Abstract

GPS enabled wireless devices transmit locational details like latitude, longitude, altitude, etc to the base station. In the base station (out post) the receiver will receive this information in the wireless frequency and will further send the information (device id, longitude, latitude, altitude and velocity) in string format to a serial port.

From the serial port of the receiver, information is transferred to a computer in outpost through the socket via socket output stream. Further from the outpost, data string is passed on to police station where these strings are then tokenized to appropriate information and stored into the database with particular outpost id. Each police station is a superset of various outposts in its limit. The police station will add its id to the string and forward it to the repeater and central control room. Simultaneously it will also generate a KML file for each device and user can open the KML on Google Earth™ to see the current location of GPS enabled wireless device.

Repeater Stations will have number of police stations under its circle. The information in string format is received and stored into the database with particular police station id in the repeater station and simultaneously a KML file is generated at the repeater station also. User can locate the device in real time position and current path can be seen on Google Earth™.

In the central control room, strings are received from different repeaters with their unique identity number and these received data is stored in the database. Here also user can view the starting point, current location and trace path of the device.

The main aims of this project are:

1. To read the string from receiver via serial communication
2. To decode the string to valuable information like latitude, longitude, altitude, device id and velocity and save it in a database for future use
3. To transmit information strings from outpost to police station, police station to repeater and finally from repeater to central control room
4. To show the current location of the device and trace the path over Google Earth™, at each level, i.e., Outpost, Police Station, Repeater and Central Control Room levels

The advantage of this system will be:

1. The history of the device location is saved and can be viewed at a later stage which will be helpful in investigations
2. Locating any missing person, or if any person lost the path then they can be guided to a nearby safe location

CHAPTER 1

INTRODUCTION

Purpose

The aim of this document is to trace the current location of GPS enabled Wireless Device so that if a person is missing we are able to find it. The information about the location of device such that longitude, latitude, altitude and velocity pass through various checkpoints like police station and central zone offices and they are able to get the location of device on Google Earth.

There are immediate needs for automated surveillance systems in commercial, law enforcement and military applications.

What is needed is continuous monitoring of wireless device to alert security officers to find the current location and track it.

It is the outcome of rigorous consideration of the requirements of the customer, by the various groups.

This document will provide a baseline for design of the database, user interfaces, coding & evaluation of test plans. It will be used as a solid foundation for continued product evaluation.

This describes the requirements and specifications of Simulation of data via serial communication. It explains the functional features of the system, along with interface details, design constraints and related considerations such as performance characteristics. The document is intended for users and owners of GPS enabled Radio Devices.

The simulation of data via serial communication has the following Benefits.

- To get the current location of GPS enabled Wireless Radio devices.
- To trace the path of the GPS enabled Wireless Radio devices.
- To inform station if the person is missing.
- To save the history for future use.

Scope:

This document illustrates the details of Simulation of data via serial communication which is a JAVA based design .This system uses a GPS enabled Radio Devices to get the geographic location. If the person is missing or a device gets damaged or device is showing the path other than its fixed path, the person who is on the other end receives the information about real time location. This is a fully automatic system with real-time monitoring mechanism.

This project uses Serial Communication through RXTX as a basic functionality and is a desktop based application. RXTX is a native lib providing serial and parallel communication for the Java Development Toolkit (JDK). RXTX is licensed under the GNU LGPL license.

Definitions and Assumptions:

A) Assumptions:

1) The users of the application are connected to each other through LAN.
2) Receiver must be connected with the system through serial port.
3) Google Earth must be installed on the system.

Abbreviations:

- **IP Address-** Internet Protocol Address.
- **LAN -** Local Area Network.
- **JVM -** Java Virtual Machine.
- **MB -** Mega Bytes.
- **JDK -** Java Development Kit.
- **GPS-** Global Positioning System
- **RXTX-** Receiver Transmitter

CHAPTER 2

ANALYSIS

Proposed System:

Constraints are boundary conditions on how the system must be designed and constructed. Examples include: legal requirements, technical standards, strategic decisions.

- Constraints exist because of real business conditions. For example, a delivery date is a constraint only if there are real business consequences that will happen as a result of not meeting the date. If failing to have the subject application operational by the specified date places the organization in legal default, the date is a constraint.
- Preferences are arbitrary. For example, a date chosen arbitrarily is a preference.
- Preferences, if included in the FRD, should be noted as such.

Problem Statement:

The Problem of	Designing a GPS Tracking System application. The fundamental objective of GPS Tracking application is to provide security and providing the user, with the facility to monitor the location of device and take an action when any sort of abnormal activity is detected.
Affects	The application designed to provide security and sends position of the remotely connected device.
The impact of which is	Continuously monitoring of device.
A successful solution would be some features such as	1) Working at network layer. 2) Establishing remote connections. 3) Checking the security at your premises.

Product Position Statement:

For	Forest Department, Military, defense department etc.
Who	Needs to get the global position of device.
That	• Easily locate the person and if the person is lost then one can send the rescue team immediately • Saves the time. • Provides security
Our Product	•It operates 24/7. •It can inform the user if the person is lost •It reduces the bandwidth and archiving space needed by transmitting or recording only data on relevant events. •It relieves security personnel from continuous tracking. •It enables a quick search of the person.

Functional Requirements:

Major requirements in the project are:

- SYSTEM: Needs a system which will receive the information from device and show position on Google Earth.
- GPS ENABLED RADIO DEVICES: A device which send information to the system.
- POSTGRESQL 9.0.8: Database to store location details for future comparison.
- LANGUAGE: This project uses JAVA as a basic code design language and implements graphical user interface.

Non-Functional Requirements:

One of the most important non-functional requirements is security. Security requirements can come in many different forms:

Availability- Availability requirements define that the GPS enabled wireless radio device should be available.

Response Time – These requirements define that device should send the information continuously after specific time period.

Documentation:

Documentation describes how technology, information & people will be molded into an operating information system. The challenge is to get all three of these factors to work together. For the implementation plan to be successful the various development activities must be appropriately sequenced & the necessary financial & political support must be maintained.

Hardware and Software Considerations:

System Requirements:

Minimum Requirements:

- ✓ Pentium IV
- ✓ 128 MB RAM or more
- ✓ I/O ports to handle the connectivity among computers.
- ✓ LAN card.
- ✓ Color monitor.
- ✓ Standard keyboard
- ✓ Standard mouse.

System Requirements:

Minimum Requirements:

- ✓ JVM
- ✓ PostgreSQL
- ✓ Google Earth

Error Handling & System modifications:

Error handling mechanism is provided by displaying appropriate messages for the corresponding errors, which occur during the working of the system.

On completion of the software development, testing activities take place to demonstrate that the application functions according to its agreed specification.

Software testing is carried out in two parts as FAT (Factory Acceptance Tests) which are performed at the developer's site.

All the tests are carried out according to a 'Test Specification', which is a document prepared in relation to the Functional specification. The test specification should clearly state the test strategy, test environment, success criteria, user representation in the tests, test observation reporting & acceptance procedure. During the software testing there may be number of failures or some symptoms. In this case the test team should reach an agreement on the action to be taken:

1. The symptoms do not significantly affect the main functionality.
2. A remedial solution is outlined.
3. Procedure for change verification is agreed.

State required reliability:

1. Mean-Time-Between-Failure is the number of time units the system is operable before the first failure occurs.
2. Mean-Time-To-Failure is the number of time units before the system is operable divided by the number of failures during the time period.
3. Mean-Time-To-Repair is the number of time units required to perform system repair divided by the number of repairs during the time period.

CHAPTER 3

DESIGNING

Design Details:

Design is a meaningful engineering representation of something that has to be built. Software design sits at technical kernel of software engineering. It is the iterative process through which the requirements are translated into a ‘blueprint’ for constructing software.

A Short Program Description:

WalkieTalkie class is used to read the data from files and send to the another class called **DeviceReader. DeviceReader** will read the data, received through serial port and pass to the **OutPost** class. **OutPost** class is connected to another class **PoliceStation. OutPost** class send data to this class via socket. **PoliceStation** class receive the data and tokenize it and store it in the database called PS_Database and also send the data to the class called **Repeater. Repeater** class receive the data and tokenize it and store it in the database called RPTR_Database and also send the data to the class called **CentralRepeater. CentralRepeater** class receive the data and tokenize it and store it in the database called Central_RPTRDatabase. **KMLCreator** class is attached to the **PoliceStation, Repeater and CentralRepeater** class. **KMLCreator** class is used to generate a kml file for each device which store the position information like longitude, latitude, altitude and velocity. Another class **Locator** is used for read the kml files and map the position on the Google Earth.

PROJECT OUTPUT:

WALKIE TALKIE DATA SENDER

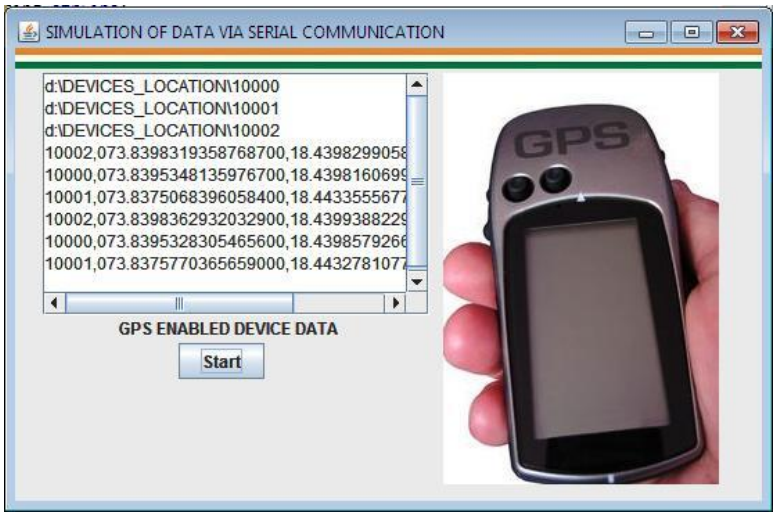


Fig 3.1. Walkie-talkie

Police Station

Fig 3.2. Police Station

Repeater

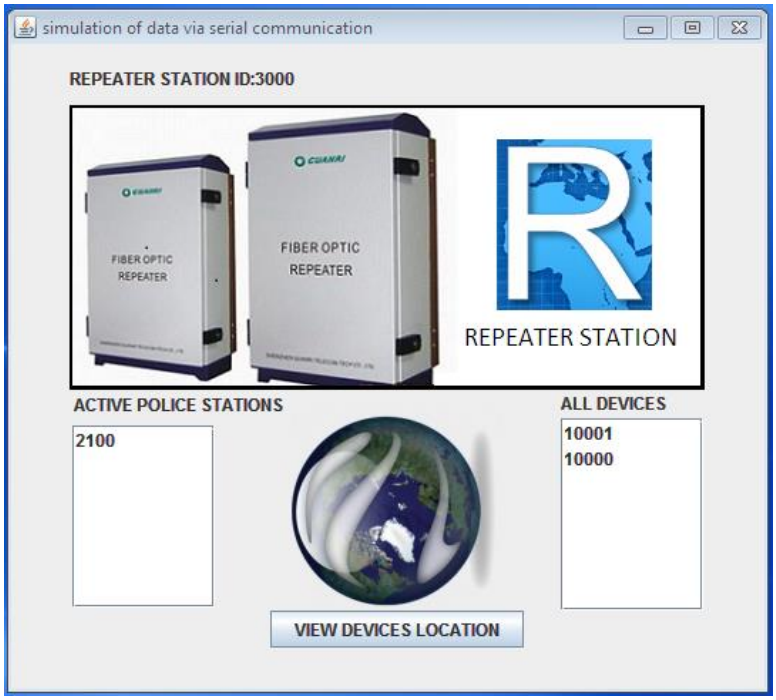


Fig 3.3. Repeater

Central Repeater

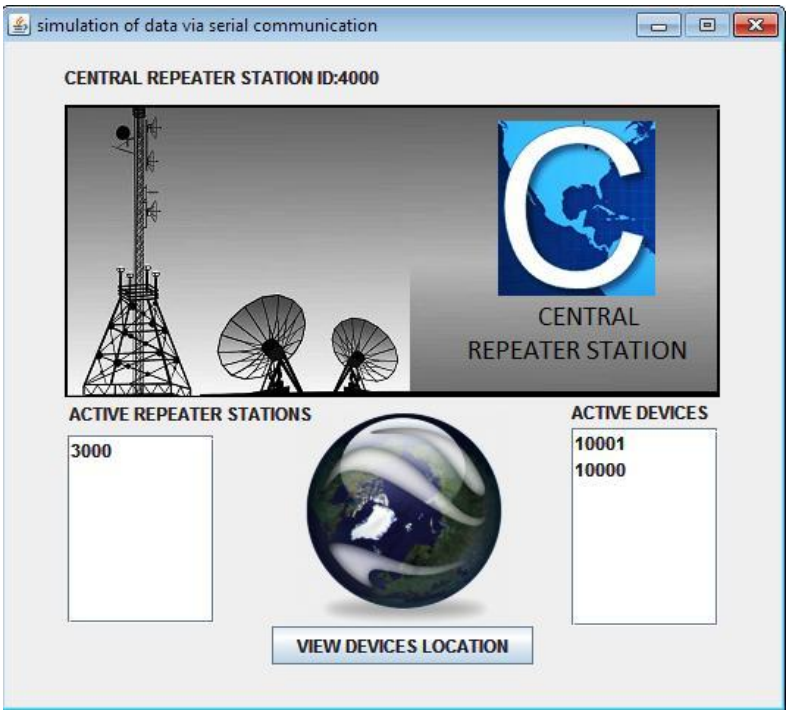


Fig 3.4. Central Repeater

Device Location

