Traffic Management System

Project Objective:

The primary objective of this project is to develop an integrated approach for urban traffic management and real-time transit information dissemination using IoT technology. This initiative aims to enhance traffic efficiency, reduce congestion, improve public transportation services, and provide commuters with real-time information for a smoother and more efficient travel experience.

IOT Sensor Design:

The IOT sensor design plays a crucial role in the success of this project. It involves the deployment of various sensors and data collection devices throughout the city infrastructure. These sensors should include:

- **Traffic Flow Sensors:** These sensors monitor vehicle speed, count, and direction, providing real-time data to the traffic management system.

- **Traffic Cameras:** High-definition cameras capture images and video feeds at key intersections and road segments to enable traffic monitoring and incident detection.

- **Environmental Sensors:** Sensors measuring air quality, weather conditions, and noise levels help factor in environmental considerations for traffic management decisions.

- **Public Transit Sensors:** IoT devices on buses, trams, and subway cars collect data on their locations, passenger loads, and schedules.

- **Pedestrian Sensors:** These sensors monitor pedestrian movement and help in optimizing pedestrian crossings and traffic signal timings.

- **Emergency Sensors:** Sensors on emergency vehicles and at critical locations help provide priority and clear pathways in emergency situations.

Real-Time Transit Information Platform:

The Real-Time Transit Information Platform serves as a central component of the project and includes the following features:

- **Passenger Information Displays:** Digital screens at bus stops, subway stations, and transit hubs display real-time information on public transit schedules, delays, and estimated arrival times.

- **Mobile Application:** A mobile app accessible to commuters provides up-to-the-minute information on public transportation options, routes, and service disruptions. It includes features like trip planning, payment integration, and alerts.

- **API Integration:** The platform should offer APIs for third-party developers to integrate real-time transit data into their applications, encouraging the development of transit-related tools.

Integrated Approach for Traffic Management System:

The integrated approach for traffic management involves the following key elements:

- **Data Fusion and Analysis:** Collected data from IoT sensors, traffic cameras, and transit vehicles are processed and analysed in real-time to monitor traffic conditions and identify congestion or incidents.

- **Smart Traffic Signal Control:** Traffic signals are dynamically adjusted based on real-time traffic data to optimize traffic flow and reduce congestion.

- **Public Transportation Coordination:** Transit agencies and traffic management authorities collaborate to synchronize public transportation services with traffic signal timings, reducing delays and improving the overall transit experience.

- **Emergency Response Integration:** The system includes protocols for coordinating with emergency services during accidents or incidents, ensuring rapid response and efficient traffic management.

- **Public Engagement:** Public awareness campaigns and user-friendly interfaces, such as mobile apps and digital displays, keep commuters informed and engaged in the transit system.

<u>**Solution:**</u>

The integrated Traffic Management System (TMS) outlined in the project objectives can be achieved through a carefully planned and technologically advanced solution. Here's an overview of the components and strategies involved:

1. **IoT Sensor Network:**

- Deploy a network of sensors and cameras throughout the city's infrastructure, including traffic flow sensors, traffic cameras, environmental sensors, public transit sensors, pedestrian sensors, and emergency sensors.
- Utilize cutting-edge IoT technology with low-power, long-range communication capabilities to ensure data collection from various locations across the city.

2. **Data Collection and Transmission:**

- Gather real-time data from the IoT sensor network and transmit it securely to a centralized server or cloud-based platform.
- Implement data encryption and secure communication protocols to protect sensitive information.

3. **Data Fusion and Analysis:**

- Use advanced data analytics, machine learning, and artificial intelligence algorithms to process and analyse the collected data.

- Identify traffic patterns, congestion hotspots, environmental factors, and transit service data to make informed decisions.

4. Smart Traffic Signal Control:

- Implement adaptive traffic signal control algorithms that consider real-time traffic data to optimize signal timings.
- Prioritize lanes and intersections based on congestion levels and transit schedules to reduce delays.

