|  |
| --- |
| tralee institute of technology |
| Year 3 Computing Project |
| Android Mouse Application |
|  |
| **Christopher Donovan T00154755** |
| **1/15/2013** |

|  |
| --- |
| Develop an Android app that lets the user control the cursor on the screen with their smartphone. The phone will display two buttons similar to a conventional mouse. When the user moves the phone and clicks the buttons the phone will have all the functionally of a mouse. |

Contents

[About the Project 2](#_Toc354712338)

[The Use Case Diagram 4](#_Toc354712339)

[Flow of Events 5](#_Toc354712340)

[Setting up the Android SDK and Eclipse 7](#_Toc354712341)

[Hardware 7](#_Toc354712342)

[Algorithm Problem 7](#_Toc354712343)

[The solution 8](#_Toc354712344)

[Creating the Application 9](#_Toc354712345)

[From the Android API 9](#_Toc354712346)

[From the Java API 9](#_Toc354712347)

[After the Connection 10](#_Toc354712348)

[Application Life-Cycle 16](#_Toc354712349)

[Installing the server: 18](#_Toc354712350)

[Installing the client: 18](#_Toc354712351)

[Instruction Manual: 19](#_Toc354712352)

[The Menus Srceen 22](#_Toc354712353)

[Bibliography 25](#_Toc354712354)

[Overview of the Android App Class Diagram 26](#_Toc354712355)

[Overview of the Java App Class Diagram 27](#_Toc354712356)

# About the Project

**Project Title**

Android Mouse Application

**Project Description**

The original idea for this project was proposed by Brian Clarke (T00154737).

Android app that lets the user control the cursor on a screen with their smart phone. The phone display would have an image of 2 buttons similar to a conventional PC mouse. When the user moves the phone in space and clicks the mouse buttons on the phone, the phone will have all the functionality of the mouse.

The phone will act like a cross between a Bluetooth mouse and a Wii remote.

**Main Technical Challenge**

The android app will access the accelerometer on the phone and transmit coordinates via Bluetooth the cursor on the screen. The system will need to constantly process coordinates based on not only the phone’s direction with reference to the screen but also the phones location within the 3D space in front of the screen. The phone will record the starting coordinates. After initial setup the system will need to process the data from the built in sensors and adjust for aspect and distance from screen.

An app will also run on the PC/Laptop that will process the coordinates received from the phone and using java’s Robot class (mouseMove, mousePress and maybe keyPress) provide the functionality. It would be an advantage to have an ‘Esc’ button on the phone to undo a right click and an added feature could be a complete keyboard on the phone which interacts with a text box or editor on the screen.

This app could also be used to control projectors in a classroom situation or media on TV’s (with Windows Media Centre)

**It is proposed to work with Hardware/Software/IDE/API;**

* Sony Ericsson Walkman using Android OS 2.2 (Gingerbread).
* Phone sensors; Accelerometer
* Eclipse
* Java API, Android API
* Android ADT and SDK
* Connectivity via Bluetooth
* Languages; Java, XML
* Shared-Preference to store user setting/Locale settings.

**Use Cases:**

Control a cursor on a PC screen from a remote location (within range).

The user should be able to select/deselect items on a desktop window.

**Addition User Case:**

The user should be able to access and surf the internet by controlling the cursor and transmitting text input from the phone.

# The Use Case Diagram

The user starts the server.

The user opens up the app on their phone.

Client starts, once connected.

Service starts passing data from the service to the client.

Once the server receives the data.

Coords are remapped to the screen. Cursor moves….

