**IOT PHASE 5**

**TRAFFIC MANAGEMENT SYSTEM**

****

PROJECT TITLE

Describing a traffic management system and IOT sensor setup also raspberry pi integration and code implementation explain how real time traffic monitoring system can assist commuters in making optimal route decision and improving traffic flow.

**INTRODUCTION:**

Traffic Management Systems (TMS) use a variety of technologies to manage traffic flows and the effects of congestion on the roading network. Traffic Management Systems do this by addressing the traffic management effects of accidents and slow moving or queuing vehicles, planned events and extreme weather.

**OBJECTIVE:**

The primary objective of a traffic management system is to enhance the efficiency and safety of traffic flow on roads. This involves managing and controlling various aspects of transportation, including vehicle movement, congestion, and road capacity. The specific objectives can vary, but generally, a traffic management system aims to achieve the following:

1. Congestion Reduction: One of the key objectives is to minimize traffic congestion, which leads to delays, increased travel time, and frustration among commuters. By implementing effective traffic management strategies, such as optimizing signal timings, providing real-time information to drivers, and managing traffic flow through dynamic lane control, the system strives to alleviate congestion and improve traffic flow.

2. Enhancing Safety: Another crucial objective is to enhance road safety by reducing accidents and improving overall transportation security. A traffic management system incorporates features like surveillance cameras, speed monitoring, and accident detection mechanisms to identify hazardous situations and alert authorities in real-time. This enables prompt response and improves emergency management on the roads.

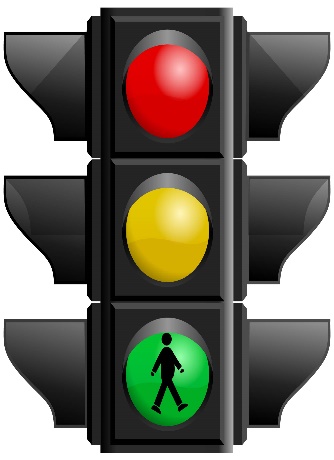
3. Efficient Resource Utilization: A traffic management system aims to optimize the utilization of transportation resources, such as road infrastructure, traffic signals, and public transportation services. By effectively managing these resources, the system can reduce fuel consumption, lower greenhouse gas emissions, and maximize the capacity of existing infrastructure.

4. Effective Incident Management: The system focuses on efficiently handling incidents such as accidents, breakdowns, and road closures. By promptly identifying and responding to such incidents, traffic management authorities can minimize disruptions, reroute traffic, and ensure efficient incident clearance.

5. Improving Commuter Experience: The objective is to provide a better experience for commuters by offering real-time traffic updates, route recommendations, and alternative transportation options. This allows commuters to make informed decisions, leading to reduced travel times, enhanced reliability, and improved overall satisfaction with the transportation system.

6. Data-Driven Decision Making: A traffic management system collects and analysis vast amounts of data from various sources, such as sensors, cameras, and GPS devices. The objective is to utilize this data to generate insights, measure performance, identify patterns, and inform decision-making processes. This data-driven approach helps optimize traffic management strategies and allows for continuous improvements.

By achieving these objectives, a traffic management system aims to create a more efficient, safer, and reliable transportation network that benefits both commuters and the larger community.

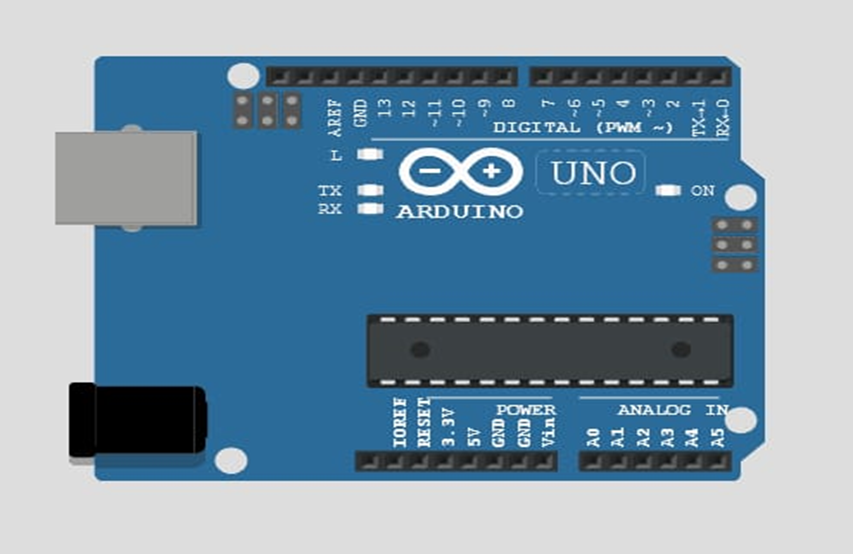


**REQUIRED COMPONENTS:**

* Arduino Uno R3 or raspberry pi
* 3 LEDS (Green, Red, Yellow)
* 1 Small Breadboard
* 1 Green Wire
* 1 Yellow Wire
* 1 Red wire
* 3 Black Wires
* 220 ohm resistors ( 3 required)
* Sensors (if required)

**ARDUINO UNO R3:**

The LEDS has been powered by Arduino UNO (board). It contains which uploaded to the board and once it simulated LED’S start blinking like a traffic light.



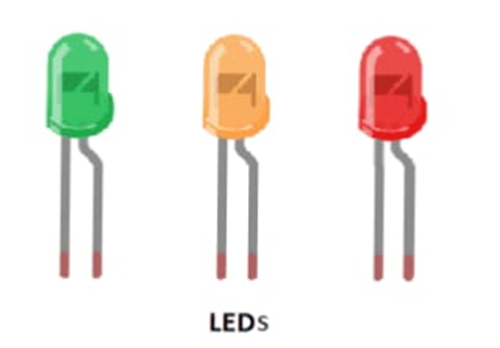
**3 LED’S (GREEN RED YELLOW):**

• Traffic signals consists of three colours: Red Yellow Green

RED: A Driver should stop his vehicle.

YELLOW: A Driver has to slow down and be ready to stop

GREEN: A Driver can start driving or keep driving

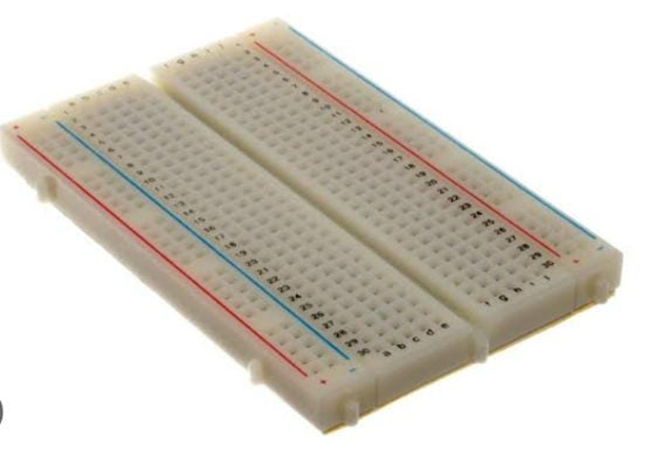


**BREADBOARD:**

• A Solderless breadboard is a matrix of electrical connection points in a plastic enclosure. They come in various sizes: The common sizes are:

• Full,60 rows,800 points including positive/negative bus strip.