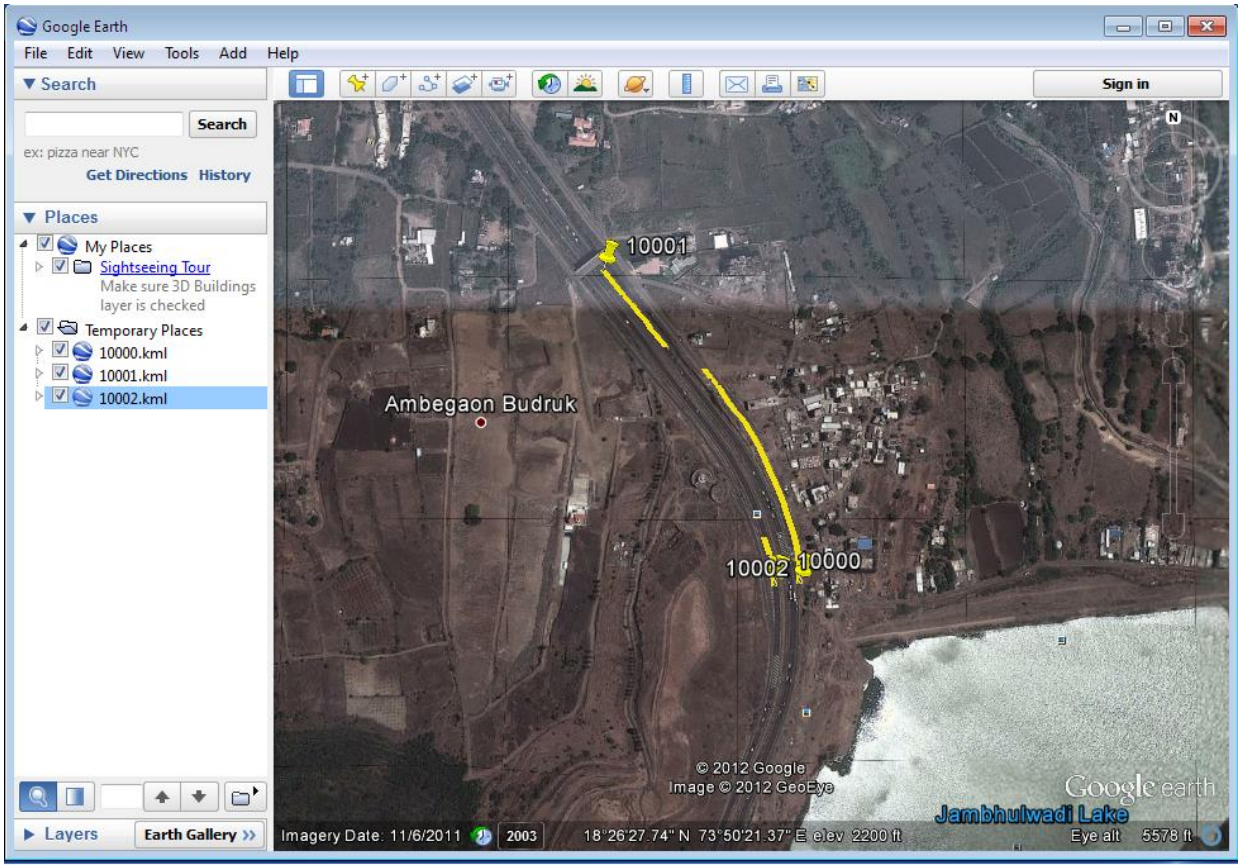


Fig 3.5. Device position on Google Earth

CHAPTER 4

MODELING

Use case Diagram 1

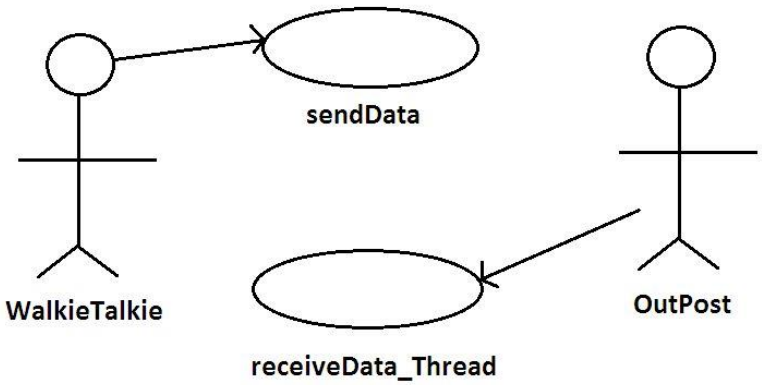


Fig 4.1

Use case Diagram 2

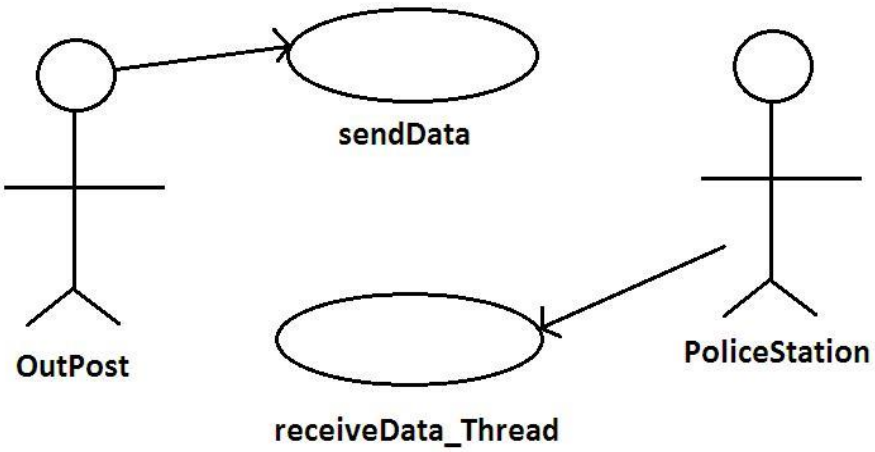


Fig 4.2

Use case Diagram 3

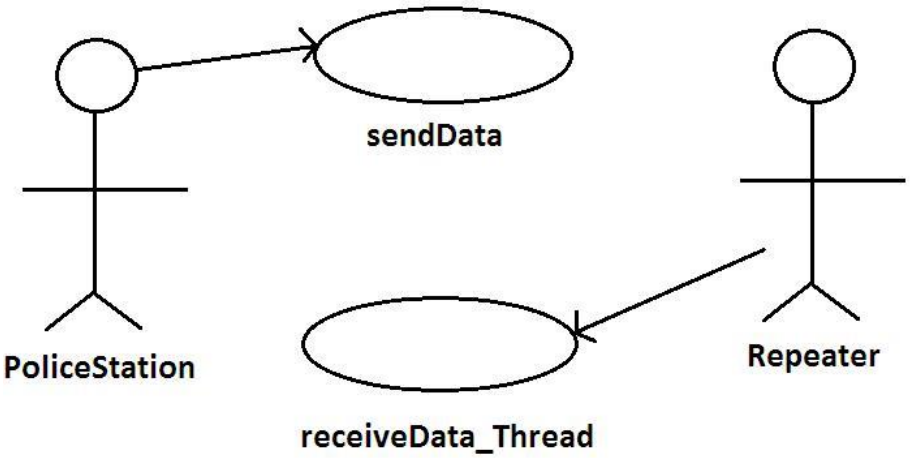


Fig 4.3

Use case Diagram 4

