**NAAN MUDHALVAN**

**INTERNET OF THINGS - IBM**

**TRAFFIC MANAGEMENT USING IOT**

**PHASE 5**

**TEAM MEMBERS**

**HARISH B (2021116317)**

**LALIT DHAKSHAN M (2021116315)**

**JAI MANI RS (2021116011)**

**SUMESH VARMA R (20211160304)**

**SOMASUNDARAM S (2021116041)**

**V SEMESTER 3rd** YEAR

**B.E BIOMEDICAL ENGINEERING**

**COLLEGE OF ENGINEERING GUINDY**

**ANNA UNIVERSITY, CHENNAI**

**TRAFFIC MANAGEMENT USING IOT**

**PHASE 1**

**Problem Statement**

Traffic congestion is a growing problem in urban areas, leading to increased travel times, fuel consumption, and air pollution. As cities continue to grow, effective traffic management becomes crucial for maintaining the quality of life for residents and the efficiency of transportation systems. Traditional traffic management methods are often inadequate to address the dynamic nature of traffic in modern cities.

To address this challenge, our project aims to develop a Traffic Management System using the Internet of Things (IoT) technology. The objective is to create a smart and adaptive system that can optimize traffic flow, reduce congestion, and enhance overall transportation efficiency in urban environments.

**Project Objectives:** Our primary project objectives are to enhance urban traffic management by achieving the following:

1. **Real-Time Traffic Monitoring:** Develop a system that continuously collects and analyzes traffic data in real-time to gain insights into traffic patterns.
2. **Congestion Detection:** Implement algorithms to identify congestion-prone areas and provide early warnings to both commuters and traffic authorities.
3. **Route Optimization:** Create a dynamic route recommendation system that suggests the most efficient routes to commuters based on real-time traffic conditions.
4. **Improved Commuting Experience:** Enhance the overall commuting experience by providing commuters with up-to-date traffic information and alternative routes to minimize delays.

**IoT Sensor Design:** To achieve our objectives, we will strategically deploy IoT sensors throughout the urban area:

1. **Traffic Flow Sensors:** Install sensors at key intersections and road segments to monitor vehicle flow, speed, and density.
2. **Smart Traffic Lights:** Implement smart traffic lights equipped with IoT capabilities to optimize traffic signal timings based on real-time data.
3. **Vehicle Data Integration:** Collaborate with car manufacturers to integrate IoT sensors into vehicles for data sharing, enhancing traffic data accuracy.

**Real-Time Transit Information Platform:** We envision a comprehensive platform accessible via web and mobile apps to provide real-time traffic information to the public:

1. **User-Centric Design:** Develop an intuitive user interface with user-centric features, such as customizable route planning and alerts.
2. **Real-Time Data Visualization:** Display real-time traffic data in an easily understandable format, including traffic congestion maps and estimated travel times.
3. **Commuter Alerts:** Send push notifications to users regarding traffic incidents, road closures, or alternative routes.
4. **Integration with Public Transit:** Incorporate data from public transit systems, allowing commuters to make informed decisions that combine various modes of transportation.

**Integration Approach:**

Our integration approach will ensure seamless operation and data flow within the system:

Data Aggregation: Aggregate data from IoT sensors, traffic cameras, and vehicle data sources into a centralized cloud-based platform.

Data Analysis: Utilize algorithms to analyze traffic data, predict congestion, and optimize traffic signal timings.

API Integration: Develop APIs for integration with third-party navigation apps and public transportation services.

**TRAFFIC MANAGEMENT USING IOT**

**PHASE 2**

INNOVATION PROPOSAL

1) Data Collection and Sensors:

Install a network of IoT sensors and cameras throughout the city to collect real-time data on traffic conditions. These sensors can include vehicle detection sensors, environmental sensors (to measure air quality), and cameras for visual data.

2) Data Processing and Analytics:

Transmit data from sensors to a centralized system or cloud platform for real-time processing. Use advanced analytics, machine learning, and AI algorithms to process and make sense of the data.

3) Traffic Flow Prediction:

Develop predictive algorithms that can anticipate traffic patterns, congestion, and potential bottlenecks based on historical data and real-time inputs.

