PUBLIC TRANSPORT OPTIMIZATION

PHASE 5: Documentation and submission.

OBJECTIVE: Objective: To enhance public transportation efficiency and user experience by collecting, analyzing, and providing real-time access to ridership data and accurate arrival time predictions using sensor systems, GPS modules, and a user-friendly mobile application.

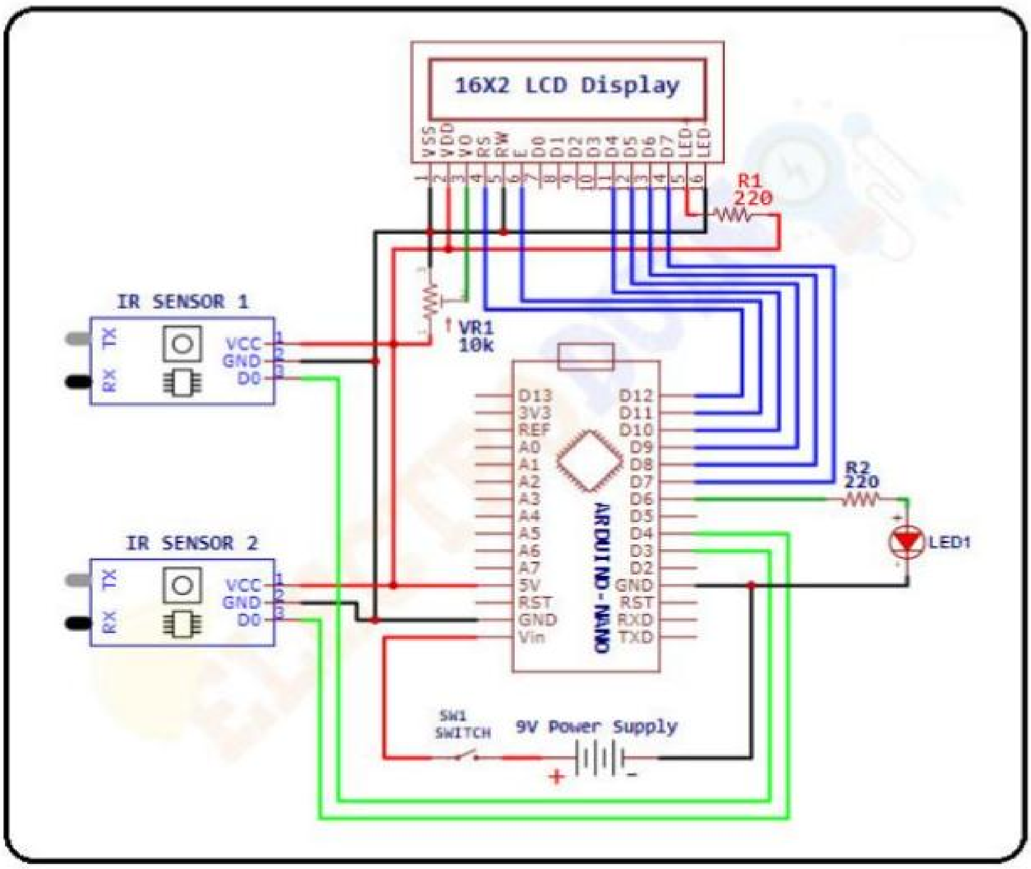
SENSOR SYSTEM DESIGN:

RIDERSHIP MONITORING:

COMPONENTS :

- ARDUINO UNO ATmega328P
- INFRARED (IR) SESNORS (2X)
- LED
- BUZZER
- RESISTOR
- LCD DISPLAY (16 X 2)
- BREADBORD
- JUMPER CABLES
- POWER SOURCE
- PC WITH ARUINO IDE

SYSTEM DESIGN:



PYTHON SCRIPT:

```
# Import the necessary libraries
from pyfirmata import Arduino, util

# Define the serial port of your Arduino (e.g., 'COM8' on Windows)
port = 'COM8'

# Initialize the Arduino board
board = Arduino(port)

# Define the pin numbers for sensors and LED
sensor_1_pin = 2
sensor_2_pin = 3
led_pin = 13  # The built-in LED pin on most Arduino boards

# Set up the pins
```

```
sensor_1 = board.get_pin('d:{}:i'.format(sensor_1_pin))
sensor_2 = board.get_pin('d:{}:i'.format(sensor_2_pin))
led = board.get_pin('d:{}:o'.format(led_pin))

# Initialize variables
peopleInside = 0

# Create an Iterator to handle input events
it = util.Iterator(board)
it.start()

try:
    while True:
        # Read the state of the sensors
        sensor1State = sensor_1.read()
        sensor2State = sensor_2.read()


        if sensor1State == 1 and sensor2State == 0:
            # Object entered the room
            led.write(1)  # Turn on the LED
            board.pass_time(10)  # Wait for 10 seconds
            led.write(0)  # Turn off the LED
            peopleInside += 1  # Increment people count
        elif sensor1State == 0 and sensor2State == 1:
            # Object exited the room
            # You can add code to control the buzzer here
            peopleInside -= 1  # Decrement people count
        print("People Inside:", peopleInside)

except KeyboardInterrupt:
    # Exit the program gracefully on Ctrl+C
    board.exit()
```

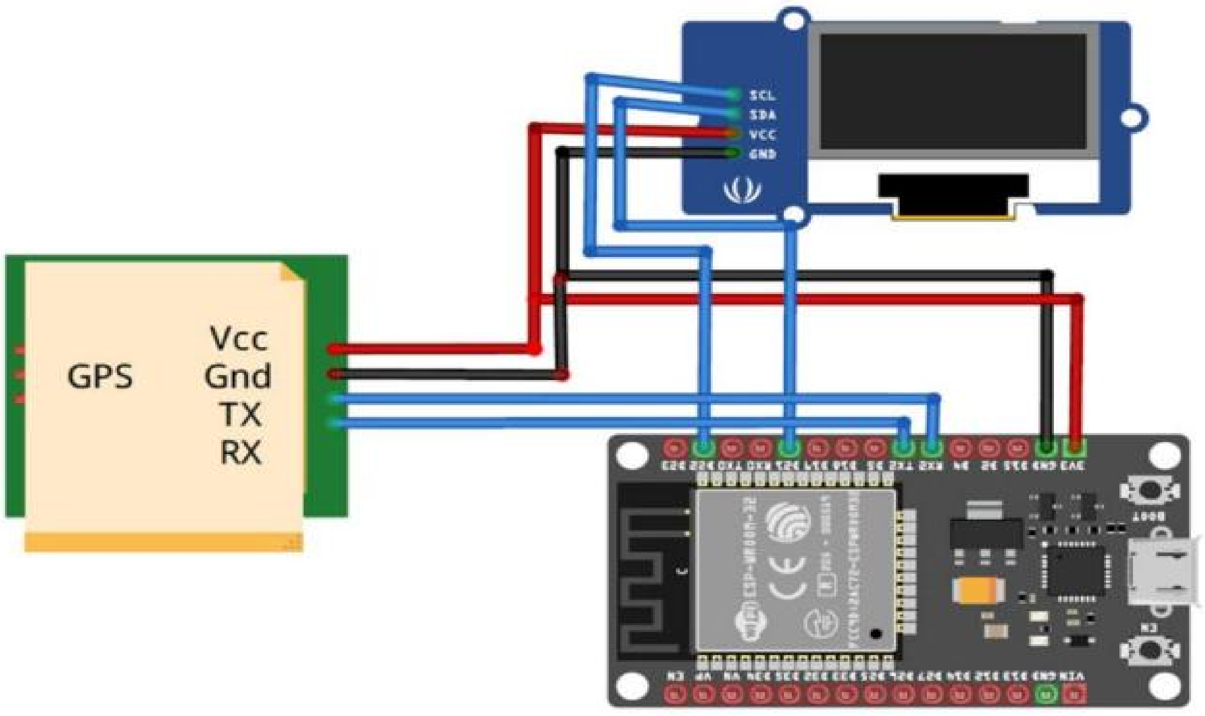
LOCATION TRACKING:

ESP32 GPS tracker- IoT based location tracking system:

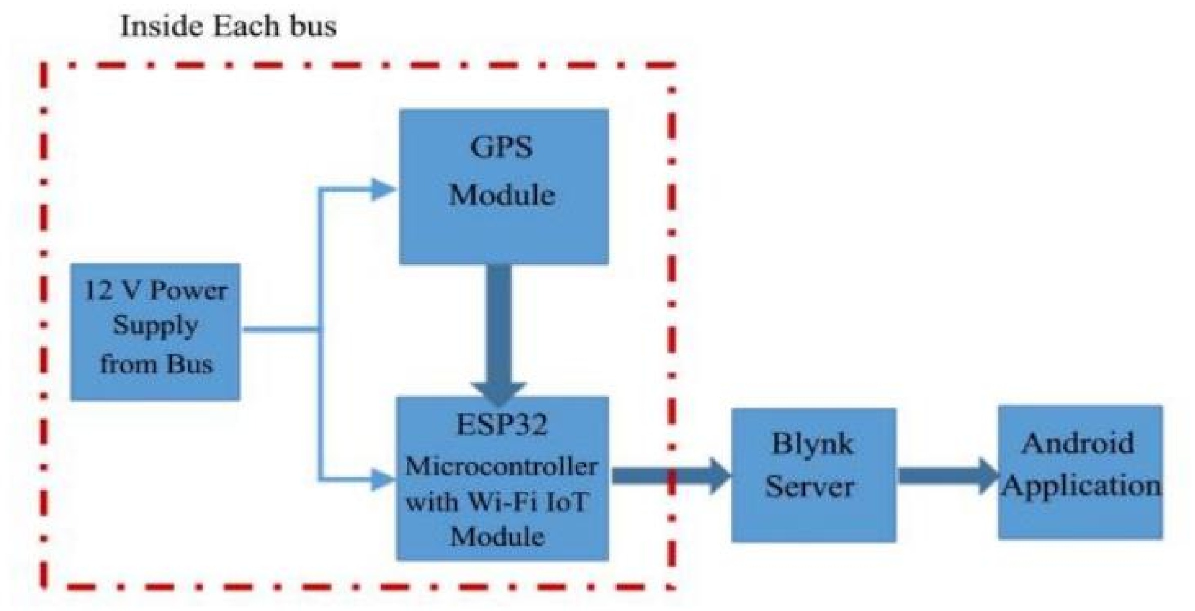
COMPONENTS:

- ESP32
- GPS Module
- OLED Display Module
- Jumper Wires
- Breadboard

SYSTEM DESIGN:



SYSTEM BLOCK DISGRAM:


Inside Each bus
12 V Power Supply from Bus
GPS Module
ESP32 Microcontroller with Wi-Fi IoT Module
Blynk Server
Android Application

STEPS INVOLVED:

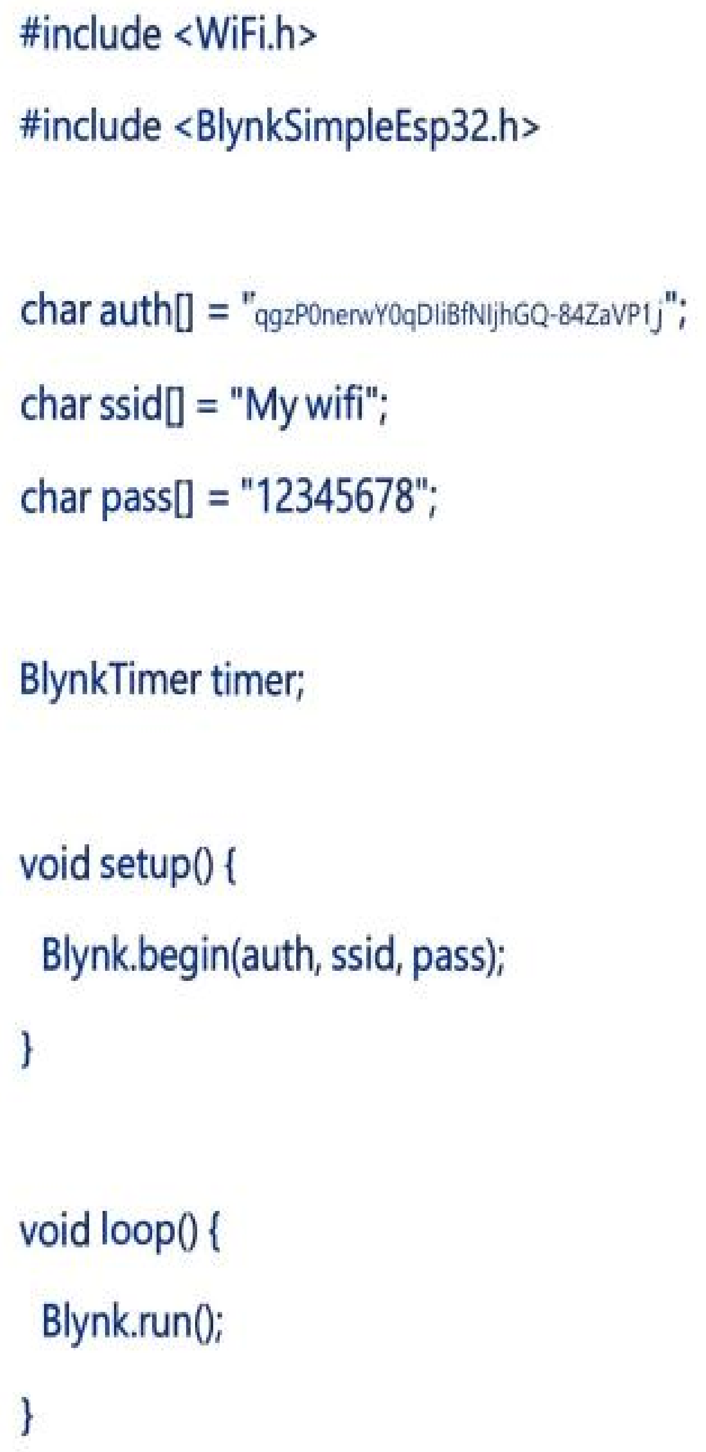
1. **SETTING UP BLYNK SERVER:**Install and configure a Blynk server on a host or a cloud-based server. Obtain the necessary credentials, including the server address, port, and authentication token for your Blynk server. These credentials will be used by your devices to connect to the server.

CONFIGURATION RASPBERRY PI AND ESP32:

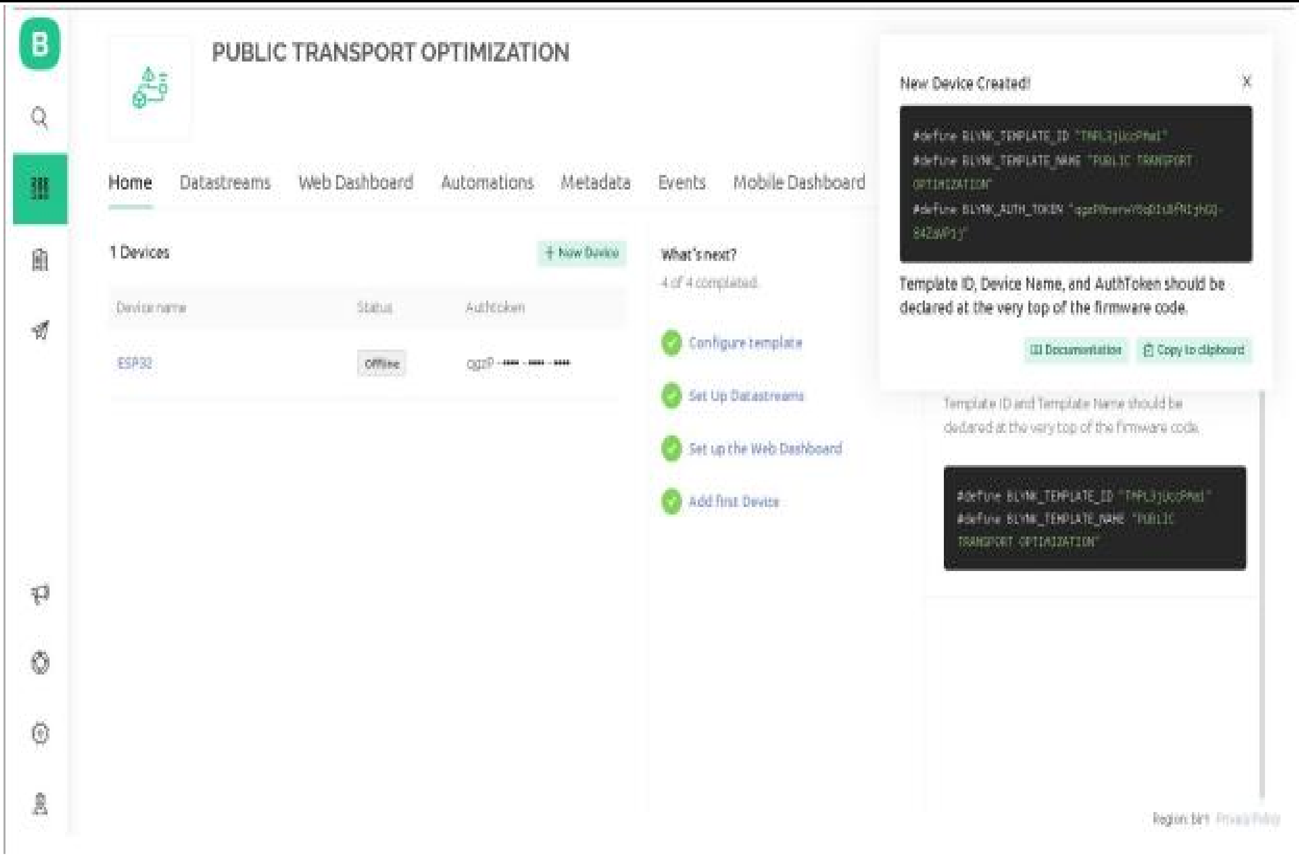
- Install the Blynk library on your Raspberry Pi and ESP32. We can find the Blynk library and documentation for your respective platforms on the Blynk website.
- Write code on your Raspberry Pi and ESP32 to collect GPS and ridership monitoring data and transmit it to the Blynk server using the Blynk library. Use the provided authentication token to connect to your Blynk server.
- Set up the appropriate virtual pins in your Blynk project to receive and display the data sent by **your devices. You can configure widgets in your Blynk project to visualize the data.**

CONNECTION ESTABLISHMENT BETWEEN SENSOR SYSTEM AND BLYNK SERVER:

By using the following code the sensor sytem can be connected to the wifi and can be linked with the Blynk server for data.



```
#include <WiFi.h>
#include <BlynkSimpleEsp32.h>


char auth[] = "qgzP0nerwY0qDIiBfNIjhGQ-84ZaVP1j";
char ssid[] = "My wifi";
char pass[] = "12345678";


BlynkTimer timer;


void setup() {
  Blynk.begin(auth, ssid, pass);
}


void loop() {
  Blynk.run();
}
```


PUBLIC TRANSPORT OPTIMIZATION

Home Datastreams Web Dashboard Automations Metadata Events Mobile Dashboard

1 Devices + New Device

Device name	Status	AuthToken
ESP32	Offline	qgzP - •••• - •••• - ••••

What's next?
4 of 4 completed
✓ Configure template
✓ Set Up Datastreams
✓ Set up the Web Dashboard
✓ Add first Device

New Device Created! X

```
#define BLYNK_TEMPLATE_ID "TMPL3jUccPMd"
#define BLYNK_TEMPLATE_NAME "PUBLIC TRANSPORT OPTIMIZATION"
#define BLYNK_AUTH_TOKEN "qgzP0nerwY0qDIu8fN1jhGQ-84ZaWP1j"
```

Template ID, Device Name, and AuthToken should be declared at the very top of the firmware code.

Documentation Copy to clipboard

Template ID and Template Name should be declared at the very top of the firmware code.

```
#define BLYNK_TEMPLATE_ID "TMPL3jUccPMd"
#define BLYNK_TEMPLATE_NAME "PUBLIC TRANSPORT OPTIAIZATION"
```

Region: blr1 Privacy Policy

DATA FEEDING FROM GPS TO BLYNK SERVER:

First install blynk python library in ESP32 board using following code,

" pip install blynk-library-python "

Then load the following code to the ESP32 board.