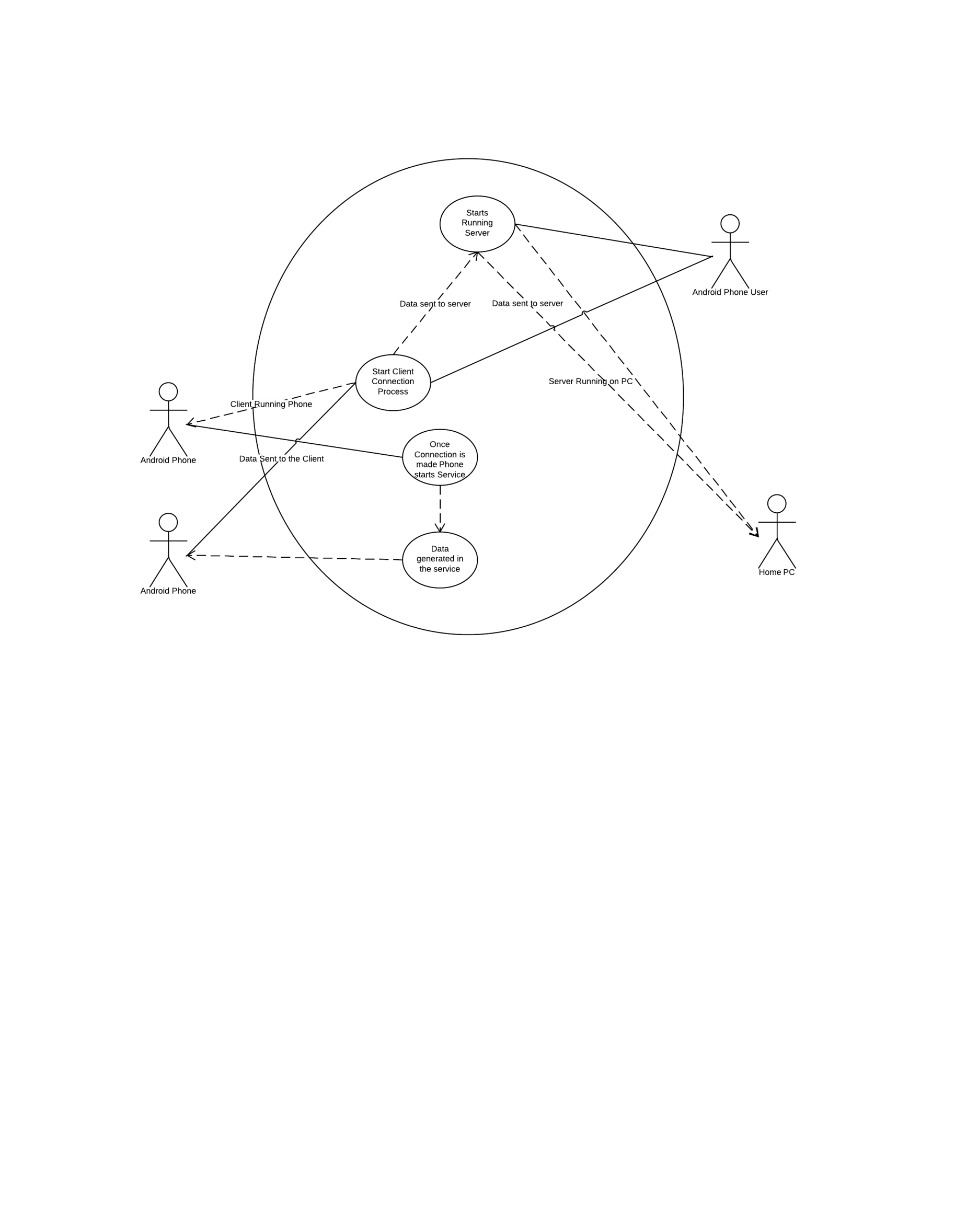


Figure Main Use Case Diagram

# Flow of Events

Open the App, starting the client and service.

