BUS TRACKING SYSTEM WITHOUT USING GPS

Saurabh Patil[1], Roshni Jaiswar[2], Viraj Kadam[3], Rohan Parange[4], Shweta Yadav[5]
[1]Assistant Professor, [2,3,4,5]Students,
Computer Department, Xavier Institute of Engineering, Mumbai, India

[1]Saurabh.terna@gmail.com
[2]rjaiswar39@gmail.com
[3]virajkadam62@gmail.com.com
[4]rohanparnage@gmail.com
[5]shwetaayadav19@gmail.com

Abstract: Reliability in public transport is of great importance today. Many travellers of public transport spend an ample amount of time waiting at bus stops. Our project focuses on presenting a solution that can tackle the above problem by providing travellers with expected bus arrival time. We focus on implementing solution for the same using IoT technology, but without having GPS module imposed on any vehicle. As we will be using static position property of Bus stops for tracking bus.[1] Estimated bus arrival time will be disseminated on bus stops, benefits the traveller to take appropriate decision. The bus stops will be made smart by using single processing unit having capability of fetching required bus data from cloud. The unit at Bus stop also display bus position information and calculate arrival time. The bus mobility units have a small micro controller OBU (On-Board Unit) enabled with Wi-Fi connectivity receiver module**.**

Keywords- AWS, OBU, without GPS, LCD, GPRS, HTTP, DHCP, MQTT.

I. INTRODUCTION

In the way people move to commutes in public transportation systems is the main problem which play an increasingly important role. It is a very cheaper way of transport. Due to heavy traffic and roadwork etc., most of the buses are not in time. At the bus terminus people have to wait for long time without having knowledge of when the bus will arrive. Anybody who wants to use the public transportation system, they can't find the time of arrival of particular bus at the particular destination even plan their departure from home accordingly.[3] Due to unexpected delays in traffic jams the bus arrival time cannot be predicted. Our main aim is to develop a system to which the user's waiting time reduced for bus and will provide him/her with all necessary details regarding the arrival/departure time of the bus, its exact location and when the bus is nearer to the bus terminal
The bus mobility units have a small micro controller.

OBU (On-Board Unit) enabled with Wi-Fi connectivity receiver module. As the bus arrives on a bus stop, its OBU establish Wi-Fi connection with Smart Bus stop Unit and data exchange will be initiated. The bus will send its unique bus number and arrival time stamp. The smart unit at bus stop propagates same information on the cloud. [2] The every other bus stop will access the stored data on the cloud by firing required bus query and calculate the estimated arrival time. The prototype will be implemented by having raspberry as single processing unit at bus stop and bus OBU will be nodeMCU board. We have used raspberry pi 3B, nodeMCU (ESP32),cloud service. Our project is on bus tracking system without using GPS, the bus stop is static we utilize the property to getting estimating time of the bus arrival. In the architecture diagram initially the bus is at bus depot, ones the bus number is selected the system get activated. After activation bus exchanges the bus number and time with the bus depot. The bus depot is connected to the cloud, the bus depot updates the information on the cloud the next corresponding bus stop fetches the information from cloud and calculates approximate time that will required by the bus to reach at it. The time is calculated using the fixed distance between the depot and bus stop. In the next scenario, the bus is at bus stop it again connected to the on board unit (OBU) at the bus stop.[1] It does same thing it exchanges the arrival

time and bus number with OBU on the bus stop its update the information to the cloud and this information is fetch by the next corresponding bus stop from the cloud. This calculates time that will be required for the bus to reach of this bus stop.

II. CHALLENGES FOR PASSENGER

Transportation is a major pain area for cities today. With the ever increasing load on public transport systems, it is really necessary to increase efficiency in these systems. Due to extreme traffic conditions, over-crowding and many other similar issues, public buses lack punctuality and reliability. There is a dire need to tackle this issue. The required solution should not only facilitate improvement in the services, but should also be a driving factor for increase in trust on the public bus transport systems. This paper presents a simple and cost effective solution to make public transportation services 'smart'[1]. The paper will present the concept, technology stack, components and the outcomes of implementing the solution. The primary goal of the proposed solution is to minimize the costs involved in implementation and to create a backend that can scale up easily with increase in demand.

