INTERNET OF THINGS

IBM – NAAN MUDHALVAN COURSE

PHASE 4 : DEVELOPMENT PART – 2

INTRODUCTION:

In the modern age, real-time transit information is pivotal for an efficient public transportation system. With the advent of IoT and web technologies, creating a responsive and real-time bus tracking system has become feasible. This project focuses on harnessing these technologies to create a simulated, yet dynamic system that can be implemented in real-world scenarios. By integrating physical simulations, firmware, databases, and web platforms, this solution aims to streamline public transportation for all stakeholders involved.

WOKWI SIMULATION:

Purpose of Simulation:

Simulation helps to:

• Test firmware without real hardware.
• Find and fix errors in a controlled setting.
• Conserve time and resources during development.

Setting Up the Simulation with Wokwi:

Wokwi doesn't support RFID simulations. Instead, we use buttons to emulate RFID tag reads.

Connections:

1. ESP32 to OLED Display:
    - OLED's GND, VCC, SDA, and SCL are connected to ESP32's respective pins.
2. ESP32 to Buttons:
    - Each button's terminal is connected to specific ESP32 pins and GND.

Configuring Virtual Components:

• Buttons imitate the RFID tag read.
• The OLED display communicates via I2C pins.
• We use ESP32 in place of the unavailable ESP8266 NodeMCU.

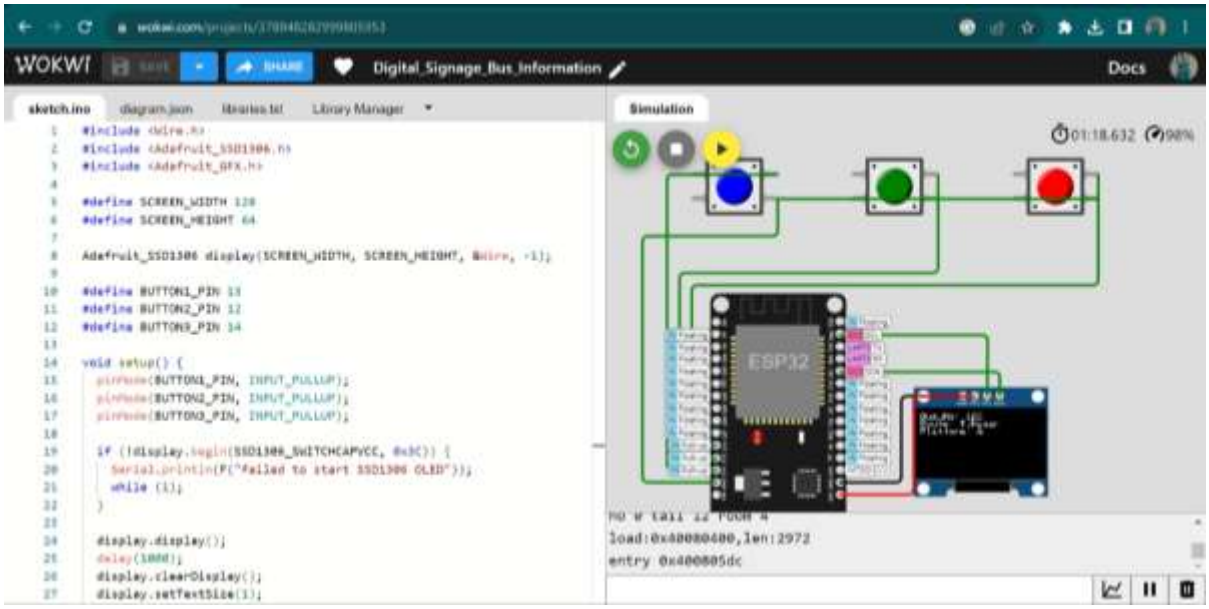
Initial State Definition:

Virtual switches/buttons represent buses, with each button press treated as an RFID read.

