Traffic Management Project Documentation

Project Overview:

The Traffic Management project aims to use IoT devices and data analytics to monitor real-time traffic conditions, improve traffic flow, and assist commuters in making informed route decisions. The project incorporates the deployment of IoT sensors, the development of a mobile app, and the integration of a Raspberry Pi for data processing. The primary objectives are to alleviate traffic congestion, enhance road safety, and provide real-time traffic updates to commuters.

Project Objectives:

- ### Real-Time Traffic Monitoring:

The primary objective of the project is to implement a real-time traffic monitoring system using IoT sensors and data analytics. This system continuously collects data from various traffic sensors to assess current traffic conditions.

- ### Traffic Flow Improvement:

The project aims to improve traffic flow in urban areas by analyzing traffic density and adjusting traffic signals accordingly. By reducing congestion, it enhances the overall traffic experience for commuters.

- ### Commuter Assistance:

The project seeks to assist commuters by providing them with real-time traffic updates and route recommendations through a mobile app.

- ### Ambulance Detection:

Another critical objective is the detection of ambulance sirens. By using sound sensors at intersections, the system identifies approaching ambulances and gives them priority in traffic signal control.

- ❖ **Road Safety Enhancement:**

Enhancing road safety is a key goal. The project's ability to detect ambulance sirens and clear their path contributes to quicker emergency responses and improved safety for all road users.

- ❖ **Traffic Management Algorithms:**

The project aims to implement intelligent traffic management algorithms on a Raspberry Pi. These algorithms process data from IoT sensors and make dynamic decisions to optimize traffic signal control.

- ❖ **Reduced Congestion:**

By achieving efficient traffic signal control, the project aims to reduce traffic congestion in urban areas, leading to shorter commute times and a better driving experience.

- ❖ **Improved Traffic Flow:**

The project intends to improve traffic flow by dynamically adapting traffic signal timing based on real-time data. This adaptation minimizes traffic jams and optimizes the flow of vehicles.

- ❖ **Sustainable Urban Mobility:**

The project aligns with the broader objective of promoting sustainable urban mobility. By reducing congestion and improving traffic flow, it contributes to the efficient use of urban road networks.

These objectives collectively serve the purpose of creating a sophisticated traffic management system that benefits commuters, improves road safety, and makes urban mobility more efficient. The project's success is measured by its ability to achieve these objectives and positively impact the urban transportation landscape.

IoT Sensor Setup:

1. Ultrasonic Sensors:

- ❖ Ultrasonic sensors are strategically deployed at key locations along roadways, such as intersections and traffic hotspots.
- ❖ These sensors utilize ultrasonic waves to measure the distance between the sensor and vehicles on the road.
- ❖ The time it takes for the ultrasonic waves to bounce back allows the system to calculate traffic density.
- ❖ Low traffic density corresponds to a longer distance, while high traffic density results in a shorter distance.

2. Sound Sensors:

- ❖ Sound sensors are positioned at intersections and traffic signal points.
- ❖ These sensors are designed to detect specific sounds, particularly ambulance sirens.
- ❖ When a sound sensor identifies an ambulance siren, it sends a signal to the system to prioritize traffic signal control, allowing the ambulance to pass through the intersection quickly.

3. Raspberry Pi Integration:

- ❖ The Raspberry Pi serves as the central processing unit for the IoT sensor data.

- ❖ It collects data from both the ultrasonic and sound sensors, processes it, and makes traffic management decisions based on the data.

4. LCD Display (optional):

- ❖ An optional component is an LCD display that provides real-time traffic information to commuters and pedestrians.

- ❖ This display is often placed at intersections to convey current traffic conditions, signal status, and alerts.

5. Traffic Lights and Pedestrian Signals:

- The system controls traffic lights and pedestrian signals.
- It uses sensor data to determine when to switch traffic lights from green to red, adjust signal timings, and prioritize ambulances.