• Half 30 rows, 400 points including positive/negative bus strip.



**CONNECTING WIRES:**

A connecting wire allows travels the electric current from one point to another point without resistivity. Resistance of connecting wire should always be near zero. Copper wires have low resistance and are therefore suitable for low resistance we need black yellow red and green wire.



**RESISTORS:**

A 220-ohm resistor is an electronic component that is used to resist the flow of electricity in a circuit. Resistors are used in a wide variety of electronic circuits to control the flow of current and protect other components from damage. 220-ohm resistors are a commonly used resistance value in electronic circuits.



**RASPBERRY Pi (if we use)**

The Raspberry Pi is a low cost, credit-card sized computer that plugs into a computer monitor or TV, and uses a standard keyboard and mouse. It is a capable little device that enables people of all ages to explore computing, and to learn how to program in languages like Scratch and Python. The Raspberry Pi is a debit card-sized low-cost computer that connects to a computer Desktop or TV and uses a standard mouse and Keyboard.

It has a dedicated processor, memory, and a graphics driver, just like a PC. It also comes with its operating system, Raspberry Pi OS, a modified version of Linux.



**ALGORITHM:**

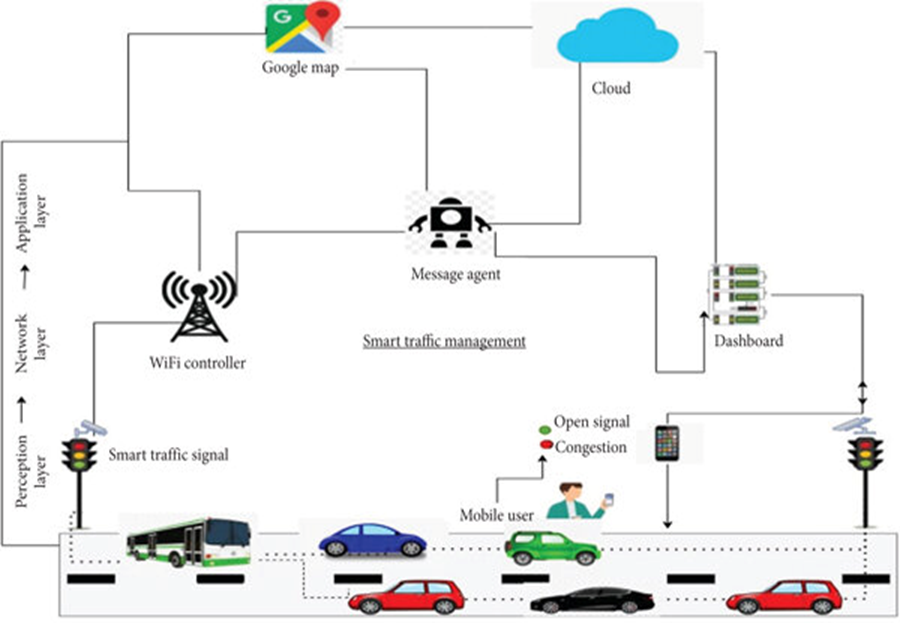
1. Fetch traffic data from an API

2. Display the traffic updates on the app interface

3. Allow users to input their starting and destination locations

4. Process the input and obtain route recommendations using a routing algorithm (e.g., Dijkstra's algorithm)

5. Display the recommended routes on the app interface.



**DEVELPOMENT OF APP:**

Developing an app for traffic management and route recommendations can be very useful for commuters. Here are a few steps to get started:

1. Define the scope of your app: Determine what specific features you want to include in your app. Some possible features could be real-time traffic updates, route recommendations based on current traffic conditions, alternative route suggestions, and personalized preferences for users such as avoiding toll roads or highways.

2. Conduct market research: Look into existing traffic management and navigation apps to understand their features and identify any gaps that your app can fill. This will help you understand your target audience and their needs.

3. Create wireframes and design: Sketch out the user interface and design the app's features. Consider making the app user-friendly, with clear navigation and intuitive controls.

4. Develop the backend infrastructure: Build the necessary server infrastructure to handle real-time traffic data and store user preferences. This will require programming skills and knowledge of databases.

5. Implement real-time traffic updates: Integrate a reliable traffic data provider into your app to provide accurate and up-to-date information to users. Options include services like Google Maps API, HERE Traffic API, or TomTom Traffic API.

6. Develop route recommendation algorithms: Use algorithms to analyse real-time traffic data and suggest the most efficient routes based on user preferences. This may involve considering factors such as current traffic conditions, historical data, and user feedback.

7. Test and optimize: Thoroughly test your app to ensure it performs well under various scenarios and conditions. Gather feedback from beta testers and make necessary optimizations based on their input.

8. Launch and promote: Once your app is ready, launch it on relevant platforms such as the App Store and Google Play. Implement marketing strategies to promote your app and reach your target audience.

Remember, developing an app is a complex process that requires time and expertise. Consider seeking professional help or collaborating with experienced developers if needed.



**WE USE MIT APP DEVELOPER FOR BUILDING OUR APP:**

**STEP I**

**Building your first app: Traffic management system:**

* Now that you've set up your computer and device, and you've learned how the Designer and the Blocks Editor work, you are ready to build the traffic management app.
* At this point, you should have the Designer or Blocks Editor open in your browser, and either an Android/iPhone device or an emulator connected to your project. (See Setup Instructions if you do not have these things setup.)

