

Platinum Analytics

Final Year Project Report

DT228

BSc in Computer Science

**Peter Collins**

**<Supervisor>**

School of Computing

Dublin Institute of Technology

**<Date>**



Abstract

### Platinum Analytics is a sports data analytics system which gains insight into player performance by means of collecting and analysing player metrics with the aid of wearable sensor technologies.

Declaration

I hereby declare that the work described in this dissertation is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.

Signed:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

<Student Name>

<Date>

Acknowledgements

Body text

Table of Contents

*(These are suggested contents only. You can add or modify if required)*

1. Introduction
   1. Overview of the project and the background behind it.
   2. Project Objectives
   3. Project Challenges
   4. Structure of the document
2. Research
   1. Eg. Research related to identifying the problem that this project solves, research into solution definition

Algorithm

Current Solutions

My program is an opensource solution which provides a sports analytics framework based on non-proprietary hardware and that is what sets it apart from competitors.

Tech Research

* 1. An overview of the technologies evaluated and selected or rejected and the rationale behind the key decisions.

1. Design
   1. Identification of a design methodology including why it was chosen
   2. Design of each of the project components eg: the UI, Network, Project Demonstration, source code layout
   3. Clearly identifying the list of features and use cases supported within the project.
2. Architecture & Development
   1. Overview of the system architecture and a diagram to represent all of the key elements within the architecture.
   2. Details of each component within the project, problems encountered and resolved, challenges overcome or worked around.
   3. Identify key development components;
   4. Identification/explanation of external APIs used versus own code ; List of classes of your code etc .
3. System Validation
   1. Testing
      1. What testing was performed, why it was selected and what are the key use cases within the project.
   2. Demonstration
      1. Identify what features can be demonstrated and show screen shots or reference a video online to show the project demonstration (for audience not at demo)
4. Project Plan
   1. Project Plan analysis and review of how it changed from the initial proposal including explanation of what changed and why, and suggestions on how to address this if the project was repeated.
5. Conclusion
   1. Analysis of the projects key elements identify the key learning obtained from the project and recommendations and suggestions for how the work can be improved on continued into the future.
6. Bibliography
7. Appendix

## 1. Introduction

Platinum analytics is a sports data analytics system which aims to gain insight into player performance by means of collecting player metrics with the aid of wearable sensor technologies.

### Overview

The aim of this project is to build a system to record knowledge about players over the course of a match. The main piece of information to be recorded by this system and arguably the most vital, is the position of the players at any moment during the match. Recording and displaying this knowledge to a coach will allow him to review the events which led up to key moments of a game. This can then aid the coach in deciding tactics and position choices for future matches.

### Project Objectives

The objective of this project is to build a system capable of tracking players across a pitch for the duration of a match measuring at all times their position with an aim to provide feedback to coaches about their players exact positions in key moments of the game.

The positioning system to be employed will be built using the bluetooth wireless communications technology. A player is given a bluetooth sensor, or node, which emits a signal that is picked up by bluetooth radio receivers placed around the side of the pitch. These receivers in turn calculate and store the distance measured between themselves and the node based on the strength of the signal received. Once the distance has been measured between at least three receivers and a node for any point in time the position of that node can derived. T

### Project Challenges

The challenges faced in this project come in both hardware and software format.

On the hardware side, the challenge will be to find a suitable solution for tracking players on a pitch. On the software side it will be to display the gathered data in a meaningful way to the system's users.

### Structure of this Document

In this document I will outline in detail the steps taken to build the systems used to gain and, store and analyse the data used to measure player performance. I will also show the outcomes of the final system implementation discussing the expected vs actual results and findings. I will also outline the technological challenges faced and the solutions provided to overcome those challenges.

## 2. Research

1. Research
   1. Eg. Research related to identifying the problem that this project solves, research into solution definition

Algorithm

Current Solutions

My program is an opensource solution which provides a sports analytics framework based on non-proprietary hardware and that is what sets it apart from competitors.