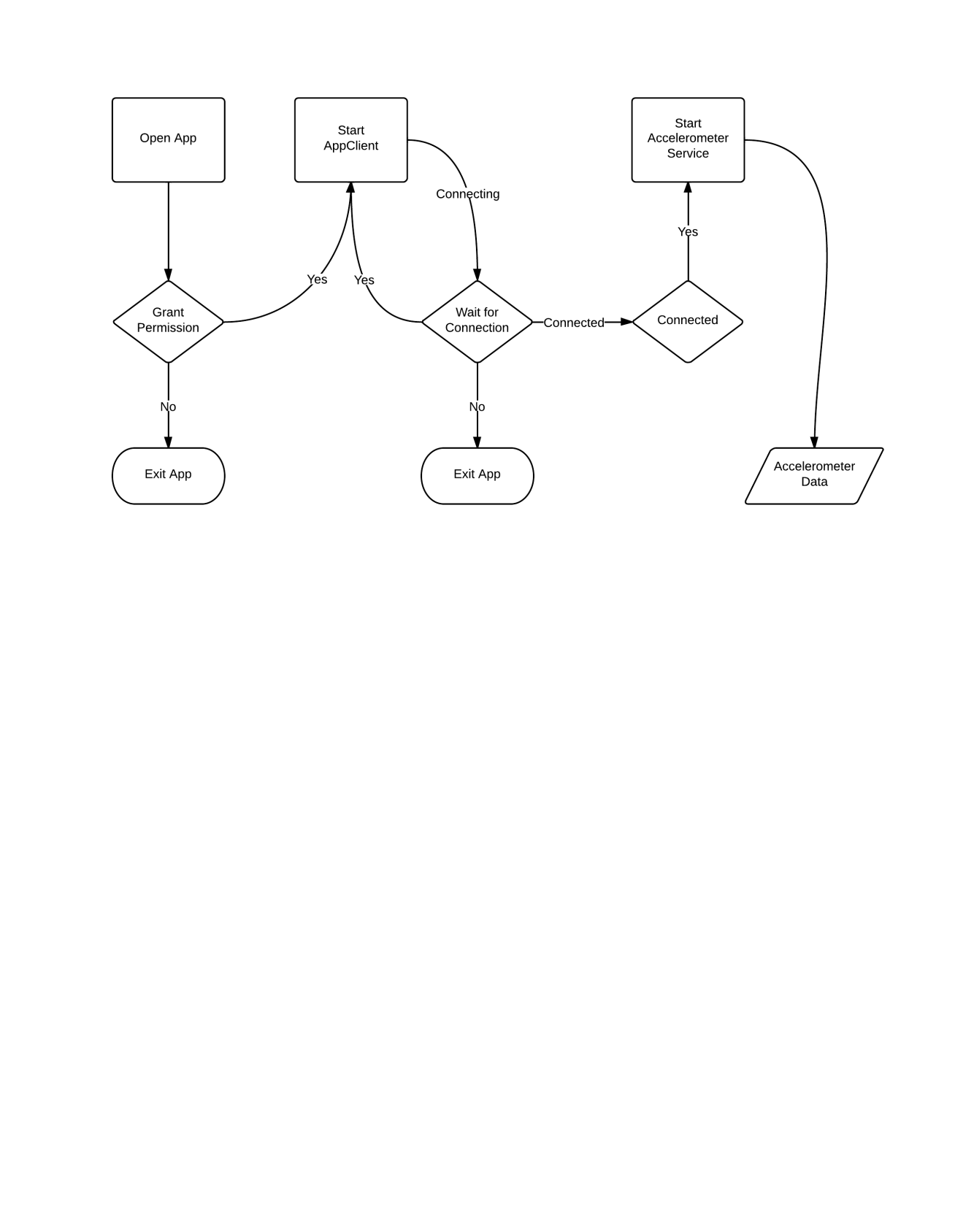


Figure Opening the App

Sending the data across the network.

