**5. PUBLIC TRANSPORT OPTIMIZATION**

1. **PROBLEM DEFINITION**

Optimization helps businesses reduce costs and increase revenues. Optimization also benefits the market by reducing deadweight and inefficiencies.

Optimizing a public transport network involves making changes to the routes, schedules, and infrastructure of the network to improve efficiency, reduce costs, and improve the overall quality of service for passengers.That's what transportation optimization does,

and the rewards are significant:

* Shipping cost reductions up to 50%.
* Increased efficiency and expanded capacity for shippers and carriers.
* Higher levels of customer satisfaction.

1. **PROJECT DEFINITION**

The Internet of Things, or IoT, refers to the billions of physical devices around the world that are now connected to the internet, all collecting and sharing data. Thanks to the arrival of super-cheap computer chips and the ubiquity of wireless networks, it's possible to turn anything, from something as small as [a pill](https://www.zdnet.com/article/how-sensors-enabled-eli-lilly-to-improve-the-patient-experience/)to something as big as [an aeroplane](https://www.zdnet.com/article/ten-examples-of-iot-and-big-data-working-well-together/), into a part of the IoT. Connecting up all these different objects and adding sensors to them adds a level of digital intelligence to devices that would be otherwise dumb, enabling them to communicate real-time data without involving a human being. The Internet of Things is making the fabric of the world around us more smarter and more responsive, merging the digital and physical universes. The benefits of the IoT for business depend on the particular implementation; agility and efficiency are usually top considerations. The idea is that enterprises should have access to more data about their own products and their own internal systems, and a greater ability to make changes as a result.

With all those sensors collecting data on everything you do, the IoT is a potentially vast privacy and security headache. Take the smart home: it can tell when you wake up (when the smart coffee machine is activated) and how well you brush your teeth (thanks to your smart toothbrush), what radio station you listen to (thanks to your smart speaker), what type of food you eat (thanks to your smart oven or fridge), what your children think (thanks to their smart toys), and who visits you and passes by your house (thanks to your smart doorbell). While companies will make money from selling you the smart object in the first place, their IoT business model probably involves selling at least some of that data, too.

The huge amount of data that IoT applications generate means that many companies will choose to do their data processing in the cloud rather than build huge amounts of in-house capacity. Cloud computing giants are already courting these companies: Microsoft has its [Azure IoT suite](https://azure.microsoft.com/en-gb/suites/iot-suite/), while [Amazon Web Services](https://aws.amazon.com/iot/?tag=zd-buy-button-20&ascsubtag=__COM_CLICK_ID__%7C29efec04-70df-46ae-89a2-b27acb4a6145%7Cdtp) provides a range of IoT services, as does [Google Cloud](https://cloud.google.com/solutions/iot/).

 the public transport systems in all countries across the world are not as effective as they should and could be. Many public transport systems are not even capable of providing the basic service of carrying people from A to B in time, let alone offering them a convenient, comfortable riding experience. It is no surprise then, that despite [an increasing number of people using public transport](https://www.apta.com/mediacenter/ptbenefits/Pages/Public-Transportation-Use-is-Growing-.aspx), very few are satisfied with the services they receive. That's because most existing public transportation systems, even those in developed countries, are not very efficient to operate or convenient to use. They experience problems like frequent delays due to maintenance and other contingencies and suboptimal allotment of vehicles to different routes, leading to crowded coaches in certain places and empty rides in others.

IoT technology is widely used in many fields. Based on the latest research on smart homes and IoT technology, Hui et al. outlined specific requirements for smart home construction and proposed requirements based on specific quality specifications of smart homes. The building blocks are divided into seven independent sections. Wu offered the design and management of museum collections and RFID-based intelligent navigation systems, a new way to build smart museums. It gave everyone a new understanding of the IoT, but their experimental sources were not clear enough. As a result, the research findings do not have exact references. Optimization of vehicle routes has always been a key issue in logistics. Discussed by local and international researchers, Braekers conducted a categorization review of the literature on vehicle routing issues published between 2009 and June 2015, based on a modified version of the taxonomy at [http://www.braekers.com](http://www.braekers.com/). The 277 existing coverages were classified, and the development trend of VRP literatures was analyzed. Yao et al. proposed the box-to-collection station heterogeneous vehicle routing problem and used particle swarm optimization (PSO) to solve the problem. To improve the efficiency of the particle swarm algorithm, he adopted adaptive inertia weights and a local search strategy. Their research data are quite old, which are quite different from the actual data under current research background.