Tech Research

* 1. An overview of the technologies evaluated and selected or rejected and the rationale behind the key decisions.

### Introduction

#### Trilateration Algorithm Research

#### Current Alternative Solutions to Sports Analytics

##### Fitbit

Statsports

### Technologies Evaluated, Selected and Rejected

#### Bluetooth Sensor

The sensor worn by the player is a small battery powered computer consisting of a heat sensor for measuring temperature, a light sensor for measuring brightness, an accelerometer for measuring movement. A bluetooth radio transmitter on the sensor transfers this information back to multiple receivers placed in exact locations on the side of the pitch. With this system in place the exact position of the player can be calculated at any point during the match.

The sensor chosen for this project is the TI Electronics SensorTag.

#### Bluetooth Receivers

Initially the aim was to use mobile phones as receivers. This changed to Raspberry Pis during the course of the project

In this section I will give an overview to the technologies which were evaluated, which ones were selected and which ones were rejected along with the reasons for each.

One of the main technologies used from the very beginning of this project was Git. Git is a version control system, meaning that it is a place for managing all software written. It records all changes made in the history of the project which allows user to return to any snapshot in time for the course of the project.

#### Version Control System

Git is the Version Control System for managing software in this project. As mentioned above, Git is a tool which is used for managing all written software in the development and release stages of the project. Github is the provider of the Git repository made accessible via the web. (1)

In the following paragraphs I will go into detail about how Git was employed in this project.

###### Branches

**master** - This branch only contains finished releases. The code should always be in a production ready state, meaning that it is ready to be deployed with no more development or unit testing required.

**develop** - This branch is the branch with the latest delivered changes which are ready for release. This can also be known as a continuous integration branch in which multiple new features can be tested for compatibility with each other.

**feature** - There are feature branches for each different deliverable within the scope of the project.

**release** - This branch is created off the develop branch and merged back into master branch. Release branches represent all code from developed features which are tested and complete and ready for deployment. Release branches should be named *release* followed by the version number.

This branching scheme has been inspired by 'A Successful Git Branching Model by Vincent Driessen(2).

###### Initialize a directory

Initialize a git repo in the current directory and fetch the platinum-analytics codebase.

git init

git remote add origin [git@github.com](mailto:git@github.com):peteehb/platinum-analytics.git

git fetch

git pull origin develop

###### Committing

A commit command is issued when a developer wants to store any changes made. The developer adds the changed files to the commit and finishes with a message to describe those changes. A push command is used to sync the local branch with the remote branch on github. Before issuing a push request, a pull request is issued which retrieves any changes which may have been made by other users to the same branch.

git pull

git add .

git commit -am "Commit Message"

git push

###### Branching

Branching in GIT allows the developer to test new features without changing the develop branch. The developer has no commitment to add the changes should they fail to meet the requirement of the feature. Any successfully developed and tested feature can be merged back into the develop branch.

git branch -b new-feature develop

###### Merging

When development is complete on a branch the changes are merged back into the develop branch by either opening a pull request or merging the branch with the merge command.

git checkout develop

git pull --no-ff

git merge new-feature

git push

When working in a team it is usually better to create pull requests instead of merging feature branches. The main advantage of using pull requests is that they give a way for other team members and managers to review code before it is accepted into the main branch. Github shows every change made to files, displaying updates and also changes which will cause conflicts.

###### Conflicts

Conflicts are collisions which arise "when two branches change the same part of the same file and then those two branches are merged." [(3)] To resolve conflicts Git displays all conflicts in code by marking the duplicate lines and the changes between them. The developer then chooses which of the duplicate lines are correct and deletes the others.

