CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

Android is made up of several necessary and dependent parts, including the following:

A hardware reference design that describes the capabilities required for a mobile device to support the software stack.A Linux operating system kernel that provides low-level interface with the hardware,memory management, and process control, all optimized for mobile devices.Open-source libraries for application development, including SQLite,WebKit,OpenGL,and a media manager.A run time used to execute and host Android applications, including the Dalvik virtual machine and the core libraries that provide Android-specific functionality. The run time is designed to be small and efficient for use on mobile devices. An application framework that agnostically exposes system services to the application layer, including the window manager and location manager, content providers, telephony, and sensors. A user interface framework used to host and launch applications. Preinstalled applications shipped as part of the stack.

1.2 **Scope Of The Project**

Buses are provided by the Government as a public service, quality of which will directly determine the convenience of public travel. It is an important criterion for qualify of service standards that bus reaches the station on time and reports which station it is located accurately. Presently at the original station and terminal station, punctuality can be guaranteed because of dedicated staffs on duty there. However, for most of the middle stations, punctuality can not be guaranteed and also, it is difficult to be assessed. It might be a good idea using the GPS system for monitoring the bus when moving, but to expand the scope

of GPS usage is actually difficult due to the high cost of GPS system. For the time being, reporting bus station is to rely on driver's manual operation, so making a mistake and misleading passengers is inevitable when driving the bus. To solve that, paper provides a solution; it develops a system of bus monitoring and management based on GPRS technology. GPRS is short for General Packet Radio-Service; GPRS uses packet switching technology, especially suitable for intermittent, sudden and frequent, small amounts of data transfer. The system paper designed bases on GPRS technology will play a good effect from many aspects

1.3 OBJECTIVE OF THE PROJECT

This objective to monitor the bus using the centralized server and get the bus information using Android Smartphone.

1.4 **EXISTING SYSTEM:**

In the existing system, there is no tracking of Buses happening. GPS based Vehicle is only the solution but still arrival Timing of the Buses are not intimated to the bus stop. traffic system mainly depends on driver's manual operation, such as punctuality of the bus's arrival on bus station.

1.5 PROPOSED SYSTEM

The proposed model, Zigbbe is attachéd with the bus and another Zigbee is attached with the Bus Stop. The Bus Number and the Route is intimated to the Bus stop by the bus during it's arrival and the Stop name is intimated to the bus from the Bus stop.

1.6 MODIFICATION:

In the modification, as Zigbee is costly to implement, we modify the same process in a prototype manner with Graphical Path Virtualisation. Once the Bus starts from the Bus Depot it intimates to the nearest Bus Stop as it is approaching, Android Mobile user can send the request of his / her Source and

Destination of the Route so that the Server will identify the Nearest bus and the Time taken for the us bus to reach the requested stop. So that the Mobile user can plan his / her Travel according to the timing of the arrival of bus.

1.7 LITERATURE SURVEY

IEEE 802.15.4 LOW RATE –WIRELESS PERSONAL AREA NETWORK COEXISTENCE ISSUES

IEEE 802.15.4 is a proposed standard addressing the needs of low-rate wireless personal area networks or LR-WPAN with a focus on enabling wireless sensor networks. The standard is characterized by maintaining a high level of simplicity, allowing for low cost and low power implementations. Its operational frequency band includes the 2.4GHz industrial, scientific and medical band providing nearly worldwide availability; additionally, this band is also used by other IEEE 802 wireless standards. Coexistence among diverse collocated devices in the 2.4 GHz band is an important issue in order to ensure that each wireless service maintains its desired performance requirements. This paper presents a brief technical introduction of the IEEE 802.15.4 standard and analyzes the coexistence impact of an IEEE 802.15.4 network on the IEEE 802.11b devices.

DESIGN AND IMPLEMENTATION OF A WIRELESS SENSOR NETWORK FOR SMART HOMES

Wireless sensor networks (WSNs) have become indispensable to the realization of smart homes. The objective of this paper is to develop such a WSN that can be used to construct smart home systems. The focus is on the design and implementation of the wireless sensor node and the coordinator based on ZigBee technology. A monitoring system is built by taking advantage of the GPRS network. To support multi-hop communications, an improved

routing algorithm based on the Dijkstra algorithm is presented. Preliminary simulations have been conducted to evaluate the performance of the algorithm.

A JOINT MODEL FOR IEEE 802.15.4 PHYSICAL AND MEDIUM ACCESS CONTROL LAYERS

Many studies have tried to evaluate wireless networks and especially the IEEE 802.15.4 standard. Hence, several papers have aimed to describe the functionalities of the physical (PHY) and medium access control (MAC) layers. They have highlighted some characteristics with experimental results and/or have attempted to reproduce them using theoretical models. In this paper, we use the first way to better understand IEEE 802.15.4 standard. Indeed, we provide a comprehensive model, able more faithfully to mimic the functionalities of this standard at the PHY and MAC layers. We propose a combination of two relevant models for the two layers. The PHY layer behavior is reproduced by a mathematical framework, which is based on radio and channel models, in order to quantify link reliability. On the other hand, the MAC layer is mimed by an enhanced Markov chain. The results show the pertinence of our approach compared to the model based on a Markov chain for IEEE 802.15.4 MAC layer. This contribution allows us fully and more precisely to estimate the network performance with different network sizes, as well as different metrics such as node reliability and delay. Our contribution enables us to catch possible failures at both layers

SECURE LOW COST AMR SYSTEM BASED ON GPRS TECHNOLOGY

This project presents the design and implementation of a secure low cost automatic meter reading (AMR) system that measures and transmits the total electrical energy consumption to main server using general packet radio

service (GPRS) technology provided by GSM networks. The proposed AMR system consists of three main parts: Accurate digital meter, a transmission facility and the billing server. To make affordable AMR system a low cost off-the-shelf materials are used. Successful demonstration of the system prototype has made it possible to be implemented in the kingdom of Bahrain and other Middle East countries on a larger scale for meter reading applications.

CHAPTER 2

REQUIREMENTS ANALYSIS

Requirement analysis determines the requirements of a new system. This project analyses on product and resource requirement, which is required for this successful system. The product requirement includes input and output requirements it gives the wants in term of input to produce the required output. The resource requirements give in brief about the software and hardware that are needed to achieve the required functionality.

2.1 HARDWARE REQUIREMENTS

Operating system	:	Windows Xp
Processor	:	Intel Core 2 Duo processor (2.2 GHz)
Hard disk	:	160 GB
RAM	:	2 GB DDR

2.2 SOFTWARE REQUIREMENTS

Platform	:Windows Xp,
Front End	: Java JDK1.5.
Back End	:MS SQL server 2000

2.3 TECHNOLOGIES USED

2.3.1 R.java, Resources and Assets

The directory "gen" in an Android project contains generated values. "R.java" is a generated class which contains references to resources of the "res" folder in the project. These resources are defined in the "res" directory and can be values, menus, layouts, icons or pictures or animations. For example a resource can be an image or an XML files which defines stringsIf you create a new resources, the corresponding reference is automatically created in "R.java". The references are static int values, the Android system provides methods to

access the corresponding resource. For example to access a String with the referenceid"R.string.yourString"use the method getString(R.string.yourString)); Please do not try to modify "R.java" manually.While the directory"res" contains structured values which are known to the Android platform the directory "assets" can be used to store any kind of data. In Java you can access this data via the AssetsManager and the method getAssets().

2.3.2 Android Operation System

Android is an operating system based on Linux with a Java programming interface. It provides tools, e.g. a compiler, debugger and a device emulator as well as its own Java Virtual machine (Dalvik Virtual Machine - DVM). Android is created by the Open Handset Alliance which is lead by Google.

Android uses a special virtual machine, e.g. the Dalvik Virtual Machine. Dalvik uses special bytecode. Therefore you cannot run standard Java bytecode on Android. Android provides a tool "dx" which allows to convert Java Class files into "dex" (Dalvik Executable) files. Android applications are packed into an .apk (Android Package) file by the program "aapt" (Android Asset Packaging Tool) To simplify development Google provides the Android Development Tools (ADT) for Eclipse . The ADT performs automatically the conversion from class to dex files and creates the apk during deployment.Android supports 2-D and 3-D graphics using the OpenGL libraries and supports data storage in a SQLitedatabase.

Every Android applications runs in its own process and under its own userid which is generated automatically by the Android system during deployment. Therefore the application is isolated from other running

applications and a misbehaving application cannot easily harm other Android applications.

Important Android components

An Android application consists out of the following parts:

- Activity - Represents the presentation layer of an Android application, e.g. a screen which the user sees. An Android application can have several activities and it can be switched between them during runtime of the application.
- Views - The User interface of an Activities is build with widgets classes which inherent from "android.view.View". The layout of the views is managed by "android.view.ViewGroups".
- Services - perform background tasks without providing an UI. They can notify the user via the notification framework in Android.
- Content Provider - provides data to applications, via a content provider your application can share data with other applications. Android contains a SQLite DB which can serve as data provider
- Intents are asynchronous messages which allow the application to request functionality from other services or activities. An application can call directly a service or activity (explicit intent) or ask the Android system for registered services and applications for an intent (implicit intents). For example the application could ask via an intent for a contact application. Application register themself to an intent via an IntentFilter. Intents are a powerful concept as they allow to create loosely coupled applications.
- Broadcast Receiver - receives system messages and implicit intents, can be used to react to changed conditions in the system. An application can register as a broadcast receiver for certain events and can be started if such an event occurs.

Other Android parts are Android Widgets or Live Folders and Live Wallpapers . Live Folders display any source of data on the homescreen without launching the corresponding application.