1. **DESIGN THINKING**

* **PROJECT OBJECTIVES:**  Such as real-time transit information, arrival time prediction, ridership monitoring, and enhanced public transportation services.

1. Vehicle and equipment maintenance

2. Reduce traffic congestion

3. Efficient travel

4. Improved operational awareness

5. Smarter freight stowage

6. Improved safety

7. Environmental improvements

* **IOT SENSOR DESIGN:** IoT (Internet of Things) sensors play a crucial role in optimizing public transportation systems by providing real-time data and insights to improve efficiency, safety, and passenger experience. Here are some common IoT sensors used in public transportation optimization:  
    
  **1. “GPS and GNSS Sensors: “** Global Positioning System (GPS) and Global Navigation Satellite System (GNSS) sensors are used to track the real-time location of vehicles. This data is essential for tracking bus or train movements, optimizing routes, and providing accurate arrival time predictions to passengers.  
    
  **2. “Vehicle Health Monitoring Sensors: “** IoT sensors can monitor the health of transportation vehicles by collecting data on engine performance, fuel consumption, tire pressure, and other crucial metrics. This data helps in preventive maintenance, reducing breakdowns, and improving overall fleet efficiency.  
    
  **3. “Passenger Counting Sensors: “** These sensors use infrared, ultrasonic, or camera technology to count the number of passengers boarding and disembarking vehicles. Passenger data helps transit agencies adjust service frequency and allocate resources based on demand.  
   **4. “Smart Fare Collection Sensors: “** IoT sensors integrated into fare collection systems, such as contactless payment readers or ticket validators, enable seamless, fast, and secure payment processes. They also collect data on fare transactions for analysis.  
   **5. “Environmental Sensors: “** Sensors for measuring environmental factors like air quality, temperature, humidity, and noise levels can be installed in transportation hubs and vehicles. This information can be used to monitor and improve passenger comfort and safety.  
   **6. “Traffic and Intersection Sensors: “** IoT sensors placed at intersections and traffic lights help optimize traffic flow and reduce congestion. This is especially important for buses and other forms of public transportation that operate on city streets.  
   **7. “Security and Surveillance Sensors: “** Cameras and sensors can be used for both passenger safety and security. They can monitor for accidents, suspicious activities, or unauthorized access to restricted areas within transportation facilities.  
   **8. “Energy Consumption Sensors: “** Public transportation agencies can use energy consumption sensors to monitor and optimize the energy usage of their vehicles and facilities, helping to reduce operational costs and environmental impact.  
   **9. “Weather and Road Condition Sensors: “** These sensors provide data on weather conditions and road surfaces, allowing transportation operators to make informed decisions about route changes and adjustments during adverse weather conditions.  
    
  **10. “Wi-Fi and Cellular Connectivity Sensors: “** IoT sensors for connectivity enable passengers to access Wi-Fi and cellular networks while on public transportation, improving the passenger experience and enabling real-time communication.  
    
  **11. “Seat Occupancy Sensors: “** In the era of COVID-19 and social distancing measures, seat occupancy sensors can help manage capacity and maintain safe distances between passengers.  
    
  **12. “Wheelchair and Accessibility Sensors:”** Sensors can be used to monitor and improve accessibility features, such as wheelchair ramps and priority seating for individuals with disabilities.
* **REAL TIME TRANSIT INFORMATION PLATFORM**: When applied to the design of an IoT (Internet of Things) platform for real-time information transmission, it can lead to more user-friendly and effective solutions. Here's how you can apply design thinking principles to the design of such a platform:  
    
  **1. “Empathize (Understand the Users): “**   - Identify the key stakeholders and users of the IoT platform. This may include transportation operators, passengers, maintenance personnel, and other relevant parties.  
     - Conduct user research to understand their needs, pain points, and expectations regarding real-time information transmission.  
     - Create user personas to represent different user groups and their unique requirements.  
    
  **2. “Define (Frame the Problem): “**  
     - Clearly define the problem you are solving with the IoT platform. For example, it could be improving passenger information, optimizing transportation operations, or enhancing safety.  
     - Develop a problem statement that guides the design process, such as "How might we provide real-time information that improves the passenger experience on public transportation?"  
    