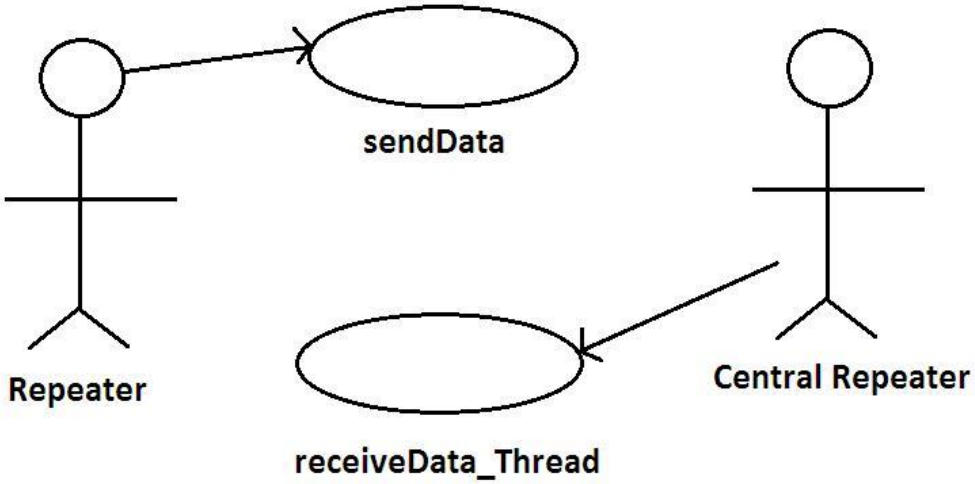


Fig 4.4

Class Diagram - I

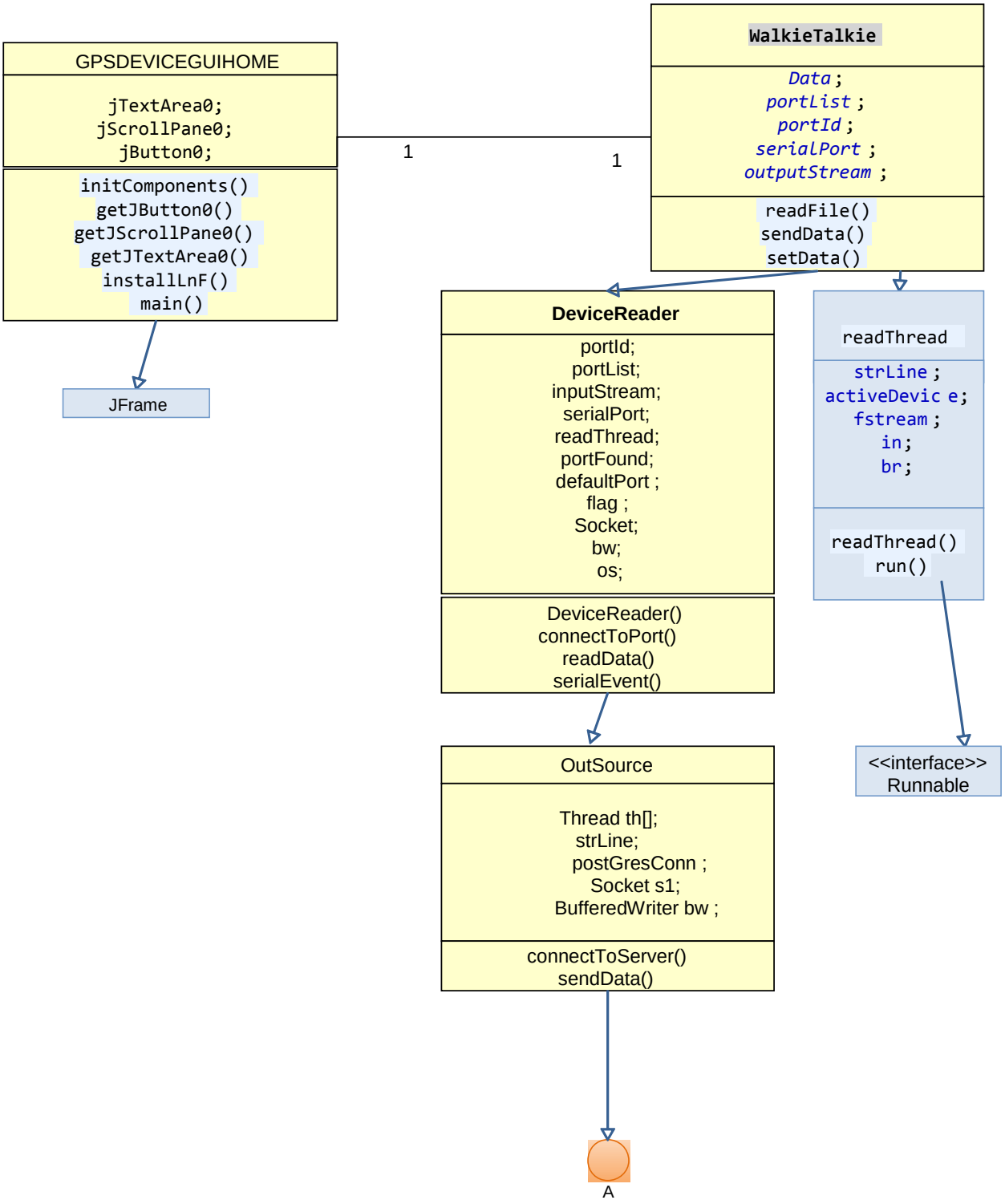


Fig 4.5

Class Diagram - II

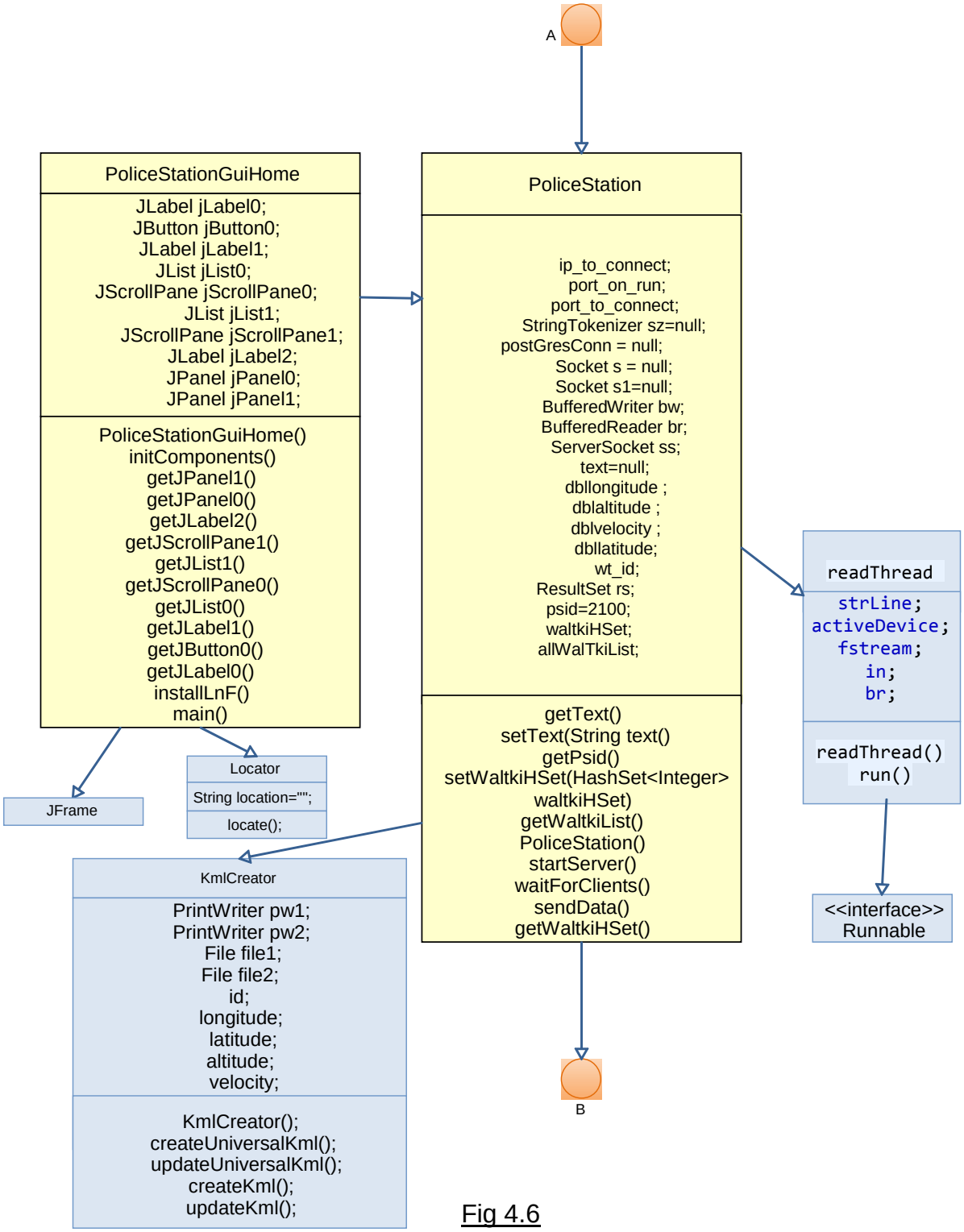


Fig 4.6

Class Diagram - III

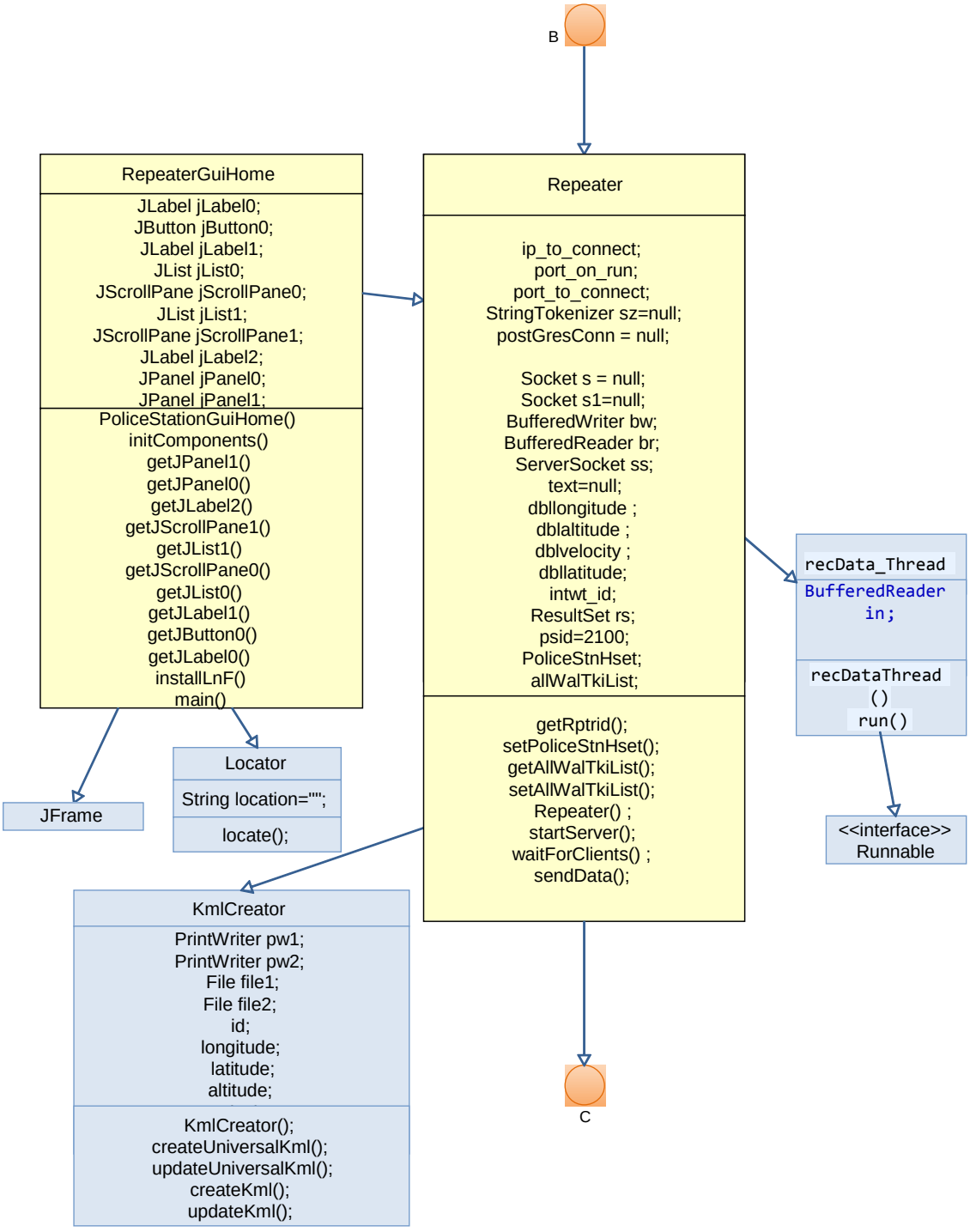


Fig 4.7

Class Diagram - IV

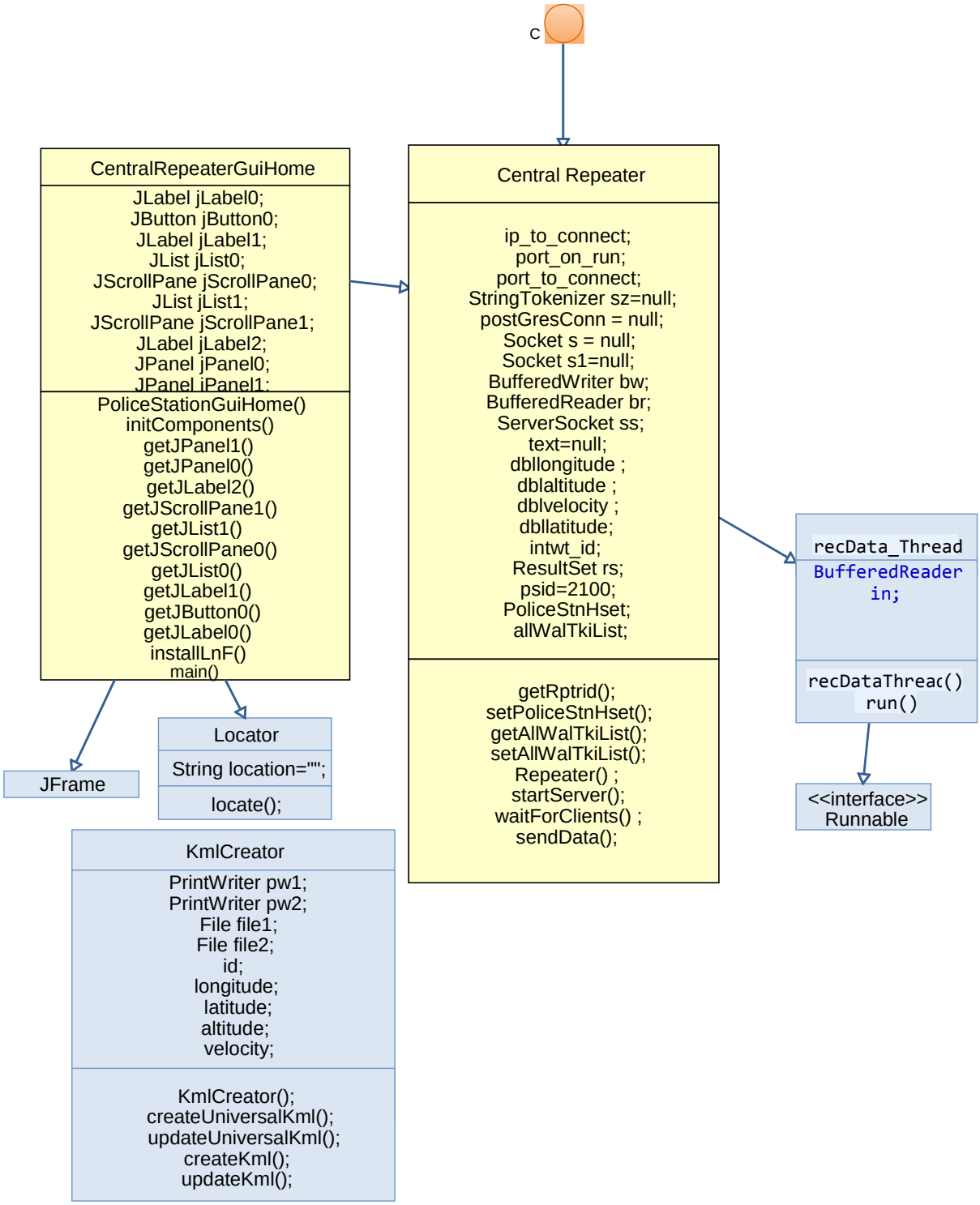