4) Adaptive Traffic Control:

Implement an adaptive traffic signal control system that can dynamically adjust traffic light timings based on real-time traffic data. This could include prioritizing lanes with heavy traffic or dynamically changing signal patterns during special events.

5) Dynamic Route Guidance:

Develop a mobile app or integrate with existing navigation apps to provide real-time traffic information to drivers. Suggest alternative routes to reduce congestion and travel times.

6) Public Transportation Integration:

Integrate data from public transportation systems, such as buses and trains, into the traffic management system. Coordinate traffic signals to prioritize public transportation during peak hours.

7) Emergency Response Integration:

Implement a system that can quickly respond to emergencies, such as accidents or road closures, by rerouting traffic and notifying relevant authorities.

8) User Engagement and Feedback:

Engage with the community through mobile apps and online platforms to gather feedback and insights from residents and commuters. Use this feedback to further improve the system.

9) Environmental Impact Monitoring:

Continuously monitor air quality and other environmental factors. Use this data to assess the environmental impact of traffic and implement measures to reduce pollution.

10) Data Security and Privacy:

Ensure that data collected from IoT devices is secure and anonymized to protect the privacy of individuals.

11) Scalability and Future-Proofing:

Design the system to be scalable and adaptable to accommodate the growing needs of the city and emerging IoT technologies.

12) Collaboration with City Authorities:

Collaborate with city authorities, transportation departments, and relevant stakeholders to ensure the system aligns with city planning and policies.

13) Testing and Iteration:

Conduct extensive testing and simulations to fine-tune the system's algorithms and ensure its effectiveness.

14) Public Awareness and Education:

Educate the public about the benefits of the IoT-based Traffic Management System and how they can use it to reduce congestion and improve their daily commute.

15) Cost Analysis and Sustainability:

Evaluate the costs associated with implementing and maintaining the system and assess its long-term sustainability and economic benefits.

17) Regulatory Compliance:

Ensure that the system complies with all relevant regulations and standards, particularly regarding data privacy and environmental impact.

**TRAFFIC MANAGEMENT USING IOT**

**PHASE 3**

**SENSOR DESIGN SIMULATION USING WOKWI**

**INTRODUCTION:**

In phase 3, we have used the Arduino Uno in designing a circuit to manage traffic using IOT. The simulation of the circuit was done using WOKWI simulator. Simulations are also an essential tool for testing and refining Intelligent Transportation Systems (ITS) and Internet of Things (IoT) solutions, such as smart traffic lights, dynamic traffic signal control, and adaptive traffic management systems.

**REQUIREMENTS:**

Creating a traffic management system in an online simulation platform like Wokwi requires multiple components, including a microcontroller (such as an Arduino) and a way to visualize and control the system like LEDS and a pedestrian pushbutton.

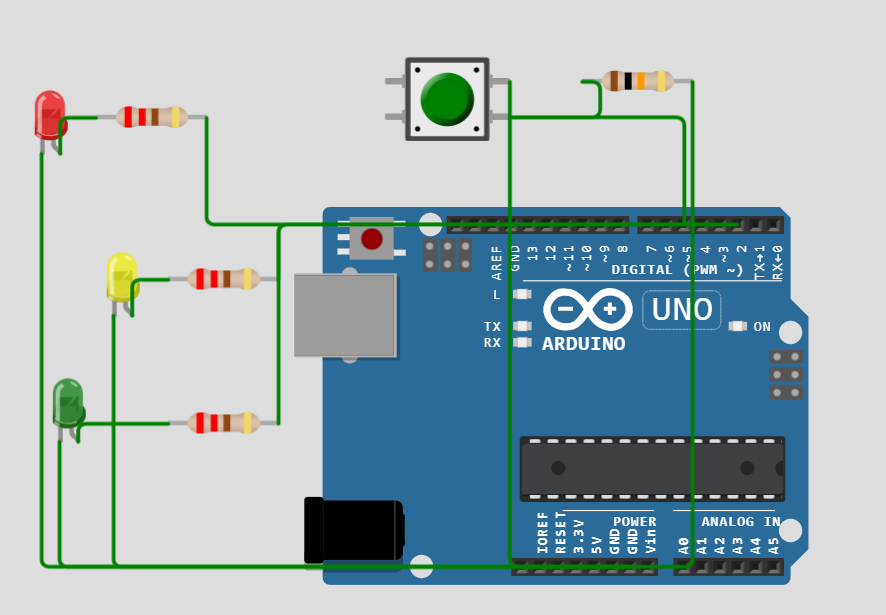
* Arduino Uno:

The Arduino Uno is an [open-source](https://en.wikipedia.org/wiki/Open-source) [microcontroller board](https://en.wikipedia.org/wiki/Single-board_microcontroller) based on the [Microchip](https://en.wikipedia.org/wiki/Microchip_Technology) [ATmega328P](https://en.wikipedia.org/wiki/ATmega328P) [microcontroller](https://en.wikipedia.org/wiki/Microcontroller) (MCU) and developed by [Arduino.cc](https://en.wikipedia.org/wiki/Arduino) and initially released in 2010.[[2]](https://en.wikipedia.org/wiki/Arduino_Uno#cite_note-2)[[3]](https://en.wikipedia.org/wiki/Arduino_Uno#cite_note-What_is_Arduino?-3) The [microcontroller board](https://en.wikipedia.org/wiki/Single-board_microcontroller) is equipped with sets of digital and analog [input/output](https://en.wikipedia.org/wiki/Input/output) (I/O) pins that may be interfaced to various [expansion boards](https://en.wikipedia.org/wiki/Expansion_board) (shields) and other circuits



* LEDS: to represent the traffic lights
* Breadboard and Wires
* Resistors: 220Ω(3N) , 10KΩ(1N)

**CIRCUIT DESIGN:**



**PROGRAM:**

const int redPin = 2;

const int yellowPin = 3;

const int greenPin = 4;

const int pedestrianButtonPin = 5;

void setup() {

  pinMode(redPin, OUTPUT);

  pinMode(yellowPin, OUTPUT);

  pinMode(greenPin, OUTPUT);

  pinMode(pedestrianButtonPin, INPUT\_PULLUP);

}

void loop() {

  // Normal traffic light sequence

  digitalWrite(redPin, HIGH);

  delay(5000);

  digitalWrite(redPin, LOW);

  digitalWrite(yellowPin, HIGH);

  delay(2000);

  digitalWrite(yellowPin, LOW);

  digitalWrite(greenPin, HIGH);

  delay(5000);

  digitalWrite(greenPin, LOW);

  // Check if the pedestrian button is pressed

  if (digitalRead(pedestrianButtonPin) == LOW) {

    // Pedestrian crossing sequence

    digitalWrite(redPin, HIGH);

    delay(2000);

    digitalWrite(redPin, LOW);

    delay(500);

    digitalWrite(yellowPin, HIGH);

    delay(1000);

    digitalWrite(yellowPin, LOW);

    delay(500);

    digitalWrite(redPin, HIGH);

    delay(2000);

    digitalWrite(redPin, LOW);

  }

}

**EXPLANATION:**

The provided Arduino code simulates a basic traffic management system with traffic lights and a pedestrian crossing button.

1. The code begins by defining the pins for the three traffic light LEDs (red, yellow, and green) and the pedestrian button. These pins are connected to the Arduino.

2. In the `setup` function, the pins are configured:

- `redPin`, `yellowPin`, and `greenPin` are set as OUTPUT pins, indicating that they control the traffic light LEDs.

- `pedestrianButtonPin` is set as an INPUT\_PULLUP pin, enabling the internal pull-up resistor. This configuration ensures that when the pedestrian button is not pressed, the pin reads HIGH (due to the pull-up resistor).

3. The `loop` function contains the main logic for the traffic light and pedestrian crossing sequences:

- The traffic light sequence follows a standard pattern: red light for 5 seconds, yellow light for 2 seconds, and green light for 5 seconds. This pattern repeats continuously.