Security and permissions

Android defines certain permissions for certain tasks. For example if the application want to access the Internet it must define in its configuration file that it would like to use the related permission. During the installation of an Android application the user get a screen in which he needs to confirm the required permissions of the application.

AndroidManifest.xml

An Android application is described the file "AndroidManifest.xml". This file must declare all activities, services, broadcast receivers and content provider of the application. It must also contain the required permissions for the application. For example if the application requires network access it must be specified here. "AndroidManifest.xml" can be thought as the deployment descriptor for an Android application.

The "package" attribute defines the base package for the following Java elements. It also must be unique as the Android Marketplace only allows application for a specfic package once. Therefore a good habit is to use your reverse domain name as a package to avoid collisions with other developers.

"android:versionName" and "android:versionCode" specify the version of your application. "versionName" is what the user sees and can be any string. "versionCode" must be an integer and the Android Market uses this to determine if you provided a newer version to trigger the update on devices

which have your application installed. You typically start with "1" and increase this value by one if you roll-out a new version of your application.

"activity" defines an activity in this example pointing to the class "de.vogella.android.temperature.Convert". For this class an intent filter is registered which defines that this activity is started once the application starts (action android:name="android.intent.action.MAIN"). The category definition (category android:name="android.intent.category.LAUNCHER") defines that this application is added to the application directory on the Android device. The @ values refer to resource files which contain the actual values. This makes it easy to provide different resources, e.g. strings, colors, icons, for different devices and makes it easy to translate applications.

The "uses-sdk" part of the "AndroidManifest.xml" defines the minimal SDK version your application is valid for. This will prevent your application being installed on devices with older SDK versions.

Activities and Lifecyle

The operating system controls the life cycle of your application. At any time the Android system may stop or destroy your application, e.g. because of an incoming call. The Android system defines a life cycle for an activities via pre-defined methods. The most important methods are:

- onSaveInstanceState() - called if the activity is stopped. Used to save data so that the activity can restore its states if re-started
- onPause() - always called if the Activity ends, can be used to release ressource or save data
- onResume() - called if the Activity is re-started, can be used to initiaze fields

The activity will also be restarted if a so called "configuration change" happens. A configuration change for examples happens if the user changes the orientation of the device (vertical or horizontal). The activity is in this case restarted to enable the Android platform to load different resources for these configuration, e.g. layouts for vertical or horizontal mode. In the emulator you can simulate the change of the orientation via CNTR+F11.

1.8. Context

The class android.content.Context provides the connections to the Android system. It is the interface to global information about the application environment. Context also provides the method which allows to receive Android services, e.g. the Location Service . As Activities and Services extend the class "Context" you can directly access the context via "this".

User Interface

In an Android application, the user interface is built using Viewand ViewGroup objects. There are many types of views and view groups, each of which is a descendant of the View class.

View objects are the basic units of user interface expression on the Android platform. The View class serves as the base for subclasses called "widgets," which offer fully implemented UI objects, like text fields and buttons. The ViewGroup class serves as the base for subclasses called "layouts," which offer different kinds of layout architecture, like linear, tabular and relative.

A View object is a data structure whose properties store the layout parameters and content for a specific rectangular area of the screen. A View object handles its own measurement, layout, drawing, focus change, scrolling, and key/gesture interactions for the rectangular area of the screen in which it resides. As an

object in the user interface, a View is also a point of interaction for the user and the receiver of the interaction events.

View Hierarchy

On the Android platform, you define an Activity's UI using a hierarchy of View and ViewGroup nodes, as shown in the diagram below. This hierarchy tree can be as simple or complex as you need it to be, and you can build it up using Android's set of predefined widgets and layouts, or with custom Views that you create yourself.

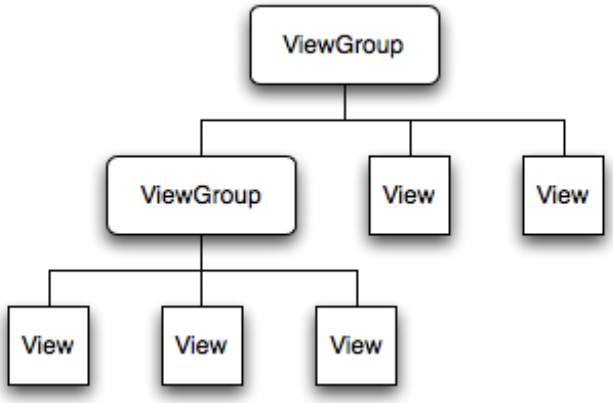


Fig :Hierarchy

In order to attach the view hierarchy tree to the screen for rendering, your Activity must call the setContentView() method and pass a reference to the root node object. The Android system receives this reference and uses it to invalidate, measure, and draw the tree. The root node of the hierarchy requests that its child nodes draw themselves — in turn, each view group node is responsible for calling upon each of its own child views to draw themselves. The children may request a size and location within the parent, but the parent object has the final decision on where how big each child can be. Android parses the elements of your layout in-order (from the top of the hierarchy tree), instantiating the Views and adding them to their parent(s). Because these are

drawn in-order, if there are elements that overlap positions, the last one to be drawn will lie on top of others previously drawn to that space.

For a more detailed discussion on how view hierarchies are measured and drawn, read How Android Draws Views.

Layout

The most common way to define your layout and express the view hierarchy is with an XML layout file. XML offers a human-readable structure for the layout, much like HTML. Each element in XML is either a View or ViewGroup object (or descendant thereof). View objects are leaves in the tree, ViewGroup objects are branches in the tree (see the View Hierarchy figure above).

The name of an XML element is respective to the Java class that it represents. So a <TextView> element creates aTextView in your UI, and a <LinearLayout> element creates a LinearLayout view group. When you load a layout resource, the Android system initializes these run-time objects, corresponding to the elements in your layout.

For example, a simple vertical layout with a text view and a button looks like this:

<?xmlversion="1.0"encoding="utf-8"?>
<LinearLayoutxmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="fill_parent"
android:layout_height="fill_parent"

2.3.3 MS SQL Server 2000

Microsoft SQL Server 2000 is a full-featured relational database management system (RDBMS) that offers a variety of administrative tools to ease the burdens of database development, maintenance and administration. In this article, we'll cover six of the more frequently used tools: Enterprise Manager, Query Analyzer, SQL Profiler, Service Manager, Data Transformation Services and Books Online. Let's take a brief look at each:

Enterprise Manager is the main administrative console for SQL Server installations. It provides you with a graphical "birds-eye" view of all of the SQL Server installations on your network. You can perform high-level administrative functions that affect one or more servers, schedule common maintenance tasks or create and modify the structure of individual databases.

Query Analyzer offers a quick and dirty method for performing queries against any of your SQL Server databases. It's a great way to quickly pull information out of a database in response to a user request, test queries before implementing them in other applications, create/modify stored procedures and execute administrative .

SQL Profiler provides a window into the inner workings of your database. You can monitor many different event types and observe database performance in real time. SQL Profiler allows you to capture and replay system "traces" that log various activities.

Service Manager is used to control the MS SQL Server (the main SQL Server process), MSDTC (Microsoft Distributed Transaction Coordinator) and SQLServerAgent processes.

Data Transformation Services (DTS) provide an extremely flexible method for importing and exporting data between a Microsoft SQL Server installation and a large variety of other formats.

Books Online is an often overlooked resource provided with SQL Server that contains answers to a variety of administrative, development and installation issues. It's a great resource to consult before turning to the Internet or technical support.

CHAPTER 3

SYSTEM DESIGN

This chapter describes the overall and the detailed architectural design. It also describes each module that is to be implemented in first phase.

UML DIAGRAMS:

UML is simply anther graphical representation of a common semantic model. UML provides a comprehensive notation for the full lifecycle of object-oriented development.

ADVANTAGES

- To represent complete systems (instead of only the software portion) using object oriented concepts
- To establish an explicit coupling between concepts and executable code
- To take into account the scaling factors that are inherent to complex and critical systems
- To creating a modeling language usable by both humans and machines

UML defines several models for representing systems

- The class model captures the static structure
- The state model expresses the dynamic behavior of objects
- The use case model describes the requirements of the user
- The interaction model represents the scenarios and messages flows
- The implementation model shows the work units
- The deployment model provides details that pertain to process allocation

DATA FLOW DIAGRAM:

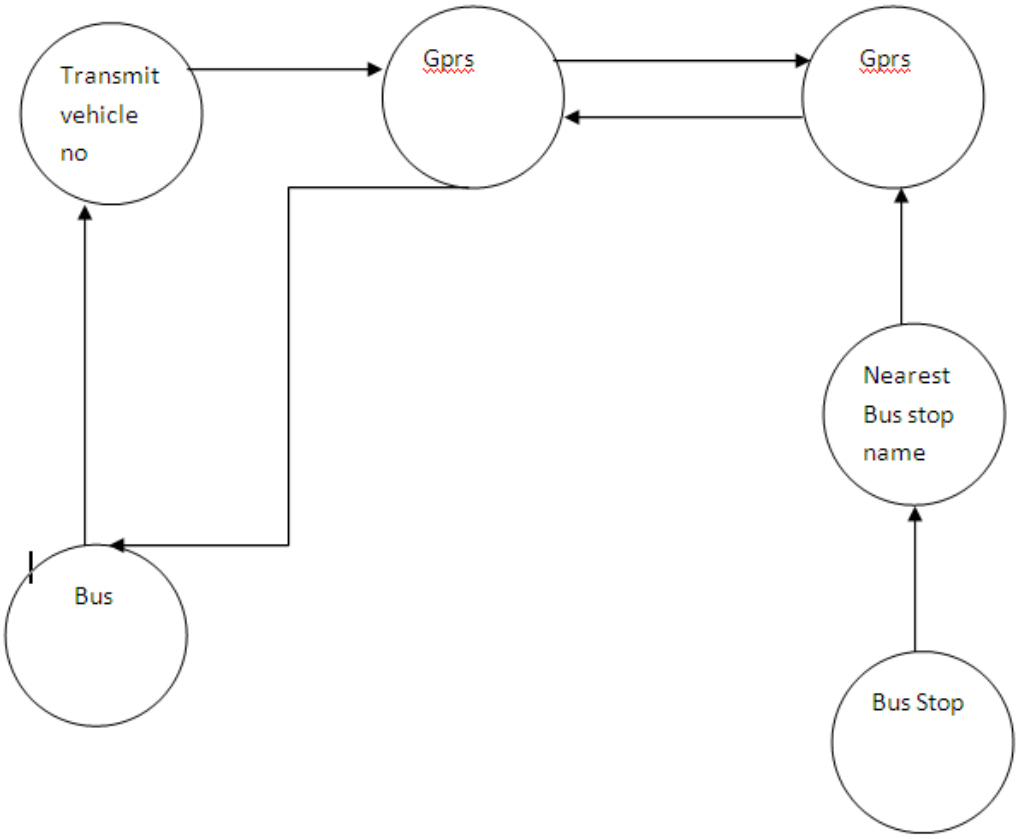


Fig:3.0 Dataflow Diagram

A modular design reduces complexity, facilities change (a critical aspect of software maintainability), and results in easier implementation by encouraging parallel development of different part of system. Software with effective modularity is easier to develop because function may be compartmentalized and interfaces are simplified. Software architecture embodies modularity that is software is divided into separately named and addressable components called modules that are integrated to satisfy problem requirements. Modularity is the single attribute of software that allows a program to be intellectually manageable.

3.1 System Architecture

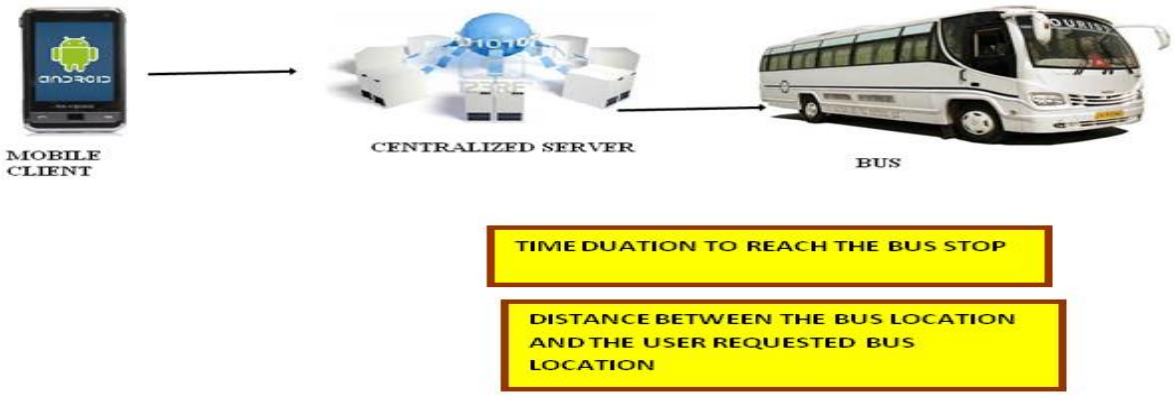


Fig:3.1 System Architecture

The five important criteria that enable us to evaluate a design method with respect to its ability to define an effective modular design are: Modular decomposability, Modular Comps ability, Modular Understandability, Modular continuity, Modular Protection.(fig:3.1)The following are the modules of the project, which is planned in aid to complete the project with respect to the proposed system, while overcoming existing system and also providing the support for the future enhancement.

3.2 Use Case Diagram

Use case diagrams overview the usage requirement for system. they are useful for presentations to management and/or project stakeholders, but for actual development you will find that use cases provide significantly more value because they describe “the meant” of the actual requirements.(fig:3.2) A use case describes a sequence of action that provide something of measurable value to an action and is drawn as a horizontal ellipse.

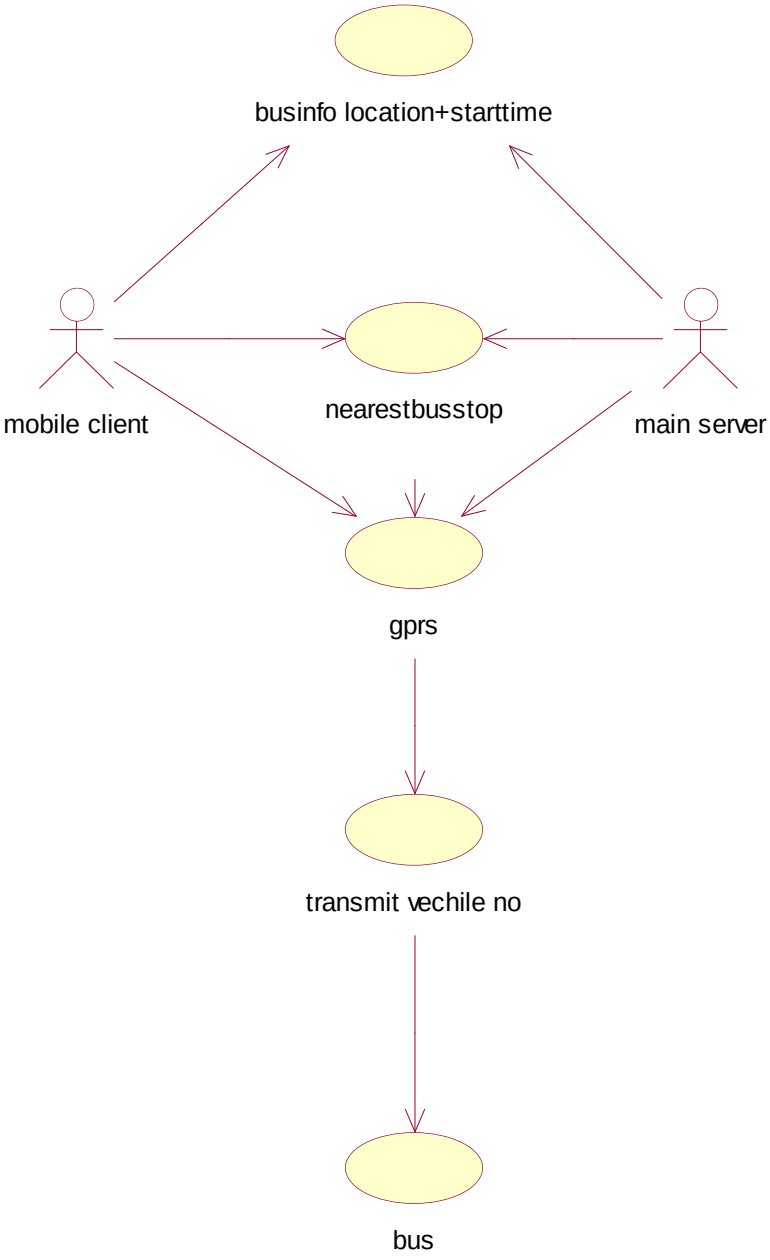


Fig:3.2 Use Case Diagram

- The class model captures the static structure
- The state model expresses the dynamic behavior of objects
- The use case model describes the requirements of the user
- The interaction model represents the scenarios and messages flows

- The implementation model shows the work units
- The deployment model provides details that pertain to process allocation

3.3 Sequence Diagram

Sequence diagram model the flow of logic within your system in a visual manner, enabling you both to document and validate your logic, and commonly used for both analysis and design purpose.(fig:3.3) Sequence diagram are the most popular UML artifact for dynamic modeling, which focuses on identifying the behavior within your system.

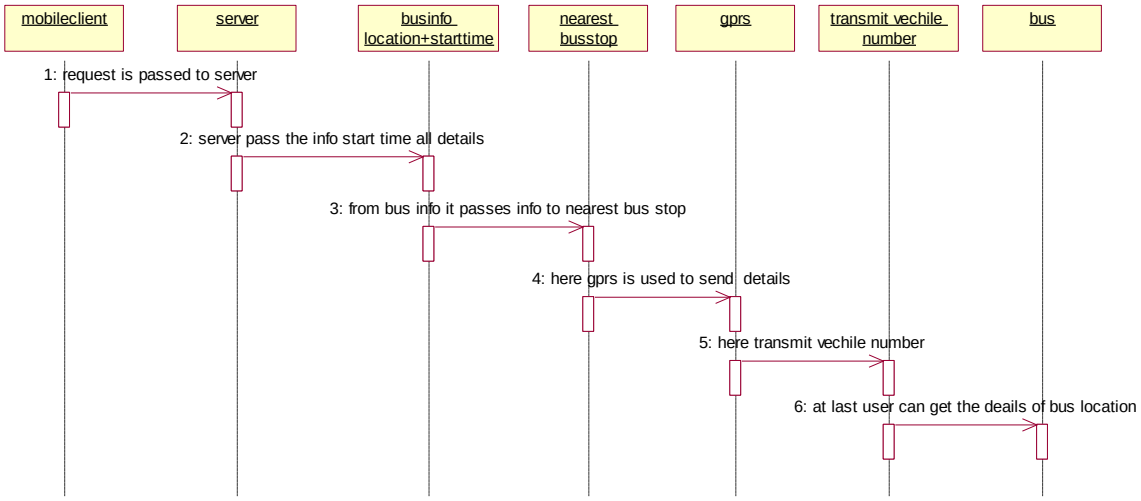


Fig:3.3 Sequence Diagram

3.4 Collaboration Diagram

Another type of interaction diagram is the collaboration diagram. A collaboration diagram represents a collaboration**, (fig:3.4)**which is a set of objects related in a particular context, and interaction, which is a set of messages exchange among the objects within the collaboration to achieve a desired outcome.

