Public Transport Optimization Phase 5

Objective:

The objective of this project is to optimize public transportation systems by leveraging the Internet of Things (IoT) technology. The project aims to enhance the efficiency, reliability, and overall user experience of public transportation services by collecting and analyzing real-time data from IoT devices deployed in vehicles, infrastructure, and passenger hubs.

IoT Device Setup:

Vehicle Sensors: IoT sensors will be installed in public transport vehicles, including buses, trams, and trains. These sensors will collect data such as vehicle location, speed, fuel consumption, passenger count, and engine health.

Passenger Counters: IoT cameras or sensors will be placed at vehicle entry and exit points to track passenger counts and occupancy in real-time.

Infrastructure Sensors: IoT sensors will be deployed at bus stops, train stations, and other passenger hubs to collect data on passenger foot traffic, environmental conditions (e.g., temperature, humidity), and vehicle arrival times.

Platform Development:

The IoT data collected from the devices will be transmitted to a centralized platform for analysis and decision-making. The platform will consist of the following components:

IoT Gateway: Data from the IoT devices will be sent to a central gateway for aggregation and forwarding. This gateway will be responsible for data preprocessing and transmission.

Cloud Data Storage: The preprocessed data will be stored in a cloud-based database for real-time and historical analysis.

Data Analysis and Prediction: Machine learning models will be developed to analyze the data and make predictions about vehicle schedules, maintenance needs, and passenger demand.

Dashboard: A user-friendly dashboard will be created for transportation authorities and commuters to access real-time information about vehicle locations, passenger counts, and estimated arrival times.

Code Implementation:

The code for this project will be written in various programming languages depending on the component:

IoT Device Code: Each IoT device will have its own code to read sensor data and transmit it to the IoT gateway. This code may be written in C, Python, or other suitable languages.

IoT Gateway Code: The gateway will have code to aggregate data, perform preprocessing, and securely transmit it to the cloud. It may use MQTT, HTTP, or other communication protocols.

Cloud Data Storage Code: This involves setting up and managing databases, which can be done using platforms like AWS, Azure, or Google Cloud.

Data Analysis and Prediction Code: Machine learning models will be implemented using Python, along with libraries like TensorFlow or scikit-learn.

Dashboard Code: Web-based dashboards can be developed using HTML, CSS, and JavaScript, along with frameworks like React or Angular.

Explanation in Detail:

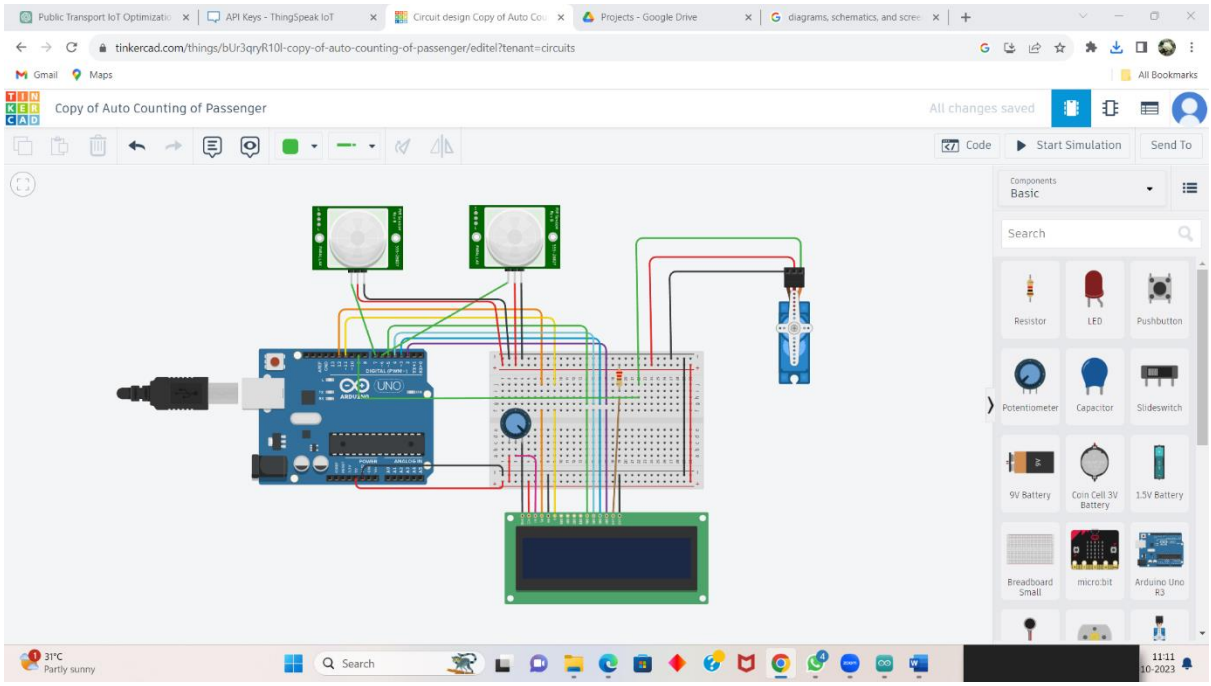
The system will collect real-time data on vehicle locations, passenger counts, and environmental conditions. This data will be analyzed to optimize transportation routes, schedules, and vehicle maintenance. Passengers will have access to accurate arrival time predictions, reducing wait times and improving their experience. Transportation authorities will have tools to make data-driven decisions to enhance the efficiency and reliability of their services.

The project's success will lead to more sustainable, convenient, and user-friendly public transportation systems, benefiting both commuters and the environment.