6. Mobile App Interface:

- While not a physical sensor, the mobile app interface plays a crucial role in the sensor setup.
- It acts as a user interface to access real-time traffic information, route recommendations, and ambulance alerts.
- Commuters can receive data from sensors and make informed route decisions via the mobile app.

The IoT sensor setup is a critical component of the Traffic Management project, as it provides real-time data to the system for traffic density measurement and ambulance detection. These sensors work in tandem with the Raspberry Pi and mobile app to achieve the project's objectives of traffic flow improvement and commuter assistance.

Mobile App Development:

- **Real-time Traffic Updates**: The mobile app provides users with real-time traffic updates, indicating traffic congestion, road closures, and accident alerts.
- **Route Recommendations**: The app suggests optimal routes based on traffic conditions, ensuring users can choose the least congested paths.
- **Ambulance Detection**: The app uses sound sensor data to identify approaching ambulances and provides priority traffic signal control to clear the ambulance's path.
- **User-Friendly Interface:**The mobile app features a user-friendly interface with intuitive navigation, interactive maps, and customization options for route preferences.

Raspberry Pi Integration:

1. Data Collection and Processing:

- ❖ The Raspberry Pi serves as the central hub for data collection and processing.
- ❖ It receives data from IoT sensors, including ultrasonic and sound sensors.
- ❖ This data includes information about traffic density, vehicle distances, and ambulance sirens.

2. Real-Time Traffic Analysis:

- ❖ The Raspberry Pi is responsible for performing real-time traffic analysis.
- ❖ It continuously processes incoming sensor data to assess the current traffic conditions at various intersections and road segments.
- ❖ It calculates traffic density and determines whether traffic is low or high.

3. Traffic Signal Control:

- ❖ One of the core functions of the Raspberry Pi is traffic signal control.
- ❖ Based on the real-time traffic analysis, the Raspberry Pi dynamically adjusts traffic signal timings.
- ❖ For example, during periods of high traffic density, it extends the green light duration to optimize traffic flow.
- ❖ Conversely, in low traffic situations, it reduces the green light duration to save energy and prevent unnecessary stops.

4. Ambulance Detection:

- ❖ The Raspberry Pi is programmed to detect ambulance sirens using sound sensors.

- ❖ When an ambulance siren is detected, the Raspberry Pi takes immediate action to prioritize the ambulance's passage. It switches traffic signals to green and clears the ambulance's path.

5. Mobile App Integration:

- ❖ The Raspberry Pi communicates with the mobile app to share real-time traffic updates and recommendations.

- ❖ It sends data to the mobile app, allowing users to access current traffic conditions, receive route suggestions, and receive alerts about ambulance movements.

6. LCD Display Control (optional):

- ❖ In cases where LCD displays are used to provide real-time traffic information at intersections, the Raspberry Pi controls the content displayed on these screens.

- ❖ It updates the LCD displays with relevant traffic information, including signal status and alerts.

7. Connectivity to IoT Sensors:

- ❖ The Raspberry Pi is connected to IoT sensors, both ultrasonic and sound sensors, via GPIO (General Purpose Input/Output) pins.
- ❖ It receives data from these sensors and interprets it for further processing.

8. Communication with Traffic Lights:

- ❖ The Raspberry Pi communicates with traffic lights and pedestrian signals to implement its traffic management decisions.
- ❖ It sends signals to control the lights, changing them from red to green and vice versa as needed.

9. Dynamic Traffic Algorithms:

- ❖ The Raspberry Pi hosts dynamic traffic management algorithms that use incoming sensor data to make real-time decisions regarding traffic signal timings.
- ❖ These algorithms play a crucial role in achieving traffic flow improvement.

10. Mobile App Interface:

- ❖ The Raspberry Pi interacts with the mobile app interface, enabling users to access real-time traffic information.
- ❖ It sends traffic updates, route recommendations, and ambulance alerts to the mobile app for display.