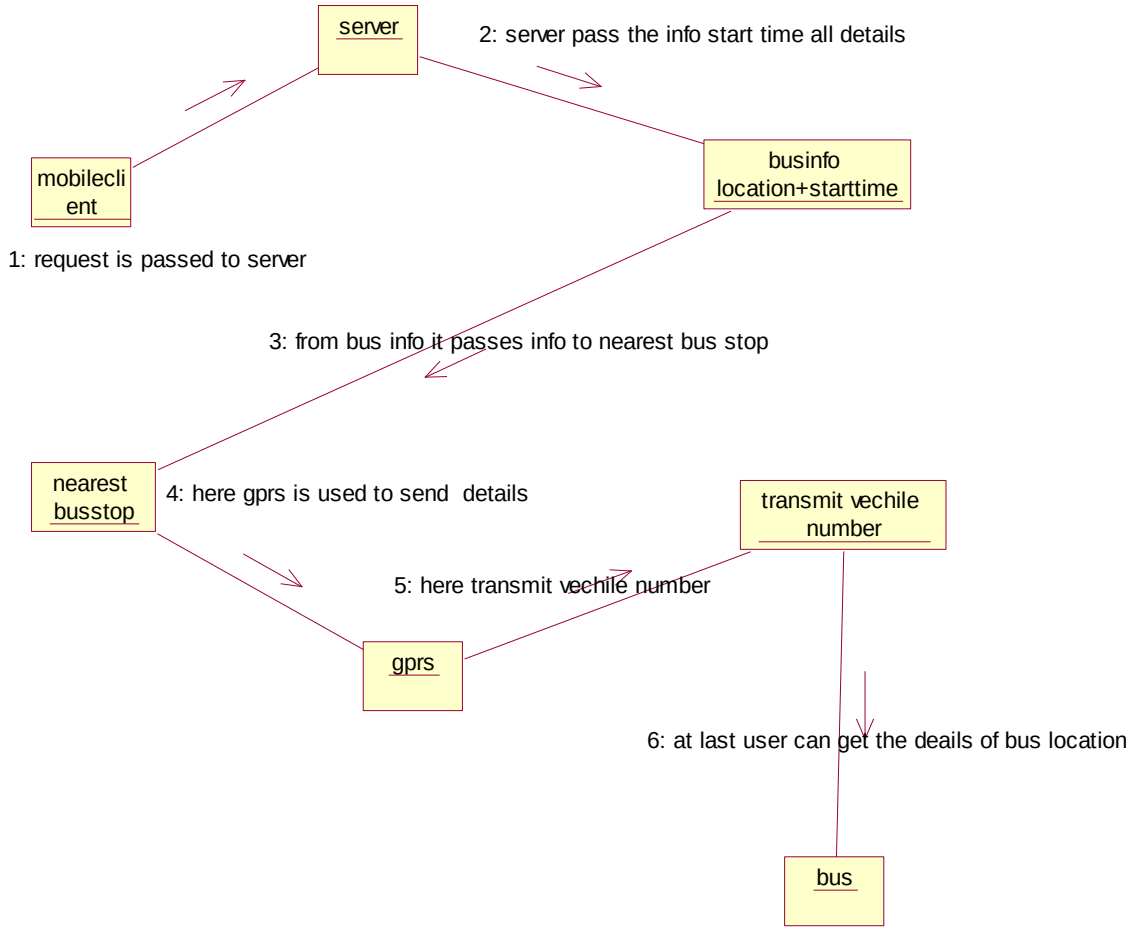


Fig:3.4 Collaboration Diagram

3.5 Activity Diagram

Activity diagram are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. The activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system.(fig:3.5) Activity diagram consist of Initial node, activity final node and activities in between.

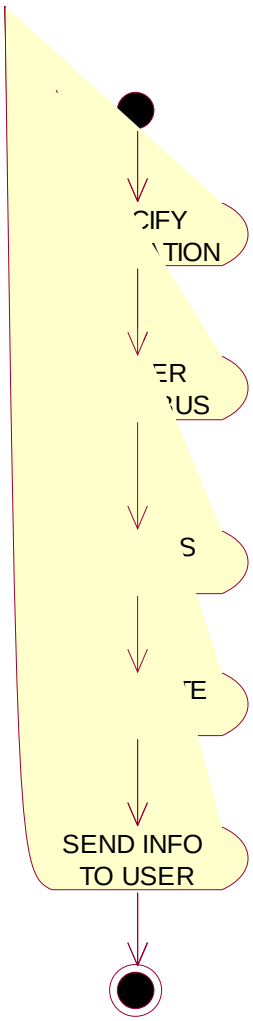


Fig:3.5 Activity Diagram

CHAPTER 4

IMPLEMENTATION AND TESTING

4.1 Implementation and Modules

MODULES:

A modular design reduces complexity, facilities change (a critical aspect of software maintainability), and results in easier implementation by encouraging parallel development of different part of system. Software with effective modularity is easier to develop because function may be compartmentalized and interfaces are simplified. Software architecture embodies modularity that is software is divided into separately named and addressable components called modules that are integrated to satisfy problem requirements.

Modularity is the single attribute of software that allows a program to be intellectually manageable. The five important criteria that enable us to evaluate a design method with respect to its ability to define an effective modular design are: Modular decomposability, Modular Comps ability, Modular Understandability, Modular continuity, Modular Protection.

The following are the modules of the project, which is planned in aid to complete the project with respect to the proposed system, while overcoming existing system and also providing the support for the future enhancement.

MODULE DECOMPOSITION

This project entails four main modules that interoperate to form the entire system:

- Server Module
- Client Module (Mobile Terminal)

- Bus Location Application Module
- Database & Mobile Terminal Connectivity Module

Server Module

Server module is directly connected to the database where the vehicle's information such as the place and exact area will be processed and saved in the database. Commuter communicates to the server only via their Android mobile phones.

The Database is connected to the client that is the passenger's ANDROID mobile and the request will also processed with the records in the database of the server.

Server module plays a vital role in communicating with the vehicle and also with the passenger's ANDROID mobile as to process the request to send the proper information back to the passenger.

Implementation

The server module is developed with JSP and Servlet. It is an efficient management framework along with real time communications and other telephony protocols.

Servlet

Apache Tomcat version 6.0 is the servlet container that is used in the official Reference Implementation for the Java Servlet and Java Server pages technologies. We use JCreator. Applications would be developed as separate web applications (preferably deployed on Tomcat) as a composition of services.

Server module is directly connected to the database. Vehicle's information such as the place and exact area will be processed and saved in the database. Commuter communicates to the server only via their ANDROID mobile phones. Server Module processes Request messages and sends Reply messages to the Client module (ANDROID enabled).

Client

The client part is the Android which will be with the passenger. After getting the Start stop from the passenger the request will be send to the server to process the data input. The server will fetch the exact result of the nearest vehicle from the passenger start place and that will be sent as a reply to the passenger. Then the passenger sends a request to the server to know the time taken and distance between their current position and nearest available vehicle.

This information will also be processed in the server and sent back to the passenger's ANDROID mobile toolkit as reply, so that the passenger can take a decision to wait until the next vehicle comes or take some other source to move on

Implementation

The client part is the ANDROID which will be with the passenger.

Emulator

The emulator provided by the SDK is compatible and closely emulates the operation of a real cell phone. The PC mouse is used to interact with the emulator whereas the cursor keys are used to navigate through the displays.

It obtains the Start-Stop terminals from the passenger - Request message - & it will be sent to the server to process the data. The server will fetch the exact result of the nearest vehicle from the passenger start place and is sent as Reply message to mobile phone. After debugging and compiling code in IDE, we obtain a MIDlet application program, which can be installed on the real mobile phone by wireless or cable download.

Bus location application module

It is a program developed in JAVA Swing technology which displays an animated graphical image of a PTC Bus on a road. Each PTC Bus object will communicate the server and inform the location of PTC Bus moment for each second. In our project we program three such buses on a road map.

Implementation

It is a program developed in JAVA Swing technology.

ANDROID

The ANDROID is a complete development environment. Swing is an application for producing GUI application for simulation of Bus moving along a road and for simultaneous intimation of Bus location to the server. Each PTC Bus object will communicate the server and inform the location of PTC Bus moment for each second.

The communication to the server is done through GPRS. This application streams the data through http protocol and sends the value of the current location, speed of the PTC Bus to the server.

Database & Mobile Terminal Connectivity Module

It is the most basic connection type & can only be opened and closed.

1. InputConnection interface:

- The InputConnection interface represents a device from which data is read.
- Its **openInputStream** method returns an input stream for the connection.

2. OuputConnection interface:

- The OuputConnection interface represents a device to which data is written.

- Its o**penOutputStream** method returns an output stream for the connection.

Implementation

The DB connectivity used in this project is described below.

DB Module Connectivity

The application used here to connect our database to server is Java Database Connectivity (JDBC). JDBC is based on the X/Open SQL call-level interface specifications. JDBC application can be represented in a three-tier model - commands are sent to a middle tier of services, which then sends the commands to the data source.

The reason we chose JDBC was because it proved to be a more up-to-date version and would work better with MS – Access.

Testing is a set of activities that can be planned in advance and conducted systematically. For this reason a template for software testing, a set of steps into which we can place specific test case design techniques and testing methods should be defined for software process.