###### Release a Version

When all of the development in the current scope of the project is completed a version can be released. A version branch is created off the master branch with a given version number. This branch is then tagged, which creates the release.

git checkout -b release-1.2 develop

git checkout master

git merge --no-ff release-1.2

git tag -a v1.2

git checkout develop

git develop

git merge --no-ff release 1.2

git push

##### SSH

Ssh was used to run commands on the raspberry pis. It allows remote login to the rpi, returning a console in which to run commands.

By placing my laptop's public into */home/pi/.ssh/authorized\_keys* I am able to login to the pi without entering a password. This process is repeated with the pi's keys placed on the db server. This facilitates the usage of scp to sync local files onto the db server, which also requires no input from the user.

SSH is also used to access the application and database servers to run commands to initialize these components.

##### VIM

Vim is a text editor found on almost all Linux systems. It is used to edit text files such as configurations on the Raspberry Pis, APP and DB servers.

##### CRON JOBS

A cron job is used to launch a python script which starts the bluetooth monitoring on the Raspberry Pis. This script is run on startup so the user does not need to interact with the pi at all once it has been powered on.

sudo crontab -e

@reboot /opt/platinum/mon/startup.sh

@reboot /opt/platinum/mon/syncfiles.sh

###### startrup.sh

python StartProgram.py 1 False

Another cron job is scheduled to run a python script to check if an internet connection has come available. If it has detected a wifi signal of a known network and established a connection to that network, the python script copies all local SensorReadings csv files to the application server.

###### syncfiles.sh

python SyncLocalFiles.py

##### Raspberry PI

### Technologies Rejected

##### Android

Android was initially chosen to be the technology used for building the receivers with. It is the name of the Operating System which is run on a large percentage of mobile phones today. Android applications are built using the Java Language.

By using android smart phones I would be able to take advantage of their inbuilt features.

###### Why Android was Rejected

Prior to the beginning of this project I had little experience programming for Android. There came a critical point in the development phase where I realised that I would not be able to achieve the aims of the project using these tools. Also it was the only element in the project so which was not written in Python. Both PNM-Mon and PNM-DB are Django Web Applications which are coded in Python. I felt that not only would I be streamlining the continuity between the different applications but that I would also be reducing the level of project risk and the delivery of a working system.

### Conclusion

## 3. Project Design and Methodology

### Introduction

In this section I will give an overview of the software development methodology which was used and why it was chosen. I will also discuss in detail the design of each of the system's components, listing each components features and which use cases they provide for.

### Software Methodology

#### Introduction

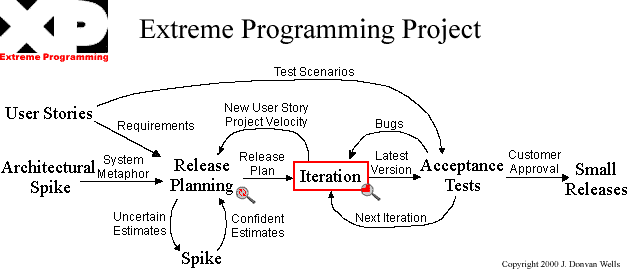
Discuss software methodologies. Find sources

Agile

#### Identification of Methodology

Discuss xtreme programming

For this project I used an agile methodology. The Agile Process used was Extreme Programming. According to this definition from extremeprogramming.org, Extreme Programming "is a lot like a jigsaw puzzle. Individually the pieces make no sense, but when combined together a complete picture can be seen."



Xtreme facilitates the rapid prototyping of software which

### Use Cases

### System Components

In this section I will give a detailed system overview and presentation of each component in the project and detail how the components communicated with each other within the system.

##### How does the positioning work.

The three or more bluetooth monitors are placed in exact locations around the pitch. It is of utmost importance that these monitors be placed in the correct location around it so that the triliteration algorithm may accurately calculate the player's position.