Fig 4.8

ER Diagram

Central_RPTRDatabase

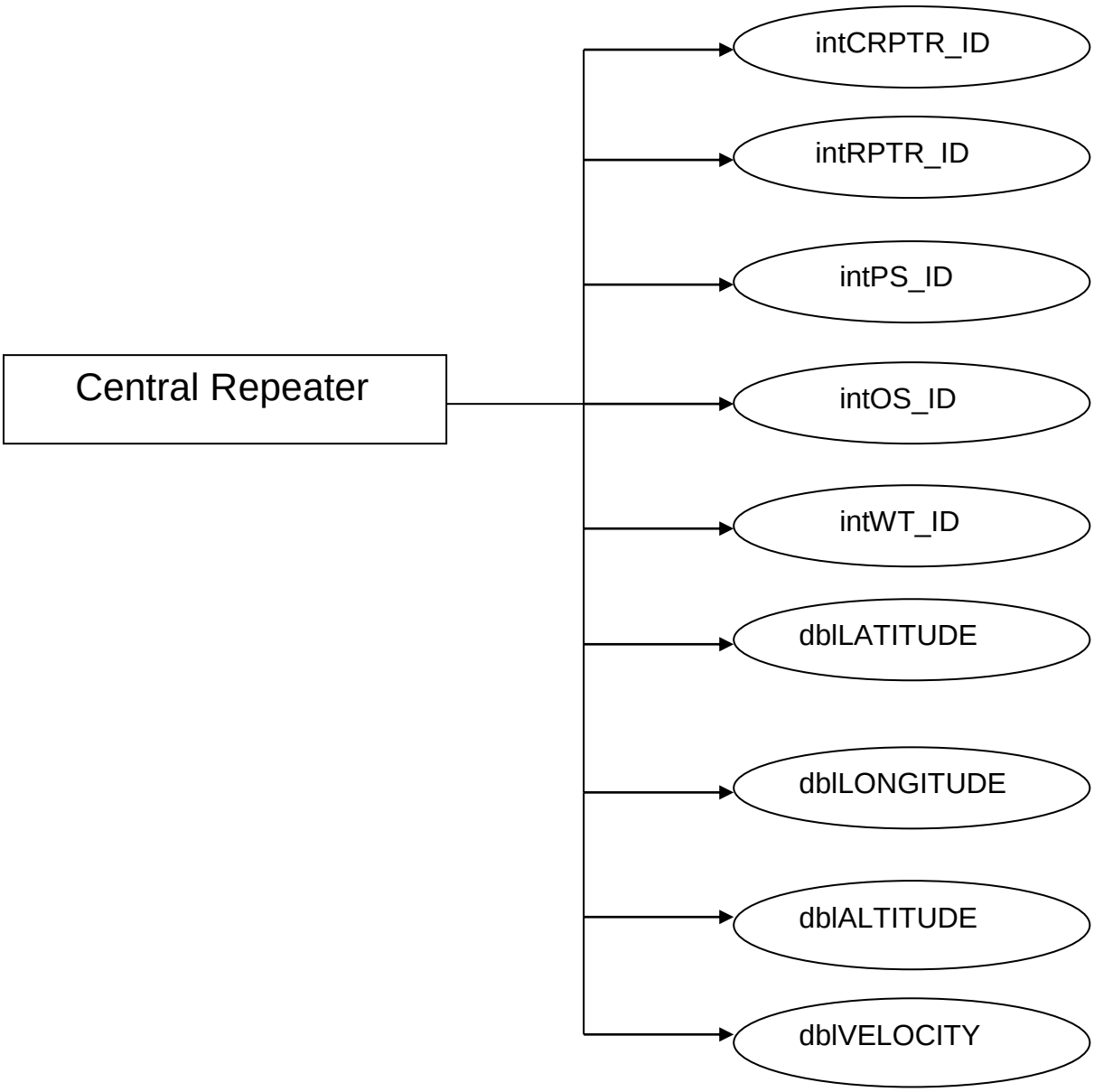


Fig 4.9

RPTRDatabase

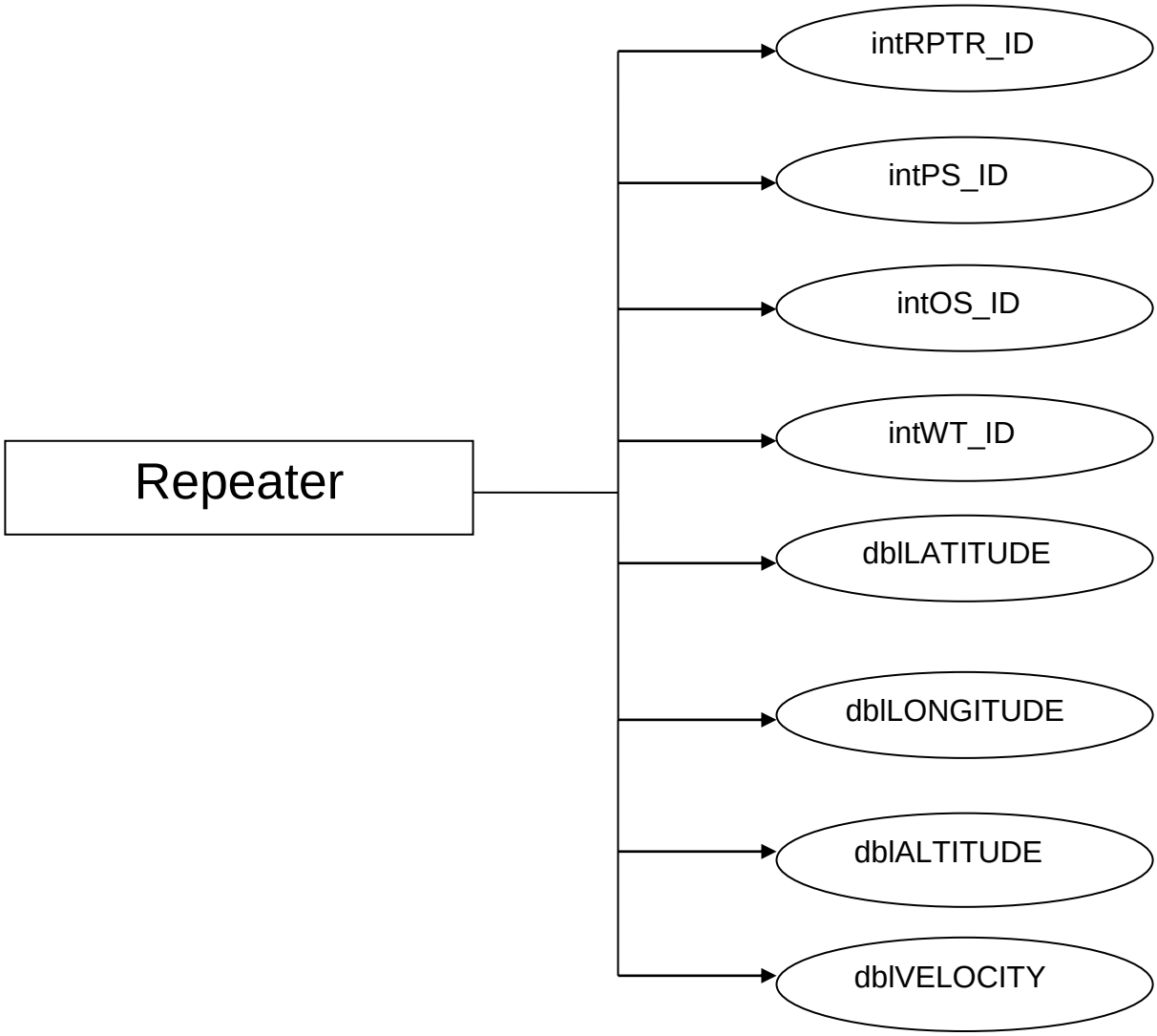


Fig 4.10

PS_Database

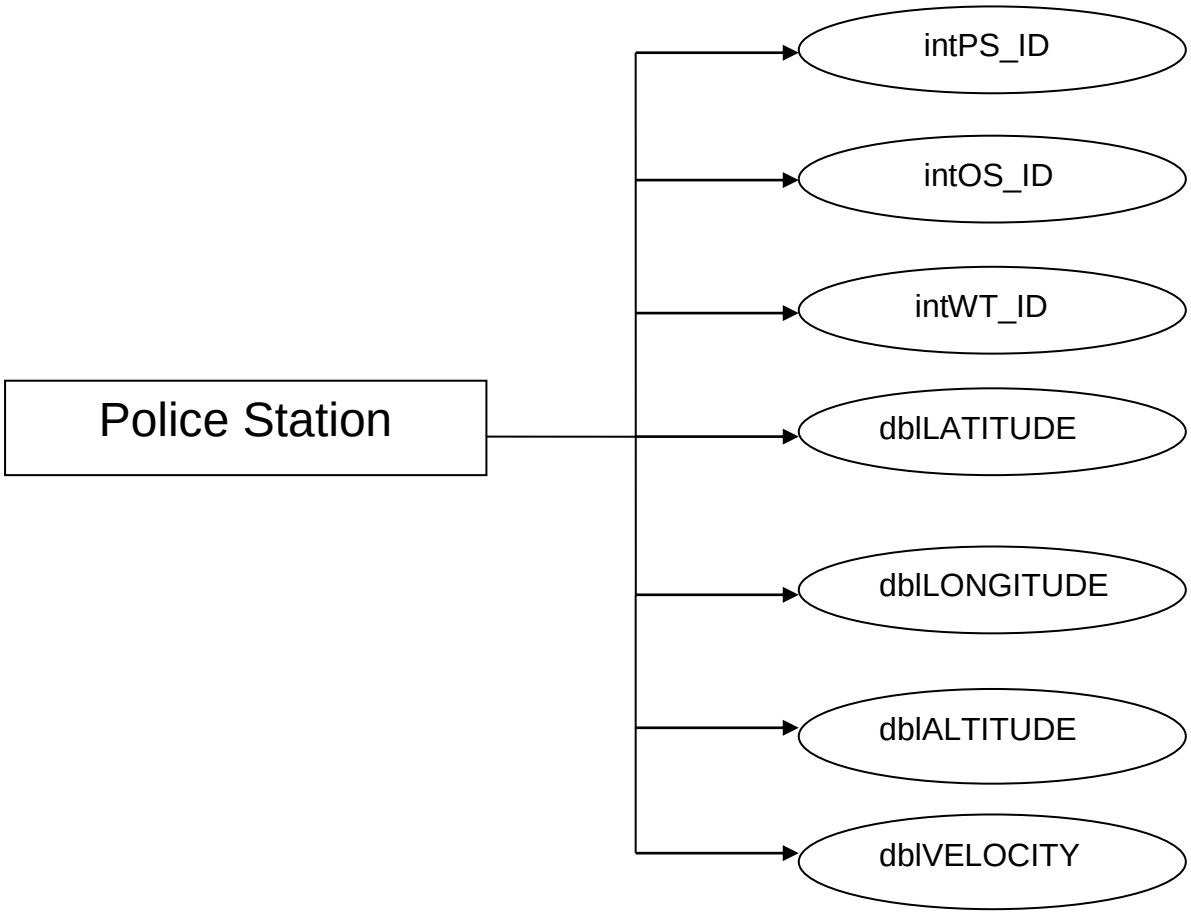


Fig 4.11

SEQUENCE DIAGRAM

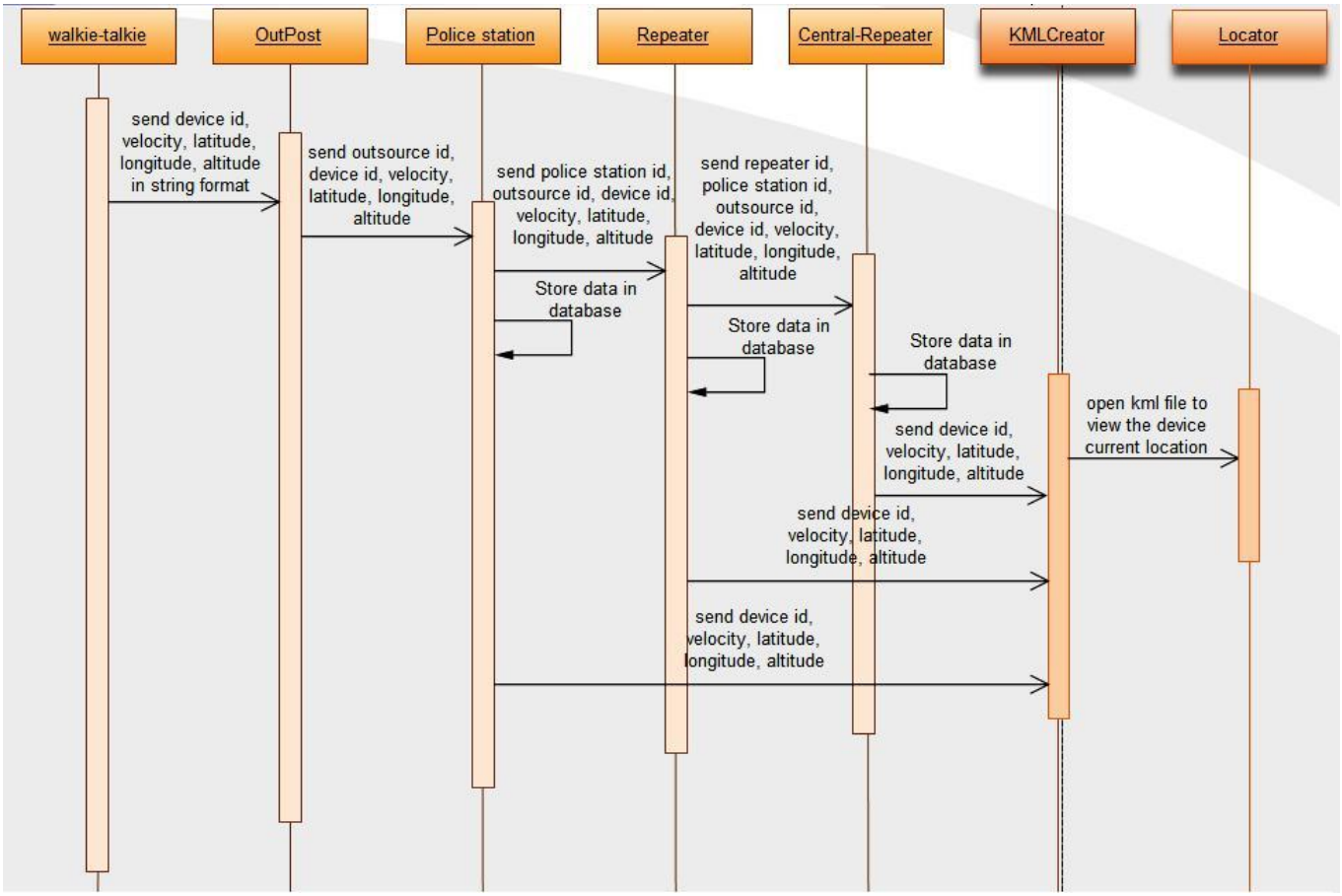
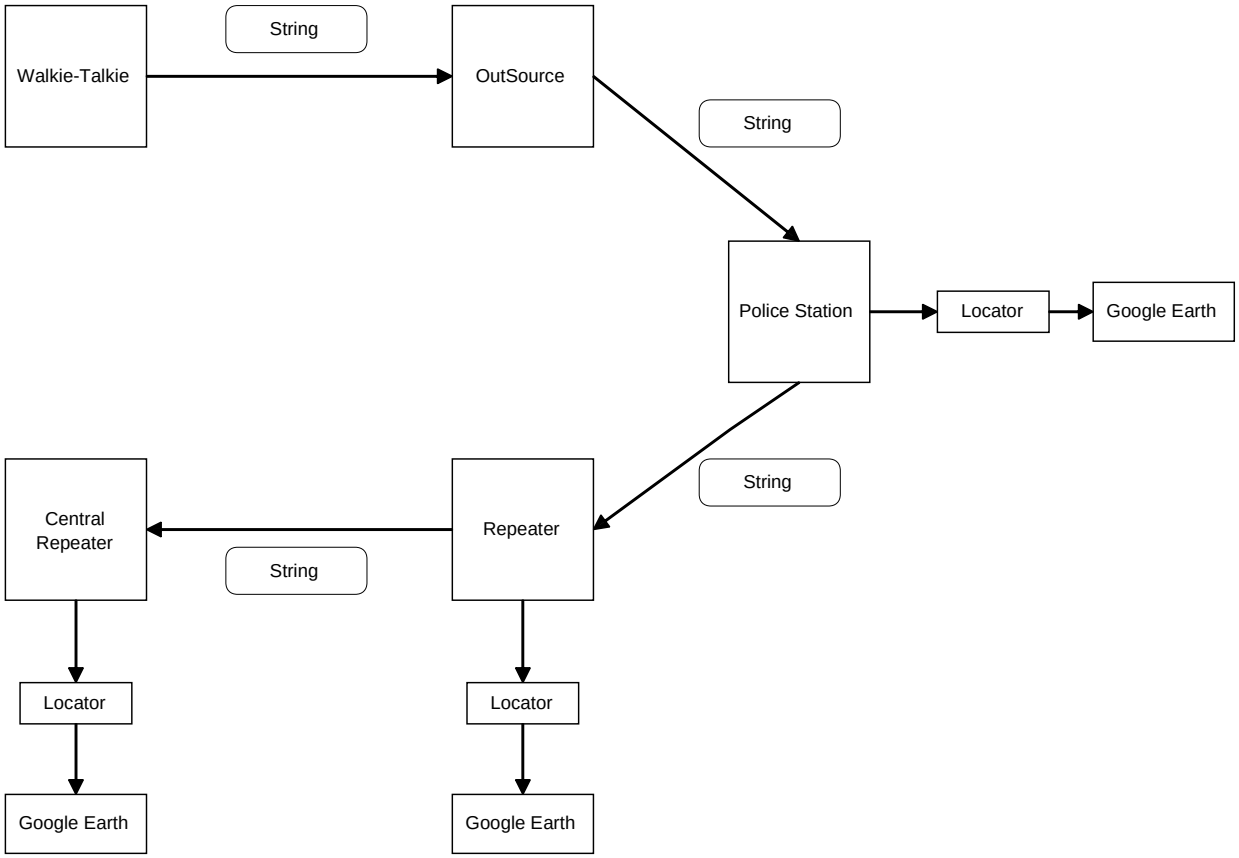