4.2 Testing

Testing often accounts for more effort than any other software engineering activity. If it is conducted haphazardly, time is wasted, unnecessary effort is expanded, and even worse, errors sneak through undetected. It would therefore seem reasonable to establish a systematic strategy for testing software

4.3 Type of Testing

There are two type of testing according their behaviors

I .Unconventional Testing

II.Conventional Testing

4.3.1 Unconventional Testing

Unconventional testing is a process of verification which is doing by SQA (Software Quality Assurance) team. It is a prevention technique which is performing from beginning to ending of the project development. In this process SQA team verifying the project development activities and insuring that the developing project is fulfilling the requirement of the client or not.

In this testing the SQA team follows these methods:

1. Peer review
2. Code walk and throw
3. Inspection
4. Document Verification

4.3.2 Convensional Testing

Conventional Testing is a process of finding the bugs and validating the project. Testing team involves in this testing process and validating that developed project is according to client requirement or not. This process is a correction technique where testing team find bugs and reporting to the development team for correction on developed project built.

4.4 Methodologies :

4.4.1 Unit Testing

The procedure level testing is made first. By giving improper inputs, the errors occurred are noted and eliminated. Then the web form level testing is made. For example storage of data to the table in the correct manner.

In the company as well as seeker registration form, the zero length username and password are given and checked. Also the duplicate username is given and checked. In the job and question entry, the button will send data to the server only if the client side lidations are made.

The dates are entered in wrong manner and checked. Wrong email-id and web site URL (Universal Resource Locator) is given and checked.

4.4.2 Integration Testing

Testing is done for each module. After testing all the modules, the modules are integrated and testing of the final system is done with the test data, specially designed to show that the system will operate successfully in all its aspects conditions. Thus the system testing is a confirmation that all is correct and an opportunity to show the user that the system works.

4.4.3 Module Testing:

Module Testing is a process of testing the system, module by module. It includes the various inputs given, outputs produced and their correctness. By testing in this method we would be very clear of all the bugs that have occurred.

4.4.4 Interface Testing:

The Interface Testing is performed to verify the interfaces between sub modules while performing integration of sub modules aiding master module recursively.

4.4.5 Validation Testing:

The final step involves Validation testing, which determines whether the software function as the user expected. The end-user rather than the system developer conduct this test most software developers as a process called "Alpha and Beta Testing" to uncover that only the end user seems able to find.

The compilation of the entire project is based on the full satisfaction of the end users. In the project, validation testing is made in various forms.

4.5 MAINTENANCE

The objectives of this maintenance work are to make sure that the system gets into work all time without any bug. Provision must be for environmental changes which may affect the computer or software system. This is called the maintenance of the system. Nowadays there is the rapid change in the software world. Due to this rapid change, the system should be capable of adapting these changes. In our project the process can be added without affecting other parts of the system.Maintenance plays a vital role. The system liable to accept any

modification after its implementation. This system has been designed to favour all new changes. Doing this will not affect the system's performance or its accuracy.

4.5.1 Testing Strategies

A number of software testing strategies have been proposed in the literature. All provide the software developer with a template for testing and all have the following generic characteristics:

- Testing begins at the component level and works "outward" toward the integration of the entire computer-based system.
- Different testing techniques are appropriate at different points in time.
- The developer of the software conducts testing and for large projects, independent test group.
- Testing and debugging are different activitccommodated in any testing strategy.

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 CONCLUSION

The application of GPRS technology to the bus monitoring system, can solve many problems. In this way, bus service quality and operational efficiency will improved a lot. So the solution can play a great effect, of practically significant. GPRS technology used in the system can implement their respective advantages and disadvantages. Such kind of system model is also suitable for many other applications of industrial site.

5.2 FUTURE WORK

In future we can use real time GPS and track the exact Location of the bus, so that we may able to get reach the bus Stop more effectively.

APPENDIX 1 (SAMPLE CODING)

SAMPLE CODING:

```
public class BusMonitoringAct extends Activity {

   /** Called when the activity is first created. */

       HttpResponse response;

         HttpClient httpclient;

         String userMobile = null;

         String responseText;

   public void onCreate(Bundle savedInstanceState)

{

super.onCreate(savedInstanceState);

setContentView(R.layout.main);

       final EditText txt_uname = (EditText)findViewById(R.id.txt_uname);

       final EditText txt_pass = (EditText)findViewById(R.id.txt_pass);

       final EditText txt_ip = (EditText)findViewById(R.id.txt_ip);

        Button submit = (Button)findViewById(R.id.submit);

       Button reset = (Button)findViewById(R.id.reset);

       submit.setOnClickListener(new OnClickListener(){

       public void onClick(View view){
```

```
    if((txt_uname.getText().toString().intern()=="dhana")&&
(txt_pass.getText().toString().intern()== "karthik"))
{
        ServerIPAddress.setIpaddress(txt_ip.getText().toString());
        Toast.makeText(getApplicationContext(), "Logged in",10).show();
        Intent intent = new Intent(BusMonitoringAct.this,Menu.class);
        startActivity(intent);
         }else{
              Toast.makeText(getApplicationContext(), "Invalid user Name and
Password", Toast.LENGTH_SHORT).show();
             txt_uname.setText("");
              txt_pass.setText("");
              txt_ip.setText("");
         }
       }
      });
      reset.setOnClickListener(new OnClickListener(){
        public void onClick(View view){
        txt_uname.setText("");
        txt_pass.setText("");
```

```
            txt_ip.setText("");
        }
             });
            }
public class DestinationMenu extends ListActivity{
    static String text = "";
     HttpResponse response;
            HttpClient httpclient;
            String responseText;
            String message = "";
            int destination = 0;
            int source = 0;
    public void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
        ArrayList<String> al = new ArrayList<String>();
        al.add("STOP2");
        al.add("STOP3");
        al.add("STOP4");
        al.add("STOP5");
```

```
al.add("STOP6");
al.add("STOP7");
al.add("STOP8");
al.add("STOP9");
//al.add("STOP10");
al.add("Submit");
Log.v("Client List",al.toString());
setListAdapter(new ArrayAdapter<String>(this, R.layout.list_item, al));
ListView lv = getListView();
lv.setTextFilterEnabled(true);
lv.setOnItemClickListener(new OnItemClickListener() {
public void onItemClick(AdapterView<?> parent, View view,
int position, long id) {
String selectedText = ((TextView) view).getText().toString();
if(selectedText.intern() == "Submit"){
//Toast.makeText(getApplicationContext(),selectedText,10).show();
//Toast.makeText(getApplicationContext(),"Source
Destjava..."+ServerIPAddress.getSource(),Toast.LENGTH_LONG).show();
if(text.intern()== "STOP2"){
destination = 116;
```

```
}else if(text.intern()== "STOP3"){
        destination = 170;
}else if(text.intern()== "STOP4"){
        destination = 230;
}       else if(text.intern()== "STOP5"){
        destination = 371;
}       else if(text.intern()== "STOP6"){
        destination = 470;
}       else if(text.intern()== "STOP7"){
        destination = 503;
}       else if(text.intern()== "STOP8"){
        destination = 551;
}       else if(text.intern()== "STOP9"){
        destination = 666;
} if(ServerIPAddress.getSource().intern()== "STOP1"){
        source = 52;
} if(ServerIPAddress.getSource().intern()== "STOP2"){
        source = 116;
} if(ServerIPAddress.getSource().intern()== "STOP3"){
```

```
        source = 170;
}       if(ServerIPAddress.getSource().intern()== "STOP4"){
        source = 230;
}       if(ServerIPAddress.getSource().intern()== "STOP5"){
        source = 371;
}       if(ServerIPAddress.getSource().intern()== "STOP6"){
        source = 470;
}       if(ServerIPAddress.getSource().intern()== "STOP7"){
        source = 503;
}       if(ServerIPAddress.getSource().intern()== "STOP8"){
        source = 551;
}       if(ServerIPAddress.getSource().intern()== "STOP9"){
        source = 666;
}
 String uLoc = ""+source;
 String uDest = ""+destination;
 getConnection(uLoc,uDest);
}else{
        text = selectedText.trim();
```

```
                    Toast.makeText(getApplicationContext(),"Destination
selected...."+text,Toast.LENGTH_LONG).show();

                    ServerIPAddress.setDestination(text);

                }

        });

        }       private void getConnection(String source,String destination){

                try {

Toast.makeText(getApplicationContext(),destination,10).show();

Toast.makeText(getApplicationContext(),source,10).show();

                httpclient = new DefaultHttpClient();

        HttpGethttpget=newHttpGet("http://"+ServerIPAddress.getIpaddress()+":
8080/LBS/UserLoc?uLoc="+source+"&uDest="+destination);

                response = httpclient.execute(httpget);

                HttpEntity entity = response.getEntity();

                responseText = EntityUtils.toString(entity);

                Log.v("Exception",  responseText.toString());

Toast.makeText(getApplicationContext(),responseText.toString(),10).show();
```

```
            }
catch (Exception e)
{

Toast.makeText(this,"error"+e.toString(),Toast.LENGTH_LONG).show();
            }
        }
```

NEW BUS 1:

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.URL;
import java.net.URLConnection;
import java.util.Date;

/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.ImageIcon;
import javax.swing.JLabel;
import javax.swing.JPanel;

/**
 *
 * @author Administrator
 */
public class NewBus extends Thread{
    public static Thread t;
    String stat=null;
    String loce=null;
    String StartTimeB1;
    String RreachedTimeB1;
    int send;
    Date StartT;
    Date EndT;
    public ImageIcon icon1;
    public JLabel iconbus1;
    public JPanel root1;
    public int speed;

    NewBus(){
        Main.date=new Date();
        StartT=Main.date;
        int
starttime=(Main.date.getHours()*60*60)+Main.date.getMinutes()*60+Main.dat
e.getSeconds();
```