The Raspberry Pi is at the core of the Traffic Management system, where it processes data from IoT sensors, makes intelligent traffic management decisions, and interfaces with both traffic control devices and the mobile app. Its integration plays a vital role in achieving the project's objectives of traffic flow improvement and commuter assistance

Code Implementation:

- ❖ Python scripts are employed to control and read data from ultrasonic and sound sensors.

```
from gpiozero import LED, DistanceSensor, InputDevice
import time
from RPLCD.i2c import CharLCD # Library for I2C LCD display
# Traffic light control
red_light = LED(17)
yellow_light = LED(27)
green_light = LED(22)
# Pedestrian crossing signals
pedestrian_red = LED(5)
pedestrian_green = LED(6)
# Ultrasonic sensor for traffic density
traffic_sensor = DistanceSensor(echo=4, trigger=18, max_distance=2)
# Sound sensor for ambulance detection
ambulance_siren = InputDevice(23)
# LCD display setup
lcd = CharLCD('PCF8574', 0x27)
lcd.clear()
# Traffic state variables
current_state = "low_traffic"
red_duration = 15 # Duration for red light in seconds
green_duration = 20 # Duration for green light in seconds
yellow_duration = 3 # Duration for yellow light in seconds
# Pedestrian crossing intervals
pedestrian_wait_duration = 3 # Time for pedestrians to wait
pedestrian_cross_duration = 5 # Time for pedestrians to cross
while True:
# Traffic density detection
if traffic_sensor.distance < 0.5: # Adjust this threshold
if current_state != "high_traffic":
```

```
current_state = "high_traffic"
green_light.on()
yellow_light.off()
red_light.off()
pedestrian_red.on()
pedestrian_green.off()
lcd.clear()
lcd.write_string("High Traffic")
time.sleep(green_duration)
else:
if current_state != "low_traffic":
current_state = "low_traffic"
red_light.on()
yellow_light.off()
green_light.off()
pedestrian_red.off()
pedestrian_green.on()lcd.clear()
lcd.write_string("Low Traffic")
time.sleep(red_duration)
# Ambulance detection
if ambulance_siren.is_active:
yellow_light.on()
pedestrian_red.on()
lcd.clear()
lcd.write_string("Ambulance Alert")
time.sleep(yellow_duration) # Keep yellow light on for a short duration
pedestrian_red.off()
time.sleep(yellow_duration) # Allow pedestrians to cross
# Traffic light status on LCD
lcd.cursor_pos = (1, 0)
lcd.write_string(f"Red: {red_light.is_active}")
lcd.cursor_pos = (1, 8)
lcd.write_string(f"Yellow: {yellow_light.is_active}")
lcd.cursor_pos = (2, 0)
lcd.write_string(f"Green: {green_light.is_active}")
# Pedestrian crossing logic
if pedestrian_green.is_active:
time.sleep(pedestrian_cross_duration)
pedestrian_green.off()
pedestrian_red.on()
else:
time.sleep(pedestrian_wait_duration)
pedestrian_red.off()
pedestrian_green.on()
red_light.on()
green_light.off()
time.sleep(1) # Delay between light changes
red_light.off()
yellow_light.on()
time.sleep(1)
yellow_light.off()
```

The mobile app is developed using platform-specific technologies such as Android Studio for Android and Xcode for iOS.The app communicates with the Raspberry Pi for real-time traffic updates and route recommendations.

Html code:

```
<!DOCTYPE html>
<html>
<head>
<title>Traffic Information</title>
<link rel="stylesheet" type="text/css" href="styles.css" />
</head>
<body>
<header>
<h1>Real-Time Traffic Information</h1>
</header>
<div class="traffic-info">
<div class="traffic-light-status">
<h2>Traffic Light Status</h2>
<p id="red-light-status">Red Light: Off</p>
<p id="yellow-light-status">Yellow Light: Off</p>
<p id="green-light-status">Green Light: Off</p>
</div>
<div class="pedestrian-crossing">
<h2>Pedestrian Crossing</h2>
<p id="pedestrian-status">Pedestrian: Waiting</p>
</div>
<div class="ambulance-alert">
<h2>Ambulance Alert</h2>
<p id="ambulance-status">Ambulance: No Alert</p>
</div>
</div>
<script src="script.js"></script>
</body>
</html>
```