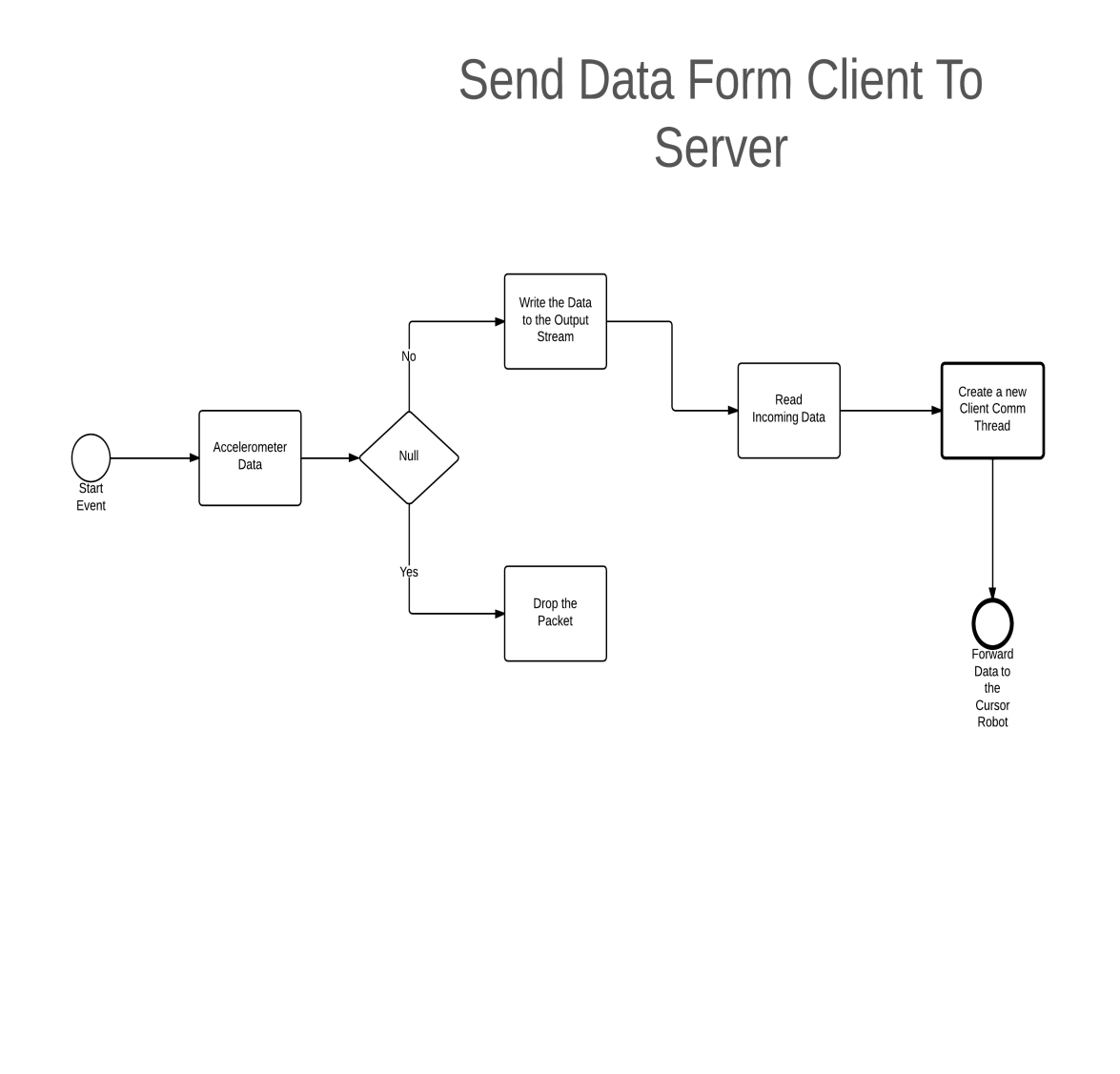


Figure Sending Data Across the Network

The Intiantion Process

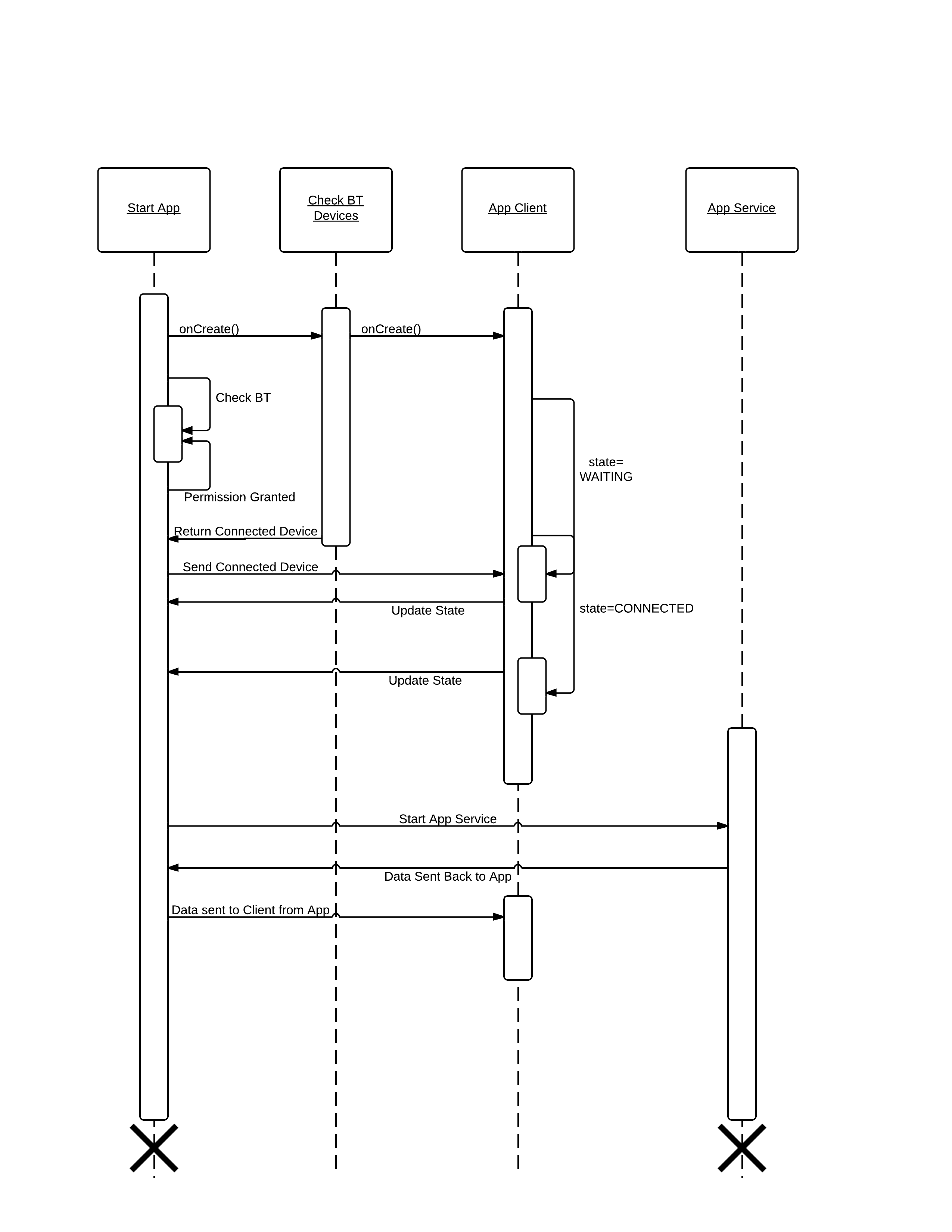


Figure the Connection Process

# Setting up the Android SDK and Eclipse

Both android and eclipse are new technologies that I haven’t used in the past. The android developer website is the main jumping out point for any new android developer and I started by downloading the ADT bundle pack for windows, as both my PC and laptop runs on this platform. I did have some initial issues installing the SDK on windows 8 but after a week or so there was an update release that solved the issue.

