Plymouth University

School of Computing, Electronics and Mathematics

PRCO304
Final Stage Computing Project
2017/2018

BSc (Hons) Computing

Matthew Rayner

10245732

Positional Home Automation

Acknowledgements

I would like to thank my project supervisor, David Lancaster, for all the help he provided both for the technical side and also academic side.

I would also like to thank Sam Delivett my study skill support specialist for help with the wordier side of this report and the project in general.

Lastly, I would like to thank my friends and peers who have provided all sorts of helpful advice, input and feedback throughout this project.

Abstract

This report describes the computing project that I undertook in order to achieve the goal of being able to automate certain processes and devices controlled by the location of a user.

The report begins with some background into why I chose this topic, then goes into how I approached the task and what tools I used.

The next part goes into the design of the system both from a domain standpoint and from a code architecture view.

Next is the development stage which is broken down into each separate deliverable, after that I talk about how I tested the system both as it was under development and after.

Finally comes a summary section including an end project report, post-mortem and an overall conclusion.

There is an appendix section at the end which contains some project documentation such as a brief user guide, the weekly highlight reports and the Project Initiation Document.

Table of Contents

Word Count

Word Count: ≈5300.

Code URL

Code URL: https://liveplymouthac-my.sharepoint.com/:f:/g/personal/matthew_rayner_students_plymouth_ac_uk/EhYSDW9mL_xIkVUYJDQgvI0B35xiJYVKgHpEWN5vz9zWdg?e=wa9f0X

1. Introduction

1.1. Background

With the recent rise of interest in interest home automation thanks to the falling price and supporting devices, such as the Google Home and the Amazon Echo, I wanted to create a system that allows for even easier control than using your voice. I wanted to be able to control devices based on location with enough accuracy to control devices just by walking between rooms. This would be for fans of home automation but also could be used as an assistive technology to help those with difficulties that may find that automating these simple processes can help with their overall quality of life.

Home automation is where one or more devices in a user's home can be controlled without direct operation from the user, usually via a network connection to other devices on the user's home network or to the outside internet.

Many devices can be smart home devices or home automation compatible such as:

- Lights
- Mains sockets
- Heating systems
- Security systems
- Heat, smoke or CO detectors

Google claims to have sold one Google Home device every second between October 2017 and January 2018 (Chandra & Huffman, 2018) so there is definitely a lot of interest in these sorts of devices.

The aims of the project were to create a system of positional tracking using low-cost devices or devices that a user would already own (such as a phone). These devices would interact with a user's existing home automation devices, such as smart lights.

While the code that I aimed to produce would only target a few devices (such as the Philips Hue smart lights) as these are the devices that I own and can test against, I aimed to build the code in a way that would allow support for other devices to be added easily.

1.2. Objectives & Deliverables

1.2.1. Objectives

The objectives that I had from my PID were:

1. Be able to measure the signal strength of Bluetooth devices to the base stations.
2. Allow users to create zones based on the current position of their phone.
3. Provide users with an interface where they can attach actions to events.

These are the very minimum of what an end user would expect from a usability stand point. These were only used as a starting point for designing and implementation, and these were broken down into many more sub-tasks for what the system should also do.

1.2.2. Deliverables

1.2.2.1. Bluetooth receiving module

The Bluetooth receiving module is a small light-weight application that measures the received signal strength indicator (RSSI) of each device and sends that data to a central distributor for other modules to read from.

It sends the name of the Bluetooth device, the id of the receiving device, the RSSI of the Bluetooth signal and the time that the signal was received.

There should be at least one of these running on the network, but the user could have many more as each one is individually identified. Having more devices connected to the system allows for a wider area to be covered or for more accurate positioning by overlapping the areas that the stations cover.

1.2.2.2. Control module

This module reads the distance data from the central distributor and uses a set of rules defined in a data store. Once a rule has been met the corresponding action should be performed. This is what would determine what room the user is in.
There should only ever be one of these running on the network.

Currently actions could be something as simple as printing a message to the standard output (stdout) or sending a web request to a specified URL in order to integrate with smart home devices either directly or via third part services such as IFTTT. (IFTTT, n.d.)

1.2.2.3. Set-up module

This module would be a web server that displays configuration data and allows users to easily create new zones and actions from their phone or other device with a web browser.

I decided that this should be a web-server as that allows the user to interact and use it easily from any device on their network without having to download any new software.

The reason this is a separate executable and is not just built into the control module is that the set-up module does not need to be running all the time. This means that after the system has been configured this module could be stopped which means that no one can come along and change the values used by the system.

Another reason I did not want to include the web-server into the control module is that it would require using a Java web server container which requires a lot more development time to set-up compared to a node.js express app.

1.2.2.4. Phone app module
This module is a simple phone app that allows the user to use their own phone as a Bluetooth Low Energy (BLE) beacon. This means they do not need to buy any additional hardware, although the system is designed to work with any Bluetooth Low Energy device.

The app would also include a web view that displays the setup web server allowing users the change the configuration of their system all from a single app rather than having to change out to a web browser.

2. Method of approach

2.1. Supervisor Sessions and Highlight Reports
The assignment required us to have weekly sessions with our project supervisors. This helped both from a technical aspect as my supervisor was very knowledgeable in areas around networking, which my project heavily relies on, but also helped just by having someone check up on the project every week, which helped me incentivise myself to keep steadily working on the project rather than leaving lots of work until the end.

