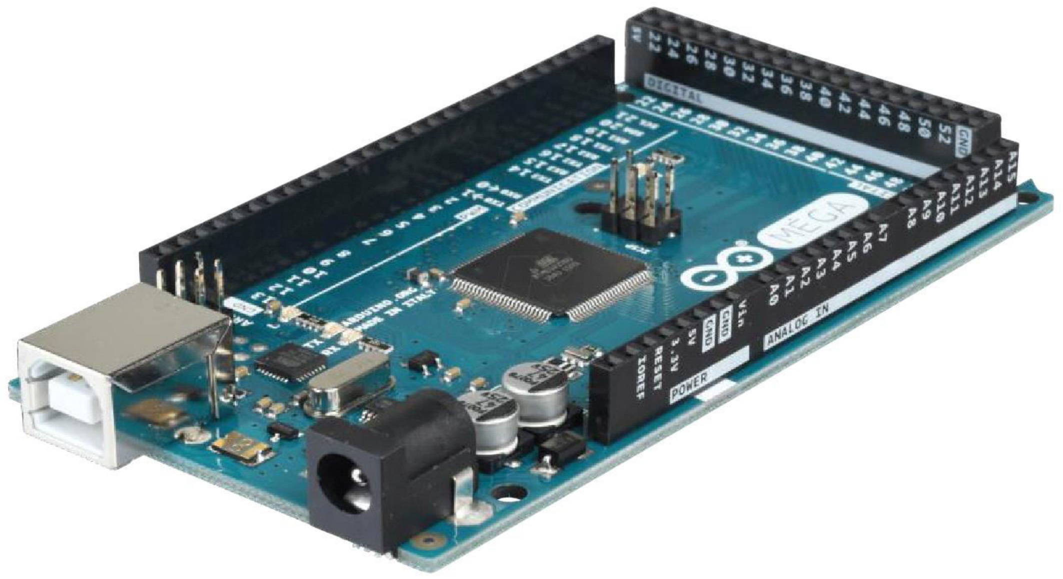
IoT in Traffic Management

Traffic management is one of the biggest infrastructure hurdles faced by developing countries today. Developed countries and smart cities are already using IoT and to their advantage to minimize issues related to traffic. The culture of the car has been cultivated speedily among people in all types of nations. In most cities, it is common for people to prefer riding their own vehicles no matter how good or bad the public transportation is or considering how much time and money is it going to take for them to reach their destination.

REQUIREMENTS

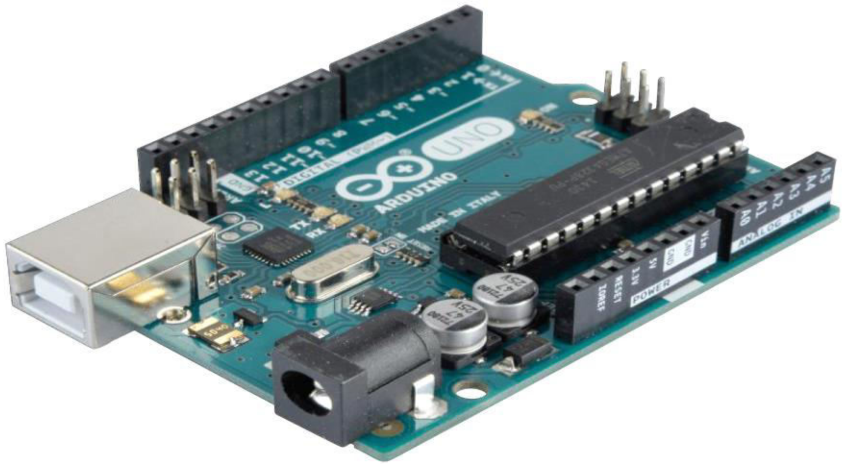
Hardware Components

- **Microcontroller (Arduino Mega 2560)**: The Arduino Mega 2560 is a micro- controller 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