# Hardware

In order to develop this I am using my Sony Ericson WTl9i android phone which is running on Android 4.0.4 Ice Cream Sandwich (API level 15).

I also need a blue tooth dongle as neither my PC nor Laptop is Bluetooth enabled. The dongle will allow me to transmit data over a TCP (transmission control protocol) serial link.

# Algorithm Problem

This is a difficult algorithm to create as the Accelerometer data is measures in g the force of gravity that is applied to the axis(s). In order to measure the distance that the phone has travelled the simplest formula is:

Distance = Velocity \* Time

The issue with this algorithm is that it means firstly:

A reading would need to be taken and measured in order to obtain the velocity and time, and applied again to the next reading meaning that there would be some overhead in the speed with could lead to some latency.

Secondly:

In order to avoid double integral formula using only the accelerometer sensor, as this can lead to huge amount of drift (TALKS, Google Tech, 2010) . The way around this is to use a combination of accelerometer and gyroscope sensors, of which unfortunately I only have one available the accelerometer.

The gyroscope is a relatively new sensor and is only available on some high end android and iPhones, I have tried to find a definitive list of android device that carry the gyroscope, and this list on the android forum website is the best I could find (HHENNE, 2011).

# The solution

In order to processing the reading’s as quickly as possible the next best solution is use the tilt and pitch of the phone, which means dividing the phone into quadrants

The next step is to use the accelerometer sensor data to determine which axis is moving i.e:



Figure The Phone Quadants

If –x and –y

then

top left corner is facing down then move to the bottom right

else if –x and y

then

top left corner is facing up then move to the up right

else if x and –y

then

top right corner is facing down then move to the bottom left

If +x and +y

then

Figure Phone Quadrants

top right corner is facing up then move to the up left

This is only a simple interpretation of the phone movements. The values would then be added or subtracted from the current position cursor.

Another issue with the accelerometer sensor is the noise that it generated in order to filter the noise the force of gravity must be removed (TALKS, Google Tech, 2010) (ANDROIDSITE, 2013).

## Creating the Application

## From the Android API

From the Android Documentation I will be using the android.bluetooth APIs which provide the necessary classes need to establish a Bluetooth connection. The Bluetooth connection process involves:

* Requesting permission from the user to enter the Bluetooth process.
* Checking that the phone’s Bluetooth option is set to on.
* Scanning for nearby by devices.
* Connecting to a selected device.
* Storing the device information. (ANDROIDSITE, 2013)

I will also need to have a service running in the background that consistently process and update the accelerometer data form the phone. Along with this service I will need to have a client running also in the background. The client and service both need to communicate main activity class.