This supervisor sessions were followed up with a highlight report that summarised the work done that week. These can be found in Appendix 14.2.

2.2. Agile

As I was developing this on my own rather than as part of a team and there was very little development time I decided to use an agile approach so that I could get started as soon as possible.

Due to the nature of the assignment and only being a team of one, none of the more formal agile methodologies fitted fully as they mostly revolve around managing a team of people. The way that I ended up working was probably closest to a Kanban style of working where the task was split up into issues and I completed them one or (occasionally) two at a time (Radigan, n.d.).

A big part of many agile methodologies (especially Kanban) is to have a visual “board” of the current state of the project and the state of all current tasks, whether that board be physical or digital. While I did try this at first, I found that it was not very helpful as I already knew what the state of all the tasks in the project was (as I was the one doing all of them). So, this did not get used a whole lot after the initial period of the project.

2.3. Languages/Platform

2.3.1. RabbitMQ
I decided to use RabbitMQ to send messages between all of the different modules that I have. Doing things like this allows each component to act much more independently than if they were directly sending messages to each other. It also means that the only IP address that needs to be known is the address of the RabbitMQ server.

2.3.2. Noble
Noble (Node.js Bluetooth Low Energy) is a simple node.js library that allows a node.js application to act as a Bluetooth LE central module (Bluetooth LE's version of a server). This then allows the reset of the application to access data such as the signal strength and name of the Bluetooth device.

2.3.3. Java
Java is the language that I am the most familiar with, therefore this is the language I used whenever possible. It is also the main language used when developing Android apps, so I did not have a lot of choices as I needed to create an Android app as part of the system.

For the control module I targeted Java version 1.7 to allow for the best compatibility possible while still containing the features I needed.

2.3.4. JavaScript/NodeJS
I decided to use JavaScript running on the NodeJS platform for the Bluetooth receiving module as it seems much easier than using Java, also I was familiar with the language after doing the client-side web scripting module last semester.

2.3.5. BASH
There were some tasks on the Raspberry Pi that I found myself having to do repeatedly, when this happened I created a BASH script that would simplify the process.

2.3.6. Raspberry Pi
The Raspberry Pi is a small and cheap Linux based computer. I used it as it is a very well-known platform so there is plenty of support available to find for it on the internet.

Being a Linux OS, it is also very quick and easy to install any software I needed using the *apt-get* command.

2.3.7. GIT
I used GIT as my version control as it is what I have used in the past, both at work and for my own personal projects, and when using the tools built into modern IDEs is very fast and easy to work with, especially when working independently rather than as part of a team.

2.3.8. NPM
NPM is a JavaScript library manager for node.js applications allowing for easy additions of dependencies to applications.

I used this for the Bluetooth receiving module and also for the setup module.

2.3.9. Maven
I used maven for handling any dependencies that the control module needed.

2.3.10. JetBrains IntelliJ IDE
I used JetBrains' IntelliJ as my Integrated Development environment (IDE) not only because of the way that in can speed up programming by using the built-in code generation features but also down to me needing several of the tools that are built in, such as:

- an SSH client that I needed for connecting to and interacting with my Raspberry Pi devices.
- A client for testing RESTful web services that I used for testing the Philips Hue API.

- Built in support for version control using GIT.
- Full support for both Java and JavaScript projects.
- Easy integration of the maven builds and dependency management tool.

2.3.11. Philips Hue

The Philips Hue system is a line of smart light bulbs that connect to your home network. The two main reasons I used this as my main output of my system is that it has a very RESTful API that can be used to control the lights, but more importantly I already owned a set, so this did not add to the development costs.

3. Legal, social, ethical & professional issues

3.1. Data Protection & Privacy

Currently the only data that the system stores about a user is the name of their Bluetooth device that the user interacts with.
Any data used by the system is kept locally on the devices and is never sent over the internet.

3.2. Licencing

All dependencies used have licences that allow the reuse of their code usually, so long as the code is still attributed to their respective owners.

Licence types of dependencies of the Bluetooth receiver module.

Dependency	Licence type
amqp.node	MIT
Log.js	MIT
noble	MIT
node-cleanup	MIT
serial-number	ISC

Licence types of dependencies of the control module.

Dependency	Licence type
RabbitMQ amqp-client	MPL, GPL, ASL
JSON-java	JSON
JUnit	EPL

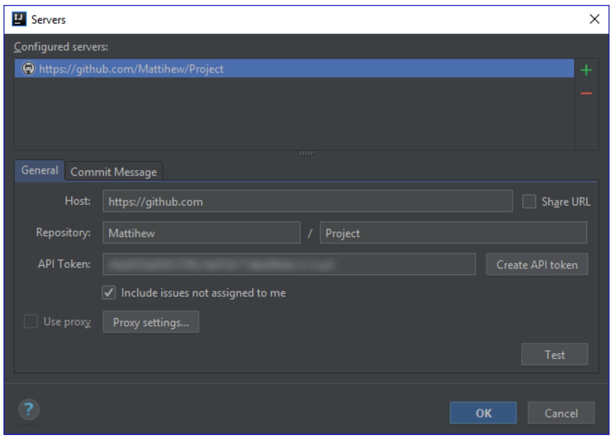
4. Project management

4.1. Code

4.1.1. GitHub

The code for this project eventually needs to be put onto my university OneDrive for final submission, however, while I was working on it I decided to keep it in a Git repository hosted on GitHub. Doing this provides several useful functions even when working on my own and not as part of a team, such as branches that allow me to work on some aspect of the code completely separated from the rest with a way to switch between different versions of the code quickly and easily.