**TRAFFIC MANAGEMENT SYSTEM APP**

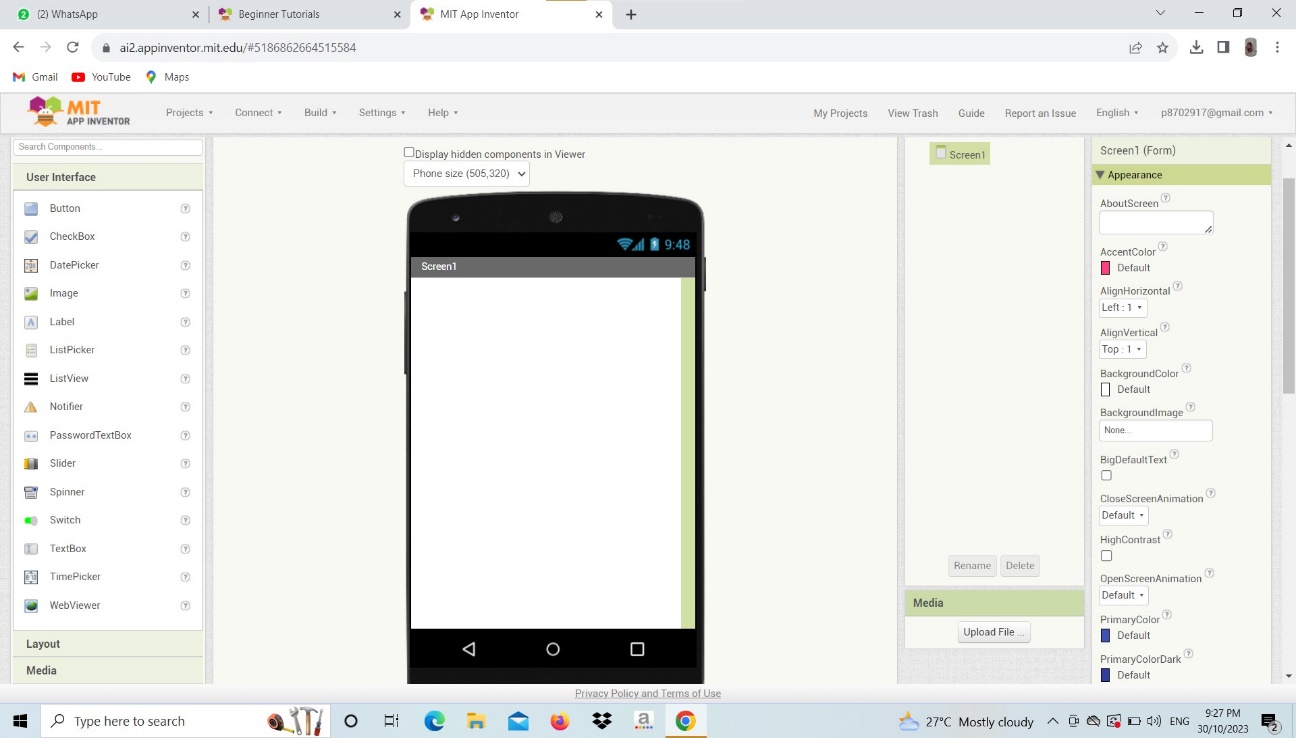
It is a simple app which would consist of some of the navigations to move on from our current location. It could also show various locations around you and which you could easily reach the destination by having the app we have built let us look out the step by step procedure on developing this.

**STEP II**

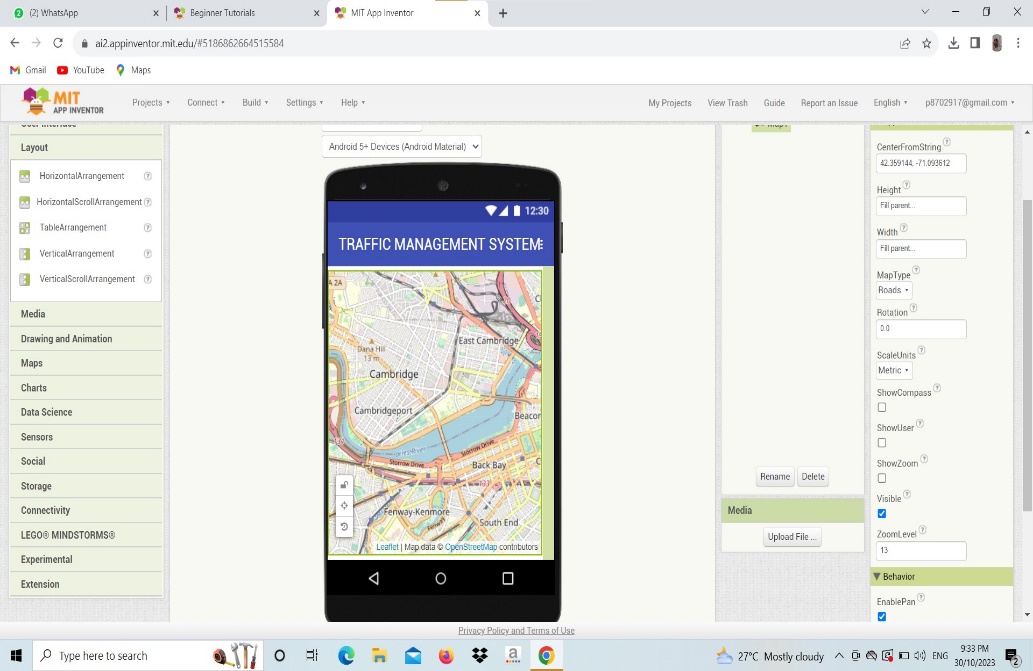
**Steps for selecting components and setting properties**

Step 1a : From the User Interface palette, drag and drop the Button component to Screen1

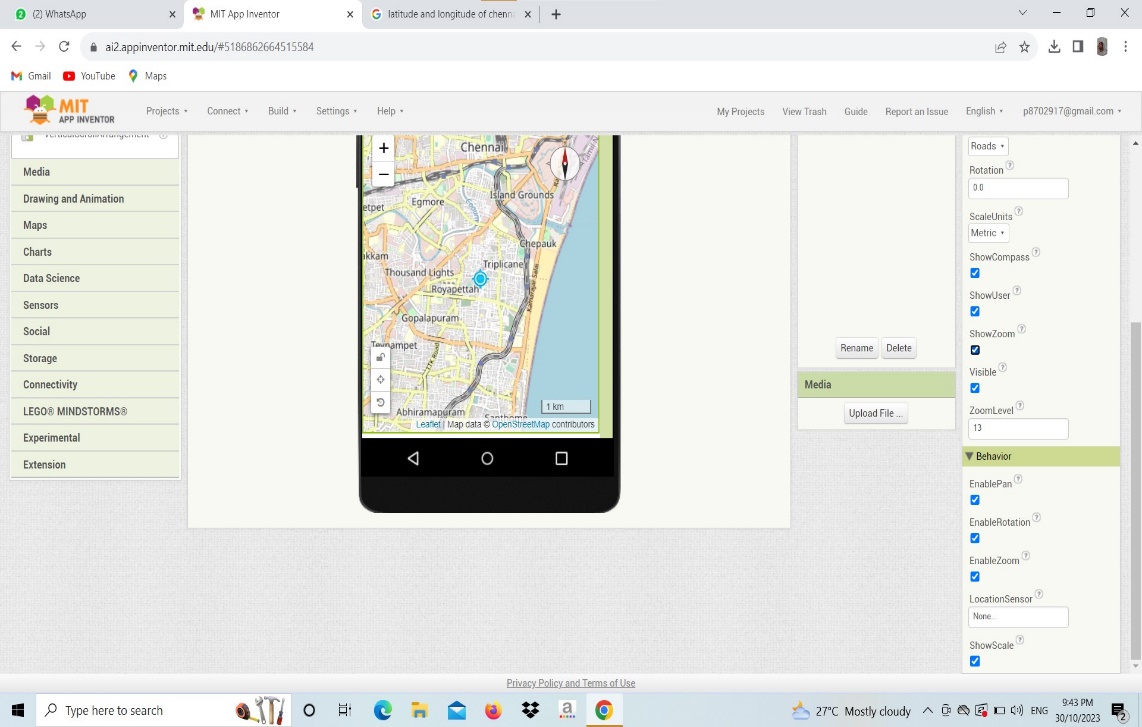
Step 1b : To make the button have an image of some map or some traffic light, in the Properties pane, under Image, click on the text "None..." and click "Upload New" . A window will pop up to let you choose the image file. Click "Choose File" and then navigate to the location of the traffic.png file you downloaded earlier . Click the traffic.png file, click "Open", and then click "OK".