5. Real-Time Transit Information Platform:

- Develop a user-friendly mobile application and passenger information displays at transit stops and stations.
- Integrate APIs for third-party developers to encourage the creation of transit-related apps and services.
- Provide real-time updates on transit schedules, delays, estimated arrival times, and trip planning features.

6. Public Engagement:

- Launch public awareness campaigns to inform commuters about the benefits of the integrated TMS.
- Use digital displays, mobile apps, and social media to keep the public informed about real-time traffic conditions, alternative routes, and sustainability goals.

7. Emergency Response Integration:

- Establish protocols for real-time coordination between the TMS and emergency services.
- Provide emergency vehicles with priority pathways during critical situations.

8. Multi-Modal Commuter Rewards Program (Innovation):

- Develop a gamified platform that incentivizes commuters to choose sustainable transportation options.
- Enable real-time suggestions and tracking of carbon footprint reduction.
- Encourage community engagement through team challenges and competitions.

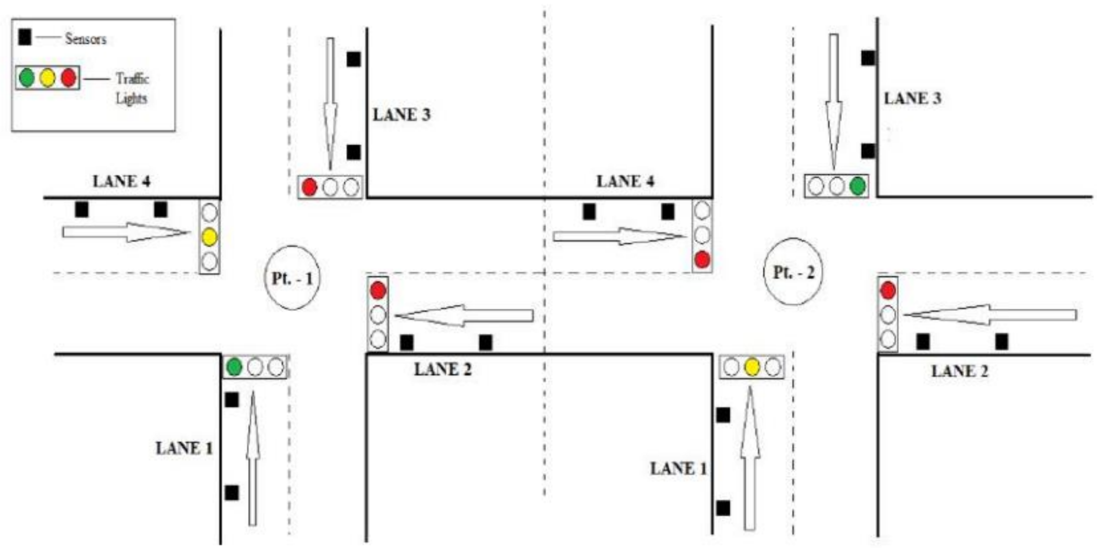
9. Public Transportation Coordination:
- Facilitate collaboration between transit agencies and traffic management authorities to synchronize transit services with traffic signal timings.
- Implement real-time updates and coordination mechanisms for efficient transit operations.

10. Data-Driven Insights:
- Use data generated by the TMS, including commuter behaviour and traffic patterns, to make informed decisions for urban planning and transportation infrastructure improvements.

11. Scalability and Future-Proofing:
- Design the TMS solution to be scalable, allowing for expansion to cover more areas as the city grows.
- Incorporate flexibility to integrate future technologies and adapt to changing urban transportation needs.

This comprehensive solution for the Traffic Management System leverages IoT technology, data analytics, real-time information dissemination, and community engagement to create a more efficient, sustainable, and user-friendly urban transportation ecosystem. It addresses traffic congestion, enhances public transit, and empowers commuters to make informed and eco-conscious transportation choices.