  **3. “Ideate (Generate Ideas): “**  
     - Brainstorm potential IoT solutions that address the defined problem. Encourage creativity and diverse perspectives from your design team.  
     - Consider various IoT sensors, data sources, and communication technologies that can be used to transmit real-time information.  
     - Explore different ways to visualize and present information to users, such as mobile apps, digital signage, or in-vehicle displays.  
    
  **4. “Prototype (Create Solutions): “**  
     - Build low-fidelity prototypes of the IoT platform to test and refine ideas quickly.  
     - Experiment with different hardware and software components, including IoT sensors, data processing algorithms, and user interfaces.  
     - Gather feedback from users and stakeholders as you iterate on your prototypes.
* **INTEGRATION APPROACH:**  This method ensures that the integration is user-centered, addresses real problems, and results in effective solutions. Here's how to apply design thinking to the integration of IoT into transportation optimization:

**1. “Test (Gather Feedback): “**  
   - Conduct usability testing and gather feedback on the integration prototypes. Engage with stakeholders to understand how the proposed solutions meet their needs and expectations.  
   - Identify any usability issues, technical challenges, or potential roadblocks that may arise during implementation.  
  
**2.** **“Implement (Develop and Deploy the Integration): “**  
   - Based on feedback and insights from testing, proceed with the development of the IoT integration solutions. This may involve creating custom software, deploying IoT sensors, and setting up data analytics infrastructure.  
   - Ensure that the integration aligns with the goals of transportation optimization and that it can be scaled for broader use.  
  
**3.** **“Iterate (Continuous Improvement): “**  
   - The design thinking process is iterative, so continue to gather feedback and make improvements to the IoT integration as it is deployed and used.  
   - Stay open to evolving user needs, technological advancements, and changes in transportation trends.  
  
**4.** **“Scale and Deploy (Rollout): “**  
   - Once the IoT integration has been refined and tested, scale it for wider deployment across the transportation system.  
   - Provide training and support to transportation staff and operators who will use the integrated solutions.  
  
**5.** **“Monitor and Maintain (Ongoing Operations): “**  
   - Continuously monitor the performance of the IoT integration, collect data, and measure its impact on transportation optimization goals.  
   - Regularly maintain and update the integration to ensure it remains effective and aligned with changing needs.

1. **MICROCONTROLLER:**

We had chosen ESP32 micro controller as well as Arduino UNO micro controller.

1. **ESP32 -** The ESP32 is a very versatile System On a Chip (SoC) that can be used as a general purpose microcontroller with quite an extensive set of peripherals including WiFi and Bluetooth wireless capabilities. The high-level option is MicroPython for ESP32. Where the user doesn't even really need to install anything on their computer: anything that can open a serial terminal will do. The Python language itself is much more beginner friendly than the C language used by Arduino and ESP-IDF.
2. **Arduino UNO -** The Arduino IoT Cloud is a platform that allows anyone to create IoT projects, with a user friendly interface, and an all in one solution for configuration, writing code, uploading and visualization. Arduino uses a variant of the C++ programming language. The code is written in C++ with an addition of special methods and functions.
3. **SENSORS:**

**1. “GPS and GNSS Sensors: “** Global Positioning System (GPS) and Global Navigation Satellite System (GNSS) sensors are used to track the real-time location of vehicles. This data is essential for tracking bus or train movements, optimizing routes, and providing accurate arrival time predictions to passengers.  
  
**2. “Vehicle Health Monitoring Sensors: “** IoT sensors can monitor the health of transportation vehicles by collecting data on engine performance, fuel consumption, tire pressure, and other crucial metrics. This data helps in preventive maintenance, reducing breakdowns, and improving overall fleet efficiency.  
  