Firmware in Simulation:

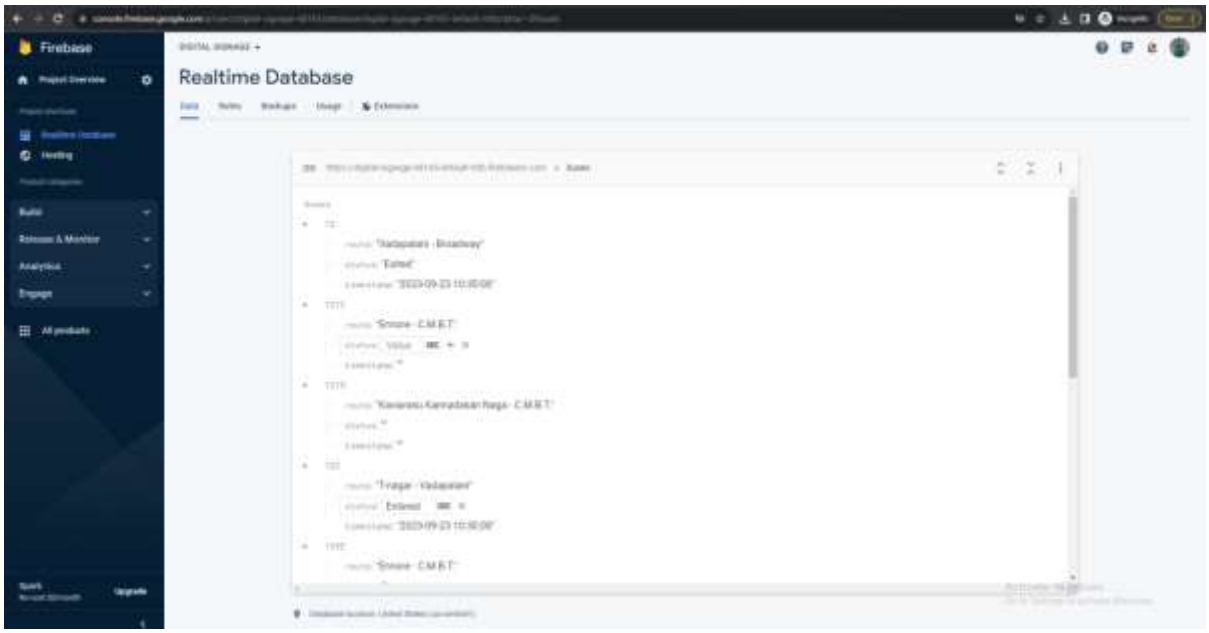
Using Wokwi's Code Editor:

• The code initializes the OLED and maps button presses to specific bus details, reflecting "Entry" or "Exit" on the OLED.



REALTIME DATABASE DEVELOPMENT:

The core of our project is the real-time data that powers our system. Using Firebase's Real-time Database, we ensured instantaneous data updates, high reliability, and low latency.



Integration with Firmware:

The firmware, designed for ESP32, communicates directly with Firebase. Whenever an RFID (or in our simulation, a button) is detected, the firmware sends data to the Firebase database, updating the status of the respective bus and logging the timestamp. This real-time data flow ensures that all subsequent platforms, like the webpage, get the most recent information instantly.

```
#include <FirebaseESP32.h>
//... other includes

FirebaseData firebaseData;

//... Setup functions

void updateFirebase(String busNo, String route, String platform) {
  String path = "/buses/" + busNo;
  Firebase.setString(firebaseData, path + "/route", route);
  //... Other Firebase set operations
}
```

Integration with Webpage:

Our web application leverages Firebase's JavaScript SDK. It listens for changes in the database and immediately reflects these changes on the webpage, giving users up-to-date bus statuses without any manual refreshes. This ensures that users always have real-time data at their fingertips.

```
let busesRef = firebase.database().ref('buses');

busesRef.on('value', (snapshot) => {
   const data = snapshot.val();
   // Logic to update the webpage based on data
});
```

Firmware code:

The final code after integrating the firebase database is

```
#include <Wire.h>
#include <Adafruit_SSD1306.h>
#include <Adafruit_GFX.h>
#include <WiFi.h>
#include <FirebaseESP32.h>

// Firebase configuration
#define FIREBASE_HOST "  https://digital-signage-68165-default-
rtdb.firebaseio.com"
#define FIREBASE_AUTH
```

```
"S1TJvEP7KMP2XB2fOYdTvHQdHgliEz2osjnGHcJn"
#define WIFI_SSID " POCOM3"
#define WIFI_PASSWORD "1234"

FirebaseData firebaseData;

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -
1);

#define BUTTON1_PIN 13
#define BUTTON2_PIN 12
#define BUTTON3_PIN 14

void setup() {
  pinMode(BUTTON1_PIN, INPUT_PULLUP);
  pinMode(BUTTON2_PIN, INPUT_PULLUP);
  pinMode(BUTTON3_PIN, INPUT_PULLUP);

  // Setup display
  if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    Serial.println(F("Failed to start SSD1306 OLED"));
    while (1);
  }
  display.display();
  delay(1000);
  display.clearDisplay();
  display.setTextSize(1);

  // Connect to Wi-Fi
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.print(".");
  }
  Serial.println();

  // Connect to Firebase
  Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
}

void loop() {
```

```
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(SSD1306_WHITE);

  if (!digitalRead(BUTTON1_PIN)) {
    display.setCursor(0, 0);
    display.println("Bus_No: 12C");
    display.println("Route: T.Nagar");
    display.println("Platform: A");
    updateFirebase("12C", "T.Nagar", "A");
  } else if (!digitalRead(BUTTON2_PIN)) {
    display.setCursor(0, 0);
    display.println("Bus_No: 7M");
    display.println("Route: Central");
    display.println("Platform: B");
    updateFirebase("7M", "Central", "B");
  } else if (!digitalRead(BUTTON3_PIN)) {
    display.setCursor(0, 0);
    display.println("Bus_No: 72");
    display.println("Route: Vadapalani");
    display.println("Platform: C");
    updateFirebase("72", "Vadapalani", "C");
  }

  display.display();
  delay(2000);
}

void updateFirebase(String busNo, String route, String platform) {
  // Update Firebase with bus information
  String path = "/buses/" + busNo;
  Firebase.setString(firebaseData, path + "/route", route);
  Firebase.setString(firebaseData, path + "/platform", platform);
}
```

Webpage Development:

The webpage serves as the user-facing platform, presenting real-time bus data in an organized, user-friendly manner. It segregates buses based on their 'Entered' or 'Exited' statuses and provides essential details like routes and timestamps.

Integration with Backend Database

To ensure real-time updates on the webpage, we integrated it directly with our Firebase backend. Using asynchronous calls, the webpage fetches real-time data and

displays it instantaneously. This dynamic integration ensures that even if a bus's status changes, the update is immediately visible to users.

```
let enteredBusesList = document.getElementById("enteredBuses");
//... Accessing other DOM elements

busesRef.on('value', (snapshot) => {
   const data = snapshot.val();
   // Logic to display data in "enteredBusesList" and other elements
});
```

HTML CODE:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Bus Information</title>
    <link ref="stylesheet" type="text/css"
href="C:\Users\Mahendran\OneDrive\Desktop\IOT PROJ\image.css">

  </head>
  <style>
  body{
    background-image:url(https://img.freepik.com/free-photo/hotel-reception-
unfocused_1203-1209.jpg);
    background-repeat: no-repeat;
    background-size: cover;
  }
  .container {
    display: flex;
    justify-content: space-between;
  }


  #date, #time {
    font-size: 50px;
  }

  h1 {
    font-size: 60px;
  }

  .table-container {
    width: 45%;
  }
```

```
  .table-container.left {
    float: left;
    margin-right: 20px;
  }

  .table-container.right {
    float: right;
    margin-left: 20px;
  }

  table {
    border-collapse: collapse;
    width: 100%;
  }

  th, td {
    padding: 8px;
    text-align: left;
    border-bottom: 1px solid #a09b9b;
  }

  th {
    background-color: #b4afaffe;
  }

  .table-container h2 {
    font-size: 50px;
  }
  h1{
    font-family: 'Trebuchet MS', 'Lucida Sans Unicode', 'Lucida Grande',
'Lucida Sans', Arial, sans-serif;
  }
  h2{
    font-family:'Trebuchet MS', 'Lucida Sans Unicode', 'Lucida Grande',
'Lucida Sans', Arial, sans-serif ;
  }
  table{
    font-family: 'Trebuchet MS', 'Lucida Sans Unicode', 'Lucida Grande',
'Lucida Sans', Arial, sans-serif;
    font-size:20px;

  }
  script{
    font-family: 'Trebuchet MS', 'Lucida Sans Unicode', 'Lucida Grande',
'Lucida Sans', Arial, sans-serif;
  }
  </style>
```

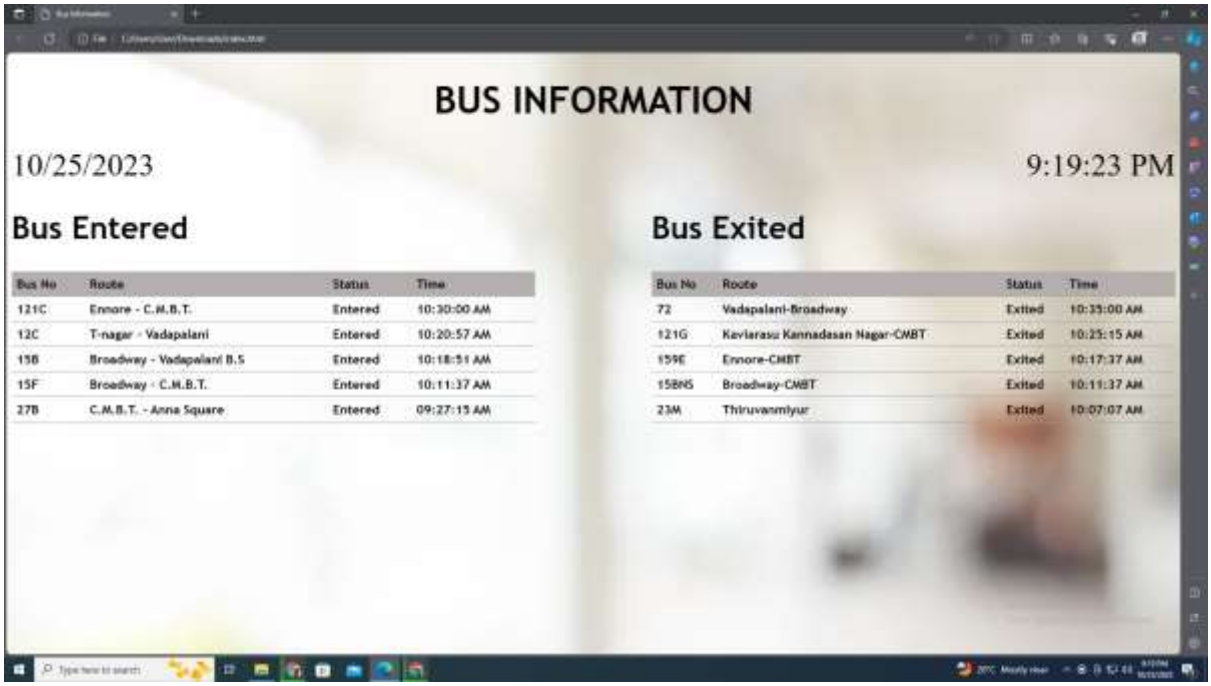
```
<body>
  <center><h1>BUS INFORMATION</h1></center>
  <div class="container">
    <span id="date"></span>
    <span id="time"></span>
  </div>
  <div class="table-container left">
    <h2>Bus Entered</h2>
    <b><table>
      <tr>
        <b><th>Bus No</th></b>
        <b><th>Route</th></b>
        <b><th>Status</th></b>
        <b><th>Time</th></b>
      </tr>
      <tr>
        <b><td>121C</td></b>
      <b><td>Ennore - C.M.B.T.</td></b>
      <b><td>Entered</td></b>
      <b><td> 10:30:00 AM</td></b>
      </tr>
      <tr>
      <b><td>12C</td></b>
      <b><td>T-nagar - Vadapalani</td></b>
      <b><td>Entered</td></b>
      <b>  <td> 10:20:57 AM</td></b>
      </tr>
      <tr>
      <b><td>15B</td></b>
      <b><td>Broadway - Vadapalani B.S</td></b>
      <b><td>Entered</td></b>
      <b><td> 10:18:51 AM</td></b>
      </tr>
      <tr>
      <b><td>15F</td></b>
      <b><td>Broadway - C.M.B.T.</td></b>
      <b><td>Entered</td></b>
      <b><td> 10:11:37 AM</td></b>
      </tr>
      <tr>
      <b><td>27B</td></b>
      <b> <td>C.M.B.T. - Anna Square</td></b>
      <b><td>Entered</td></b>
      <b><td> 09:27:15 AM</td></b>
      </tr>
    </table></b>
  </div>
```