The location of the for each of the monitors, or receivers is as follows:

Monitor

|  |  |  |
| --- | --- | --- |
| ID | Mandatory | Position |
| Monitor 1 | Yes | Halfway line at side of pitch |
| Monitor 2 | Yes | Halfway line at far side of pitch |
| Monitor 3 | Yes | Behind the endline haldway across the pitch |
| Monitor 4 | No | Behind the endline halfway across the pitch at the other end. |
|  |  |  |
|  |  |  |
|  |  |  |

As stated above, Monitors 1, 2 and 3 are all mandatory for the Triliteration Algorithm. The Algorithm requires there to be a minimum of three receivers measuring the distance between themselves and a sensor at a single point in time in order to calculate a position of where the player is on the pitch. Monitor 4 is not required but it would help improve the accuracy of sensor positioning readings at the farthest points on the pitch from Monitor 3.

#### Bluetooth Wearable Sensor Sensortag

*Sensortag* is the name of the chosen sensor which players will wear while being monitored by the Platinum Analytics program.

#### Bluetooth Monitor PNM-MON

*pnm-mon* is a Python application deployed on the three Raspberry Pi computers used for collecting the information from

Android

Prior to the beginning of this project I had little experience programming for Android.

#### Web Frontend PNM-WEB

*pnm-web* is the name given to the frontend web application which provides the web interface for users to interact with Platinum Analytics.

#### Database PNM-DB

*pnm-db* is a postgres backed SQL database server which is used for the persistent storage of all of the program's data. A REST API sits in front of the database which serves the program's data to users through *pnm-web* over the HTTP protocol.

### Conclusion

In this section I gave an overview of the software methodology which was employed in the project and why it was chosen, a list of the use cases which outline what functionality was required at a high level and an overview of the projects components.

1. Architecture & Development
   1. Overview of the system architecture and a diagram to represent all of the key elements within the architecture.
   2. Details of each component within the project, problems encountered and resolved, challenges overcome or worked around.
   3. Identify key development components;
   4. Identification/explanation of external APIs used versus own code ; List of classes of your code etc .

## 4. Architecture and Development

### Introduction

In this section I will give an overview of the system architecture representing all of the key elements which make up the project's architecture. I will also detail each of the components in the project

### System Architecture

Platinum Analytics is a multi-tiered application consisting of several hardware components. In this section I will detail the initialization, development and deployment of each component as well as giving an overview of how the system components interact with each other.

#### PNM-Mon - Architecture Setup & Deployment

*PNM-Mon* is the name given to the Python based Bluetooth monitoring program responsible for collecting data from the sensors worn by the players which runs on Raspberry Pi Linux computers. In all there are three Raspberry Pi computers deployed in this project, the minimum number required to perform the trilateration algorithm detailed previously in Chapter 2.

The setup of the Bluetooth monitoring component required the acquiring three Raspberry Pi computers. Each Raspberry Pi in turn required a Bluetooth radio, a Wi-Fi radio, a battery pack and a memory card to run the operating system off of.

Two types of Raspberry Pi were used in this project, one Model 2-B and two Model 3-Bs. The Model 2-B does not have an onboard Bluetooth or Wi-Fi radio and so USB alternatives were used instead. The Model 3-B does have onboard Bluetooth and Wi-Fi radios and so no additional hardware was required for these computers.

A battery pack was required for each Raspberry Pi so that they could be deployed around a sports pitch away from any source of mains power.

The memory card required for each Raspberry Pi allowed for the installation of the Raspbian Gnu-Linux operating system. Raspbian is a stripped down Debian distro which is tailored for the Raspberry Pi computer. Raspbian also comes with Python 2.7 installed by default, reducing the amount of additional software required to be installed on the system.

To configure the system on first boot each Pi was connected via a wired internet connection. Before we are able to run any commands the Pi we must find out the Pi's IP address and connect to it over SSH.