I decided to use GitHub as the host of my repository because I have used it a lot before so am very familiar with using it and did not need to spend any time learning a new platform. Another reason for using GitHub is that due to it being one of the most popular online code repositories there is a lot of support for it integrated into programs, such as Integrated Development Environments (IDE).



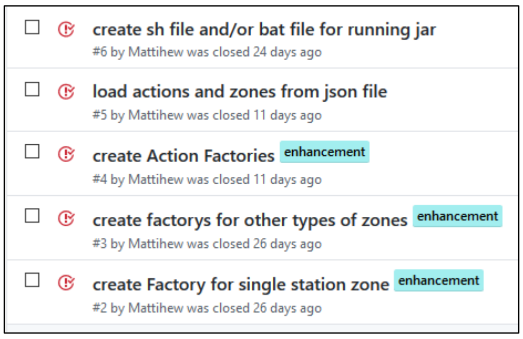
Configured servers:
https://github.com/Mattihew/Project

Host: https://github.com ☐ Share URL
Repository: Mattihew / Project
API Token: [redacted]
☑ Include issues not assigned to me
☐ Use proxy

4.2. Issues

As I was already using GitHub as my code repository it made sense to also use it as my issue tracker as that allows issues, commits and branches to all reference each other. I did not make much use of the issue tracker early on in the project as at the time the project was not very complicated so there were not any issues that could not be solved straight away.

As the project progressed and the codebase grew in size it became more important that issues were added to the list so that I could solve them at a later date as some issues required some major changes to the code in order to fix them.


create sh file and/or bat file for running jar
#6 by Mattihew was closed 24 days ago

load actions and zones from json file
#5 by Mattihew was closed 11 days ago

create Action Factories enhancement
#4 by Mattihew was closed 11 days ago

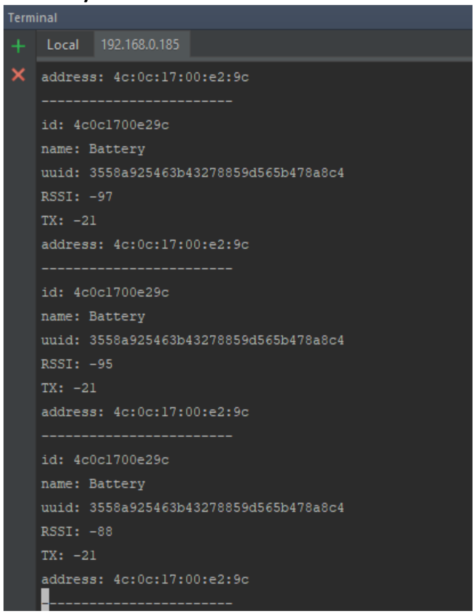
create factorys for other types of zones enhancement
#3 by Mattihew was closed 26 days ago

create Factory for single station zone enhancement
#2 by Mattihew was closed 26 days ago

5. Technologies

5.1. Feasibility Study

The goals of the feasibility study were to check that detecting signal strength between an Android phone and a Raspberry Pi using Bluetooth Low Energy would be possible. This was done by creating an Android app that used the Android API to advertise the phone as a peripheral and a NodeJS application that used the 'noble' library to read information about nearby Bluetooth devices.



```
address: 4c:0c:17:00:e2:9c
------------------------
id: 4c0c1700e29c
name: Battery
uuid: 3558a925463b43278859d565b478a8c4
RSSI: -97
TX: -21
address: 4c:0c:17:00:e2:9c
------------------------
id: 4c0c1700e29c
name: Battery
uuid: 3558a925463b43278859d565b478a8c4
RSSI: -95
TX: -21
address: 4c:0c:17:00:e2:9c
------------------------
id: 4c0c1700e29c
name: Battery
uuid: 3558a925463b43278859d565b478a8c4
RSSI: -88
TX: -21
address: 4c:0c:17:00:e2:9c
------------------------
```

6. Requirements

1. Be able to measure the signal strength of Bluetooth devices to the base stations.
2. Allow users to create zones based on the current position of their phone.
3. Provide users with an interface where they can attach actions to events.
4. Allow creating actions that control Philips Hue lights.
5. Allow creating simple zones with a minimum and maximum distance.

7. Design

7.1. System Architecture

At the very highest level the system is split up into the different modules explained in the deliverables section, these are the Bluetooth receiving module, the control module, the Android app, and the set-up module. These are all separate executables that would each use different methods of connecting to each other.

I decided to use a RabbitMQ message broker to send data between the different modules as I had used it previously while at work and also during the Distributed Systems (NET302) module last semester.

I decided to use the JSON data exchange format using UTF-8 encoding for all RabbitMQ messages as this can be easily encoded and decoded in both Java and JavaScript.

7.2. Domain Model

7.2.1. Edge

An edge represents a single Bluetooth signal strength measurement. It contains the name of the Bluetooth device, the name of the base station, the signal strength between them, and the time the measurement was made at.