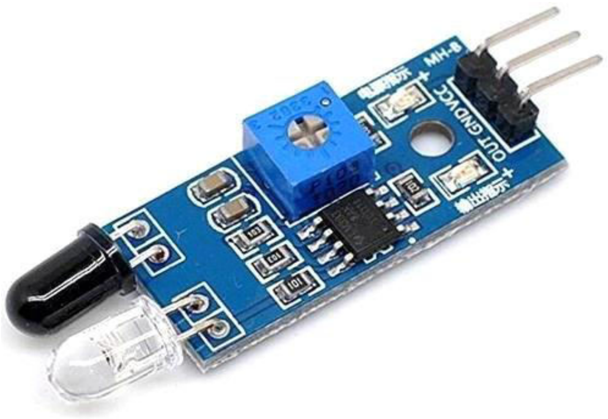
Arduino Mega 2560.

- **Microcontroller (Arduino Uno)**: The Arduino UNO is an open-source micro- controller 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.


Arduino Uno.

- **LEDs**: LEDs are used for the purpose of signaling according to the trafficcondition.

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



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

- ## Software Requirement

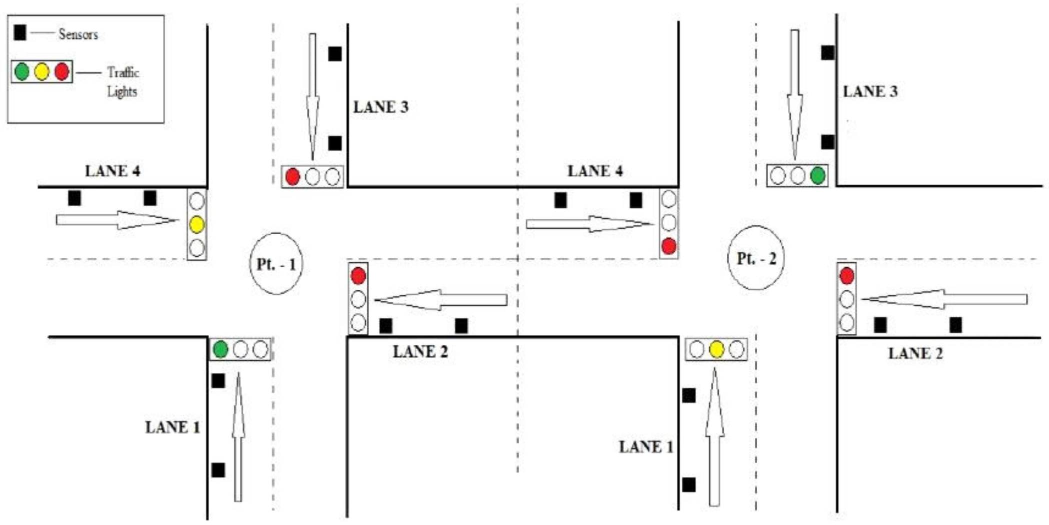
- **Arduino IDE**: The Arduino integrated development environment (IDE) is a cross-platform application (for Windows, MacOS, Linux) that is written in the programming language Java. It is used to write and upload programs to Arduino board.
The source code for the IDE is released under the GNU General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

- **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.

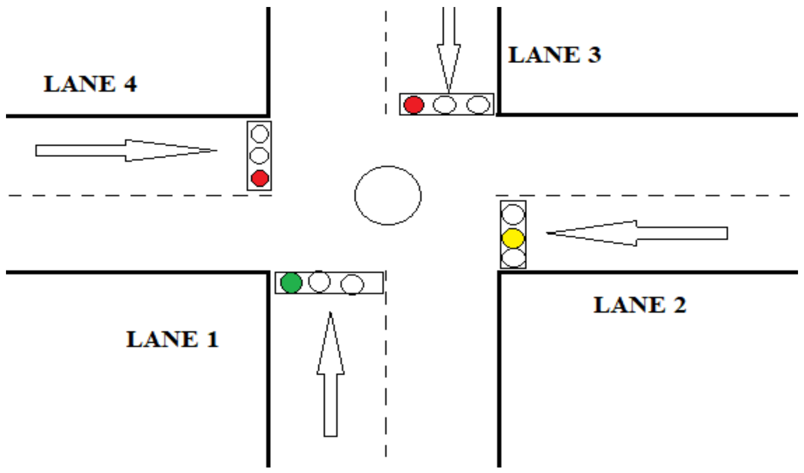
Method

In this proposed system, the traffic lights are LEDs and the car counting sensor is an ultrasonic sensor. Both blocks are connected to a Microcontroller using physicalwires. The Microcontroller is the traffic light controller which receives the collectedsensor data and manages the traffic lights by switching between green, yellow and red. The Microcontroller computes the number of cars in the street of the inter- section it is monitoring based on the distances measured by the ultrasonic sensor and the timing between those measurements. The Microcontroller then sends the number of cars every minute to the local server. This communication is done using the Microcontroller serial port. The local server exchanges the data received with the cloud server in order to better predict the changes in timings of the traffic light.This communication is done using Wi-Fi. More specifically, the cloud server uses an equation that takes the data received (number of cars) as input then determinesthe time interval of LEDs needed for a smooth traffic flow. This calculated timeis then compared to the current actual time of the LEDs (this data is saved in a database on the cloud server). The server then comes up with a decision. If the current actual green time is less than the calculated time, the decision is to increasethe green time, else to decrease the green time

A View of Signals at Different Lanes

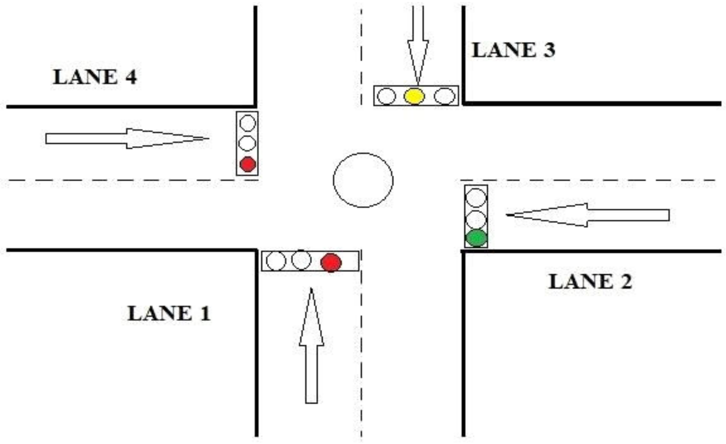


Control of previous Intersection



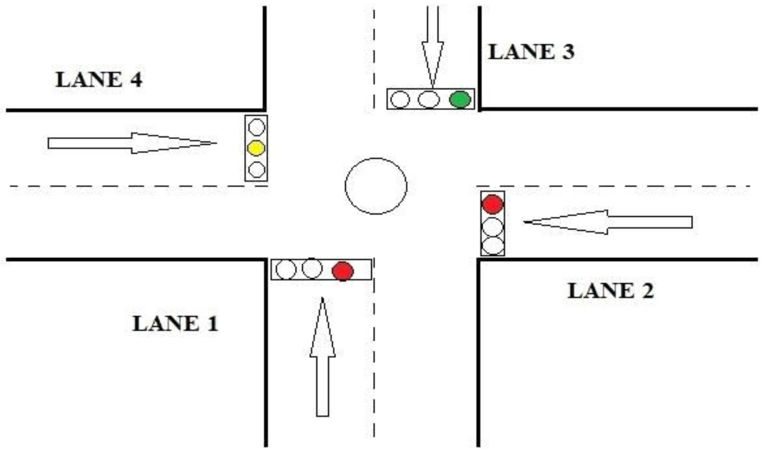
Signal at Lane 1

In the above figure, Lane 1 is open with green signal and other lanes are closed withred signal.