```
        StartTimeB1=starttime+"";
        icon1=new ImageIcon("../EconomicalLSB/src/image/busico.gif");
        iconbus1=new JLabel(icon1);
        root1=new JPanel();
        root1.add(iconbus1);
        root1.setBounds(43, 495, 8, 8);
        t = new Thread(this, "LSB");
        System.out.println("Start Time: " + StartTimeB1);
        t.start(); // Start the thread
}
// This is the entry point for the second thread.
    @Override
public void run()
 {
Try
 {
    for(int i=0;i<Main.X.length;i++){
      Main.date=new Date();
      setLocation(i);
      String hr=""+Main.date.getHours();
      String min=""+ Main.date.getMinutes();
      String sec=""+Main.date.getSeconds();
      Main.timelab.setText("Time:"+hr+":"+min+":"+sec);
      String loc=  getBusLocation(Main.X[i]);
      String status=getBusStatus(Main.X[i]);
      Main.locate.setText(loc);
      Main.loccros.setText(status);
      Thread.sleep(200);

}
}
 catch (InterruptedException e)
{
System.out.println("Child interrupted.");
}

}

    void setLocation(int i)
{
        Try
 {
            send++;
```

```
          Main.ob.root.setBounds(Main.X[i], Main.Y[i] - 30, 8, 8);
         if(send==20)
{
          URL yahoo = new
URL("http://"+Main.ipaddress.getText()+":"+Main.portno.getText()+"/LBS/BU
S1?currLoc="+Main.X[i]+"&speed=1");
          URLConnection yc = yahoo.openConnection();
          BufferedReader in = new BufferedReader(new
InputStreamReader(yc.getInputStream()));
          String inputLine;

          while ((inputLine = in.readLine()) != null)
{
            //System.out.println(inputLine);
          }
          in.close();
         send=0;
         }
        }
catch (IOException ex)
{
          Logger.getLogger(NewBus.class.getName()).log(Level.SEVERE, null,
ex);
      }
    }
private String getBusLocation(int i)
 {

       if(i==52)
{

      loce="stop1";
      }
     if(i==116)
{
       loce="stop2";
      }
       if(i==170)
{

      loce="stop3";
      }
        if(i==230)
```

```
{

      loce="stop4";
      }
       if(i==371)
{

      loce="stop5";
      }
    if(i==470)
{

      loce="stop6";
      }
       if(i==503)
{

      loce="stop7";
      }
        if(i==551)
{

      loce="stop 8";
      }
        if(i==666)
{

      loce="stop 9";
      }
       if(i==742)
{

      loce="stop 10";
      }
      return loce;
   }

   private String getBusStatus(int i)
 {
       if(i==81)
{
       stat="near stop 2";
       }
```

```
        if(i==143)
{
        stat="near stop 3";
        }
         if(i==196)
{
        stat="near stop 4 ";

        }
        if(i==300)
{
        stat="near stop 5";
        }
        if(i==436)
{
        stat="near stop 6";
        }
        if(i==488)
{
        stat="near stop 7";
        }
        if(i==513)
{
        stat="near stop 8";
        }

        if(i==604)
{
        stat="near stop 9";
        }
         if(i==700)
{
        stat="near stop 10";
        }
         if(i==724)
{
        stat="Reached stop 10";

    Main.fixbtn.setEnabled(true);

        Main.date=new Date();
        EndT=Main.date;
        System.out.println("Start time"+StartT);
```

```
        System.out.println("End time"+EndT);

        }
        return stat;
    }


}
```

NEW BUS 2:

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.URL;
import java.net.URLConnection;
import java.util.Date;

/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.ImageIcon;
import javax.swing.JLabel;
import javax.swing.JPanel;

/**
 *
 * @author Administrator
 */
public class NewBus2 extends Thread
{
    public static Thread t;
    String stat=null;
    String loce=null;
    String StartTimeB2;
    String RreachedTimeB2;
    int send;
    Date StartT;
    Date EndT;
    public ImageIcon icon1;
    public JLabel iconbus1;
```

```
   public JPanel root1;
   public int speed1;

   NewBus2()
{
       Main.date=new Date();
       StartT=Main.date;
       @SuppressWarnings("deprecation")
       int
starttime=(Main.date.getHours()*60*60)+Main.date.getMinutes()*60+Main.dat
e.getSeconds();
       StartTimeB2=starttime+"";
       icon1=new ImageIcon("../EconomicalLSB/src/image/busico.gif");
       iconbus1=new JLabel(icon1);
       root1=new JPanel();
       root1.add(iconbus1);
       root1.setBounds(43, 495, 8, 8);
       t = new Thread(this, "LSB");
       System.out.println("Start Time: " + StartTimeB2);
       System.out.println("Length of array"+Main.X.length);
       t.start(); // Start the thread
}
// This is the entry point for the second thread.
   @Override
public void run()
{
try
{
   for(int i=0;i<Main.X.length;i++)
{
     Main.date=new Date();
     setLocation(i);
           @SuppressWarnings("deprecation")
     String hr=""+Main.date.getHours();
           @SuppressWarnings("deprecation")
     String min=""+ Main.date.getMinutes();
           @SuppressWarnings("deprecation")
     String sec=""+Main.date.getSeconds();

     String loc=  getBusLocation(Main.X[i]);
     String status=getBusStatus(Main.X[i]);
     Main.locate2.setText(loc);
     Main.loccros2.setText(status);
```

```
     Thread.sleep(200);
}
}
catch (InterruptedException e)
{
System.out.println("Child interrupted.");
}

}

   void setLocation(int i)
{
        try
{
           send++;
          Main.ob.root2.setBounds(Main.X[i], Main.Y[i] - 30, 8, 8);
        if(send==20)
{
          URL yahoo = new
URL("http://"+Main.ipaddress.getText()+":"+Main.portno.getText()+"/LBS/BU
S2?currLoc="+Main.X[i]+"&speed=1");
          URLConnection yc = yahoo.openConnection();
          BufferedReader in = new BufferedReader(new
InputStreamReader(yc.getInputStream()));
          String inputLine;

          while ((inputLine = in.readLine()) != null)
{
            //System.out.println(inputLine);
          }
          in.close();
        send=0;
        }
      } catch (IOException ex)
{
          Logger.getLogger(NewBus.class.getName()).log(Level.SEVERE, null,
ex);
      }
   }


   private String getBusLocation(int i)
{
```

```
       if(i==52)
{
       loce="stop1";
       }
      if(i==116)
{
       loce="stop2";
       }
       if(i==170)
{
         System.out.println("stop2");
      loce="stop3";
       }
        if(i==230)
{
       loce="stop4";
       }
       if(i==371)
{
      loce="stop5";
       }
    if(i==470)
{
       loce="stop6";
       }
       if(i==503)
{
         System.out.println("stop4");
      loce="stop7";
       }
        if(i==551)
{
       loce="stop 8";
       }
        if(i==666)
{
       loce="stop 9";
       }
       if(i==742)
{
         loce="stop 10";
       }
```

```
      return loce;
   }

   private String getBusStatus(int i)
{
      if(i==81)
{
      stat="near stop 2";
      }
      if(i==143)
{
      stat="near stop 3";
      }
       if(i==196)
{
      stat="near stop 4 ";

      }
      if(i==300)
{
      stat="near stop 5";
      }
      if(i==436)
{
      stat="near stop 6";
      }
      if(i==488)
{
      stat="near stop 7";
      }
      if(i==513)
{
      stat="near stop 8";
      }

      if(i==604)
{
      stat="near stop 9";
      }
       if(i==700)
{
      stat="near stop 10";
      }
```

```
        if(i==724)
{
       stat="Reached stop 10";

    Main.fixbtn.setEnabled(true);

       Main.date=new Date();
       EndT=Main.date;
       System.out.println("Start time"+StartT);
       System.out.println("End time"+EndT);

       }
       return stat;
   }


}
```

NEW BUS 3:

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.URL;
import java.net.URLConnection;
import java.util.Date;

/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.ImageIcon;
import javax.swing.JLabel;
import javax.swing.JPanel;

/**
 *
 * @author Administrator
 */

public class NewBus3 extends Thread
```

```
{
   public static Thread t;
   String stat=null;
   String loce=null;
   String StartTimeB3;
   String RreachedTimeB3;
   int send;
   Date StartT;
   Date EndT;
   public ImageIcon icon1;
   public JLabel iconbus1;
   public JPanel root1;
   public int speed3;
   NewBus3()
{
       Main.date=new Date();
       StartT=Main.date;
       @SuppressWarnings("deprecation")
       int
starttime=(Main.date.getHours()*60*60)+Main.date.getMinutes()*60+Main.dat
e.getSeconds();
       StartTimeB3=starttime+"";
       icon1=new ImageIcon("../EconomicalLSB/src/image/busico.gif");
       iconbus1=new JLabel(icon1);
       root1=new JPanel();
       root1.add(iconbus1);
       root1.setBounds(43, 495, 8, 8);
       t = new Thread(this, "LSB");
       System.out.println("Start Time Bus 3: " + StartTimeB3);
       t.start(); // Start the thread
}
// This is the entry point for the second thread.
   @Override
public void run()
{
try
{
   for(int i=0;i<Main.X.length;i++)
{
     Main.date=new Date();
     setLocation(i);
            @SuppressWarnings("deprecation")
     String hr=""+Main.date.getHours();
```

```
            @SuppressWarnings("deprecation")
      String min=""+ Main.date.getMinutes();
            @SuppressWarnings("deprecation")
      String sec=""+Main.date.getSeconds();

      String loc=  getBusLocation(Main.X[i]);
      String status=getBusStatus(Main.X[i]);
      Main.locate3.setText(loc);
      Main.loccros3.setText(status);
      Thread.sleep(200);
}
}
catch (InterruptedException e)
{
System.out.println("Child interrupted.");
}