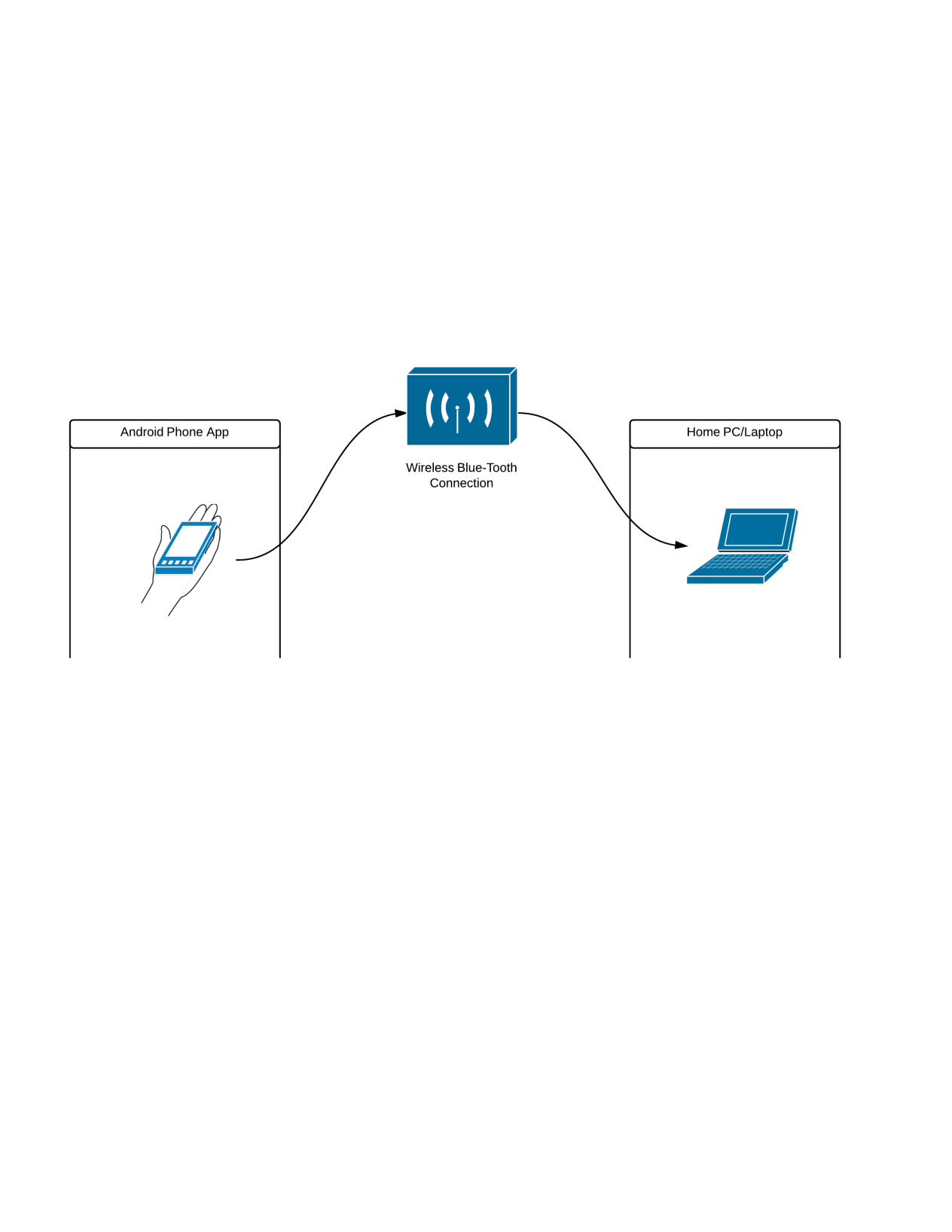


Figure Bluetooth Wireless Connection

## From the Java API

In order for my application run I must provide a client/server mechanism that will allow my application sitting on the phone to interact with the program sitting on the end-users device. I will be using the java.net package which includes the Socket and SeverSocket classes needed to maintain the connection between the server and client programs. I will also be using the Java Robot class in order to control the mouse cursor position on the screen.

**The client program:**

The client program will run on the android phone a connection has being established the client will connection to the listening port on the server and passed the current phone position to the server program.

**The server program:**

The server program will sit on the end-users device once a connection has being established the server will begin to listen for the client application. Once the client is connected to the serve the data will be passed to the Robot object in the server application this will control the position of the cursor on the end-users device.

**The client/server process:**

* Start the server and open a stream connection.
* Client sends a message looking for the server port number.
* Server returns the port number to the client.
* The connection is established using TCP (Transmission Control Protocol), which is a reliable data transfer protocol.
* The data being transferred will be the starting co-ords of the x, y position, and the current x and y position each time the mouse is moved.

## After the Connection

Once the connection has being established the movement algorithm will be used to determine the new position of the mouse on the screen. The data will have to be in constant flow in order to ensure that the movement is synchronized. Must modern day android phone have built in sensors; I will be using these sensors to determine the new position of the phone in 3D space. The Accelerometer object will allow me to determine the movement of the phone and also the speed at which the phone is moved. There are four different types of motion measurements:

* Gravity Sensor
* Gyroscope
* Linear Accelerometer
* Rotation Vector Sensor

I will be writing and testing the four different types in order to determine the best method to use for this application is. The data transfer rate will also be a consideration for this activity, because of the mouse’s constant movement the new position will have to be determined instantly, to ensure there is no latency between the movement of the device and the movement of the cursor on the screen.

# The Blue-Cove Jar File

# The Accelerometer

# Threading

# Network Programming

## Application Life-Cycle

The activity life –cycle is a built in system process that allows the application complete a cycle. When a activity is started the onCreate() method is used to instantiate the operations and attributes for that activity, this method takes a Bundle as an argument the Bundle object contains the current state of the objects. This method must be implemented in each Android activity.

Figure The Android Activity Life-Cycle (ANDROIDSITE, 2013)

My first activity is the connection class in my create method I will be calling the Bluetooth adapter object and established the Bluetooth connection after that I will be calling the TCP connection object again, to establish the link between the client and server classes.

I will then be passing the current activity state to the main application class. Using the connection I will be passing the current position of the mouse to the server Robot class, which will control the cursor on the screen.