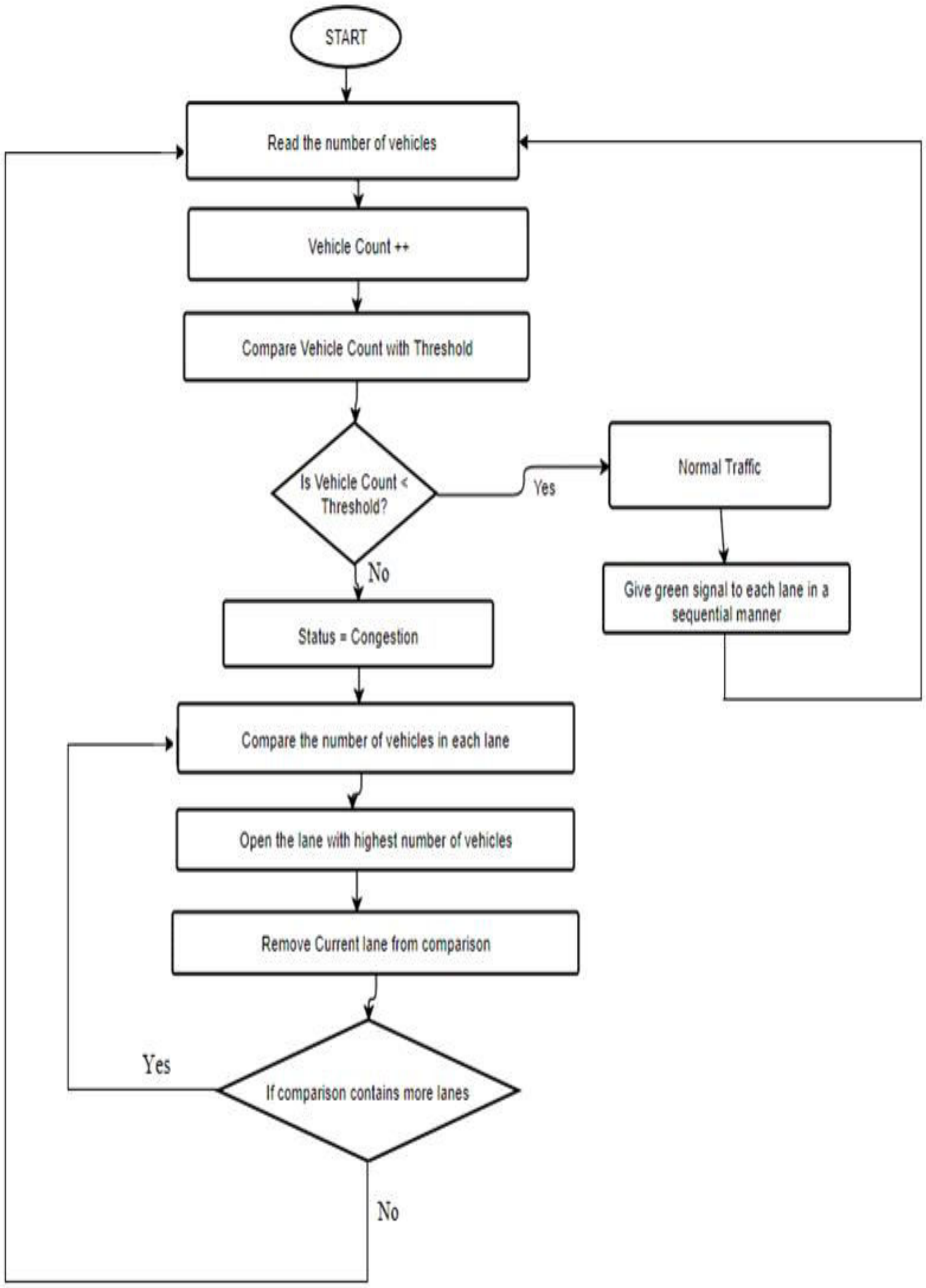
Signal at Lane 2

In the above figure, Lane 2 is open with green signal and other lanes are closed withred signal.