**3. “Passenger Counting Sensors: “** These sensors use infrared, ultrasonic, or camera technology to count the number of passengers boarding and disembarking vehicles. Passenger data helps transit agencies adjust service frequency and allocate resources based on demand.  
 **4. “Smart Fare Collection Sensors: “** IoT sensors integrated into fare collection systems, such as contactless payment readers or ticket validators, enable seamless, fast, and secure payment processes. They also collect data on fare transactions for analysis.  
 **5. “Environmental Sensors: “** Sensors for measuring environmental factors like air quality, temperature, humidity, and noise levels can be installed in transportation hubs and vehicles. This information can be used to monitor and improve passenger comfort and safety.  
 **6. “Traffic and Intersection Sensors: “** IoT sensors placed at intersections and traffic lights help optimize traffic flow and reduce congestion. This is especially important for buses and other forms of public transportation that operate on city streets.  
 **7. “Security and Surveillance Sensors: “** Cameras and sensors can be used for both passenger safety and security. They can monitor for accidents, suspicious activities, or unauthorized access to restricted areas within transportation facilities.  
 **8. “Energy Consumption Sensors: “** Public transportation agencies can use energy consumption sensors to monitor and optimize the energy usage of their vehicles and facilities, helping to reduce operational costs and environmental impact.  
 **9. “Weather and Road Condition Sensors: “** These sensors provide data on weather conditions and road surfaces, allowing transportation operators to make informed decisions about route changes and adjustments during adverse weather conditions.  
  
**10. “Wi-Fi and Cellular Connectivity Sensors: “** IoT sensors for connectivity enable passengers to access Wi-Fi and cellular networks while on public transportation, improving the passenger experience and enabling real-time communication.  
  
**11. “Seat Occupancy Sensors: “** In the era of COVID-19 and social distancing measures, seat occupancy sensors can help manage capacity and maintain safe distances between passengers.  
  
**12. “Wheelchair and Accessibility Sensors:”** Sensors can be used to monitor and improve accessibility features, such as wheelchair ramps and priority seating for individuals with disabilities.

1. **CONNECTIVITY:**

1. **BLE -** Bluetooth Low Energy is a wireless, low-power personal area network that operates in the 2.4 GHz ISM band. Its goal is to connect devices over a relatively short range. BLE was created with IoT applications in mind, which has particular implications for its design.
2. **WIFI -** WIFI stands for Wireless Fidelity which is a wireless technology standard for wireless Internet access. It is used as a replacement for cable connections and other types of wires.
3. **ZIGBEE -** Zigbee is a wireless protocol that is used to allow Smart Devices such as light bulbs, sockets, plugs, smart locks, motion sensors and door sensors to communicate with each other over a "PAN" (Personal Area Network).
4. **CLOUD:**

**Beeceptor -** Beeceptor is a no code solution for building and hosting a mock server. Beeceptor gives you a dashboard to intercept and inspect all HTTP requests in real time. Simulate higher latencies by introducing delays and timeouts. This helps you validate rarely reachable code paths. Use reverse proxy to quickly switch APIs endpoints. E.g. A/B testing by switching API endpoints or versions without any redeployment. Send hyper-customized responses using the Handlebar template.

1. **PROTOCOL:**
2. **MQTT -** MQTT is a standards-based messaging protocol, or set of rules, used for machine-to-machine communication. Smart sensors, wearables, and other Internet of Things (IoT) devices typically have to transmit and receive data over a resource-constrained network with limited bandwidth.
3. **HTTP -** The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems. This is the foundation for data communication for the World Wide Web (i.e. internet)
4. **AMQP -** Advanced Message Queuing Protocol (AMQP) is an open source published standard for asynchronous messaging by wire. AMQP enables encrypted and interoperable messaging between organizations and applications. The protocol is used in client/server messaging and in IoT device management.
5. **PUBLIC PLATFORM:**

We have planned to display the data in the open source web server or in a suitable application with a highly convenient protocol and safety measures.

1. **PYTHON SCRIPT FOR IOT COMPONENTS USED IN THE PROJECT**

import random  
import time  
  
# Simulate GPS data  
class GPSSensor:  
    def get\_location(self):  
        latitude = random.uniform(40, 45)  
        longitude = random.uniform(-75, -70)  
        return latitude, longitude  
  
# Simulate fuel consumption data  
class FuelSensor:  
    def get\_fuel\_level(self):  
        return random.uniform(0, 100)  
  
# Simulate temperature and humidity data  
class EnvironmentSensor:  
    def get\_temperature(self):  
        return random.uniform(15, 30)  
  
    def get\_humidity(self):  
        return random.uniform(30, 70)  
  
# Simulate ultrasonic sensor and camera  
class UltrasonicCameraSensor:  
    def detect\_obstacle(self):  
        return random.choice([True, False])  
  