# Search for the Raspberry Pi's IP address

# Attempt to connect to each IP with the following command until the right # address is found

ssh pi@192.168.0.14

# Once the correct IP is found we can enter the password and connect to the # pi

Please enter password for user pi@192.168.0.14: \*\*\*\*\*\*\*

Once we have found the right IP address and entered the correct password for the *pi* user we gain access to the Pi. From here we are presented with a terminal interface on which to run commands on the Pi.

With an SSH connection to the Pi established, the following commands were run on the commandline which enable the Wi-Fi and Bluetooth radios and also install the software libraries required to run *PNM-Mon*.

# Enable the wireless and bluetooth radios

sudo iwconfig wlan0 up

sudo hciconfig hci0 up

# Add wireless network connection details to wireless client conf file

sudo vim /etc/wpa\_supplicant.conf

# Add the following lines to the file and save

network = {

ssid="wireless-network-name"

psk="wireless-access-code"

}

# With the network address added attempt to connect

sudo dhclient

# Install bluetooth monitoring software

sudo apt-get install bluez -y

The next step is to download the *PNM-Mon* software from github onto the Pi.

# Connect to git and download platinum-analytics repo

git init

git remote add origin [git@github.com](mailto:git@github.com):peteehb/platinum-analytics.git

git fetch

git checkout develop

With all of the software in place, the final step is to set up the scheduled tasks, also known as Cron jobs, which start *PNM-Mon* on boot and check for an internet connection. Further details on the implementation of the scripts which these scheduled tasks run can be found later in this chapter.

# Open the super users task scheduler

sudo crontab -e

# Add the following lines to the file.

@reboot python /opt/platinum/mon/StartProgram.py 1 False

\*/5 \* \* \* \* /opt/platinum/mon/syncfiles.sh

With the scheduled tasks added the Pi is now ready for deployment and requires no further input from the user, simply powering on the unit is all that is required to start *PNM-Mon*.

#### PNM-DB - Architecture Setup and Deployment

This section will outline the steps required to setup the Linux server with the software required to deploy *PNM-DB*. *PNM-DB* runs on a Debian Linux Server, subsequently all of the following commands are runnable on any Debian based derivate distros, such as Ubuntu or Mint.

##### Postgres

The first piece of software required for *PNM-DB* is Postgres. Postgres is an open source object-relational database system. (4) It is the database used to persistently retain and store all data collected by the *PNM-Mon* monitors. The following commands install Postgres and also create the database *'pnmdb'* on the system. We also add the user *'platinum'* for access to the database from external applications.

# Update software packages to latest versions and install postgres

sudo apt-get update

sudo apt-get install postgresql -y

# Once postgres has been installed change to the 'postgres' system user and # create the pnmdb database and platinum user

sudo su postgres

psql

create database pnmdb;

\q

##### Django-Rest-Framework

With postgres successfully installed we can now deploy the *PNM-DB Web API*. Django-Rest-Framework was the tool chosen with which to build the API. Django-Rest-Framework is ...

Django provides a way of creating and managing postgres database tables through Python with the use of an object-relational-mapper. Object Relational Mapping (ORM) is a technique that lets you query and manipulate data from a database using an object-oriented paradigm.(5) This means that we can represent our postgres tables and relationships as Python objects which we call 'Models'. This enables us to create our database tables without writing an SQL.

We can then expose these Python objects over HTTP by Serializing the Models to JSON objects and making them accessible through URLs.

With these features that django-rest-framework provides, we can now server our database data over HTTP. Create-Read-Update-Delete requests made to defined URLS can now alter data in the database.

To setup the *PNM-DB Web API* the following steps were run on the database server.

# Connect to git and download platinum-analytics repo

git init

git clone [git@github.com](mailto:git@github.com):peteehb/platinum-analytics.git

# Create a virtualenv, Install requirements and run pnm-db

mkvirtualenv pnm-db

pip install -r requirements.txt

cd pnmdb/

python manage.py syncdb

python manage.py makemigrations

python manage.py migrate

python manage.py runserver 0.0.0.0:8000

##### Deployment

The above command will only run *pnm-db* for as long as the user is on the server. This is not suitable for a production environment and so a more permanent method of deployment had to be found.