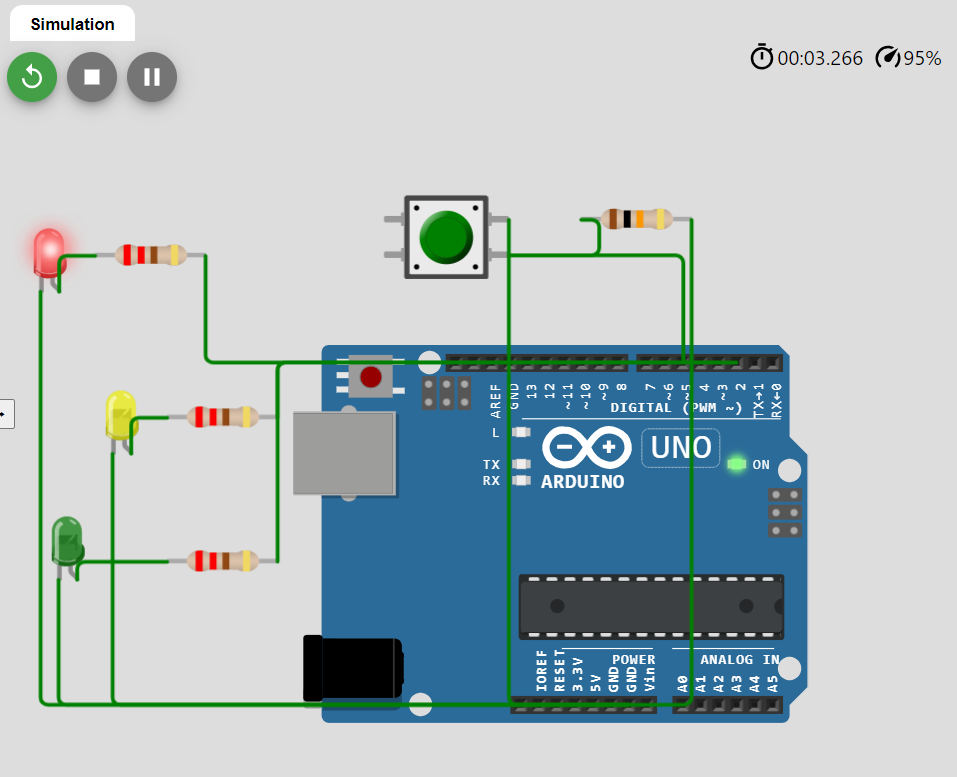
- It then checks if the pedestrian button is pressed by monitoring the state of `pedestrianButtonPin`. If the button is pressed (reads LOW), it initiates the pedestrian crossing sequence.

- During the pedestrian crossing sequence, the red light turns on for 2 seconds, followed by a 0.5-second delay, then the yellow light for 1 second, another 0.5-second delay, and finally the red light for 2 seconds. This sequence allows pedestrians to safely cross the road.

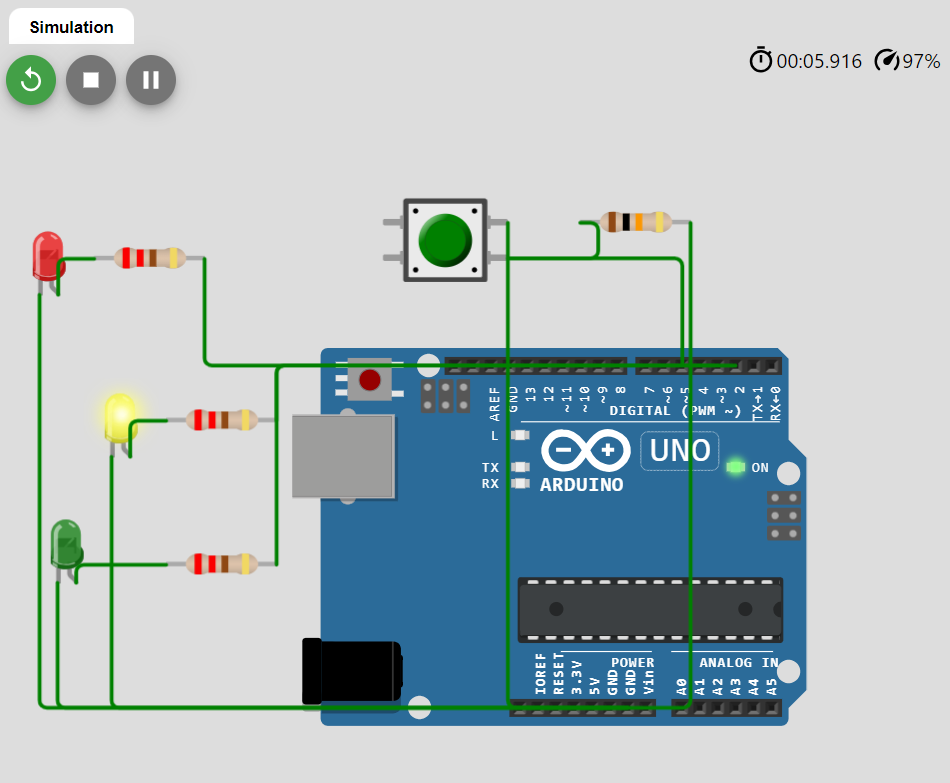
4. The `delay` function is used to control the timing of the different phases in the traffic light and pedestrian crossing sequences.