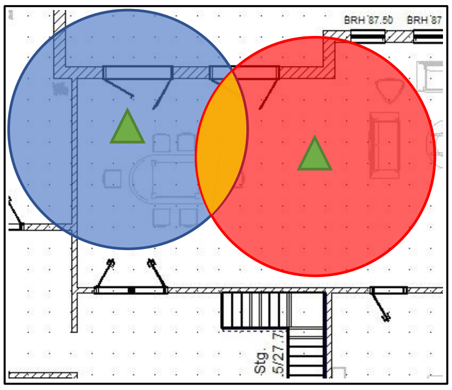
7.2.2. Point

Traditionally a 3D point is defined in relation to three perpendicular axes. However, in my system a point is defined by its distance from the base stations, this is fine as zones are also defined relative to the base stations.

If you wanted to be able to identify a unique position for a device, it would have to be in range of at least 4 base stations.

7.2.3. Zones

A zone is a simple concept where a zone is simply an area that a point can either be said to be inside or outside of. This is easier to explain with examples of the two main types of zones I have implemented.



There are single-station zones that have a minimum and maximum distance from a single station. This is represented by the blue or red areas in the image to the right.

There are also multi-station zones that are a combination of multiple single-station zones, this is represented by the yellow area in the image to the right.

7.2.4. Actions

Actions are the outputs of the system. Each action has a single direction and is only triggered when entering the assigned zone, by that I mean that, for example, you would have one action that turns lights on when you enter a zone in the room and then another separate action to turn them off when you enter the zone outside the room.

Currently only 3 types have been implemented but as an action only has a single *trigger()* method it is fairly easy to add new types.

7.3. Datastore Design

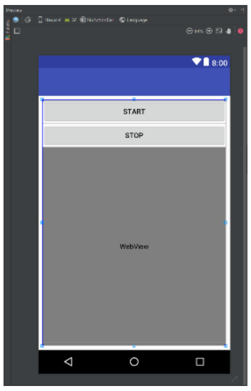
The current actions and zones are stored in a Java map that is keyed on the action and has the value of the zone that will trigger the action. The map's entry set can then be iterated over, and each zone can be evaluated, and the matching action can be triggered.

This would also allow for easy storing of this data in a database as there would be a table for zones, a table for actions and then an association table linking the two together.

7.4. UI Design

7.4.1. Android App

As the app is very simple I wanted the design to reflect that, so the main screen only has 2 buttons and a web view displaying a view of the set-up module server.



Changing other settings such as the polling rate and power of the Bluetooth transmitter is under a separate settings menu accessible by pressing the " ⋮ " symbol in the top right as is standard in android apps.

7.5. UML Diagrams

Some UML diagrams can be seen in appendix 14.4.

8. Development

8.1. Environment setup

Before I started any development work I made sure that my environment was setup. This included things such as creating and cloning a git repository from GitHub to my local machine and adding a *.gitignore* file that would ignore things like the compiled output and any IDE specific files.

Also, from a more physical standpoint I had to get some Raspberry Pi devices and get them connected to my home network so that I could send files to them from my desktop. Once they were connected and I could SSH into them, I could then start to install some of the software that I would need, such as Java and Node.

Once all this was setup I could then enable auto deployment in my IDE that used SFTP to send any changed file to the Raspberry Pi as the files were worked on.

8.2. Bluetooth Receiver module

I started development with the Bluetooth receiver module as this was one of the most important parts and the whole project revolved around getting this part working.

8.2.1. Noble

I started by installing Noble and its prerequisites, this was made very easy by following the instructions on their GitHub page (Mistry, 2018).
Once this had been installed I added the Noble library to my project and tested that it could pick up existing Bluetooth devices. I also tested that I could see the signal strength changing based on distance by installing an app from the Android app store that makes the phone act as a Bluetooth LE peripheral.

8.2.2. RabbitMQ

Once I had Noble working and I could get the signal strength of nearby Bluetooth devices. I then started working on the RabbitMQ transmission code. The first part of this was installing the RabbitMQ server onto one of the Raspberry Pi devices. This was a simple case of running the *apt-get* command and letting the Linux system handle any dependencies needed.

To check that the server was installed and running I used the built-in management web UI which is accessible by pointing a web browser at http://server-name:15672/ from here I could see the number of clients connected to the server and I could also add messages to specific exchanges or queue directly from the browser.

Now that this was all in place I could start on the RabbitMQ client, for this I used the Squaremo amqp.node library, this allows for simple implementations of the Advanced Messaging Queue Protocol (AMQP) which RabbitMQ implements. I tested that the messages were being sent to the server by viewing the management web UI where I could see that messages were being received and I could also read the contents of the messages to verify that the correct data was being sent.

8.2.3. Logging

While creating the module I had many points where values were being displayed using the *console.log()* method, I changed this to use a proper logging implementation so that I could easily toggle the output on or off as I would not want the data being displayed under normal usage.

8.3. Android App

Once I could read the signal strength of devices from the Bluetooth receiving module I could start making the Android phone app. This was a very simple app that at first would just have a button to switch the transmission of Bluetooth on or off. This was very simple and did not even require any third-party libraries as everything I needed it to do could be done just using the android SDK.

I added a settings page to the Android app to allow for configuring of settings, such as the polling rate and power of the Bluetooth transmission, as users may wish to change these as they will affect the power consumption of the app.

I did this also using Android's built in Preference class and supporting classes. This means that it will be very easy to add new settings in the future when I carry on the project.