To overcome this issue an alternative method of deployment was required. The method decided upon was to run the django-rest-framework application persistently using UWSGI. This is achieved by installing a plugin for the preinstalled webserver Apache called *mod\_wsgi*. This plugin runs the django application in a process which it itself manages. This requires no input from the user once *mod\_wsgi* is installed and *PNM-DB* django application is properly configured.

# Install apache dev tools

sudo apt-get install apache-threaded-dev

# Get and install the mod\_wsgi plugin

wget https://github.com/GrahamDumpleton/mod\_wsgi/archive/4.4.21.tar.gz

tar xvfz mod\_wsgi\_4.4.21.tar.gz

./configure

make

sudo make install

# Load mod\_wsgi into Apache and enable

cd /etc/apache2/modes-available

vim mod\_wsgi.so

# Append the following line

LoadModule wsgi\_module /usr/lib/apache2/modules/mod\_wsgi.so

# Enable plugin

cd /etc/apache2/mods-enabled

sudo ln -s ../mods-available/mod\_wsgi.load

# Restart apache

sudo /etc/init.d/apache2 restart

One final step in deploying *pnm-db* is to add an entry to Apache's *sites-available* file. This is a file which tells the apache which websites and webapplications are running on the server. In this file we define the location of the wsgi application for *pnm-db*, along with its root directory and which port number it is accessed on. Once the file has been edited a final restart to the apache process will make *pnm-db* available and finish the deployment process.

# Tell Apache to listen to all connections made on port 8000.

Listen 8000

<VirtualHost \*:8000>

# Tell Apache the location of the wsgi application

WSGIScriptAlias / /opt/platinum/db/pnmdb/wsgi.py

# Create a process and run *pnm-db* in the defined virtualenv

WSGIDaemonProcess pnmdb python-path=/opt/platinum/db/pnmdb:/home/elpok/.virtualenvs/pnm-db/lib/python2.7/site-packages

WSGIProcessGroup pnmdb

Alias /static /opt/platinum/db/static

# Set permissions to allow access to *pnm-db* directory

<Directory /opt/platinum/db>

Options -Indexes

Order deny,allow

Allow from all

</Directory>

# Define log files for *pnm-db*

ErrorLog ${APACHE\_LOG\_DIR}/pnmdb\_error.log

CustomLog ${APACHE\_LOG\_DIR}/pnmdb\_access.log combined

</VirtualHost>

#### PNM-WEB Architecture and Deployment

This section will outline the steps required to setup and deploy *PNM-Web* to a Linux server. In this project, I deployed *PNM-Web* to the same Linux server which runs *PNM-DB*.

The steps to installing *PNM-Web* are similar to that of *PNM-DB* as they are both at their core Django applications.

# Connect to git and download platinum-analytics repo

git init

git clone [git@github.com](mailto:git@github.com):peteehb/platinum-analytics.git

# Create a virtualenv, Install requirements and run pnm-web

mkvirtualenv pnm-web

pip install -r requirements.txt

cd pnmweb/

python manage.py syncdb

python manage.py makemigrations

python manage.py migrate

python manage.py runserver 0.0.0.0:7999

As with *pnmdb*, running *pnmweb* with the above command will only keep the application running for as long as the user is logged in. Once again we need a more permanent solution for running the *pnmweb* application.

In order to deploy *pnmweb* persistently we must add a VirtualHost entry to Apache's *sites-available/default* configuration file.