The Main Android Activity Methods

onCreate(Bundle savedInstanceState); //this method is created when the activity starts

onStart(); //this method starts the activity

onRestart();

onResume();

onPause();

onStop();

onDestroy();

|  |
| --- |
| tralee institute of technology |
| User Manual |
| Android Mouse Application |
|  |
| **Christopher Donovan T00154755** |
| **1/15/2013** |

# Installing the server:

Install the project CD navigate to

<your drive>:\androidmouseproject\MouseServerEXE

And the appServer.exe file is there; double click to run the server.

# Installing the client:

Unfortunately creating a published .apk file took longer than I thought and I ran out of time. This means that the will have eclipse Juno installed on their machine:

For those who do:

Select create a new project; now click others, android, from existing project, and select the path to the folder on your drive and finish.

Run the server before the app.

# Instruction Manual:

Granting Permissions:

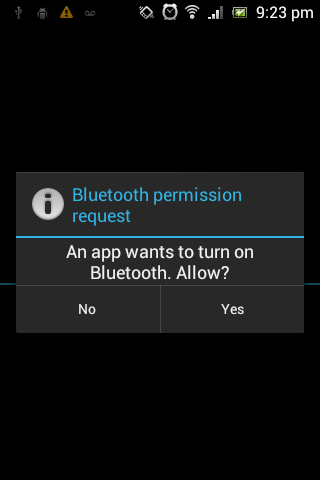


Figure 9 Opening Screen

The opening screen of the application when Bluetooth is turned off, the user must grant the application permission to turn on the Bluetooth.

If permission is granted the application will change the state of the Bluetooth object to ON.

If permission is not granted the application will shut down, also if Bluetooth is not enabled (available) on the device the application will be shut down.

Pairing Request:



Figure 10 Pairing Device Request

If the user grants permission the application will turn the Bluetooth state to on. The user will then be the device will then scan for available devices. An available device must be discoverable to the phone. Once he allotted time has lapsed the user will be asked to pair to device to the available device. The default time is set to 150 seconds; I have changed that to make the device discoverable 300 seconds. In order to use my device the user must be paired with the host device that will run the server.

Pairing Devices:

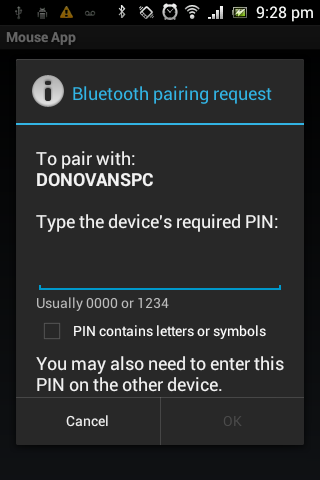


Figure 11 Enter the PIN

Once the user select to pair the devices they will be asked to enter a pin. The pin is only used for that device and session. If the user disables bluetooth they will have to re-pair the devices. After entering the pin on the phone the use will be prompted to enter the same pin on the host device. Once this is compeleted the devices will be connected.

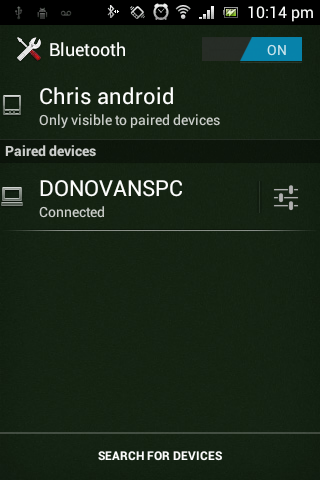
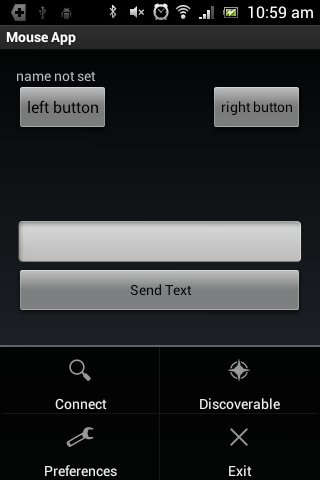


Figure 12 Connected Devices

# The Menus Srceen



The main application screen with and with the option menu inflated. The screen itself is simple all of the real work is going on in the background. Th eMenu option can beaccessed through the normal andnroid menu button (on my phone bottom left).