8.4. Control Module

When I had completed the Bluetooth receiving module and the data was being sent to the RabbitMQ server I could then start on the control module that would read those messages from the RabbitMQ server, parse the data, and then make a decision on what (if any) action to perform, such as turning on a light or sending a web request.

I first created the part that would read messages from RabbitMQ, I used the official RabbitMQ Java client library that can be added as a maven dependency for easy dependency management. Once I had got the client code connecting to the server I could create a queue and attach it to the exchange I was using for communication between the control module and the Bluetooth receivers.

I then had to have a way to convert the string that I received from RabbitMQ into an object that I could pass around and do operations on it. I decided to use JSON as the method for formatting the message data as it is easy to create and read in Java and especially in JavaScript.

Once this data had been created as a Java object it could then be processed by the control module and be used to determine if the device that it represents has moved into any zones and should therefore trigger any actions.

8.5. Set-up module

I had created a separate NodeJS express application that would be a web-server that allows users to configure zones and actions. This would allow them to add new or modify existing zones and actions while the system is running.

I started by creating the method of exchanging data between the NodeJS server and the Java control module, this would make use of the existing RabbitMQ server, however, this time I would need to implement a request-response system so that the relevant data for the page that the user is on can be displayed.

To do this I made use of the remote procedure call pattern, this allows the NodeJS server to request a specific bit of information (such as the list of supported actions) and wait until the control module responds with the data. This works well with JavaScript callback functions.

This was, however, all I had time to implement so the desired functionality was not finished at the time of submission.

9. Testing

9.1. Manual Testing

The majority of the testing that I did while developing this project was manual due to the nature of the project relying heavily on external inputs and outputs. This was especially relevant for the Bluetooth receiving module as all it did was read the signal strength of the Bluetooth devices and send that to a RabbitMQ exchange. This meant that in order to test it I had to physically move a Bluetooth device closer and further away from the base station and inspect the result in the console.

9.2. Integration testing

While working on the control module the main form of testing was integration testing as I needed some form of input to test that the system would output the correct values. Rather than spending time creating a system for creating mock inputs I just used the already tested Bluetooth receiving module to send values to the control module.

9.3. Unit Testing

I found that creating unit tests was difficult for this project as it relies a lot on external inputs and external outputs. This meant that only small sections of the internal processing could be easily unit tested.

For the tests that I created I used the Java JUnit library as it integrates well with the IDE I was using, it also allows for Continuous Integration testing via services such as Travis-CI.

10. End-project report

10.1. Requirements overview

When comparing the final product to the initial requirements two main points can be made. The first being that the core functionality is complete, by that I mean that actions (such as controlling Philips hue lights) can be triggered by a user entering or leaving a room.

However, the requirements about providing the user with an "easy" interface for configuring this behaviour have not been met, as I ran out of time to work on the set-up module.

10.2. Project changes

While finishing up work on the control module's main functionality I realised that there would not be enough time to fully implement the set-up module, so I changed the control module to read from a JSON text file on start-up.

It was also planned to use a database for storing the zone and trigger data but that was also dropped after the set-up module got removed.

11. Project post-mortem

11.1. What went well

I am very happy with how the core functionality turned out, being able to walk around my house and having lights turn on and off as I move between rooms is very cool and it feels satisfying to see it working as I intended.

I am also happy with the way that the code works in that it should be very simple to add new types of actions very easily.

11.2. What didn't go well

The biggest disappointment was that I ran out of time and did not implement all the functionality that I wanted. This was primarily down to underestimating the time it would take to complete each section. This meant that I had to create workarounds, such as reading zone and action data from a file instead of using a database and a configuration web server.

11.3. Next Steps

I absolutely plan on continuing this project as I would love to have the original design with easier setup implemented for my own personal use.

There are a few practices that I have learnt while undertaking this project that I would like to try to increase my use of in the future, such as breaking the code down so that unit testing is easier. There are a few sections of my code that were not possible to test due to how highly coupled the processing code and the input/output code was.

12. Conclusions

The main goal of the project was to create a system that could control smart devices based on the position of the user in house using off the shelf hardware. That goal has been met, the system can read from a set of actions and their corresponding zones where it then waits for data to be received from the Bluetooth receiving modules to determine if the position lies within a zone.
However, currently usability is very poor as it is not very easy to change any of the zones or action data. This was down to underestimating the time taken for the tasks and overestimating how much I could get done in the limited time frame.

While overall, I am happy with the current state of the system, I do look forward to continuing to work on it after this is finished as I enjoyed working on it and would love to see it working with all the features that I planned from the start.

13. References

Chandra, R. & Huffman, S., 2018. *how google home and google assistant helped you get more done in 2017.* [Online]
Available at: https://www.blog.google/products/assistant/how-google-home-and-google-assistant-helped-you-get-more-done-in-2017/
[Accessed 18 04 2018].
IFTTT, n.d. *About IFTTT.* [Online]
Available at: https://ifttt.com/about
[Accessed 15 Feb 2018].
Mistry, S., 2018. *noble readme.* [Online]
Available at: https://github.com/noble/noble
Radigan, D., n.d. *What is Kanban.* [Online]
Available at: https://www.atlassian.com/agile/kanban
[Accessed 2 may 2018].