Fig 4.12

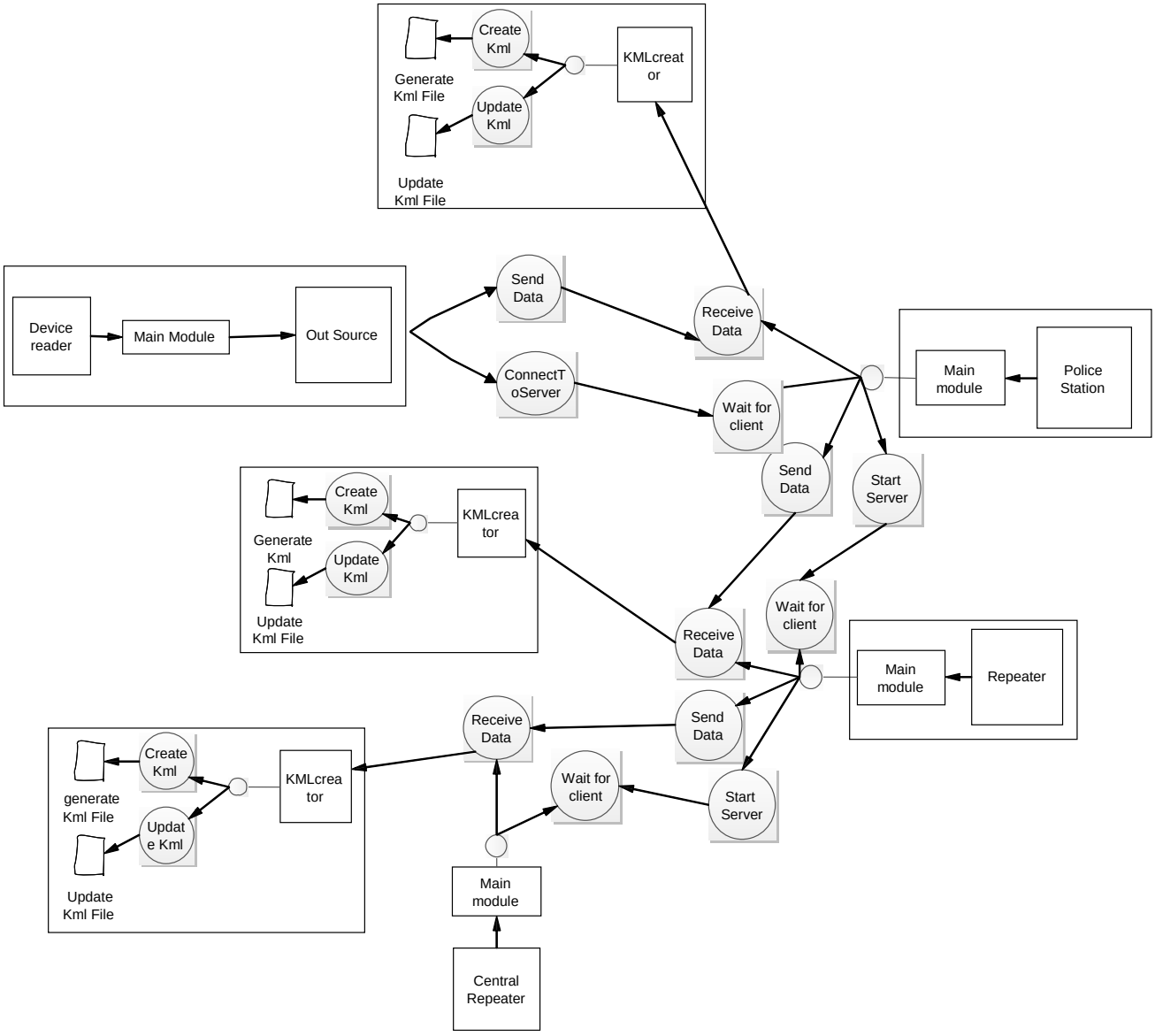
Data Flow Diagrams

Decompose the context level diagrams to determine the functional requirements. Data flow diagrams should be decomposed down to the functional primitive level. These diagrams are further decomposed during design.

DFD Level – 0



DFD Level - 1


Create Kml
KMLcreat or
Generate Kml File
Update Kml
Update Kml File
Device reader
Main Module
Out Source
Send Data
ConnectT oServer
Receive Data
Wait for client
Main module
Police Station
Send Data
Start Server
Create Kml
KMLcrea tor
Generate Kml
Update Kml
Update Kml File
Receive Data
Wait for client
Main module
Repeater
Receive Data
Send Data
Create Kml
KMLcrea tor
generate Kml File
Updat e Kml
Update Kml File
Wait for client
Start Server
Main module
Central Repeater

CHAPTER 5

CODING

Programming Language:

Java is a new and powerful language that provides many useful features to the software developer.

Why Move to Java?

When deciding whether to move existing applications to Java, a tradeoff between the advantages and disadvantages of such a move must be considered. This section identifies many of the advantages of Java programs over C and C++ based applications. The following section considers some disadvantages of using Java and roadblocks to any software transition effort

Platform Independence:

One of the most compelling reasons to move to Java is its platform independence. Java runs on most major hardware and software platforms, including Windows 95 and NT, the Macintosh, and several varieties of UNIX. Java applets are supported by all Java-compatible browsers. By moving existing software to Java, you are able to make it instantly compatible with these software platforms. Your programs become more portable. Any hardware and operating system dependencies are removed. Although C and C++ are supported on all platforms that support Java, these languages are not supported in a platform-independent manner. C and C++ applications that are implemented on one operating system platform are usually severely intertwined with the native windowing system and OS-specific networking capabilities. Moving between OS platforms requires recompilation, as a minimum and significant redesign, in most cases.

Object Orientation:

Java is a true object-oriented language. It does not merely provide the capability to implement object-oriented principles-it enforces these principles. You can develop object-oriented programs in C++, but you are not required to do so-you can use C++ to write C programs as well. Java does not allow you to slip outside the object-oriented framework. You either adhere to Java's object-oriented development approach or you do not program in Java.

Security:

Java is one of the first programming languages to consider security as part of its design. The Java language, compiler, interpreter, and runtime environment was each developed with security in mind. The compiler, interpreter, and Java-compatible browsers all contain several levels of security measures that are designed to reduce the risk of security compromise, loss of data and program integrity, and damage to system users. Considering the enormous security problems associated with executing potentially untrusted code in a secure manner and across multiple execution environments, Java's security measures are far ahead of even those developed to secure military systems. C and C++ do not have any intrinsic security capabilities. Can you download an arbitrary entrusted C or C++ program and execute it in a secure manner?

Reliability:

Security and reliability go hand in hand. Security measures cannot be implemented with any degree of assurance without a reliable framework for Program execution. Java provides multiple levels of reliability measures, beginning with the Java language itself. Many of the features of C and C++ that are detrimental to program reliability, such as pointers and automatic type conversion, are avoided in Java. The Java compiler provides several levels of additional checks to identify type mismatches and other inconsistencies. The Java runtime system duplicates many of the checks

performed by the compiler and performs additional checks to verify that the executable byte codes form a valid Java program.