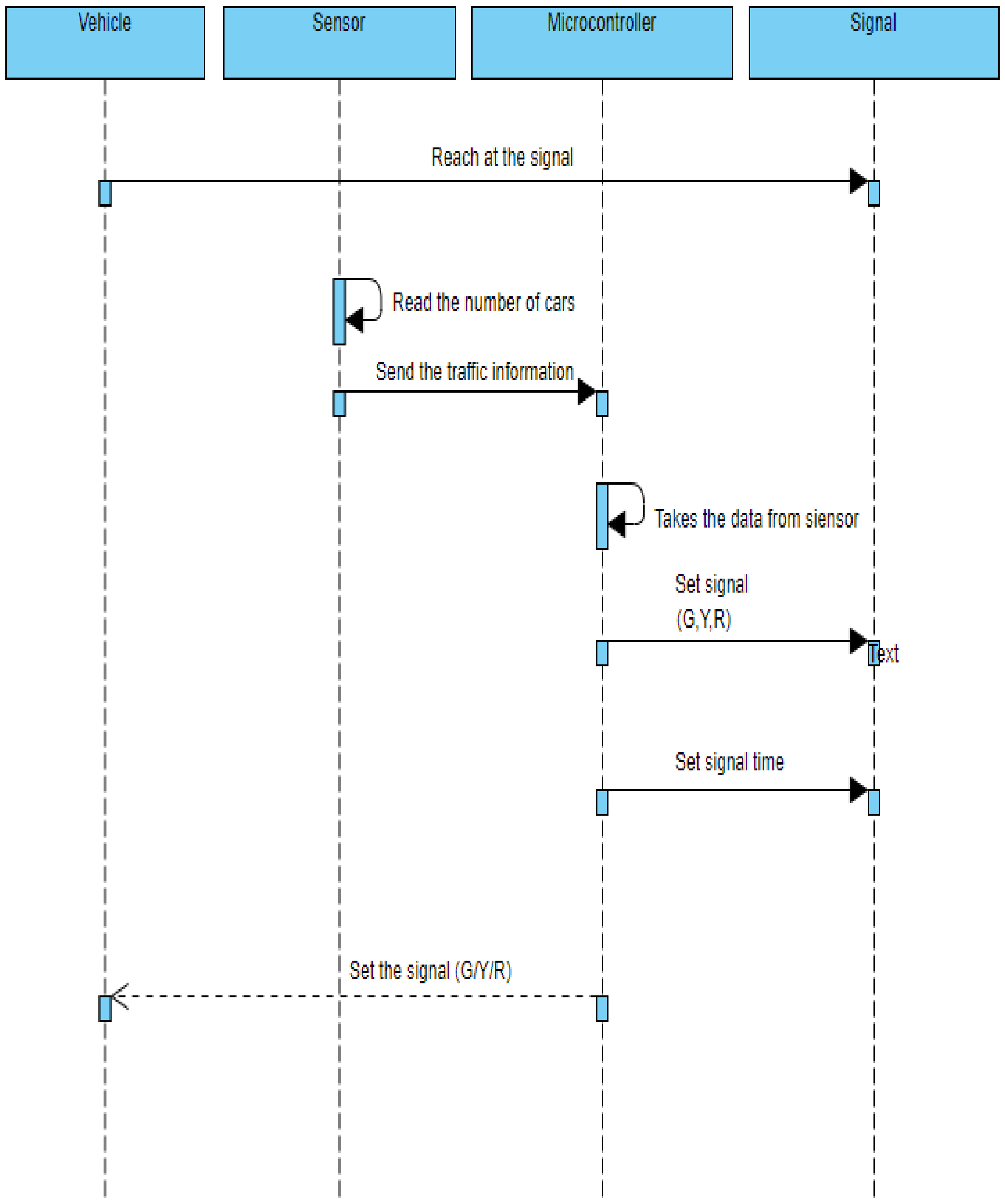
Signal at Lane 3

Diagrams


START
Read the number of vehicles
Vehicle Count ++
Compare Vehicle Count with Threshold
Is Vehicle Count < Threshold?
Yes
Normal Traffic
Give green signal to each lane in a sequential manner
No
Status = Congestion
Compare the number of vehicles in each lane
Open the lane with highest number of vehicles
Remove Current lane from comparison
If comparison contains more lanes
Yes
No