14. Appendix

14.1. User Guide

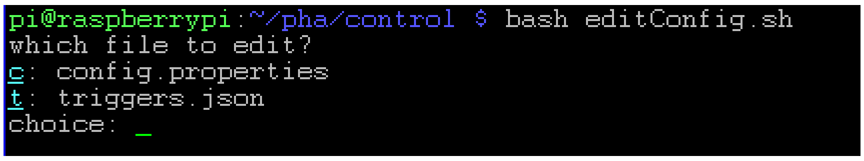
14.1.1. Installation

Currently the easiest method of installation is to use the packaged *.img* file that has been setup for raspberry pi 3 devices with all dependencies already included.
Other wise installation process will depend on the platform the code is being run on, the biggest thing this makes a difference for is the noble library as it requires its own custom Bluetooth device drivers, See the noble GitHub page for more details.

Once the system has been installed simply run the control module on one of the devices and the Bluetooth receiving module on all devices to start the system.

14.1.2. Configuration

To configure the control module run the *editConfig.sh* command located in the *~/pha/control/* folder.

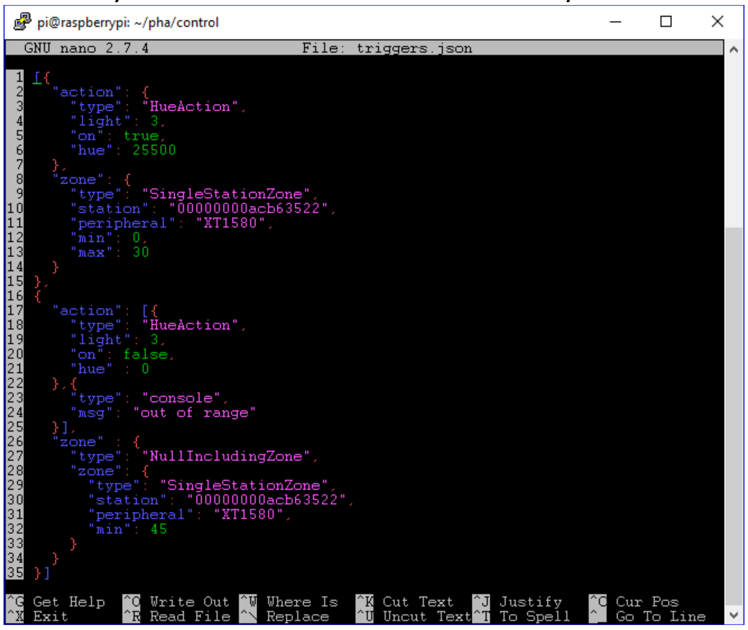


```
pi@raspberrypi:~/pha/control $ bash editConfig.sh
which file to edit?
c: config.properties
t: triggers.json
choice: _
```

This will prompt which of the files to edit.

The config.properties file contains settings such as the timeout time for edges in a point and what keywords to look for when parsing the triggers.json file. This generally would not need to be edited by a user.

The triggers.json file is the one that contains all action and zone information. Currently this is how a user would set the system of for their needs.



```
pi@raspberrypi: ~/pha/control
  GNU nano 2.7.4                     File: triggers.json

[{
  "action": {
    "type": "HueAction",
    "light": 3,
    "on": true,
    "hue": 25500
  },
  "zone": {
    "type": "SingleStationZone",
    "station": "00000000acb63522",
    "peripheral": "XT1580",
    "min": 0,
    "max": 30
  }
},
{
  "action": [{
    "type": "HueAction",
    "light": 3,
    "on": false,
    "hue" : 0
  },{
    "type": "console",
    "msg": "out of range"
  }],
  "zone" : {
    "type": "NullIncludingZone",
    "zone": {
      "type": "SingleStationZone",
      "station": "00000000acb63522",
      "peripheral": "XT1580",
      "min": 45
    }
  }
}]

^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit       ^R Read File  ^\ Replace    ^U Uncut Text ^T To Spell   ^_ Go To Line
```

At the very top level is an array of triggers from there you can use the tables below to add any action and map it to any zone.

Trigger

Attribute	*is Optional*	*Allowed values*
Action	false	Any action or array of actions
Zone	false	Any zone object

Hue Action

Attribute	*is Optional*	*Allowed values*
Type	false	“HueAction”
light	false	0-255
on	true	true, false
bri	true	0-255
hue	true	0-65536
sat	true	0-255

Console action

Attribute	*is Optional*	*Allowed values*
Type	false	“console”
msg	false	Any String

SingleStationZone

Attribute	*is Optional*	*Allowed values*
Type	false	“SingleStationZone”
station	false	Any string
peripheral	false	Any string
min	false	Any number
max	true	Any number

MultiStationZone

Attribute	*is Optional*	*Allowed values*
Type	false	“MultiStationZone”
method	false	AND, OR
zones	false	An array of zones

NullIncludingZone

Attribute	*is Optional*	*Allowed values*
Type	false	“NullIncludingZone”
zone	false	any zone

14.2. PID

PID

POSITIONAL HOME AUTOMATION

Matt Rayner | PRCO304 | 31/01/2018

Contents

Introduction

This project will be to create a simple and easy to use system that provides users with accurate positional tracking that they can then use for home automation.

It will use the Bluetooth signal strength from a Bluetooth low energy device (such as a user's phone) to each base station to calculate the distance each user is from each base station. This can then be triangulated to calculate the users position in the environment.

The processing is done by the base stations so that any Bluetooth LE device can be used rather than relying on a user's phone to provide some processing.