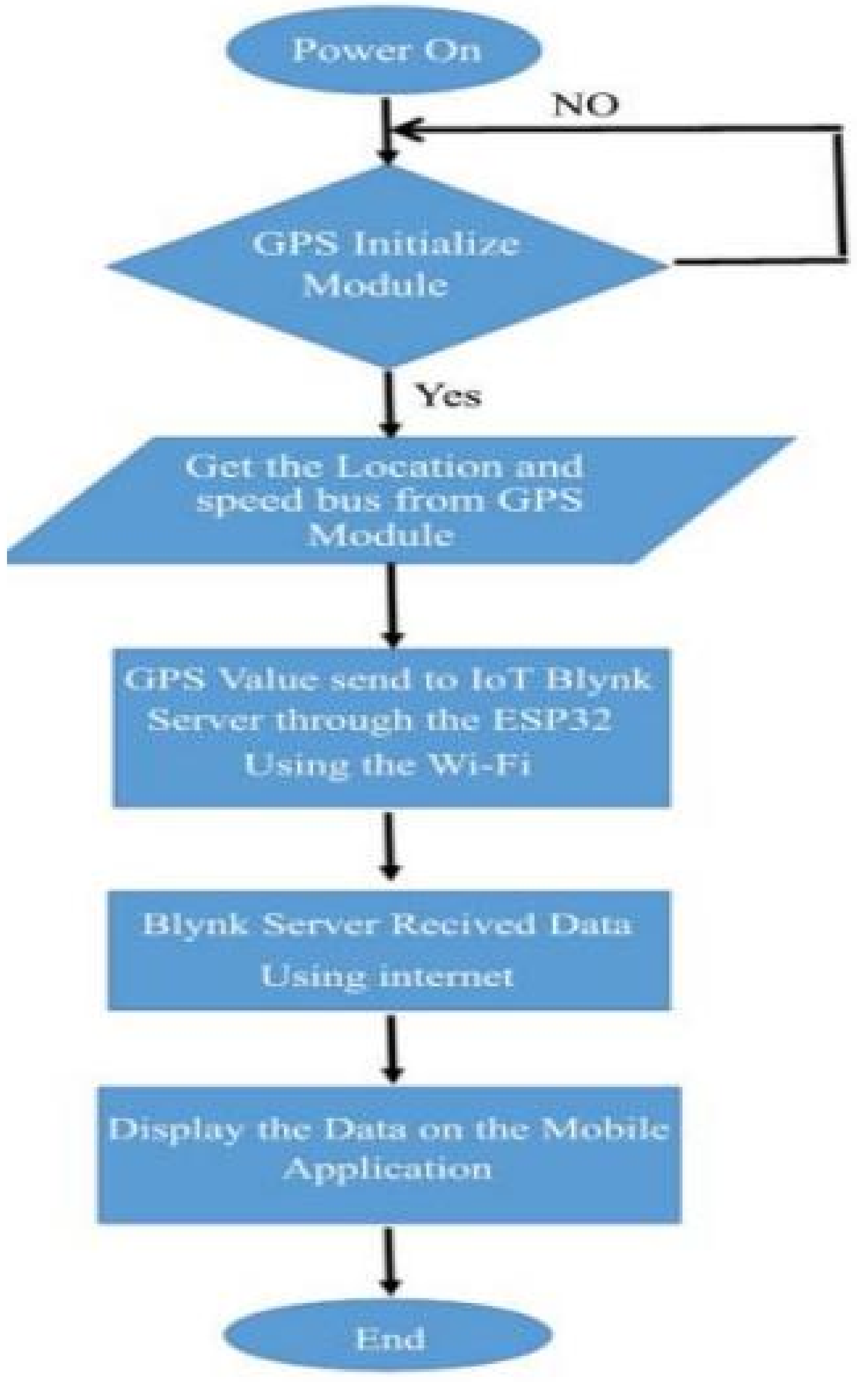
```
# Import the necessary libraries
from pyfirmata import Arduino, util

# Define the serial port of your Arduino (e.g., 'COM8' on Windows)
port = 'COM8'
```

```
# Initialize the Arduino board
board = Arduino(port)

# Define the pin numbers for sensors and LED
sensor_1_pin = 2
sensor_2_pin = 3
led_pin = 13  # The built-in LED pin on most Arduino boards

# Set up the pins
sensor_1 = board.get_pin('d:{}:i'.format(sensor_1_pin))
sensor_2 = board.get_pin('d:{}:i'.format(sensor_2_pin))
led = board.get_pin('d:{}:o'.format(led_pin))

# Initialize variables
peopleInside = 0

# Create an Iterator to handle input events
it = util.Iterator(board)
it.start()

try:
    while True:
        # Read the state of the sensors
        sensor1State = sensor_1.read()
        sensor2State = sensor_2.read()
        if sensor1State == 1 and sensor2State == 0:
            # Object entered the room
            led.write(1)  # Turn on the LED
            board.pass_time(10)  # Wait for 10 seconds
            led.write(0)  # Turn off the LED
            peopleInside += 1  # Increment people count
        elif sensor1State == 0 and sensor2State == 1:
            # Object exited the room
            # You can add code to control the buzzer here
            peopleInside -= 1  # Decrement people count
        print("People Inside:", peopleInside)

except KeyboardInterrupt:
    # Exit the program gracefully on Ctrl+C
    board.exit()
```