```
  <div class="table-container right">
    <h2>Bus Exited</h2>
    <b><table>
      <tr>
      <b><th>Bus No</th></b>
    <b><th>Route</th></b>
  <b><th>Status</th></b>
<b><th>Time</th></b>
      </tr>
      <tr>
      <b><td>72</td></b>
    <b><td>Vadapalani-Broadway</td></b>
  <b><td>Exited</td></b>
<b> <td>10:35:00 AM</td></b>
      </tr>
      <tr>
      <b><td>121G</td></b>
    <b><td>Kaviarasu Kannadasan Nagar-CMBT</td></b>
  <b><td>Exited</td></b>
<b><td>10:25:15 AM</td></b>
      </tr>
      <tr>
      <b><td>159E</th></b>
      <b><td>Ennore-CMBT</th></b>
      <b><td>Exited</th></b>
      <b><td>10:17:37 AM</th></b>
      </tr>
      <tr>
      <b><td>15BNS</th></b>
      <b><td>Broadway-CMBT</th></b>
      <b><td>Exited</th></b>
      <b><td>10:11:37 AM</th></b>
      </tr>
      <tr>
      <b><td>23M</th></b>
      <b> <td>Thiruvanmiyur</th></b>
      <b><td>Exited</th></b>
      <b><td>10:07:07 AM</th></b>
      </tr>
    </table></b>
  </div>

  <script>
    // create a function to update the date and time
    function updateDateTime() {
      // create a new `Date` object
      const now = new Date();
```

```
        // get the current date and time as strings
        const currentDate = now.toLocaleDateString();
        const currentTime = now.toLocaleTimeString();

        // update the `textContent` property of the `span` elements with the
respective ids
        document.querySelector('#date').textContent = currentDate;
        document.querySelector('#time').textContent = currentTime;
      }

      // call the `updateDateTime` function every second
      setInterval(updateDateTime, 1000);
    </script>

    <!-- Firebase App (the core Firebase SDK) -->
    <script src="https://www.gstatic.com/firebasejs/8.0.1/firebase-
app.js"></script>
    <!-- Add Firebase products that you want to use -->
    <script src="https://www.gstatic.com/firebasejs/8.0.1/firebase-
database.js"></script>

    <script>
        // Your Firebase web app's configuration
        var firebaseConfig = {
            apiKey: "AIzaSyCIT03heHx3ib6Vcyw9wk7xk_uJlCwzQMY",
            authDomain: "digital-signage-68165.firebaseapp.com",
            databaseURL: "https://digital-signage-68165-default-
rtdb.firebaseio.com",
            projectId: "digital-signage-68165",
            storageBucket: "digital-signage-68165.appspot.com",
            messagingSenderId: "797024346735",
            appId: "1:797024346735:web:ba9d0b910677923546239a"
        };
        // Initialize Firebase
        firebase.initializeApp(firebaseConfig);
        import { initializeApp } from "firebase/app";
import { getDatabase } from "firebase/database";

// TODO: Replace the following with your app's Firebase project configuration
// See: https://firebase.google.com/docs/web/learn-more#config-object
const firebaseConfig = {
  // ...
  // The value of `databaseURL` depends on the location of the database
  databaseURL: "https://digital-signage-68165-default-rtdb.firebaseio.com",
};

// Initialize Firebase
const app = initializeApp(firebaseConfig);
```

```
// Initialize Realtime Database and get a reference to the service
const database = getDatabase(app);


        let enteredBusesList = document.getElementById("enteredBuses");
        let exitedBusesList = document.getElementById("exitedBuses");

        // Get reference to buses node in the Firebase Real-time Database
        let busesRef = firebase.database().ref('buses');

        busesRef.on('value', (snapshot) => {
            const data = snapshot.val();
            // Clear lists
            enteredBusesList.innerHTML = '';
            exitedBusesList.innerHTML = '';

            for (let bus in data) {
                let listItem = document.createElement("li");
                listItem.textContent = `Bus No: ${bus}, Route:
${data[bus].route}, Timestamp: ${data[bus].timestamp}`;

                if (data[bus].status === "Entered") {
                    enteredBusesList.appendChild(listItem);
                } else if (data[bus].status === "Exited") {
                    exitedBusesList.appendChild(listItem);
                }
            }
        });
        </script>

  </body>
</html>
```

WEB PAGE:



FINAL RESULT:

The culmination of the project yields a dynamic system, fusing Wokwi simulations, Firebase's real-time backend, and a responsive frontend. It exemplifies the potential of integrating state-of-the-art web technologies with traditional transportation systems for smarter public transport solutions.