Model Diagram:



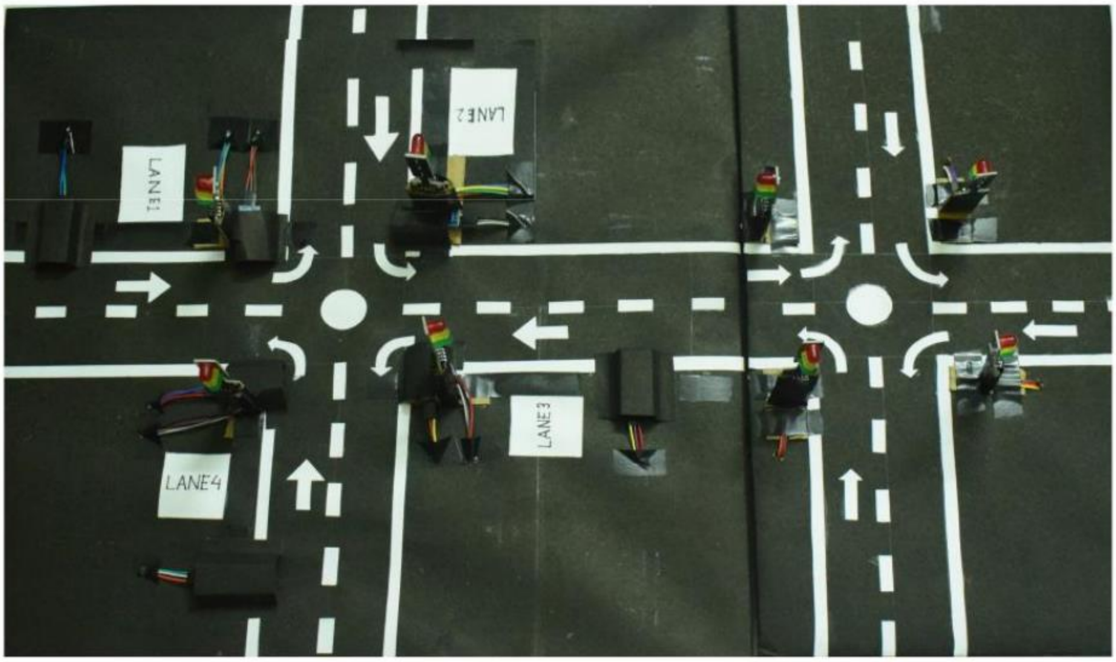
Program:

```
import paho.mqtt.client as mqtt

import json

import random

import time


# MQTT broker information

mqtt_broker = "your_mqtt_broker_address"

mqtt_port = 1883

mqtt_topic = "traffic_data"
```