Business need

There is currently a lack of a system for accurately tracking a person's location within their house using their existing phone and some low-cost base stations.

This system will provide the software that a user can then use with there own hardware

These actions can then be set by the users to trigger set actions, such as turning on/off lights or other devices.

Project Objectives

1. Be able to measure the signal strength of Bluetooth devices to the base stations
2. Allow users to create zones based on the position of their phone.
3. To provide the user with an interface where they can attach actions to events

Initial Scope

The initial scope will not include vertical positioning and will focus on positioning users on a single floor.

The initial scope will only include a small set of actions such as controlling Philips hue smart lighting and other devices that can be controlled via IFTTT integration.

Resources and Dependencies

I already have a Raspberry pi 3 that includes both WIFI and Bluetooth, this can be used while developing the initial scope of my project.

My plan is to use up to three raspberry pi 3 devices, so I would need to source two more.

Method of Approach

I will use an incremental approach where the first step will be writing the code to calculate the distance from the user's phone to a single base station. The next step will be to provide the user with an interface for calibration of events and actions. The final step will be to expand the system to work with up to three base stations.

To calculate the distance between a user's phone and the base station I will use the Bluetooth signal strength between them.

I will use Node/JavaScript to capture the Bluetooth information. This information will then be sent across the network using a messaging system, such as RabbitMQ, to a central server running a Java application for processing.

A Phone app will be required to enable Bluetooth and to provide an interface for calibrating the system.

Initial Project Plan

Increment 1 will be to get a single base station to send Bluetooth signal strength data to a central server and for the central server to be able to switch lights on and off.

Increment 2 will be to create a web-server that users will be able to use to set-up and calibrate the system. These settings will have to be then saved to a database, so they can be used later.

Increment 3 will be to get multiple base stations talking to the central server and for it to make decisions based on the data received form all of them.

Stage	Expected Start Date	Expected Completion Date	Products/Deliverables/ Outcomes
1. Initiation	22/01/18	02/02/18	PID
2. Investigation and feasibility study	02/02/18	16/02/18	Design documents
3. Initial high-level design	02/02/18	16/02/18	Design documents
4. Increment 1	16/02/18	02/03/18	Increment requirements and design; sub-system

			sending Bluetooth range to central server.
5. Work-based learning module	26/02/18	16/03/18	
6. Increment 2	02/03/18	16/03/18	Increment requirements and design; sub-system providing users with a calibration interface
7. Increment 3	16/03/18	30/03/18	Increment requirements and design; sub-system utilising three different base stations.
8. System and user acceptance testing	30/03/18	13/04/18	Test Results
9. Assemble & complete final report	13/04/18	27/04/18	Final Report

CONTROL PLAN

To control the execution of the project I will have:

1. Weekly Highlight reports
2. Weekly Meetings with my project supervisor

COMMUNICATION PLAN

Weekly reviews will take place between me and my supervisor, excluding 2-3 weeks when I will be away completing my work-based learning module.

Initial Risk List

Risk Description	Probability	Severity	Impact	Management strategy	Contingency
Time lost due to Illness	Medium	Medium	Medium	Plan with contingencies for time lost	Reduce scope of project
Loss of completed work/code	low	high	Medium	I'll keep all code backed up on GitHub and all documentation on google drive.	Redo work and reduce scope.

Inaccurate estimations for project plan	medium	medium	Medium	Plan with tolerances for task that take longer	Reduce scope
Difficulty learning new technologies	low	medium	medium	Investigate how to use different technologies/libraries during design stage.	Investigate alternative technologies or reduce scope.
Difficulty implementing unfamiliar technologies	Medium	medium	medium	Investigate how to use different technologies/libraries during design stage.	Investigate alternative technologies or reduce scope.

Initial Quality Plan

Quality Check	**Strategy**
Requirements	In scope and SMART
Design	Requirements, relevant best design practices
Sub-system V&V (each sub-system based on design)	In scope, meets requirements, passes unit test, passes manual testing
System V&V (each system based on design)	In scope, meets requirements, each subsystem meets its quality check.

Legal, ethical, social and/or professional issues

Users may have an issue with where their tracking data is sent, that is why the system will be designed to work completely without an internet connection and will not store where the users are other than during calibration.

14.3. Highlight reports

PRCO304: Highlight Report
Name: Matt Rayner
Date: 08-02-2018
Review of work undertaken I have been creating a demo android app and node JavaScript script to check that reading the signal strength between two devices is possible. This has proven that this will be possible.
Plan of work for the next week Designing how all the different aspects of the system will be structured and communicate with each other, including creating any design documents. PID needs improving and re-submitting.
Date(s) of supervisory meeting(s) since last Highlight **07-02-2018** **31-01-2018**
Brief notes from supervisory meeting(s) since last Highlight PID still needs improvements. Better to focus on getting started with the product before starting on work-based learning module

PRCO304: Highlight Report
Name: Matt Rayner
Date: 14/02/2018
Review of work undertaken Bluetooth receiver and transmitter tested and working. Use of RabbitMQ proven to be effective as well. Feasibility written-up.