**OUTPUT:**

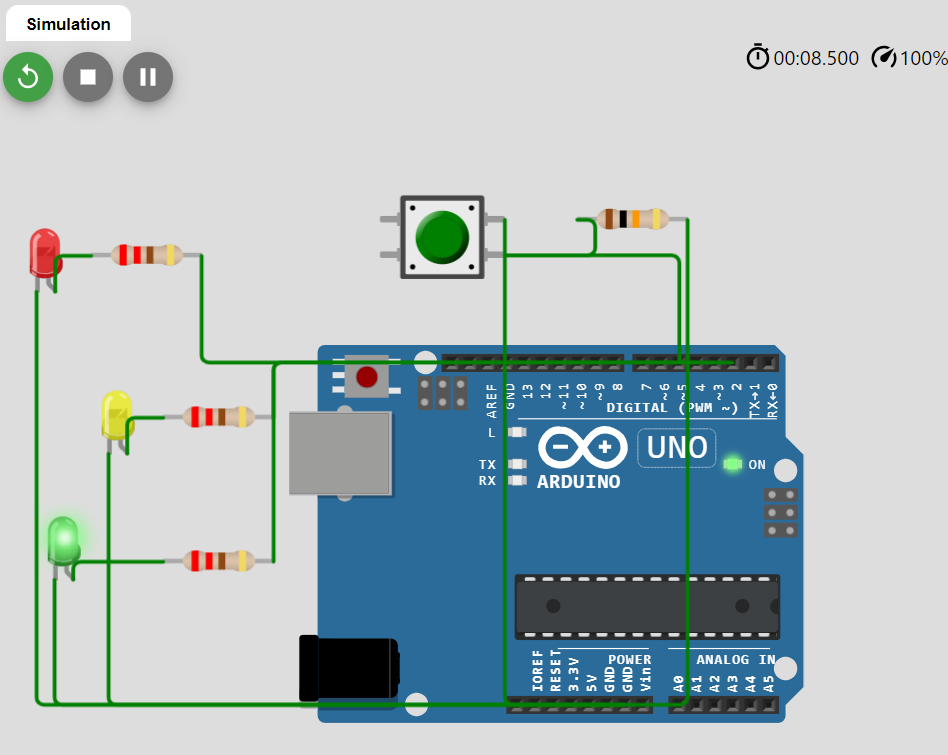
Red Light:



Yellow Light:



Green Light:



**A screenshot of a computer

Description automatically generated**

https://wokwi.com/projects/380672718522131457

**TRAFFIC MANAGEMENT USING IOT**

**PHASE 4**

**In this technology project you will continue building your project by developing the platform as per project requirement. Use web development technologies wherever needed. After performing the relevant activities create a document around it and share the same for assessment.**

**ABOUT AZURE:**

Azure is Microsoft's cloud computing platform offering a broad range of cloud services, including infrastructure, platforms, and software. It enables businesses to build, deploy, and manage applications and services in Microsoft-managed data centers. Azure supports virtual machines, databases, AI, IoT, and more, with global data center coverage, robust security, and a pay-as-you-go pricing model. Its scalability, developer tools, and integration with Microsoft services make it a popular choice for organizations seeking cloud solutions to drive innovation, streamline operations, and meet their digital transformation needs.

**Code to display Real time Transit Information azure:**

pip install requests

import requests

import azure.functions as func

def main(req: func.HttpRequest) -> func.HttpResponse:

# Replace with your actual transit API endpoint

api\_endpoint = "https://your-transit-api.com/realtime"

# Replace with your API credentials if needed

api\_headers = {"Authorization": "Bearer YOUR\_API\_KEY"}

response = requests.get(api\_endpoint, headers=api\_headers)

if response.status\_code == 200:

transit\_data = response.json()

return func.HttpResponse(transit\_data, mimetype="application/json")

else:

return func.HttpResponse("Failed to retrieve transit data", status\_code=500)

**Code for Interfacing Data with Azure:**

import machine

import utime

from azure.iot.device import IoTHubDeviceClient, Message

# Define Azure IoT Hub connection string and device ID

CONNECTION\_STRING = "HostName=TrafficManagement.azure-devices.net;DeviceId=RaspPi;SharedAccessKey=7ryCImqSUWqhp5ChKRrkZxNCwpjqtJHL2AIoTGkwJBs="

DEVICE\_ID = "RaspPi"

# Initialize the IoT Hub client

client = IoTHubDeviceClient.create\_from\_connection\_string(CONNECTION\_STRING)

# GPIO pins for the HC-SR04 sensor

trigger\_pin = machine.Pin(2, machine.Pin.OUT)

echo\_pin = machine.Pin(3, machine.Pin.IN)

# Traffic light control pins (simulated)

red\_light = machine.Pin(10, machine.Pin.OUT)

yellow\_light = machine.Pin(11, machine.Pin.OUT)

green\_light = machine.Pin(12, machine.Pin.OUT)

def measure\_distance():

trigger\_pin.value(0)

    utime.sleep\_us(2)

    trigger\_pin.value(1)

    utime.sleep\_us(10)

    trigger\_pin.value(0)

    while echo\_pin.value() == 0:

        pulse\_start = utime.ticks\_us()

    while echo\_pin.value() == 1:

        pulse\_end = utime.ticks\_us()

    pulse\_duration = utime.ticks\_diff(pulse\_end, pulse\_start)

    distance = (pulse\_duration \* 0.0343) / 2  # Speed of sound is approximately 343 meters per second

    return distance

def control\_traffic\_lights(distance):

if distance < 10:

red\_light.value(0)

yellow\_light.value(1)

green\_light.value(0)

elif 10 <= distance < 20:

red\_light.value(1)

yellow\_light.value(0)

green\_light.value(0)

else:

red\_light.value(0)

yellow\_light.value(0)

green\_light.value(1)

while True:

distance = measure\_distance()

control\_traffic\_lights(distance)

# For simulation purposes, print the distance and the traffic light state

print("Distance: {:.2f} cm".format(distance))

# Send distance data to IoT Hub

telemetry\_data = {"distance\_cm": distance}

message = Message(telemetry\_data)

client.send\_message(message)

utime.sleep(2) # Adjust the sleep duration as needed

**Steps we followed while working with Azure:**

1. We created an IoT Hub in Microsoft Azure and Deployed it.

A screenshot of a computer

Description automatically generated

1. We created a Device Id, RaspPi and created a Primary Connection String, by using this, we interfaced it with our Wowki Code, to establish the web application:

A computer screen shot of a computer screen

Description automatically generated

**DeviceID:** RaspPi

**Primary Connection string**:HostName=TrafficManagement.azure-devices.net;DeviceId=RaspPi;SharedAccessKey=7ryCImqSUWqhp5ChKRrkZxNCwpjqtJHL2AIoTGkwJBs=

The remaining step is to integrate this code in wokwi, importing azure.iot.device which is already installed using the command**:**

**pip intall azure-iot-devices**