}

    void setLocation(int i)
{
        try
{
           send++;
          Main.ob.root3.setBounds(Main.X[i], Main.Y[i] - 30, 8, 8);
        if(send==20)
{
          URL yahoo = new
URL("http://"+Main.ipaddress.getText()+":"+Main.portno.getText()+"/LBS/BU
S3?currLoc="+Main.X[i]+"&speed=1");
          URLConnection yc = yahoo.openConnection();
          BufferedReader in = new BufferedReader(new
InputStreamReader(yc.getInputStream()));
          String inputLine;

          while ((inputLine = in.readLine()) != null)
{
            //System.out.println(inputLine);
          }
          in.close();
        send=0;
        }
      }
```

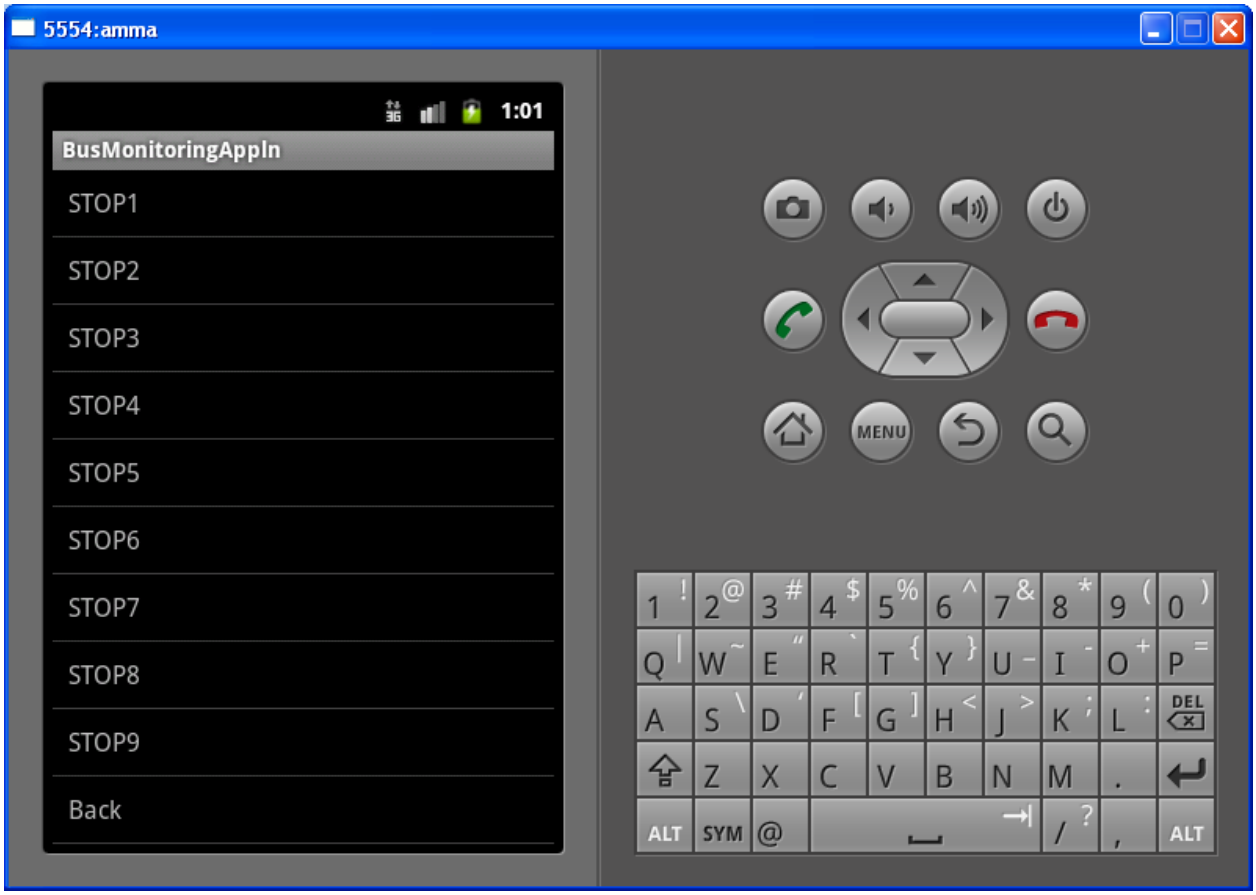
```
catch (IOException ex)
{
          Logger.getLogger(NewBus.class.getName()).log(Level.SEVERE, null,
ex);
      }
   }


   private String getBusLocation(int i)
{

      if(i==52)
{
        loce="stop1";
      }
     if(i==116)
{
         loce="stop2";
      }
       if(i==170)
{
         loce="stop3";
      }
        if(i==230)
{
         loce="stop4";
      }
       if(i==371)
{
        loce="stop5";
      }
    if(i==470)
{
         System.out.println("stop 4 near");
      loce="stop6";
      }
       if(i==503)
{
       loce="stop7";
      }
        if(i==551)
{
       loce="stop 8";
```

```
      }
       if(i==666)
{
        loce="stop 9";
      }
       if(i==742)
{
        loce="stop 10";
      }
      return loce;
   }

   private String getBusStatus(int i)
{
       if(i==81)
{
       stat="near stop 2";
       }
       if(i==143)
{
       stat="near stop 3";
       }
        if(i==196)
{
       stat="near stop 4 ";
          }
       if(i==300)
{
       stat="near stop 5";
       }
       if(i==436)
{
       stat="near stop 6";
       }
       if(i==488)
{
       stat="near stop 7";
       }
       if(i==513)
{
       stat="near stop 8";
       }
```

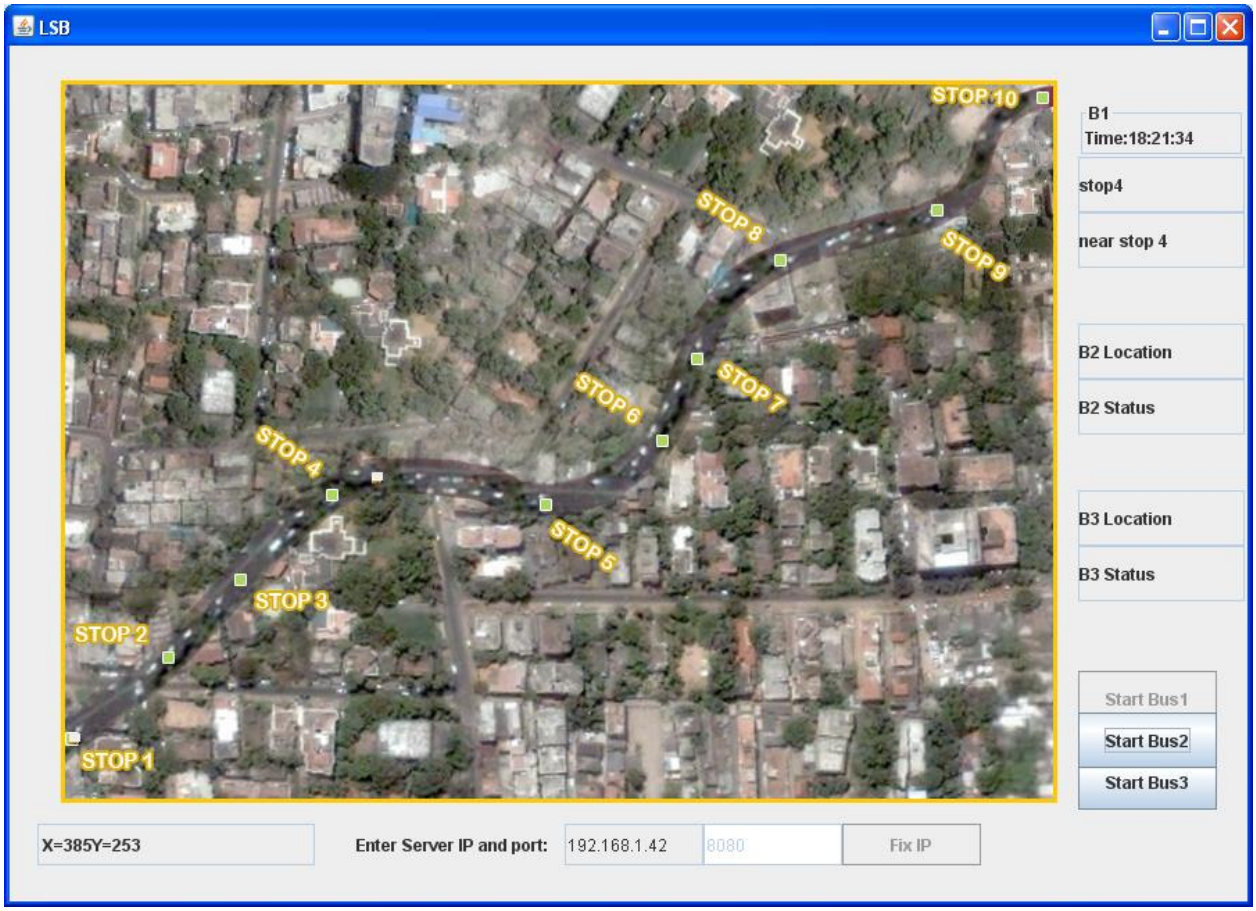
```
        if(i==604)
{
        stat="near stop 9";
        }
         if(i==700)
{
        stat="near stop 10";
        }
         if(i==724)
{
        stat="Reached stop 10";

     Main.fixbtn.setEnabled(true);

        Main.date=new Date();
        EndT=Main.date;
        System.out.println("Start time"+StartT);
        System.out.println("End time"+EndT);