Plan of work for the next week Improve and resubmit PID after further input from supervisor. Start on first iteration of product.
Date(s) of supervisory meeting(s) since last Highlight: None
Brief notes from supervisory meeting(s) since last Highlight N/A

PRCO304: Highlight Report
Name: Matt Rayner
Date: 23/02/18
Review of work undertaken
Plan of work for the next week
Date(s) of supervisory meeting(s) since last Highlight
Brief notes from supervisory meeting(s) since last Highlight

PRCO304: Highlight Report
Name: Matt Rayner
Date: 01/03/2018
Review of work undertaken Very little work has been completed due to moving back home in order to work on the work based learning module for the next three weeks. I have started on the web server that will be used for configuring the system but only some basic html has been implemented so far.
Plan of work for the next week *Mostly just carry on with work based learning module.* *Also carry on working on config server and getting it to change the data stored in database.*
Date(s) of supervisory meeting(s) since last Highlight 01/03/2018
Brief notes from supervisory meeting(s) since last Highlight Little talked about as little has been worked on while completing work based learning module.

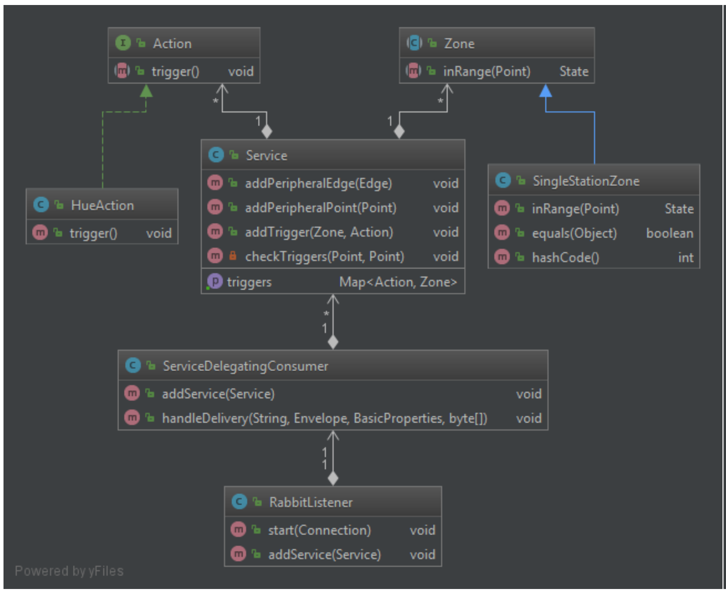
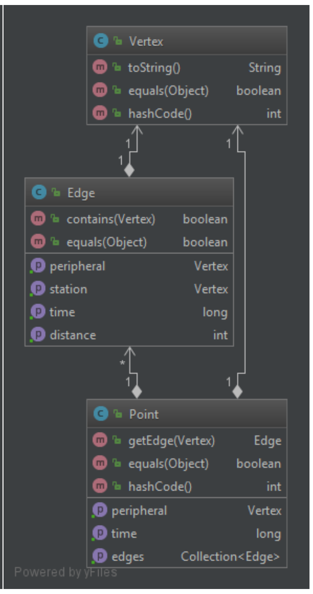
PRCO304: Highlight Report
Name: Matt Rayner
Date: 21/05/2018
Review of work undertaken A little progress made on getting database set-up and connected to java app for reading system config, but mostly carrying on working on work-based learning module.
Plan of work for the next week

Carrying on working on database and reading config from it. Not expecting much progress though.
Date(s) of supervisory meeting(s) since last Highlight 1/03/18
Brief notes from supervisory meeting(s) since last Highlight Little progress made so little was talked about. Agreed that better to focus on work-based learning module.

PRCO304: Highlight Report
Name: Matt Rayner
Date: 21/05/2018
Review of work undertaken Have added rabbitmq RPC communication between java control logic and nodejs webserver allowing the nodejs server to request data from the control server and display it to the user.
Plan of work for the next week Taking the data returned from the rabbit queue and displaying it in html pages
Date(s) of supervisory meeting(s) since last Highlight: none
Brief notes from supervisory meeting(s) since last Highlight N/A

PRCO304: Highlight Report
Name: Matt Rayner
Date: 21/05/2018
Review of work undertaken Bluetooth data is can now be displayed in a web browser. Minor refactoring of code to read from properties instead of hardcoded strings.
Plan of work for the next week Need to get two base stations working together to start creating more accurate zones. Also planning on gathering measurements such as max range and how environment affects the signal strength.
Date(s) of supervisory meeting(s) since last Highlight: 22/03/2018
Brief notes from supervisory meeting(s) since last Highlight Talked about how/where I will be demoing the product as I need more space than other projects. Also talked about what measurements need to be taken to justify my chosen solution.

14.4. UML


Action
trigger() void
Zone
inRange(Point) State
HueAction
trigger() void
Service
addPeripheralEdge(Edge) void
addPeripheralPoint(Point) void
addTrigger(Zone, Action) void
checkTriggers(Point, Point) void
triggers Map<Action, Zone>
SingleStationZone
inRange(Point) State
equals(Object) boolean
hashCode() int
ServiceDelegatingConsumer
addService(Service) void
handleDelivery(String, Envelope, BasicProperties, byte[]) void
RabbitListener
start(Connection) void
addService(Service) void
Powered by yFiles

Vertex
toString() String
equals(Object) boolean
hashCode() int
Edge
contains(Vertex) boolean
equals(Object) boolean
peripheral Vertex
station Vertex
time long
distance int
Point
getEdge(Vertex) Edge
equals(Object) boolean
hashCode() int
peripheral Vertex
time long
edges Collection<Edge>
Powered by yFiles