Simplicity:

The Java language was designed to be a simple language to learn, building on the syntax and many of the features of C++. However, in order to promote security, reliability, and simplicity, Java has left out those elements of C and C++ that contribute to errors and program complexity. In addition, Java provides automated garbage collection, freeing you from having to manage memory de-allocation in your programs. The end result of Java's focus on simplicity is that it is easy to get up to speed writing Java programs for those who have programmed in C or C++. Java programs are also less complex than C and C++ programs due to the fact that many of the language elements that lead to program complexity have been removed.

Language Features:

The Java language provides many language features that make it preferable to C or C++ for modern software development. On the top of this list is Java's intrinsic support for multi-threading, which is lacking in both C and C++. Other features are its exception-handling capabilities, which were recently introduced into C++; its strict adherence to class and object-oriented software development; and its automated garbage-collection support. In addition to these features, Java enforces a common programming style by removing the capability to slip outside the class and object-oriented programming paradigm to develop C-style function-oriented programs.

Standardization:

Although C and C++ have been standardized by the American National Standards Institute (ANSI), many C and C++ compilers provide custom enhancements to the language, usually through additional preprocessor directives.

Because these enhancements usually make their way into source code programs, a general lack of standardization results. Java does not yet suffer from any standardization problems because its syntax and semantics are controlled by a single organization.

The Java API:

The predefined classes of the Java API provide a comprehensive platform-independent foundation for program development. These classes provide the capability to develop Windows and network programs that execute on a wide range of hosts. The Java I/O stream classes also provide a very useful set of filters for I/O processing. Whereas C and C++ may provide more extensive software libraries, none of these libraries provides as much platform-independent power as Java's API.

Transition to Distribute Computing:

Sun has taken important steps to support fully distributed computing with its support of Remote Objects for Java. This product provides the capability to develop remote interfaces between Java objects and objects developed in other languages. The Java Interface Definition Language (IDL) can be used to support Common Object Request Broker Architecture (CORBA) integration.

Rapid Code Generation:

Because Java is an interpreted language, it can be used to rapidly prototype applications that would require considerably more base software support in languages such as C or C++. The Java API also contributes to the capability to support rapid code generation. The classes of the Java API provide an integrated, easy-to-use repository for the development of application-specific software. Because the Java API provides high-level windows and networking support, custom application prototypes can be constructed more quickly using these classes as a foundation.

Ease of Documentation and Maintenance:

Java software is essentially self-documenting when doc comments and the java doc tool are used to generate software documentation. The excellent Java API documentation is an example of the superior documentation capabilities provided by Java. Because Java software is inherently better-structured and documented than C or C++ software, it is generally easier to maintain. In addition, the package orientation of

Java software affords considerable modularity in software design, development, documentation, and maintenance.

Disadvantages of Java:

Java provides many benefits that make it an attractive language to use to develop new applications. The previous section discussed some of the advantages of porting existing code to Java. This section identifies some of the disadvantages.

Performance:

Java is interpreted, and although its execution is efficient, it might not meet the performance demands of those applications in which execution speed is of paramount importance. Examples of these types of applications include numerical "number crunching" programs, real-time control processes, language compilers, and modeling and simulation software. Just because your application fits into one of these categories does not necessarily rule out Java, however. For example, the Java compiler is written in Java and performs admirably for small programs. However, its performance is greatly enhanced when it is compiled into native machine code instructions. Java-to-C translators allow programs to be developed in Java and translated into C for native machine code compilation. The translation process generally improves the performance of Java programs.

Retraining:

Although Java is simple, easy to learn, and based on C++, some training may be required to get programmers up and running writing Java code. This is especially true if the programmers have been using C++ in a nonstructural, non-object-oriented fashion. After you made the mental transition to the Java object-oriented programming model, you became much more comfortable and efficient in writing Java programs.

The Java Compiler:

javac, the Java compiler is supplied as part of Sun's JDK. The javac compiler is written entirely in Java, so it's available for any platform that supports the Java run-time

system. The ability to support its own development environments is an important stage in a language's development. Java makes this bootstrapping automatic by supplying a ready-to-run compiler at the same cost as porting the interpreter.javac turns Java source code into a compiled class that contains Java virtual machine byte-code. By convention, source files are named with a .java extension; the resulting class files have a .class extension. javac considers a file to be a single compilation unit. Classes in a given compilation unit share certain features like package and import statements. javac allows you one public class per file and insists the file have the same name as the class. If the filename and class name don't match, javac issues a compilation error. A single file can contain multiple classes, as long as only one of the classes is public. You should avoid packing lots of classes into a single file. The compiler lets you include extra non-public classes in a .java file, so that you can implement a class that is tightly coupled to another class without a lot of hassle. But you should have more than one class in a file if the public class in the file is the only one that ever uses additional classes.

Now for an example, The source code for the following class should be placed in a file called BigBird.java:

```
package animals.birds
public class BigBird extends Bird {
 ...
}
```

We can then compile it with:

```
% javac BigBird.java
```

Unlike the Java interpreter, which takes a class name as its argument, javac requires an actual filename to process. The above command produces the class file BigBird.class and stores it in the same directory as the source file. While it's useful to have the class file in the same directory as the source when you are working on a simple example, for most real applications you'll need to store the class file in an appropriate place in the class path.

The Java compiler is more intelligent than your average compiler and replaces some of the functionality of a make utility. For example, javac compares the modification times of the source and class files for all referenced classes and recompiles them as necessary. A compiled Java class remembers the source file from which it was compiled, so as long as the source file is in the same directory as the class file, javac can recompile the source if necessary. If, in the above example, class BigBird references another class, animals.furry.Grover, javac looks for the source Grover.java in an animals.furry package and recompiles it if necessary to bring the Grover.class class file up to date.

It's important to note that javac can do its job even if only the compiled versions of referenced classes are available. Java class files contain all the data type and method signature information source files do, so compiling against binary class files is as type safe (and exception safe) as compiling with Java source code.

Program Structure:

The main () Method

Every Java program contains a public class with a main() method that serves as the entry point for that program. A typical Java program often references other Java classes that are included within the same package as the program's class or in other compiled packages.

The Java main () method takes a single args[] parameter of the String class. The number of arguments passed via the main () method invocation is determined by args.length.

Java programs do not pass the program name as the first argument to the program

The Java main() method has a void return type. The exit() method of the java.lang.System class can be used to return an exit code.

Packages

All Java classes are defined relative to a package, even if it is the default noname package.

Importing Classes

Java programs reference classes that are defined in other packages using the import statement.

Functions and Variables Declared Outside of Classes

Java strictly adheres to a class-oriented approach to program design. It is impossible to define a method or variable outside the scope of a class. At least one public class must be defined within a Java program to support the main () method.

What Is Swing?

Swing is a major component of the JFC, which is the result of a large collaborative effort between Sun, Netscape, IBM, and other companies. Swing provides a large number of useful GUI controls that originated with Netscape's Internet Foundations Classes (IFC). The Swing components go far beyond the IFC, to the point where there is no visible resemblance between Swing components and those of the IFC. Swing also provides the capability to quickly and easily change the *look and feel (L&F)* of a single component or group of components. This capability, known as *pluggable look and feel (PL&F)*, is a hallmark feature of Swing.

I/O:

The java.io package contains the fundamental classes for performing input and output operations in Java. These I/O classes can be divided into four basic groups:

- Classes for reading input from a stream.
- Classes for writing output to a stream.
- Classes for manipulating files.
- Classes for serializing objects.

All fundamental I/O in Java is based on streams. A stream represents a flow of data, or a channel of communication. Conceptually, there is a reading process at one end of the stream and a writing process at the other end. The character stream

classes, which are called readers and writers, handle Unicode characters appropriately.

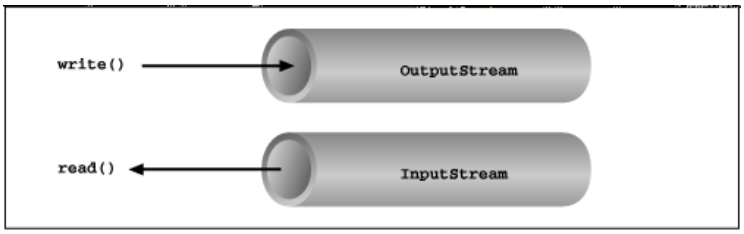


FIGURE 5.1: InputStream,OutputStream

Input Streams and Readers

The InputStream class is an abstract class that defines methods to read sequentially from a stream of bytes. Java provides subclasses of the InputStream class for reading from files, StringBuffer objects, and byte arrays, among other things. Many of the byte-oriented InputStream subclasses have character-based Reader counterparts. Thus, there are subclasses of Reader for reading from files, character arrays, and String objects.

Reader

The Reader class is the abstract superclass of all other character input stream classes. It defines nearly the same methods as InputStream, except that the read() methods deal with characters instead of bytes:

read()

read(char[] cbuf)

read(char[] cbuf, int off, int len)

Output Streams and Writers:

The OutputStream class is an abstract class that defines methods to write a stream of bytes sequentially. Java provides subclasses of the OutputStream class for writing to files and byte arrays, among other things. Other subclasses of OutputStream can be chained together to provide additional logic, such as writing multibyte data types or converting data to a string representation. It is also easy to define a subclass

of OutputStream that writes to another kind of destination. Writer class is an abstract class that defines methods to write to a stream of characters sequentially. Many of the byte-oriented subclasses of OutputStream have counterparts in the character-oriented world of Writer objects. Thus, there are subclasses of Writer for writing to files and character arrays.

OutputStream

The OutputStream class is the abstract superclass of all other byte output stream classes. It defines three write() methods for writing to a raw stream of bytes:

write(int b)
write(byte[] b)
write(byte[] b, int off, int len)

Some OutputStream subclasses may implement buffering to increase efficiency. OutputStream provides a method, flush(), that tells the OutputStream to write any buffered output to the underlying device, which may be a disk drive or a network.

File:

This class supports a platform-independent definition of file and directory names. It also provides methods to list the files in a directory, to check the existence, readability, writeability, type, size, and modification time of files and directories, to make new directories, to rename files and directories, and to delete files and directories. The constants defined by this class are the platform-dependent directory and path separator characters, available as a String or char.
getName() returns the name of the File with any directory names omitted. getPath() returns the full name of the file, including the directory name. getParent() returns the directory of the File. If the File is an absolute specification, then getAbsolutePath() returns the complete filename. Otherwise, if the File is a relative file specification, it returns the relative filename appended to the current working directory.

Socket stream:

Socket Input Stream

public InputStream getInputStream()
throws IOExceptionReturns an input stream for this socket.
If this socket has an associated channel then the resulting input stream delegates all of its operations to the channel. If the channel is in non-blocking mode then the input stream's read operations will throw an IllegalBlockingModeException.