III. DESIGN PHASE

3.1 Architecture Diagram

The components of the diagram are bus depot, bus stop, aws cloud ,lcd ,buses. Initially the bus is at the bus depot. the bus number and the source destination is selected . After entering the information the wi-fi module gets activated.[4] the bus connects to the mesh network. the information is exchanged between the bus and the raspberry pi. The data received by the raspberry is updated on the aws cloud using MQTT protocol.[6] the raspberry pi acts as both publisher and subscriber. it publishes the information to the aws cloud. the other bus stops acts as subscriber and fetch the time and calculate the estimate arrival time of the bus .the time is displayed on the LCD.

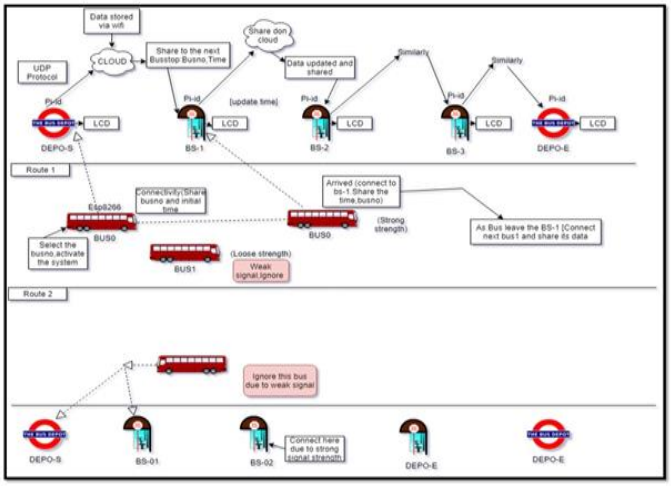

Fig.1. Architecture diagram of bus tracking system.

3.2 Activity diagram

The activity diagram defines the activities that will be performed by the system. Initially the bus will be at the bus depo. the conductor is the activator. The conductor will select the bus number along with the source and destination and the system will get activated .The Esp32 module in the bus will send the bus number entered by the conductor along with source and destination and the current time will be stored .After receiving the data the bus depo will update the information that the bus has left the bus depo along with its source and and destination.

The next bus stop will fetch this information from the AWS cloud and calculate the estimate arrival time that will be required to reach its location and display this information on the LCD. Once the bus arrives at the bus stop it's signal strength will be check by the raspberry pi. If the signal is strong then bus will get connected to the hotspot and it will then send data to the raspberry pi. The raspberry pi will send this data to the cloud and the neighbouring bus stops

will fetch this data and display it on the display unit. Whenever the bus arrives at the bus stop it raspberry will check whether it has reached the destination or not.

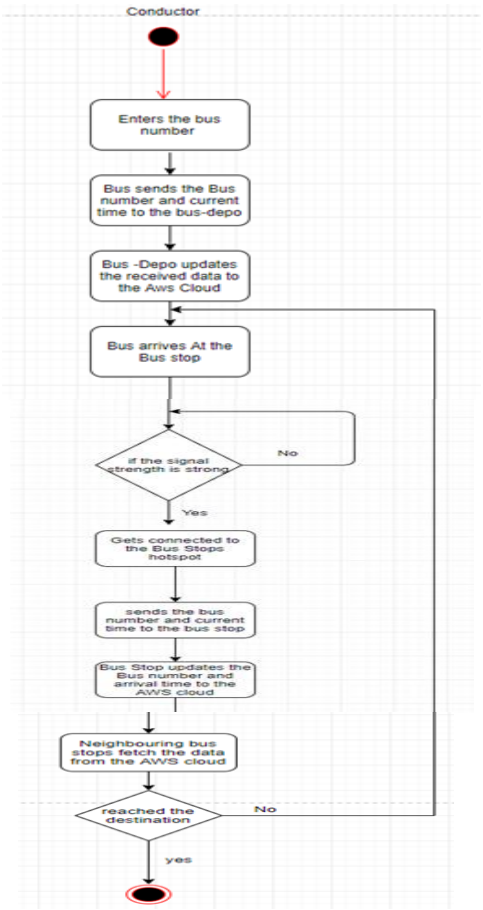

Fig 2 Activity diagram for the bus tracking system

IV. IMPLEMENTATION

4.1 MQTT Client

MQTT is a publish/subscribe protocol that allows edge-of-network devices to publish to a broker. Clients connect to this broker, which then mediates communication between the two devices. [6]Each device can subscribe, or register, to particular topics. When another client publishes a message on a subscribed topic, the broker forwards the message to any client that has subscribed.

MQTT is bidirectional, and maintains state full session awareness. If an edge-of-network device loses connectivity, all subscribed clients will be notified with the “Last Will and Testament” feature of the MQTT server so that any authorized client in the system can publish a new value back to the edge-of-network device, maintaining bidirectional connectivity.

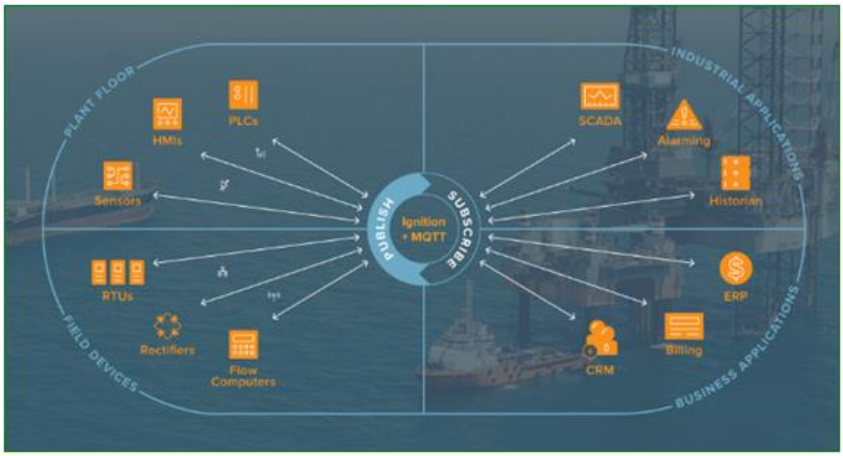

Fig.3. Architecture of MQTT protocol.

4.2 DHCP Server

The Raspberry (present at the Bus-stop) will used as a DHCP Server. It will provide internet to the esp32 module present in the buses. The Esp32 module will get connected to the hotspot of the raspberry pi.[6] you need to install some extra software in order for the Raspberry Pi to act as a Wi-fi router and access point. Install a compatible driver, configure HostAPD and so on. The details are outside the scope of this project, although I've had consistently good results with the Edimax Wireless 802.11b/g/n nano USB adapter – it's small, cheap and easy to work with.

Install RaspAP from your RaspberryPi's shell prompt:

```
$ wget -q https://git.io/voEUQ -O /tmp/raspap && bash /tmp/raspap
```

The installer will complete the steps in the manual installation (below) for you. After the reboot at the end of the installation the wireless network will be configured as an access point as follows:

- IP address: 10.3.141.1
  - Username: admin
  - Password: secret
- DHCP range: 10.3.141.50 to 10.3.141.255
- SSID: `raspi-webgui`
- Password: ChangeMe

4.3 Hardware Components

Raspbian is an open source operating system based on Debian optimized for the raspberry pi hardware. Esp32 will act as a initiator and has wi-fi which gets connected to the rpi hotspot. Rpi has a RAM size of 512MB. GSM module is used to provide internet to the Rpi. Aws cloud used for storing and fetching data. LCD will display the arrival time along with bus number .Miscellaneous component will be used as a connection part.

Table 1: IoT hardware components used

Sr. no.	Product Name	No. of Quantity
1.	Raspberry Pi3	2
2.	ESP32	2
3.	GSM Module	2
4.	AWS cloud	1
5.	Displaying Components	2
6.	Miscellaneous Components (Battery, Wires, Registers, led...)	-

4.3 AWS Connection

AWS IOT secure, bidirectional communication between Internet-connected devices such as sensors, actuators,[4] embedded micro-controllers, or smart appliances and the AWS Cloud. This enables you to collect telemetry data from multiple devices, and store and analyse the data.
Amazon DynamoDB is a NoSQL database that supports key-value and document data models, and enables developers to build modern,[5] serverless applications that can start small and scale globally to support petabytes of data and tens of millions of read and write requests per second.

Create table in dynamo db table:-

```
table = dynamodb.create_table(
   TableName='Bus',
   KeySchema=[
      {
         'AttributeName':'Arrival_time',
         'KeyType': 'HASH'
      }
   ]
```

Here 'Arrival_time is the primary key

'Arrival_time

1. Firstly, the raspberry pi is getting connected to AWS IOT.
2. Raspberry pi is acts as a thing in AWS IOT.
3. The data uploaded on Dynamo DB contains the attribute Bus number, hop count, vehicle number ,time stamp .

upload data on dynamo table

```
table.put_item(
Item={
"Bus_number":busNum,
"Route":route,
"Vehicle_number": vehicle_no,
 "Arrival_time": arrival_time,
 "Hop":Hop,
 "bStop_id":'10'
            })
```

Arrival _time	Bus_ number	Route	Hop	bStopid	Vehicle _no

4. The data fetched by the Raspberry pi is getting uploaded on Dynamo DB after the connection with AWS IOT Fetch the data using bus_number as key
IndexName='Bus_number-index' KeyConditionExpression=Key('Bus_number').eq(33))

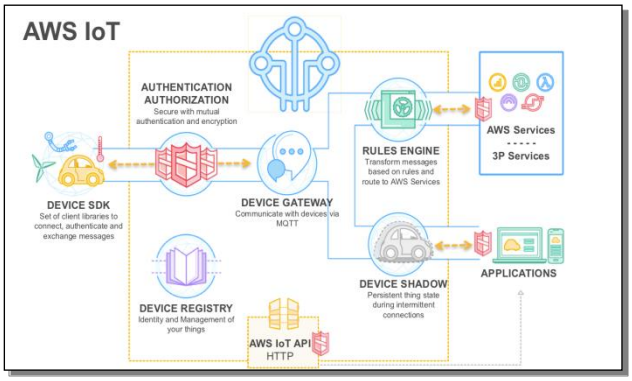

Fig. 4. AWS IOT architecture

A Thing is been created in AWS which is a device connected to AWS. The Data sent by esp32 is been fetched by Rpi .[3] Rpi sends the data to AWS. The data on AWS is then get stored on Dynamo Db with an initial arrival time , serial number and Bus number.

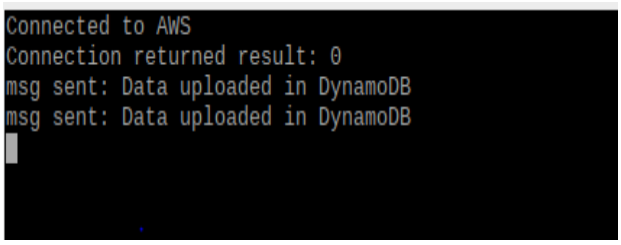

Fig.5. Displaying rpi get connected to AWS cloud and send data to dynamo db

The data stored in Dynamo DB is been displayed on LCD .now[2] the passenger waiting at bus stop able to know arrival time of the bus

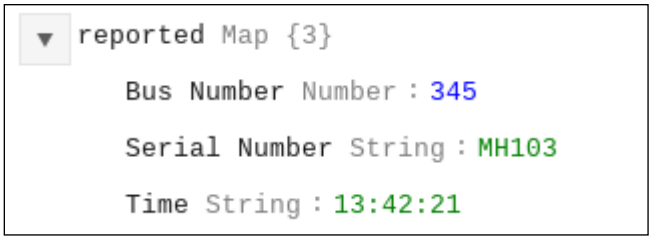

Fig. 6. Data stored in dynamo DB

V. SUMMARY AND CONCLUSION

Our system provides arrival signal as red, yellow, green which indicates the number of bus stop is the bus behind to arrive at bus stop to the passenger by displaying the data on lcd placed on the bus stop. It is completely Iot based software. It is a display lcd and three led light indicator. On the lcd display the Bus no. will be displayed and corresponding led will be blinking showing the status of respective bus to be arrived at that bus stop whether it is 2 bus stop behind, 4 bus stop behind or 6 bus stops behind.

VI. REFERENCES

[1] Jay Lohokare1*, Reshul Dani2*, SumedhSontakke3^,Asst. Prof. Rahul Adhao4**Department of Computer and IT, ^Department of Electrical EngineeringCollegeofEngineering,Pune1lohokarejs13.comp@coep.ac.in,2reshulsd13.comp@coep.ac.in, 3sontakkesa15.elec@coep.ac.in, 978-1-5090-3404-8/17/$31.00 ©2017 IEEE

[2] International Journal of Innovations & Advancement in Computer Science, IJIACS, ISSN 2347 – 8616 ,Volume 6, Issue 12,December 2017 Scalable Tracking System for Public Buses using IoT Technologies Jay Lohokare1, Department of Computer and It ,Department of Electrical EngineeringCollege of Engineering, Pune

[3] IEEE Sponsored 9th International Conference on Intelligent Systems and Control (ISCO)2015

[4] International Journal of Innovative Research in Science, Engineering and Technology(An ISO 3297: 2007 Certified Organization)Website: www.ijirset.comVol. 6, Issue 7, July 2017

[5] AWS IoT: Developer Guide Copyright © 2018 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

[6] https://techtutorialsx.com/2018/05/17/esp32-arduino-sending-data-with-socket-client/

[7] https://github.com/billz/raspap-webgui