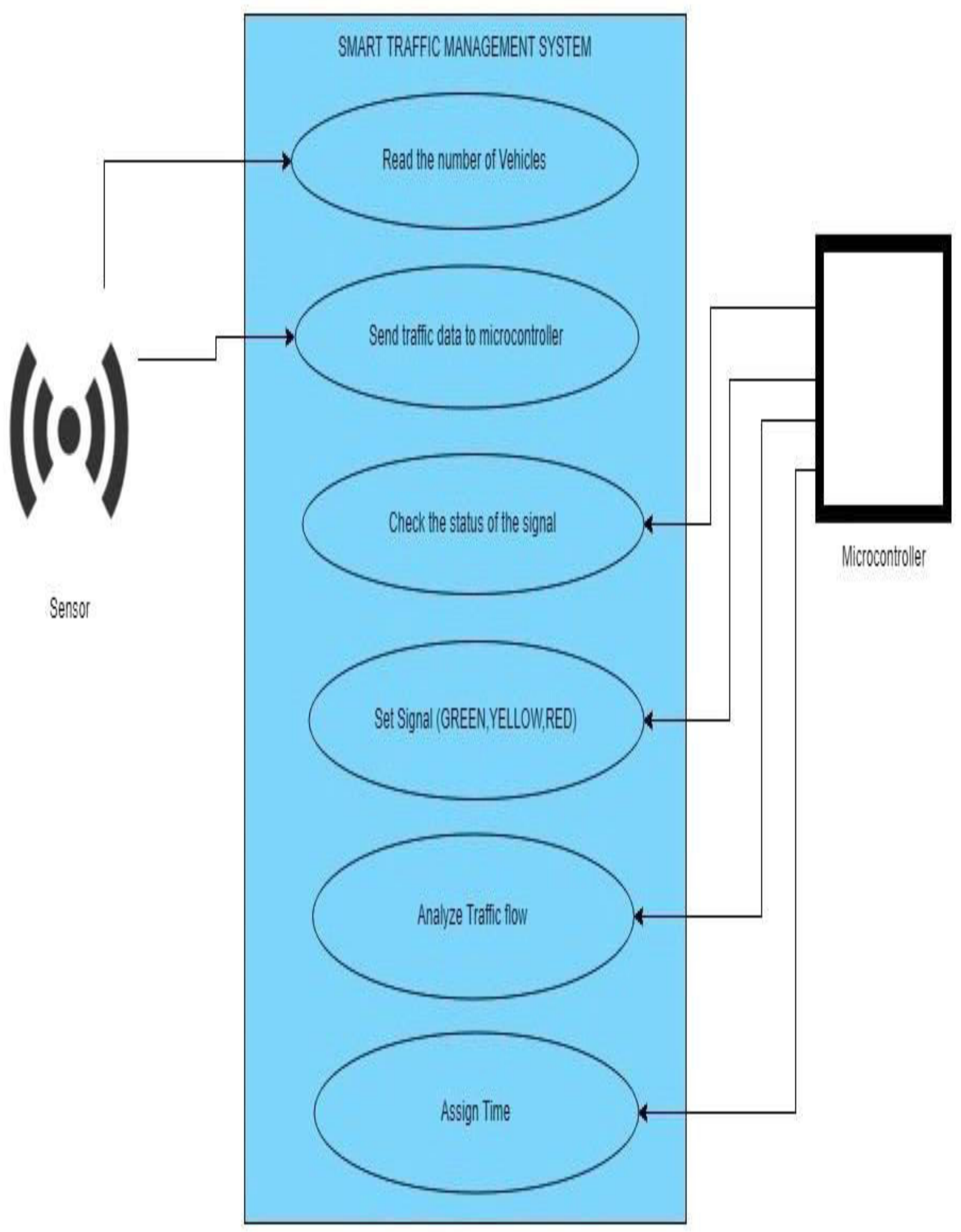
Flowchart.