**Connect:**

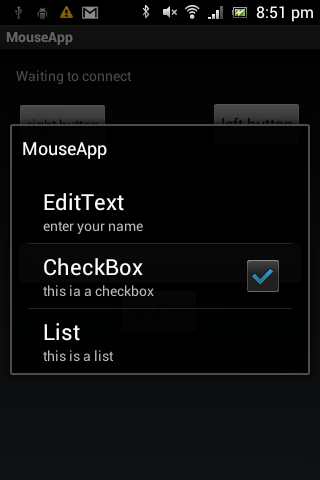
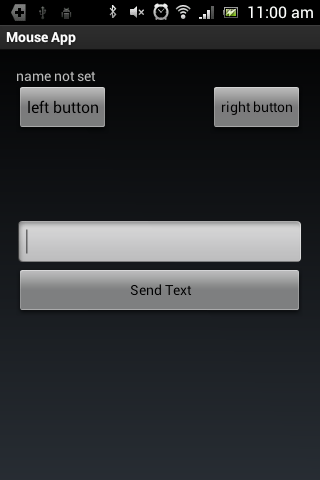
The user will have the option to connect or reconnect if the phone it not connected.

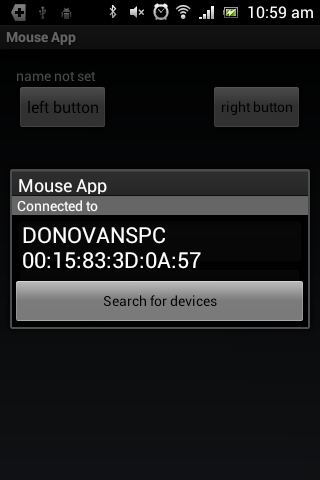
**Discoverable:**

The users Phone will be discoverable to all bluetooth devices in there area.

**Preferences:**

The user will be able to select different option in order t customize the app to there style. These will inculde changing the display name, selecting the color of the cursor and turning on and off the data display.





# The Conclusion

This was the most channelling project I have done, all new concepts, new software, new tools and new to android.

The android set up did take some getting used to, especially the onPause(), onResume() process. Also the R file and the manifest file are too files that you know about until they don’t work. Overall I enjoy learning android and will definitely be keeping it up.

The project itself just seem to go form one issue to another, the Bluetooth is temperamental at best, the accelerometer is very noisy at least one is. I found some of the project to be very changeling one of those parts was the threading process, trying to understand and implement the thread processing was most difficult.

# Bibliography (VOGELLA, Lars, 2012)

ANDROID DEVELOPER. n.d. *permissions.html*. [online]. [Accessed 17 Mar 2013]. Available from World Wide Web: <<http://developer.android.com/guide/topics/security/permissions.html>>

ANDROIDSITE. 2013. *Android*. [online]. [Accessed 20 Jan 2013]. Available from World Wide Web: <<http://developer.android.com/index.html>>

BLUECOVE. 2008. *BlueCove API*. [online]. Available from World Wide Web: <<http://bluecove.org/bluecove/apidocs/overview-summary.html>>

GARGENTA, Marko. 2011. *Learning Android*. California: O'Reilly.

HHENNE. 2011. *Androids with gyros*. [online]. [Accessed 14 Feb 2013]. Available from World Wide Web: <<http://androidforums.com/android-lounge/308045-androids-gyros.html>>

*How to write a test plan*. [online]. [Accessed 2013]. Available from World Wide Web: <<http://www.wikihow.com/Write-a-Test-Plan>>

JAVAREVISITED. 2012. *javarevisited*. [online]. [Accessed Mar 2013]. Available from World Wide Web: <<http://javarevisited.blogspot.ie/2012/01/difference-thread-vs-runnable-interface.html>>

LEE, Wei-Meng. 2012. *Android 4 Application Development*. Indianapolis: Wiley/Worx.

MEIER, Reto. 2010. *Adroid 2 Application Development*. Indianapolis: Wiley Press.

OOPSOFT. *Thread Processing*. [online]. [Accessed 2013]. Available from World Wide Web: <<http://coopsoft.com/ar/AndroidArticle.html>>

TALKS, Google Tech. 2010. *Sensor Fusion on Android Devices: A Revolution in Motion Processing*. [online]. [Accessed 14 Feb 2013]. Available from World Wide Web: <<http://www.youtube.com/watch?v=C7JQ7Rpwn2k>>

TRAVIS. 2010. *Android Application Development*. [online]. [Accessed 20 Jan 2013]. Available from World Wide Web: <<http://thenewboston.org/list.php?cat=6>>

VOGELLA, Lars. 2012. *JUnit Tutourial*. [online]. [Accessed 2013]. Available from World Wide Web: <<http://www.vogella.com/articles/JUnit/article.html>>

# (How to write a test plan) (BLUECOVE, 2008)

(OOPSOFT) (JAVAREVISITED, 2012)

# C:\Users\Christopher\Documents\GitHub\androidmouseproject\MouseApp\MouseAppDiagrams\Android_Package.gifOverview of the Android App Class Diagram

# Overview of the Java App Class Diagram