****

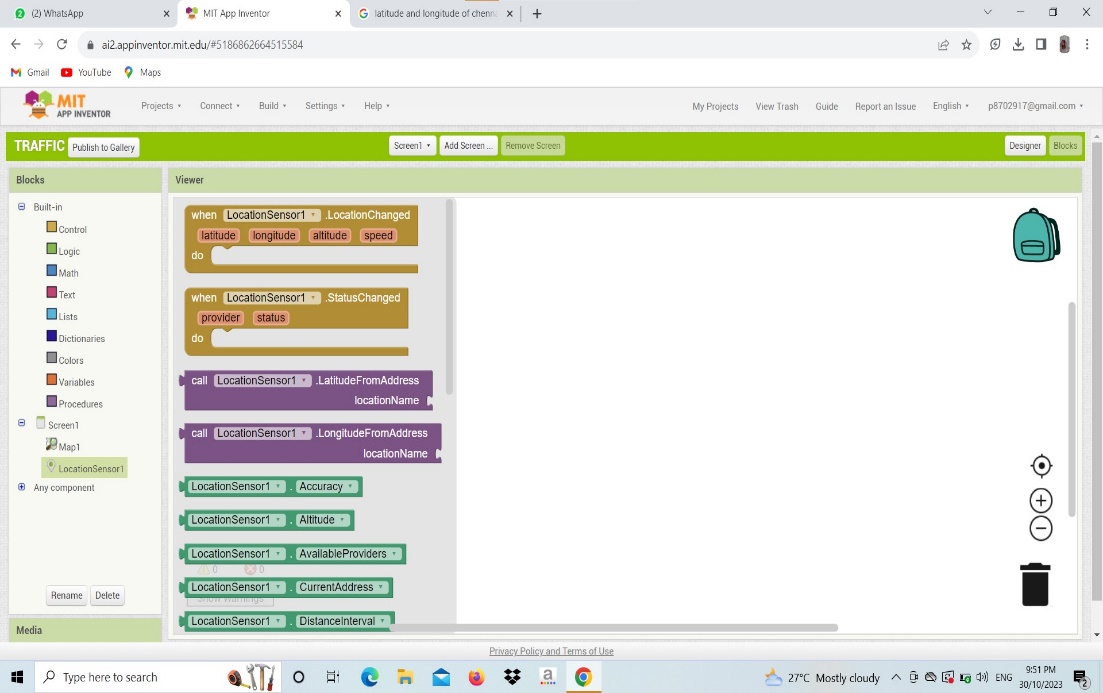
Step2: In the next step, give a title to your app as TRAFFIC MANAGEMENT SYSTEM and change the other properties of it as per the requirements. And also bring a map and do the changes as needed.

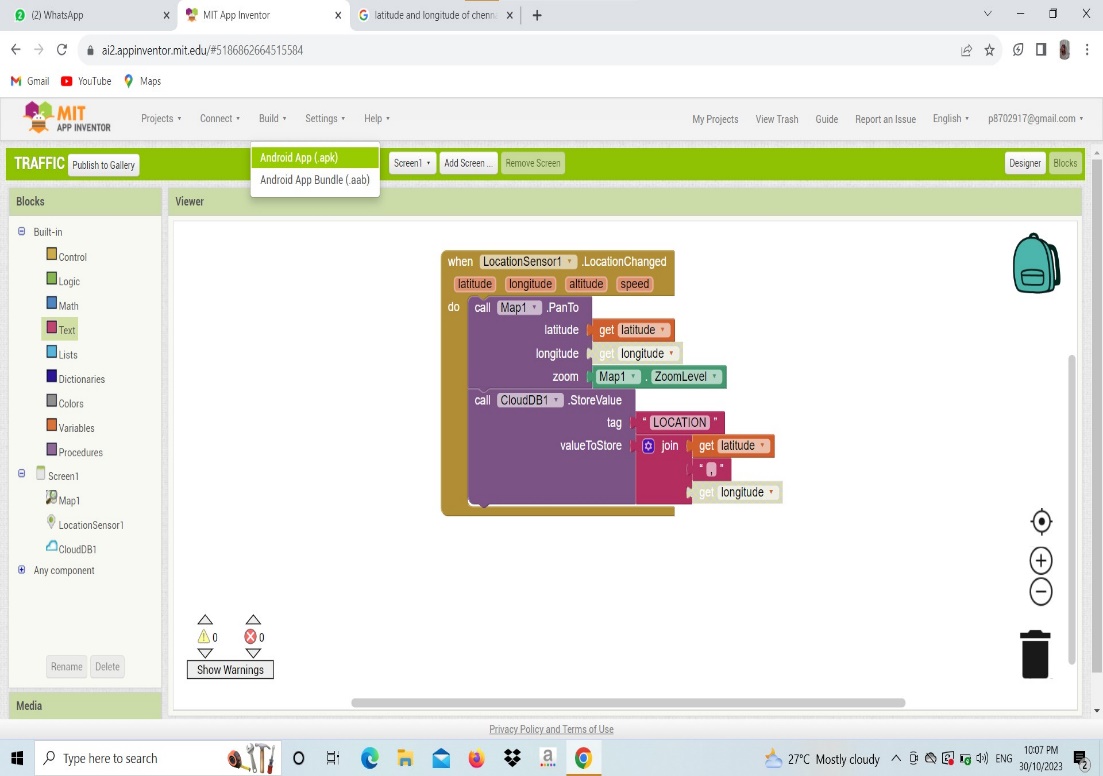


Step 3: Then add location sensor and change the properties of those location sensor and initiate this location sensor in the map properties.