Css code:

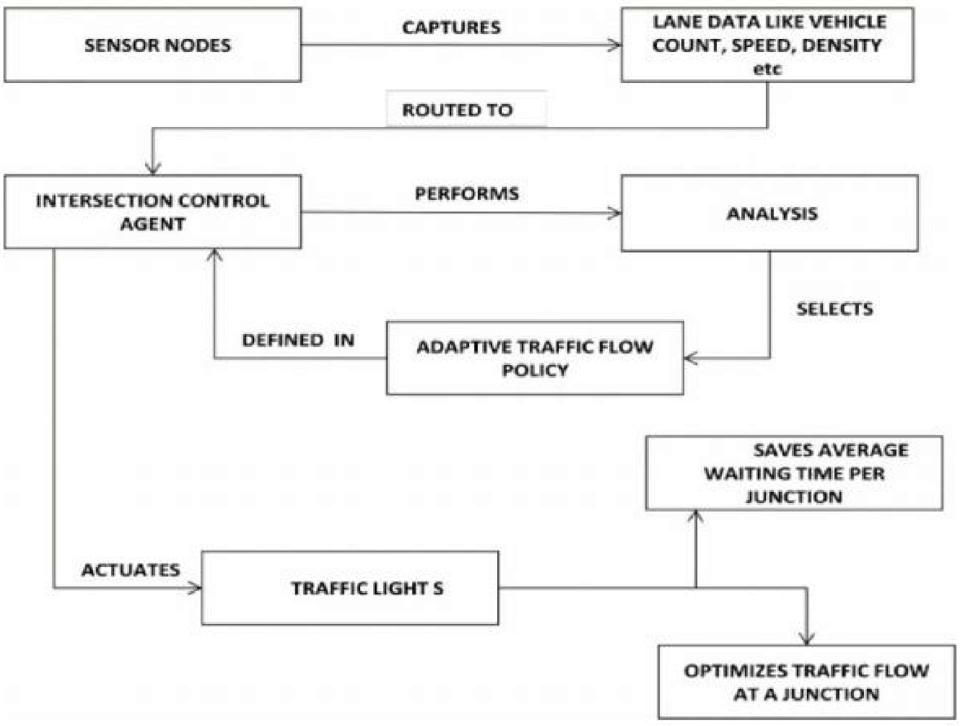
```
body {
font-family: Arial, sans-serif;
background-color: #f3f3f3;
margin: 0;
padding: 0;
}
header {
background-color: #333;
color: #fff;
text-align: center;
padding: 10px;
}
.traffic-info {
max-width: 800px;
margin:auto;
background-color: #fff;
border: 1px solid #ccc;
padding: 40px;
border-radius: 5px;
display: flex;
flex-direction: column;
margin-top: 40px;
}
.traffic-light-status, .pedestrian-crossing{
margin-bottom: 35px;
padding: 0 20px 20px 20px;
background-color: #f9f9f9;
border: 1px solid #ddd;
border-radius: 5px;
}
.ambulance-alert{
padding: 0 20px 20px 20px;
background-color: #f9f9f9;
border: 1px solid #ddd;
border-radius: 5px;
}
#red-light-status, #yellow-light-status, #green-light-status,
#pedestrian-status, #ambulance-status {
margin: 0;
}
#red-light-status.on, #yellow-light-status.on, #green-light-status.on,
#pedestrian-status.on, #ambulance-status.on {
color: green;
}
#red-light-status.off,  #yellow-light-status.off,  #green-light    status.off,  #pedestrian-
status.off, #ambulance-status.off {
color: red;
}
```