# Simulate ticket validator  
class TicketValidator:  
    def is\_valid\_ticket(self):  
        return random.choice([True, False])  
  
# Simulate suspicious activity sensor  
class SuspiciousActivitySensor:  
    def detect\_suspicious\_activity(self):  
        return random.choice([True, False])  
  
# Simulate weather sensor  
class WeatherSensor:  
    def get\_weather\_conditions(self):  
        conditions = ["Sunny", "Rainy", "Cloudy", "Windy"]  
        return random.choice(conditions)  
  
# Main function for Smart Transit Optimization  
def smart\_transit\_optimization():  
    ticket\_validator = TicketValidator()  
    suspicious\_activity\_sensor = SuspiciousActivitySensor()  
    weather\_sensor = WeatherSensor()  
  
    while True:  
        # Check the ticket validator  
        if not ticket\_validator.is\_valid\_ticket():  
            print("Invalid Ticket Detected. Please resolve the issue.")  
  
        # Check for suspicious activity  
        if suspicious\_activity\_sensor.detect\_suspicious\_activity():  
            print("Suspicious Activity Detected. Please take appropriate action.")  
  
        # Get weather conditions  
        weather = weather\_sensor.get\_weather\_conditions()  
        print(f"Weather Condition: {weather}")

# Simulate noise level monitoring  
class NoiseSensor:  
    def get\_noise\_level(self):  
        return random.uniform(40, 90)  
  
# Main function to monitor and optimize smart transit  
def smart\_transit\_optimization():  
    gps = GPSSensor()  
    fuel\_sensor = FuelSensor()  
    environment\_sensor = EnvironmentSensor()  
    ultrasonic\_camera\_sensor = UltrasonicCameraSensor()  
    noise\_sensor = NoiseSensor()  
  
    while True:  
        # Get data from sensors  
        latitude, longitude = gps.get\_location()  
        fuel\_level = fuel\_sensor.get\_fuel\_level()  
        temperature = environment\_sensor.get\_temperature()  
        humidity = environment\_sensor.get\_humidity()  
        obstacle\_detected = ultrasonic\_camera\_sensor.detect\_obstacle()  
        noise\_level = noise\_sensor.get\_noise\_level()

1. **PROGRAM:**
2. **DIAGRAM**

{

  "version": 1,

  "author": "SACS",

  "editor": "wokwi",

  "parts": [

    { "type": "wokwi-esp32-devkit-v1", "id": "esp", "top": -12.8, "left": 180.8, "attrs": {} },

    {

      "type": "wokwi-hc-sr04",

      "id": "ultrasonic1",

      "top": -31.94,

      "left": -1.1,

      "attrs": { "distance": "240" }

    },

    {

      "type": "wokwi-hc-sr04",

      "id": "ultrasonic2",

      "top": -32.14,

      "left": -184.37,

      "attrs": { "distance": "104" }

    },

    {

      "type": "wokwi-led",

      "id": "led1",

      "top": 196.6,

      "left": 95.24,

      "attrs": { "color": "red" }

    }

  ],

  "connections": [

    [ "esp:TX0", "$serialMonitor:RX", "", [] ],

    [ "esp:RX0", "$serialMonitor:TX", "", [] ],

    [ "ultrasonic1:VCC", "esp:VIN", "red", [ "v0" ] ],

    [ "ultrasonic1:GND", "esp:GND.2", "black", [ "v0" ] ],

    [ "ultrasonic1:TRIG", "esp:D13", "green", [ "v0" ] ],

    [ "ultrasonic1:ECHO", "esp:D12", "green", [ "v0" ] ],

    [ "ultrasonic2:GND", "esp:GND.2", "black", [ "v93.61", "h255.99", "v-19.13", "h1.42" ] ],

    [ "ultrasonic2:VCC", "esp:VIN", "red", [ "v111.32", "h288.11", "v-26.93" ] ],

    [ "ultrasonic2:ECHO", "esp:D14", "green", [ "v97.15", "h262.44", "v-53.14" ] ],

    [ "ultrasonic2:TRIG", "esp:D15", "green", [ "v102.11", "h397.16", "v-36.14" ] ],

    [ "led1:C", "esp:GND.1", "black", [ "v7.36", "h195.18", "v-106.29" ] ],

    [ "led1:A", "esp:D22", "green", [ "v0.98", "h178.81", "v-149.51" ] ]