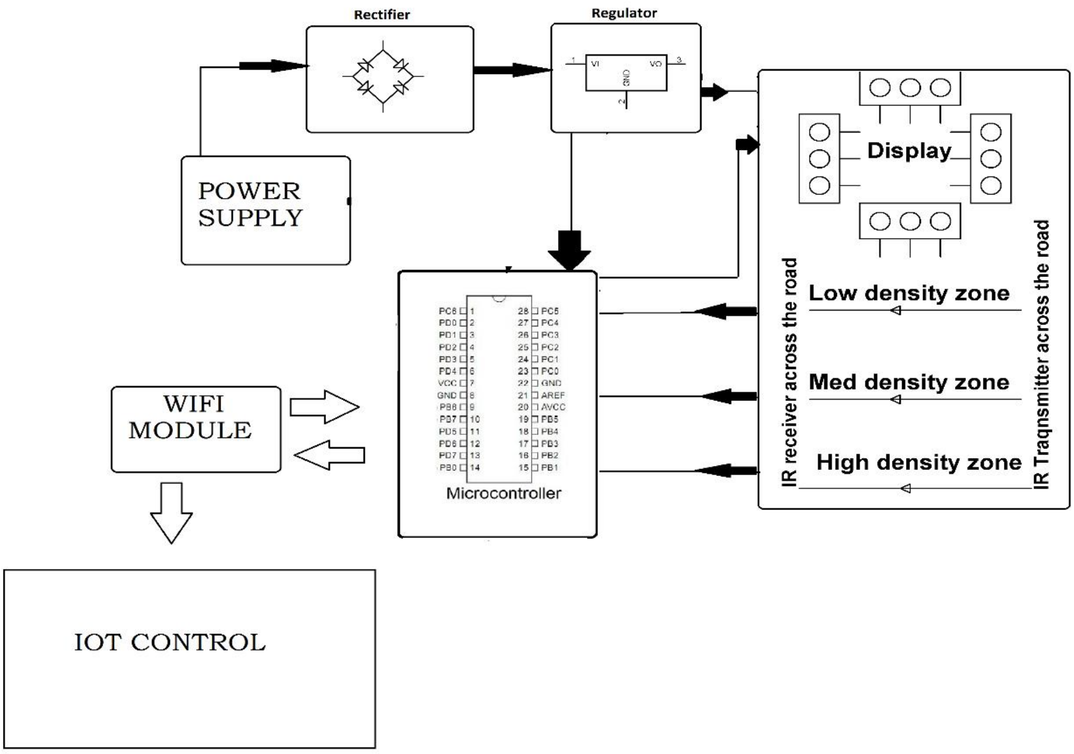
```python
# Simulated IoT device information
device_id = "iot_device_1"
location = "Intersection A"

# Create an MQTT client
client = mqtt.Client(device_id)

# Callback when the client successfully connects to the MQTT broker
def on_connect(client, userdata, flags, rc):
    if rc == 0:
        print(f"Connected to MQTT broker at {mqtt_broker}")
    else:
        print(f"Failed to connect to MQTT broker with code {rc}")

# Set the on_connect callback
client.on_connect = on_connect

# Connect to the MQTT broker
client.connect(mqtt_broker, mqtt_port)

# Function to generate simulated traffic data
def generate_traffic_data():
    timestamp = int(time.time())
```

```
    vehicle_count = random.randint(0, 50)
    average_speed = round(random.uniform(10, 60), 2)
    status = "normal" if random.random() < 0.9 else "congested"

    traffic_data = {
        "device_id": device_id,
        "location": location,
        "timestamp": timestamp,
        "vehicle_count": vehicle_count,
        "average_speed": average_speed,
        "status": status,
    }
    return json.dumps(traffic_data)
try:
    # Loop to continuously send traffic data
    while True:
        traffic_data = generate_traffic_data()
        # Publish traffic data to the MQTT topic
        client.publish(mqtt_topic, traffic_data)
        print(f"Published: {traffic_data}")
        time.sleep(5)  # Simulate sending data every 5 seconds
except KeyboardInterrupt:
    print("Script terminated by the user.")
    client.disconnect()
```

Implementation Diagram:



Circuit Diagram:



<u>Web Application:</u>

Index.html:

```
<!DOCTYPE html>
<html lang=”en”>
<head>
   <meta charset=”UTF-8”>
   <meta name=”viewport” content=”width=device-width, initial-scale=1.0”>
   <title>Traffic Management</title>
   <link href=https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css rel=”stylesheet”>
   <script src=https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js></script>
   <link rel=”stylesheet” href=”./style.css”>
   <script src=”./script.js”></script>
</head>
<body>
    <div class=”container-fluid  text-center bg-danger” style=”min-height:15vh;”>
      <h2 class=”text-white  py-4 px-5 “  >TRAFFIC MANAGEMENT SYSTEM</h2>
   </div>
```

```
   <div class=”container”>
      <h2 class=”text-center mt-5” >Traffic Light
Simulation</h2><br><br>
      <div class=”row mt-3”>
         <div class=”col-md-3”><h4>Lane 1</h4>
            <h5>Vehicle Count = <span id=”ph-level”>12</span></h5>
            <div id=”traffic-light”>
               <input type=”radio” name=”traffic-light-color” id=”color1”
value=”color1” />
               <input type=”radio” name=”traffic-light-color” id=”color2”
value=”color2”/>
               <input type=”radio” name=”traffic-light-color” id=”color3”
value=”colo3” />
              </div>
         </div>
         <div class=”col-md-3”><h4> Lane 2</h4>


            <h5>Vehicle Count =<span
id=”conductivity”>42</span></h5>
            <div id=”traffic-light”>
               <input type=”radio” name=”traffic-light-color” id=”color1”
value=”color1” />
               <input type=”radio” name=”traffic-light-color” id=”color2”
value=”color2”/>
               <input type=”radio” name=”traffic-light-color” id=”color3”
value=”colo3” />
```

```
            </div>
        </div>
        <div class=”col-md-3”><h4>Lane 3</h4>

          <h5>Vehicle Count = <span
id=”temperature”>18</span></h5>
          <div id=”traffic-light”>
            <input type=”radio” name=”traffic-light-color” id=”color1”
value=”color1” />
            <input type=”radio” name=”traffic-light-color” id=”color2”
value=”color2”/>
            <input type=”radio” name=”traffic-light-color” id=”color3”
value=”colo3” />
            </div>
        </div>
        <div class=”col-md-3”><h4>Lane 4</h4>

          <h5>Vehicle Count = <span
id=”temperature1”>1</span></h5>
          <div id=”traffic-light”>
  <input type=”radio” name=”traffic-light-color” id=”color1”
value=”color1” />
Hi  <input type=”radio” name=”traffic-light-color” id=”color2”
value=”color2”/>
  <input type=”radio” name=”traffic-light-color” id=”color3”
value=”colo3” />
```

```
</div>
            </div>
        </div>
    </div>
</body>
</html>
```

Style.css:

```
#traffic-light {
  Margin-left: 20px;
  Margin-top: 40px;
  Background-color:#333;
  Width:120px;
  Height:320px;
  Border-radius:30px;
}
Input {
  Appearance: none;
  Position: relative;
  Left: 50%;
  Width:80px;
  Height:80px;
  Margin-top: 30px;
```

```
 Margin-left:-40px;
 Background-color: grey;
 Border-radius: 100%;
 Display: block;
&#color1 {
  Background-color: darken( #FF0000,15%);
  &:hover {
    Animation: blink1 1.1s step-end infinite;
  }
  &:checked {
    Background-color: #FF0000;
    Box-shadow: 0 0 6em lighten( #FF0000,10%);
  }
 }
 &#color2 {
  Background-color: darken(#FFFF00,15%);
  &:hover {
    Animation: blink2 1s step-end infinite;
  }
  &:checked {
    Background-color: #FFFF00;
    Box-shadow: 0 0 6em lighten(#FFFF00,10%);
  }
 }
```

```
  &#color3 {
    Background-color: darken(#00FF00,15%);
    &:hover {
      Animation: blink3 1s step-end infinite;
    }
    &:checked {
      Background-color:#00FF00;
      Box-shadow: 0 0 6em lighten(#00FF00,10%);
    }
  }
}
@keyframes blink1 {
  0% {
    Background-color:#FF0000;
    Box-shadow: 0 0 6em lighten(#FF0000,10%);
  }
  50% {
    Background-color: darken(#FF0000,15%);
    Box-shadow: 0 0 0em transparent;
  }
}
@keyframes blink2 {
  0% {
    Background-color: #FFFF00;
```

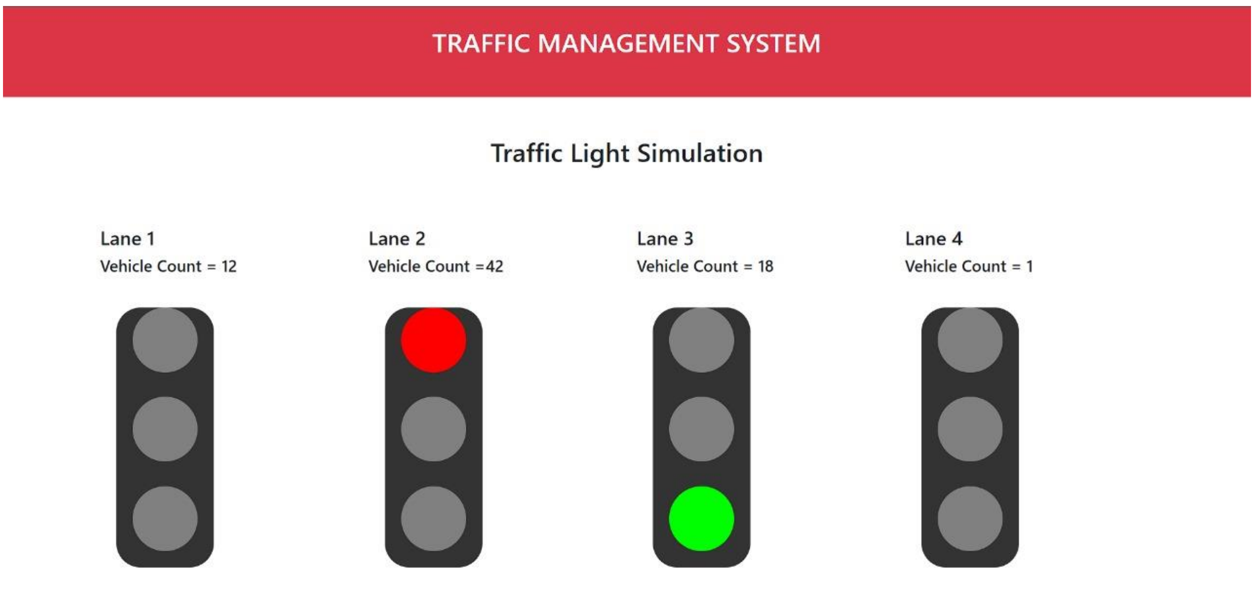
```
  Box-shadow: 0 0 6em lighten(#FFFF00,10%);
 }
 50% {
  Background-color: darken(#FFFF00,15%);
  Box-shadow: 0 0 0em transparent;
 }
}
@keyframes blink3 {
 0% {
  Background-color: #00FF00;
  Box-shadow: 0 0 6em lighten(#00FF00,10%);
 }
 50% {
  Background-color: darken(#00FF00,15%);
  Box-shadow: 0 0 0em transparent;
 }
}
```

Script.js:

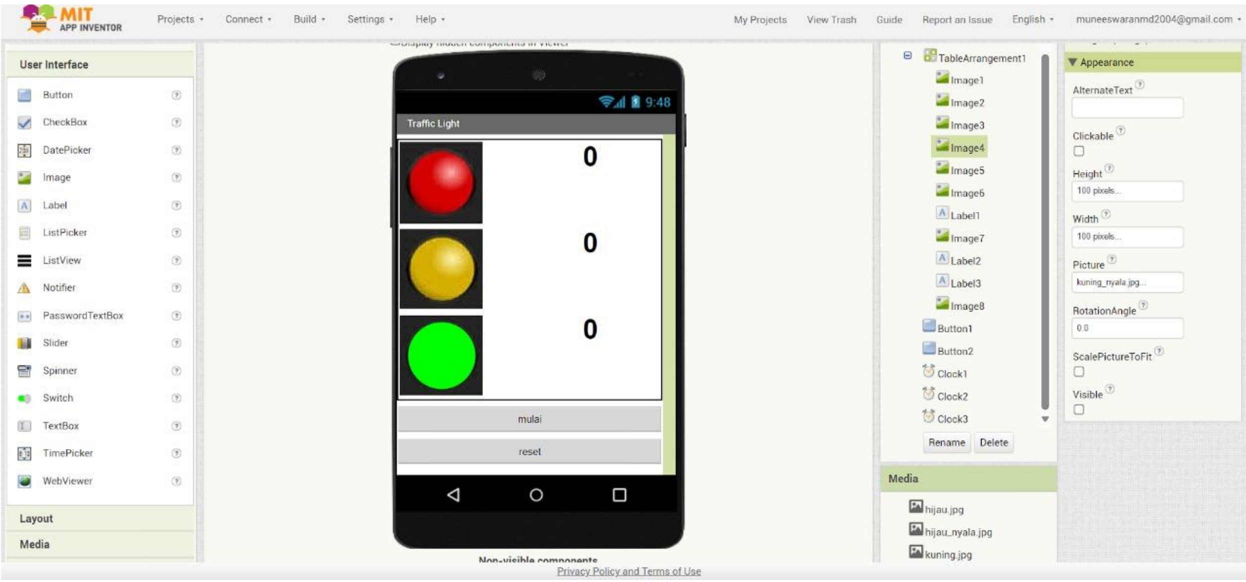
```
Function updateData() {
  // Simulated data (replace with actual data retrieval logic)
  Const phValue = (Math.random() * 14).toFixed(2);
  Const conductivityValue = (Math.random() * 50).toFixed(2);
  Const temperatureValue = (Math.random() * 30).toFixed(2);
```

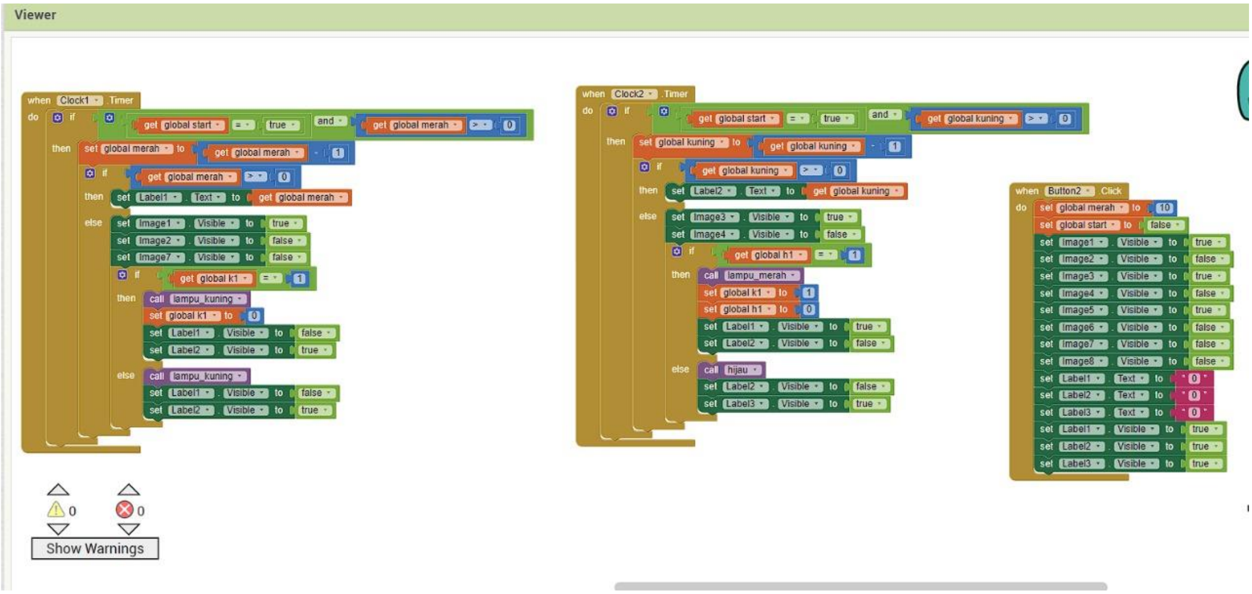
```
            Const temperature1Value = (Math.random() * 30).toFixed(2);

            // Update the HTML elements with the new data
            Document.getElementById("ph-level").textContent = phValue;
            Document.getElementById("conductivity").textContent =
conductivityValue;
            Document.getElementById("temperature1").textContent =
temperature1Value;
            Document.getElementById("temperature").textContent =
temperature1Value;
        }

        // Poll for updates every 5 seconds (adjust as needed)
        setInterval(updateData, 11000);
```

Output:



Mobile Application:




Viewer
when Clock1 . Timer
when Clock2 . Timer
when Button2 . Click
0
0
Show Warnings