# Define which port to listen to for pnmweb traffic

Listen 7999  
<VirtualHost \*:7999>

# Path to Django Applications wsgi file  
 WSGIScriptAlias / /opt/platinum/web/pnmweb/platinum/wsgi.py

# Start the wsgi daemon in the pnmweb virtual environment  
 WSGIDaemonProcess platinum python-path=/opt/platinum/web/pnmweb/platinum:/home/elpok/.virtualenvs/pnm-web/lib/python2.7/site-packages  
 WSGIProcessGroup platinum  
  
 Alias /static /opt/platinum/web/pnmweb/static

# State the path to the base directory of pnmweb  
 <Directory /opt/platinum/web>  
 Options -Indexes  
 Order deny,allow  
 Allow from all  
 </Directory>

# Define logs for pnmweb   
 ErrorLog ${APACHE\_LOG\_DIR}/pnmweb\_error.log  
 CustomLog ${APACHE\_LOG\_DIR}/pnmweb\_access.log combined  
  
</VirtualHost>

### Program Functionality

In this chapter I will walk throw the program flows which represent the implementation of the use cases listed in Chapter 3.

#### Flows

#### PNM-DB -Functionality

In this section I outline the Python Models which represent the database tables and the functionality which allows access to these objects through HTTP.

#### PNM-Mon - Functionality

This section will provide detail into the functionality and feature set of the *PNM-Mon* software, showing the modules used and discussing what each of the classes is used for.

The program is run with the following command by a scheduled job on startup of the Pi.

*python StartProgram.py 1 False*

The first argument given is the ID of the monitor. This ID is an integer between 1 - 3 and tells the system which monitor this is. It is important to know which ID the monitor is as the program expects the monitors to be placed in

Overview of the program flow

Overview of each class and what it is used for

Discuss the different modes, intermittent vs stable connection mode

### Program Flows

In this chapter I will detail all the system wide functionalities of the interconnected programs which make up the Platinum Analytics technology stack. I will detail how pnm-mon the Python Program running on the Raspberry Pi's detects, collects and stores readings from deployed sensors. How the triliteration algorithm calculates the position of the player based on the sensor readings. How the Pis sync their data to a persistent data store. How a user views the gained knowledge

##### Collect a Sensor Reading

**Steps**

1. Bluetooth Monitor reads output of '*sudo hcitool blescan --duplicates*'
2. Monitor searches for the line from output:

"> HCI Event:"

When the monitor reads this line instantiate a new Python Dictionary to store the reading with the keys "receiver", "address", "rssi", "distance" and "timestamp"

1. Monitor searches for line:

">Mac\_address: xx.xx.xx.xx.xx"

When this line is read in add the underlined text to the readingdictionary under the key "address".

1. Monitor searches for line:

">Rssi -00dbm"

Add the number recorded on this line to the key "rssi"

##### Convert the rssi value to distance and save it under "distance"

1. Record a timestamp and return the reading dictionary to the method which called it.

##### Store a Sensor Reading

##### Post a Sensor Reading

##### Sync Sensor Readings

Sensor readings of the players positions are stored on each of the *pnm-mon* monitors. After a match sensor readings can be synced to the database and stored persistently by connecting each of them to a known Wi-Fi network. Once the Pi is powered on within range of the preconfigured network in its configuration files it is able to sync local *SensorReading.csv* files. A script takes care of the sync automatically on boot or when an internet connection is established.

##### Perform Triliteration

##### Generate Heatmaps

##### Display knowledge gained

#### StartProgram.py

**class** **StartProgram(**object**):**

#### BluetoothMonitor.py

**class** **BluetoothMonitor(**object**):**

#### Process Manager.py

**class** **Process(**object**):**

#### DataWriter Base Class

**class** **DataWriter:**

#### CsvDataWriter

**class** **CsvDataWriter(**DataWriter**):**

#### DatabaseDataWriter

**class** **DatabaseDataWriter(**DataWriter**):**

### Sync Script

##### Syncing Sensor Readings to database

### Conclusion

## 5. System Validation

### Introduction

In this section I detail the testing and validation done to the system to verify that the features developed provide for all of the use cases stated above in Chapter 3.

### Conclusion