URL for Tinkercad Project of Passenger Counting :

https://www.tinkercad.com/things/dAuGJH1r0NS-copy-of-auto-counting-of-passenger/editel?tenant=circuits

ScreenShot of Passenger Counting in Public Transport:

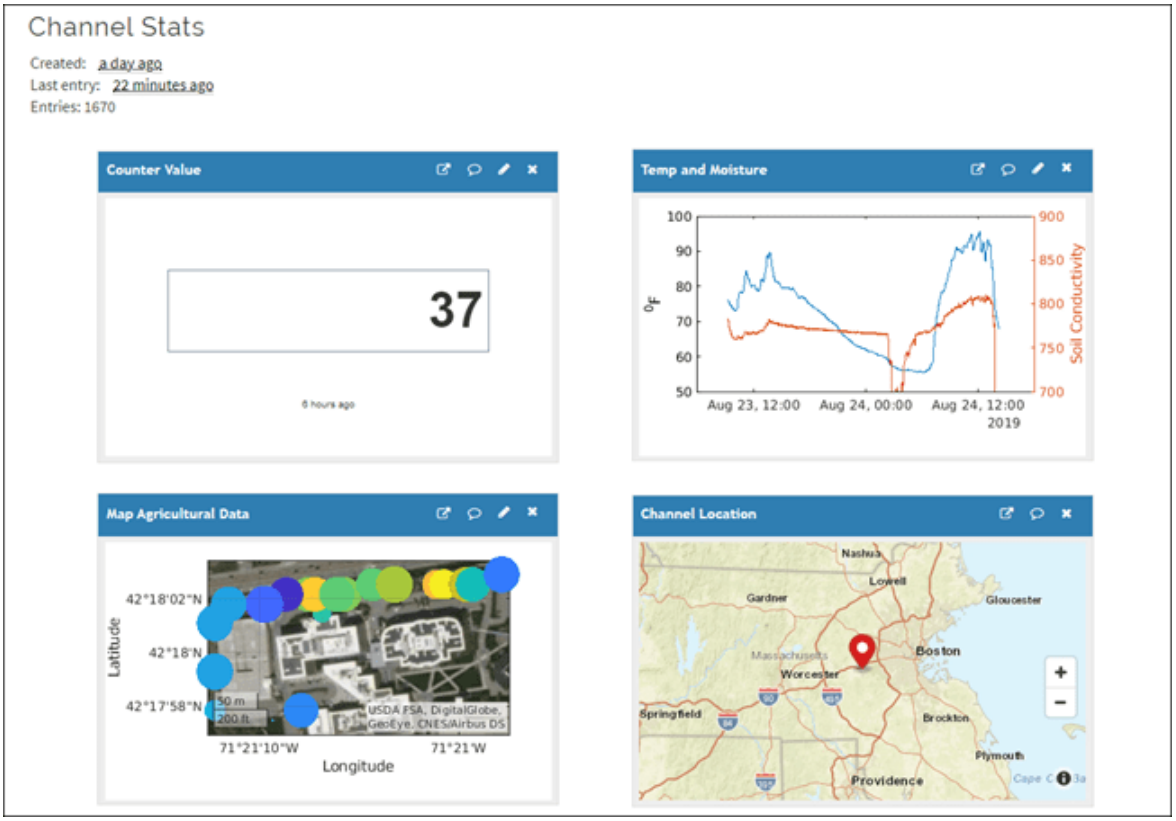


Code for Running above Circuit:

```
#include <LiquidCrystal.h>
#include <Servo.h>
#include <ThingSpeak.h> // Include the ThingSpeak library
// Define your ThingSpeak channel details
char thingSpeakAddress[] = "api.thingspeak.com";
unsigned long channelID = 2303456; // Replace with your channel ID
const char * writeAPIKey = " 6EKT0ALDBXGG60Q1"; // Replace with your Write API Key
// Rest of your code...
void setup() {
  // Initialize ThingSpeak
  ThingSpeak.begin(client); // Initialize the ThingSpeak library
  // Rest of your setup code...
}
// Update ThingSpeak with passenger count
void UpdateThingSpeak(int count) {
```

```
  ThingSpeak.setField(1, count); // Field 1 is for passenger count
  int status = ThingSpeak.writeFields(channelID, writeAPIKey);
  if (status == 200) {
    Serial.println("ThingSpeak update successful");
  } else {
    Serial.println("Error updating ThingSpeak");
  }
}
// Update the passenger count and ThingSpeak when a passenger enters or exits
void UpdatePassengerCounter(int x) {
  Passenger = Passenger + x;
  lastRIPdetected = 0;
  if (Passenger >= 0) {
    UpdateThingSpeak(Passenger); // Update ThingSpeak with the new passenger count
  }
}
```

Thingspeak Channel stats of Passengers Counting:



Platform UI Code for Public Transport Optimization:

```
<!DOCTYPE html>
<html>
<head>
  <title>Bus Passenger Counter</title>
</head>
<body>
  <h1>Bus Passenger Counter</h1>
  <p>Passenger Count: <span id="passengerCount">Loading...</span></p>
  <script>
    // Function to update passenger count from ThingSpeak
    function updatePassengerCount() {
      // Make an AJAX request to fetch the passenger count from ThingSpeak
      var xhr = new XMLHttpRequest();
      xhr.open("GET",
"https://api.thingspeak.com/update?api_key=OL6MICDSS2G0VN7J&field1=0", true);
      xhr.onreadystatechange = function () {
```

```
            if (xhr.readyState == 4 && xhr.status == 200) {
                var count = xhr.responseText;
                document.getElementById("passengerCount").textContent = count;
            }
        };
        xhr.send();
    }
    // Periodically update passenger count (e.g., every 10 seconds)
    setInterval(updatePassengerCount, 10000);
    // Initial update
    updatePassengerCount();
  </script>
</body>
</html>
```

Output for above Program: