Intelligent Public Transport Monitoring System

A **Final Project Report** submitted in partial fulfillment of
the requirements for the degree of

Bachelor of Technology

by

Aniket Shirsat	**Gr.No.**	**101020**
Harshal Sarda	**Gr.No.**	**101189**
Shantanu Hadgekar	**Gr.No.**	**101416**

Under the guidance of
Prof. Mrs. S. N. Shilaskar

DEPARTMENT OF ELECTRONICS ENGINEERING
VISHWAKARMA INSTITUTE OF TECHNOLOGY PUNE
(An Autonomous Institute Affiliated to University of Pune)

2013 - 2014

Bansilal Ramnath Agarwal Charitable Trust's

VISHWAKARMA INSTITUTE OF TECHNOLOGY, PUNE - 37

(An Autonomous Institute Affiliated to University of Pune)

CERTIFICATE

This is to certify that the **Final Project Report** entitled **Intelligent Public Transport Monitoring System** has been submitted in the academic year **2013-14** by

Aniket Shirsat	**Gr.No.**	**101020**
Harshal Sarda	**Gr.No.**	**101189**
Shantanu Hadgekar	**Gr.No.**	**101416**

under the supervision of **Prof. Mrs. S. N. Shilaskar** in partial fulfillment of the requirements for the degree of Bachelor of Technology in **Electronics / Electronics and Telecommunication Engineering** as prescribed by University of Pune.

Guide/Supervisor

Name: Prof. Mrs. S. N. Shilaskar

Signature:

Head of the Department

Name: Prof. A. M. Chopde

Signature:

External Examiner

Name:

Signature:

Acknowledgments

It is matter of great pleasure for us to submit this **B.Tech. Final Year project report** on **Intelligent Public Transport Monitoring System**, as a part of curriculum for award of **Bachelor of Technology in Electronics and Telecommunication.**

We are thankful to our project guide **Prof. Mrs. S. N. Shilaskar**, Assistant Professor in Electronics Engineering Department for her constant encouragement and able guidance.

We are also thankful to **Prof. A. M. Chopde, Head of Electronics Engineering Department** for his valuable support.

We take this opportunity to express our deep sense of gratitude towards those, who have helped us in various ways, for preparing this B.Tech. Final Year Project. At last but not the least, we are thankful to our parents, who encouraged and inspired us with their blessings.

Date: _____

Aniket Shirsat

Harshal Sarda

Shantanu Hadgekar

Contents

List of Figures

Abstract

Our project aims to provide an efficient and user friendly way to monitor public transport. The proposed system, which is mainly based on GPS and ZigBee, highly enhances and simplifies the complete public transportation system. A GPS enabled device is installed on all the vehicles to be monitored which then transmits the location details of a particular vehicle to a Central base station. This data transfer is achieved through internet. All such data from different vehicles is then aggregated, appropriately processed and then finally relayed to a front end user. Such a system is more efficient and convenient in the sense that it entertains a passenger′s demands and gives bus locations in real time. Also, the Estimated Time of Arrival (ETA) of a particular bus at a particular stop can be calculated using suitable algorithms.

Chapter 1

Introduction

The public transportation system of a country is one of the major factors in terms of which, the country's progress is recorded. India is a vast country having many major cities. People in these cities rely on public transports for their everyday commute. Public transport plays a great role in these people's lives. If the system was to break down for a day or so, there will be turmoil and the activities will be adversely affected. Management of a public transport system is a tough thing to do. Ensuring the timely running of buses is not easy. There may be delay in the arrival of buses at the stops due to unforeseen factors like accidents on roads, traffic jams, faults in the vehicle, etc. Due to these delays, people's activities get affected and this causes frustration among the people over time who then prefer not to use the public transport and use their own vehicles which results in burning of more fuel and unnecessary spending of more money. To make the commuters aware about the possible delays in the arrival and other incidents related to the buses, there should be a system which would enable the people at the stops to know about the real time happenings of the buses. When the real time events are displayed to the people, they will know about any possible delays and change their schedule accordingly. This will also lead to a more reliable public transport system. The real time events can be displayed by tracking each bus with the help of a specialized module which finds the location of the bus and transmits it along with some other data related to the bus, to a central station which then updates information at various bus stops for people to see.

All this can be done without using expensive and complex hardware. A good public transport system will only encourage more people to use it. By simplifying the method by which the people shall know about various buses will only add to the positive attributes of the public transportation system. Also, as they will know about the ETAs of various stops

beforehand, they can manage their activities in a better manner. This will lead to more usage of public transports thereby leading to more revenue generated by the government. Also, due to the use of advanced technologies, the system will be transparent and faults can be detected in a hassle free manner.

Chapter 2

Literature Survey

Even the most advanced countries of the world are highly dependent on their public transport systems. European countries have a dense network of trains and trams and buses running through the cities. The management system of these transports is very good and has been implemented very nicely. Trains come and leave on schedule and bus stops are updated with the real time information of the buses on a minute by minute basis.

Several works have been done in the field of Intelligent Transport System and most of them take advantage of the freely available GPS to develop several applications which are used to make transportation service easier for the transport management centers, drivers and passengers.

The basic concept of a public transport monitoring system is similar across all the methods which have been used to implement such a system. The vehicle to be tracked is equipped with a module which detects its location and transmits it to an authorized station which processes the information and updates the information across all the bus stops.

Some methods are more optimised than others. The basic method is to just detect the location of the bus and update the ETA and some other parameters at the bus stops. The most advanced methods have the ability to find out the number of passengers waiting at a bus stop, assigning priority according to the number of passengers at the stop and then routing the bus according to this priority. Also, bus stops get updated with more information like the number of people already in the bus, number of people getting out of the bus at a particular stop, and the number of people getting in the bus at that particular stop. Buses can also include an LCD screen which displays information to the people inside. These are different from the standard LED displays which are present in buses nowadays. The LCDs display more information than just the next stop. They can display the position of the bus on a map in real time, the current

speed of the bus, the ETA of the bus for a particular stop, etc.

2.1 Papers referred

In [2], the authors have described a system which makes use of GPS and GSM for monitoring a bus. The system consists of four modules: BUS Station Module, In-BUS Module, BASE Station Module and BUS Stop Module. BUS Station Module sends the initialization information containing the bus number and license plate number to InBUS Module and BASE Station Module using SMS. In-BUS Module then starts transmitting its location and number of passengers to BASE Station Module. BASE Station Module is designed to keep track record of every bus, process user request about a particular bus location out of BUS Station and update buses location on bus stops. Bus stop module receives buses location information coming towards that stop from BASE Station module and displays the information on a dot matrix display.

Authors in [1] present an Autonomic Architecture for Real-Time Traffic Network Management in congested urban transportation networks. The architecture assumes the availability of spatially distributed controllers in the network which is capable of monitoring the traffic within a predefined sub network and provides efficient control strategies for its traffic. Theses controllers could be dynamically configured to operate in teams to develop integrated traffic management schemes that best cope with the observed traffic pattern in the network. In addition, authors in [1] proposed that for the next generation location-based services (LBS) to become truly ubiquitous, they will have to rely on mobile platforms upon which multiple sensors and measurement systems have been integrated to provide continuous, three-dimensional positioning and orientation. He further submits that Next-generation LBS need theoretically sound methods to translate position into location information.

Furthermore, authors in [8] analyzed different vehicle location technologies used in automatic vehicle monitoring and management system. They claim that Automatic Vehicle Location (AVL) technologies have the potential for improving fleet efficiency and providing better route planning and scheduling. However, because there are a number of possible AVL technologies from which to choose, they concluded that transportation managers must analyze and evaluate their system needs and requirements before implementing any single or integrated approach to AVL.

In addition, authors also present a methodology for identifying travelers transportation

modes by tracking GPS equipped mobile devices in the traffic stream. Various mobile phone service providers have location-based services (LBS) that track the locations of their mobile phones. One major concern in using mobile phones for traffic monitoring is that the phones are not necessarily in passenger vehicles. The mobile device can be in a car, bus, or other modes that have distinct speed and acceleration profiles. In addition, querying the mobile device has monetary cost implications, and the higher the number of location queries from the server the higher the associated cost. Their work focused on the feasibility of using the characteristics of the trail of GPS data stream to identify the mode on which the mobile device is located. The research was carried out in Toronto whose LBS can only provide GPS data once every five minutes, because of the sampling limitation, a GPS data logger is used to collect the trip data and the logged data is sampled at varying frequencies as if they are coming from the mobile phones.

Author in [4], proposed wireless communication alternatives for intelligent transportation systems, they submitted that communication systems are the basis of every effective and reliable traffic control and management application. While cellular-based communication through commercial carriers is widely used for online traffic management applications, they submitted that public agencies have also begun to consider other technologies, such as WiFi and WiMAX. Most such agencies still seek additional guidelines for the selection of suitable wireless options for different traffic control and management applications under different physical and environmental conditions. Performance and reliability are among the most important parameters to be considered when examining wireless communication options for traffic control and management applications.

In [5], the authors have described a system which is Demand Responsive Transit (DRT). It entertains a passengers demands and gives bus locations in real time. The real time synchronization of The Flexible Bus System makes it information rich and unique as compared to other DRTs. They have suggested using ZigBee for communication between a bus and a bus stop. The bus stop is connected to a central station by a wired or wireless internet connection. The data obtained by the bus stop through ZigBee from the bus is transmitted to the central station and the appropriate information is then updated to all the other bus stops regarding a particular bus.

2.2 London iBus System

One example of a transport monitoring system is London iBus. The iBus system can locate every bus to an accuracy of about ten metres, or its distance from the nearest stop by around ten seconds. It does this using several instruments like Global Positioning System (GPS), Odometers, Speedometers, etc. The system mainly relies on the GPS data which is used to roughly determine the location. This data is then transmitted to the Central system via GPRS. The Central System then makes a best guess of the bus position and depicts the overall image derived from the data provided by all buses.

2.3 Indian railways trainradar.com

Another example of a transport monitoring system is Indian Railways Rail Radar.

This gives real time information about trains running in India. Every train is shown on India's map and various details about the trains are displayed. The trains are colour coded to show whether they are on time or are late. The current speed and ETAs are also displayed.

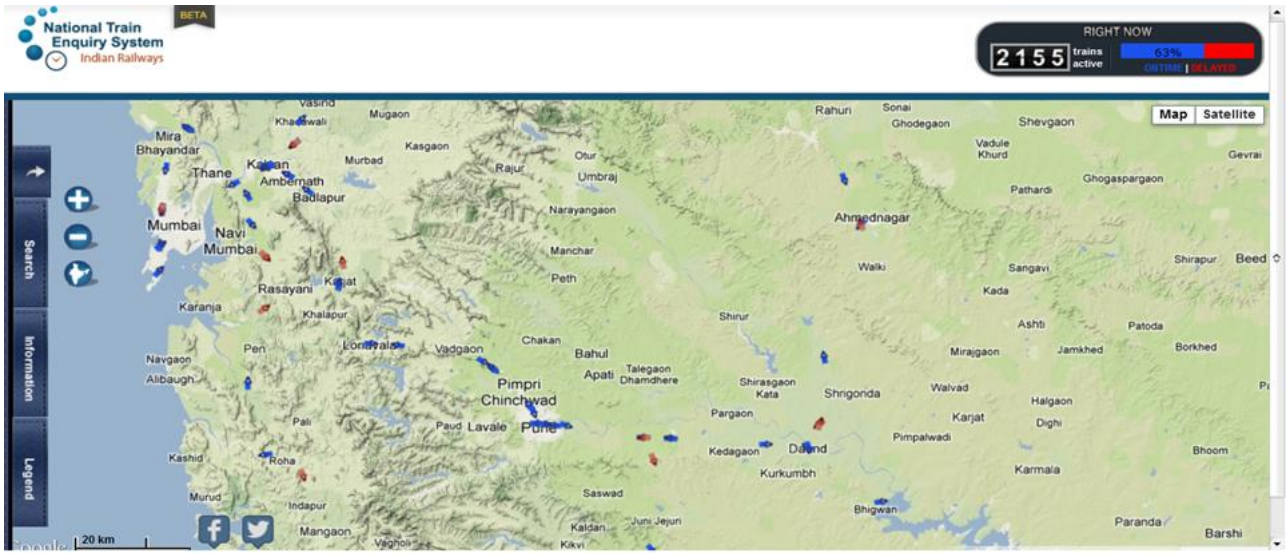

Figure 2.1: Screenshot of www.railradar.com

Chapter 3

Components

3.1 RaspberryPi

The RaspberryPi is a credit-card-sized single-board computer developed in the UK by the RaspberryPi Foundation with the intention of promoting the teaching of basic computer science.The RaspberryPi has a Broadcom BCM2835 system on a chip, which includes an ARM1176JZF-S 700 MHz processor (The firmware includes a number of Turbo modes so that the user can attempt overclocking up to 1 GHz, without affecting the warranty), VideoCore IV GPU, and was originally shipped with 256 megabytes of RAM, later upgraded to 512 MB. It does not include a built-in hard disk or solid-state drive, but uses an SD card for booting and long-term storage.

The Foundation provides Debian and Arch Linux ARM distributions for download. Tools are available for Python as the main programming language, with support for BBC BASIC (via the RISC OS image or the Brandy Basic clone for Linux), C and Perl.

Since RaspberryPi is a full featured mini-computer, it can support huge displays and higher level peripherals like Ethernet and USB. At the same time, it also features low level peripherals like General Purpose Input Output (GPIO) and Universal Asynchronous Receiver Transmitter (UART).

3.1.1 Specifications of RasberryPi Model B Rev.2

- System-on-a-chip (SoC): Broadcom BCM2835 (CPU + GPU. SDRAM is a separate chip stacked on top)
- CPU: 700 MHz ARM11 ARM1176JZF-S core

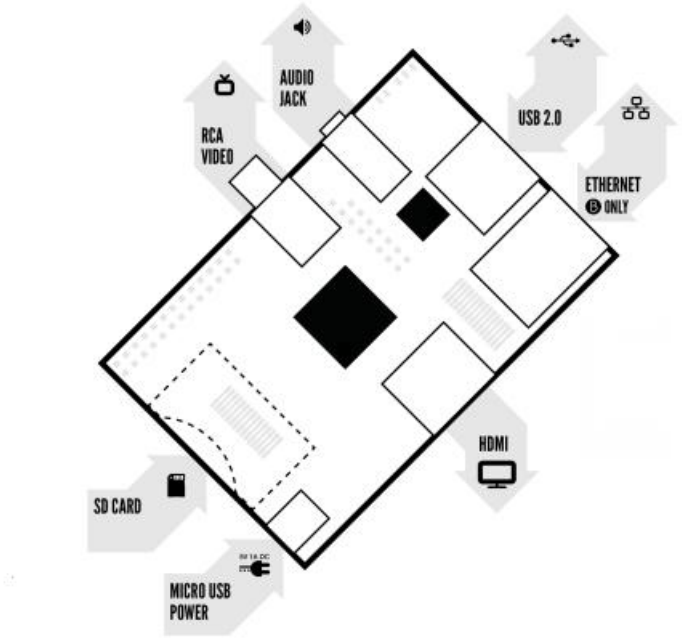

Figure 3.1: Overview of RaspberryPi

- GPU: Broadcom VideoCore IV, OpenGL ES 2.0, OpenVG 1080p30 H.264 high-profile encode/decode
- Memory:(SDRAM) 512MB
- USB 2.0 ports: 2
- Video outputs: Composite video Composite RCA, HDMI (not at the same time)
- Audio outputs: TRS connector 3.5 mm jack, HDMI
- Audio inputs: none, but a USB mic or sound-card could be added
- Onboard Storage: Secure Digital SD / MMC / SDIO card slot Onboard Network: 10/100 wired Ethernet RJ45
- Low-level peripherals: General Purpose Input/Output (GPIO) pins, Serial Peripheral Interface Bus (SPI), I2C, I2S, Universal asynchronous receiver/transmitter (UART)
- Real-time clock: None
- Power ratings: 700 mA, (3.5 W)

- Power source: 5 V (DC) via Micro USB type B or GPIO header
- Size: 85.0 mm x 56.0 mm x 17 mm
- Weight: 40g

Figure 3.2: RaspberryPi

3.1.2 Python

Python is a widely used general-purpose, high-level programming language. Its design philosophy emphasizes code readability, and its syntax allows programmers to express concepts in fewer lines of code than would be possible in languages such as C. Python is free to use, even for commercial products, because of its OSI-approved open source license. Modules to interface General Purpose Input Output pins, UART communication and two-wire communication on RaspberryPi are provided by the RaspberryPi Foundation.

3.2 Global Positioning System (GPS)

SimpleLink GPS CC4000 Module by GNS has a GPS driver and firmware fully integrated into Module. It supports industry standard NMEA 0183 interface protocol communication. It implements UART host protocol. RaspberryPi receives the NMEA protocol strings over the RX-pin on the RaspberryPi. The UART works at 9600 baud rate for perfect synchronization of devices on both ends.

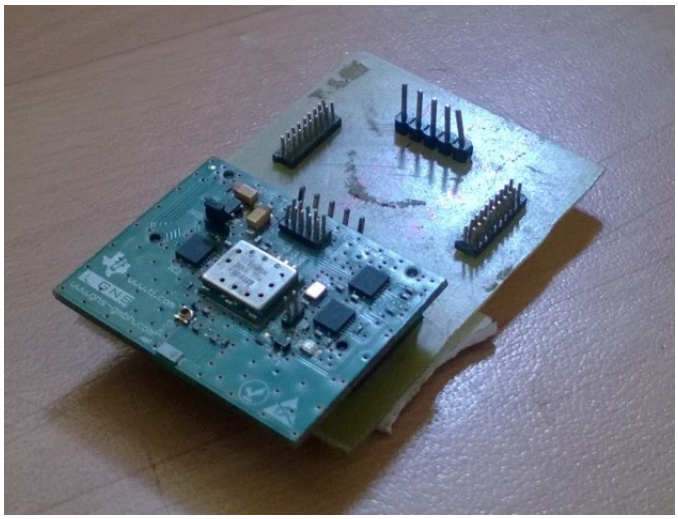

Figure 3.3: GPS CC4000

3.2.1 GPS Interfacing: Hardware

The GPS module has the following pins for interfacing.

- VCC 3.3V
- GROUND
- PUSH TO FIX
- TX (UART)
- FIX AVAILABLE

The VCC and GROUND are used to provide supply of 3.3V regulated to the module.

The Push-to-fix is an input pin for the module that has to be pulled high to enable GPS tracking. If the pin is at low voltage, the GPS tracking is turned off. Hence, using a microcontroller, we can disable the GPS module in order to save battery power.

The TX pin transmits GPS data in NMEA protocol in UART host protocol at 9600 baudrate. Microcontrollers with a standard UART module can interface with the GPS module. The TX pin from the GPS module is to be connected to the RX pin on the microcontroller.

3.2.2 GPS Interfacing: Software

NMEA 0183 is a combined electrical and data specification for communication between marine electronics such as echo sounder, SONARs, anemometer, gyrocompass, autopilot, GPS receivers and many other types of instruments. It has been defined by, and is controlled by, the National Marine Electronics Association.

These are some of the data format specification in NMEA protocol.

- GGA: Time, Position and Fix Type Data
- GSA: Latitude, Longitude, UTC Time of Position Fix and Status GPS Receiver Operating Mode, Satellites Used in the Position Solution, and DOP Values
- GSV: Number of GPS Satellites in View, Satellite ID Numbers, Elevation, Azimuth, and SNR Values
- VTG: Course and Speed Information Relative to the Ground
- RMC: Recommended Minimum Specific GPS Data

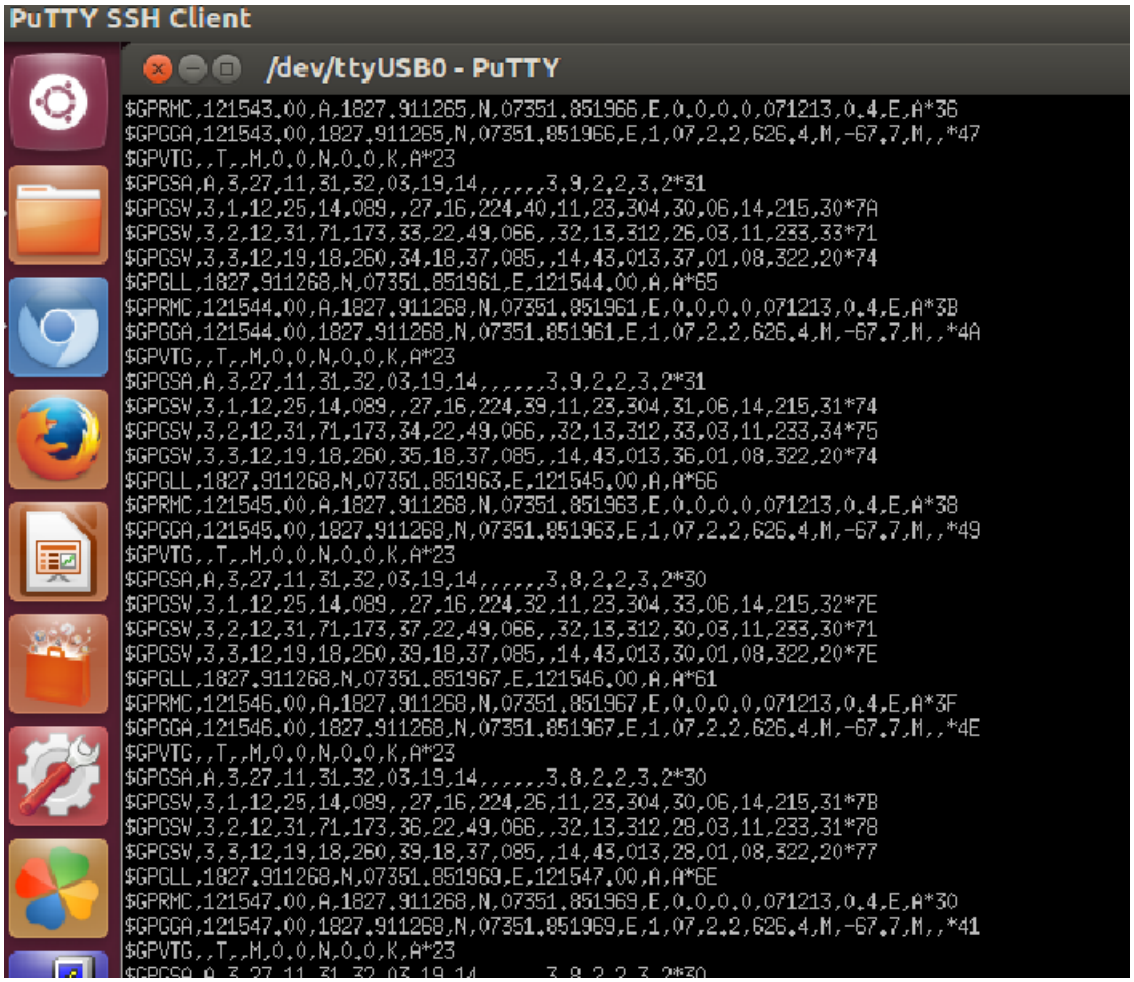

Figure 3.4: Sample Data from GPS CC4000

3.3 ZigBee

ZigBee is a specification for a suite of high level communication protocols used to create personal area networks built from small, low power digital radios. ZigBee is based on the IEEE 802.15.4 standard. Though low-powered, ZigBee devices can transmit data over long distances by passing data through intermediate devices to reach more distant ones, creating a mesh network; i.e., a network with no centralized control or high-power transmitter/receiver able to reach all of the networked devices. The decentralized nature of such wireless ad hoc networks makes them suitable for applications where a central node can't be relied upon.

ZigBee is used in applications that require only a low data rate, long battery life, and secure networking. ZigBee has a defined rate of 250 kbit/s, best suited for periodic or intermittent data or a single signal transmission from a sensor or input device. The technology defined by the ZigBee specification is intended to be simpler and less expensive than other wireless networks, such as Bluetooth or Wi-Fi.

ZigBee communication is best suited for this project because of its low power consumption and high reliablity. When it comes to public transport the boarding points are always fixed. These bus stops have to be detected accurately to facilitate location identification on a quantized level. While we do have GPS co-ordinates it is very difficult to detect a bus stop just from the GPS coordinates received from the bus. That is why we used ZigBee in our prototype.

3.3.1 Features

- Range - Outdoor line of sight: up to 50 meters with directional antenna.
- Transmit Power: up to 1 watt / 30 dBm nominal.
- Receiver Sensitivity: up to 107 dBm.
- AT Command Modes for configuring Module Parameters
- Direct sequence spread spectrum technology.
- Analog to digital conversion and digital I/O line support.

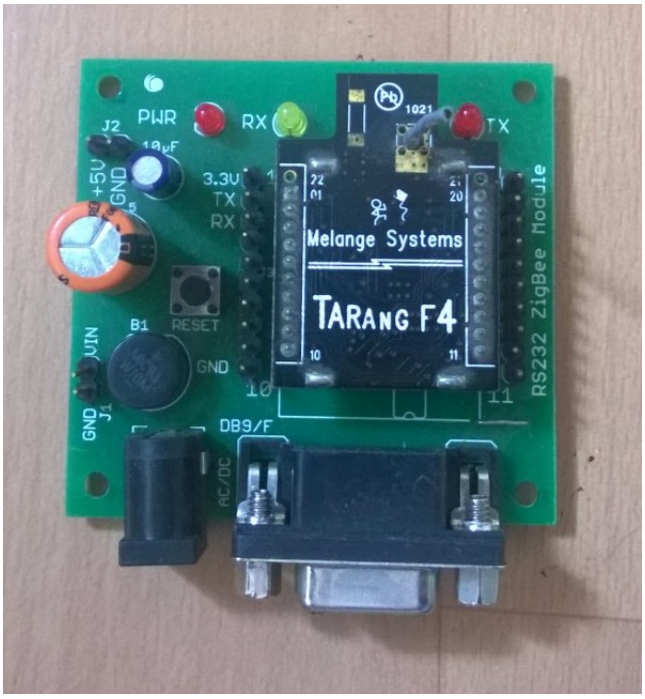

Figure 3.5: ZigBee

3.3.2 Interfacing ZigBee

ZigBee modules have the ability to transmit Digital, PWM, Analog or Serial RS232 signals wirelessly. To communicate over UART, we just need three basic signals which are namely, RXD (receive), TXD (transmit), GND (common ground). So to interface UART with micro-controller, we just need the basic signals.

We can select between 16 direct sequence channels. All the ZigBee modules in our system should be configured to the same channel. Only ZigBee modules configured with same serial settings i.e. baudrate, channel, data bits can communicate with each other.

3.4 MSP430 Microcontroller

The MSP430G2553 from Texas Instruments MSP430 Series 16-bit MCU has 16KB flash, 512 bytes RAM, up to 16-MHz CPU speed, 10-bit ADC, capacitive touch enabled I/Os and universal serial communication interface.

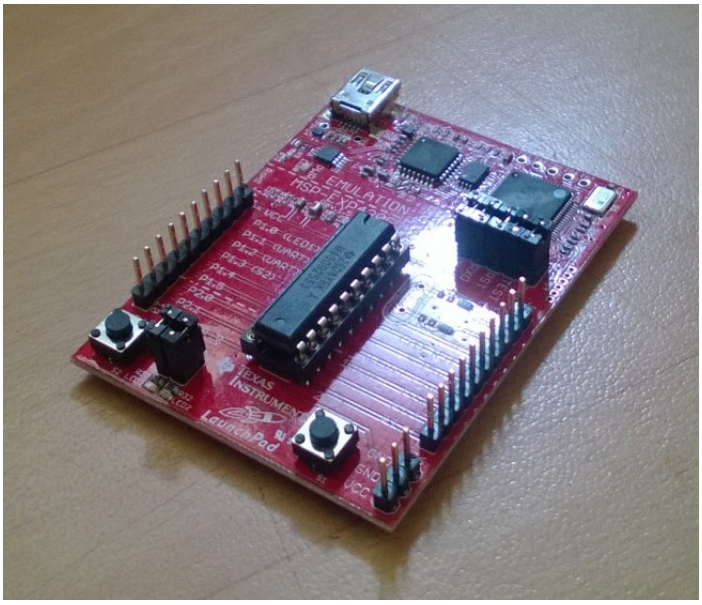

Figure 3.6: TI′s Launchpad MSP430G2553 Microcontroller Board

3.4.1 MSP-EXP430G2 LaunchPad features

- USB debugging and programming interface featuring a driverless installation and application UART serial communication with up to 9600 baud rate.
- Supports MSP430G2xx2, MSP430G2xx3, and MSP430F20xx devices in PDIP14 or PDIP20 packages.
- Two general-purpose digital I/O pins connected to green and red LEDs for visual feedback.
- Two push button for user feedback and device reset.
- Easily accessible device pins for debugging purposes or as socket for adding customized extension boards.

- High-quality 20-pin DIP socket for an easy plug-in or removal of the target device.

MSP430 microcontroller is best suited for our application because of its low cost, low power consumption and availability of ample features. MSP430 has an inbuilt UART module and hence is compatible with ZigBee module. We configure the UART at 9600 baud rate with 8-bit data length and no parity bit.

Chapter 4

In Vehicle Module

The In-Vehicle module is the one which is mounted on any vehicle that is to be tracked. It has RaspberryPi as the main controlling unit. Following are the functions of this In-Vehicle module:

1 Get the current GPS coordinates to get real time location details of the bus.

2 Continuously check for ZigBee signals to update the last bus stop details.

3 Post this information on the Internet in such a way so that it can be accessed by all other commuters.

4 Diplay the consolidated information on the accompanying display unit for the travellers inside to know about upcoming stops.

4.1 Components

- RaspberryPi
- ZigBee
- GPS (CC4000)
- Display

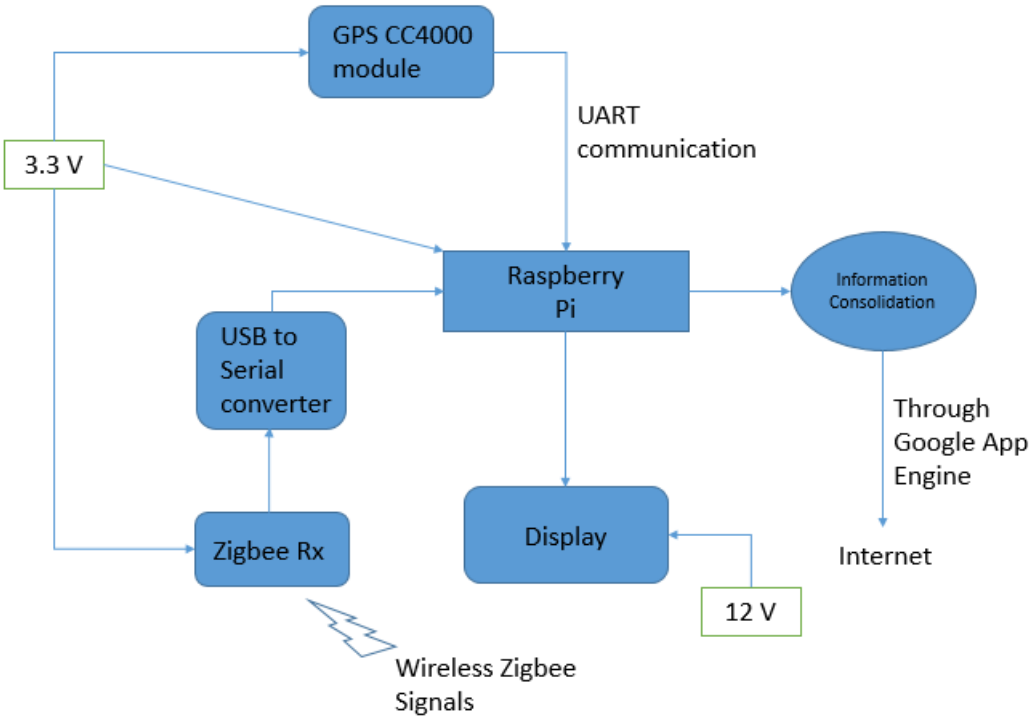

Figure 4.1: InVehicle Module Block Diagram

4.2 Working

RaspberryPi is used as the main controlling unit on the in-vehicle module. It is connected to a display on the vehicle. The RaspberryPi has prestored dataset of the route to be followed and the bus-stand related information like bus-stand name, distance from starting point, bus-stand UniqueID. This data set is stored in the form of lists and dictionaries in the python interface. Each RaspberryPi is also assigned a Unique BusID. This serves as the unique identifier for the bus to be tracked.

The GPS module CC4000 is connected to the RaspberryPi. The connection is established using UART prtocol and takes up only 4 pins viz. VCC(Power Supply), GND(Ground), Dataline and PUSH to FIX. The power supply required by CC4000 is 3.3V. The TX on GPS is connected to the RX pin on RaspberryPi. Data is transmitted through this line in UART protocol format. The data format is 8-bit data, 9600 baudrate, no parity, 1 start-bit and 1 stop-bit, which is predefined by the manufacturer and provided in the datasheet. In one second about two strings on NMEA protocols are received.

The RaspberryPi is connected to a ZigBee. Since RaspberryPi only has one serial port

which is used by the GPS, we convert a USB port into a serial port using hardware. We use a USB-to-serial convertor for this purpose. The USB-to-Serial convertor converts the USB port into a virtual serialport which can be read on the RaspberryPi like any other normal serial-port.

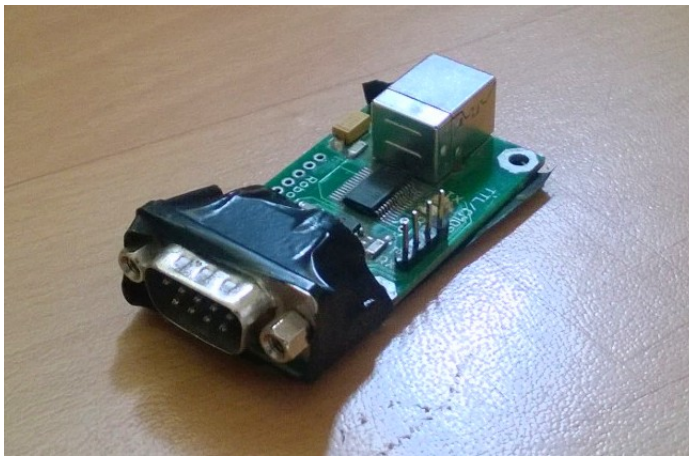

Figure 4.2: USB to Serial Convertor

The RaspberryPi receives the data via two serial ports which are defined as 'ttyAMA0' and 'ttyUSB0'. Using python programming we accept the data via the serial ports, by using a python module 'serial'. The 'serial' module is the interface between interactive python and serial ports.

Acquiring data from the ports at the same time can not be achieved using a single threaded python code. Hence we use threading capablities of python provided by the python module 'threading'. Separating the threads that acquire data from the two serial ports makes the code concurrent. We can easily acquire data simulataneously and store it on our system using these features of python. The GPS string received via serial-port 'ttyAMA0' is in NMEA protocol, and is converted to a list in python. The list consists of information received in one sample of GPS dataset. The list elements can be identified by the starting keywords such as $GPGGA, $GPGLL,etc. The ZigBee connected to the serial-port 'ttyUSB0' receives data only in the proximity of the bus-stand, since the transmitting zigbee on the bus-stand transmits data continuously and ZigBee works in the range of upto 50 meters. The format of the data transmitted is predefined and is of the format '$BUS-STAND0xxxxx'. The port 'ttyUSB0' is continously open for receiving data in a separate thread. The data is received in the form of a string. When the data is received, it is checked whether it is in the predefined format. If not, the data is rogue data and is ignored. If the data matches the predefined format than the information is extracted

from the string. The information consists of the uniqueID of the bus stand. This information is stored on the RaspberryPi. When the data is received and verified, it means that the bus has reached the bus-stand. Now again the same information is received by the zigbee module for the time the bus is in proximity of the bus-stand. We turn off the ZigBee module for 15 seconds after receiving any data in correct format. In about 15 seconds, the bus exits the proximity of the bus-stand under normal circumstances. This is done to avoid accepting data which might get repeated many times.

As soon as it receives any data it forwards it to RaspberryPi which then concatenates it with the other data and transmits it through internet to the main server which is maintained by Google app engine in our case. Many such data packets are simultaneously transmitted by all different buses in the city. The unique BusID helps identify the origin of the packets of data which in turn helps the server differentiate all the buses in the city.

All the components in the InVehicle module are powered by a 3.3 Volt power supply. The accompanying display on the other hand requires a 12 Volts DC supply. So we use a single 12 volt battery in the complete module. This is directly given to the Display and also converted to 3.3 Volt which is required by RaspberryPi, GPS and ZigBee.

The display shows the position of the bus on google maps in real-time. It also shows information about the previous and next bus stands.

Chapter 5

Bus Stand Module

The bus-stand module is installed on every bus stand in the city. It continously transmits the unique identification number of the bus-stand. The buses after coming near a bus stand, acquire the signal and can confirm that they have reached the bus-stand.

5.1 Components

- MSP430 Microcontroller
- ZigBee

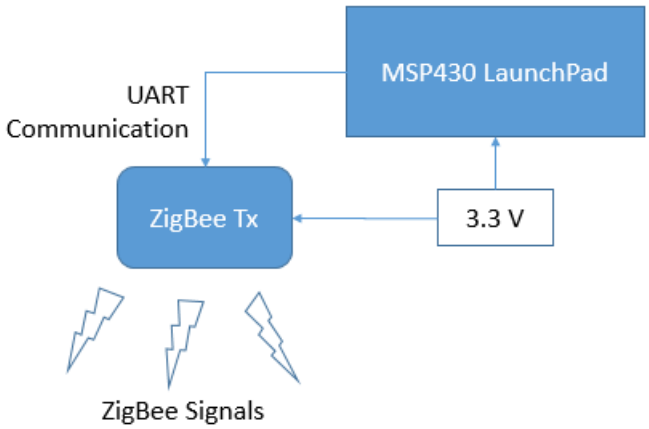

Figure 5.1: Bus-Stand Module Block Diagram

5.2 Microcontroller

MSP430 microcontroller is best suited for our application because of its low power consumption and availibility of ample features. MSP430 is programmed using the Code Composer Studio Suite provided by Texas Instruments. The coding language used is C. We configure the UART at 9600 baudrate with 8-bit data length and no parity bit, 1 start bit and 1 stop bit.

The Microcontroller is programmed to transmit a string continuously via UART. The string is preloaded into the MSP430 at initial stage. The string corresponds to the uniqueID of the bus-stand.

5.3 ZigBee

The ZigBee on the bus-stand module transmits the unique identification code of the bus-stand over a predefined standard channnel of ZigBee in a predefined format. The transmissions are continous at an interval of 5 seconds.

It is better to send the signals at a constant predefined interval of 5 seconds, because transmitting continuously at full speed will exhaust the on-board battery very quickly.

5.4 UniqueID

Each bus stand in the system is assigned a uniqueID while designing the system. The data set containing information of all bus stands with their corresponding uniqueID's are stored in the in-vehicle module.

Each bus-stand transmits the assigned uniqueID. When the vehicle reaches the proximity of the bus-stand, it receives the uniqueID and searches it in the pre-installed dataset. The uniqueID gives information of the bus-stand name and its position with respect to other bus-stands.

The format of the uniqueID is defined as 'BUS-STAND0xxxxx'. Here x can be a value between 0 and 9. Maximum UniqueID's possible with this format is 10,000. This much capacity is enough to cover all the bus-stands in a city or a metropolis.

The starting point is given a uniqueID which is a multiple if 100. The consecutive bus-stands are given uniqueID's in a incremental fashion. This makes bus-stand tracking easy as we are following an ascending pattern.

Chapter 6

Google App Engine

6.1 What Is Google App Engine?

Google App Engine is a Platform as a Service (PaaS) offering that lets you build and run applications on Google′s infrastructure. App Engine applications are easy to build, easy to maintain, and easy to scale as the traffic and data storage needs change. It acts like a web server for any required application. Once the application is uploaded it is simply deployed by Google to make it available on the Internet. The Google App Engine supports 4 run-time environments for now: Java, Python, PHP and Go. We are using the Python run time environment (webapp2 framework) in our prototype. Its also supports GQL which is a database management and querying script like SQL (Structured Query Language). The syntax for it is similar to SQL.

6.2 How it is used in our application?

The development API for Google app engine is freely available. Also the internal management of servers and the data is handled by Google so that the focus remains only on the application to be deployed.

The RaspberryPi receives all the information from GPS and ZigBee modules as explained earlier. It then consolidates all that information and makes it ready to be sent online. The back-end App Engine script then helps in sending this consolidated information to a Google server.

A typical sample for transmitted data has the following :

1 BusID : The BusID helps identify each seperate bus in the city

2 Latitude, Longitude : As received from GPS module

3 Speed : As received from GPS module

4 Last-bus stand : The Bus-standID which denotes a fixed and unique bus stop. This field denotes the last bus stop crossed by the bus under consideration.

The server is configured to maintain a database of this information received. The information received is structured properly using appropraite identifiers for easy identification. After that a front-end App Engine script is made to run on user-terminal's web browser. The App Engine Server is trigerred as per the user request and the appropriate data from the database is received on the front end after which it is handled by a local script running on user's browser. Thus essentially, a web server is maintained using Google App Engine which accumulates all the information from buses and handles all the requests from the front end user providing him with all the required information.

Chapter 7

Google Maps API

Google Maps is a web mapping service application and technology provided by Google. It is widely used over the world and has a vast range of applications.

7.1 What is it?

Google launched the Google Maps API to allow developers to integrate Google Maps into their websites. It is a free service and is available to anyone who wishes to use Google Maps in their application. By using the Google Maps API, it is possible to embed the Google Maps service into an external website, on to which site specific data can be overlaid. The complete API is based on JavaScript. Using this API, any GPS co-ordinates can be mapped and information can be shown with the use of appropriate markers.

7.2 How is it used in our application

As already described, we would be getting the real time GPS co-ordinates from any desired location using the CC-4000. These co-ordinates will then be transferred to the RaspberryPi which will then transmit them to an appropriate server. Now, these GPS cordinates are dynamically read by the front end application code by running a script. And finally, using the Maps API, the real time target moving animation can be shown using an appropriate marker. Every time the code reads a new GPS cordinate, it changes the postion of the marker on the map accordingly. This gives us a brief idea about current location of the target as well as its motion profile. Also, as there will be GPS data from a lot of targets, it is necessary to identify each of them uniquely

so that the front end user can get location information about the desired targets.

7.3 Approach

Almost everything in the Maps API is object based. The first step in using the API is to get a key from Google which is required for any of our code to work. Implementing things on the Maps API is possible because of the predifined objects provided by Google. The first thing in the code is to initialize the Maps Object. The Map is then centered on to a particular co-ordinate and zoom level is set. After this initializing process, the main code starts. There is a different object to create a marker and show it on any particular location. This location is decided by the GPS co-ordinates received from the target end. Changing these GPS co-ordinates according to the movement of the target sets up the moving animation of the marker. The marker also carrries a unique identifier (eg. bus number) to filter out only the data which is requested by the user.

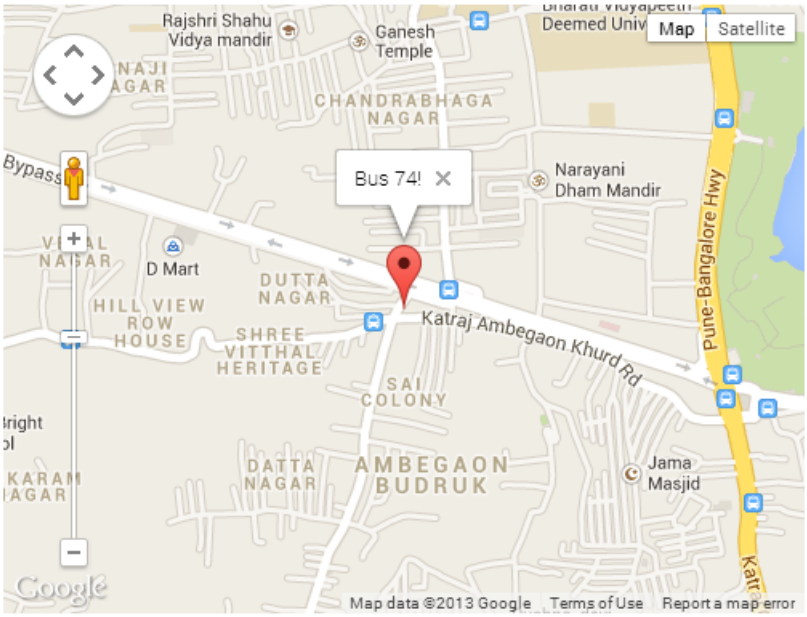

Figure 7.1: Marker Generation on Google Maps using API

Chapter 8

Joining All Pieces

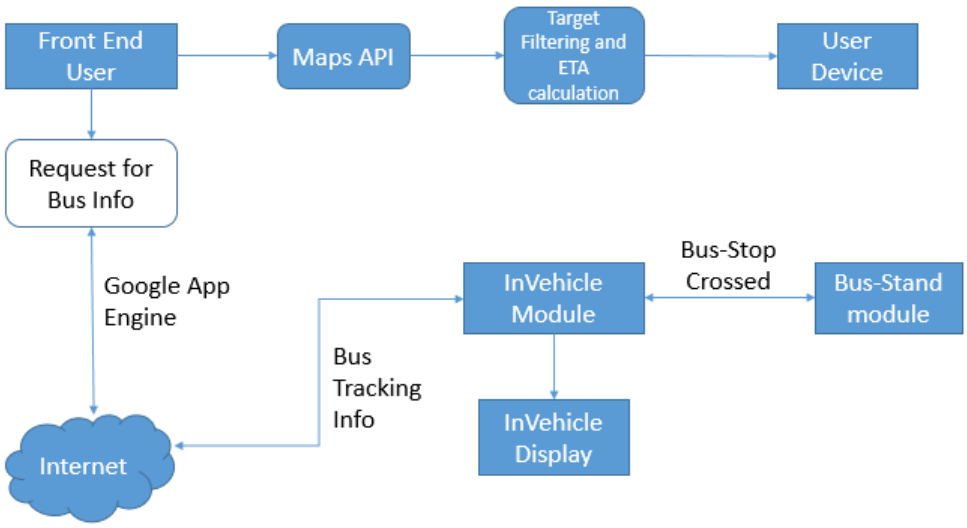

Figure 8.1: System Block Diagram

1. The User first requests for tracking information of a particular bus.

2. This request is then addressed by the Google App engine and it is trigerred to pull out data specific for that bus.

3. In the mean time the InVehicle module continuously collects data from the GPS modules and filters out all the relevant information needed. The Bus-Stand module also keeps the Invehicle module updated about the last bus stop crossed by any particular bus. All

the information is then consolidated by the InVehicle module and is transmitted to the Internet using the Google App Engine Back-End script.

4. Now all the relevant information requested is sent to the user.

5. This data which now includes GPS co-ordinates and a unique busID is now mapped on Google Maps using the Maps API.

6. Using an appropriate algorithm, ETA is calculated.The ETA is calculated with the help of the speed of the bus which is transmitted by the GPS module.

7. This data is then displayed on the user device.

8. Also, there is a display mounted in the bus which gives information about the next scheduled stop and ETA for the same.

Chapter 9

Conclusion

The proposed Intelligent Public Transport Monitoring System combines positioning hardware, and a communications platform that can be used to monitor and track a public transport in real time. This enables the transport management centres to observe, collect and analyze location information about the vehicle in real time. This data will give the users the ability to make better and more informed decisions while also providing quicker response to emergencies. The benefits to the passengers mean better on time performance and less waiting time at bus stops.

The Future scope of this Project includes combining all the separate modules to have a one single board which includes RaspberryPi, GPS and GSM. This not only reduces the cost but also makes the whole system miniaturized. Next up, the ETA calculation can be refined by considering the time taken by the last buses that travelled along the same route. According to the time taken by buses between two bus-stands, traffic between these places can be estimated which in itself is a important information for traffic management. The system can then also be made much more mobile-friendly including alarms when the desired bus comes in close to boarding stop of the user.

Hence, a basic Vehicle Tracking system was developed using GPS and ZigBee which can then be converted into a full-fledged public transport and traffic management system by subsequent work in similar direction.

Bibliography

1. ”GPS Based Automatic Vehicle Tracking Using RFID ”
   International Journal of Engineering and InnovativeTechnology (IJEIT) Volume 1, Issue 1, Pages 31-35
   Devyani Bajaj and Neelesh Gupta,(2012)

2. ”Public Transportation Management Service using GPS-GSM”
   International Journal of Research in Computer and Communication Technology, IJRCCT, ISSN-2278-5841, Vol-1, Issue -3, Aug - 2012
   G. Kiran Kumar1, Dr.A. Mallikarjuna Prasad

3. ”The Flexible Public Transport System Using GSM as a Communication Medium”
   IACSIT International Journal of Engineering and Technology, Vol. 4, No. 2, April 2012
   Mahesh Sachidanandam and Sivakumar P.

4. ”Wireless Communication Alternatives for Intelligent Transportation Systems: A Case Study ”
   Journal of Intelligent Transportation Systems: Technology, Planning,and Operations ,Volume 15, Issue 3,pages 147-160.
   Yan Zhou, Glenn Hamilton Evans Jr.,Mashrur Chowdhury, Kuang-Ching WangRyan Fries, (2011)

5. ”The Flexible Bus Systems Using Zigbee as a Communication Medium”
   IEEE (2011)
   Razi Iqbal, Yukimatsu Kenichi, Tatsuya Ichikawa

6. ”GPS-GSM Integration for Enhancing Public Transportation Management Services”
   2010 Second International Conference on Computer Engineering and Applications
   Umar Farooq, Tanveer ul Haq, Muhammad Amar, Muhammad Usman Asad, Asim Iqbal

7. ”Intelligent location models for next generation location-based services ”
   Journal of Location Based Services ,Volume 1, Issue 4,pages 237-255
   Allison Kealy,Stephan Winter Gnther Retscher,(2007)

8. ”Vehicle Location Technologies in AutomaticVehicle Monitoring and Management Systems ”
IVHSJournal, Volume 1, Issue 3, Pages 295-303.
John M. Watje, Denis SymesROBERT S. OW,(2007),