Under abnormal conditions the underlying connection may be broken by the remote host or the network software (for example a connection reset in the case of TCP connections). When a broken connection is detected by the network software the following applies to the returned input stream :-

•The network software may discard bytes that are buffered by the socket. Bytes that aren't discarded by the network software can be read using read.

•If there are no bytes buffered on the socket, or all buffered bytes have been consumed by read, then all subsequent calls to read will throw an IOException.

•If there are no bytes buffered on the socket, and the socket has not been closed using close, then available will return 0.

Closing the returned InputStream will close the associated socket.

Returns:
an input stream for reading bytes from this socket.
Throws:
IOException - if an I/O error occurs when creating the input stream, the socket is closed, the socket is not connected, or the socket input has been shutdown using shutdownInput()

Socket output Stream:

public OutputStream getOutputStream() throws IOExceptionReturns an output stream for this socket.
If this socket has an associated channel then the resulting output stream delegates all of its operations to the channel. If the channel is in non-blocking mode then the output stream's write operations will throw an IllegalBlockingModeException.

Closing the returned OutputStream will close the associated socket.

Returns:
an output stream for writing bytes to this socket.
Throws:

IOException - if an I/O error occurs when creating the output stream or if the socket is not connected.

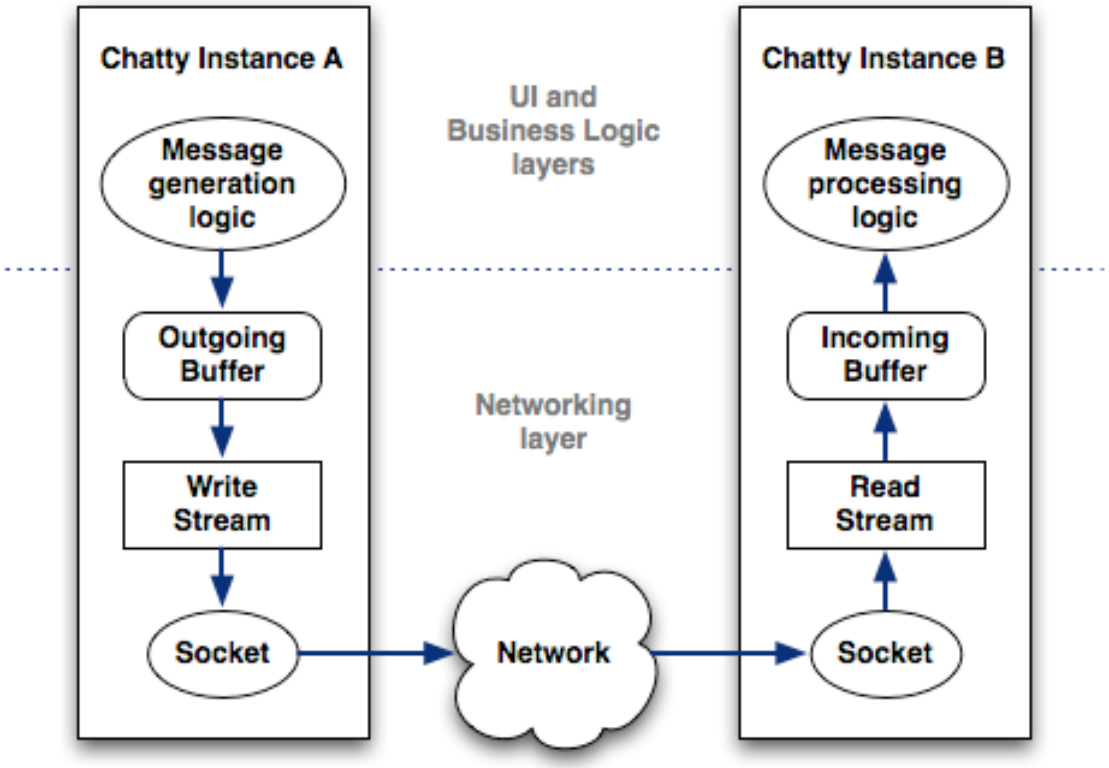


Figure: Socket Input Stream and output Stream

CHAPTER 6
TESTING

Testing Objectives:

a) Testing is a process of executing a program with the intent of finding an error.

b) A good test case is one that has a higher probability of finding an as yet undiscovered error.

c) A successful test is one that uncovers error an as yet undiscovered error.

Our objectives are to design tests that systematically uncover different errors with minimum amount of time and effort. Testing also provides a good indication of software reliability and quality.

Test case Design Methods:

a) White Box Testing:

White box testing, sometimes called glass box testing, is a test casedesign method that uses the control structure of the procedural design to derive test cases. Using white box testing method, the software engineer can derive test cases that

1. Guarantee that all independent paths within a module have been exercised at least once,
2. Exercise all logical decisions on their true and false sides,
3. Execute all loops at their boundaries and within their operational bounds, and
4. Exercise internal data structures to ensure their validities.

b) Black Box Testing:

Black box testing, sometimes called behavioral testing, focuses on the functional requirements of the software. That is, black box testing enables the software engineer that to derive sets of input conditions that will fully exercised all functional requirements for a program. Black box testing is not an alternative to white box techniques. Rather, it is a complementary approach that is likely to uncover a different class of errors than white box methods.

Test strategies:

The test strategy consists of series of different tests that will fully exercise video surveillance The purpose of these tests is to uncover systems limitations and measure its full capabilities .A list of various planned tests and a brief explanation follows below

a) Unit Testing:

Unit Testing makes heavy used of White Box testing techniques, exercising specific paths in a module control structure to ensure completed coverage and maximum error detection.

The Unit test will test all modules of video surveillance individually. The modules to be tested include Desktop Sharing and Controlling, File Transfer, Chatting.

b) Integration Testing:

The above-specified modules are subjected to integration testing in which the tested unit are integrated and made to work as a whole. Testing is done at the interfaces of all the modules, where the communication between the modules is tested.

Black Box Testing techniques are used exclusively during Integration.

c) Validation Testing:

After the software has been integrated (Constructed), a set of high order test are conducted, validation criteria (established during requirements analysis) must be tested. Validation Testing provides final assurance that software meets all functional, behavioral, and performance requirements.

Black Box Testing techniques are used exclusively during Validation.

Test Documentation:

The following documentation will be available at the end of the test phase:

- Test plan
- Test cases
- Test Report

Test Plan:

A test plan documents the strategy that will be used to verify and ensure that a product or system meets its design specifications and other requirements. A test plan is usually prepared by or with significant input from Test Engineers.

Depending on the product and the responsibility of the organization to which the test plan applies, a test plan may include one or more of the following:

- *Design Verification or Compliance test* - to be performed during the development or approval stages of the product, typically on a small sample of units.
- *Manufacturing or Production test* - to be performed during preparation or assembly of the product in an ongoing manner for purposes of performance verification and quality control.
- *Acceptance or Commissioning test* - to be performed at the time of delivery or installation of the product.
- *Service and Repair test* - to be performed as required over the service life of the product.
- *Regression test* - to be performed on an existing operational product, to verify that existing functionality didn't get broken when other aspects of the environment are changed (e.g., upgrading the platform on which an existing application runs).

Test Case:

A test case in software engineering is a set of conditions or variables under which a tester will determine whether an application or software system is working correctly or not. The mechanism for determining whether a software program or system has passed or failed such a test is known as a test oracle. In some settings, an oracle could be a requirement or use case, while in others it could be a heuristic. It may take many test cases to determine that a software program or system is considered sufficiently scrutinized to be released. Test cases are often referred to as test scripts, particularly when written. Written test cases are usually collected into test suites.

Test Id	Do	Expected Result	Actual Result
TA-001	Click on start button of walkie talkie.	Data should be read and transfer.	Success.
TA-002	Enter port no on run, ip address to connect and port no to connect.	Central Station should be connected.	Success.
TA-003	Click on start button to start the central repeater.	-if no error is detected then it will start. -if error is detected it will give error message.	Success. Success.
TA-004	Enter port no on run, ip address to connect and port no to connect.	Repeater Station should be connected.	Success
TA-005	Click on start button to start the repeater.	-if no error is detected then it will start. -if error is detected it will give error message.	Success
TA-006	Enter port no on run, ip address to connect and port no to connect.	Station should be connected.	Success
TA-007	Click on start button to start the station.	-if no error is detected then it will start.	Success

		-if error is detected it will give error message.	
TA-008	Click on view location.	Current location should be displayed.	Success

Test Report:

The test report is the primary work deliverable from the testing phase. It disseminates the information from the test execution phase that is needed for project managers, as well as the stakeholders, to make further decisions. Anomalies and the final decomposition of the anomalies are recorded in this report to ensure the readers know the quality status of the product under test.

CHAPTER-7

CONCLUSION

We have worked with creating simulation of automatic packet reporting system. In this project we have worked with Google earth to navigate the location of the GPS enabled wireless device, for that we have generated the key Hole Markup Language File which is used to locate the longitude, latitude, altitude and velocity of the device on the Google Earth.

As there is database for each station such as police station and central zonal offices. It will store the device information by accepting the string and tokenizing it into id , longitude , latitude , altitude and velocity and then storing it.

The program still lacks some desired functionality but fulfills most of the requirements**.**

We conclude that it indeed provides real-time location of the GPS enabled Wireless Device. User interfaces should facilitate easy interaction among users to enter the port number to run on, port number to connect and IP address to connect.

CHAPTER 8

FUTURE SCOPE

A GPS navigation device is any device that receives Global Positioning System (GPS) signals for the purpose of determining the device's current location on Earth.

GPS devices provide latitude and longitude information, and some may also calculate altitude, although this is not considered sufficiently accurate or continuously available enough (due to the possibility of signal blockage and other factors) to rely on exclusively to pilot aircraft.

GPS devices are used in military, aviation, marine and consumer product applications.

GPS devices may also have additional capabilities such as:

- containing all types of maps, like streets maps, which may be displayed in human readable format via text or in a graphical format.
- providing suggested directions to a human in charge of a vehicle or vessel via text or speech.
- providing directions directly to an autonomous vehicle such as a robotic probe.
- providing information on traffic conditions (either via historical or real time data) and suggesting alternative directions.
- providing information on nearby amenities such as restaurants, fueling stations, etc.
- providing tracking information to forest department.

The GPS program provides critical capabilities to military, civil and commercial users around the world. In addition, GPS is the backbone for modernizing the global air traffic system.

CHAPTER 9

REFERENCES

Books:

1) The Complete Reference - JAVA 6 by Herbert Schlitz.

2) Core Java (Volume I & II) - Sun Microsystems Press

URL:

http://java.sun.com

http://www.java.com

http://www.javaworld.com

https://developers.google.com/kml/documentation/kmlreference