DEVELOPMENT PART-1 PHASE-3

PUBLIC TRANSPORTATION AND OPTIMIZATION

|  |  |
| --- | --- |
| DATE | 18/10/2023 |
| PROJECT NAME | PUBLIC TRANSPORTATION AND OPTIMIZATION |

INTRODUCTION:

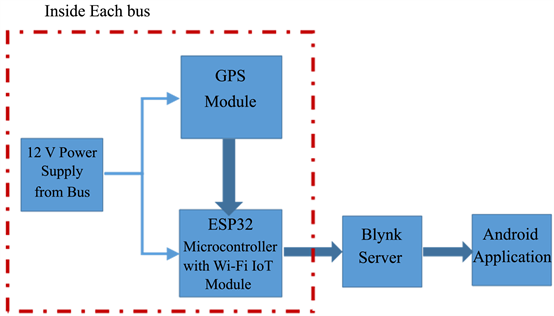
Public transportation refers to shared passenger transportation services such as buses, trolleybuses, trains, ferries, and expresses transportation like the metro

[[1](https://www.scirp.org/journal/paperinformation.aspx?paperid=112889#ref1)]. Intelligent Transportation Systems (ITS) has a subsystem called Smart Public Transportation (SPT). It can intelligently monitor public transit networks to ensure their operation and to provide clients with information on excursions and system operating conditions.

[[2](https://www.scirp.org/journal/paperinformation.aspx?paperid=112889#ref2)]. Smart public Design of an intelligent real-time Public Transportation Monitoring System based on IoT transport systems, depending on several technologies, allows SPTS to retrieve data from multiple sensor systems and to manage and control the transportation network. There are many innovative technologies, which promoted the development and implementation of smart public transport systems, such as Geographical Information Systems (GIS), Automatic Vehicle Location Systems (AVLS) and Traveler Information Systems.

**Proposed System Block Diagram:**

The prototype of the proposed system is in figure. It consists of an Android application designed for users who want real-time information about the buses. The app will display information about buses such as real-time location on Google Maps, speed, distance, and arrival time of each bus. The proposed system includes an ESP32 with a Wi-Fi built-in module, a GPS module, and an Android app connected to the server.



**Implement of the Proposed System:**

The prototype of smart public transportation is in Figure. It consists of two parts: The first part is the communication unit, which includes a GPS module and an ESP32 micro-controller with a Wi-Fi built-in module. This unit is used in the public transportation system for vehicle monitoring and tracking. With the help of the GPS module, it can determine the current position and calculate the speed of buses. The GPS data is transferred to the Blynk server with the help of the Wi-Fi module for storage and analysis. Then it is displayed on the mobile phone application. The second part is the mobile application. The android application gets data from the Blynk server and provides the required data to the user based on the information provided in the android application.

**ESP32 Microcontroller:**

The ESP32 is a microcontroller with a Wi-Fi module, an open-source IoT platform that is characterized by low-cost and low-power system-on-a-chip (SOC). An ESP32 has a dual-core structure and internal modules such as Wi-Fi, Bluetooth, and many Peripheral Interfaces such as IR, SPI, CAN, Ethernet, and temperature sensors . The specifications of the ESP32 are given in Table

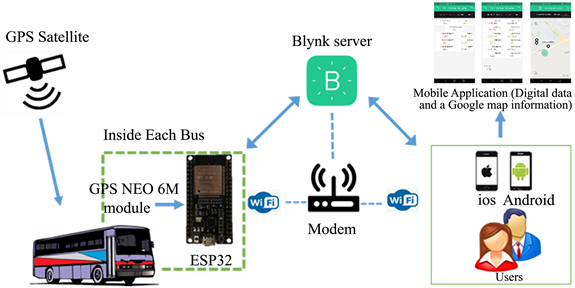


FIGURE: Prototype of smart public transportation architecture.

**Table** . ESP32 specification.

|  |  |
| --- | --- |
| CPU | Tensilica Xtensa LX6 32 bit Dual Core at 160/240 MHz |
| SRAM | 520 KB |
| FLASH | 2 MB (max. 64 MB) |
| Voltage | 2.2 V to 3.6 V |
| Operating Current | 80 mA average Free |
| Programmable | Free (C, C++, Lua, etc.) |
| Open Source | Yes |
| Wi-Fi | 802.11 b/g/n |
| Bluetooth | 4.2 BR/EDR + BLE |
| UART | 3 |
| GPIO | 32 pins |

**GPS Module:**

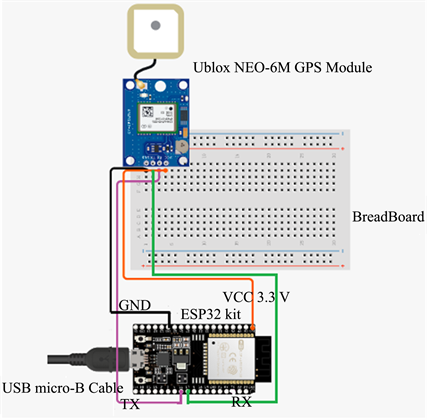
GPS (Global Position System) used for positioning and tracking buses based on satellite communication. GPS satellites cover the entire earth at all times. To get accurate GPS location data, there should be a minimum of three satellites. The NEO-6M GPS module used in the proposed system is small and works on very low power, making it ideal for tracking applications. The GPS module operates at 3.3 V, as a result, powered by connecting the GPS module to the 3.3 V pin of the ESP32. respectively shown in figure.

**Mobile Application:**

The Android application is critical to this system’s success. It assists passengers by providing information about the bus they need to take. It works as a link between the user and the server. In this system, the android application is designed by using the Blynk platform. The Blynk platform is an IoT platform, offering a way to design wonderful applications in minutes for Android and iOS smartphones. It can be used to manage different controllers, such as Raspberry Pi, ESP8266, ESP32, etc. The Blynk platform’s architecture includes Blynk libraries, Blynk server, and Blynk apps [[22](https://www.scirp.org/journal/paperinformation.aspx?paperid=112889#ref22)]. The Android app shown in figure. includes a map that will show the passenger’s location, as well as track the present location of the bus. It also provides the estimated time for the bus to arrive, the speed of the moving bus, and displays the nearest bus to the user by calculating the distance between the passenger and the bus location.

**Distance Calculation:**

The Haversine formula was adopted to calculate the distance that will appear in the Android app. It calculates the distance between the passenger and the bus location using the latitude and longitude of the bus and the passenger who is at home, work, or at the bus stop. The following equations can be used to calculate the distance [[23](https://www.scirp.org/journal/paperinformation.aspx?paperid=112889#ref23)].



**Figure:**GPS module to ESP32 connection.

a=sin2(Δφ/2)+cosφ1⋅cosφ2⋅sin2(Δλ/2)

c=2⋅atan2(a−−√⋅1−a−−−−√)

d=R⋅c

where: (“φ”) = latitude;

(“λ”) = longitude;

(“∆φ”) = latitude 2 − latitude 1;

(“∆λ”) = longitude 2 − longitude 1;

R = earth radius (6371 meters);

d = distance between the two location.

**Arrival Time Calculation:**

The arrival time of each bus, which will appear in the Android app, is calculated based on the following equation.

t=d/s∗60

where:

t = arrival time of bus.

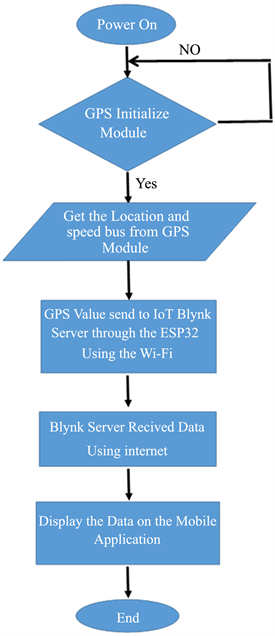
d = distance between the passenger and bus location that get from Equation (3).

s = average speed of bus along route.

To convert the arrival time (t) that will be obtained from Equation (3) from an hour into a minute, it is multiplied by 60 (one hour into a minute).

**Work the Proposed System:**

The proposed system method is working procedure first; the user must determine his location by activating the location feature in a smartphone. To get the information entered the application will provide the details about buses, bus location, bus speed, bus arrival time, nearest bus from a user by offering the distance between user location and bus. This information will assist the passenger to select their suitable bus. In the flowchart that describes the proposed system work in figure.



**Figure 4**. Smart public transportation system flowchart.

**Results and Discussion:**

For testing the efficiency of the proposed system, the prototype has been installed (GPS unit and ESP32) inside a vehicle with supplied internet to use the possibilities offered by the Internet of Things. That vehicle roamed through multiple roads in Mosul city for several days and at different times, for collecting and recording data (latitude, longitude, speed, distance, and time of arrival).

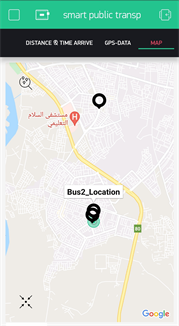
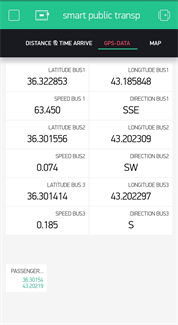
**Location Accuracy Analysis:**

The proposed system has been tested and table . Display data obtained from several streets in the city of Mosul to determine the level of accuracy of the data obtained through GPS on several days and at different times.

**Distance Accuracy Analysis:**

figure presents a graphic comparison of the distance level, the accuracy obtained through the calculation of the great circle distance equation, and the comparison between the total distance obtained from each of the test data and the total distance through real distance.

The Haversine equation used to measure the accuracy of the obtained distance. Based on a study of 4 days with different times, the average difference between the calculated values computed by the Haversine equation compared to the data obtained from a real distance is (177 meters) with an error ratio between 8 to 225 meters. This error is due to the fact that the Haversine equation

(a)(b)(c)

**Figure 5**. Mobile app for prototype of smart public transportation

(a) Buses and passenger location on map,

(b) GPS data,

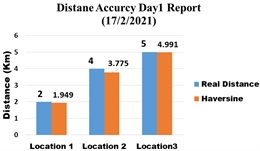
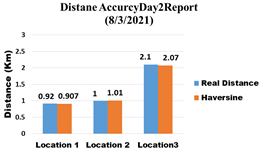
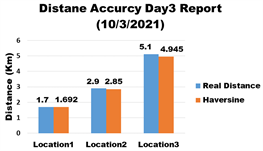
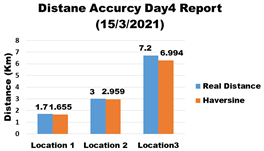
(c) Distance and arrival time of each bus.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Date | Time | Latitude | Longitude |
| 1 | 17/2/2021 | 10:58 am | 36.316739 | 43.191141 |
| 2 | 17/2/2021 | 11:04 am | 36.350198 | 43.183983 |
| 3 | 17/7/2021 | 11:10 am | 36.343765 | 43.182983 |
| 4 | 17/7/2021 | 11:13 am | 36.337387 | 43.162712 |
| 5 | 17/2/2021 | 11:19 am | 36.310638 | 43.207447 |
| 6 | 6/3/2021 | 12:30 am | 36.3015221 | 43.202267 |
| 7 | 8/3/2021 | 11:38 pm | 36.30159 | 43.20216 |
| 8 | 8/3/2021 | 11:39 pm | 36.301559 | 43.202206 |
| 9 | 8/3/2021 | 11:40 pm | 36.310688 | 43.203171 |
| 10 | 8/3/2021 | 11:43 pm | 36.295567 | 43.195332 |
| 11 | 8/3/2021 | 11:55 pm | 36.389160 | 43.187370 |
| 12 | 9/3/2021 | 10:05 am | 36.377155 | 43.149384 |
| 13 | 9/3/2021 | 10:07 am | 36.350117 | 43.238834 |
| 14 | 9/3/2021 | 10:15 am | 36.376621 | 43.153072 |
| 15 | 9/3/2021 | 10:16 am | 36.379650 | 43.153046 |
| 16 | 9/3/2021 | 10:17 am | 36.381821 | 43.150658 |
| 17 | 9/3/2021 | 10.19 am | 36.382019 | 43.156296 |
| 18 | 10/3/2021 | 10:07 am | 36.335922 | 43.202955 |
| 19 | 10/3/2021 | 10:15 am | 36.310688 | 43.203171 |
| 20 | 10/3/2021 | 10:24 am | 36.301514 | 43.202290 |
| 21 | 11/3/2021 | 11:32 am | 36.374424 | 43.147087 |
| 22 | 11/3/2021 | 11:33 am | 36.371513 | 43.136124 |
| 23 | 11/3/2021 | 11:34 am | 36.370644 | 43.133099 |
| 24 | 15/3/2021 | 7:15 pm | 36.301514 | 43.202446 |
| 25 | 15/3/2021 | 7:16 pm | 36.302582 | 43.201664 |
| 26 | 24/3/2021 | 1:25 pm | 36.370796 | 43.133259 |
| 27 | 24/3/2021 | 1:27 pm | 36.365589 | 43.120064 |
| 28 | 24/3/2021 | 1:28 pm | 36.358574 | 43.111202 |

**Arrival Time Accuracy Analysis:**

The arrival time between two points calculated based on Equation (4) as shown in figure, where time accuracy depends on the accuracy of the distance in table

.

(a)(b)(c)(d)

**Figure** . Graphic comparison of distance (a) Day 1, (b) Day 2, (c) Day 3, (d) Day 4.

**Table** . Results obtained applying of the proposed System

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No. | Date | Time | Distance (Km) | Average Speed (Km/h) | Arrival Time (minute) |
| 1 | 15/3/2021 | 10:58:12 am | 3.775 Km | 49 Km/h | 4.5 minute |
| 2 | 15/3/2021 | 11:15:20 am | 1.494 Km | 59 Km/h | 1.98 minute |
| 3 | 15/3/2021 | 11:20:33 am | 4.949 Km | 60 Km/h | 4.99 minute |
| 4 | 21/3/2021 | 3:06:12 pm | 9.450 Km | 42 Km/h | 11.340 minute |
| 5 | 21/3/2021 | 3:12 :30 pm | 6.618 Km | 40 Km/h | 7.94 minute |
| 6 | 21/3/2021 | 3:16:33 pm | 5.229 Km | 31 Km/h | 6.275 minute |
| 7 | 21/3/2021 | 3:19:50 pm | 2.237 Km | 40 Km/h | 3.63 minute |
| 8 | 21/3/2021 | 3:23:13 pm | 1.928 Km | 49 Km/h | 2.314 minute |
| 9 | 24/3/2021 | 1:25:15 pm | 10.500 Km | 59 Km/h | 12.6 minute |
| 10 | 24/3/2021 | 1:25:50 pm | 10..442 Km | 64 Km/h | 12.503 minute |
| 11 | 24/3/2021 | 1:26::18 pm | 10.398 Km | 47 Km/h | 12.478 minute |
| 12 | 27/4/2021 | 11:35:24 am | 2.229 Km | 37 Km/h | 3.6 minute |
| 13 | 27/4/2021 | 11:36:26 am | 2.130 Km | 37 Km/h | 2.972 minute |
| 14 | 27/4/2021 | 11::37:17 am | 1.432 Km | 45 Km/h | 1.998 minute |
| 15 | 27/4/2021 | 11:37:58 am | 0.938 Km | 47 Km/h | 1.309 minute |

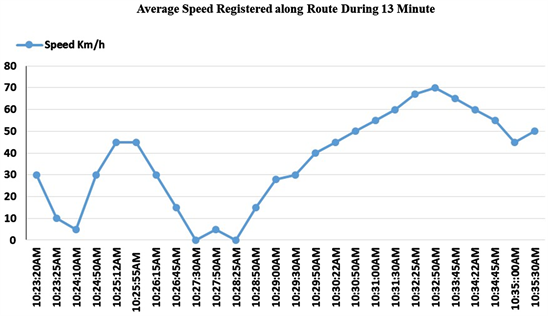


Figure: Graphic of average speed over the time the GPS-Data.

The proposed system is easy to use than other systems and more cost-efficient. It also provides greater performance than RFID and GPS-GSM systems. The application of this proposed system showed that using three GPS-Data only:

latitude, longitude, and speed, it is possible to find information about buses, including arrival time, distance. This proposed system could be a valuable guide for those people involved in the study of GPS-Data gathered from moving buses. In the proposed system, the present location of the bus can be known by the latitude and longitude of the bus. The distance between the location of the bus and the location of the passenger is calculated based on the Haversine equation. The bus’s arrival time is calculated based on Equation (4). All information is loaded onto the Blynk server and displayed on the mobile application.

**Conclusion and Future Work:**

This paper introduces a prototype of a smart public transport system that is designed and implemented using IoT technology, GPS, and ESP32. This system shows that it can use GPS-Data only to get real-time information about buses, such as the current location of the bus, speed, arrival time, and distance. This system offers solutions for users of public transport who take a bus to reach their destinations. This system will help them to reduce the waiting time at the bus station because it provides passengers with necessary information about buses, like the current location of buses, the speed of the bus, arrival time, and the distance. The distance between the user and the bus location, which is calculated using the Haversine equation with a minimum error of 8 meters, and the arrival time of each bus is calculated based on the distance and the average speed of the bus registered along the road.

Python code:

import tkinter as tk  
import requests  
  
def get\_route\_info():  
 current\_location = current\_location\_entry.get()  
 destination = destination\_entry.get()  
   
 # send a request to the bus route API to retrieve the route and arrival time  
 route\_info = requests.get(f"https://api.busroute.com?origin={current\_location}&destination={destination}")  
   
 # extract the route and arrival time from the response  
 route = route\_info["route"]  
 arrival\_time = route\_info["arrival\_time"]  
   
 # display the route and arrival time in the results label  
 results\_label["text"] = f"Route: {route}\nEstimated Arrival Time: {arrival\_time}"  
   
# create the main window  
window = tk.Tk()  
window.title("Bus Route App")  
  
# create the input fields and buttons  
current\_location\_label = tk.Label(text="Current Location:")  
current\_location\_entry = tk.Entry()  
destination\_label = tk.Label(text="Destination:")  
destination\_entry = tk.Entry()  
get\_route\_button = tk.Button(text="Get Route", command=get\_route\_info)  
  
# create the results label  
results\_label = tk.Label(text="")  
  
# add the elements to the window  
current\_location\_label.pack()  
current\_location\_entry.pack()  
destination\_label.pack()  
destination\_entry.pack()  
get\_route\_button.pack()  
results\_label.pack()  
# run the main loop

window.maimloop()