Js code:

```
function updateTrafficStatus() {
const redLightStatus = document.getElementById("red-light-status");
const yellowLightStatus = document.getElementById("yellow-light    status");
const greenLightStatus = document.getElementById("green-light    status");
const pedestrianStatus = document.getElementById("pedestrian    status");
const ambulanceStatus = document.getElementById("ambulance-status");
const isRedOn = true; // Replace with actual data
const isYellowOn = false; // Replace with actual data
const isGreenOn = true; // Replace with actual data
const isPedestrianWaiting = true; // Replace with actual data
const isAmbulanceAlert = false; // Replace with actual data
// Update traffic light status
redLightStatus.textContent = `Red Light: ${isRedOn ? "On" : "Off"}`;
yellowLightStatus.textContent = `Yellow Light: ${isYellowOn ? "On" :
"Off"}`;
greenLightStatus.textContent = `Green Light: ${isGreenOn ? "On" :
"Off"}`;
// Update pedestrian crossing status
pedestrianStatus.textContent = `Pedestrian: ${isPedestrianWaiting ?
"Waiting" : "Crossing"}`;
// Update ambulance alert status
ambulanceStatus.textContent = `Ambulance: ${isAmbulanceAlert ?
"Alert" : "No Alert"}`;
// Add or remove 'on' class to highlight active status
isRedOn ?
(redLightStatus.classList.add("on"),redLightStatus.classList.remove("of
f")) :
(redLightStatus.classList.remove("on"),redLightStatus.classList.add("of
f"));
isYellowOn ?
(yellowLightStatus.classList.add("on"),yellowLightStatus.classList.remo
ve("off")) :
(yellowLightStatus.classList.remove("on"),yellowLightStatus.classList.a
dd("off"));
isGreenOn ?
(greenLightStatus.classList.add("on"),greenLightStatus.classList.remove
("off")) :
(greenLightStatus.classList.remove("on"),greenLightStatus.classList.add
("off"));
isPedestrianWaiting ?
(pedestrianStatus.classList.add("on"),pedestrianStatus.classList.remove
("off")) :
(pedestrianStatus.classList.remove("on"),pedestrianStatus.classList.add
("off"));
isAmbulanceAlert ?
(ambulanceStatus.classList.add("on"),ambulanceStatus.classList.remove("
off")) :
(ambulanceStatus.classList.remove("on"),ambulanceStatus.classList.add("
off"));
}
// Initial update
updateTrafficStatus();
// Simulated real-time updates
setInterval(() => {
// Fetch and update data from Python IoT code
updateTrafficStatus();
}, 5000); // Update every 5 seconds (adjust as needed)
```

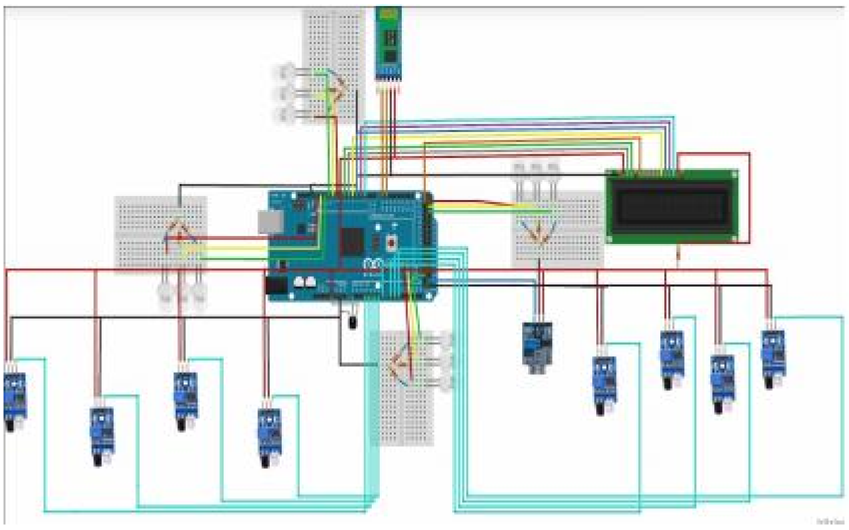
System Architecture:

Traffic Management System Architecture

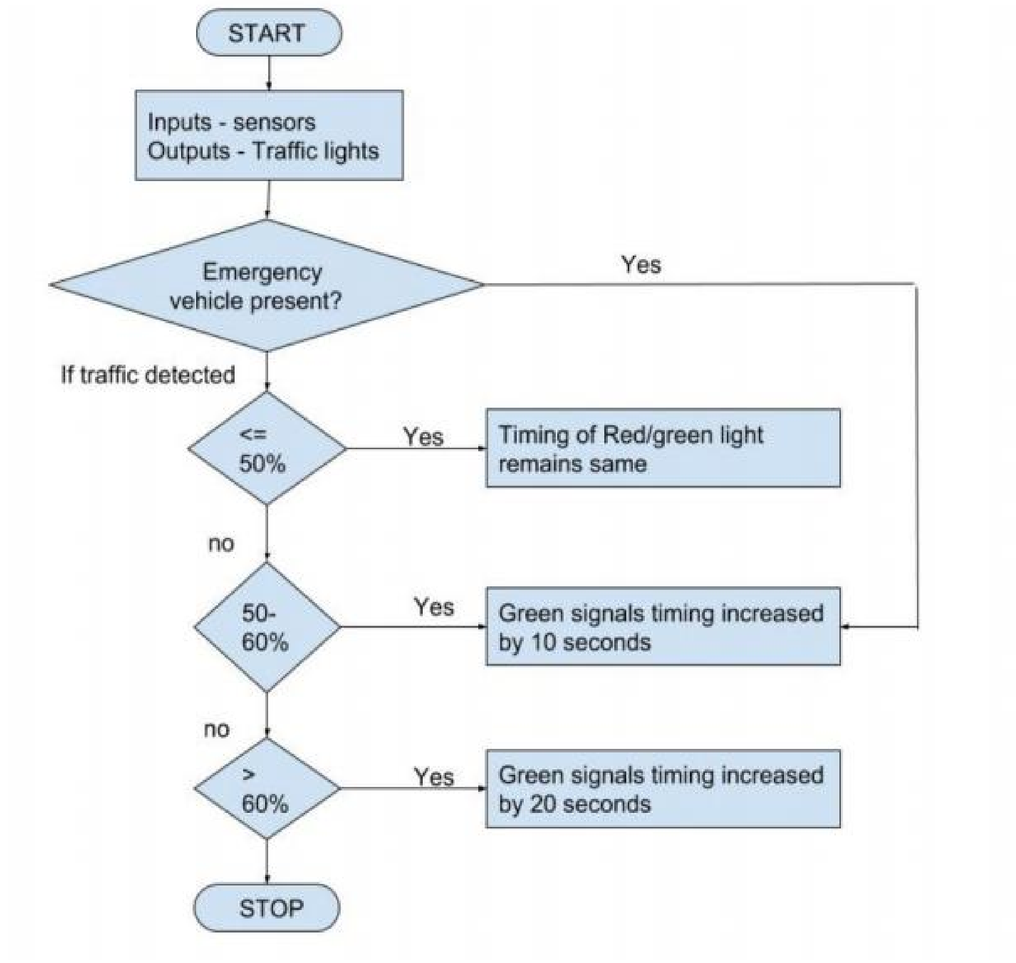


SENSOR NODES
CAPTURES
LANE DATA LIKE VEHICLE COUNT, SPEED, DENSITY etc
ROUTED TO
INTERSECTION CONTROL AGENT
PERFORMS
ANALYSIS
SELECTS
DEFINED IN
ADAPTIVE TRAFFIC FLOW POLICY
SAVES AVERAGE WAITING TIME PER JUNCTION
ACTUATES
TRAFFIC LIGHT S
OPTIMIZES TRAFFIC FLOW AT A JUNCTION