  ],

  "dependencies": {}

}

1. **SKETCH**

#define BLYNK\_TEMPLATE\_ID "TMPL26V4fGv5q"

#define BLYNK\_TEMPLATE\_NAME "Test"

#define BLYNK\_AUTH\_TOKEN "XEHxNF\_Ur1Nt2p7wB5B20dNI1ZUwj34P"

#include <WiFi.h>

#include <WiFiClient.h>

#include <BlynkSimpleEsp32.h>

int duration1 = 0;

int distance1 = 0;

int duration2 = 0;

int distance2 = 0;

int dis1 = 0;

int dis2 = 0;

int dis\_new1 = 0;

int dis\_new2 = 0;

int entered = 0;

int left = 0;

int inside = 0;

#define LED 2

#define PIN\_TRIG1 15

#define PIN\_ECHO1 14

#define PIN\_TRIG2 13

#define PIN\_ECHO2 12

BlynkTimer timer;

char auth[] = BLYNK\_AUTH\_TOKEN;

char ssid[] = "Wokwi-GUEST";   // your network SSID (name)

char pass[] = "";

#define BLYNK\_PRINT **Serial**

long get\_distance1() {

  // Start a new measurement:

  digitalWrite(PIN\_TRIG1, HIGH);

  delayMicroseconds(10);

  digitalWrite(PIN\_TRIG1, LOW);

  // Read the result:

  duration1 = pulseIn(PIN\_ECHO1, HIGH);

  distance1 = duration1 / 58;

  return distance1;

}

long get\_distance2() {

  // Start a new measurement:

  digitalWrite(PIN\_TRIG2, HIGH);

  delayMicroseconds(10);

  digitalWrite(PIN\_TRIG2, LOW);

  // Read the result:

  duration2 = pulseIn(PIN\_ECHO2, HIGH);

  distance2 = duration2 / 58;

  return distance2;

}

void myTimer() {

**Serial**.println("100");

  dis\_new1 = get\_distance1();

  dis\_new2 = get\_distance2();

  if (dis1 != dis\_new1 || dis2 != dis\_new2){

**Serial**.println("200");

    if (dis1 < dis2){

**Serial**.println("Enter loop");

      entered = entered + 1;

      inside = inside + 1;

      digitalWrite(LED, HIGH);

      Blynk.virtualWrite(V0, entered);

      Blynk.virtualWrite(V2, inside);

      dis1 = dis\_new1;

      delay(1000);

      digitalWrite(LED, LOW);

    }

    if (dis1 > dis2){

**Serial**.println("Leave loop");

      left = left + 1;

      inside = inside - 1;

      Blynk.virtualWrite(V1, left);

      Blynk.virtualWrite(V2, inside);

      dis2 = dis\_new2;

      delay(1000);

    }

  }

}

 void setup() {

**Serial**.begin(115200);

  pinMode(LED, OUTPUT);

  pinMode(PIN\_TRIG1, OUTPUT);

  pinMode(PIN\_ECHO1, INPUT);

  pinMode(PIN\_TRIG2, OUTPUT);

  pinMode(PIN\_ECHO2, INPUT);

  Blynk.begin(auth, ssid, pass, "blynk.cloud", 8080);

  timer.setInterval(1000L, myTimer);