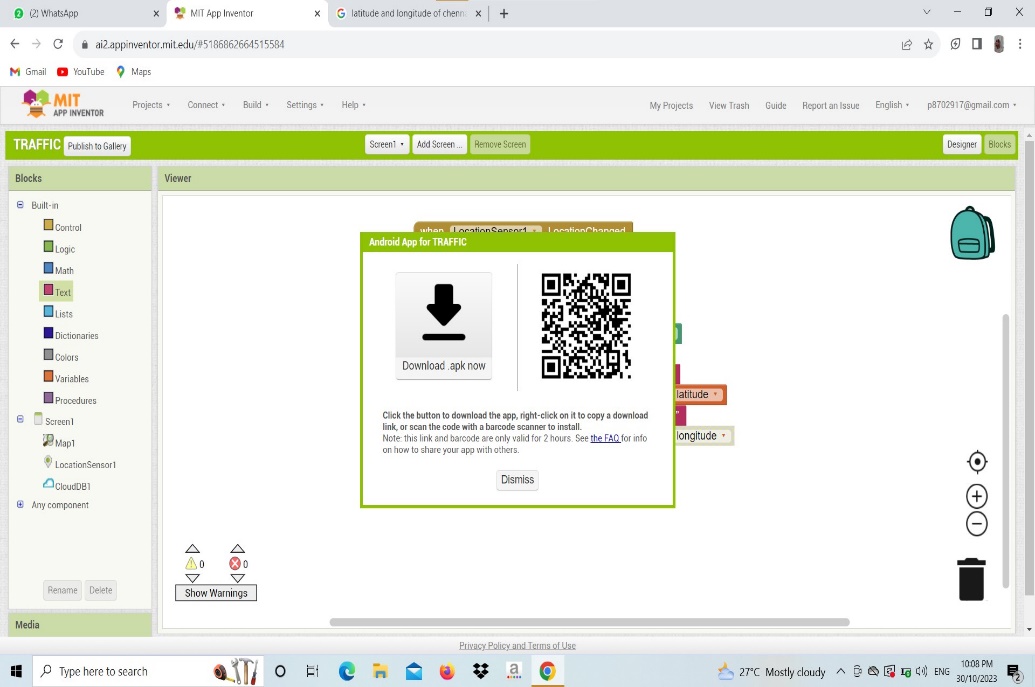


Step 4: Go to the blocks code and drag and drop some of the blocks into and out of this TMS app. As we know first we are going to drag from map a “get latitude” and also “get longitude” and an empty block is chosen and it dragged and dropped this process get repeated until we get all the necessities.





Step 5: After completing all these just scan using MIT inventor II or else download an .apk file and install in your mobile application.



**PROGRAM:(Using HTML, CSS, and JavaScript):**

Here's a program using HTML, CSS, and JavaScript to create a simple app for traffic updates and route recommendations:

HTML CODE:

A program using html is used to create a display App just by running a html Program . here is a example of the app we create for a traffic updates and routes.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Traffic App</title>

<link rel="stylesheet" href="styles.css">

</head>

<body>

<div id="app">

<h1>Real-Time Traffic Updates</h1>

<p id="trafficData">Fetching traffic data...</p>

<label for="startLocation">Start Location:</label>

<input type="text" id="startLocation" placeholder="Enter start location">

<label for="endLocation">End Location:</label>

<input type="text" id="endLocation" placeholder="Enter end location">

<button id="routeBtn">Get Route Recommendations</button>

<div id="routeRecommendations"></div>

</div>

<iframe src="https://www.google.com/maps/embed?pb=!1m28!1m12!1m3!1d1997699.2574773899!2d77.29474156810653!3d12.060277246200265!2m3!1f0!2f0!3f0!3m2!1i1024!2i768!4f13.1!4m13!3e6!4m5!1s0x3a5265ea4f7d3361%3A0x6e61a70b6863d433!2schennai!3m2!1d13.082680199999999!2d80.27071839999999!4m5!1s0x3ba859af2f971cb5%3A0x2fc1c81e183ed282!2scoimbatore!3m2!1d11.0168445!2d76.9558321!5e0!3m2!1sen!2sin!4v1698151026534!5m2!1sen!2sin" width="600" height="450" style="border:0;" allowfullscreen="" loading="lazy" referrerpolicy="no-referrer-when-downgrade"></iframe>

<script src="app.js"></script>

</body>

</html>

CSS CODE:

Next we have to create a .css program using a name style.css by giving a font colour and margins to the display page we create here we have a css program which will be linked with the above html program to run and get a display app.

body {

font-family: Arial, sans-serif;

margin: 0;

padding: 0;

}

#app {

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

height: 100vh;

}

h1 {

margin-bottom: 20px;

}

button {

padding: 10px 20px;

border-radius: 5px;

background-color: #007bff;

color: #fff;

border: none;

cursor: pointer;}

JAVA SCRIPT CODE:

Java script plays an important role in fetching the data from the recovered resource. Java script consists of fetching a data using start and end location. We can save this as app.js this also will be linked with the html program to proceed output.

document.addEventListener('DOMContentLoaded', () => {

const trafficDataElement = document.getElementById('trafficData');

const startLocationInput = document.getElementById('startLocation');

const endLocationInput = document.getElementById('endLocation');

const routeRecommendationsElement = document.getElementById('routeRecommendations');

const routeBtn = document.getElementById('routeBtn');

// Fetch traffic data from API

fetch('https://www.google.com/maps/embed?pb’)

.then(response => response.json())

.then(data => {

trafficDataElement.innerText = `Current Traffic: ${data.status}`;

})

.catch(error => {

trafficDataElement.innerText = 'Failed to fetch traffic data.';

});

routeBtn.addEventListener('click', () => {

const startLocation = startLocationInput.value;

const endLocation = endLocationInput.value;

if (startLocation && endLocation) {

// Logic to get route recommendations using a routing algorithm

const routeRecommendations = getRouteRecommendations(startLocation, endLocation);

// Display route recommendations on the app interface

routeRecommendationsElement.innerHTML = '';

for (const recommendation of routeRecommendations) {

const recommendationItem = document.createElement('p');

recommendationItem.innerText = recommendation;

routeRecommendationsElement.appendChild(recommendationItem);

}

} else {

Alert ('Please enter both start and end locations.');

}

});

// Function to get route recommendations using a routing algorithm

function getRouteRecommendations(startLocation, endLocation) {

// Implement your routing algorithm logic here

// Example:

const route1 = `Route 1: ${startLocation} - Waypoint 1 - Waypoint 2 - ${endLocation}`;

const route2 = `Route 2: ${startLocation} - Waypoint 3 - Waypoint 4 - ${endLocation}`;

return [route1, route2];

}

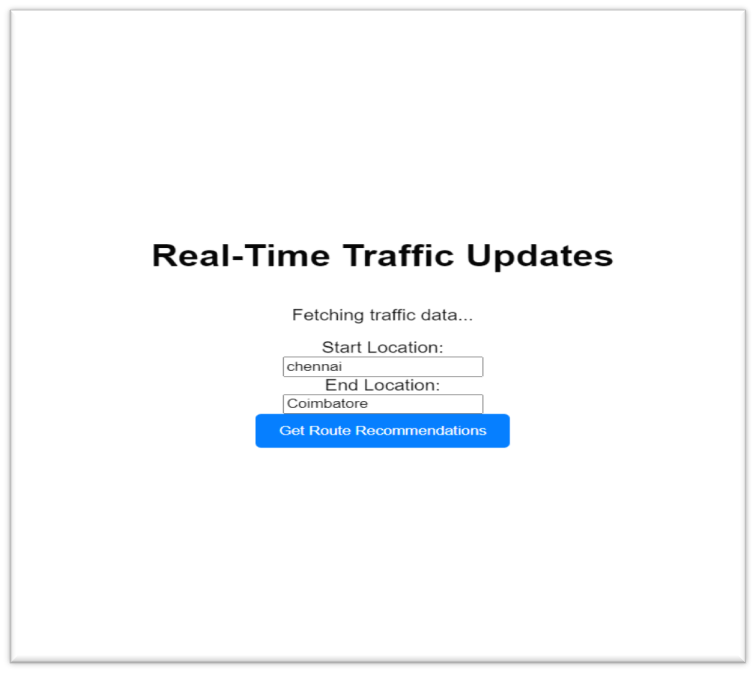
});

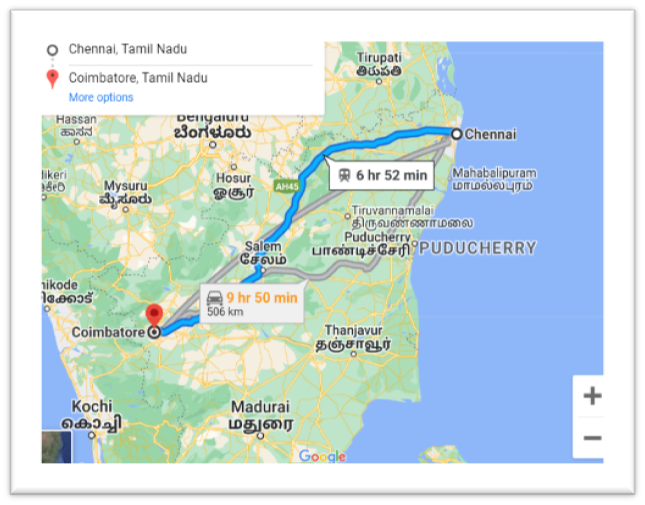
Here we can see a sample output of this traffic updates and route recommendation The data are fetched and shown using html code and after executing this run and get the required output.

**OUTPUT:**

Here is an output for the above html , css , javascript program we have given above,







PLATFORM USED FOR SIMULATING OUR SOURCE CODE:

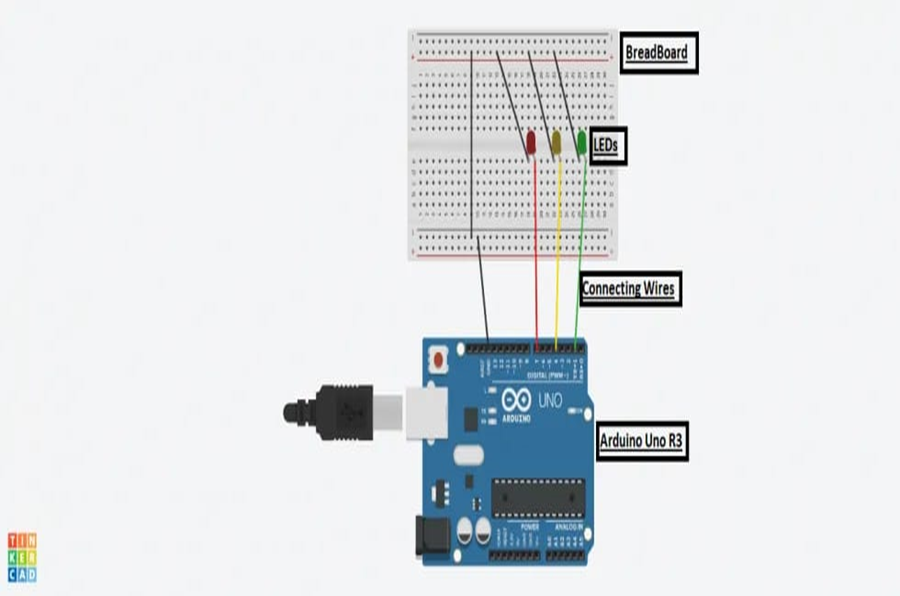
* We also can use this wowki platform to simulate our IOT codes and also we can use TinkerCad instead of wowki WowKi is a platform that offers a variety of components for building Internet of Things (IoT) projects.
* To create a traffic light control management system using Wowki, you will need the following components:

1. Wowki IoT Development Board: This board serves as the main controller for your project. It is equipped with a microcontroller and various input/output interfaces to connect and control other components.

2. LED Modules: These modules represent the traffic lights. You will need three LED modules, each representing the colors red, green, and yellow.

3. Jumper wires: These wires are used to establish connections between the Wowki development board and the LED modules.

**CIRCUIT DIAGRAM:**



STEPS TO CREATE TRAFFIC MANAGEMENT SYSTEM IN ARDUINO IN WOWKI:

Here's an overview of how to connect the components:

1. Connect the red LED module to one GPIO pin on the Wowki development board. This will be the pin used to control the red light.

2. Connect the green LED module to another GPIO pin on the Wowki development board. This will be the pin used to control the green light.

3. Connect the yellow LED module to a third GPIO pin on the Wowki development board. This will be the pin used to control the yellow light.

4. Ensure that each LED module is properly connected to a power supply, such as the 3.3V or 5V output pins on the Wowki development board.

5.Change the colors of the LED wires necessary to show the differentiation on different colored wires.

6. Then give black color to the wires that has been placed on the ground (GND).

7.Do any changes if you want to change the color or size or any other thing if necessary.

8. Write the necessary code to control the traffic lights based on the desired logic. This can be done using the Wowki IoT coding platform or by using Python, Arduino, or other compatible programming languages.

SOURCE CODE:

Void setup ()

{

//put your source code here to run once:

Serial.begin (115200);

if(!Serial) Serial.println (“serial is not ok”);

pinMode (3, OUTPUT);

pinMode (4, OUTPUT);

pinMode (5, OUTPUT);

}

Void loop ()

{

//put your main code here to run repeatedly:

digitalWrite (3, HIGH);

Serial.println (“green is on”);

delay (2000);

digitalWrite (3, LOW);

Serial.println(“green is off”);

digitalWrite (4, HIGH);

Serial.println (“yellow is on”);

delay (600);

digitalWrite (4, LOW);

Serial.println(“yellow is off”);

digitalWrite (5, HIGH);

Serial.println (“red is on”);

delay (1800);

digitalWrite (4, HIGH);

Serial.println(“yellow is on”);

delay(400);

digitalWrite (5, LOW);

Serial.println(“red is off”);

digitalWrite(4, LOW);

Serial.println(“yellow is off”);

}

SIMULATE AND TEST:

1. Click the "Start Simulation" button to run your simulation.

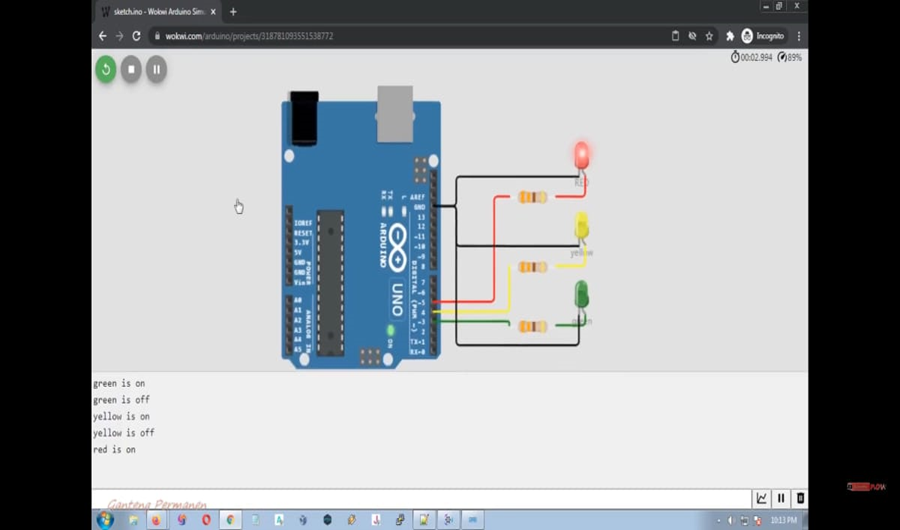
2. Press the push-button switches to change the traffic light sequence. Each button corresponds to a different traffic light state.

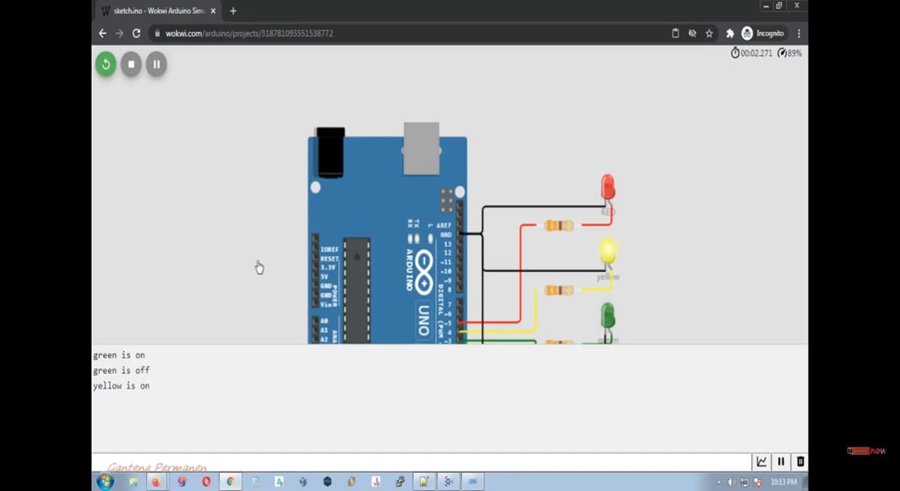
3. Observe how the LEDs change their states based on the button presses.

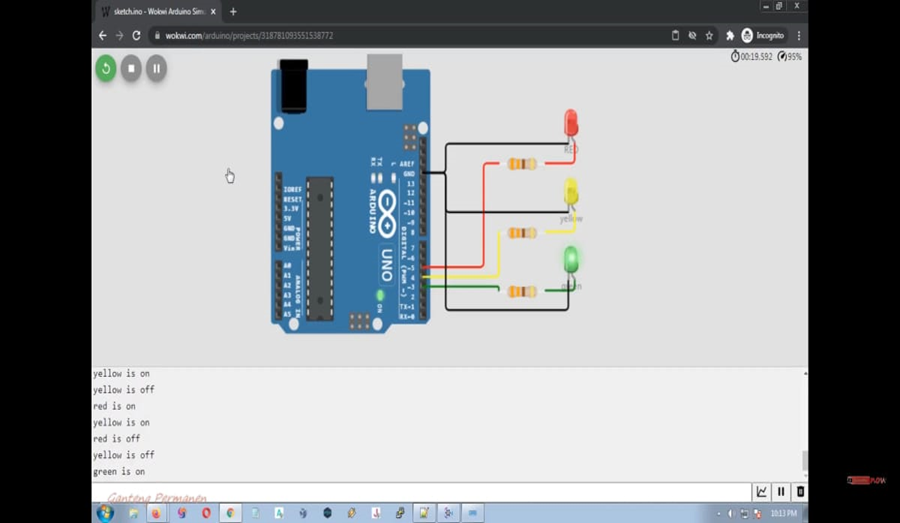
You should see the traffic light sequence (Red -> Yellow -> Green) in response to the button presses.

* That's it! You've successfully created a traffic light controller in wowki.
* This project helps you understand the basics of input and output control using Arduino and simulates a simple traffic light system.