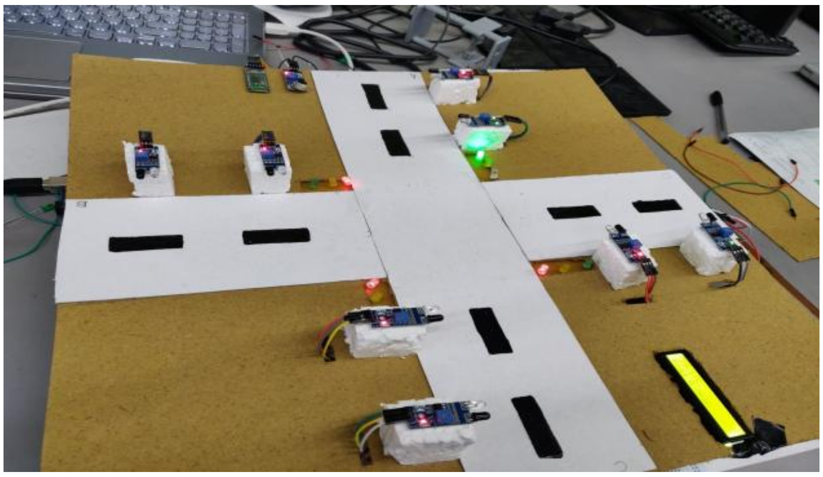
Circuit Diagram

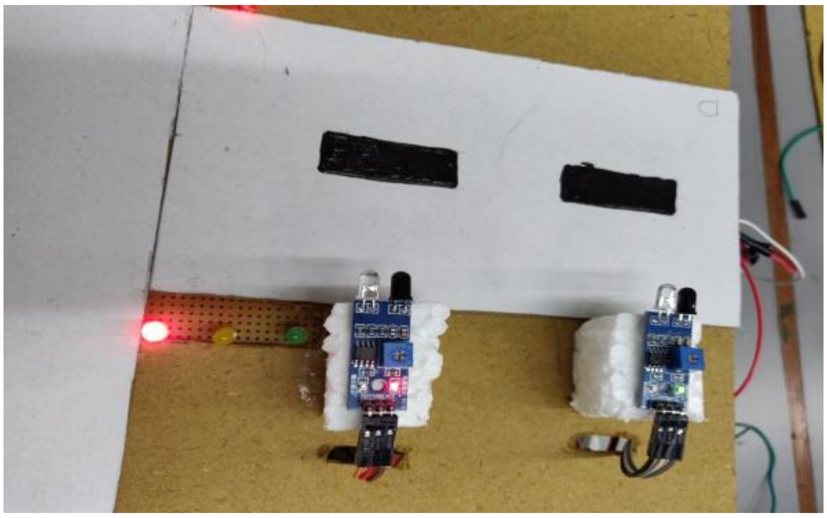


Flow Chart


START
Inputs - sensors
Outputs - Traffic lights
Emergency vehicle present?
Yes
If traffic detected
<=
50%
Yes
Timing of Red/green light remains same
no
50-60%
Yes
Green signals timing increased by 10 seconds
no
>
60%
Yes
Green signals timing increased by 20 seconds
STOP

Photos of the circuit





Real-Time Traffic Monitoring Benefits:

The real-time traffic monitoring system offers several advantages:

- **Reduced Congestion:** By providing real-time updates and intelligent traffic signal control, traffic congestion is reduced, leading to quicker commutes.

- Enhanced Safety: The system's ability to detect ambulance sirens and clear their path increases road safety.

- **Informed Route Decisions:** Commuters can make informed route decisions based on real-time traffic data, avoiding congested areas.

- **Improved Traffic Flow**: Traffic signals adapt to current conditions, improving overall traffic flow and reducing gridlock.

- The Traffic Management project is designed to create a more efficient and safer road network for commuters, ultimately contributing to better urban mobility and reduced travel times.

Conclusion:

This project documentation provides an overview of the Traffic Management project, including its objectives, sensor setup, mobile app development, Raspberry Pi integration, and code implementation. It also highlights the benefits of the real-time traffic monitoring system for commuters and traffic management.