Sequence Diagram



Sequence Diagram.

Use Case Diagram


SMART TRAFFIC MANAGEMENT SYSTEM
Read the number of Vehicles
Send traffic data to microcontroller
Check the status of the signal
Set Signal (GREEN,YELLOW,RED)
Analyze Traffic flow
Assign Time
Sensor
Microcontroller

Use Case Diagram.

Traffic Software Applications in ITMS

The next component is traffic software applications in ITMS. There are different traffic software applications, such as Waze, Google Maps, Navigator, TomTom GO, TomTom GO, HERE WeGo, MapQuest, INRIX, City mapper, Waze for Cities, Trans Nav,OptiMap, Trans Modeler, Vissim, Aimsun Next, PTV Visum, PTV Vistro, PTV Map&Guide, PTV xServer, TomTom Traffic, TomTom Maps, HERE HD Live Map, and so on, that employ the generated data in real time. These applications provide navigation, real-time traffic information, route optimization, and other features to the intelligent traffic management system (ITMS) to help drivers make informed decisions on the road. They are constantly updated to provide the latest information and new features to improve the driving experience. It is challenging to determine which traffic software application is "better" because it primarily relies on personal demands and preferences. Some of the previously mentioned traffic software applications, which will be covered in the next section, have received a lot of positive feedback for the precision of their data, the real-time traffic updates that they provide, and the user-friendly nature of their user interfaces.

ITMS Applications

This section covers a wide range of ITMS applications that all serve to highlight the effects of video-based network vehicle monitoring systems, including environmental impact assessment, safety monitoring, and TSCS.

Traffic Signal Control Systems (TSCSS)

People are leaving their hometowns in search of places that provide greater employment opportunities and a higher quality of life than what they can find in their current locations. As a direct consequence of the fast urbanization that is taking place, cities are seeing growth in the total amount as well as the variety of traffic. The problems caused by traffic are as follows:

- Increases the total amount of travel time;
- The use of fuel between intersection lines;
- Increased contributions to the air pollution caused by emissions;
- Unfortunate events;
- Managing an emergency is challenging.

Discussion

In the previous sections, we presented the techniques for monitoring vehicles in ITMS. Although there are still open questions and areas for improvement, future research will continue to advance the capabilities of video-based traffic surveillance systems. In this section, we discuss our outlook on the potential future developments of ITMS by discussing enhancing system efficiency, surveillance system on a network, how weather forecasts, incident reports, and online weather data are integrated into ITMS, comprehensive knowledge of traffic scenes, the role of vehicle spatial occupancy, and strategies needed for developing efficient ITMS. These highlight the need for continued research and development in ITS, to fully realize its potential for improving traffic management and safety

Program

LED

To control an LED connected to GPIO17, you can use this code:

```
from gpiozero import LED
from time import sleep

led = LED(17)

while True:
    led.on()
    sleep(1)
    led.off()
    sleep(1)
```

Run this in an IDE like Thonny, and the LED will blink on and off repeatedly.
LED methods include on(), off(), toggle(), and blink().

Button

To read the state of a button connected to GPIO2, you can use this code:

```
from gpiozero import Button
from time import sleep

button = Button(2)

while True:
    if button.is_pressed:
        print("Pressed")
    else:
        print("Released")
    sleep(1)
```

Button functionality includes the properties is_pressed and is_held; callbacks when_pressed, when_released, and when_held; and methods wait_for_press() and wait_for_release.

Button + LED
To connect the LED and button together, you can use this code:

```
from gpiozero import LED, Button

led = LED(17)
button = Button(2)

while True:
    if button.is_pressed:
        led.on()
    else:
        led.off()
```

Results and Analysis

The proposed system helps in better time based monitoring and thus has certain advantages over the existing system like minimizing number of accidents, reducing fuel cost and is remotely controllable etc.

The proposed system is designed in such a way that it will be able to control the traffic congestion as well as track the number of vehicles. The administrator of the system can access local server in order to maintain the system.