**Indoor Positioning System**

**CS4900**

**Senior Design Project**

**Group Members**

**Davion Teh**

**Kelvin Yap**

**Rodney Dulin**

# Index

* [Index](#h.9e21dqukkh5r)
* [Abstract](#h.3hjzlyv36gvt)
* [Introduction](#h.kxbpus2qlmnd)
* [Project Goals](#h.xfrx8knqcsdq)
* [Background](#h.2et92p0)
* [Redpin Components](#h.2nej3dj10ylf)
* [TCP/ IP Server](#h.2gf78cxtpspj)
* [USER POSITIONING](#h.82a92fnhqc5h)
* [Interval Labeling](#h.x5oz33ts4pwv)
* [Locator Algorithm](#h.rmkaecu9qhr3)
* [Motion Detection](#h.7ix4t6s0luts)
* [Stories](#h.tyjcwt)
  + [Stories Cont. Functional](#h.avigfcq5wek0)
* [Spikes](#h.3dy6vkm)
  + [Phase 1](#h.cugy66czt6qh)
  + [Phase 2](#h.84n5tk79bmsj)
* [Legal](#h.r4656n63rsp)
  + [Licenses](#h.ogdkrsdegx0s)
  + [Intellectual Property (IP)](#h.bjmmje3d53un)
  + [Non-Disclosure Agreement (NDA)](#h.d0t7iowbbyfq)
* [Resources](#h.1jbj286z02h8)
* [Feasibility](#h.ekxlcjndlh4q)
* [References](#h.3rdcrjn)
* [Glossary](#h.2s8eyo1)
* [API - Application programming interface](#h.ktxks5d9g4v7)

# Abstract

We are living in the age where people are heavily relying on navigation technology for travelling purposes, be it long or short distance. The rise of Global Positioning System (GPS) technology, its high precision and accuracy to be able to guide travelers correctly to their destination has spark interest and demand for more of this technology. However, the downside of GPS is that it is rendered useless when the device is under some sort of cover like building or tunnels. After many attempts to solve this issue, a new system has been discovered to cover what GPS could not do. The new tracking system is called Indoor Positioning System.

The reason why GPS does not work indoor is that solid matter blocks the signals it tries to send to the satellite. IPS on the other hand does not rely on satellite; instead it relies on sending signals to multiple WIFI routers (which is also called an “Access Point”) then receives its current location from the server after some geological math computation. This new discovery may sound easy, but in practice it is currently very difficult to efficiently and accurately track the position of a device with the current system design.

This system requires heavy infrastructure deployment of access point to even "try" to achieve an accuracy of at most 1m error. There is a lot of parameter that has to be considered (for example: accuracy, coverage, integrity, availability, update rate, latency, costs, infrastructure, privacy, approval, robustness, intrusiveness and etc.) and there could be more that needs to be included but has not been discovered to be significant currently.

This report displays the resources and information required for this IPS project, and to determine if this project could be completed by the end of Fall 2014.

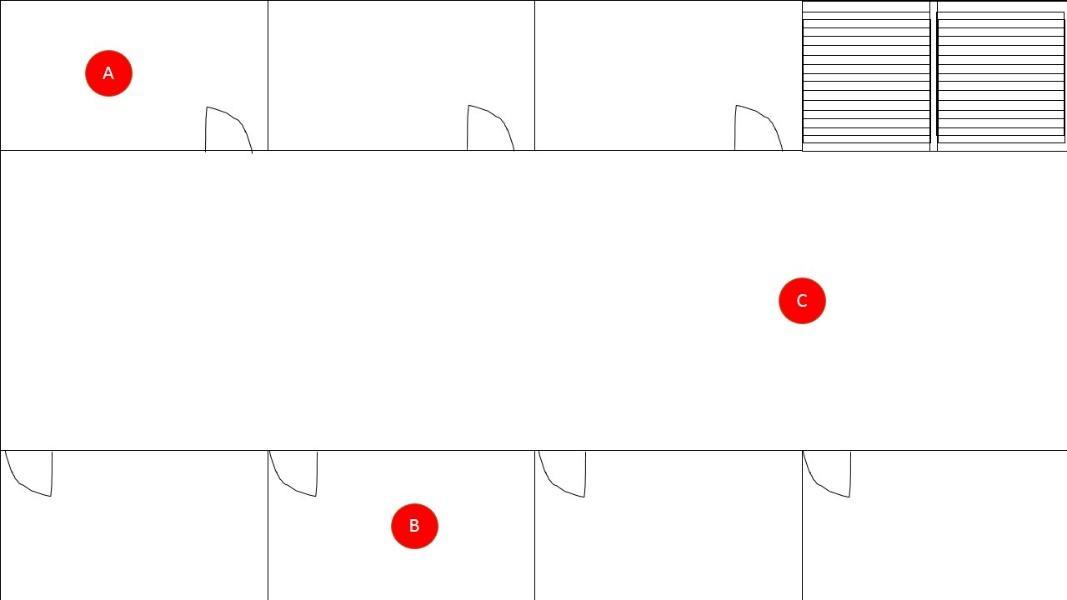
# 

# 

# **Introduction**

Global Positioning Systems(GPS) are not an accurate form of indoor position tracking because of obstructions to the signal. A solution is needed to track indoor positions. In brief context of how Indoor Positioning generally works (since there is more than one approach), imagine that there are two people, A1 and A2, standing at the end of a hallway which acts as access point. Whenever someone enters or exits the hallway, they will have to pass by A1 or A2 that are standing next to the entrance/exit of the hallway. A1 or A2 will then report the details of the person that had just passed by to the server, and the server will then know the location of the person because he has just recently in proximity of A1 or A2. This is one of the approaches to create an IPS, which is called “Choke Point Concepts”.

We used an open source code called “Redpin” which developed with the goal of providing at least room level accuracy while it not needing to do trainings and setup phase, which is very time, costly. Redpin uses a “fingerprint system”, meaning it does not provide geographical coordinate and instead it provides “symbolic identifiers”.

*Figure 1- Location are determined by Symbolic reference instead of Geographical coordinate*

Further details such as IPS concept, Redpin’s approach, what we are trying to do will be further discussed in the main part of this report. The system is based on 3 different parts:

* Wi-Fi Access Points [AP]
* A positioning server
* An Android device

There are two phases to consider in this system. The first phase called offline (calibration) supposed to build a RSSI map. The second one called online (positioning) supposed to locate the device in an indoor environment (room). The whole project was built on a Windows Machine to allow us to develop with all the environments and the IDE needed.

We used the Android SDK with the ADT plugin for Eclipse to create our different android applications.

# 

# Project Goals

Phase 1:

* Able to deploy the application to work on Android tablet 4.x
* Should at least provide room level (i.e., small room) accuracy
* Must be able to adapt to changes in the environment

Phase 2:

* Develop a function where a button is pressed and send status message to the server side
* Improving Location Fingerprinting through Motion Detection and Asynchronous Interval Labeling

# Background

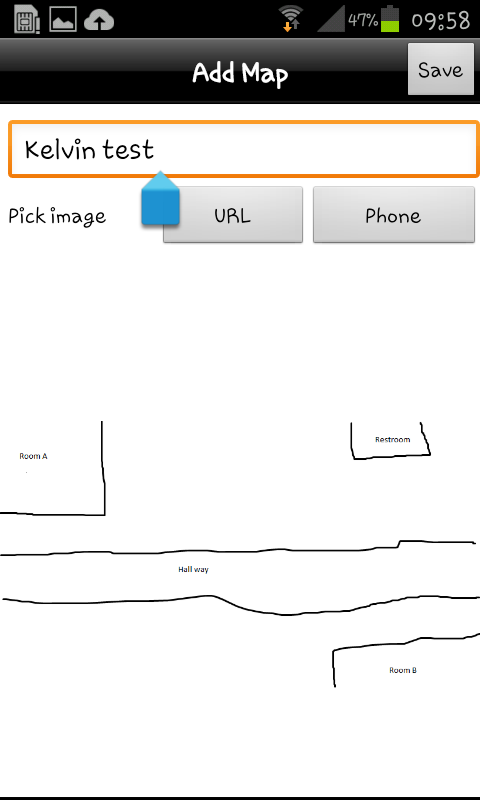
There are many solutions to indoor positioning. Many of them are not open source and can be very expensive. Projects like wifarer or Meridian use a proprietary system and let you create your app using their SDK. We live in a world where technology is available and changing daily. We did research in this subject and decided we needed something open source, used on an open source platform preferably Android, and something user friendly. These reasons are why we chose to update the Redpin project.

The aim of the Redpin project is to create an open source systems that allows indoor positioning up to room-level description using android mobile platform. The project was first designed and written in 2010 by a group at the Institute for Pervasive Computing in Zurich, Switzerland. It was originally developed for Android version 2.2. Redpin is originally based on known systems like Place Lab and Radar. The client for this project is designed for Dr. John Kapenga. Redpin is a fingerprint-based indoor localization system designed and built to run on an android mobile platform. With Redpin, it uses WIFI access points on a mobile phone to detect the user location within a building.

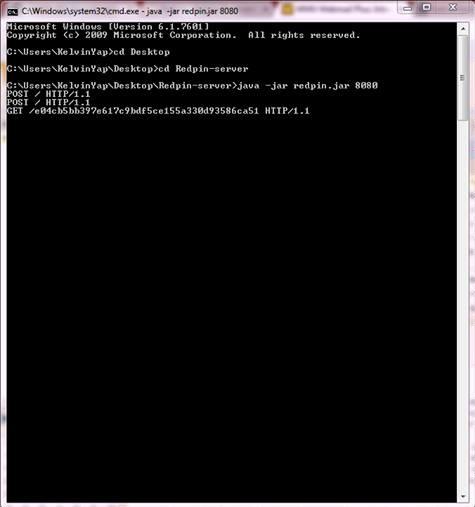
To achieve such high accuracy from the noisy WiFi signal, however, such systems require prior manual calibration

Localization of a mobile user or device can be reduced to the problem of mapping a set of RSS measurements to a known symbolic identifier, for example a room number. The unique measure to a location for this project is called a *fingerprint.* Note that, many fingerprints can be assigned to the same location.

In order to achieve a room level accuracy, by selecting the correct location given a measurement, we measure the signal strength of all Wi-Fi access points that are in range. On the other hand, reducing the problem to this simple mapping of room-level identifier entails the advantage that rooms are divide by walls that absorb and/or reflect electromagnetic radiation and thereby contribute in making a measurement unique.



**Figure - Uploaded map on redpin being displayed in device**



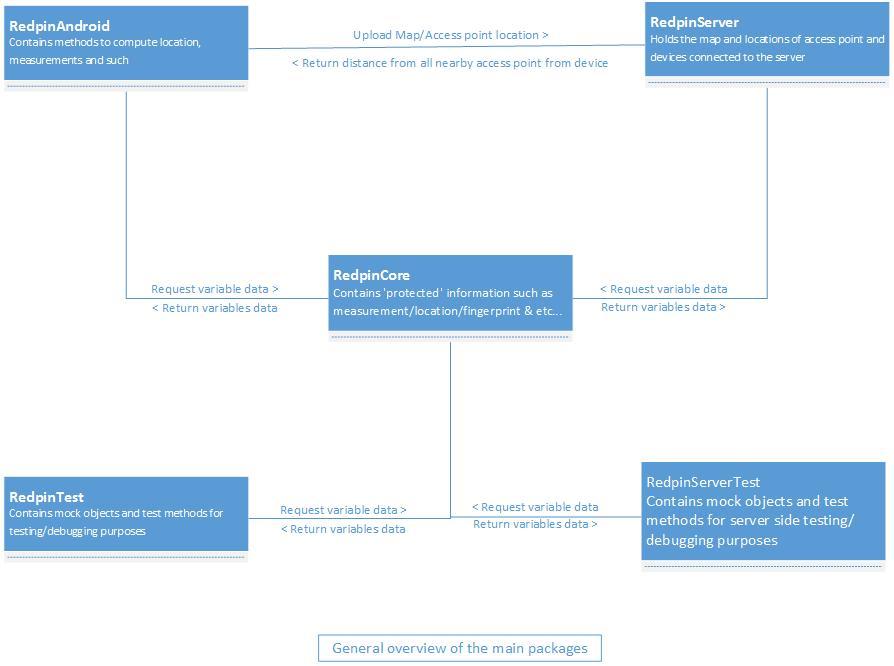
**Figure - Server displaying log showing a file has been uploaded to server**

The key concept of this indoor positioning system project is to allow users of the system to create and manage the locations (i.e., the symbolic identifier that denote a location) in a collaborative way. Using this Redpin application, every user can create, modify and, most importantly, use location information that was created by other users.

Our main implementation for this project is to be able to get all android devices contribute to the system that uses the same database of fingerprints. Location systems typically output Cartesian coordinates, which, for indoor

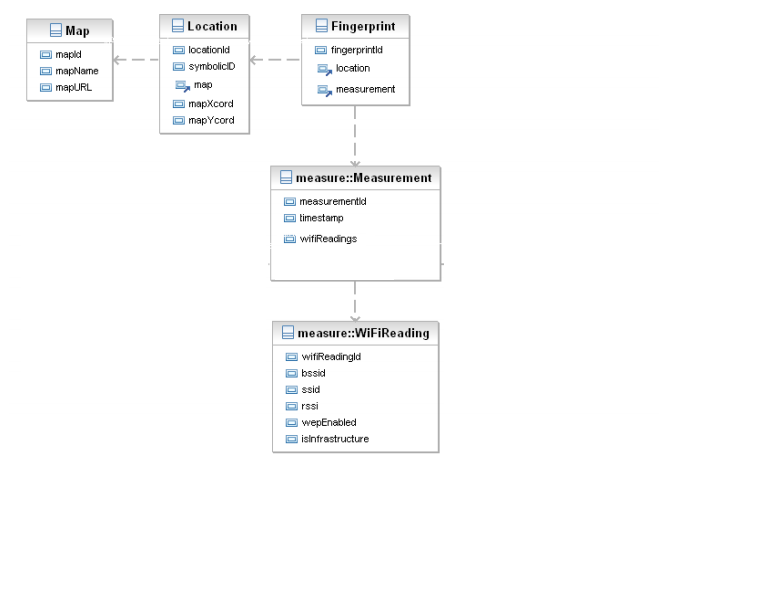
settings, are often mapped to rooms based on available map data. This process allows to easily share knowledge about locations and enables of quick mapping of that building. However for this project, this implementation aspect would bring up security issues/implications which are yet to addressed in our current work.

The Redpin project contains a server, the mobile client, which in our case will be Android, and the Redpin core files. The server allows clients to store fingerprints in a database and also lets the clients receive maps. The maps are images of the floor plan in this case. It also locates the mobile device. Redpin Android is the mobile client, these contains all code that is used to make the application run on the mobile device. The device can be used to upload maps to the server, display locations and any other nearby fingerprints.



Thus, after installing the software (apk file) into the android platform, we wanted our localization system to be user friendly and easy to setup. Our system will have no information about neither the building, nor the Wi-Fi access points. During initialization, the application is measuring the RSS of all WiFi access points that are in range. This process is known as “*packet sniffing*”. The key concept of Redpin is to let the clients of the system to create and manage the locations manually. Thus, the user can choose from a list of known floor plans to that user current position. In addition, several fingerprints may be stored for the same identifier with a different timestamp. This mechanisms allows the Redpin to be able to continuously updating to adapt to changes in the electromagnetic environment.

In order to display not only the name of the current location but also show the position on the floor plan, the system must be given image files of each floor. These images can be uploaded to the server anytime.

Figure : Redpin Data Model

### Redpin Components

Redpin consist of two basic components: a *Packet Sniffer* component that gathers and collects information about different wireless devices in range in order to create a *fingerprint*. The second component is a *Locator* component, which stores measured fingerprints in a central database(repository) and contains the algorithm to locate a mobile devices for our cases which only support android platform. The sniffer component has and can only run on the android mobile device, the locator component will be run on a central server.

### TCP/ IP Server

The Redpin carries its duties and provides several services for android mobile clients. First, the server provides a service that allows client to store fingerprints in a central database. This particular service will be called whenever an android mobile users stores or redefines a location. Next service will allows the mobile clients to retrieve maps, for example images of the floor plan that are associated with a certain location. Lastly, the server will provide a location services for that user. It will be able to retrieve that user fingerprint, and thus the location of that user can be detected according to the best matches RSS measurements.

### USER POSITIONING

To determine the current location of that android mobile device, we will need to find the one known fingerprint that matches the current measurement best. If the current location cannot be predicted accurately, the user is prompted to enter a label for his or her current location. By allowing multiple measurements for the same location and by collecting WiFI fingerprint Redpin can provide the correct symbolic location.

### Interval Labeling

Using intervals also enables a different kind of labeling. By detecting intervals of device immobility, the system can defer location labeling to a more appropriate time, and refer to longer time periods that are easy for users to remember.. This greatly improves the user experience, as users need not provide labels while at the labeled location, where they are likely engaged in some other activity.

Interval labeling allows the user to update the radio map with all data taken while the device is stationary. Because intervals provide more cues to users(starting time, ending time, and duration of interval), users are more likely to remember where they were during an interval that at an instant.

### Locator Algorithm

In order to locate the user, the locator compares the current measurement with all known fingerprints in the central database by calculating the distance measure(Interval labeling). If a fingerprint can be detected whose distance to the current measurement is smaller than the threshold, the associated location will be returned back to the user. If multiple fingerprints are found, the system will thus return the best matches.

### Motion Detection

This location system perform motion detection, it uses WiFi signal strength to both localize a device and infer whether it is moving. If the device is stationary, the system does not need to computer the user position(which might interfere with regular communications as both activities share the same WiFi card). In contrast, we use motion information not only for position, but also aid the training: If the device is stationary, the system can collect stable WiFi measurements.

# 

# Stories

Having Indoor positioning system as our developing experiment target, our group has been searching the net for information regarding the IPS. We found an open source code called "Redpin" which the goal of its development is to create a new concept of IPS system, after checking through its documentation, it is a form of IPS and since it is open source, we decided to use this as our main experimental development.

At first we wanted tried it out on our IPhones and Android (and tablet) to test if the cores are working. However, we were unable to get it running for quite some time (a week plus) for unknown reasons. After some further research for possible solution, we have found that Redpin is developing to work only for Android 2.2 (not earlier version, not later version, must be exactly version 2.2). This makes us set our first objective to first update it to a version of android that are usable now, which is Android 4.3 Kit Kat.

When we try to build build the code into our IDE (Eclipse with Android Development Platform), Here comes the funny part, the program itself could not be built even though we have not touch anything, it seems to contain errors that we could not be comprehend (because we don't understand how it works, yet). So then, it has been stuck for about 2 weeks and we were almost going to drop this project because if we cannot even get the code to build, there might be the case where the code does not even work in the first place.

After a few weeks of trying, Kelvin manages to find the reason behind it. He noticed that there were missing source code in order to build this Redpin project. He found that besides the Redpin Android, it requires also the Redpin Core and the Redpin Server to be on the same build path.

At the same time, we were handed a few android tablet of version 2.2 and we tested the Redpin on it, the result was that it manages to run, but is not working because (1) there was no server, (2) there was no access point. So, the mini checkpoints that has to be done first now is to first update the program to v4.3 Kit kat usable, then setup a server and check to see if it could communicate with the server, then lastly setting up access point and check if it could communicate with the access point and make sure all 3 (device, access point and server) are working. The positioning server where the databases are all re-created.

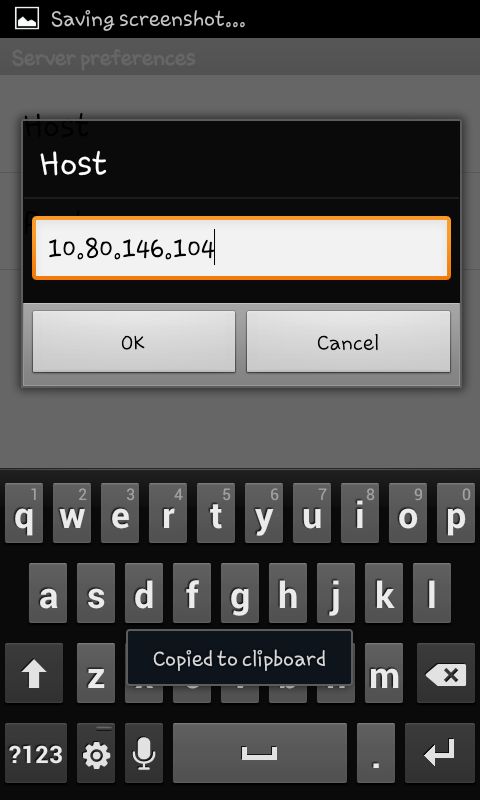


Figure - Attempt to connect to server from device using LAN IP of server

Code will need to be added or removed when upgrading. For example after upgrading to Android version 3.0 and above will need an action bar added including a menu button. Tablets and some phones do not include a menu button and will Need this to get to a menu of options.

Figure - Main page of redpin in smartphone android



Figure - Redpin’s main page with no map in android smartphone (Tablet does not have the bottom 3 options)

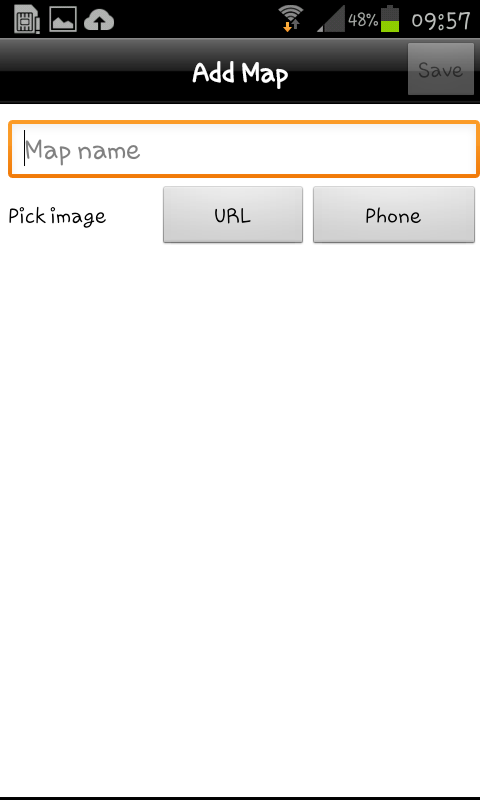
After modifying a few lines in the version code section, we have manage to get redpin installed in a version 4.1 and 4.3 Kitkat android, at an attempt to see if the original function still works, we tried uploading an image to the server from the device and it works fine, but another attempt to reupload the same image will cause the device to crash, our hunch tells us that it might be a missing method to handle “image validation”.

Figure - Attempt to upload image to server



Figure - Image being uploaded to server

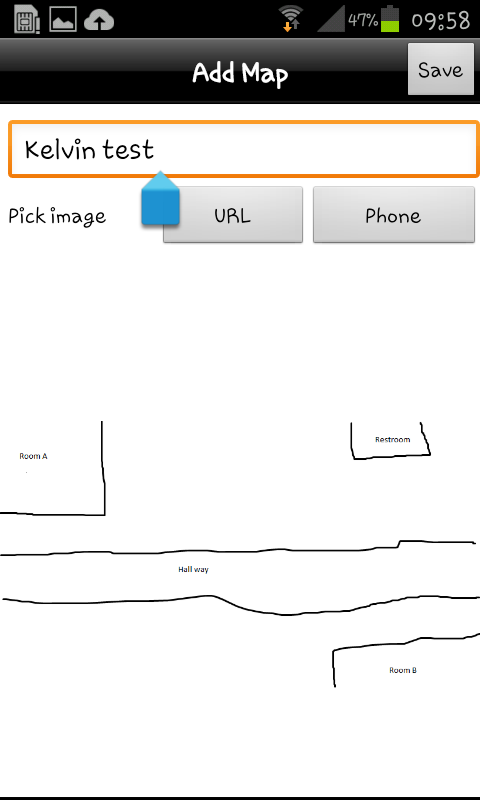


Figure - Uploaded map being display on device

The other errors we found is the GUI does not support zooming function, doing so WILL cause the application to crash.

Thus our release 1 will be:

(1) Update to android v4.3 Kit kat usable

(2) make sure the cores are working.

#### Stories Cont. Functional

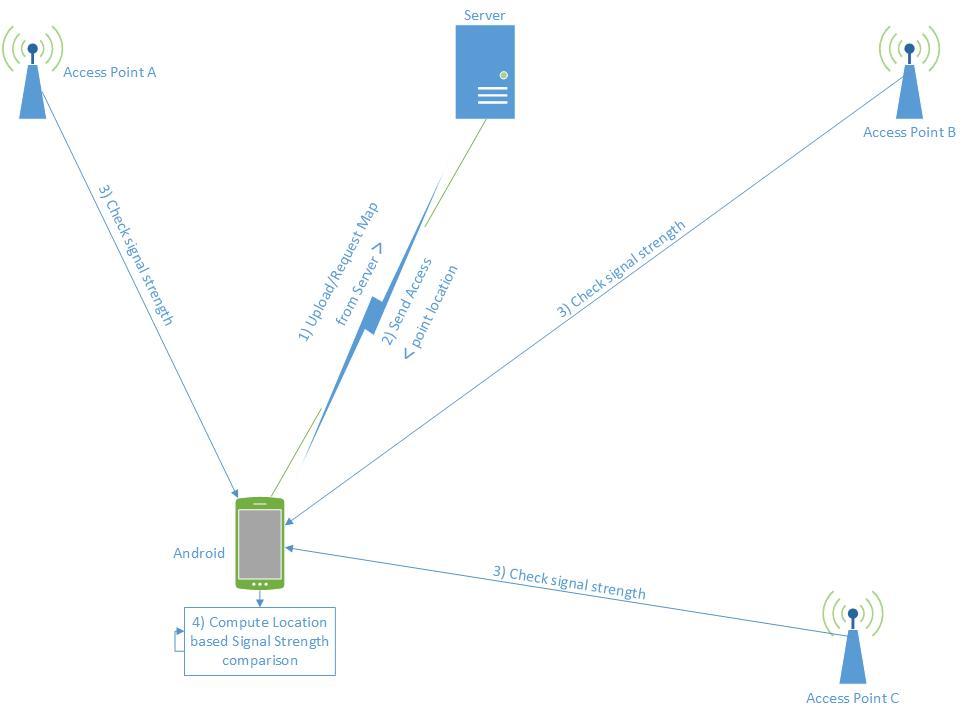
* Create documentation so that the next group of users

can use the software

* Send the data(location), positioning request to the server
* The positioning can be reached by any android app using get request
* Choose between different maps
* Set the map pin to the location send by the server
* GET and PUSH messages show up in the server window whenever we do an action on the devices

# Spikes

Spikes are tests and for the start of this project, the spikes will be just that. The first thing we had to do was some research. In order to do this project we had to have a little background in android and indoor positioning systems.

* determine if there were any other products or programs that would work like Redpin and if they would be a better fit for our project.
* look at what version of android Redpin was at and what the newest version of android was and how difficult it would be to update
* we had to determine how indoor positioning systems worked, and particularly how Redpin works

We also had to test in order to see if Redpin works in its current state, since the project must work to be upgraded. Test if the server communicates with the program on an Android device, which versions will and will not work. Which PC operating systems will work; then come spikes of the actual project working together.

If the original Redpin works in its current state, the project could then proceed to spike developing phase. Currently, these are the core spikes that are needed:

#### Phase 1

* What version can Redpin run on?
  + Target version: 4.1 and above
* What language is used to program the interface?
  + XML
* Is the interface supported on all devices that runs on android?
  + EG: Smartphone, Tablets
* How to create the GUI?
  + Target: Java Canvas
* What will the GUI look like?
* Is it possible to convert an image to a usable map?
* What type of image file format are supported?
  + EG: png, jpeg, bitmap, gif
* What are the minimum/maximum size image supported?
* How to validate the image file?
* How to obtain data from wifi access point?
* Is it possible to uniquely identify access point with MAC address?

#### Phase 2

* What OS could the server be run on?
* Can the device communicate with the server?
* What information are being sent/received?
* What information should be stored on the server?
* What information should be stored in the device?
* Is it possible to upload an image to the server from the device?
* Can the server convert uploaded image to usable map?
* What happens if a map already exist?
* How to validate the maps?
* Can the device compute its location?
* How precise are the computations?

After all these spikes has been created and all of them are tested to be working independently, The created standalone spikes will then be implemented as a whole. Integration problem will occur but it will not take too long to be solved since all individual spikes are already tested to be working as long as there’s not many changes other than variable refactoring.

# Legal

### Licenses

This project is [free software](http://en.wikipedia.org/wiki/Free_software); it can be redistribute and/or modify it under the terms of the [GNU General Public License](http://en.wikipedia.org/wiki/GNU_General_Public_License)Version 2 as published by the [Free Software Foundation](http://en.wikipedia.org/wiki/Free_Software_Foundation). This work is distributed in the hope that it will be useful, but without any warranty; without even the implied warranty of merchantability or fitness for a particular purpose.

### 

### Intellectual Property (IP)

This project is being developed as a Senior Design project for Western Michigan University (WMU) under the supervision of Dr. John Kapenga, WMU will retain the intellectual rights to the software.

### Non-Disclosure Agreement (NDA)

No non-disclosure agreement is being used at this time. The project is maintained on GitHub, which is freely and openly accessible to anyone who wishes to view it, and is thus tracked by search engines such as Google, where it is able to be searched for by anyone on the planet.

# Resources

For this project most of the resources are things most people use every day. Software that we will need will be free and some open source. Any other resource we may need will be discussed with the client to decide who will pay. The resources we need are simple and few.

* A device running an Android Operating system greater than version 2.2. Preferably a tablet that can be upgraded to Android version 4.3.
* Redpin source code
* Android and java SDK
* Android developer tools
* A computer with the Redpin Program and server running to act as a server.
* Digital maps of anywhere we will be using the program.

Possibly a Wi-Fi access point or extender to test.

# Feasibility

Based on the resource that we have, (All needed source code has been obtained and update has been undergoing with weekly TPS report showing good progress) and the given period, (release 1 has to be completed by the end of Spring semester 2014), getting release 1 (scope of release 1 is to update to android 4.3 usable and make sure CORE features such as uploading map, location tracking are working) done by the semester are considered to be a feasible goal.

Next, we had a meeting with a member from the Computer Engineering senior design team where he helped us and show us how everything works with this indoor position system and also the limitations they face.

The biggest problem right now we will be facing is that we will need to learn how to adapt to the android platform. Our team are sure to be able to overcome this since they are so much tutorials around the websites. This kind of project is clearly an engineer project: to setup a wifi based positioning system

indoor.

Even if we did not managed to make everything by ourselves (Concept, steps, virtual machine, setup of the API internally) this project required us to understand each step, and finally create this positioning system. This project was very interesting because of it’s complexity to make everything works together.

With this project we were allowed to see each part of a basic full Wi-fi based positioning system in details and to handle and monitoring every communications.

This project was also interesting because we used very different technology for each part of the system improving our skills in these technologies.

It was also a good team project because of it’s ability to be split into parts for each member of the team. It allowed us to work in parallel and compare our results, trying to merge all we have done in the separated parts.

# References

<http://www.gnu.org/licenses/licenses.html>

<http://www.redpin.org>

<http://developer.android.com/tools/sdk/eclipse-adt.html>

<http://www.vs.inf.ethz.ch/publ/papers/bolligph-redpin2008.pdf>

[www.meridianapps.com](http://www.meridianapps.com)

[www.wifarer.com](http://www.wifarer.com)

# Glossary

**ADT** – Android Development Tools, an Eclipse plugin that helps develop for the Android

Operating System.

**GPL 2** – GNU General Public License version 2

**CEAS** - College of Engineering and Applied Sciences at Western Michigan University

**GUI** - Graphical User Interface

**RSSI** - Received signal strength indication

**RSS** - Received Signal Strength

# API - Application programming interface

**ADT** - Android Development Tools