OUTPUT:







IF WE USE RASPBERRY Pi WE CAN USE THE BELOW CODE AND GET THE OUTPUT:

Certainly! Here's an example code for traffic control management using a Raspberry Pi and Python:

python

import RPi.GPIO as GPIO

import time

# GPIO pin numbers

red\_pin = 17

yellow\_pin = 27

green\_pin = 22

button\_pin = 16

# Set up GPIO

GPIO.setmode(GPIO.BCM)

GPIO.setup(red\_pin, GPIO.OUT)

GPIO.setup(yellow\_pin, GPIO.OUT)

GPIO.setup(green\_pin, GPIO.OUT)

GPIO.setup(button\_pin, GPIO.IN, pull\_up\_down=GPIO.PUD\_UP)

def change\_lights(state):

# Turn off all lights

GPIO.output(red\_pin, GPIO.LOW)

GPIO.output(yellow\_pin, GPIO.LOW)

GPIO.output(green\_pin, GPIO.LOW)

# Set lights based on the state

if state == "red":

GPIO.output(red\_pin, GPIO.HIGH)

print("Red Light")

elif state == "yellow":

GPIO.output(yellow\_pin, GPIO.HIGH)

print("Yellow Light")

elif state == "green":

GPIO.output(green\_pin, GPIO.HIGH)

print("Green Light")

# Initialize traffic lights

change\_lights("red")

while True:

try:

# Wait for button press

GPIO.wait\_for\_edge(button\_pin, GPIO.FALLING)

# Change the lights

change\_lights("green")

time.sleep(5)

change\_lights("yellow")

time.sleep(2)

change\_lights("red")

time.sleep(5)

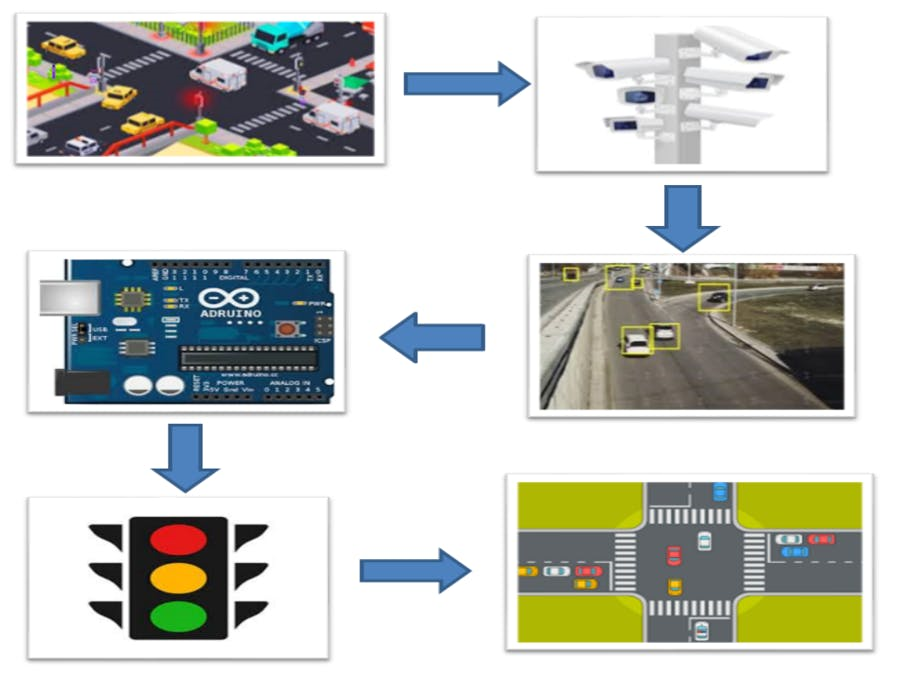
except KeyboardInterrupt:

# Clean up GPIO on Ctrl+C exit

GPIO.cleanup()

* In this example code, we use the `RPi.GPIO` library to control the GPIO pins on the Raspberry Pi.
* Three GPIO pins are used to connect LEDs representing red, yellow, and green traffic lights (with appropriate resistors).
* Another GPIO pin is used as the input for a button that triggers the change in traffic light states.
* The `change\_lights` function turns off all lights and sets the lights based on the given state.
* In the main loop, we wait for a button press (falling edge detection) using the `wait\_for\_edge` function from RPi.GPIO.
* Once the button is pressed, the traffic lights cycle from green to yellow to red with appropriate delays using the `time.sleep` function.
* You can further customize this code to include additional features like pedestrian crossing signals, countdown timers, or integration with sensors or cameras for more advanced traffic control systems.

**BLOCK DIAGRAM:**



**FEATURES:**

1. Real-time traffic updates: The app fetches traffic data from an API and displays it to users.

2. Route recommendations: Users can enter their start and end locations, and the app suggests the best routes based on current traffic conditions.

3. User-friendly interface: The app provides a user-friendly interface for easy interaction and input of location details.

**ADVANTAGES:**

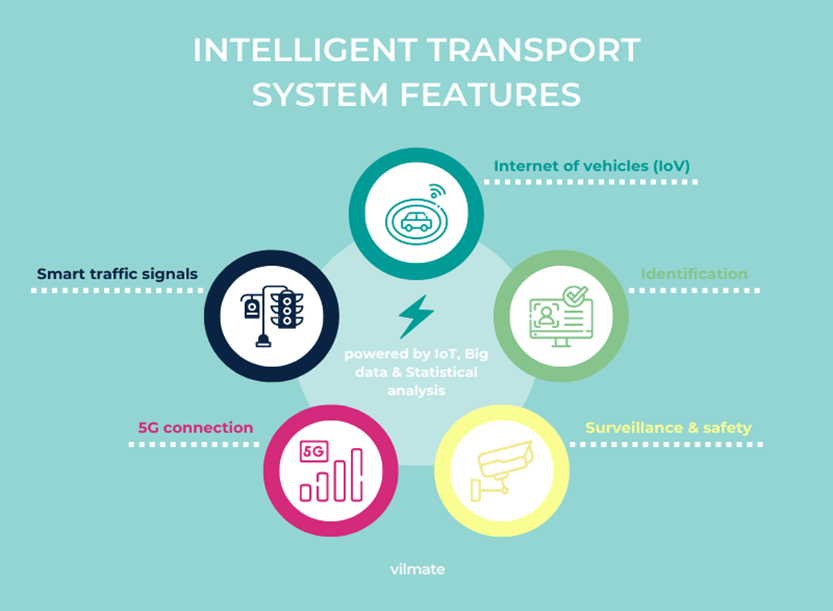
1. Time-saving: Users can avoid traffic congestion by following the recommended routes, saving time during their commute.

2. Reduce stress: Real-time traffic updates help users plan their journeys better, reducing stress

**DISADVANTAGES:**

1. Dependence on traffic data sources: The accuracy and reliability of the app heavily rely on the quality of the traffic data providers.

2. Connectivity requirements: The app requires a stable internet connection to retrieve real-time traffic data efficiently.



**CONCLUSION:**

By building a traffic control management system using dataset loading and preprocessing techniques, we can make informed decisions to optimize traffic flow and enhance overall road safety. This project highlights the importance of utilizing datasets and leveraging advanced technologies to improve traffic management effectiveness.

Through this presentation I want to conclude that the traffic management system is necessary and basic need for today ‘s busy peoples.