        }
        return stat;
    }


}
```

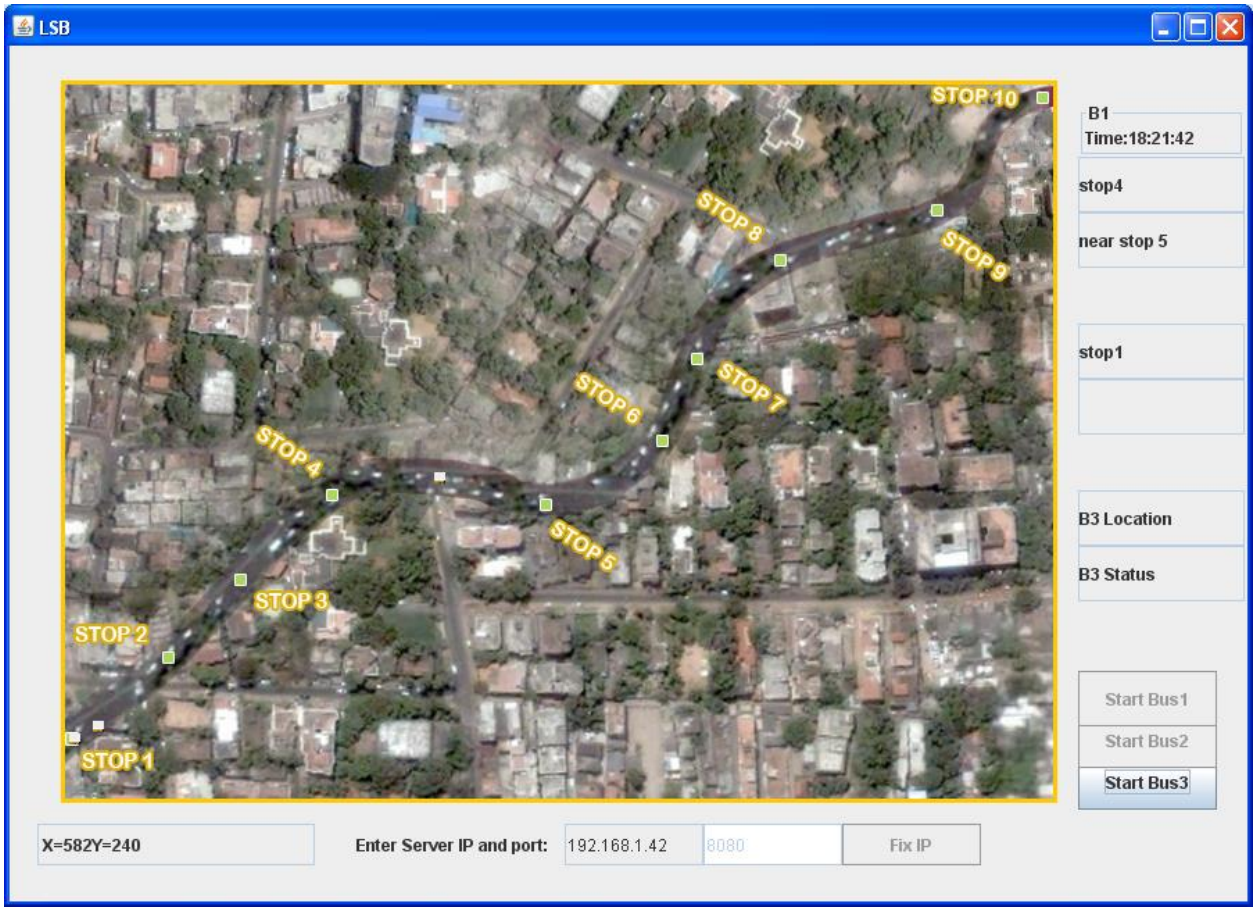
APPENDIX 2 (SNAPSHOTS)

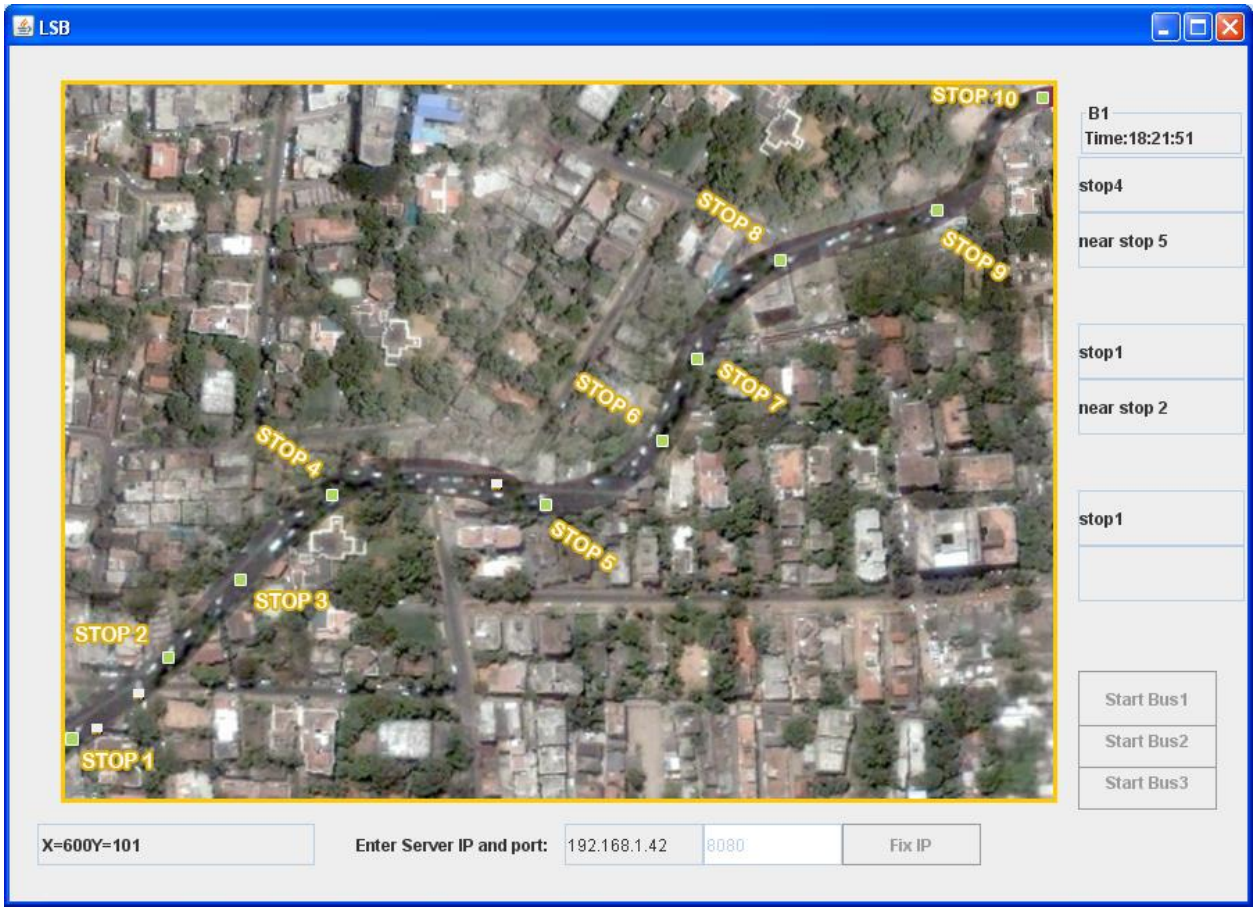
SNAPS SHOTS:


5554:amma
1:01
BusMonitoringAppIn
STOP1
STOP2
STOP3
STOP4
STOP5
STOP6
STOP7
STOP8
STOP9
Back
MENU
ALT
SYM
ALT


5554:amma
1:01
BusMonitoringAppIn
STOP2
STOP3
STOP4
STOP5
STOP6
STOP7
STOP8
Destination selected....STOP5
Submit
MENU


LSB
STOP 10
STOP 8
STOP 9
STOP 6
STOP 7
STOP 4
STOP 5
STOP 3
STOP 2
STOP 1
B1
Time:18:21:34
stop4
near stop 4
B2 Location
B2 Status
B3 Location
B3 Status
Start Bus1
Start Bus2
Start Bus3
X=385Y=253
Enter Server IP and port:
192.168.1.42
8080
Fix IP


LSB
STOP 10
STOP 8
STOP 9
STOP 6
STOP 7
STOP 4
STOP 5
STOP 3
STOP 2
STOP 1
B1
Time:18:21:42
stop4
near stop 5
stop1
B3 Location
B3 Status
Start Bus1
Start Bus2
Start Bus3
X=582Y=240
Enter Server IP and port:
192.168.1.42
8080
Fix IP


LSB
STOP 10
STOP 8
STOP 9
STOP 6
STOP 7
STOP 4
STOP 5
STOP 3
STOP 2
STOP 1
B1
Time:18:21:51
stop4
near stop 5
stop1
near stop 2
stop1
Start Bus1
Start Bus2
Start Bus3
X=600Y=101
Enter Server IP and port:
192.168.1.42
8080
Fix IP

REFERENCES

.

[1] Hong, J.; Zhu, Q.; Xiao, J. Design and Realization of Wireless Sensor Network Gateway Based on ZigBee and GPRS. 2009 2nd International Conference on Information and Computing Science, Manchester, UK, 2009; pp. 196–199

[2] ZigBee Alliance. ZigBee specification [EB/OL],2008. http://www.ZigBee.org

[3] MC13192 and MC9S08GB60 Reference Manual. http://www.freescale.com, 2007.

[4] Specifications for Low-Rate Wireless Personal Area Networks(LRWPANs). 2003.

[5] IEEE 802.15.4 2003. Wireless Medium Access Control(MAC) and Physical Layer(PHY)

[6] http://www.ieee.org,.

[7] Deng Z.C □ GPRS-technology and market[J] □ Communication World□2000□(9)□27-31.