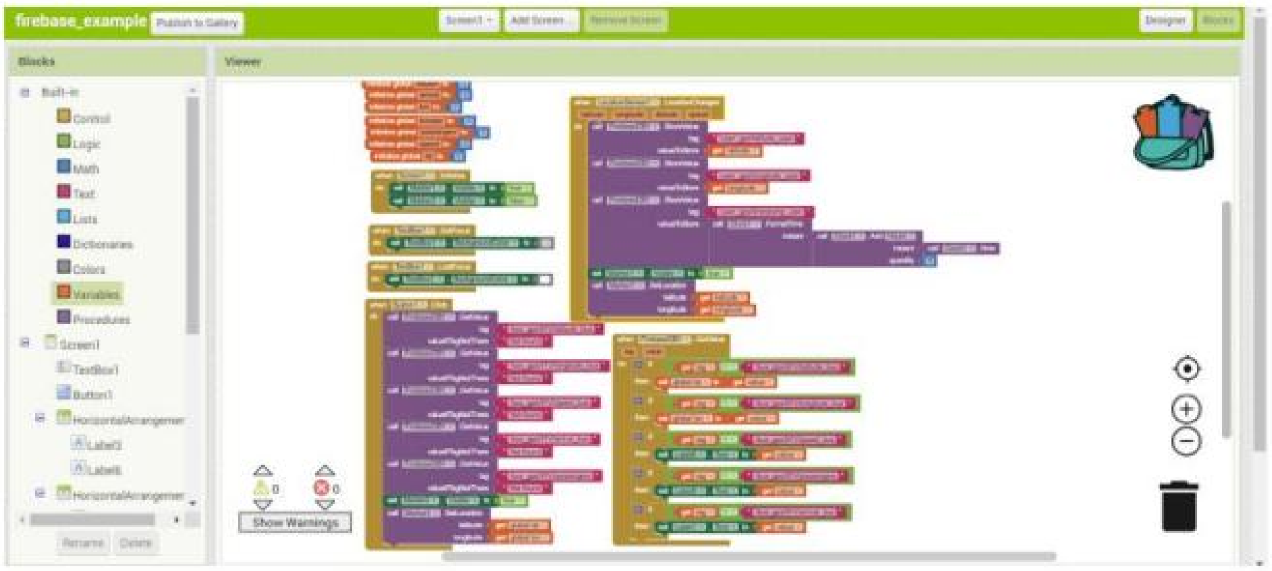
PLATFORM DESIGN USING MIT APP INVENTOR:

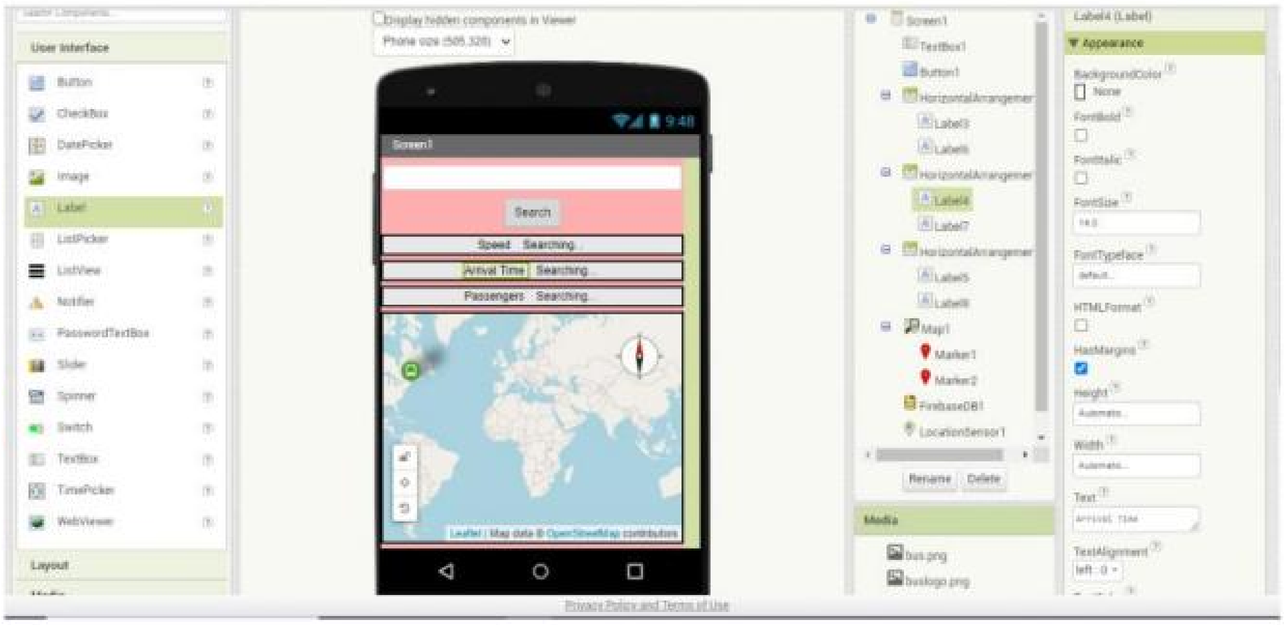
➢ Design the app interface: Use the visual designer to create the user interface for your public platform. You can design screens for viewing GPS data and ridership information.

➢ Add Blynk components: In MIT App Inventor, add Blynk components that allow your app to communicate with the Blynk server. Configure these components with the server address, port, and authentication token.


Power On
NO
GPS Initialize Module
Yes
Get the Location and speed bus from GPS Module
GPS Value send to IoT Blynk Server through the ESP32 Using the Wi-Fi
Blynk Server Recived Data Using internet
Display the Data on the Mobile Application
End

REAL-TIME TRANSIT INFORMATION PLATFORM DESGN USING MIT APP INVENTOR:


firebase_example
Publish to Gallery
Screen1
Add Screen...
Remove Screen
Designer
Blocks
Viewer
Built-in
Control
Logic
Math
Text
Lists
Dictionaries
Colors
Variables
Procedures
Screen1
TextBox1
Button1
HorizontalArrangement
Label3
Label6
Rename
Delete
Show Warnings


User Interface
Button
CheckBox
DatePicker
Image
Label
ListPicker
ListView
Notifier
PasswordTextBox
Slider
Spinner
Switch
TextBox
TimePicker
WebViewer
Layout
Display hidden components in Viewer
Phone size (505,320)
9:48
Screen1
Search
Speed Searching...
Arrival Time Searching...
Passengers Searching...
Leaflet | Map data © OpenStreetMap contributors
Screen1
TextBox1
Button1
HorizontalArrangement
Label3
Label6
Label4
Label7
Label5
Label8
Map1
Marker1
Marker2
FirebaseDB1
LocationSensor1
Rename
Delete
Media
bus.png
buslogo.png
Label4 (Label)
Appearance
BackgroundColor
None
FontBold
FontItalic
FontSize
14.0
FontTypeface
default...
HTMLFormat
HasMargins
Height
Automatic...
Width
Automatic...
Text
Arrival Time
TextAlignment
left : 0
Privacy Policy and Terms of Use