}

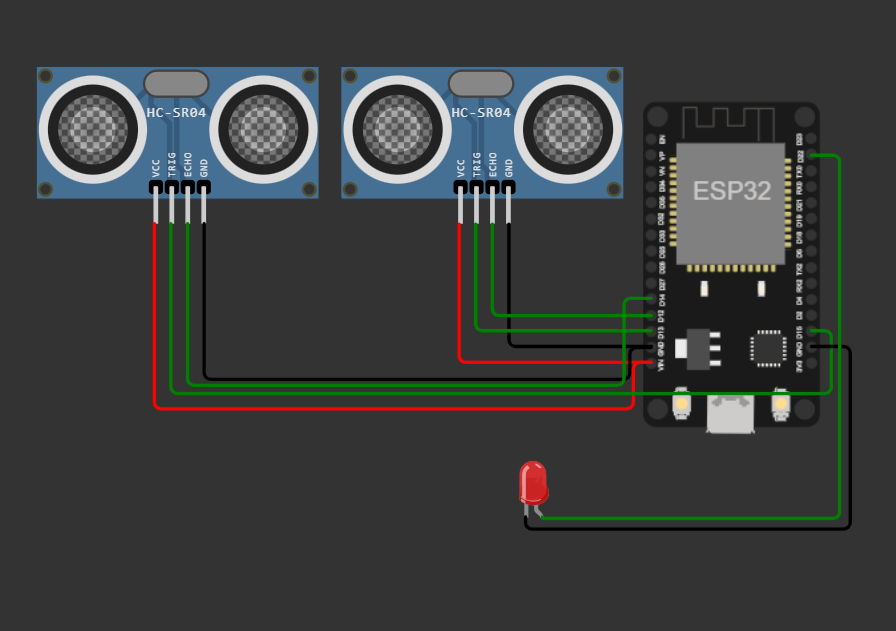
void loop() {

  Blynk.run();

  timer.run();

}

1. **STIMULATOR:**

****

1. **STIMULATOR OUTPUT:**

load:0x3fff0030,len:1156

load:0x40078000,len:11456

ho 0 tail 12 room 4

load:0x40080400,len:2972

entry 0x400805dc

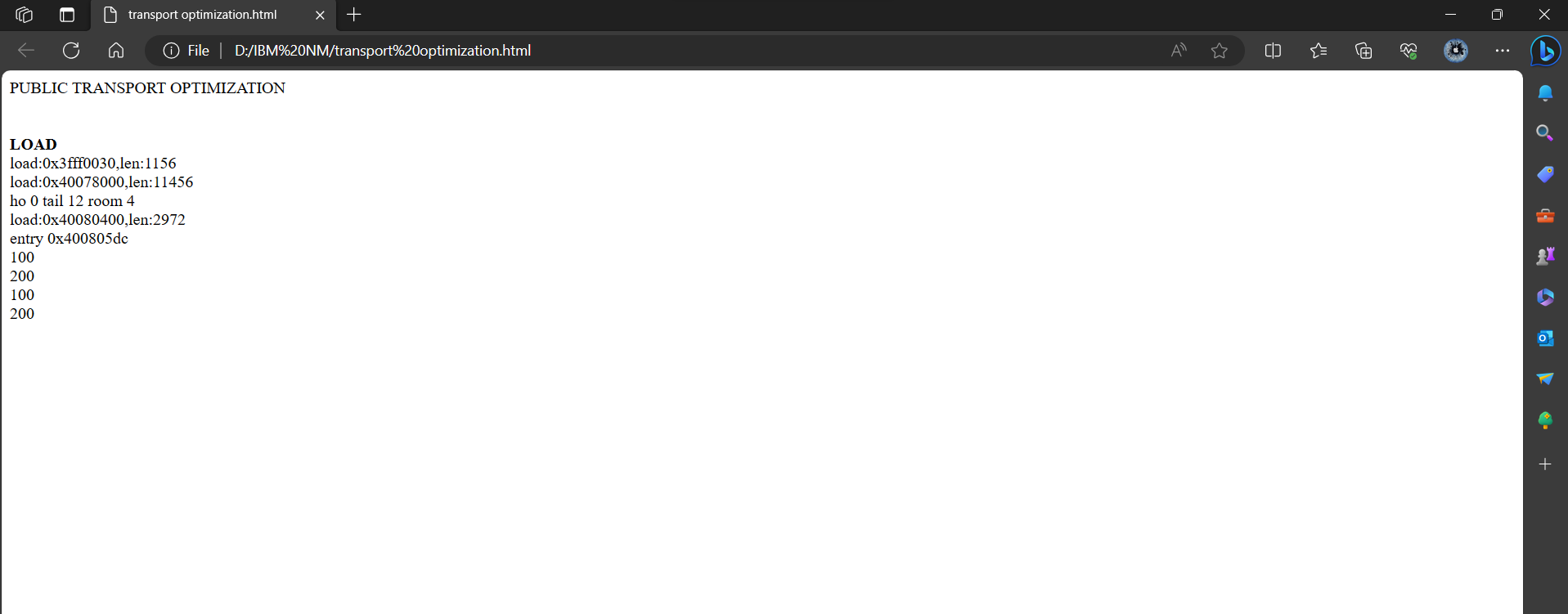
100

200

100

200

1. **OUTPUT IN WEBPAGE:**

****