

RFID domestic location detection

by Simon Philipp Schäfer

BSc (Hons) Computer Systems

Staffordshire University

A project submitted in partial fulfilment of the award of the degree of BSc(Hons)

Computer Systems from Staffordshire University

Supervised by Carolin Bauer

April 2008

Number of words: \sim 22000



Preface and Acknowledgements

I want to thank everyone who was related to this project. At first there are my supervisors Carolin Bauer and Graham Mansfield who helped me a lot with their good suggestions about what I could do better and what went wrong. Then I want to thank the people from Bitmanufaktur (<http://www.bitmanufaktur.de/>) for their hardware and support. I also want to thank the people from Trolltech for their great framework and the qt-interest mailing list who helped me out when I got strange and stupid questions regarding the Qt framework.

I also want to thank the friends of mine who helped me with L^AT_EX and read over my report.

And last but not least I want to thank all the people who create Linux, KDE and all the great tools that I used for creating my project and sped up my progress.

Table of Contents

List of Figures	viii
------------------------	-------------

List of Tables	ix
-----------------------	-----------

1 Introduction	1
1.1 Background	1
1.2 Problem	1
1.3 Delimitation	1
1.4 Goal	2
2 Research	3
2.1 Methodologies	3
2.1.1 Waterfall life cycle	3
2.1.2 Rapid application development (RAD)	5
2.1.3 Extreme Programming (XP)	7
2.1.4 Code like hell and see what happens (CLHASWH)	14
2.1.5 Conclusion	14
2.2 Location detection basics	16
2.3 Mathematical fundamentals	17
2.3.1 One device	17
2.3.2 Two devices	18
2.3.3 Three devices	19
2.4 Technologies and their Hardware	20
2.4.1 WLAN	20
2.4.2 Bluetooth	21
2.4.3 RFID	22
2.4.4 ZigBee	22
2.4.5 Conclusion	23
2.4.6 Chosen Hardware	24
2.4.7 Analysis of the technology	28
2.4.8 Writing a prototype to access the Hardware	31
2.5 Operating systems	32
2.5.1 Microsoft Windows	32

2.5.2	Linux	32
2.5.3	Conclusion	33
2.6	Licences	33
2.6.1	Conclusion	34
2.7	Programming languages and libraries	34
2.7.1	Closer description of chosen language and libraries	35
2.8	Code Version Control Systems	37
2.8.1	Conclusion	38
2.9	Build environments	38
2.9.1	Plain makefile	38
2.9.2	SCons	39
2.9.3	CMake	39
2.9.4	Conclusion	39
2.10	Software Project Management	40
2.10.1	trac	40
2.10.2	dotproject	41
2.10.3	Conclusion	42
2.11	Databases	42
2.11.1	Oracle Database	42
2.11.2	Informix and DB2	42
2.11.3	MySQL	43
2.11.4	Conclusion	43
2.12	Web server	43
2.12.1	Apache	43
2.13	Inter Process Communication	44
2.13.1	D-Bus	44
2.13.2	DCOP	44
2.13.3	Socket	45
2.13.4	Conclusion	45
2.13.5	A detailed look on how D-Bus works	45
2.14	Similar projects	46
2.14.1	Location-based media	46
2.14.2	Magic Map	46
2.14.3	Conclusion	47
2.15	Project surroundings	47
3	Analysis	48
3.1	Research Conclusions	48
3.2	Project overview	49
3.3	General system test plan	50

4	Design	52
4.1	Modularisation	52
4.2	Firmware	53
4.2.1	Current functionalities	53
4.2.2	Needed functionalities	53
4.3	OpenBeacon Configurator	54
4.3.1	Functionalities	54
4.3.2	UML Class Diagram	54
4.4	Gain data Daemon	55
4.4.1	Functionalities	55
4.4.2	Incoming data	55
4.4.3	Outgoing data	55
4.4.4	UML Class Diagram	56
4.5	Measured data	56
4.5.1	Functionalities	56
4.5.2	Incoming data	56
4.5.3	Outgoing data	57
4.6	Optional application - Map generation application	57
4.6.1	Functionalities	57
4.6.2	Incoming data	57
4.6.3	Outgoing data	58
4.6.4	UML Class Diagram	58
4.7	Generate position daemon	58
4.7.1	Functionalities	58
4.7.2	Incoming data	59
4.7.3	Outgoing data	59
4.7.4	UML Class Diagram	59
4.8	Person data administration interface	59
4.8.1	Functionalities	60
4.8.2	Incoming data	60
4.8.3	Outgoing data	60
4.8.4	UML Class Diagram	60
4.9	Show position of a person	60
4.9.1	Functionalities	61
4.9.2	Optional functionalities	61
4.9.3	Incoming data	61
4.9.4	Outgoing data	61
4.9.5	UML Class Diagram	62
4.10	Optional application - Music follows you administration interface	62

4.10.1	Functionalities	62
4.10.2	Incoming data	63
4.10.3	Outgoing data	63
4.10.4	UML Class Diagram	63
4.11	Optional application - Music follows you	63
4.11.1	Functionalities	64
4.11.2	Incoming data	64
4.11.3	Outgoing data	64
4.11.4	UML Class Diagram	64
4.12	Common classes	64
4.12.1	OpenBeacon Node Communication Class	64
4.12.2	Data Exchange Class	65
4.12.3	Log Class	66
4.12.4	Optional - Map Classes	67
4.13	Network protocol for SSL connections	69
4.14	D-Bus Interface specification	70
4.14.1	gain data interface	70
5	Implementation	73
5.1	Installation and Configuration of the surroundings	73
5.1.1	Installation	73
5.1.2	Configuration	73
5.1.3	Adding the basic directory structure to the Subversion repository	76
5.1.4	Database and phpMyAdmin	77
5.2	Coding style	78
5.3	Milestones	80
5.3.1	OpenBeacon Configurator + helper classes	80
5.3.2	Firmware	80
5.3.3	Data gain daemon	82
5.3.4	Hardware Simulator	83
5.3.5	Generate position daemon	84
5.3.6	Administrate person data	86
5.3.7	Show position	87
6	Testing	88
6.1	OpenBeacon Configurator	88
6.1.1	Menus	88
6.1.2	Dialogs	89
6.1.3	Main Window	91
6.2	Firmware	92

6.3	Gain Data	93
6.4	Generate Position	95
6.5	Administrate Person Data	97
6.5.1	Menus	97
6.5.2	Dialogs	98
6.5.3	Main Window	98
6.6	Show Position	100
6.6.1	Menus	100
6.6.2	Dialogs	102
6.6.3	Dock Widgets	103
6.6.4	Main Window	105
6.6.5	Process	106
6.7	The whole system	108
6.8	Hardware Simulator - gain data daemon backend	109
6.9	Outcome	109
7	Critical Evaluation	110
7.1	Planning	110
7.1.1	Chosen Hardware	110
7.2	Design and Implementation	110
7.3	Conclusion	111
8	Prospect of the future	112
9	Acronym directory	114
10	References	116
A	Gantt charts	118
A.1	First Gantt chart	118
A.2	Second Gantt chart	118
B	Prototype source code	121
B.1	Main executable	121
B.2	Send executable	123
B.3	Include for common information	123
B.4	CMakeLists.txt build file	124
C	Several source codes	125
C.1	Time test sources	125
C.1.1	Java	125
C.1.2	C	125

C.1.3	C++	125
D	User Manuals	126
D.1	Compilation	126
D.2	OpenBeacon Configurator - Application	127
D.2.1	Pre requirements	128
D.2.2	Flashing a device	129
D.2.3	Configuring a device	132
D.2.4	Personalise the configurator	135
D.3	Data gain - Daemon	137
D.3.1	Configuration of the daemon	137
D.3.2	Command line parameter	138
D.3.3	Configuration and start up examples	138
D.4	Generate position - Daemon	140
D.4.1	Configuration of the daemon	140
D.4.2	Command line parameter	141
D.4.3	Example start	141
D.5	Administrate person data - Application	141
D.6	Show position - Application	145
D.7	Hardware Simulator - Application	148
E	Logbook	150

List of Figures

2.1	Example of Triangulation	17
2.2	Trilateration: wrong placement	17
2.3	Trilateration: right placement	17
2.4	Only one device	18
2.5	Only two devices	18
2.6	Three devices	20
2.7	Front of CCC Sputnik RFID tag	24
2.8	Back of CCC Sputnik RFID tag	24
2.9	Front of OpenBeacon USB node	25
2.10	Back of OpenBeacon USB node	25
2.11	Number ray of the original strength values	29
2.12	Number ray of the strength values after using the formula	29
3.1	Layers of the system	50
4.1	Project modularisation and flow of data	52
4.2	UML class diagram of the OpenBeacon Configurator	54
4.3	UML class diagram of the Gain Data Daemon	56
4.4	UML class diagram of the Map generation application	58
4.5	UML class diagram of the generate position daemon	59
4.6	UML class diagram of the Person data administration interface	60
4.7	UML class diagram of the Show position of a person application	62
4.8	UML class diagram of the OpenBeacon Node Communication Class	65
4.9	UML class diagram of the Data Exchange Class	66
4.10	UML class diagram of the Log Class	67
4.11	UML class diagram of the Map Class	68
4.12	UML class diagram of the Map Scene Class	69
5.1	First start of phpMyAdmin	78
5.2	phpMyAdmin MySQL interface	79
5.3	UML class diagram of the adjusted gain data daemon	83
5.4	UML class diagram of the adjusted data exchange server	84
5.5	UML class diagram of the adjusted data exchange client	85

5.6	Diagram of the adjusted project structure	85
A.1	Gantt chart of the planning phase	119
A.2	Gantt chart after analysing phase	120
D.1	The main window of the OpenBeacon Configurator	128
D.2	OpenBeacon Configurator with expanded device selection	130
D.3	OpenBeacon Configurator with selected image	131
D.4	OpenBeacon Configurator after flashing	132
D.5	OpenBeacon Configurator before command execution	133
D.6	OpenBeacon Configurator after command execution	134
D.7	OpenBeacon Configurator after command execution with an argument	135
D.8	OpenBeacon Configurator preference dialog	136
D.9	Default Commands Dialog from the preferences	136
D.10	Empty entry added to command list	137
D.11	Recently added entry will be changed	137
D.12	The main window of the Administrative person data application	142
D.13	Select connect from file menu	142
D.14	Database connection dialog	143
D.15	Main screen with selected entry	143
D.16	Colour selection	144
D.17	Picture selection	145
D.18	Main window of show position application	146
D.19	Database entries widget filled with a few entries	146
D.20	Connect to Generate position Dialog	147
D.21	Main window with connections build up and a visible tag	147
D.22	Widget that appears on mouse over	148
D.23	hardware simulator main screen	149

List of Tables

2.1	Methodology rating	15
2.2	h^2 Characteristics	19
2.3	Technology statistics	23
2.4	Dimensions of CCC Sputnik RFID tag	24
2.5	Dimensions of OpenBeacon USB hardware	25
2.6	Speed comparison of programming languages	35
9.1	Acronym directory	115
D.1	Configuration values for the configuration file	137
D.2	Command line parameters for the data gain daemon	138
D.3	Configuration values for the OpenBeacon USB Strategy	139
D.4	Command line parameters for the generate position daemon	141

1 Introduction

1.1 Background

Person localisation is a big issue nowadays, but mostly localisation is used to track down people who would not like to be tracked down. Most of the people think that location detection is only used in negative ways. This might be right in some ways but there are also cases where this detection might be used in everyday life. One of these cases is an audio stream that will follow the person or a stream that changes if a person moves from one place to another. An example of this can be used in museums where people have their own guided tour. They lend a wireless headset and then go from one exhibit to another and every time they reach the area around the exhibit, a voice will tell them what they want to know about it. Another example of use would be a sound stream that will follow the person from one room to another, keeping the noise level in rooms that are unallocated, as low as possible.

1.2 Problem

Most location detection systems are currently not affordable by the community, and there are not many applications available on the market, which make use of the advantage of person localisation. Therefore the first problem to solve is to find hardware that is affordable and easy to handle. The technology must also have an appropriate accuracy for applications that are utilising the location detection.

1.3 Delimitation

There are several ways of performing location detection, different hardware devices, and technologies. I limit myself to only one hardware device and one technology, but will design the project such that it should be possible to use other hardware as well. The project will provide a 2 dimensional location detection, with 3 dimensional location detection being outside the scope of the project.

1.4 Goal

This project will provide a basis of tools which provide an interface to the person location system tracker to easy build applications that can make use of this information. This project also includes a small application which will make use of this system.

2 Research

In the projects research stage, different technologies and tools as well as some development techniques will be discussed.

2.1 Methodologies

Methodologies are describing the method a project should be handled. They mostly define several stages or phases in which the developer had to complete a few tasks. Each methodology has its own ways to handle tasks like analysis, design, implementation and testing. If a project follows a specific methodology then the developers as well as the project manager should always know in which state the project is and how long it will take to reach a previously defined point. Some methodologies also have a detailed description how the customer should be included in the development process.

This project needs a methodology because the developer should always know what the current status of the project is and how much work is still left. This project has a defined beginning and a defined end, therefore the planing and designing can be made for the whole project, right before the coding will start.

2.1.1 Waterfall life cycle

The knowledge for this methodology came from the following two books, Fioravanti (2005, p.73–77) and Hansen (1998, p.138–140).

In the waterfall methodology all defining work has to be done before the coding can be started. Therefore all use cases and functionalities have to be described right before the implementation will start. This is why the project will be divided into single stages, each stage then will be done after its predecessor. So entering the next phase of a project is only possible if the one before is already completed. The five main stages of the waterfall methodology are (in this order): feasibility study, analysis, project specification, development, integration and testing, deployment and maintenance. In the following these phases will be described in detail.

Feasibility study

In this stage the feasibility of the project is checked: is it affordable, technically manageable and legal.

Analysis

In this stage of the project the functional details needed to be analysed and written down. The outcome of the stage are several documents that will describe the system and its surroundings in detail. Following is a detailed description affords a:

- Requirement specification document (RSD), describes the application and all its use cases, explained in all their detail. This document includes information regarding the user interfaces, as long as there are such interfaces. It will also show the workflow of the application.
- System test plan (STP), this document includes a full strategy about testing the application. All the possible test cases should be written down, as well as their expected outcome.

Project specification

In this phase the project manager will do the most work. He needs to define the software architecture and the technologies of this project. Therefore he will use the RSD to create those specifications adequate to each defined task of the application. This last aspect of this stage is to verify the milestones and the time line of the whole project. The project specification document (PSD) will be the outcome of this phase. This document includes all previously analysed data and all aspects of the architecture in detail.

Development

This stage is the main stage of the project. The project manager has to keep track of the project and has to look at the current stage of the project and if there will be problems that will cause a noncompliance to the planned time schedule. The developers on the other hand have to program and document the code. The outcome of this stage will be the source code and a code description.

Integration and testing

This phase is split into two parts. The first part is the integration of all modules to one project, if necessary the developers need to reprogram the relevant parts. After

everything fits together the second part will be the testing of the whole project. Depending on the developed project, this part needs to be a testing environment or not. The test phase normally is divided into two subparts, the alpha and the beta phase. Whilst in the alpha phase everything is tested in a closed environment and with a special look if everything is working like it is described in the STP (see Analysis), the beta phase includes costumers to test the product in their environment.

Deployment

At the deployment phase the project will be installed in the customers environment. All the debug and logging features that were necessary for the testing phase are switched off or taken from the code. Usually the end of the beta, start of the maintenance and start of the deployment phase are at the same time

Maintenance

In the maintenance phase the errors detected during the test phase will be corrected and delivered to the customer.

Pros and cons

- **Pro**

- Usable for small teams.
- Good if the project has a fixed finishing date

- **Con**

- Not very flexible, if something went wrong during the research stage it will break the methodology if it will be found at the implementation stage.
- Not usable for projects that will be improved during the development process.

2.1.2 Rapid application development (RAD)

The Rapid application development (RAD) uses the waterfall idea for its internal phases. Instead of waiting that every phase ends, RAD is focused to create a prototype real fast. The prototype is split into different phases, analysis, design and prototyping. After the prototype is done it will be shown to the users of the system. Then the users have the ability to ask for changes and improvements. After this

the whole prototyping phase starts over again, the changes and improvements of the users get involved into the new analysis, design and prototyping phase. Then the new prototype will again be brought to the users. This will loop as long as the users have something left to improve or the project manager will call a stop to the project.

The vantages of this system are that the users are included into the development process and their wishes are included directly on the development. The user can identify himself with the product and acquires the functions it provides faster.

One of the disadvantages is that the project has no defined end, if the users are asking for more and more features then it will not have an end. In that case the project manager has to stop the prototyping phases before the project runs out of money. Another disadvantage is that the user might have the feeling that the project is nearly done after the last prototype. But the prototype is still a prototype, a program that is put together fast with no thought out design of the full system, no user documentation and mostly no source documentation (Hentzen, 2002, p.25–28)

Pros and cons

- **Pro**

- Good for projects with no fixed end.
- Good for larger teams.
- Good for projects where the specification may change during development.
- Good for the customer to see that there is progress during the development.
- User is directly included into the development process.

- **Con**

- Bad for projects with fixed end.
- Bad if the customer want to have more and more features after every cycle.
- Bad because the application shown is a prototype only and the customer may think that it is a full, nearly done application.
- Not usable for non customer orientated projects.

2.1.3 Extreme Programming (XP)

The following methodology is written out of the knowledge from the following resources, Wikipedia (2007a) and the following resources to verify the information provided by Wikipedia. First of all the book and web page by the inventor of extreme programming Beck (2005) and Beck (2007), as well as the books by Fioravanti (2005, p.108–132) and by Hightower (2004, p.1–9).

Extreme Programming is based on the RAD methodology but adds a some more rules regarding the surroundings and the treatment of the customers. The whole methodology is based on the following five values and a bunch of principles and practices. But as Kent himself is saying theses rules are bendable and some might be dropped in certain circumstances and therefore other rules might come up. Those rules are depending on the company and the product the company is producing, if for example the project is a project which might risk human lives it should be possible, in any state of the project, to follow the whole chain of development back to the ground to see if it fits all the human security issues.

Five values

- **Communication**

The extended communication should not only be between programmers and the customers, but also between the programmers. More details will be discussed in the following. The communication is helping to solve problems as soon as they come up or if solving is not possible at the moment give all the communication partners something to think about, so they can come up with some solutions later on.

- **Simplicity**

In XP simplicity means that the project should start with simplest solution to acquire the project aim. More features will be added later on. The designing should not be done for the whole project but only for the daily task. The advantage is that a simple working application is ready real soon. The disadvantage is that it will get real hard to change some designing failures that occur when new features need to be included. On the other hand if the designing would have taken too long, so that the requirements have changed, the design had to be updated again.

The simplicity also should be focused on the team. If the team knows how to program a graphical user interface it would be best to this as dynamicly as possible but if the team does not know much about programming graphical user interfaces then it should use a front end to create the GUIs.

- **Feedback**

The feedback is connected close to the communication and simplicity. It splits itself up into three different types.

The systems feedback, where unit tests (small programs that test the application) are performed.

The customer feedback, are functional or acceptance tests, where the users of the systems test it and tell what is good and what isn't. And maybe some specifications will change during the development and the customer then might have the opportunity to react fast and tell those changes to the developers.

The teams feedback, every time the user wants to have a change the team directly will discuss the details and how long it will take to do it. Because of the pair programming and solving small problems at a time (see below) each developer will also have the chance to see the code of other programmers and then might have some advantages to this code. This will not only help with improving the source code but developers will also learn from more experienced ones

- **Courage**

Courage can be shown in several ways. Sometimes it is just telling the truth, like if some code is not good, or if a previously decided way is not as good as it should be courage meant that it should be spoken out and perhaps fully redone, no matter how hard it was to produce it.

- **Respect**

Developers should respect the work of each other and therefore not disturb or destroy the work by committing non compilable source. This will interrupt the work of all developers. They also should show their respect by handing in solutions with high quality and good design.

Respect should also be shown between the developers, they should care about each other and the project. Every one is important not only one individual.

Principles

The principles are building the fundament of XP. They are used to be more concrete if it comes to practical situations.

- **Humanity**

Humans develop software, therefore the developers should be treated as humans. They should not fear their job. They should also have the possibility

to receive validation or have the opportunity to expand their knowledge and their skills.

- **Economics**

Software is build to solve a business problem and therefore should also be profitable. The features with the highest value to the customer should be solved first so the value of the whole product is as high as possible

- **Mutual Benefit**

There are some situations in a project where solutions cost one person a lot of time whilst another might save time with it. An example might be a lot of in-line documentation in the source code, this might help a person that had nothing to do with the code so far but will slow down the person who is writing the code. Instead of doing so the initial programmer should write some automated tests and perhaps refactor the code to make the source less complex.

- **Self Similarity**

If a problem is similar to a another already solved problem it might be good to use the previously solved algorithm and just change the different parts. When the first algorithm was well tested then the new one should be running better than a whole new developed one.

- **Improvement**

“Nobody is perfect”, therefore the design and source code should always be improved. If a method is being reviewed to be enhanced by some features, and the developer who is reviewing it finds a better/easier way to solve the already solved problem than the new better solution should be implemented

- **Diversity**

Teams which are build of people who are alike, won’t work well together. The members of a team need to have various skills, different kinds of thinking and the interest to offer their view to the team.

- **Reflection**

During the cycles the team should come up with some time for reflection. In those time they speak of the failures they have done and how to prevent them in the future. They should inform each other how they solved the problems, so that every team member can learn and criticise.

- **Flow**

Flow means that all activities of the software development should be done

simultaneously instead of splitting it up into several phases. An example might be the technical documentation of the source code. It should be done during the development at the time the programmer knows best what was solved and how.

- **Opportunity**

With Opportunity, XP wants to let the developers see that a problem is a good opportunity to learn something new.

- **Failure**

If the developer does not know which of the ways for solving the problem should be chosen, then all ways should be implemented and then the developer can decide which one will do the job. Even if none of them is capable of solving the problem, the developer would have learned something out of the implemented ways.

- **Quality**

The project should be build with a high quality. Higher quality does not mean that it takes more time to achieve it. In most cases, higher quality leads to a faster development of the project. Because the structure of the source might be better thought through and it is easier to find and eliminate bugs.

- **Baby Steps**

XP says that it is better to make small incremental changes than make big ones. The reason for this is that on the one hand the user may see the application more often and on the other hand it will decrease the error rate.

- **Accepted Responsibility**

With accepted responsibility XP means that if a developer accepts to do a part of the project, the person is also responsible for the design, implementation and the testing.

Practices

The following practices are a collection of the primary Practices introduced by the XP methodology. Not all of the practices introduced in XP have to be followed by the developers or project managers but they should show a basis where the developers can start with. In some cases the list of practices should be increased (or decreased) to fit better into the structure of the project. The Corollary Practices are meant to be used by experienced XP users, beginners on the other hand are allowed to not use them.

- **Sit together**

To increase productivity the team should sit together in one room where every developer can directly communicate with another.

- **Informative workspace**

create a work space which is related to the project the team at the workspace is doing. For a person which is not included into the project and its problem it should be easily understandable what the project is about and what its current problems are.

- **Energised work**

By energised work XP means that a developer should not work longer than he/she is able to. Problems are getting solved best when the developer who thinks about them is prepared, rested and relaxed and has his mind free from other stuff.

- **Pair programming**

With pair programming the XP practice means that the teams are split up into two developer teams where each pair is developing in front of one PC. The person who is the one who is typing is called the driver and his job is to code the current task and think about the details. The other developer is called the navigator. The navigators job is to review the code while it is written and to have a view on the global system. Those roles are not fixed. The two developers have to change the roles every half an hour or after a unit test.

One of the pros is that a novice programmer can learn fast from an experienced one. One of the cons are that experienced developer might get bored of helping one who is not that experienced.

- **Stories**

Special customer wanted functionalities should be stored as stories, they are not specific requirements to a system. Something like “*Provide a way to simply open saved search*” might be a story, each should be written down on a piece of paper and then can be chosen by the developers whom wants to solve it.

- **Weekly cycle** Each week should be used as a cycle to get further in the development. At the beginning of each week there should be a meeting where the programmers review their progress from the last week and if it is like they expected last week. This is also the time where immediate solved stories for the customers might be assigned for the week and where the stories got broken up into tasks which the team members can assign to.

- **Continuous integration**

The developers should always work on a most up to date version of the application source, therefore the developers should frequently, like every one or two hours, submit their changes to the code repository. This will help to reduce problems that will occur if a developer tries to integrate the source on a later state of the development.

- **Test driven development**

In this practice the programmer needs to write unit tests before the actual code. This should help the developer with thinking through in which cases his code can fail. XP says that a code is tested well as soon as the developer can not think of a single flaw to test.

- **Incremental Design**

Because XP insists of programming and planning for only the work that needs to be done on one day, the system will reach a point where it got stuck. In this case the developers should think about the whole design and if necessary improve the design of the whole system by refactoring. This can lead to a rewrite of code, but this improvements then should lead to an easier to use code base, which allows the developers to go back to their daily planning and coding strategy.

- **Corollary - Real Customer Involvement**

In XP the whole team is not only the team that is developing the application. It is also the customer who will use the system. Thats why the customer should be included into the development, by answering question and helping out if developers do not understand the business problem itself.

- **Corollary - Shared Code**

Collective code ownership not only means that the code belongs to every one, but also that everyone has access to all the code. In case the system has an error, any programmer has the ability and the right to solve the error.

- **Corollary - Single Code Base**

The project should have only one code base, where every developer is committing source to. The development of a single developer might be made in a temporary branch but as soon as something is done and tested it should be committed back to the code base. This reduces complications if different developers develop on the same file and need to synchronise their work of weeks.

- **Corollary - Daily Deployment**

The current running version of a project should be deployed each night, this decreases the gap between the version the developer is using and the version the user is using. If the user is using a version that is close to the developer version, false decisions made by the developers might be found faster and would not cost much time, which might be invested better in some other part of the project.

Those practices are a selection of all the available practices provided by XP.

Criticism

Even if Extreme programming sounds like a perfect methodology, it has also some points of criticism. The methodology fits best if it is a project without a specific delivery date and has a bigger development team (up to 10 persons). If the team consist only of one or two persons XP would interrupt more than it would help. On teams that are larger than 10 persons the team should be split up into smaller teams.

Pros and cons

- **Pro**

- Good for larger teams (up to 10, or if larger split teams).
- Good for projects without a fixed ending date.
- Good for the developers, because they can enhance their knowledge as well (see “Pair programming“).
- Good for projects with ongoing payment.

- **Con**

- Bad for small teams (smaller than 2).
- Bad for projects with a fixed ending date.
- Bad if the project receives on fixed payment.

2.1.4 Code like hell and see what happens (CLHASWH) or Cowboy Coding

CLHASWH is an agile methodology and it is worth to mention it, because a lot of small projects use it and die or delay a lot¹. This methodology is agile because of the lack of structure, therefore the developer just starts programming without making any deeper thoughts about the internal structure of the project. A project is in need of a specification document which limits the surroundings and features each module should contain. If such a document is not existent the developer might want to add a few small features every now and then, just to make it more feature rich, but as a result of permanently adding features the source will get more and more messy and it will get harder to determine where errors are coming from and how they can be solved. When this stage is reached the handling of the project is much too complicated and the developer might lose any impulse to work on the project any further. The work will get slower until it stalls. The resulting project then has no well designed code (which is very hard to bring back to live), nearly no documentation and mostly a lot of open bugs. (Hentzen, 2002, p.36–37)

Pros and cons

- **Pro**
 - Good for very small projects.
 - Good for one person development.
- **Con**
 - Bad if project expands to a big projects.
 - Bad if a customer wants to see some design documents.

2.1.5 Conclusion

The Extreme Programming methodology is one of the most mature methodologies. There are two reasons why this methodology will not fit into this project. The first one is that XP is designed to work with a bigger development team, if the team is just one person then the methodology will loose some of it basic strengths like the pair programming, communication, developer feedback and all the other things that need at least two persons. XP is also very costumer oriented and therefore is not

¹This relays to the personal experience of the author, regarding previous projects (private ones and those made in a company, which did not use any methodology) and some open source projects he watched at the internet over the years

Methodology	Project length		Team-size		Complexity ²	Overall
	fixed	open end	small	big		
Waterfall	*****	*	*****	***	**	***
RAD	**	****	***	***	**	***
XP	**	*****	*	*****	****	****
CLHASWH	**	***	****	*	*	**

Table 2.1: Methodology rating (1 * is low, 5 are high)

suitable with this project. Rapid application development as the predecessor of XP has a similar disadvantage, each iteration needs the feedback of the customer, which is not given in this project. The other disadvantage of RAD is that it has no defined end for the project and it can easily happen that the project would need more time than it has given. The CLHASWH methodology is way to unstructured to use it in the actual project. This methodology can mostly be used in small projects with a low level of complexity. As soon as a project reaches a certain level of complexity the patching of the already written code would take more time than a full redesign and rewrite of the whole project. The Waterfall methodology has a good basic structure and is also usable by smaller teams, which made it perfect to use in this project. Anyway it still needs some adjustments in its different stages to fit with this project.

First of all this project will not have a feasibility study, beta testing phase nor deployment, because the project was an idea and not an actual project for a company. The Maintenance phase will stay but only as a result of the testing phase, to correct open errors.

In the analysis stage it will be necessary to figure out how the hardware communicates and how it is working, to determine the planning and design of the modules and functionality. To determine how the hardware is working the analysis outcome will be a small prototype which communicates with the hardware and displays its information. Another change to the analysis stage will be the outcome, the requirement specification document will be the system specification included in this document and the system test plans will only cover a quick overview of how the whole system should act

At the design stage it is necessary to know how the selected libraries are working. It would be essential to know how much work they already do and what still is to be done. Consequently some small test classes need to be written to see how it actually works.

The testing and documentation stages are not split up like the Waterfall model has it, but they will be done in parallel. Every time a module is tested it will be documented in the user manual or the installation instructions.

The last change will made will be for the project report. In order to have an up

to date project report it will be written during the whole project and not at the end of the project, when half of the project and the past problems are already forgotten. This will also cover the polishing phases after each successful written stage. Those polishing will be done in parallel with the preparations of the upcoming stage.

2.2 Location detection basics

To locate a point in a two dimensional area, the location detection system receives certain data. This data are distances to target and positions from where the distance information is coming.

There are several methodologies to retrieve information about the distance from a base station to a target. This project will only concentrate on three of them.

Distance retrieving through:

1. Signal Strength

If the distance is retrieved with the signal strength, the base station measures the signal strength of the targets signals. The retrieved signal strength then has to be compared with a table of previously made measurements. Such a table would show signal strengths and the associated distances.

2. Packet loss

If the packet loss methodology is used to retrieve the distance, then the base station will record the amount of data the target is sending back. If for example the target should send 10 packets and the base station is only retrieving 5, then the target is further away as if the base station is getting all 10 packets. Again there is a table needed where the packet loss is associated to the distance.

3. Time of Arrival

This methodology can be realized in two ways. In the first option the target is just stupid and sends all packets back to the base station. In this case the base station will send a packet which includes a timestamp of the sending time. and then just waits for this packet to come back. If the packet came back then the base station just needs to take the difference between the timestamp and the current time and will then see how much time the packet took.

On the second way the target has more intelligence, and is sending its own timestamp and the base station just needs to compare the received timestamp to its own current time and again the time difference gives the information

how long the packet take. This last way has one big problem, both target and base station need to sync their time and this can be hard to implement.

Both ways again need a table where time is associated to distance.

Not only the distance is important to locate the target, also the position of the base stations should be taken into consideration. Therefore we look into the methodology of trilateration.

Trilateration is similar to triangulation (see figure 2.1), where a distance from a target is calculated by two given Points and the angles from the target to those points. Trilateration on the other hand uses the locations of three points and their distances to the target. Three points are the minimum for a two dimensional location detection.

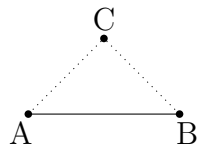


Fig. 2.1: Example of Triangulation

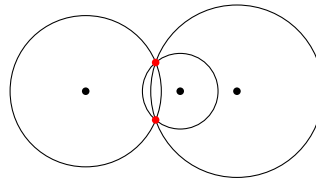


Fig. 2.2: Trilateration:
wrong placement

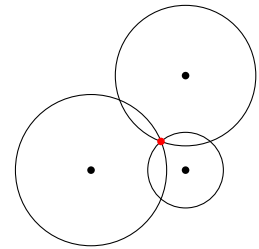


Fig. 2.3: Trilateration:
right placement

As you can see on figure 2.2, the 3 points are not allowed to be on one line, a better approach is shown on figure 2.3.

2.3 Mathematical fundamentals for location detection

This section describes how to calculate a trilateration.

2.3.1 One device

If there is only one device it would not be clear where exactly the person, which should be located, is. Only the distance to that specific device is known.

As can be seen in figure 2.4 the person can be anywhere on the circle.

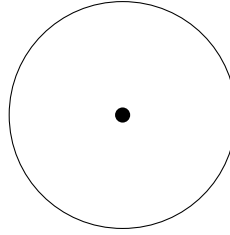


Figure 2.4: what it would look like if there would be only one device

2.3.2 Two devices

If there were only two devices it would be much easier to clarify where the person is. The exact position is still not found, but instead of an infinite amount of possible positions it is reduced to 2 possible positions.

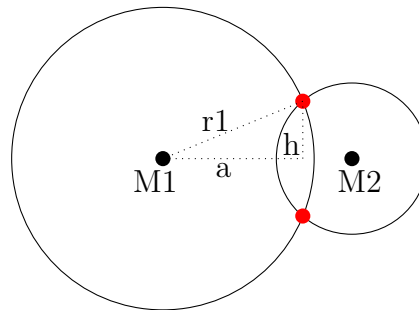


Figure 2.5: what it would look like if there would be only two devices

To obtain the positions of the red dots in figure 2.5 one have to calculate using the following formulas:

The given values are:

Circle one: x_1, y_1, r_1

Circle two: x_2, y_2, r_2

Where x_n and y_n are the two coordinations of the circles and r_n are the radii of the circles. At first the distance (d) between the two middle points needs to be calculated:

$$\begin{aligned} d_x &= x_2 - x_1 \\ d_y &= y_2 - y_1 \\ d &= \sqrt{d_x^2 + d_y^2} \end{aligned} \tag{2.1}$$

If d is zero then the two circles have the same middle point and either the two circles are identical or do not have any intersection points. After the distances between the two middle points is calculated the distance a (see figure 2.5) needs to

be calculated:

$$a = \frac{r_1^2 - r_2^2 + d^2}{2 * d} \quad (2.2)$$

The last distance that need to be calculated is the height h (see figure 2.5).

$$\begin{aligned} h^2 &= r_1^2 - a^2 \\ h &= \sqrt{r_1^2 - a^2} \end{aligned} \quad (2.3)$$

The following table shows the characteristics of h^2 .

$h^2 < 0$	no intersection point
$h^2 = 0$	exactly one intersection point
$h^2 > 0$	two intersection points with the distance $2 * h$

Table 2.2: h^2 Characteristics

After the height is calculated the coordination(s) of the intersection point(s) can be calculated:

$$\begin{aligned} xi_1 &= x_1 + \frac{a}{d} * d_x - \frac{h}{d} * d_y \\ yi_1 &= y_1 + \frac{a}{d} * d_y + \frac{h}{d} * d_x \end{aligned} \quad (2.4)$$

$$\begin{aligned} xi_2 &= x_1 + \frac{a}{d} * d_x + \frac{h}{d} * d_y \\ yi_2 &= y_1 + \frac{a}{d} * d_y - \frac{h}{d} * d_x \end{aligned} \quad (2.5)$$

Equation 2.4 shows how the first intersection point is calculated and 2.5 how the second. (Kowalski, 2008)

2.3.3 Three devices

Finally if there are 3 devices it is possible to detect the correct position of the person to locate.

To calculate the location of the red dot in figure 2.6, the previously created equations, 2.4 and 2.5, for getting the coordinates of the two intersection points will be needed. Both have a maximum of two results, which will result into (xi_1, yi_1) and (xi_2, yi_2) . Now the only test that needs to be performed is checking which of those

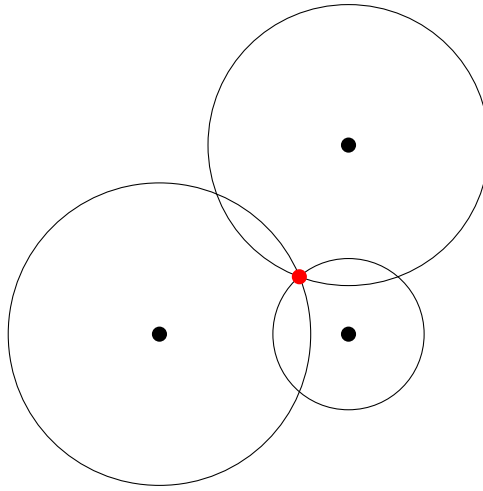


Figure 2.6: what it would look like if there would be three devices

two points is on the third circle. The formula for calculating this is as follows:

First calculate the distance from the circles middle $M(M_x, M_y)$ to the point $P(X, Y)$.

$$d = \sqrt{(X - M_x)^2 + (Y - M_y)^2} \quad (2.6)$$

Then compare the distance d with radius r of the circle, if $d < r$ then the point is in the circle, if $d > r$ then the point is not in the circle and if $d = r$ then the point is on the circles border.

2.4 Technologies and their Hardware

This section will cover a few technologies that can be used for location detection. The described technologies are limited to those who are easy working with a common personal computer.

2.4.1 WLAN

Wireless Local Area Network (WLAN) is commonly used for computers and other devices that need access to a Local Area Network (LAN). Mostly it is build up with access points, on which every wireless device must register. The communication is mostly driven over the 5 GHz and the 2.4 GHz public spectrum bands (see IEEE 802.11).

The access points have several information about every device that is registered with them. One of the useful information are the MAC address and the sending strength. With the MAC address every registered device is explicitly identifiable,

because every device should have a unique MAC address on the world. With the sending strength it is possible to find out how big the distance between the access point and the device is. The position accuracy of the devices is very high, around *10cm*.

To get information about the strength of the signal, the access points must have the ability to display it. In most cases access points do not have this ability with their default firmware, therefore the access points needed to get flashed with a new operating system. After flashing the access points are getting the ability to run custom software. This Software can be used to provide the requested data upon the signal strength. Those operating systems are mostly build on top of a small Linux and extend the original features considerably.

An example for such a firmware is dd-wrt (<http://www.dd-wrt.com/>). Dd-wrt provides access through a web interface as well as through a console using the ssh protocol. The console interface enables the use of self written programs, which then can use the router for their own needs. The console also allows the administrator to configure or to install new packages on the router.

To make location detection possible with the WLAN technology the following parts are needed. Like already explained in 2.3, three receiving devices are needed. At the WLAN technology those receiving devices are the access points. Suitable devices for clients are all devices that can connect to an access point, like mobile phones, PDAs and notebooks.

2.4.2 Bluetooth

Bluetooth was constructed as a low powered wireless connection. It is mainly used for applications that do not use a very high bandwidth. The Bandwidth varies in each standard, Bluetooth 1.0 has about 721 kbit/s and Bluetooth 2.0 has around 2.1 mbit/s.

The main application area are transmissions between mobile phones and communication between the peripheral computer hardware and the computer itself.

Like above in the WLAN section, we again need three devices that are capable of receiving Bluetooth signals and have the capability of getting attached to a computer. Additional to the receiving devices the project would be in need of a device that will act as a client, which sends a signal to the master device. The master devices are mostly called dongles, those are small device which are connected to the computers USB port. A master device can handle a maximum of 7 clients, so building up a detection network can become complex if the number of clients is larger than 7 (Muller, 2002, p.18–27). Possible client devices again are mobile phones, PDAs and notebooks.

The costs of building a network on top of the Bluetooth technology would be a little cheaper than the WLAN approach, as long as the number of clients is less than or equal to 7. If there are more than 7 clients this solution will become more expensive.

2.4.3 RFID

The Radio-Frequency Identification (RFID) technology is separated into two main areas of RFID chips (further more called tags). The first area is using passive tags the second one uses active ones. Passive tags do not have a power supply and only work through the short frequency impulse that hits them if a reader is trying to read the data that is stored on the chip. Because of that, the passive chips have only a very short range in which their data can be read, approximately 10 cm. The active chips on the other hand have their own power supply which gives them the opportunity to enforce the signal they are meant to send back to the reader. Those enforcement gives them the ability to broadcast their signal in a wider range than the passive ones, approximately 50 and more meters, depending on the attached power supply and the surroundings.

Because of the fact that every tag answers every reader the tags do not need to register themselves at the reader. The basic functionality is that the reader asks, by sending a specific signal on a specific frequency, all tags to answer him by sending their data back. Then all the tags that are in range will answer the sender by sending their data back to the sender.

The hardware for this technology is not yet buyable in every store because its mainly used by the industry for logistic solutions, and there are no real suitable applications to the normal human being up to now. The expensive parts on this technology are still the readers, clients which are sending the signals are the way more cheaper than on WLAN or Bluetooth. The construction costs of passive tags are a few pennies, only the costs of active tags is higher and is about 20 £.

2.4.4 ZigBee

The following information are taken from the official ZigBee Specification (Alliance, 2007) as well as some online technology specifications provided on wikipedia (Wikipedia, 2007b).

ZigBee is a technology that was developed by the industry to have an “simple, just working” low-cost wireless technology. These wireless connections should be used by any devices that need to connect to each other. Those devices are those used in the industry as well as home devices. The ZigBee technology is an advancement of

the WLAN and the Bluetooth technologies. The signal ranges are pretty much the same as those from WLAN and it is using the 2,4 GHz spectrum band.

The ZigBee network is using three different types of devices in its network structure.

- The **ZigBee coordinator (ZC)** is used as the coordinator of a whole network. It has also the capability of bridging the traffic into other networks.
- The **ZigBee Router (ZR)** is capable of being used as end device but also can send data to other routers or the coordinator.
- The **ZigBee End Device (ZED)** is the device which is used in applications, it does not have the capability to relay data, it only can communicate with routers or the coordinator.

Those technology is mostly used by the industry for industrial use, therefore the devices are not purchasable on every store.

2.4.5 Conclusion

ZigBee might have been the best choice for this project, because the parts are cheap and yet well elaborated. But the problem is that it is not easy to get parts for such a small project like this one.

Technology	Range	Accuracy	Transfer rate	Prizes ^a		Overall rating
				tag	node	
Bluetooth	8m	±5m	2.1 Mbit/s ^b , 480 Mbit/s ^c	100€(70£)	11€(8£)	*
WLAN	10-100m	±10cm	1-108 Mbit/s	100€(70£)	50€(35£)	***
RFID	10-80m	±10cm	< 20 Kbit/s	20€(15£)	85€(60£)	*****
ZigBee	10-100m	n/a	20-250 Kbit/s	<5€(3.5£)	<5€(3.5£)	****

^aall prizes are approximated

^bIn Version 2

^cIn Version 3

Table 2.3: Technology statistics (1 * is low, 5 are high)

Table 2.4.5 shows the key values of each technology and an overall rating done by the developer. The Rating is based on how the technology will fit into the given parameters, like having a good accuracy. The accuracies were measured by the developer. Because he did not have any ZigBee technology the accuracy of this technology could not be tested. But like written in the above statement the ZigBee technology was separated from the project as soon as it was evident that the needed parts are not easy purchasable.

Both the Bluetooth and the WLAN technology were not chosen because they both have no “independent sending-only” tags. Which means that the user who wants to be located needs at least a mobile phone or a PDA with those capabilities for detection. Both devices are not very cheap and both are way to big to carry around.

Those facts and the fact that the RFID technology is a relatively new and interesting technology made the decision, that it would be nice to develop the system on top of the RFID technology. Anyway the system should be designed to easily adapt other technologies by simply implementing the interface for the other hardware.

2.4.6 Chosen Hardware

After some searches for suitable hardware, the OpenBeacon project (<http://www.openbeacon.org/>) was found. This project provides both needed parts for the location detection.

Tag

The tags are small round electronic units with the following measurements:

Diameter:	3.5cm
Depth:	0.7cm

Table 2.4: Dimensions of CCC Sputnik RFID tag

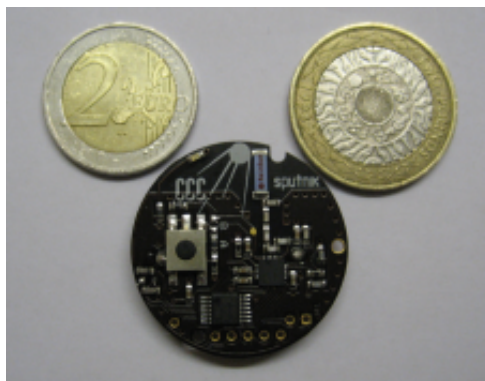


Figure 2.7: Front of CCC Sputnik RFID tag



Figure 2.8: Back of CCC Sputnik RFID tag

Figure 2.7 shows the front of the selected hardware tag. This specific tag has an additional function which allows to send a press of a button to the receivers and on figure 2.8 the battery case with the imprinted tag id are well visible.

The tag needs a CR 2032 Battery with 3V. The battery will not be replaced for approximately one year. The tag has also a small little light emitting diode (LED) which blinks around every second to show that the tag is working and sends data.

Reader

The OpenBeacon project also supplies an adequate reader that uses the USB port to provide the data. The dimensions³ of the OpenBeacon USB node are shown in table 2.5.

Node		Antenna	
Width:	7.5cm ~ 8cm with an applied antenna	Length:	9.7cm
Height:	2.5cm	Diameter:	1cm
Depth:	1.2cm	Socket length:	1.7cm
		Socket diameter:	1cm

Table 2.5: Dimensions of OpenBeacon USB hardware

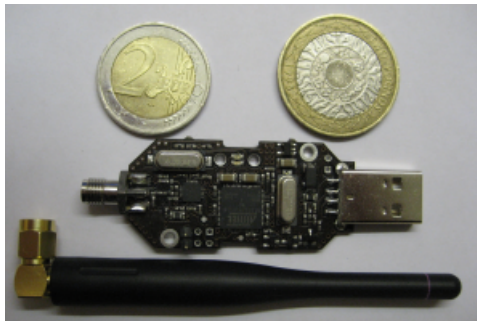


Figure 2.9: Front of OpenBeacon USB node

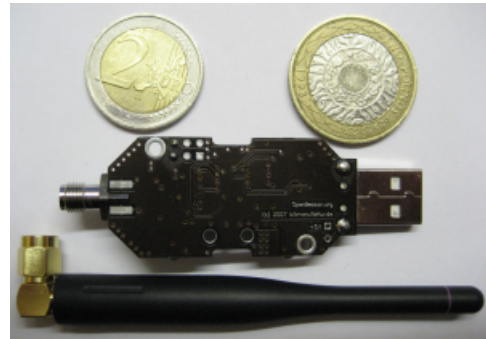


Figure 2.10: Back of OpenBeacon USB node

Figure 2.9 shows the upper side of the OpenBeacon USB node on the left end is the connector to the 2,4GHz antenna and on the right side is the USB port. Figure 2.10 shows the back of the OpenBeacon USB node. Both figures show the unattached 2,4GHz antenna on the bottom. The antenna itself has a free turnable joint which allows the user to turn the antenna in different directions when it is applied to the node.

To perform the location detection there are two of the tags and three reader. This project needs two tags to have the possibility to check what happens if more than one tag is in range.

Changes to operating system

To operate with the hardware the operating systems needed a few changes and a new set of tools to communicate with the reader.

³Height and depth might change when antenna is applied and turned in some way

Kernel changes

First thing that had to be changed was the kernel. This needed to be done to let any kind of software address the hardware and talk to it, therefore a device was added. The device that was needed for the USB reader is a serial USB device in this case an abstract control model (ACM) device. If the kernel module cdc-acm is not already available, then the kernel needs to be rebuild with the enabled option.

To enable USB ACM support. the following needs to be marked as module or built in:

```
Device Drivers
-> USB Support
    -> Support for Host-side USB
        -> USB Modem (CDC ACM) Support
```

Listing 2.1: Where to enable ACM support

If it was build as a module then it would help to simply start the module by entering `modprobe cdc-acm` and if it was built in the computer needed to be restarted to load the new kernel. After that is done a device will show up as soon as the OpenBeacon USB node is plugged into a free USB port. The upcoming devices name will be `/dev/ttyACM?`. The ? stands for the number of the device.

To **flash** the reader with a new firmware, the operating system is in need of an USB serial device. If the `usbserial` module does not exists the following must be set to module or built in.

```
Device Drivers
-> USB Support
    -> USB Serial Converter support
        -> USB Serial Converter support
```

Listing 2.2: Where to enable USB serial

Again if it was as module just load it with `modprobe usbserial` or restart the computer to enable the new kernel. When the computer booted again plug in the reader and as soon it is plugged in there should be a device called `/dev/ttyUSB?`. Again the ? stands for the number of the device. If the device does not show up then the systems device manager is not configured to handle the USB serial module. To configure this right the following must be done as root:

```
$ echo "BUS==\"usb\", KERNEL==\"ttyUSB*\", GROUP=\"uucp\", MODE
    =\"0666\"" >> /etc/udev/rules.d/10-custom.rules
$ echo "BUS==\"usb\", KERNEL==\"ttyACM*\", GROUP=\"uucp\", MODE
    =\"0660\"" >> /etc/udev/rules.d/10-custom.rules
```

```
$ udevstart
```

Listing 2.3: How to enable usbserial in udev

After this the reader needs to be plugged out and re-plugged in and the modules need to be reloaded. If the device still does not show up, the following needs to be done, as root again:

```
$ lsusb
Bus 001 Device 001: ID 0000:0000
Bus 002 Device 004: ID 03eb:6124 Atmel Corp.
Bus 002 Device 002: ID 04f3:0210 Elan Microelectronics Corp.
Bus 002 Device 001: ID 0000:0000
$ modprobe usbserial vendor=0x03eb product=0x6124
```

Listing 2.4: How to modprobe usbserial

The output from `lsusb` should show an Atmel Corp. device. The first number before the colon is the vendor id, the one after it is the product id. If they differ, the `modprobe` call needs to be adjusted appropriately. After all that work, the reader is ready to get flashed.

Tools

In order to flash the OpenBeacon node with a new firmware version, the Linux operating system needed to be extended by the following tools:

- A compiler to build firmware binaries that could run on the OpenBeacon node. The compiler used is the `gcc-arm`, this special version of the `gcc` produces binaries for the arm processor.
- A tool to write the binary firmware file on the OpenBeacon node, those tools are normally called flash tools. The flash tool that was used for this project is a modified version from the OpenPCD (<http://www.openpcd.org/>) project that comes from the same supplier as the OpenBeacon project. Those flash tools are available from here: <http://www.openpcd.org/dl/sam7utils-0.1.0-bm.tar.bz2>.

There are only three steps needed to flash a new firmware on the OpenBeacon node.

1. **Create and compile new Firmware**

Change or create new firmware and compile it with the `gcc-arm` compiler.

2. **Delete the current Firmware** To delete the firmware, the node needs to be unplugged, then a jumper needs to be placed on the two pins. The second

step is to plug the node back into the USB port of a running computer. After 10 seconds the current firmware will be deleted and the node needs to be pulled out. After the jumper is removed the node can be plugged back into the USB port again. Now a `lsusb` should show something like it is shown in listing 2.4.

3. **Flashing the OpenBeacon node** If there is no `/dev/ttyUSB0` then the `modprobe` command from listing 2.4 should be used to create it. After the device shown up the `at91flash` script (delivered with the original firmware) should be used to flash the device. As soon as the flashing is done the last task is to unplug and re-plug the device to let the computer take notice that the device has changed.

2.4.7 Analysis of the technology

How does the hardware work

When the hardware is plugged in and the operating system is configured right, it can be tested with the `cu` program from Taylors UUCP package. The connection can be made with the following command:

```
$ cu -l ttyACM0 -s 115200
RX: 0,2213,8
RX: 0,2213,8
RX: 0,2213,8
RX: 0,2213,8
RX: 0,2213,9
RX: 0,2213,9
RX: 0,2213,9
RX: 0,2213,9
```

Listing 2.5: `cu` command to access OpenBeacon node

As shown in listing 2.5 the information from the OpenBeacon node is starting right after it is connected. Each line provides the following information (in that order):

1. **RX** to show that this is a received information from a tag
2. The **first number** is the lowest strength on which the signal is received, it can be one of the following: 0, 85, 170, 255.
3. The **second number** is the id of the tag to which the received information belong.
4. The **last number** is the amount of received packages, in this example the maximum are 10 packages.

This was the first thing that was different than expected. Expected was that the OpenBeacon and the tags would deliver the signal strength but this was a wrong assumption. But it should be no problem to implement a location system based on the packet loss distance retrieving described in section 2.2. Figure 2.11 shows a number ray of the strength values and how they relate to the distance from the tag to the node.

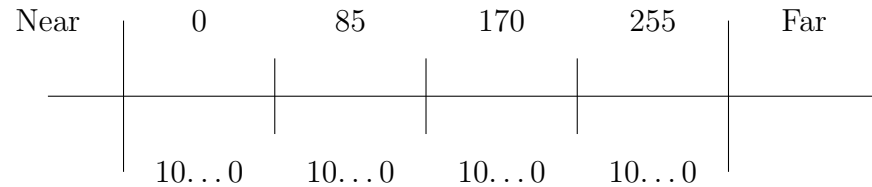


Figure 2.11: Number ray of the original strength values

To make an easier location detection the absolute strength will be calculated with the following formula: $\text{strength} = \frac{\text{power level}}{85} * 10 + (\text{maximum packages} - \text{packet loss})$. Figure 2.12 shows a the number ray with the effect the formula gives to the original strength number ray.

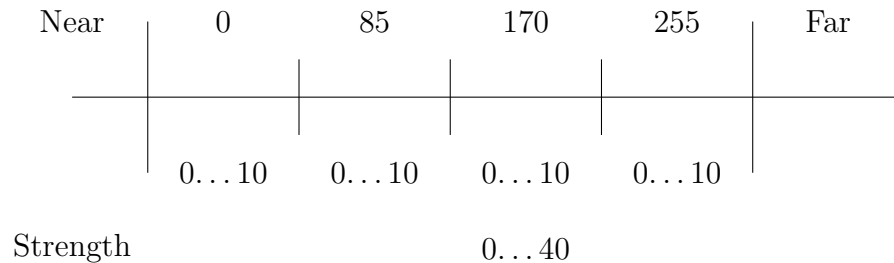


Figure 2.12: Number ray of the strength values after using the formula

The OpenBeacon node firmware also provide some configuration functionalities. The help screen, will show up as soon as the user types ? or h followed by enter.

```
RX: 0,2213,9
RX: 0,2213,9
Command received: ?

*****
* OpenBeacon USB terminal                               *
* (C) 2007 Milosch Meriac <meriac@openbeacon.de>        *
*****

*
* S          - store transmitter settings
* C          - print configuration
```



```

* I [id]      - set reader id
* FIFO [sec] - set FIFO cache lifetime in seconds
* 0           - receive only mode
* 1..4        - automatic transmit at selected power levels
* ?,H        - display this help screen
*
*****

```

Listing 2.6: Console output of the help screen

The help screen from listing 2.6 shows that it is possible to change a few settings that will be needed by the project.

```

RX: 0,2213,9
RX: 0,2213,9
Command received:C
*****
* Current configuration:                                     *
*****
*
* Uptime is 0h:4m:37s
* The reader id is 1
* The mode is 1
* The channel is 81
* The FIFO cache lifetime is 10s
*
*****

```

Listing 2.7: Console output of the settings screen

The listing 2.7 shows the settings screen. It displays the uptime, the id of the reader (it can be between 0 and 255), the mode (0 if it is in receiving mode, 1 to 4 for the power levels of sending), the channel and the lifetime of the package cache.

Missing firmware capabilities

As shown in the listings 2.7 and 2.6 it is not possible to retrieve information on a single output line. Therefore the project needs to enhance the firmware with the following abilities:

- Retrieving information on a single line for easier parsing of information like the id.
- It also misses some range queries, it should ask if the number of the id is higher than 255. At the moment it allows nearly every number, but this can be dangerous because with too high numbers the memory after the number can be lost (buffer overflow)

2.4.8 Writing a prototype to access the Hardware

This prototype should be used to find out how much work it will be to make the hardware work and with which commands it can be accessed. Basicly it should do all the basic things that can `cudo`, connect, read, write and disconnect. It will not have much error handling or checks if the user used the program in the right way.

Basic design

Basicly the prototype consists of two modules.

1. The first module would be the main module it opens the port to the hardware and will listen to it. Everything that is read from the OpenBeacon node will be printed to the screen. It will also listen to a message queue for incoming commands from the user. As soon as some commands are coming in the main module should send them to the node. Then the results should be printed, like all information that is coming from the hardware.
2. The second module is the controlling unit, whilst the first module just prints everything it gets from the OpenBeacon node, this second module has the capability to give the first module orders. Those orders are send through a message queue and are just the same commands like displayed in the help screen you can see in listing 2.6 with an additional command to let the module quit. The commanding module will only send a command and afterwards it quits, if the user wants to send more than one command, the module has to be called more than once.

Both Modules are running on the console only, there will be no graphical user interface.

Outcome

The full source code and build file can be found on the delivered CD as well as in appendix B. One of the base concept of all UNIX like systems is that “*Everything is a file*“. In close that means that a program can access a system device like it can access a file. That is the reason why it was so easy to create the prototype, there are only a few basic functions needed to work with the OpenBeacon device.

- The **open** instruction opens a file or device and returns a file descriptor which then can easily used by the other three commands. (An example can be found on listing B.1 in the appendix B at line 26)

- The **read** instruction reads bytes from the file descriptor, the command can read only a specified amount of data. (Listing B.1 in the appendix B at line 60)
- With the **write** command the program can write bytes to the file descriptor. (Listing B.1 in the appendix B at line 81)
- The **close** instruction is used to close the file descriptor after it is no longer of use. (Listing B.1 in the appendix B at line 91)

Detailed information to each of those commands can be found in the manual pages of the Linux operating system.

2.5 Operating systems

This section will cover the basic two operating systems that could have been used to realize this project. The developer needs to know which operating system the development will be done to verify that the hardware which will be used is able to work with it. This section will give a short overview over Microsoft Windows and Linux.

2.5.1 Microsoft Windows

Microsoft Windows is one of the most used desktop operating systems in the world. It has a graphical user interface and a not very powerful command shell. Most of the software that is provided for MS Windows must be purchased, but many open source projects have windows ports so they can run there as well. A big advantage of Microsoft Windows is that most of the software that is available for Windows is easy install-able and works right out of the box, but as soon as something is not happening like it should be it is not so easy to figure out what is causing the problem. A big disadvantage of this operating system is that on every new release the end user has to buy the new version. The EULA from Microsoft Windows limits its users in a lot of ways like it forbids the multiple installation on various self owned machines, every machine needs its own licence.

2.5.2 Linux

Linux is an open source operating system initially created by Linus Torvalds. Nowadays there are several thousand people who develop for Linux and tools that run on Linux. Linux itself is just the kernel which brings up the system. The kernel

also provides an easy to use interface to the hardware of the system. On top of it, programs will handle the user input. This input can be made on the console or on a graphical user interface. To get a graphical user interface the system needs an X Window system and a desktop environment that is running on top of it. The console or shell on Linux is a very powerful tool, because nearly everything that can be done in the graphical interface can also be done in the console. A big advantage for developers is that Linux provides a lot of library's to speak to hardware. A big disadvantage is that many software companies do not support Linux due to a binary incompatibility of those systems. Most of the tools that are used in Linux are free to install and use.

2.5.3 Conclusion

This project will make use of the Linux operating system. This decision was made with having in mind that searching for errors regarding device drivers are much easier to find in Linux than they will be found in Microsoft Windows. Another reason for this decision is that the developer is working best in the operating system that the developer is using at home or for personal use. Linux is even a good choice if the costs of the project should be kept low because most of the software that will be used is open source and therefore it is free of charge. This will lower the costs of the project only to the parts that need to be bought to achieve the location detection. The operating system and the development tools are free of charge and the computer on which the software will be developed is already existent.

2.6 Licences

There are a lot of licences, nearly every company has its own licences and copyright on their code, that allows or disallows several things. Most of the free open source software's (FOSS) have licenses as well. This section will cover a few of them. The most famous one is the GNU General Public Licence (GPL), it is widely spread and a lot of well known projects like the Linux kernel or the K Desktop Environment (KDE) are using it. It allows other people to see, even change the code, but only if they are too publishing their source under the GPL. The developer is allowed to charge people for his effort in writing the source but he has to publish the source as well.

Another Licence that will take effect in this project is the Q Public License Version 1.0, the official Qt (for more information regarding Qt reed section 2.7) licence. This

licence allows every developer to use this library without charge as long as he reveals the whole project code as well. If the developer wants to take money for his effort then he/she has to pay for the Qt library as well.

For further information upon the Open source licences visit the Open Source Initiative licences web page <http://www.opensource.org/licenses/alphabetical>.

2.6.1 Conclusion

This project will use the GPL to protect the sources and the developer who wrote them. By putting the source under the GPL it allows other developers to use it and change it for their own needs. After the project is done it will be released to the public so they can use it if they want to. Putting the RFID domestic location detection project under the GPL will also help with the spreading of the technology and the idea of using it in another way than it was planned to be.

2.7 Programming languages and libraries

This section will handle the choice of the programming languages and will describe why others were not taken. It will also give a short overview of the Qt library.

C and C++ were chosen, because they are fast and have the ability to work close to the underlying hardware. Qt on the other hand is a great and well thought out set of classes that will help to concentrate the programming on the main tasks. Qt provides a set of standard functionalities like strings, lists, maps and even some communication classes like TCP/IP or UDP network or D-Bus (more on this in section 2.13) inter process communication (IPC). The implementations of strings, lists and maps are significantly better than those from the Standard Template Library (STL). In addition to a good threading support, it also includes a wide set of graphical user interface classes.

Qt's event system is also a well thought out system, instead of handling messages like it is done in other library's, it has a signal slot system. This signal slot system is a very powerful mechanism. If for instance a button is pressed in a GUI, a signal is invoked, now all previously connected slots (methods of the same or from other classes) will be executed. This system is not only working for the GUI, it will also work within all classes Qt provides as well as all project classes that will be need such a functionality.

Another big advantage of Qt is its cross platform compatibility, which lets it run on at least Windows, Mac OS X and Linux. So applications written in Qt,

are easy portable. Additional information can be found on the Qt web site <http://trolltech.com/>.

Other Library's like the Microsoft Foundation Classes (MFC) or C# do not have much of the above mentioned features, and a big disadvantage is that they do not have the ability to run on other systems than windows.

Java as an alternative programming language has more disadvantages than advantages for this project. The advantages are that the code is easily ported to every system where Java will run on. The disadvantages are that it is necessary to have Java itself installed on the system as well as it is using a lot of memory when it is running. The virtual machine requires a lot of memory, or just allocates it to have it ready just in case the application would need it. Also the garbage collector is a nice thing to have but if the programmer can think of its own and knows how to handle memory well, he will not need it. As table 2.6 shows, Java consumes a lot more time than a plain C or C++ program. Those measurements were done with the Linux `time` command and the sources can be found in appendix C.1.

Time	C	C++	Java
real time ^a	0m0.002s	0m0.003s	0m0.165s
user CPU time	0m0.001s	0m0.001s	0m0.100s
system CPU time	0m0.001s	0m0.003s	0m0.022s

^aTime that elapsed from invocation to termination

Table 2.6: Speed comparison of programming languages

As web programming languages there will be used HTML and PHP. Both are capable of running on different operating systems or web servers. This enables flexibility to the end user who chooses for himself on which system he will put the web configuration interfaces. Another web programming language is ASP (Active Server Page). It is designed to run on Microsoft Servers. Even if it is possible to run ASPs on other systems it will never run as good as on Microsoft systems, because the support of other systems is not given through Microsoft, but through third party companies.

2.7.1 Closer description of chosen language and libraries

The main language that will be used in this project is C++, due to the easy handling with devices under Linux and due to the good speed advantage like shown in table

2.6. To support C++ the Qt library will be used. Qt offers a lot of subclasses that make it easier to code in C++. In addition to the helping Qt classes, Qt also provides an easy signal slot mechanism to communicate between objects. Both topics will be described below.

Useful Qt subclasses

Qt is separated into several theme related parts.

- **Threading**

Qt provides a threading system that is easy to use and has the same capabilities like a non threaded Qt application like the very usable signal and slot inter object communication (see section 2.7.1). This allows a quick response from one thread to another.

- **QtCore**

The QtCore class collection has all the standard base classes like the Qt String class QString or several types of lists, QList and maps, QMap. Those core classes are used in every other part of Qt and are building the fundament of the Qt library.

- **QtDBus**

The QtDBus collection holds all classes and methods that are needed for the D-Bus IPC.

- **QtGui**

The QtGui subsystem includes all classes that are used to create GUIs. it has the capability to create GUIs in the source code by just putting the several components together or just include static created (with the included Qt designer) user interfaces.

- **QtNetwork**

This subsystem is the network system of Qt and includes a lot of functions that make it easier to work with network communications. It provides the standard UDP and TCP communication as well as encrypted communication through SSL or protocol specific communication like ftp or http.

- **QtOpenGL** This subsystem of Qt provides a set of functionality for OpenGL graphic handling.

- **QtScript** QtScript provides a powerful scripting language to those Qt applications that are in need of one.

- **QtSql** With QSql the Qt library provides simple access to several databases. The type of database may be chosen at runtime. This option allows the user to have much more flexibility with using the application and does not force to use a specific product
- **QtSvg** The QtSvg facility provides Qt with the ability to create and display Scalable Vector Graphics (SVG).
- **QtTest** the QtTest subsystem provides classes which allow the developer to create unit tests. Those tests can be used to verify that parts of the new created classes are going well. If in a later stage of development a change will be made to an old class and the old class provides some unit tests, the developer has the ability to check if the new changes broke anything.
- **QtXml** The QtXml module provides handling of the Extensible Markup Language (XML), stream based reader and writer as well as an C++ implementation of the “Simple API for XML” (SAX) and the “Document Object Model” (DOM).

Qt’s signal and slot inter object communication

Qt provides a very powerful system which enables a communication between several objects, called the signals and slots system. This system allows objects to send signals if something important or something which is interesting for other objects occurs. Those signals can be optionally connected to slots, but they do not have to be. If they are not connected they will be ignored by the application. But if they are connected other objects can immediately react to the event that happened in the other object. A signal can also be attached to multiple slots and vice versa.

This key-system makes the big difference from most of the other development frameworks around.

2.8 Comparison between various code version control systems

A code repository is a system that watches over the different versions of a software project. Those different versions are mostly called revisions. Every upload of changed code will be reflected in an increment of the revision number. Those systems help software developers in keeping track of their project sources. If for example a developer has programmed something wrong and the whole project is not working anymore then the developer can undo everything to the last submitted

state that was working.

The Concurrent Versions System (CVS) and Subversion (SVN) are both working with this type of mechanism. The big difference between those two is their handling of commits, whilst the CVS is uploading all the files that have changed, SVN is just uploading a difference file to the server. On the first look this is no big deal, but if the project has large data files greater than 100MB, the CVS will upload 100MB every time there was a little change even if the change was just two changed byte. SVN on the other hand would only upload a difference file which describes the difference. If now someone updates its version of the project only the difference would be retrieved, if this person uses CVS it has to download the whole file.

There are even other advantages like the handling itself, for example the user of SVN has to authenticate himself only once at the server, after that login information will be stored at the local code repository. This helps keeping the commands, regarding this repository, simple.

2.8.1 Conclusion

This project will use the SVN code repository system because it fits well with the Software Project Management System that was chosen (see section 2.10 for more details). It is known to minimise the traffic it is generating and therefore will be perfect to send data to a server which is not located at the university and therefore has a slower data connection.

2.9 Build environments

Build environments are used to automatically build a whole project. If they are more powerful, they can even find needed libraries and link them to the project. There are a lot of build environments, but on this section we are only looking on three of them, CMake, SCons and a plain makefile.

2.9.1 Plain makefile

The plain makefile is just a small script that is used to build the application out of the source. The other two build environments have more intelligence. Those have the ability to find needed library's and link them to the application. They also provide mechanisms to configure the build type of the application, if for example

an application can be build with console, graphical or both interfaces this could be specified before building.

2.9.2 SCons

SCons on the other hand consists of a single script, which uses python (a cross platform scripting language) for the configuration and build tasks. It has a few special functions which detect libraries. Anyway a SCons construction file is a large file and for small projects this file is sometimes larger (in size) than the whole project. The building of such a construct file is much more complex and needs knowledge in python as well.

2.9.3 CMake

CMake in the end, built on top of easy readable configuration files and CMake modules. These modules are used to find the librarys that are needed for the project. The configuration file is an easy to create and read file, which uses the modules to find and link the librarys. None the less it has the ability to compile the source on several platforms, as long as the source is capable of doing it. CMake creates a makefile which then can be used to build the application. Additional to the easy producible and understandable configuration file CMake is also faster than the other two build environments.

The new version of KDE (K Desktop Environment) is using CMake as its building environment and is heavily based on Qt (see section 2.7), therefore the Qt integration in CMake is well structured and easy to handle (Neundorf, 2006).

2.9.4 Conclusion

All three building environments have their pros and cons, the plain makefile is easy to create but it is not very usable if it comes to automatic path finding for includes and libraries. It is also not able to be used on more than one platform and therefore the application needs a makefile file for each platform. SCons has the abilities to find the paths and libraries even on different platforms but it is complex and needs additional knowledge of a scripting language. Those arguments are making CMake the best choice, because it is usable on different platforms, it finds paths and libraries by its own and the syntax, even if it is a new one to learn, is clear structured, easy to learn and needs less space than the SCons script.

Because of those arguments the project will make use of the CMake building environment.

2.10 Software Project Management

Software Project Management systems are used to manage a whole project and everything that is useful to a project. Those systems provide a wide set of functions to have an overview and a view of the current state of the project. This project will make use of one of them to have an easy system to manage the project and all of its aspects. A bug and ticket system will help to not forget things that still need to be done. The system will also provide other interested people (like the supervisor) with an overview of the system. This section will give an overview over two web based systems, trac and dotproject.

2.10.1 trac

Trac (<http://trac.edgewall.org>) is a web-based software configuration management (SCM) combined with a web interface to the project related SVN (see section 2.8) repository.

The functions trac provides are:

- **A project related Wiki**

This wiki can be used to give more information regarding the project.

- **A time line**

The time line system gives information about actions that have happened in the last few days. Those actions are updates to the source, creation of new tickets (see below), closing of tickets, edits on the wiki, solved milestones (see below) and much more

- **A road map**

This road map shows all open milestones and information to those. Each milestone has a percentage bar below its name which shows how far the milestone is done. The information to each milestone can be set by the developer, so it can be used to tell the current status as well as giving information about special bug fixes or workarounds.

- **A source browser**

The source browser is a web-interface to the project related SVN repository. It can be used to browse the source and it has the possibility to show changes which were made between several revisions.

- **A complex ticket system**

This ticket system is a big advantage of trac. Those tickets can be used to submit bugs or wishes. The tickets are assigned to several components and versions of the project and can have configurable statuses. The developer can even open tickets to show which tasks he has to solve before a milestone is done. Once a ticket is created all people have the ability to comment on the ticket or attach more information to it, but only developers have the ability to change the status.

- **An administration interface**

This interface provides the administrator with the ability to change user rights such as create/edit/delete new components and create/edit/delete new milestones.

2.10.2 dotproject

Dotproject (<http://www.dotproject.net>) is another web-based software management system, but it is designed for a company that has several projects and needs to shift resources around. It is able to handle multiple projects for multiple companies. The functions dotproject provides for each project are listed below:

- **Task**

At the task system the project manager has the ability to add/edit/delete tasks and events. Those tasks or events are shown to all the project members which can solve them.

- **Forum**

In the forum project members can discuss activities and problems or just use it for simple communication.

- **Gantt chart**

The system provides an up to date gantt chart which is generated on the fly with information that is known at this point of time.

- **Files**

The file system provides functions like up- and downloading files relating to the project. This is an easy way to provide the documentary to each developer without using a shared folder on the computer.

- **Ticket system**

The ticket system allows everyone to create a new ticket regarding to one of the products.

2.10.3 Conclusion

Both projects are very good as a project management system, but this project is used to be a single project therefore it will use the trac system, because it is more project orientated. Dotproject on the other hand is a good management system for a company or software development group which has the task to manage more than one project at a time. The integration of the code repository system in trac gives the developers and people who are interested in the details of a project more capabilities.

2.11 Databases

Databases are used to store information that appears more than once, such as addresses in an address book. The project will use databases for the application which is using the location detection. This section will give information about some databases, but will only look upon those which are cross platform compatible. The databases covered are a small overview of all available databases. It will not go into too much detail, because this project will use a database only for a very simple task which even could be handled through a simple file.

2.11.1 Oracle Database

The Oracle database is a database released by the Oracle Corporation. It runs on most server systems that are available, not only on Microsoft Windows, Apple Mac OS X and Linux but also on IBM mainframes, Sun SPARC's and several HP derivatives. The Oracle database is a commercial only database system and is built to support large databases and a huge amount of users at the same time.

2.11.2 Informix and DB2

Both, Informix and DB2 are database systems sold by IBM. The Informix database system was bought by IBM in 2001 and since then it was developed to work better together with the DB2 system. Both systems share some technology by now. In comparison to Oracle those two systems are not running on all common operating systems like Mac OS X and only DB2 is running on a mainframe system like z/OS. Both systems are closed source and are not available as free version for free projects.

2.11.3 MySQL

MySQL is one of the most popular free and open source database projects. It is only free as long as the application which is using it is free as well, if this is not the case then the MySQL database must be purchased.

MySQL uses a dedicated server which can handle multiple databases and all the connections that will be made to them. A great advantage is that most of the programming languages, web or application, have MySQL bindings to make it easy to use. MySQL supports a lot of operating systems like Microsoft Windows, Mac OS X, Linux and a few more.

2.11.4 Conclusion

Whilst Oracle, Informix and DB2 are good in their environments they are still to overpowered for this project. Additionally they all cost money, which can be avoided by using a free database system like MySQL. Even if the university has an Oracle server which might be used for the project, most of the people who might have use of this project do not have one or the money to afford one. Oracle, Informix or DB2 might fit in bigger, more database related projects, instead this project will use the database as an easier way to access a very small amount of data.

2.12 Web server

This project will use a webserver to host the chosen Software Project Management tool (see 2.10) and an easy to use configuration tool for the database. Therefore a webserver needs to be found where both are easy to install. The other restriction is that the whole project should run on a Linux system (see 2.5) and therefore webserver like the Internet Information Services (IIS) from Microsoft are ineligible. To accomplish these restrictions a widely used web server will be used for this project.

2.12.1 Apache

The apache web server is one of the most spread web servers of the world. It runs on Microsoft Windows and Linux. Due to the modular design the apache is very flexible if it comes to showing content. Several projects have added modules to have an easy way to publish their information through the apache or provide special commands that do not belong to a web server. It is capable of running PHP (PHP Hypertext Preprocessor) and various other languages which are used to display web

pages. PHP is a language that runs on the server side only. Before a page shows its results the PHP file will create it and only the HTML output will be sent to the client. This gives the developer the ability to dynamically create web pages. PHP also provides different template engines, like Smarty(<http://smarty.php.net/>), which lets the web interface work much smoother and separates the programming logic from the design of the page.

For both systems, `trac` and the database configuration tool `phpMyAdmin` are easy to follow apache installation instructions available.

2.13 Inter Process Communication

Inter process communication (IPC) is used to transfer data between two or more processes. Those communications will need a common way so the complexity and the clearness of the project will consist. IPC is not only limited to communications on one computer it could be possible that the communications might be necessary to be done over a network to let processes on different machines work with each other. This section will take a closer look upon some of the implementations.

2.13.1 D-Bus

D-Bus is part of the <http://freedesktop.org> project, and provides a system which lets applications talk to each other. This communication happens through channels. Each application that is connected to the D-Bus server has its own channel. Therefore multiple applications can wait for information that is sent by only one application. Not only information can be sent through the D-Bus system, but also commands like starting a specific part of the connected application. D-Bus also provides a communication through TCP network as well as remote procedure call (RPC) functionalities. The latter functions are very useful but they are only planned for further releases of the library, at the moment of writing it has only a rudimentary implementation of network usage. It is not possible to do a good authentication, nor are the functions officially supported or documented. Informations were taken from freedesktop.org (2008a) and Trolltech (2007).

2.13.2 DCOP

DCOP, the Desktop Communication Protocol, was intensely used by the K Desktop Environment (KDE). Every KDE application that used the DCOP mechanism had to register at a centralised DCOP server, now every other application or script which liked to use a function or pass data to a registered application just sent its

request to the server. The server then would send the request to the application and the application would react or send an answer through the server to the requesting application. Though DCOP worked well, KDE choose to take D-Bus as IPC for the upcoming KDE 4.0. Therefore DCOP is obsolete and should no longer be used.

2.13.3 Socket

Communication through sockets is an old way to exchange information. It works like opening a socket for network communication, but in this case the data will not be sent through the network it will instead be fetched by a listening application.

2.13.4 Conclusion

D-Bus and the plain socket communication are both looking very promising, whilst the DCOP is going to be obsolete in the near future. Plain socket communication has the advantage that the project might choose its own way for the communication and the protocols. D-Bus on the other hand will have the advantage of providing an interface that is easy usable by other applications. The best choice for the project would be an IPC mechanism based on the D-Bus only, but the lack of a proper network support makes it necessary that the final solution will have both, a D-Bus implementation for local usage of the daemons and an implementation based on SSL encrypted communication. As soon as D-Bus will support a fully network based IPC the system can be switched to D-Bus only.

2.13.5 A detailed look on how D-Bus works

The following information was taken from Trolltech (2007), the original D-Bus homepage (freedesktop.org, 2008a) and the D-Bus Specification (freedesktop.org, 2008b).

There are two different ways, applications can make use of the D-Bus IPC. The first is building up a connection through the D-Bus server, called the bus and the other way is by direct communication to the application. The central server allows programs from different usage spaces to talk to each other. This capability is used when hardware driver (located in the kernel space) would like to inform the system in most cases the Desktop Environment (located in the user space), if for example new hardware was plugged in or something else has happened (like a CD-ROM was inserted into the CD-ROM drive). With the centralised server technique the kernel-space driver informs D-Bus about the change, then all applications which are registered to that particular part of the bus are getting informed about what has changed and can react on it. The bus system is built to provide a many to many

connection between the applications.

The bus consists of two different sub buses the system bus and the session bus. The system bus is storing all information that depends on the system and the session bus stores information regarding the different sessions of the different users. Therefore global happenings affecting the system, like changing the CD in a CD-ROM drive will inform services attached to the system bus, while happenings in a browser which only affect the current user will be stored in the session bus.

Each service that is offered through the D-Bus needs to be registered through a service name. The service name is all lowercase and looks like a reversed domain name similar to imports in Java (freedesktop.org, 2008b). Each service provides several objects. Those objects look like paths from within an URL. Each object will implement an interface which again is dot separated. Example (taken from the D-Bus specification (freedesktop.org, 2008b)):

A service guarantees a service with the following name *org.freedesktop.Screensaver*, it has a singleton object */org/freedesktop/Application* and the object implements an interface called *org.freedesktop.ScreensaverControl*. Through this interface the system and the user can control the screensaver.

2.14 Similar projects

It is always good to know which similar projects are competitiving with the planned application, what features the other projects cover and where their weaknesses are. This section therefore will cover a few projects and ideas regarding location detection.

2.14.1 Location-based media

The locationbased media (LBM) technology is mostly combined with the Global Positioning System (GPS). Therefore it is only possible to retrieve media from a device that features both a GPS receiver and the capability of media playback. Each device then has to get the software for that specific location and will then be able to playback the media if it got in reach of a specific point.

2.14.2 Magic Map

The Magic Map project (<http://www2.informatik.hu-berlin.de/rok/MagicMap/>) was developed at the Humboldt-Universität zu Berlin (<http://www.hu-berlin.de>).

The Magic Map application uses the WLAN technology to detect the persons position.

2.14.3 Conclusion

This project will have more in common with the Magic Map project than with the LBM, because it has the ability that applications can ask a server where a specific person is located, which is not possible if it uses GPS.

2.15 Project surroundings

As a fast growing project it needs to be backuped to several positions to prevent a huge data loss. For backup issues regarding the source code a SVN repository will be set up on a different machine additional the source and the documents will be saved on local storage devices like USB sticks or different locations on the hard drive. The documents will also get stored on a different machine (through secure copy (scp)) every time a big change was made to them. Backing up the source code in a SVN repository has also the advantage of following the history of the source code and if necessary step a few steps back in development.

To administer the source and all the upcoming work with it a trac project management system will be installed on the same machine the SVN is running.

The operating system that will be used for all the work is Linux and the used distribution will be gentoo (<http://www.gentoo.org/>). Gentoo is a distribution where everything possible is built from source and therefore every library that is installed has also its developer packages installed. Other distributions like debian (<http://www.debian.org/>) or kubuntu (<http://www.kubuntu.org/>) are using special developer packages to install needed include files. Also most of the building environment is already installed on the developing machine, because it was used before. The tools that will be used by this project are CMake and gcc.

For creating all those documents the KDE integrated latex editor (Kile) will be used. The projects Integrated Development Environment (IDE) will be covered by a single editor called KDE advanced text editor (kate). The graphics are done with dia, gimp, karbon, krita and Inkscape while the UML Diagrams are done with Umbrello.

3 Analysis

3.1 Research Conclusions

- **Methodology**

This project will make use of the Waterfall Model but with some changes regarding the documentation and some changes to some stages. In short there will be: no feasibility study, a prototype for hardware communication during the analysis, a few test classes during the design stage needed to be written to determine how some libraries work, a module documentation. The testing will be done after the full implementation is done. The project report will be written along with the project in parallel to the stages.

- **Chosen Hardware**

The chosen technology is RFID based. The hardware that will be used are three OpenBeacon nodes and two CCC Sputnik tags. Three nodes to trilaterate a position and two tags so that collision can be tested.

- **Operating system**

The main operating system for this project will be Linux. It will be used by the developer to develop the project and the project itself will only be tested on Linux.

- **License**

The whole code will be licensed under the GPL.

- **Programming language and libraries**

The primary programming language will be C++ in combination with the Qt library. The Qt library will support C++ on some core features like networking and string handling as well as on the GUI. For the web-interfaces PHP will be used.

- **Code version control system**

To enable the ability for reverting code and have a save position if anything happens to the local code, the project will also be stored on a SVN server.

- **Build environment**

For a quicker way to build the modules of each project CMake will be used. CMake supports the building process with automatic search for libraries, and linking them if needed to the binary.

- **Software Project Management System**

Trac will be used for the project management. It provides easy web access to the subversion repository and has also a system, which supports some basic project management like a ticket system for the bug tracking. It also has a system for displaying the current state of the project and upcoming milestones and other functionalities to support the development.

- **Database**

The database which will be used to support some modules of this project will be the MySQL database. It is well integrated in PHP, and for open source projects there is no fee to pay.

- **IPC**

The IPC this project is using is mainly the D-Bus system. The network communication interfaces will be developed with a SSL (Secure Sockets Layer) based socket implementation, because the D-Bus support for network based IPC is still in development and does not work very well. It also lacks in security, so for the network communication a socket based SSL connection will be used.

3.2 Project overview

The system needs to be split up into three layers (see figure 3.1).

- The hardware layer:

This layer is communicating with the hardware.

- The position detection layer:

This layer does all the objectives regarding the position detection.

- The application layer:

This layer holds all the end user applications that are using the positions of the tags.

The advantages of this three layer system are that on one hand the different logics of the system are divided from each other and on the other hand each layer can be

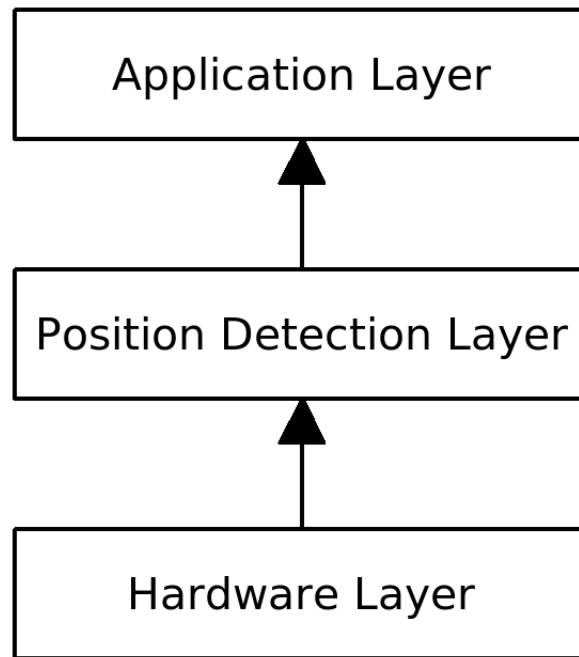


Figure 3.1: Layers of the system

expanded with functionalities on its own. The hardware layer could be extended to support different types of hardware. The position layer could be extended by a calculation for a 3D position detection. And on the application layer a lot of different applications can be implemented to support a wide set of uses for the position detection system.

3.3 General system test plan

This section will not provide the detailed description about the single tests that have to be done. But it will describe what tests in general shall be done and how they shall be accomplished.

- Each layer shall be tested for its own, if it is working as it is meant to be.
- Each layer shall be checked upon the data it provides and if this data is appropriate.
- The data-flow between the layers shall be watched if there is some inconsistency.
- Data stored persistent shall be checked if it is stored in a proper way and not malformed.

- Every application with user input, shall be tested for the correctness of the error messages if the input is not as expected.
- If input derives from the Hardware it shall be checked if it is correct.
- Testing the whole system with all its applications and hardware.

4 Design

The following sections will cover the modularisation and the modules themselves. Each module will be described in detail which data it takes and which it will provide as well as the functionalities of it. Additional to the modules the specification will also give information on the different data that will be created and stored persistent.

The whole project will be based on packet loss, but it should be designed to use other methods as well like the signal strength to perform a position detection.

The UML Diagrams that are shown here are reduced to the important functions. This is because the Qt framework needs a lot of small functions that are not doing much but are needed to react on actions like hitting a button or clicking on a menu item.

All modules that are marked as optional are implemented if there is some spare time or other modules can be done faster than planned.

4.1 Modularisation

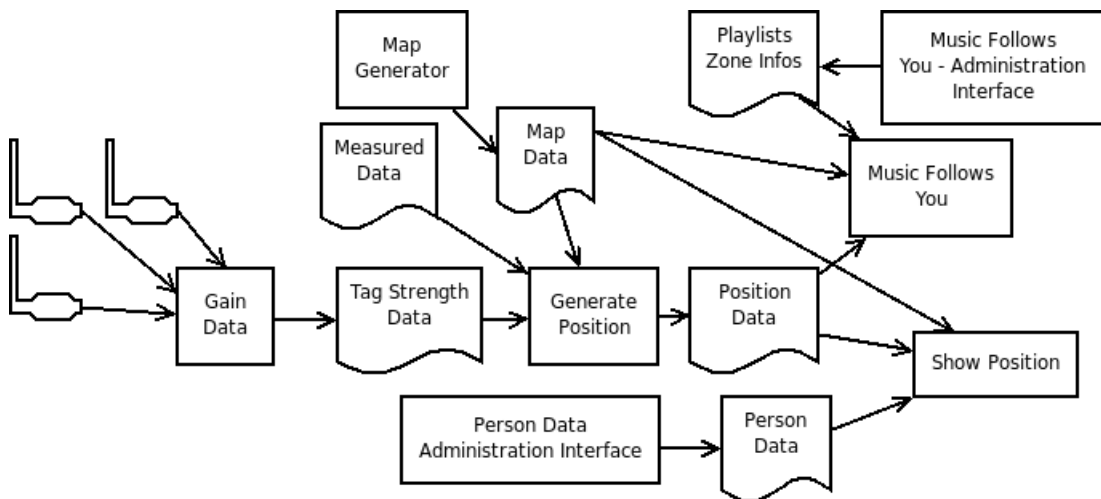


Figure 4.1: Project modularisation and flow of data

Like it is shown in figure 4.1 the project will be separated into several functional modules. Those modules will be described in the following sections. Additional to the shown modules the following will also cover the changes that needed to be made

to the firmware. The sections will also cover the data that will be stored persistent (Measurement, playlists, zone information, map and person data).

4.2 Firmware

The firmware of the OpenBeacon nodes needs some little adjustments in its functionalities. The following list will show which functionalities are already implemented and which shall be implemented additional to the existent.

4.2.1 Current functionalities

1. Reading the whole configuration at once.
2. Display a help screen. A functionality which is good if a user wants to interact but useless if a program wants to interact automatically with the node.
3. Setting the id of the reader. This setting is used to differ between multiple nodes attached to one computer.
4. Setting the cache (FIFO) size in seconds, the larger the cache size is the more packages will be received and the more accurate the node will get. But if the time for the cache is too high even small disturbances, like a small object, will have a big effect on the accuracy.
5. Setting the node to receive only, in which it will not send any packages to the CCC Sputnik RFID tag.
6. Setting the transmission power level of the OpenBeacon node to one of the 4 levels (1=0x00, 2=0x55, 3=0xAA, 4=0xFF).
7. Saving all changes that were made.

4.2.2 Needed functionalities

1. Reading only the id of the node. This functionality will lower the effort which is needed to parse for the id if the whole configuration will be read, it will also lower the amount of data that will be received. This configuration information is needed to identify each of the nodes connected to the computer
2. Reading only the FIFO cache lifetime is not directly needed by the project but it will give the node a consistent interface.

3. Reading the current power level on which the node is working. (0=receive only mode, 1=0x00, 2=0x55, 3=0xAA, 4=0xFF)
4. Reading only the uptime of the node will be used for statistics only.
5. Reading only the channel of the node. (Again this is not used directly by the project but will keep the interface consistent)

4.3 OpenBeacon Configurator

The OpenBeacon Configurator is a simple GUI, which will be used to configure the OpenBeacon nodes as easy as possible.

4.3.1 Functionalities

- Configuration of OpenBeacon nodes.
- Read information from OpenBeacon node.
- Write information to OpenBeacon node.
- Persistent saving of the information on the OpenBeacon node.

4.3.2 UML Class Diagram

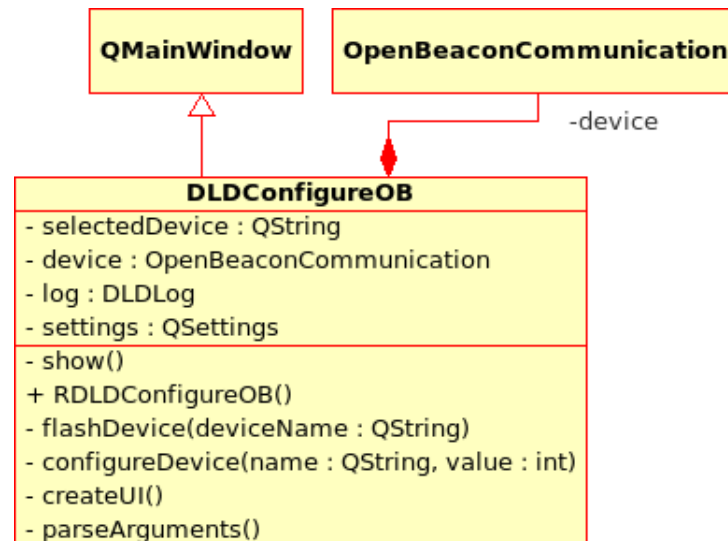


Figure 4.2: UML class diagram of the OpenBeacon Configurator

4.4 Gain data Daemon

The gain data daemon communicates with all connected OpenBeacon nodes, it has a configuration file which describes each of the connected nodes. This description is not based on their device nodes in the system, but on their configured id's because the devices may change when they are plugged into the computer in a different order than the first time. The information which is gained will be provided to interested parties through a D-Bus or SSL connection. The provided strength is calculated with the following formula: $\text{strength} = \frac{\text{power level}}{85} * 10 + (\text{maximum packages} - \text{packet loss})$

A gain data daemon has the ability to watch over more than one OpenBeacon node at once.

4.4.1 Functionalities

- Read information from several OpenBeacon nodes.
- Providing the strength data of several tags.
- Provide which maximum packet loss is set.

4.4.2 Incoming data

- The incoming data to the gain data daemon consists of the raw data (for more information see listing 2.5) of each of the OpenBeacon nodes.
- The settings of the several OpenBeacon nodes.
- The configuration of the data gain daemon.

4.4.3 Outgoing data

- One package for each update including the tag id, its strengths. (tagId|nodeId1=strength1|nodeId2=strength2|...)
- A list of all tags that were in reach of the node(s)
- A list of all the connected node(s)

4.4.4 UML Class Diagram

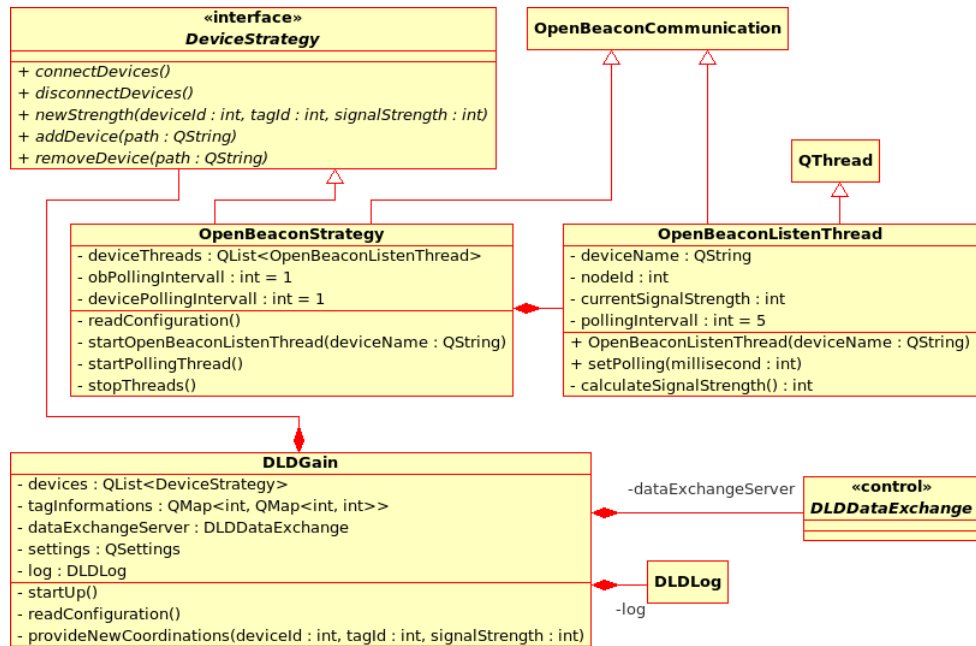


Figure 4.3: UML class diagram of the Gain Data Daemon

4.5 Measured data

The measured data is a fixed data block. The data that will be stored is based on a measurement that will be done by the developer itself and will cover a table of metric ranges compared to the packet loss or signal strength of the hardware. This data will be used by the generate position daemon described in section 4.7. The data has to be gained once during the whole use of the project and it only will be changed if the project shows some failures in measurement or if the surrounding environment will change dramatically so the data does not correspond any more.

4.5.1 Functionalities

The measured data has no functionality for itself, it will only be used by other daemons to do a proper position detection.

4.5.2 Incoming data

- packet data from data gain daemon (see section 4.4)
- metric data from the person who will do the measurement

4.5.3 Outgoing data

A table which will compare packet loss and power level to metric units. Each row of the table has to store the following data and each table needs to store the maximum of packets that could be received.

- power level
- packets received
- approximately distance in centimetre

4.6 Optional application - Map generation application

This application will be used to create map files. Those map files will be used for displaying the position of a person (see section 4.9) as well as at the daemon that will generate the position data (see section 4.7). The map data will consist of two files, a picture which will show the map itself, and a description file which will hold information upon the map. This information contains the zones and their names, the size of the map, a meter pixel correspondent, the name of the map. Zones are special regions on the map which can be used by special applications.

4.6.1 Functionalities

- Open favourite drawing tool for map drawing
- Show picture of map and create information regarding this map
- Define different zones through polygons that are built with the help of the image file
- Save map data in a map package (see 4.12.4)
- Load data from a map package
- Change data from a map package

4.6.2 Incoming data

- map image from painting tool
- information given by the user

4.6.3 Outgoing data

- map package (see 4.12.4)

4.6.4 UML Class Diagram

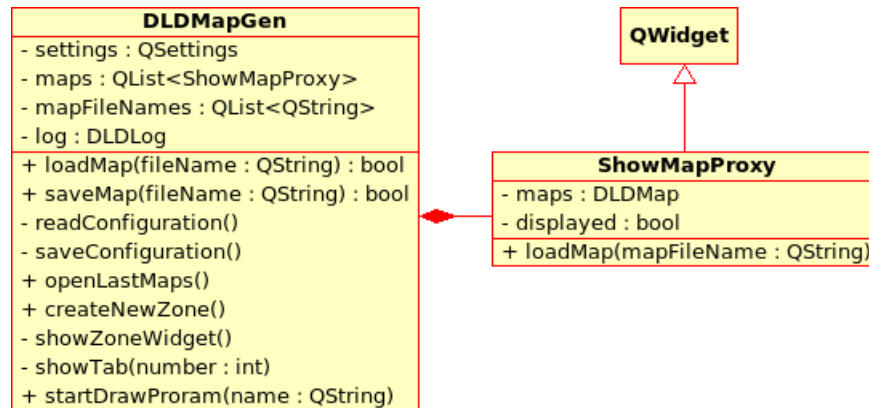


Figure 4.4: UML class diagram of the Map generation application

4.7 Generate position daemon

The generate position daemon takes the information provided by the data gain daemon (see section 4.4) and creates the position data by triangulation like described in section 2.3.

The location detection will be based on the power level and the packet loss of the arriving packages. Three packages, each from a different node are needed to determine a position of a tag. The gained position will then be provided by sending it to a specific position. The position for each tag will be in packet loss not in any standard metric units.

The gained position then will be either provided in a raw format or with the help of the measured data it will calculate an absolute position regarding a map (see 4.12.4).

4.7.1 Functionalities

- Check if all nodes are using the same packet loss parameters.
- Performing a mathematical position detection based on the strength.
- provide data in relative position
- provide data in absolute map position

- provide zone information of the tags

4.7.2 Incoming data

- strength data from the gain daemon (see outgoing data from section 4.4)
- map package (see 4.12.4)
- measured data (see section 4.5)

4.7.3 Outgoing data

- relative position data for each tag id
- absolute position data for each tag id
- zone information for each tag id

4.7.4 UML Class Diagram

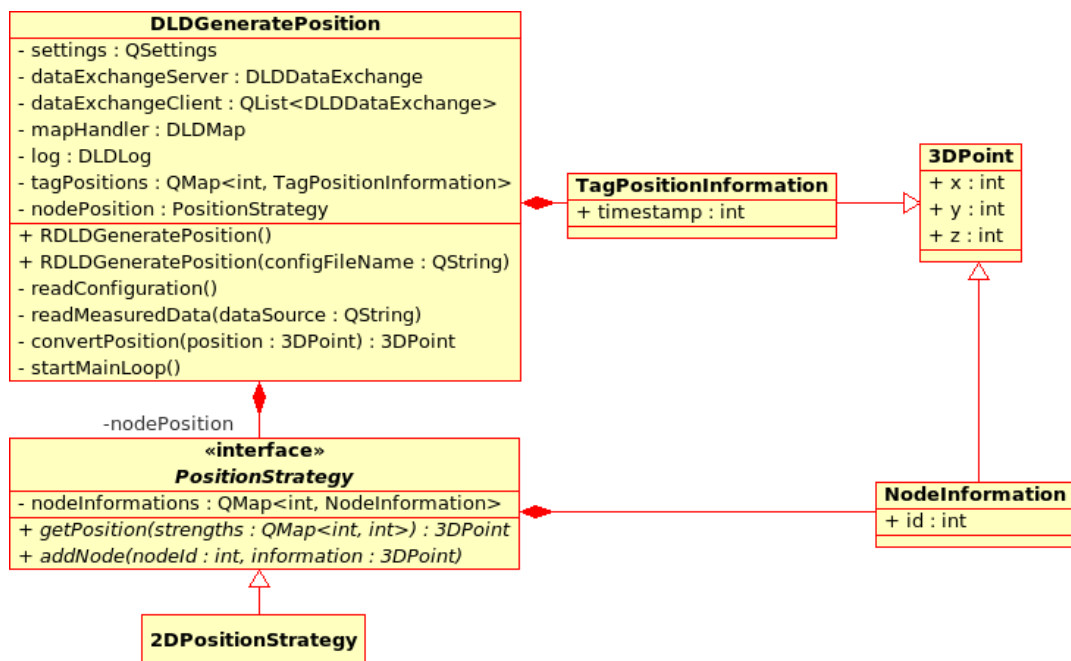


Figure 4.5: UML class diagram of the generate position daemon

4.8 Person data administration interface

This application will be used to easily add information data regarding each person on the system. It will store basic information as well as the id of the tag the person is using.

This application should be as flexible as possible and should not be fixed on persons. This will offer the possibility to use this application for objects as well.

4.8.1 Functionalities

- Change the currently used database
- Create new entries in the database
- Change entries in the database
- Delete entries from database
- Show entries in database

4.8.2 Incoming data

- Information from administrator regarding the person

4.8.3 Outgoing data

- Persistent data stored in database.

4.8.4 UML Class Diagram

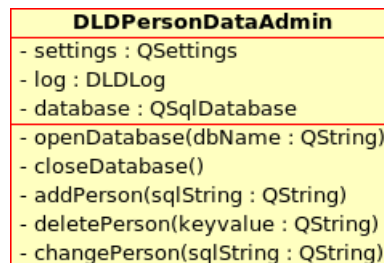


Figure 4.6: UML class diagram of the Person data administration interface

4.9 Show position of a person

This application will show the position of a tag on a map. The tag id will be assigned to a person. The information to a person will be collected from a person database where needed information upon the person will be stored. This application should be as flexible as possible, therefore the information that will be displayed should not be fixed but should be variable. This will enable the application to be used for a different set of uses.

4.9.1 Functionalities

- Select data source for assignment
- Assign tag id to person
- Display the position of a person on a map
- Select persons on the map
- Display information of visible persons

4.9.2 Optional functionalities

- Display short information in a fancy way, like a box next to the person that moves with it
- Mouse over selection of a person
- Display more than one map at once (in a tab view)

4.9.3 Incoming data

- Map package (see 4.12.4)
- Tag position data from generate position daemon (see section 4.7)
- Person information (see section 4.8)

4.9.4 Outgoing data

- Display of the position in a GUI
- Display information of a person in the GUI

4.9.5 UML Class Diagram

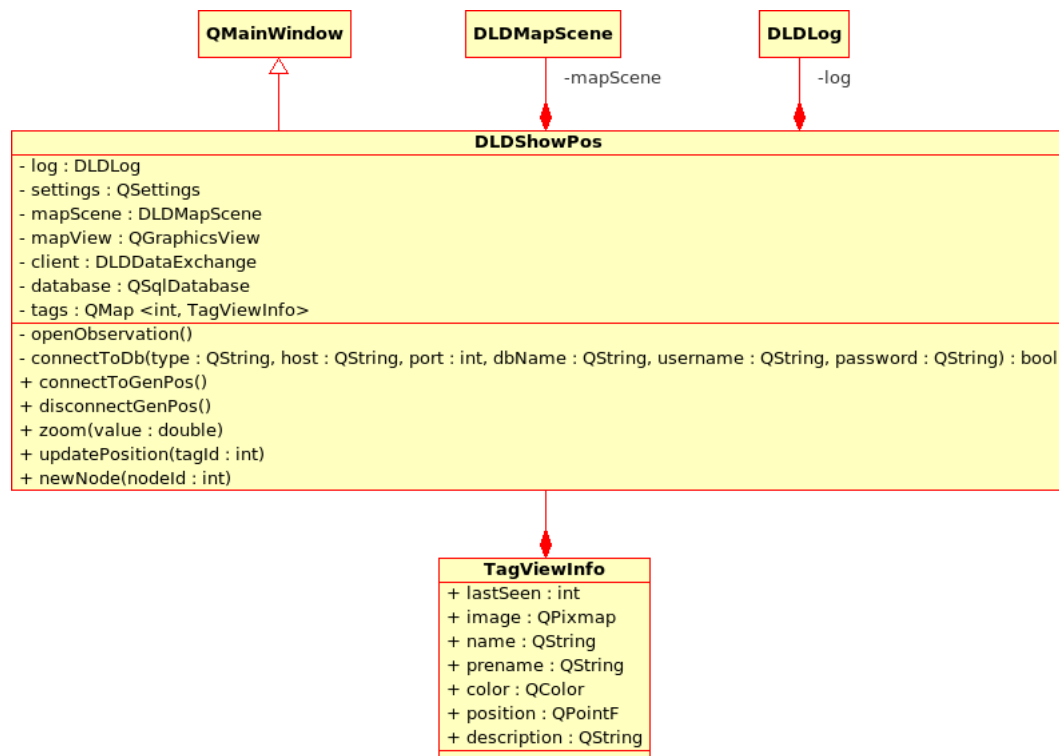


Figure 4.7: UML class diagram of the Show position of a person application

4.10 Optional application - Music follows you administration interface

This interface will be used to administrate each user of the music follows you (MFU) system. Each user has a personal login to the system and has the ability to change its zones and playlists. The administrator has the abilities to create, change, delete users and their settings.

4.10.1 Functionalities

1. Administrator

- Administrate the paths from which the users can choose their songs.
- Administrate user (create, change, delete).
- Administrate zones (create, change, delete).
- Giving zone priorities to users.
- All options a user is having for itself, but the administrator has the ability to change those for each user separately

2. User

- Administrate playlists (create, change, delete).
- Administrate zones (create, change, delete).
- Select current play-mode.
- Select current play-list.

4.10.2 Incoming data

- User name and information by the administrator
- Zone information by the administrator
- Play-list information by the user
- Zone information by the user
- Current selections by the user

4.10.3 Outgoing data

- Persistent stored data in a database with all the necessary information regarding the user

4.10.4 UML Class Diagram

There is no further planning for this module done, until the project has some spare time to actually do the optional tasks.

4.11 Optional application - Music follows you

The music follows you (MFU) application will provide another use of a domestic location detection which can be used to enhance the daily life. Its basic function is to let music follow a user through an environment. It will also provide the functionality to play music if a user enters a specific zone.

An example for the first case is an audio stream that follows a person through a flat. In detail this means if a person leaves room A and enters room B the server will recognise this and then will fade out the stream to room A and start to stream to room B. To use this properly the flat has to provide an infrastructure for this, like a computer or other device that is capable of receiving an audio stream in every room/zone where the person wants to hear music.

An example for the second case can be a museum with automated audio commentaries on the different exhibits. Each exhibit will get its own zone and as soon as a person or a group enters the zone an audio commentary will be played. A system like this would enable a museum to provide more than just a tour in a few languages. The person or group just have to register at entrance say which language is wanted and then walk on with a tag.

4.11.1 Functionalities

- Stream different playlists to different positions
- Prioritise if two tags are in one zone
- Provide different modes (music follows you, one play-list per zone)

4.11.2 Incoming data

- map package containing the zone information (see section 4.6)
- position data (see section 4.7)

4.11.3 Outgoing data

- several sound streams to several zones

4.11.4 UML Class Diagram

There is no further planning for this module done, until the project has some spare time to actually do the optional tasks.

4.12 Common classes

This section deals with classes that are used by several of the above mentioned classes.

4.12.1 OpenBeacon Node Communication Class

This class, which will be used by the “gain data daemon”(see 4.4) as well as the “configure OpenBeacon application” (see 4.3), provides methods that will support an easy way of communication with the OpenBeacon nodes

UML Class Diagram

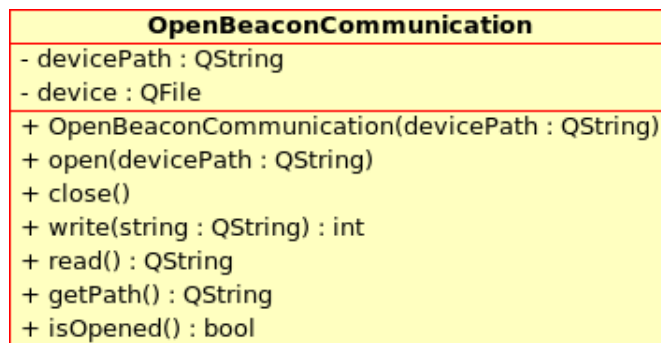


Figure 4.8: UML class diagram of the OpenBeacon Node Communication Class

Functionalities

This class provides all basic functionalities a device needs: open, close, read and write.

4.12.2 Data Exchange Class

This class provides methods for the interprocess communication. This can be achieved by using the D-Bus or the SSL strategy. If the class is used as a server then both might be used to provide data for the clients, if it is used as a client then only one strategy at a time is allowed to be used (this will prevent information collisions).

The SSL version of the data exchange class is an optional part. At first the project has to run with D-Bus and afterwards the SSL strategy implementation should be added (if the time permits it).

UML Class Diagram

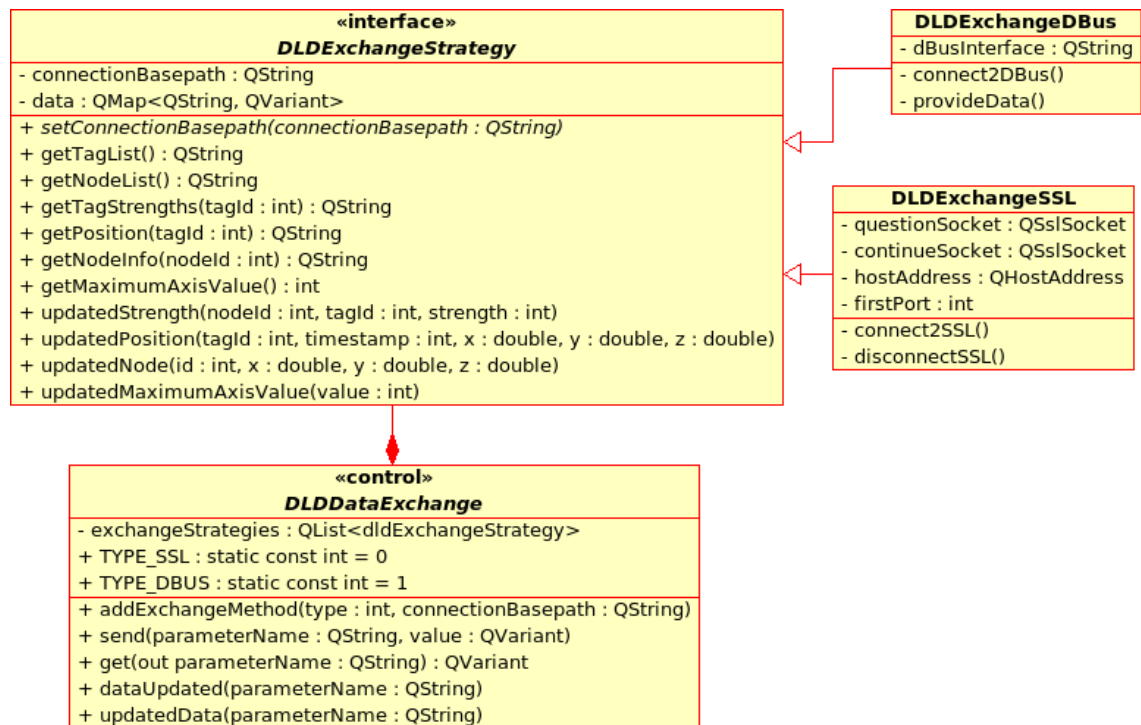


Figure 4.9: UML class diagram of the Data Exchange Class

Functionalities

- build up SSL connection to a server or client
- create a D-Bus connection
- send data to one of the connections (or to both if it is a server)
- receive data from the connections
- store all send/received data in a map

4.12.3 Log Class

The log class provides a set of methods to offer an easy logging facility. The log class will be used by every daemon or application.

UML Class Diagram

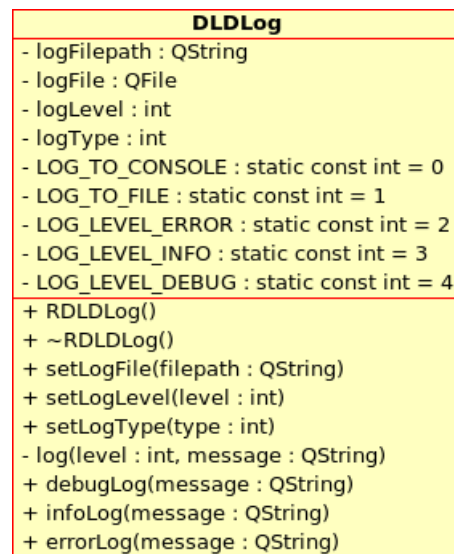


Figure 4.10: UML class diagram of the Log Class

Functionalities

- Log to console or file
- Different log levels
- Different files for each log levels

4.12.4 Optional - Map Classes

The map classes provide methods to handle the full map package and a map scene. A map package consists of a map description file and an image of the drawn map, both compressed into one file. The Map class will be used by the “generate position daemon” (see section 4.7), the “map scene” class and any other class that needs the information of a map for saving or reading. The map scene class is a derivation of the map package class and enhances it by some scene specific features. Those features can be used to display the map in applications like the “map generation application” (see section 4.6) or the “show position of person” application (see section 4.9).

UML Class Diagram - map package

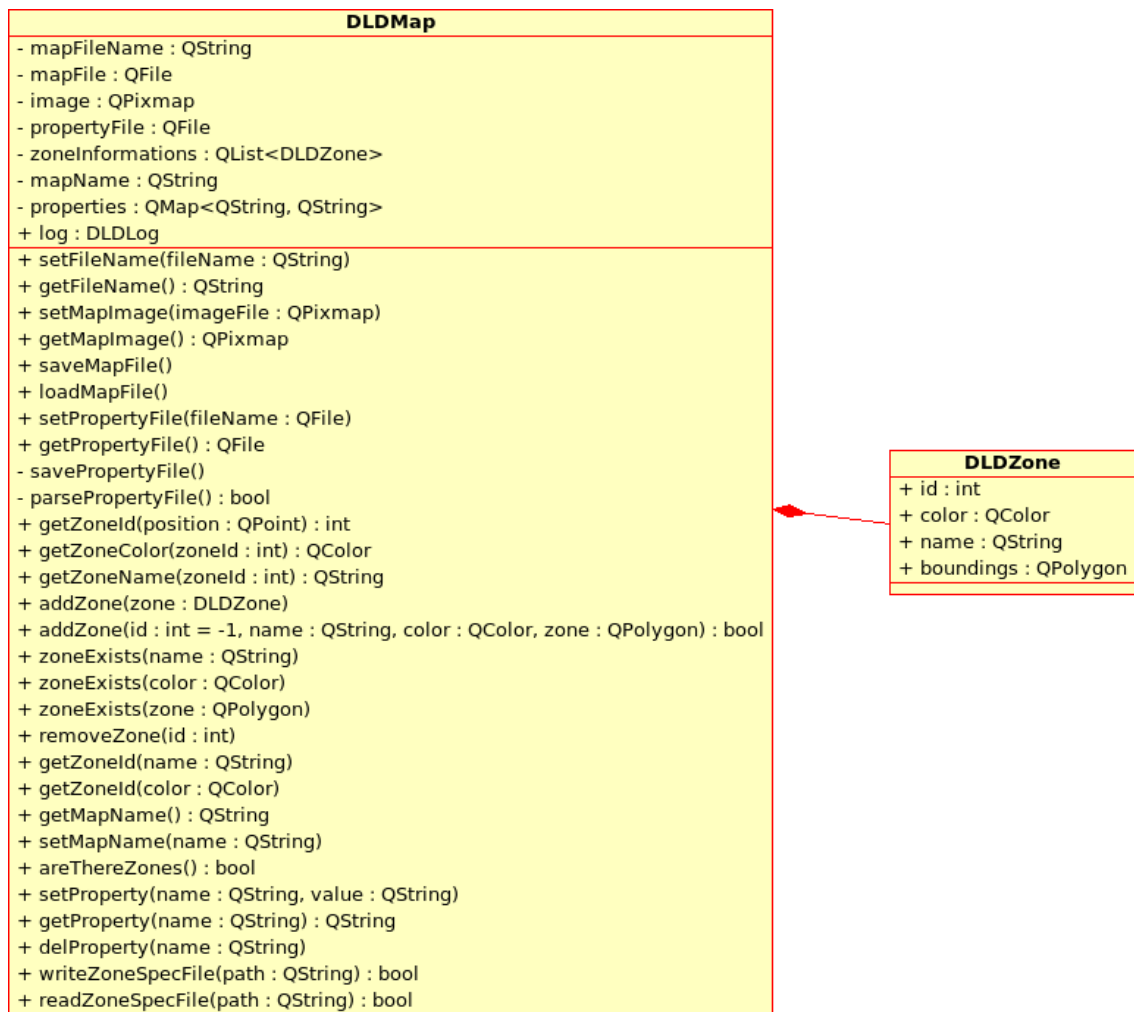


Figure 4.11: UML class diagram of the Map Class

Functionalities - map package

- add map image
- get map image
- add property file
- get property file
- handling of the properties
- open package
- save as package
- handling of the zones if there are any included

UML Class Diagram - map scene

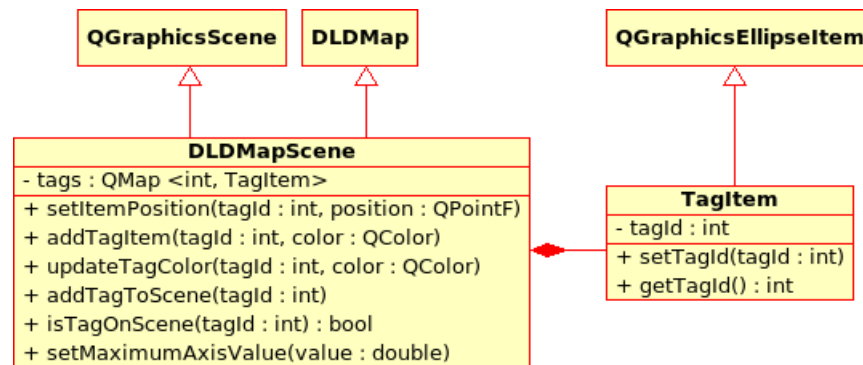


Figure 4.12: UML class diagram of the Map Scene Class

Functionalities - map scene

- add tag to a scene
- check if tag is on the scene
- set the background image
- create a coordinate system
- update the position of a tag

4.13 Network protocol for SSL connections

This section will cover the protocol and the mechanism that will be used to exchange information between the different daemons through an SSL connection. The daemons that will provide data are the “data gain daemon” (see section 4.4) and the “generate position daemon” (see section 4.7). Both provide several information regarding the positions of the tags, or persons who are wearing them, therefore the data has to be encrypted so not everyone in the net is capable of retrieving that information. The data the gain daemons provides is only interesting for the generate position daemon. The data the generate position daemon is providing on the other hand is for every application that wants to use the location data.

To keep it as simple as possible both daemons will have the same mechanisms and methods for communicating to the connected applications. Because an asynchronous communication through TCP is not possible, the SSL communication part will use two connections. One connection is used to react on client requests and the other one is used to send regular updates in a specified interval. It should be possible to dynamically specify if both, one or none of the connections should be build up.

Regular updates protocol

1. send a raw tag strength string every N seconds. (raw tag strength string: “tagId|raw|nodeId1=strength1|nodeId2=strength2|...”)
2. send a map tag strength string every N seconds. (map tag strength string: “tagId|map|nodeId1=strength1|nodeId2=strength2|...”)
3. send a zoneUpdate if one occurs. (“tagId|newZone”)

The application which sends those strings should be capable of disabling one of the sent strength strings, so that only the needed ones are being sent. Only the generate position daemon should be able to send both, but only if it is configured that way.

Reaction protocol

1. **getTagList** () will return a tag-list (tagId|tagId|...).
2. **getTagInfo** (tagId, posMode) returns the tag strength string (“tagId|nodeId1=strength1|nodeId2=strength2|...”) to the asked position mode.
3. **getTagStatus** (tagId) will return the status (Gone or Seen) of the tag.
4. **getNodeList** () returns a list of all attached nodes
5. **getNodeInfo** (nodeId) returns the position information of a node
6. **getMaximumAxisValue** () returns the maximum value of an axis in one direction
7. **getZone** (tagId) returns the current zone of the tag

4.14 D-Bus Interface specification

This section covers the D-Bus interface specification for the gain data and generate position daemons.

4.14.1 gain data interface

- **Service name:** org.dld.gain
Service for the data gain daemon.

1. **Method:** getTagList ()
returns a tag list of all seen tags so far (“tagId1|tagId2|...”)
 2. **Method:** getNodeList ()
returns a node list of all attached nodes (“nodeId1|nodeId2|...”)
 3. **Method:** getTagStrengths (tagId)
returns a strength string for the requested tag (“nodeId1=strength1|nodeId2=strength2|...”)
 4. **Method:** getNodeInfo (nodeId)
returns the coordinations of the node (“x=value|y=value|z=value”)
 5. **Method:** getMaximumAxisValue ()
returns the maximum value of an axis in one direction
 6. **Signal:** updatedStrength (nodeId, tagId, strength)
every time a node has a new strength available it gets sent through this signal
 7. **Signal:** updatedNode (id, x, y, z)
if a new node was found its data will be sent through this signal
 8. **Signal:** updatedMaximumAxisValue (value)
if the maximum axis value is changed then this signal will be emitted
- **Service name:** org.dld.genPos
Service for the generate position daemon.
 1. **Method:** getTagList ()
see above
 2. **Method:** getPosition (tagId)
returns the calculated position of the tagId
 3. **Method:** getNodeList()
see above
 4. **Method:** getNodeInfo (nodeId)
see above
 5. **Method:** getMaximumAxisValue ()
see above
 6. **Method:** getZone (tagId)
returns the zone in which the tag is currently located
 7. **Signal:** updatedPosition (tagId, timestamp, x, y, z)
every time the daemon has calculated a new valid position for a tag this signal will be emitted

8. **Signal:** updatedNode (id, x, y, z)
see above
9. **Signal:** updatedMaximumAxisValue (value)
see above
10. **Signal:** zoneChanged (tagId, newZone)
this signal is emitted as soon as a tag changes from one zone to another

5 Implementation

5.1 Installation and Configuration of the surroundings

5.1.1 Installation

To install the Software project management system (see section 2.10) with Subversion (see section 2.8) connection, the following packages need to get installed on the server:

- apache
- subversion
- trac
- phpMyAdmin

5.1.2 Configuration

The following sections cover the basic configurations for the project surroundings. The base name for the system is dld (domestic location detection) and therefore it will be used for the subversion repository and trac.

Subversion

To use a subversion code version control system with this project, the project needs a repository. The command to do this is:

```
svnadmin create /var/svn/dld
```

After the repository is created the permissions have to be set such that the apache webserver gets access to the repository as well. Therefore we create a new group called **svnusers**, adding the apache to this group and changing the permissions of the repository directory:

1. groupadd svnusers
2. usermod -G svnusers -a apache

3. `chgrp svnusers /var/svn/dld -R`
4. `chmod g+w /var/svn/dld -R`

trac

To configure trac the base system needs to be installed to the directory where the web pages are stored. On gentoo this is done by the following command:

```
webapp-config -I -d dld/trac trac 0.10.3.1
```

Assuming that 0.10.3.1 is the current version of the trac package, if not then it needs to be replaced by the current version. To initially configure trac the administrator just has to use the administration tool that comes with the trac package. The following shows how to execute it:

```
trac-admin /var/lib/trac/dld initenv
```

The program then asks for a few details and afterwards it installs the files to the directory that was given to the `webapp-config` program.

After the basic configuration is done the trac system needs to be cleaned from previous made example entries. In addition to that, a few changes need to be done to the permissions of the anonymous user and a whole new user(in this case the developer) should be added who has the right to change everything on the system. To enter the interactive control console for trac the administrator needs to execute the following command:

```
trac-admin /var/lib/trac/dld
```

```

1 milestone remove milestone1
2 milestone remove milestone2
3 milestone remove milestone3
4 milestone remove milestone4
5 milestone remove milestone1
6 milestone remove milestone2
7 milestone remove milestone3
8 milestone remove milestone4
9 version remove 1.0
10 version remove 2.0
11 component remove component1
12 component remove component2
13 permission remove anonymous WIKIMODIFY WIKICREATE
14 permission add twist BROWSER_VIEW CHANGESET_VIEW CONFIG_VIEW FILE_VIEW
   LOG_VIEW MILESTONEADMIN MILESTONECREATE MILESTONEDELETE
   MILESTONEMODIFY MILESTONEVIEW REPORTADMIN REPORTCREATE
   REPORTDELETE REPORTMODIFY REPORTSQLVIEW REPORTVIEW
   ROADMAPADMIN ROADMAPVIEW SEARCHVIEW TICKETADMIN TICKETAPPEND
   TICKETCHGPROP TICKETCREATE TICKETMODIFY TICKETVIEW

```

```
TIMELINE_VIEW TRAC.ADMIN WIKIADMIN WIKLCREATE WIKLDELETE
WIKLMODIFY WIKLVIEW
```

Listing 5.1: clean up command for the trac administration console

Listing 5.1 shows the commands that need to be entered to remove all unnecessary entries and to change the permissions. The user `twist` will be used by the developer as a nickname to the system.

Apache

First we create a password file that will be used for authorisation of the developers both through the trac interface and to authorise them to be able to commit changes to the repository.

```
htpasswd2 -c /etc/apache2/trac.htpasswd twist
```

After this is done the files `/etc/apache2/vhosts.d/00_default_vhost.conf` and `/etc/apache2/modules.d/99_trac.conf` needs to be adjusted. The `.../99_trac.conf` has to look like the one in listing 5.2 and the lines from listing 5.3 have to be added to the `<VirtualHost *:80>` section of the `.../00_default_vhost.conf` file.

```
1 ScriptAlias /dld/trac /var/www/localhost/cgi-bin/trac.cgi
2 # This location should match the ScriptAlias above, if you want trac
3 # to be hosted at http://localhost/projects instead of http://localhost
  /trac
4 # change both the ScriptAlias and the Location.
5 <Location "/dld/trac">
6     SetEnv TRACENV "/var/lib/trac/dld"
7     # Comment TRACENV above and uncomment TRACENV_PARENT_DIR
8     # below to enable one Trac installation to handle multiple
9     # Trac projects
10    #SetEnv TRACENV_PARENT_DIR "/var/lib/trac/"
11 </Location>
12 # Configure how Trac will authenticate users
13 <Location "/cgi-bin/trac.cgi/login">
14     # Using htpasswd for authentication
15     AuthType Basic
16     AuthName "trac"
17     # Create an htpasswd file for trac users and
18     # reference it here:
19     AuthUserFile /etc/apache2/trac.htpasswd
20     Require valid-user
21 </Location>
```

Listing 5.2: Adjustment of the file `/etc/apache2/modules.d/99_trac.conf`

```

1 # dld - trac
2     Alias /dld/dox /var/www/localhost/htdocs/dld/dox
3     <Location /dld/svn>
4         <IfDefine SVN>
5             DAV svn
6             SVNPath /var/svn/dld
7             SVNIndexXSLT /dld/trac/svnindex.xsl
8         </IfDefine>
9         <LimitExcept GET PROPFIND OPTIONS REPORT>
10             AuthType Basic
11             AuthName "DLD::svn"
12             AuthUserFile /etc/apache2/trac.htpasswd
13             Require valid-user
14         </LimitExcept>
15     </Location>
16     <Location /dld>
17         SetEnv TRACENV "/var/lib/trac/dld"
18     </Location>
19     # Configure how Trac will authenticate users
20     <Location /dld/login>
21         AuthType Basic
22         AuthName "DLD::trac"
23         AuthUserFile /etc/apache2/trac.htpasswd
24         Require valid-user
25     </Location>
26     Alias /dld/trac /var/www/localhost/htdocs/dld/trac
27     ScriptAlias /dld /var/www/localhost/cgi-bin/trac.cgi
28 # dld - trac - end

```

Listing 5.3: Adjustment of the file /etc/apache2/vhosts.d/00_default_vhost.conf

5.1.3 Adding the basic directory structure to the Subversion repository

For adding the basic structure it needs to be created first. To achieve this the project directories will look similar to the modules that were created back in section 4.1 all on top of the three main directories which are commonly used by subversion:

```

1 ./code/branches/
2 ./code/tags/
3 ./code/trunk/
4 ./code/trunk/dld/src/common
5 ./code/trunk/dld/src/gainData
6 ./code/trunk/dld/src/obConfig
7 ./code/trunk/dld/src/mapGen

```

```

8 | ./code/trunk/dld/src/generatePosition
9 | ./code/trunk/dld/src/showPos
10 | ./code/trunk/dld/src/pdAdmin
11 | ./code/trunk/dld/doc/

```

Listing 5.4: Tree structure of the project in the repository

The tree structure described in listing 5.4 needs to be created on the server and after creation it will be imported to the subversion repository with the following command (executed from the directory where the code directory is located, but the code directory will not be part of the repository):

```
svn import ./code/ file:///var/svn/dld/
```

After that the code can be fetched from anywhere (with the anonymous account) with the following command:

```
svn checkout http://sirtwist.homeip.net/dld/svn
```

Committing source is restricted to the developer therefore the developer needs an other command line to firstly fetch the source code:

```
svn checkout http://twist@sirtwist.homeip.net/dld/svn
```

Both the developer fetch of the source code and the anonymous one can both stay up to date with the

```
svn update
command.
```

5.1.4 Database and phpMyAdmin

Like written in the webserver (2.12) and the database (2.11) section of the Research stage, this project will make use of the apache web server and the mysql database. The database will be administrated with the phpMyAdmin (<http://www.phpmyadmin.net/>) interface. The basic installation on modern Linux systems is fairly easy, the packages just needed to be installed through the distribution package system. If the distribution does not provide those packages then the sources can be downloaded and compiled by the developer himself.

After installing all three components (the apache, the mysql database and the phpMyAdmin) the apache is running out of the box with a default start page. The configuration files of the apache are located in `/etc/apache2`. The DocumentRoot entry of the configuration file (e.g. `/etc/apache2/vhosts.d/default_vhost.include`) specifies where the web pages are stored. The phpMyAdmin interface will be found in that directory. MySQL on the other hand needs some basic configuration:

```
1 | emerge --config dev-db/mysql
```



```

2 | /etc/init.d/mysql start
3 | mysql_setpermission
4 | mysql_secure_installation

```

Listing 5.5: Basic MySQL configuration in Gentoo

The above listing 5.5 shows the basic steps to configure mysql after it was installed. In line one the basic configuration for mysql will be done. Line two starts the mysql server. In line three a shell script is used to help adding, changing and deleting users and databases. The fourth line is optional and will just make sure that insecure elements are removed from the installation (like the anonymous user, forcing a root password and setting permissions).

After the basic configuration of the MySQL database the phpMyAdmin needs to be configured to communicate with the database. This is done after the apache and mysql servers are started. The phpMyAdmin configuration is done with the web interface.

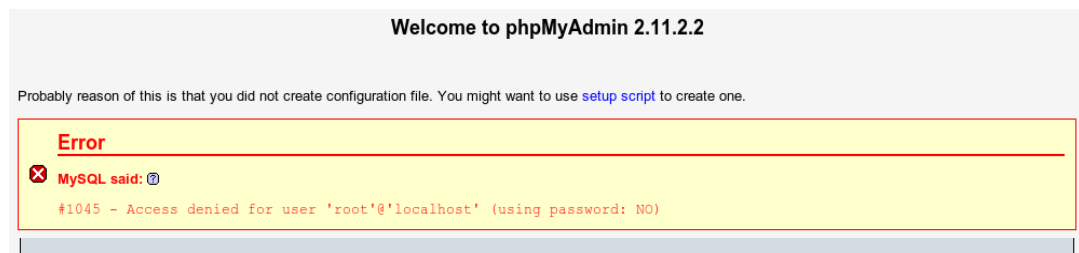


Figure 5.1: First start of phpMyAdmin

Figure 5.1 shows the page that will come up at the first start of an unconfigured phpMyAdmin interface when typing in the web address <http://localhost/phpmyadmin/> regarding that apache and phpMyAdmin are installed on the local machine. If not, localhost needs to be replaced by the appropriate address. The configuration then will start with a click on the setup script button. After the configuration (follow the instructions), phpMyAdmin will present a login dialog under the same address as above. After The login a screen similar to figure 5.2 is shown.

The phpMyAdmin interface of MySQL is very intuitive and helps creating users, tables and databases. Even people who do not know the SQL commands, have the possibility to add, change and edit things.

5.2 Coding style

The coding style should be well defined so everybody who wants to commit source knows in which style it should be sent in. To have a standard is the best way to keep the source in general readable.

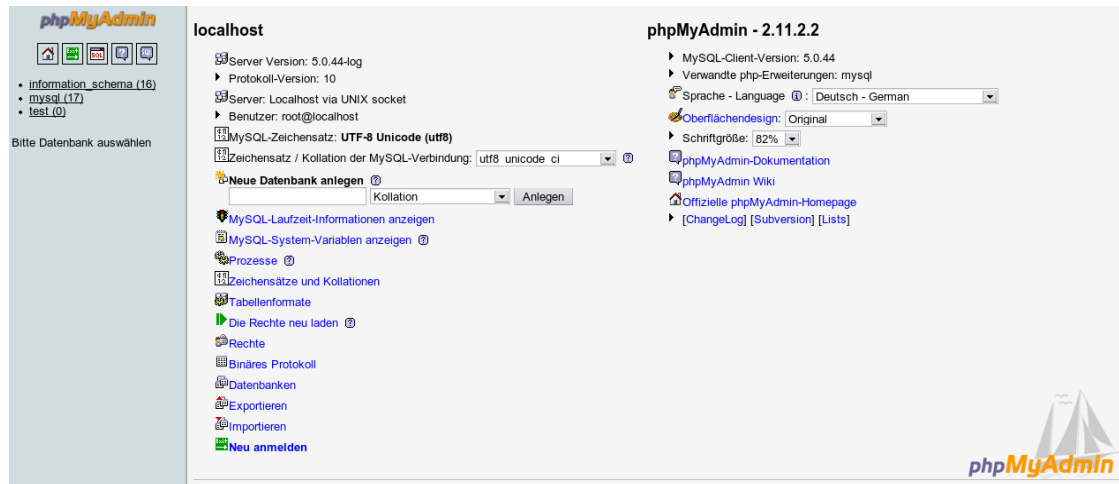


Figure 5.2: phpMyAdmin MySQL interface

- All the implementation is in the .cpp (source) files
- All the declarations are in the .h (header) files
- The code should be indented with tabs.
- A doxygen documentation on top of each function
- Comments on difficult source positions should describe what the source is meant to do
- Class names start with an uppercased letter
- Method names start with a lowercased letter
- Globals are complete uppercase
- Class, method and variable names should be easy to read or it should be easy to guess what they are for.
- Place the brace indented under the statement. Listing 5.6 shows an example.

```

1  if (x == 1)
2  {
3      ...
4  }
```

Listing 5.6: Example of right placement of braces

- All if, while, for statements require braces even if they only have one statement in between those braces

- Class names start with an uppercase letter, while method names start with a lower case one.

5.3 Milestones

This section is about the milestones that were reached during the project.

5.3.1 OpenBeacon Configurator + helper classes

The first Milestone corresponds to the finalization of the first application which will be the OpenBeacon Configurator. This application will be used to configure the OpenBeacon devices for the project. Additional to the application some common classes will be programmed and tested as well. Those common classes are the “DLD Log” class and the “OpenBeacon Communication” class. Both classes will be heavily used during the project. Thus a small application which utilizes them is perfect for testing and tuning them.

The implementation of the application went really straight forward. First the GUI was designed with the designer application, that comes with the Qt package. After the GUI design was done the functionalities were added step by step.

Problems that occurred during the development

The OpenBeacon communicator class was planned to be a helper class and only the data gain daemon should have a threaded version of the class. But during the implementation it shows that a threaded version is in need for every application that has to communicate with the openbeacon. Therefore the threaded version planned for the gain data daemon was moved to the OpenBeacon Communication class and that's the reason why the gain data daemon now has less classes to implement and there has more work been done during the implementation of the OpenBeacon Configurator

5.3.2 Firmware

To setup the environment for developing the compiler needs to be installed. To do so it needs to be downloaded from the OpenBeacon project via the following URL, <http://people.openpcd.org/meri/gnuarm-4.0.2.tar.bz2>. After the download is finished it needs to be unpacked to the root directory.

This project needs only a few additions to the currently available firmware so now we have to download and unpack the current version and add it to the repository.

The needed firmware is the estimator version 008 which can be retrieved from <http://www.openbeacon.org/dl/OpenBeacon/estimator-008.tar.bz2>.

Now the **Makefile** needs some adjustments to use the compilers from the correct destination. On top of the Makefile there are a few variables that have to get changed from:

```
1 CC=arm-elf-gcc
2 OBJCOPY=arm-elf-objcopy
3 OBJDUMP=arm-elf-objdump
4 ARCH=arm-elf-ar
```

Listing 5.7: Original head of the firmware Makefile

to:

```
1 CC=/usr/local/gnuarm-4.0.2/bin/arm-elf-gcc
2 OBJCOPY=/usr/local/gnuarm-4.0.2/bin/arm-elf-objcopy
3 OBJDUMP=/usr/local/gnuarm-4.0.2/bin/arm-elf-objdump
4 ARCH=/usr/local/gnuarm-4.0.2/bin/arm-elf-ar
```

Listing 5.8: Changed head of the firmware Makefile

The function that needs to be advanced is `void prvExecCommand (u_int32_t cmd, portCHAR * args)` which can be found in the `application/cmd.c`.

Function character	Function description
D	get the configured Id
M	get the configured mode (returns 0-4)
U	get the uptime in seconds
A	get the channel of the node
L	get FIFO cache lifetime

Code changes that were made

The only performed changes were done in the file which concerns on the commands: `application/cmd.c`. The switch-case structure was enhanced by the following lines:

```
1 // below new stuff for FYP from Simon Schaefer
2 case 'D':
3     DumpStringToUSB(" Id: ");
4     DumpUIntToUSB(env.e.reader_id);
5     DumpStringToUSB("\n\r");
6     break;
7 case 'M':
8     DumpStringToUSB(" Mode: ");
9     DumpUIntToUSB(env.e.mode);
10    DumpStringToUSB("\n\r");
```

```

11         break ;
12 case 'U':
13     DumpStringToUSB(" Uptime: ");
14     s=xTaskGetTickCount() /1000;
15     DumpUIntToUSB(s);
16     DumpStringToUSB("\n\r");
17     break ;
18 case 'A':
19     DumpStringToUSB(" Channel: ");
20     DumpUIntToUSB(nRFAPI_GetChannel());
21     DumpStringToUSB("\n\r");
22     break ;
23 case 'L':
24     DumpStringToUSB(" FIFOLifetime: ");
25     DumpUIntToUSB(PtGetFifoLifetimeSeconds());
26     DumpStringToUSB("\n\r");
27     break ;
28 // additional stuff finished

```

Listing 5.9: inserted source to application/cmd.c

Problems that occurred during the development

As the system is not case sensitive the commands were changed from lower case characters to upper case characters and in cases where this was not possible new ones were chosen.

Additional work

After the implementation went so fast and everything just worked out of the box, the decision was made to enhance the OpenBeacon Configurator by the commands. Even as the OpenBeacon Configurator already has the ability to enhance the commands manually it would be nice to add a the new command set through the “Default Commands” button as well. Therefore the existing question box which asks if the entries should be replaced was enhanced by two checkboxes where the user can select the command sets which should be used for replacement.

5.3.3 Data gain daemon

After the OpenBeacon communication class was changed to be threaded right out of the box the class diagram for the data gain daemon changed to the following:

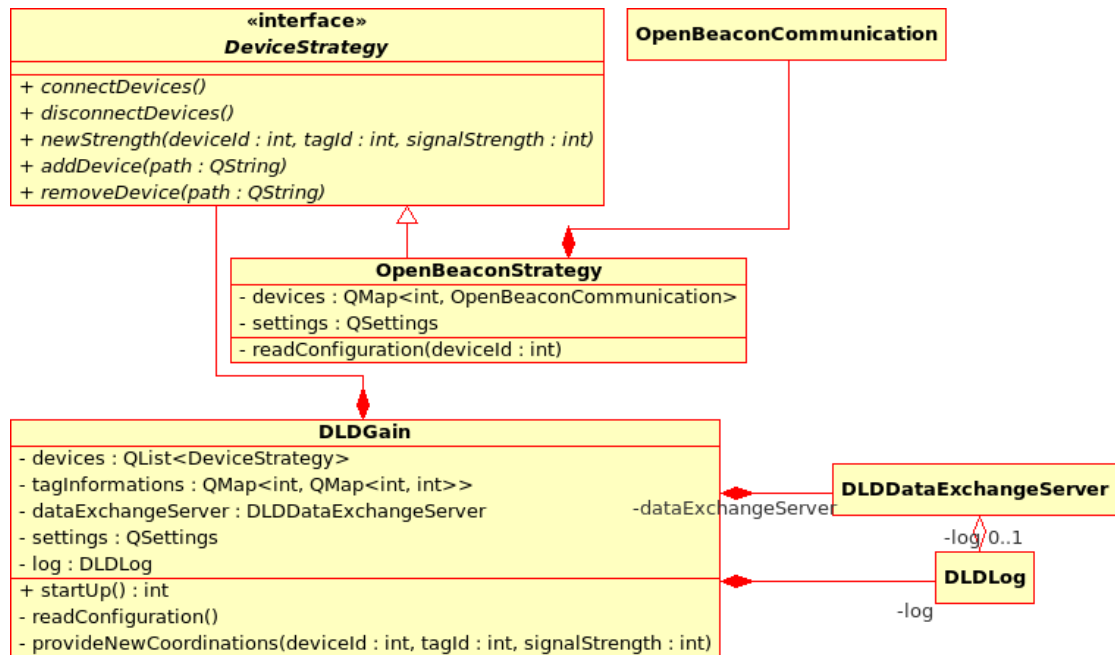


Figure 5.3: UML class diagram of the adjusted gain data daemon

To accelerate the development of the project the implementation of the SSL data exchange strategy was not done in this stage. Only the implementation of the D-Bus exchange strategy was performed. The implementation of the SSL exchange strategy will be done when the project has some spare time at the end. The only implementations made for SSL are the method headers to test if the communication strategy design is functional.

Code changes that were made

During the implementation of the D-Bus strategy the complexity of the design was not easy to implement and due to advancement of the project the planned design was reverted to an earlier version. The current version of the design has both the server and the client in one class. With this design one signal or method would be used for both directions. One of the early versions had two classes one for the server methods and one for the client methods. This old design is now the current design to speed up the development.

5.3.4 Hardware Simulator

The hardware simulator was implemented to simulate and test the different daemons and applications. The drawn representation of the values was made to visualise it for the user who is testing. Right now it is reduced to only support three nodes, so only 2D position detection is supported. The hardware simulator is integrated in the data gain daemon and works as a strategy for the device backend. The extra time

that was spent in the design stage to design a strategy pattern around the different hardware backends was not vain. The pure implementation as a new strategy did not take much time because of the well structured design.

5.3.5 Generate position daemon

The first part (receiving the signals from the data gain daemon) was done to fulfill the requirements for the distance measurement. The D-Bus Client was implemented and tested with the data gain daemon.

Code changes that were made

Again the D-Bus system worked in a different way than expected. Instead of the possibility to register each object that should be accessible through D-Bus by its own only all or no Object could be added. Therefore the class for providing the different types of the position data was split up into two classes. One for the strength position and one for the relative position data calculated by this daemon.

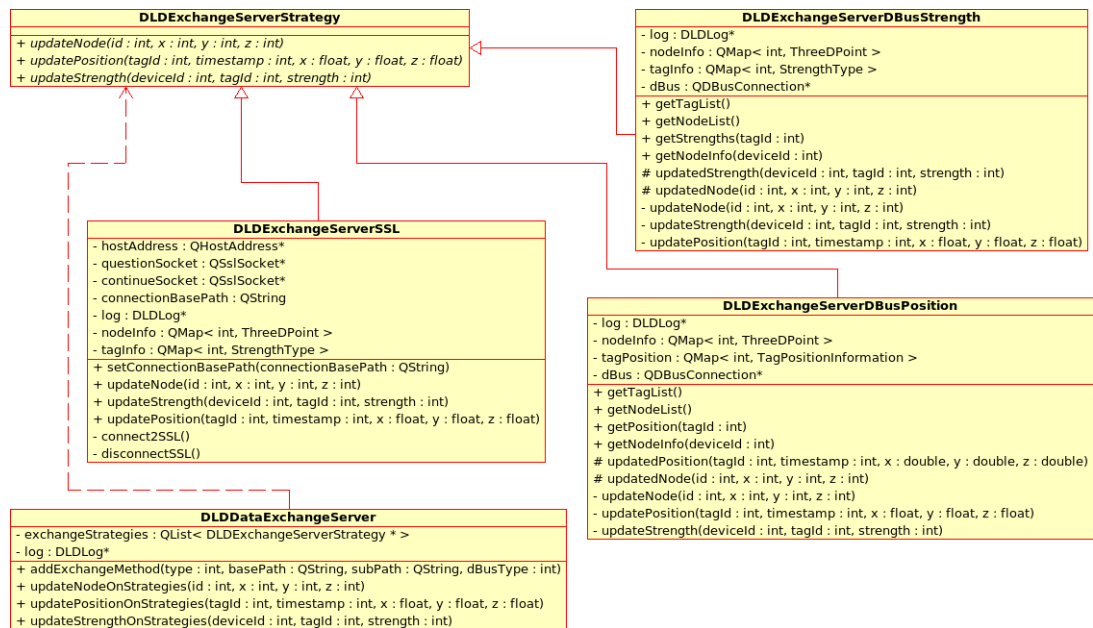


Figure 5.4: UML class diagram of the adjusted data exchange server

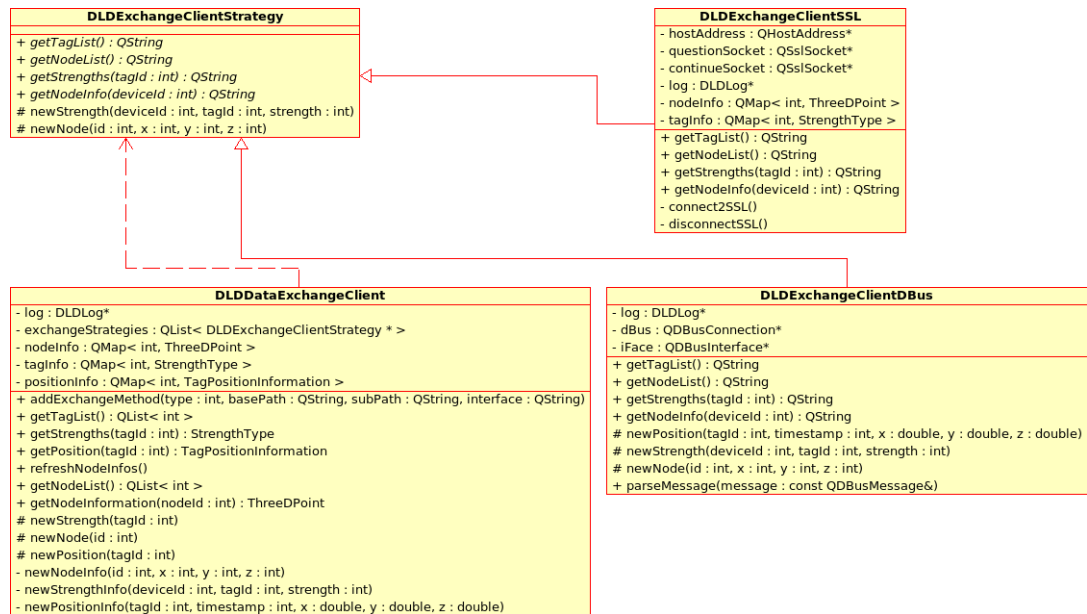


Figure 5.5: UML class diagram of the adjusted data exchange client

Changes affecting the whole project

Not only the D-Bus must be changed, but also the need of measurement data was reverted. The whole position localisation can be realized with only the relative data from the OpenBeacons. The maps and everything just needs to use the raw data which is generated and calculated by this daemon just on the basis of the raw packet loss. Therefore the system will look like shown in figure 5.6.

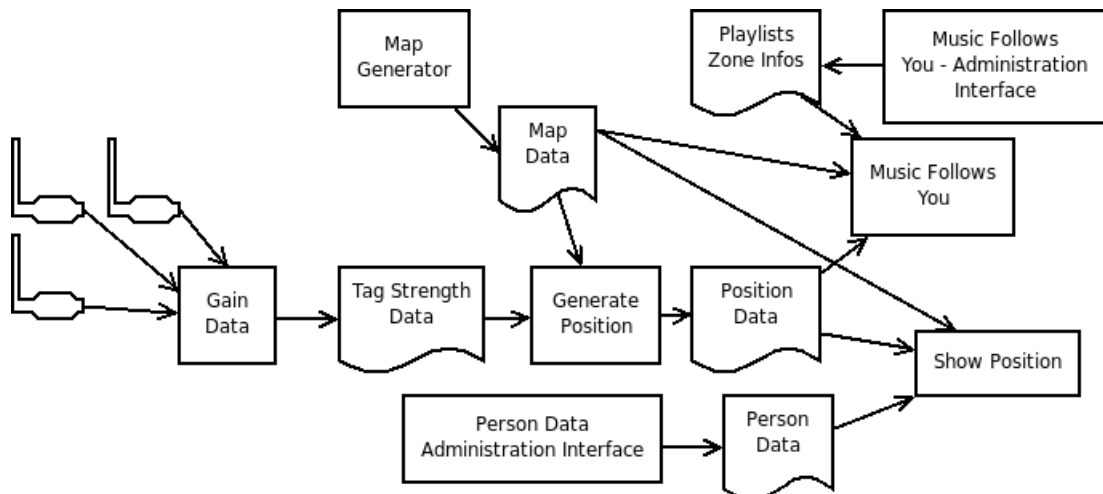


Figure 5.6: Diagram of the adjusted project structure

Problems that occurred during the development

While the RFID system worked good when only one node was used (for testing of the hardware and its performance), working with three devices at once the tag gets

confused and does not send all the data back to each node. You can see that on the strength data the nodes are receiving. If two nodes are on the same axis and only one unit is between them the deviation may have a maximum of one unit. But the data that is received in some tests differs in about 5 units which means that one circle is part of the other one. This whole problem results in the fact that RFID in this constellation (based on packet loss) and with this hardware (USB OpenBeacon nodes and the active RFID tag) is unusable right now. But for the whole project this means nothing because it was planned to be used for multiple technologies, so if someone wants to create a location detection based on another hardware (WLAN for example) then only the device strategy for it must be implemented and not the whole system.

5.3.6 Administrate person data

The implementation of the person data administration tool went straight forward and did not encounter any major problems. First the user interface was designed with the Qt designer and afterwards the functionalities of the buttons and the GUI were implemented step by step. During the implementation the example database was created too. For this purpose the database web interface phpmyadmin was used to create a database, a table and a user which can access this database. Listing 5.10 shows which command needs to be performed to create the example database, with all the needed information regarding the table and the user.

```

1 CREATE DATABASE 'dldPersons' ;
2
3 CREATE TABLE 'dldPersons'.'persons' (
4   'tagId' INT NOT NULL ,
5   'name' VARCHAR( 255 ) NOT NULL ,
6   'prename' VARCHAR( 255 ) NOT NULL ,
7   'color' VARCHAR( 7 ) NOT NULL DEFAULT '#000000',
8   'picture' BLOB NULL ,
9   'description' VARCHAR( 1024 ) NULL ,
10  PRIMARY KEY ( 'tagId' )
11 ) ENGINE = MYISAM
12
13 CREATE USER 'dldUser' '@%' IDENTIFIED BY 'dldPass';
14
15 GRANT USAGE ON * . * TO 'dldUser' '@%' IDENTIFIED BY 'dldPass' WITH
16 MAX_QUERIES_PER_HOUR 0 MAX_CONNECTIONS_PER_HOUR 0 MAX_UPDATES_PER_HOUR
17   0
18 MAX_USER_CONNECTIONS 0 ;
19 GRANT SELECT , INSERT , UPDATE , DELETE , CREATE , INDEX , ALTER ON

```

```
20 | 'dldPersons' . * TO 'dldUser' '@' %';
```

Listing 5.10: MySQL commands for creating the example database

5.3.7 Show position

Because of the changes that were made in the generate position daemon the show position daemon just shows the position information in a coordinate system which is relative to the received data from the nodes. After the implementation of the person data administrator was done a lot of code could be reused or copied and slightly changed for the show position daemon. In addition to the show position application a part of the planned map class and the whole map scene class were implemented. The map scene class is used to display the map data in a Qt style scene, therefore it was needed to fully implement it. The implemented part of the map class consists off all functionalities to store the most important parts. Designs like the one that is used for saving or loading maps were not implemented because they would only make sense if there was an implemented map generator. So the implementation was reduced to the minimum of functionalities plus some extra fancy stuff to make it interesting.

6 Testing

This chapter covers the testing of the various applications. All the tests followed the Black Box testing strategy, where the key functionalities of the applications were tested and verified.

6.1 OpenBeacon Configurator

6.1.1 Menus

Application:	OpenBeacon Configurator
Test:	File menu -> Quit
Description	Click with the mouse on the Quit entry in the File menu
Expected result:	Exit application, store settings
Actual result:	Worked like expected
Application:	OpenBeacon Configurator
Test:	Device menu -> Refresh
Description	Click with the mouse on the Refresh entry in the Device menu
Expected result:	refresh the device list and update the pull down menu
Actual result:	Worked like expected
Application:	OpenBeacon Configurator
Test:	Device menu -> Preferences
Description	Click with the mouse on the Preferences entry in the Device menu
Expected result:	Display the preference Dialog
Actual result:	Worked like expected
Application:	OpenBeacon Configurator
Test:	Help menu -> OpenBeacon Configurator Quick Help
Description	Click with the mouse on the OpenBeacon Configurator Quick Help entry in the Help menu
Expected result:	Display the Quick Help Dialog
Actual result:	Worked like expected

Application:	OpenBeacon Configurator
Test:	Help menu -> About Qt
Description	Click with the mouse on the About Qt entry in the Help menu
Expected result:	Display the About Qt Dialog
Actual result:	Worked like expected
Application:	OpenBeacon Configurator
Test:	Help menu -> About OpenBeacon
Description	Click with the mouse on the About OpenBeacon entry in the Help menu
Expected result:	Display the About OpenBeacon Dialog
Actual result:	Worked like expected
Application:	OpenBeacon Configurator
Test:	Keyboard Shortcuts for the menu entries, mentioned above
Description	Press the keyboard shortcut that is mapped to the menu command
Expected result:	React the same like clicking the menu entry
Actual result:	Worked like expected

6.1.2 Dialogs

Application:	OpenBeacon Configurator - Preference Dialog
Test:	Change/add logfile parameter
Description	Instead of printing the log messages to the shell, they should be logged into a file
Expected result:	If empty, print log messages to shell, if not write them to the file
Actual result:	Worked like expected
Application:	OpenBeacon Configurator - Preference Dialog
Test:	change base names for the devices
Description	Using different base names will result into a change in the main window, where different devices will be scanned and opened.
Expected result:	Empty entry is not allowed, if entry is changed change of pull down menu entries and listed devices
Actual result:	Worked like expected

Application:	OpenBeacon Configurator - Preference Dialog
Test:	Show the received tag specific data packets
Description	Should the data packets be displayed in the main windows console.
Expected result:	When selected tag data should be displayed on the main window console
Actual result:	Worked like expected
Application:	OpenBeacon Configurator - Preference Dialog
Test:	Change device refresh rate
Description	A change of the refresh rate should change the interval in which the pull down list is actualised
Expected result:	Refresh the pull down list every time at the given interval
Actual result:	Worked like expected
Application:	OpenBeacon Configurator - Preference Dialog
Test:	Change Sam 7 path
Description	Changes to the sam7 path will affect the flashing
Expected result:	When changed the flashing will use the new location
Actual result:	Worked like expected
Application:	OpenBeacon Configurator - Preference Dialog
Test:	Default commands button
Description	Opens a dialog, where the user can choose from the command sets, which will replace the current entries.
Expected result:	Command table will be replaced by the command sets
Actual result:	Worked like expected
Application:	OpenBeacon Configurator - Preference Dialog
Test:	Add button
Description	Adds an empty line in the command table, which the user then has to fill.
Expected result:	Add empty line to command table
Actual result:	Worked like expected
Application:	OpenBeacon Configurator - Preference Dialog
Test:	Delete button
Description	Removes the current selected line from the command table. Asks for confirmation before it takes effect
Expected result:	Remove entry from list
Actual result:	Worked like expected

6.1.3 Main Window

Application:	OpenBeacon Configurator - Main Window
Test:	Select flash device
Description	Select a flash device from the pull down menu
Expected result:	Enable the flashing controls
Actual result:	Worked like expected
Application:	OpenBeacon Configurator - Main Window
Test:	Select configuration device
Description	Select a configuration device from the pull down menu
Expected result:	Enable the configuration controls
Actual result:	Worked like expected
Application:	OpenBeacon Configurator - Main Window
Test:	Refresh Button
Description	Press the refresh button
Expected result:	Refresh the pull down menu manually
Actual result:	Worked like expected
Application:	OpenBeacon Configurator - Main Window
Test:	File selection (... button)
Description	Press the ... button to open a file selection box
Expected result:	Display a file selection box and insert selected firmware path into line edit
Actual result:	Worked like expected
Application:	OpenBeacon Configurator - Main Window
Test:	Flash button
Description	Press the flash button to start the flashing of the device, show short instructions.
Expected result:	Flashed device
Actual result:	Worked like expected
Application:	OpenBeacon Configurator - Main Window
Test:	Execute button
Description	Execute the selected command with the given argument on the device.
Expected result:	Reaction from the device, changed value on the device, response on the main window console.
Actual result:	Worked like expected

Application:	OpenBeacon Configurator - Main Window
Test:	Clear button
Description	Clear the main window console
Expected result:	Empty main window console
Actual result:	Worked like expected

6.2 Firmware

Application:	OpenBeacon firmware
Test:	new command D
Description	To test the new command “D”, use the console command cu or the OpenBeacon Configurator
Expected result:	Get only the id from the node
Actual result:	Worked like expected
Application:	OpenBeacon firmware
Test:	new command M
Description	To test the new command “M”, use the console command cu or the OpenBeacon Configurator
Expected result:	Get only the mode from the node
Actual result:	Worked like expected
Application:	OpenBeacon firmware
Test:	new command U
Description	To test the new command “U”, use the console command cu or the OpenBeacon Configurator
Expected result:	Get only the uptime from the node
Actual result:	Worked like expected
Application:	OpenBeacon firmware
Test:	new command A
Description	To test the new command “A”, use the console command cu or the OpenBeacon Configurator
Expected result:	Get only the Channel from the node
Actual result:	Worked like expected
Application:	OpenBeacon firmware
Test:	new command L
Description	To test the new command “L”, use the console command cu or the OpenBeacon Configurator
Expected result:	Get only the FIFO lifetime from the node
Actual result:	Worked like expected

6.3 Gain Data

To test the data gain daemon there are two possibilities, the OpenBeacon hardware and the hardware simulator. Both provide the same functions but the hardware simulator can perform some actions the hardware is not capable of. So each test has two actual results, one for the OB hardware and one for the hardware simulator. The created data will be provided on the D-Bus. To check if the values are correct the Qt tool qdbusviewer will be used to connect to the interface the daemon is providing.

Application:	Gain data daemon
Test:	Periodical update of the strength data
Description	Connect the qdbusviewer to the updatedStrength () D-Bus signal
Expected result:	Receive a strength string from every node every second
Actual result for OpenBeacon hardware:	Worked like expected
Actual result for hardware simulator:	Worked like expected
Application:	Gain data daemon
Test:	send updated node positions
Description	Connect the qdbusviewer to the updatedNode () D-Bus signal
Expected result:	Receive a node position string from the changed node
Actual result for OpenBeacon hardware:	this strategy does not support this feature
Actual result for hardware simulator:	Worked like expected

Application:	Gain data daemon
Test:	send updated maximum axis value
Description	Connect the qdbusviewer to the updatedMaximumAxis-Value () D-Bus signal
Expected result:	Receive the new value when it was changed
Actual result for OpenBeacon hardware:	This strategy does not support this feature
Actual result for hardware simulator:	Worked like expected

Application:	Gain data daemon
Test:	Get tag list
Description	Call the getTagList () D-Bus method through qdbusviewer
Expected result:	Receive the tag list
Actual result for OpenBeacon hardware:	Worked like expected
Actual result for hardware simulator:	Worked like expected

Application:	Gain data daemon
Test:	Get node list
Description	Call the getNodeList () D-Bus method through qdbusviewer
Expected result:	Receive the node list
Actual result for OpenBeacon hardware:	Worked like expected
Actual result for hardware simulator:	Worked like expected

Application:	Gain data daemon
Test:	Get the strengths from each node for one specific tag list
Description	Call the getStrengths () D-Bus method through qd-busviewer and enter the id of the tag
Expected result:	Receive the strength list
Actual result for OpenBeacon hardware:	Worked like expected
Actual result for hardware simulator:	Worked like expected

Application:	Gain data daemon
Test:	Get the position information from requested node
Description	Call the getNodeInfo () D-Bus method through qd-busviewer and enter the id of the node
Expected result:	Receive the position information of the requested node
Actual result for OpenBeacon hardware:	Worked like expected
Actual result for hardware simulator:	Worked like expected

Application:	Gain data daemon
Test:	Get the maximum axis value
Description	Call the getMaximumAxisValue () D-Bus method through qdbusviewer
Expected result:	Receive the maximum axis value
Actual result for OpenBeacon hardware:	Worked like expected
Actual result for hardware simulator:	Worked like expected

6.4 Generate Position

To have a properly working generate position daemon, a gain data daemon must be up and running correctly. Again the created data will be provided on the D-Bus and

to check if the values are correct the Qt tool qdbusviewer will be used to connect to the interface the daemon is providing.

Application:	Generate Position daemon
Test:	Update the position of tag
Description	Connect the qdbusviewer to the updatedStrength () D-Bus signal, the position will be updated every time the nodes send valid data
Expected result:	Receive a position string from every node every second
Actual result:	Worked like expected
Application:	Generate Position daemon
Test:	send updated node positions
Description	Connect the qdbusviewer to the updatedNode () D-Bus signal
Expected result:	Receive a node position string from the changed node
Actual result:	Worked like expected
Application:	Generate Position daemon
Test:	Send updated maximum axis value
Description	Connect the qdbusviewer to the updatedMaximumAxis-Value () D-Bus signal
Expected result:	Receive the new value when it was changed
Actual result:	Worked like expected
Application:	Generate Position daemon
Test:	Get tag list
Description	Call the getTagList () D-Bus method through qdbusviewer
Expected result:	Receive the tag list
Actual result:	Worked like expected
Application:	Generate Position daemon
Test:	Get node list
Description	Call the getNodeList () D-Bus method through qdbusviewer
Expected result:	Receive the node list
Actual result:	Worked like expected

Application:	Generate Position daemon
Test:	Get the position from requested tag
Description	Call the getPosition () D-Bus method through qdbusviewer and enter the id of the tag
Expected result:	Receive the tags position
Actual result:	Worked like expected
Application:	Generate Position daemon
Test:	Get the position information from requested node
Description	Call the getNodeInfo () D-Bus method through qdbusviewer and enter the id of the node
Expected result:	Receive the position information of the requested node
Actual result:	Worked like expected
Application:	Generate Position daemon
Test:	Get the maximum axis value
Description	Call the getMaximumAxisValue () D-Bus method through qdbusviewer
Expected result:	Receive the maximum axis value
Actual result:	Worked like expected

6.5 Administrate Person Data

6.5.1 Menus

Application:	Administrate Person Data
Test:	File -> Connect ...
Description	Click with the mouse on the Connect ... entry in the File menu
Expected result:	Show the connect to database dialog
Actual result:	Worked like expected
Application:	Administrate Person Data
Test:	File -> Quit
Description	Click with the mouse on the Quit entry in the File menu
Expected result:	Exit the application and store settings
Actual result:	Worked like expected

Application:	Administrate Person Data
Test:	Keyboard Shortcuts for the menu entries, mentioned above
Description	Press the keyboard shortcut that is mapped to the menu command
Expected result:	React the same like clicking the menu entry
Actual result:	Worked like expected

6.5.2 Dialogs

Application:	Administrate Person Data - Connect to Database Dialog
Test:	Connect to database with given values
Description	Fill out all needed fields and press OK
Expected result:	If all values are correct connect with the database, if not print an error
Actual result:	Worked like expected

6.5.3 Main Window

To test any action on the main window the Administrate Person Data application needs to be connected to the database.

Application:	Administrate Person Data - Main Window
Test:	refresh the values in the entry table
Description	Click on the refresh button
Expected result:	Refresh the list of entries in the database
Actual result:	Worked like expected
Application:	Administrate Person Data - Main Window
Test:	Delete an entry from list and database
Description	Click on the delete button
Expected result:	Remove entry from database and refresh list
Actual result:	Worked like expected
Application:	Administrate Person Data - Main Window
Test:	Add picture to entry
Description	Click on the ... button below the picture field
Expected result:	Select a picture and display it in the picture field
Actual result:	Worked like expected

Application:	Administrate Person Data - Main Window
Test:	Chose a colour for the tag
Description	Click on the ... button next to the colour field
Expected result:	Fill the colour field with the name of the colour, fill the colour picture with the selected colour
Actual result:	Worked like expected
Application:	Administrate Person Data - Main Window
Test:	Empty all fields
Description	Click on the clear button
Expected result:	All fields for the entry are cleared but nothing changed in database
Actual result:	Worked like expected
Application:	Administrate Person Data - Main Window
Test:	Show selected entry
Description	Select an entry from entry list
Expected result:	All entry fields should be filled with the data from the list
Actual result:	Worked like expected
Application:	Administrate Person Data - Main Window
Test:	Commit changes or new entry
Description	Click on the commit button, after the fields were filled with content
Expected result:	If the tag id already exists overwrite the data from the old entry, if not then create a new entry. In both cases submit the entry to the database and refresh list.
Actual result:	Worked like expected

6.6 Show Position

6.6.1 Menus

Application:	Show Position
Test:	File menu -> Load map
Description	This function is not yet implemented
Expected result:	Nothing should happen.
Actual result:	Worked like expected
Application:	Show Position
Test:	File menu -> Connect to Generate Position
Description	Click with the mouse on the Connect to Generate Position entry in the File menu
Expected result:	Open the Connect to generate Position dialog
Actual result:	Worked like expected
Application:	Show Position
Test:	File menu -> Disconnect from Generate Position
Description	Click with the mouse on the Disconnect from Generate Position entry in the File menu
Expected result:	Disconnect the connection to the Generate Position daemon
Actual result:	Worked like expected
Application:	Show Position
Test:	File menu -> Connect to database
Description	Click with the mouse on the Connect to database entry in the File menu
Expected result:	Open the Connect to database dialog
Actual result:	Worked like expected
Application:	Show Position
Test:	File menu -> Disconnect from database
Description	Click with the mouse on the Disconnect from database entry in the File menu
Expected result:	Disconnect the open database connection
Actual result:	Worked like expected

Application:	Show Position
Test:	File menu -> Quit
Description	Click with the mouse on the Quit entry in the File menu
Expected result:	Exit application and permanently store the settings
Actual result:	Worked like expected
Application:	Show Position
Test:	Show menu -> Show Person Info
Description	Check or uncheck check box to show or hide the Person Information Dock
Expected result:	Show/hide Person Information Dock
Actual result:	Worked like expected
Application:	Show Position
Test:	Show menu -> Show Mouse-over Person Information
Description	Check or uncheck check box to enable/disable the widget that appears if the mouse is moved over a tag
Expected result:	Enable/disable Mouse-over Person Information widget
Actual result:	Worked like expected
Application:	Show Position
Test:	Show menu -> Show Database list Dock
Description	Check or uncheck check box to show or hide the Database list Dock
Expected result:	Show/hide Database list Dock
Actual result:	Worked like expected
Application:	Show Position
Test:	Help menu -> About
Description	Click with the mouse on the About entry in the Help menu
Expected result:	Display About show position dialog
Actual result:	Worked like expected
Application:	Show Position
Test:	Help menu -> About Qt
Description	Click with the mouse on the About Qt entry in the Help menu
Expected result:	Display About Qt dialog
Actual result:	Worked like expected

Application:	Show Position
Test:	Keyboard shortcuts matching the menu entries
Description	Invoke the shortcut
Expected result:	For each key combination the same result like the menu entry
Actual result:	Worked like expected
Application:	Show Position
Test:	Toolbar buttons matching the menu entries
Description	Click on the button on the toolbar
Expected result:	For each toolbar button the same result like the matching menu entry
Actual result:	Worked like expected

6.6.2 Dialogs

Application:	Show Position - Connect to database dialog
Test:	all entries are filled from a previous session
Description	The system should set the focus to the first field where it does not know the content
Expected result:	All entries except the password field are filled and the focus is at the password field
Actual result:	Worked like expected
Application:	Show Position - Connect to database dialog
Test:	Connect to database if all entries are correct
Description	After clicking OK (or press enter) the dialog should connect to the database, or report an error
Expected result:	Establish connection, or report error
Actual result:	Worked like expected
Application:	Show Position - Connect to Generate Position dialog
Test:	Make selection of type
Description	Click on one of the two choices and only one is allowed to be selected
Expected result:	Choose either D-Bus or SSL, not both
Actual result:	Worked like expected

Application:	Show Position - Connect to Generate Position dialog
Test:	Connect to the generate position daemon
Description	After the choice is made and OK (or enter) is pressed, the dialog should build up a connection to the generate position daemon
Expected result:	If D-Bus is selected it should connect to the generate position daemon, if SSL is selected then it should report an error because this is not yet implemented.
Actual result:	Worked like expected

6.6.3 Dock Widgets

Application:	Show Position - Person Information Dock widget
Test:	Hide through menu selection
Description	Uncheck the check box in the menu: Show -> Show Person Info
Expected result:	Hide the Person Info dock widget, check box in menu should be unchecked
Actual result:	Worked like expected
Application:	Show Position - Person Information Dock widget
Test:	Hide/Close through the small X on the top right of the widget
Description	Click on the small X on the top right of the widget
Expected result:	Hide the Person Info dock widget, check box in menu should be unchecked
Actual result:	Worked like expected
Application:	Show Position - Person Information dock widget
Test:	Show Person Information dock widget
Description	Check the check box in the menu: Show -> Show Person Info
Expected result:	Show the Person Information dock widget
Actual result:	Worked like expected

Application:	Show Position - Database entry list dock widget
Test:	Hide through menu selection
Description	Uncheck the check box in the menu: Show -> Show Database List Dock
Expected result:	Hide the Database entry list dock widget, check box in menu should be unchecked
Actual result:	Worked like expected
Application:	Show Position - Database entry list dock widget
Test:	Hide/Close through the small X on the top right of the widget
Description	Click on the small X on the top right of the widget
Expected result:	Hide the Database entry list dock widget, check box in menu should be unchecked
Actual result:	Worked like expected
Application:	Show Position - Database entry list dock widget
Test:	Show Database entry list dock widget
Description	Check the check box in the menu: Show -> Show Database List Dock
Expected result:	Show the Database entry list dock widget
Actual result:	Worked like expected
Application:	Show Position - Database entry list dock widget
Test:	Not connected to database, so refresh button should be disabled
Description	Try to click on the button while the application is not connected to a database
Expected result:	Should not clickable
Actual result:	Worked like expected
Application:	Show Position - Database entry list dock widget
Test:	Connected to database, so refresh button should refresh list
Description	Connect to the database, and then change (new/delete/add) the database (with Administrative Person Data Application)
Expected result:	When refresh is clicked the new entry should appear in the list
Actual result:	Worked like expected

6.6.4 Main Window

Application:	Show Position - Map View
Test:	Display map
Description	If connected to generate position daemon, it should draw a coordinate system and display tags if there are known positions.
Expected result:	A drawn coordinate system with displayed tags
Actual result:	Worked like expected
Application:	Show Position - Mouse over tag
Test:	Display a custom mouse over widget
Description	If a tag is displayed move the mouse over the displayed tag
Expected result:	Update the data (default or from database) in the Person Information dock and if Show -> Show Mouse over Person Info is selected then right next to the mouse cursor a custom widget with the person information data should appear.
Actual result:	Worked like expected
Application:	Show Position - Mouse over tag
Test:	Show the correct time
Description	Update time value every time the tag got a new (or old) position
Expected result:	Updating of time (in person information dock) of the last hovered tag every time new data for this tag is coming in
Actual result:	Worked like expected
Application:	Show Position - Map View
Test:	Zoom the map view
Description	As soon as the zoom value is changed the map view should zoom in or out
Expected result:	Change the size of the map view.
Actual result:	Worked like expected

6.6.5 Process

Application:	Show Position - Freshly started
Test:	Restore settings, show blank map view
Description	At first start all Show possibilities are enabled, on later use the user might have hid a dock view and this should be stored persistently
Expected result:	Show only those widgets that where visible on the last use of the show position application
Actual result:	Worked like expected
Application:	Show Position
Test:	First Connect to Generate position then to database
Description	<ol style="list-style-type: none"> 1. Connect to generate position and wait until the tag appears on the map. 2. Before connecting to database verify that the tag is listed in it, then connect to the database.
Expected result:	<ol style="list-style-type: none"> 1. It should display the map and the tag on it, the tag has no colour and if the mouse moves over it the fields are filled with default values. 2. If database connection is built up and the tag is in the database then its colour should change to the one saved in the database and on mouse over the person information fields should be filled with the right values.
Actual result:	Worked like expected
Application:	Show Position
Test:	First Connect to Generate position then to database
Description	<ol style="list-style-type: none"> 1. Before connecting to database verify that the tag is listed in it, then connect to the database. 2. Connect to generate position and wait until the tag appears on the map.
Expected result:	<ol style="list-style-type: none"> 1. It should display the database entries in the database entries dock. 2. The tag should appear with the colour that is stored in the database and on mouse over the values from the database are visible in the person information dock
Actual result:	Worked like expected

6.7 The whole system

Application:	Domestic location detection
Test:	The whole system with the hardware simulator as hardware backend
Description	<ol style="list-style-type: none"> 1. Start data gain daemon with hardware simulator as backend 2. Start generate position daemon 3. Create an entry in the database for tag 666 (test tag id) 4. Start show position application 5. Connect to the database 6. Connect to generate position daemon
Expected result:	<ol style="list-style-type: none"> 1. Display the position of the tag in the map view. 2. If the strength values are chosen wrong and they do not have a point in common, the tag should stay on last known position 3. On mouse over the information from the database should be displayed in the person information docks
Actual result:	Worked like expected
Application:	Domestic location detection
Test:	The whole system with the OpenBeacon hardware as hardware backend
Description	<ol style="list-style-type: none"> 1. Start data gain daemon with OpenBeacon USB as backend 2. Start generate position daemon 3. Create an entry in the database for a tag that is available 4. Start show position application 5. Connect to the database 6. Connect to generate position daemon
Expected result:	<ol style="list-style-type: none"> 1. Display the position of the tag in the map view and should represent the position of the tag, in relation to the central node.. 2. If the strength values are chosen wrong and they do not have a point in common, the tag should stay on last known position 3. On mouse over the information from the database should be displayed in the person information docks
Actual result:	Sometimes the shown position is like the expected one but most of the times the position is jumping around.

6.8 Hardware Simulator - gain data daemon backend

Application:	Gain data daemon backend - Hardware simulator
Test:	Check if the backend is working appropriate
Description	<ol style="list-style-type: none"> 1. Start data gain daemon with hardware simulator backend 2. Start the qdbusviewer to watch what the gain data daemon is reporting 3. Change the node values
Expected result:	The simulated display of the three nodes should change when the node preferences are changed (coordinates and radii)
Actual result:	Worked like expected
Application:	Gain data daemon backend - Hardware simulator
Test:	Check if the data from gain data daemon is correct
Description	<ol style="list-style-type: none"> 1. Start data gain daemon with hardware simulator backend 2. Start the qdbusviewer to watch what the gain data daemon is reporting 3. Change the node values
Expected result:	Send exactly the values that the simulator is set to
Actual result:	Worked like expected

6.9 Outcome

The only major problem that occurred during the testing was, that the OpenBeacon USB hardware was sending unexpected strength values. Those values do not depend on any systematics therefore it was not possible to clarify where those strange values come from. The strength data that the nodes are receiving varied from test to test, and jumps in irregular values. Anyway the system itself is working and with better (more expensive) hardware it will work better, and can be used within the different areas it was designed for.

7 Critical Evaluation of the whole Project

7.1 Planning

7.1.1 Chosen Hardware

During research I should have taken more time on the chosen hardware, and should have figured out that it is working with packet loss and not with signal strength. This was determined after the hardware was bought and the company who builds them said that they want to change it as soon as the components which are needed to determine the signal strength got cheaper. They want to provide a system that is cheap and not a system that is able to do everything in a perfect case but would make it expensive. As soon as the signal strength components are built into the hardware the device strategy regarding the open beacon USB devices needs to be adjusted and then the whole system would work with those. Data instead of the inaccurate packet loss data.

If this project would have been for a company that would like to use it in a commercial way then all hardware should have been bought in advance and should have been checked if they are capable of solving the task in an accurate way. This includes a prototype for each hardware and a basic position detection for each.

7.2 Design and Implementation

The designing went good. But during the implementation some of the previously designed classes were not feasible. This is because I never used some of the classes in the Qt library before and therefore did not know what was exactly doable and what not. The Qt D-Bus library for example is powerful but has some restrictions during the export of the functions. I thought that it would work in a different way than it actually does, instead of assuming how something will work, I should have tested it during the design stage. In this case it would have prevented flaws in the D-Bus interface of my project. On the other hand if I had tested everything in advance the

whole design stage would have been a small implementation stage, where all classes would have been implemented in a raw version. Those misunderstanding lead to a few changes that had been made to the design during the implementation stage.

7.3 Conclusion

Even if the hardware is not suitable for this task, the project itself is a success because all three layers were implemented and the design of the hardware layer permits an easy way to implement driver for hardware other then the USB OpenBeacon devices.

8 Prospect of the future

Hardware

For the future I think that I will try to implement a few other drivers that are much more common like WLAN access points. And test the whole position location detection with a laptop that is carried around. In the later stage of the project (during easter break) I heard of a technology called zWave which is used as a replacement of X10. The latter is a protocol which lets compatible products communicate with each other over the power lines in a home. zWave on the other hand is able to do the same but wireless. Both technologies are used for home automation products. While components of X10 are mostly very expensive the zWave parts are relatively cheap, and perhaps this hardware is capable of performing a good location detection.

Applications

For further applications I have an implementation of the “Music follows you” application in my mind, which is basically designed in section 4.11. Also a map generator should be implemented to easily create maps and zones on them.

Improving the system

The Capability to add more than 3 devices to cover a larger area should be implemented as well as a 3D location detection. Both improvements would affect the “data gain daemon” and the “generate position daemon”, as well as the communication protocol between them.

Improving the communication

The communication between the layers is done by D-Bus right now but should also be able to use SSL or some other network protocols so that all “data gain daemons” do not have to be on the same system but can be meshed up through a network together by the generate position daemon.

Improving the source code

Instead of using compiled in device strategies those strategies should be loadable through a plugin system.

9 Acronym directory

Acronyms	Meanings
ACM	Abstract Control Model
AJAX	Asynchronous Javascript And XML
API	Application programming interface
ARM	Advanced RISC Machine
ASP	Active Server Pages
CLHASWH	Code like hell and see what happens
CVS	Concurrent Versions System
DCOP	Desktop COmmunication Protocol
DLD	domestic location detection
DOM	Document Object Model
EULA	End User License Agreement
FIFO	First In First Out
FOSS	Free open source software
GCC	GNU Compiler Collection
GHz	Giga Hertz
GNU	GNU's Not Unix
GPL	(GNU) General Public Licence
GPS	Global Positioning System
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
ICC	Intel C++ Compiler (ICC)
IDE	Integrated development environment
IEEE	Institute of Electrical & Electronics Engineers
IP	Internet protocol
KDE	K Desktop Environment
LAN	Local Area Network
LBM	Locationbased media
LED	Light Emitting Diode
MAC	Media Access Control (address)

Continued on Next Page...

Table 9.1 – Continued

Acronyms	Meanings
MFC	Microsoft Foundation Class(es)
MFU	Music follows you
OSI	Open Source Initiative
PERT	Program Evaluation and Review Technique
PDA	Personal Digital Assistant
PHP	PHP Hypertext Preprocessor
PSD	Project Specification Document
RAD	Rapid application development
RDLDD	RFID domestic location detection
RFID	Radio-frequency identification
RPC	Remote Procedure Call
RSD	Requirement specification document
SAX	Simple API for XML
SCM	Software Configuration Managment
SQL	Structured Query Language
ssh	Secure Shell
SSL	Secure Sockets Layer
STL	Standard Template Library
STP	System Test Plan
SVG	Scalable Vector Graphics
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UML	Unified Modeling Language
UUCP	Unix to Unix CoPy
URL	Uniform Resource Locator
USB	Universal Serial Bus
WLAN	Wireless Local Area Network
XML	Extensible Markup Language
ZC	ZigBee coordinator
ZED	ZigBee End Device
ZR	ZigBee Router

Table 9.1: Acronym directory

10 References

- Alliance, Z., 2007, Zigbee specification, [Online], Available at:
http://www.zigbee.org/en/spec_download/download_request.asp [accessed 31. October 2007].
- Beck, K., 2005, Extreme programming explained, 2nd ed, USA: Addison-Wesley.
- Beck, K., 2007, Extreme programming, [Online], Available at:
<http://ootips.org/xp.html> [accessed 31. October 2007].
- Fioravanti, F., 2005, Skills for managing rapidly changing it projects, [e-book], Hershey, PA, USA: IIRM Press, Available at:
<http://site.ebrary.com/lib/staffordshire/Doc?id=10089243&ppg=88> [accessed 12. October 2007].
- freedesktop.org, 2008a, Introduction to d-bus, [Online], Available at:
<http://www.freedesktop.org/wiki/IntroductionToDBus> [accessed 25. January 2008].
- freedesktop.org, 2008b, D-bus specification, [Online], Available at:
<http://dbus.freedesktop.org/doc/dbus-specification.html> [accessed 25. January 2008].
- Hansen, H. R., 1998, Wirtschaftsinformatik 1, Stuttgart: Lucius & Lucius.
- Hentzen, W., 2002, Software developer's guide (3rd edition), [e-book], Milwaukee, WI, USA: Hentzenwerke Publishing, Inc., Available at:
<http://site.ebrary.com/lib/staffordshire/Doc?id=10110767&ppg=56> [accessed 28. October 2007].
- Hightower, R., 2004, Professional java tools for extreme programming : Ant, xdoclet, junit, cactus, and maven, [e-book], Hoboken, NJ, USA: John Wiley & Sons, Incorporated, Available at:
<http://site.ebrary.com/lib/staffordshire/Doc?id=10114229&ppg=43> [accessed 31. October 2007].

- Kowalski, M., 2008, Schnittpunkte zweier kreise, [Online], Available at:
<http://kowalski.s4f.eu/addt/2kreise.html> [accessed 1. March 2008].
- Muller, N. J., 2002, Wireless a to z, [e-book], Blacklick, OH, USA: McGraw-Hill Professional Publishing, Available at:
<http://site.ebrary.com/lib/staffordshire/Doc?id=10045472&ppg=36> [accessed 10. October 2007].
- Neundorf, A., 2006, Why the kde project switched to cmake, [Online], Available at:
<http://lwn.net/Articles/188693/> [accessed 17. November 2007].
- Trolltech, 2007, Introduction to d-bus, [Online], Available at:
<http://doc.trolltech.com/4.3/intro-to-dbus.html> [accessed 25. January 2008].
- Wikipedia, 2007a, Extreme programming, [Online], Available at:
http://en.wikipedia.org/w/index.php?title=Extreme_Programming&oldid=168038463 [accessed 31. October 2007].
- Wikipedia, 2007b, Zigbee, [Online], Available at:
<http://en.wikipedia.org/w/index.php?title=ZigBee&oldid=167861528> [accessed 31. October 2007].

A Gantt charts

A.1 First Gantt chart

A.2 Second Gantt chart

The Gantt chart had to be updated because the the research stage was finished earlier then planned. The analysis stage description also got updated with a few more details.

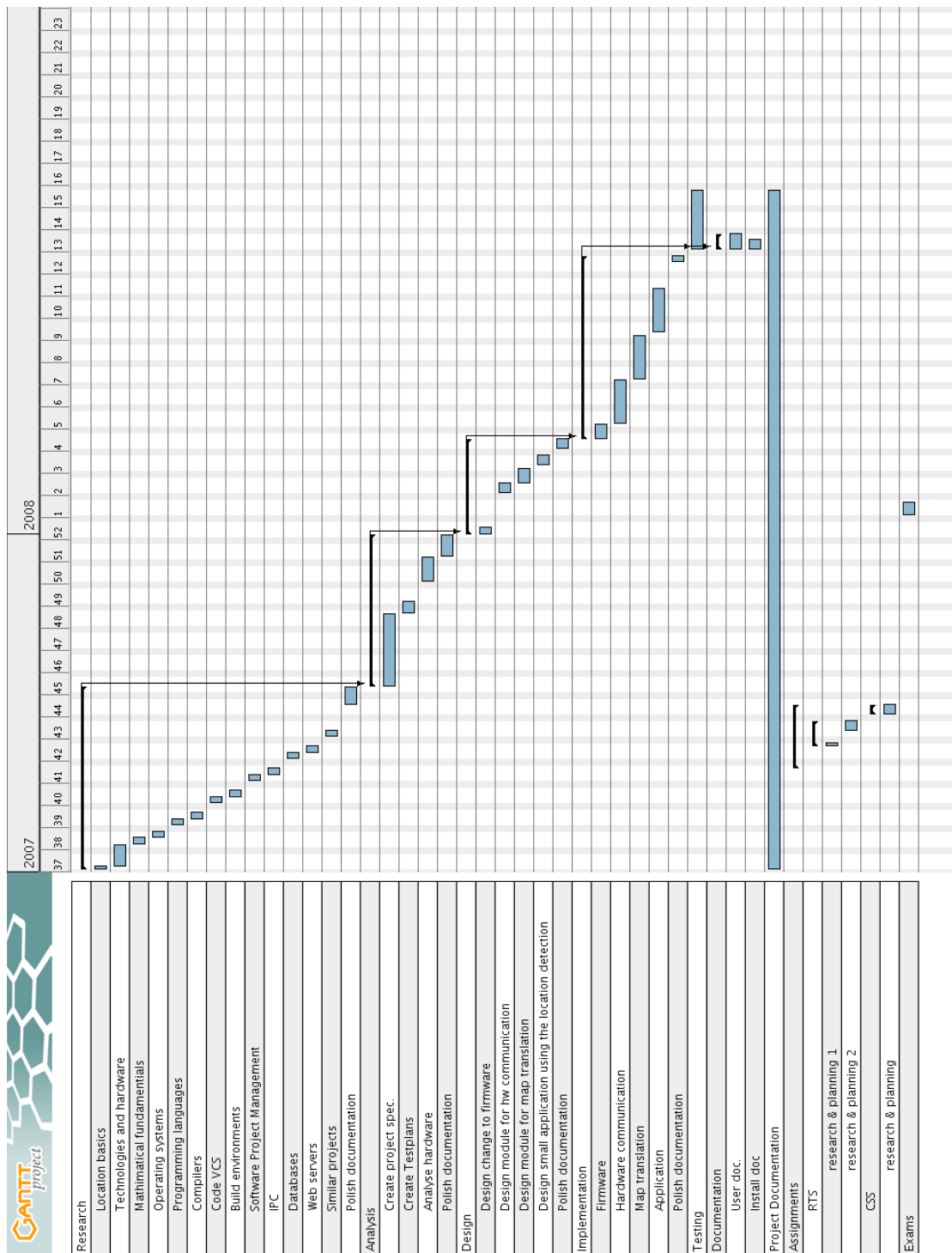


Figure A.1: Gantt chart of the planning phase

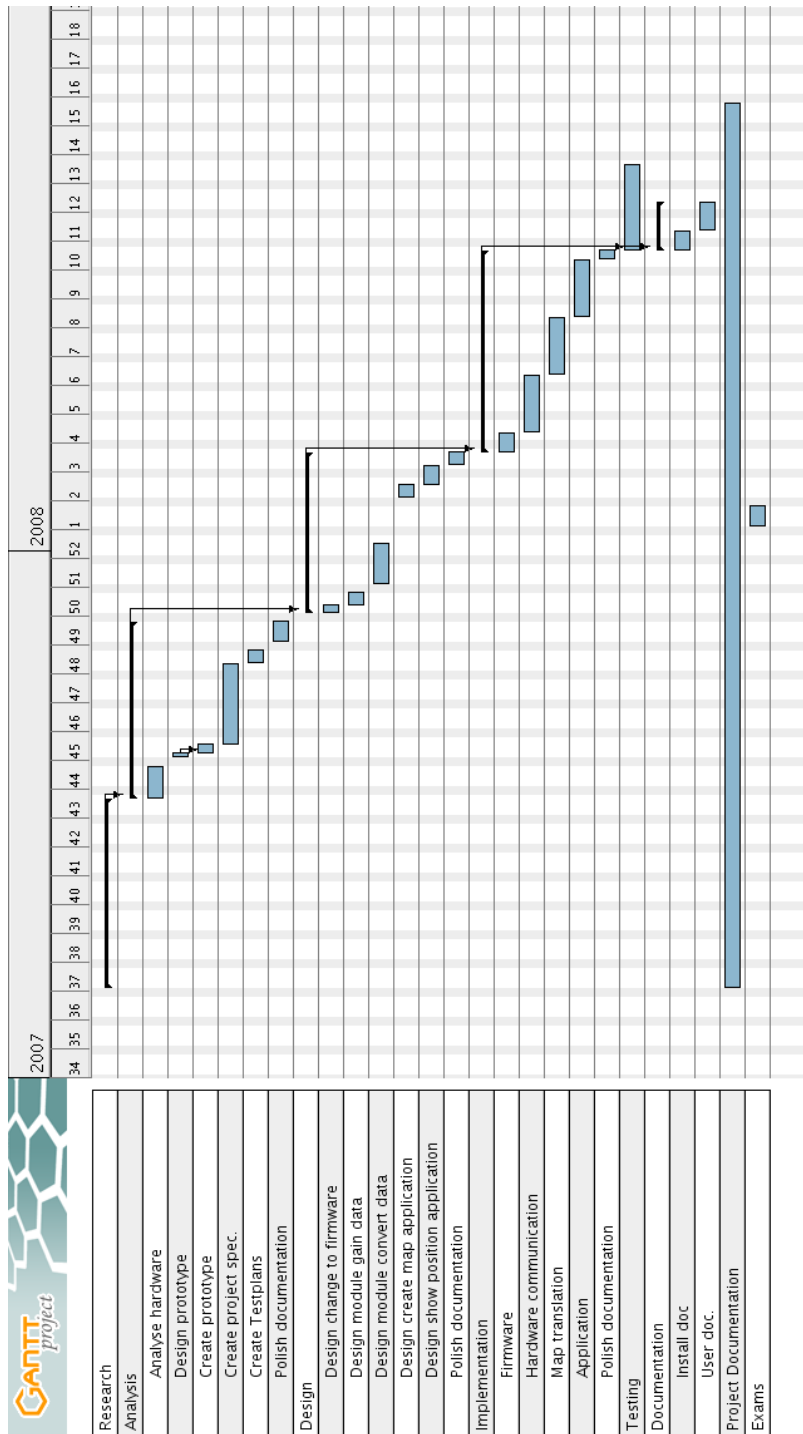


Figure A.2: Gantt chart after analysing phase

B Prototype source code

B.1 Main executable

The main executable is communicating with the OpenBeacon node and will retrieve user commands through a message queue.

```
1 #include <fcntl.h>
2 #include <termios.h>
3 #include <stdio.h>
4 #include <string.h>
5 #include <unistd.h>
6 #include "msgq.h"
7
8 #define BAUDRATE B115200
9 #define OPENBEACONMODEMDEVICE "/dev/ttyACM0"
10
11 int main(int argc, char ** argv)
12 {
13     bool run = true;
14     int fd;
15     int res;
16     struct termios newtio;
17     char buf[255];
18     int readId;
19     struct message msg;
20     int result;
21     char command[255];
22
23     // open device in rw mode, and as no controlling tty (is better
24     // because if the device sends CTRL+C it will not affect the current
25     // running console)
26     fd = open ( OPENBEACONMODEMDEVICE, ORDWR | O_NOCTTY | O_NONBLOCK );
27     if (fd < 0)
28     {
29         perror (OPENBEACONMODEMDEVICE);
30         return (-1);
31     }
32
33     // create message queue
34     if ( (readId = msgget (COMMUNICATIONKEY, PERMS | IPC_CREAT)) < 0)
35     {
36         // print error
37         return (-1);
38     }
39
40     // clear struct
```

```

41  bzero(&newtio, sizeof(newtio));
42  // setting control modes -> man 3 termios
43  newtio.c_cflag = BAUDRATE | CRTSCTS | CS8 | CLOCAL | CREAD;
44  // setting input modes -> man 3 termios
45  newtio.c_iflag = IGNPAR | ICRNL;
46  // setting output modes -> man 3 termios
47  newtio.c_oflag = 0;
48  // canonical input
49  newtio.c_lflag = ICANON;
50  // initialize all control characters
51  newtio.c_cc[VTIME] = 0; // inter-character timer unused
52  newtio.c_cc[VMIN] = 2; // blocking read until 1 character arrives
53  // clean the line and set the new settings
54  tcflush(fd, TCIFLUSH);
55  tcsetattr(fd, TCSANOW, &newtio);
56
57  while (run)
58  {
59      // read from OpenBeacon
60      res = read(fd, buf, 255);
61      buf[res] = '\0';
62      if (buf[0] != '\n')
63      {
64          printf("%s", buf);
65          buf[0] = '\0';
66      }
67
68      // read from msgq
69      result = msgrcv(readId, (void *) &msg, sizeof(msg.text), 0,
70                     MSGNOERROR | IPC_NOWAIT);
71      if (result > 0)
72      {
73          printf("Got msgq command: %s\n", msg.text);
74          if (strcmp(msg.text, "quit", 4) == 0)
75          {
76              run = false;
77          }
78          else
79          {
80              strcpy(command, msg.text);
81              strcat(command, "\r");
82              write(fd, command, 2);
83          }
84      }
85      // delete message queue
86      if (msgctl(readId, IPC_RMID, (struct msqid_ds *) 0) < 0)
87      {
88          // print error
89          return (-1);
90      }
91      close(fd);
92      return (0);
93  }

```

Listing B.1: main.cpp of prototype

B.2 Send executable

This executable is used to send commands to the main executable. Those commands are sent through a message queue

```

1 #include <stdio.h>
2 #include <string.h>
3 #include "msgq.h"
4
5 int main (int argc, char ** argv)
6 {
7     struct message msg;
8     int writeId;
9
10    if (argc != 2)
11    {
12        printf ("wrong count of arguments\n");
13        return (1);
14    }
15    if ( (writeId = msgget (COMMUNICATIONKEY, 0)) < 0)
16    {
17        return (-1);
18    }
19
20    // send command to msgq
21    msg.type = 1;
22    strcpy(msg.text, argv[1]);
23    if (msgsnd(writeId, (void *) &msg, sizeof(msg.text), IPC_NOWAIT) < 0)
24    {
25        printf ("error while sending\n");
26        return (-1);
27    }
28
29    // delete message queue
30    if (msgctl (writeId, IPC_RMID, (struct msqid_ds *) 0) < 0)
31    {
32        return (-1);
33    }
34    return (0);
35 }
```

Listing B.2: send-msg.c of prototype

B.3 Include for common information

This include file holds the informations both sources have in common.

```

1 #include <sys/msg.h>
2 #define COMMUNICATIONKEY 4711L
3 #define PERMS 0666
4
5 struct message
6 {
```

```

7 | long type;
8 | char text[5];
9 | };

```

Listing B.3: mesq.h of prototype

B.4 CMakeLists.txt build file

This section shows how simple CMakeLists.txt file, which is needed to build a project with CMake, will be.

```

1 | PROJECT ("Openbeacon Prototype")
2 | ADD_DEFINITIONS (-Wall)
3 |
4 | SET (MAINBINARY communicateWithOB)
5 | SET (SEENDBINARY commandToCommunicate)
6 |
7 | SET (MAINSRC
8 |     main.cpp
9 | )
10 |
11 | SET (SENDSRC
12 |     send-msg.c
13 | )
14 |
15 | ADD_EXECUTABLE (${MAINBINARY} ${MAINSRC})
16 | ADD_EXECUTABLE (${SEENDBINARY} ${SENDSRC})

```

Listing B.4: CMakeLists.txt of prototype

C Several source codes

C.1 Time test sources

This section holds the sources of the time test in section 2.7.

C.1.1 Java

```
1 public class test
2 {
3     public static void main (String [] argv)
4     {
5         System.out.println ("Hello World");
6     }
7 }
```

Listing C.1: main.cpp of prototype

C.1.2 C

```
1 int main (int argc, char ** argv)
2 {
3     printf ("Hello World\n");
4     return (0);
5 }
```

Listing C.2: main.cpp of prototype

C.1.3 C++

```
1 #include <stdio.h>
2
3 int main (int argc, char ** argv)
4 {
5     printf ("Hello World\n");
6     return (0);
7 }
```

Listing C.3: main.cpp of prototype

D User Manuals

All console programs can be shut down by the key combination **CTRL+C**. To test if a tool which provides data through D-Bus is working and sending the correct values, the Qt tool `qdbusviewer` can be used. The `qdbusviewer` can connect to the D-Bus interface provided by the daemons and display the data that flows through it.

D.1 Compilation

Before any of the programs can be used the programs need to be compiled. To do so you have to change into the topmost building directory (where all the application directories are located in). The directory listing should look similar to the one shown in listing D.1

```
1 CMakeLists.txt
2 common/
3 firmware/
4 gainData/
5 generatePosition/
6 obConfig/
7 pdAdmin/
8 showPos/
```

Listing D.1: Project directory list

To compile all the daemons and applications now type the following two commands:

```
cmake .
make
```

ATTENTION: After the `cmake` command is a dot and it belongs there.

If you want to enable debug flags of the applications you have to do the following:

```
cmake -DDEBUG=1 .
make
```

Only the firmware has to be compiled differently, but a pre compiled version of the current one is already in the directory. The firmware is also in need of a special `c` compiler for the arm CPU which needs to be installed on the system. To compile the firmware change into the `firmware` directory and type:

```
make
```

This will create a `openbeacon.bin` file which is the firmware and can be flashed with the OpenBeacon Configurator (see section D.7). The file with no further code changes should have a file size of 24668 byte if it differs then your compiler might have done something wrong and the firmware might not work with the hardware. So if you change the source of the firmware build it unchanged before changing it and then check if your size differs from the original.

A set of pre build firmwares is available in the directory `firmware/build/`, it contains at least the following files:

1	<code>estimator-008-dld-01.bin</code>
2	<code>estimator-008.bin</code>

Listing D.2: Pre-build OpenBeacon USB firmwares

Where the first is the one created for this project and the second one is the original one coming with the nodes.

D.2 OpenBeacon Configurator - Application

The OpenBeacon Configurator is an easy to use graphical user interface which helps the user with the configuration and the flashing of the OpenBeacon nodes.

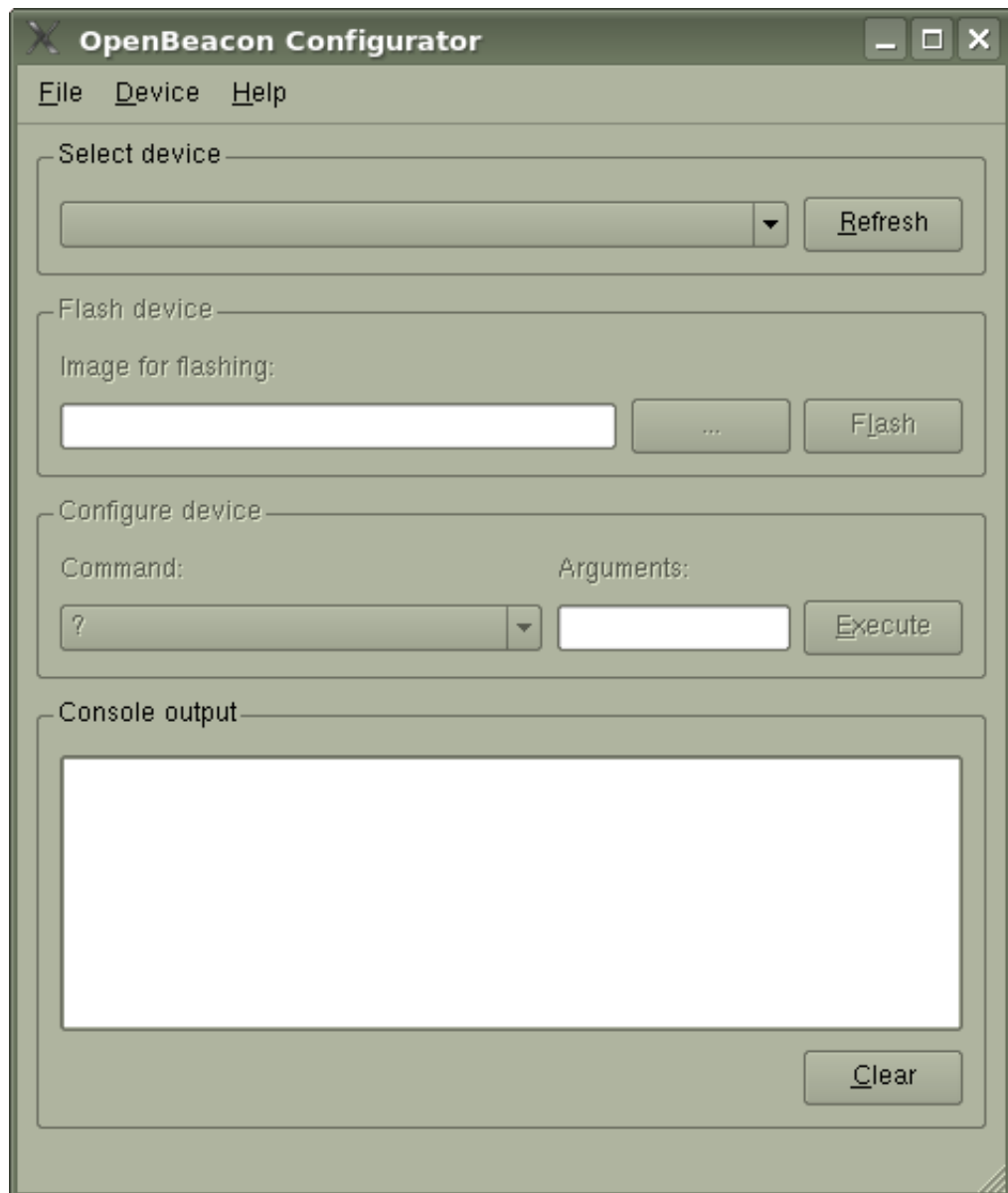


Figure D.1: The main window of the OpenBeacon Configurator

D.2.1 Pre requirements

To use the OpenBeacon Configurator a few things need to be done before.

1. The kernel must be capable of seeing the devices, therefore the following modules need to be enabled:
 - usbserial
 - cdc-acm

If they do not exist they have to be enabled in the kernel. For further notes have a look into your kernel sources and the listings 2.1 and 2.2 (both located

in the research chapter) to get a clue where to find them.

2. Install the sam7 package provided by the OpenBeacon project. You can download it from here: <http://www.openpcd.org/dl/sam7utils-0.1.0-bm.tar.bz2>.
3. While the cdc-adm module is just loaded the usbserial module needs to be loaded with certain parameters. With the following command the usbserial module gets told that it should be used for a specific product from a specific vendor:

```
modprobe usbserial vendor=0x03eb product=0x6124
```

D.2.2 Flashing a device

To flash a device, the prior system on it must be erased. To do so the following steps need to be performed:

1. Unplug the USB cable and insert the SAM-BA jumper (Pin 1+2)
2. Attach the USB cable
3. Wait ten seconds
4. Unplug the USB cable
5. Remove the SAM-BA jumper
6. Attach the USB cable
7. Wait several seconds to allow the device to be detected by Linux

After that a ttyUSB device should show up in the device list.

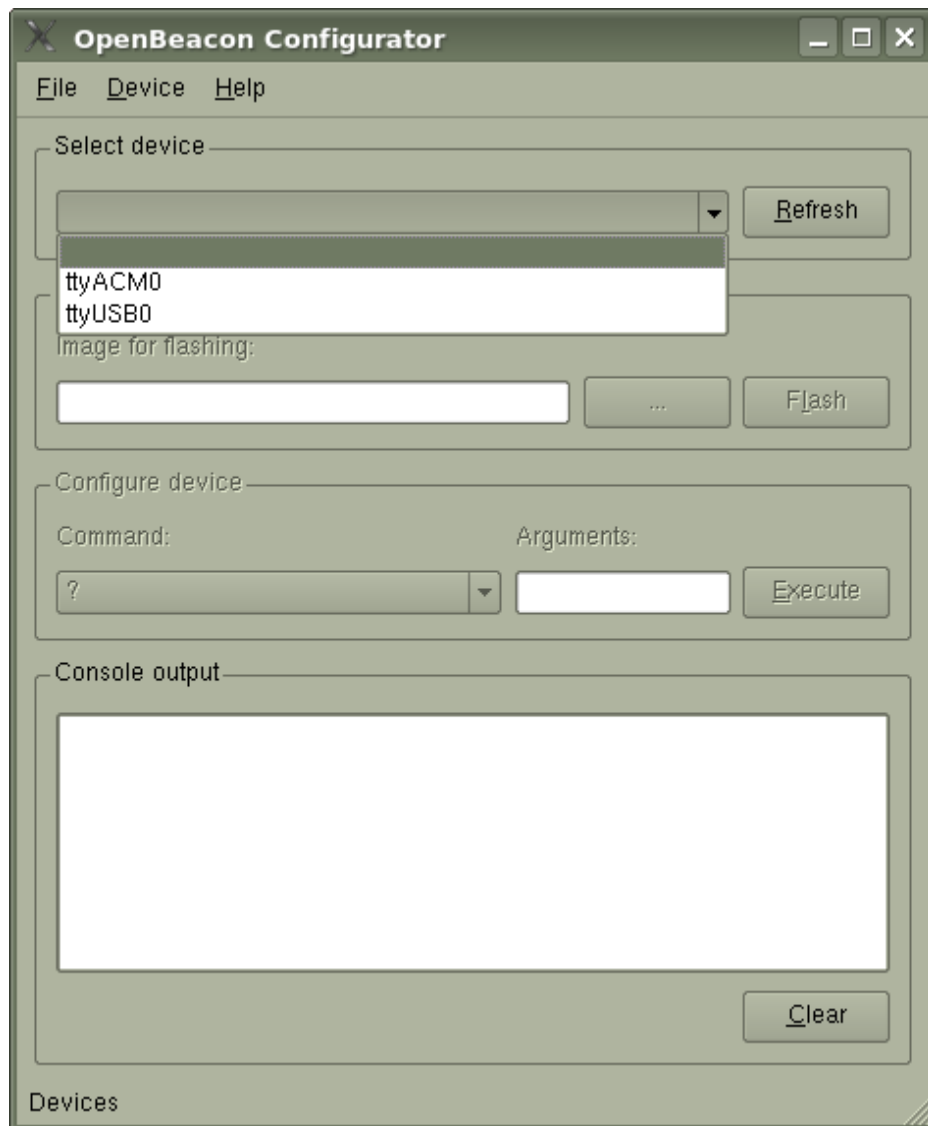


Figure D.2: OpenBeacon Configurator with expanded device selection

Figure D.2 shows that the OpenBeacon configurator has both devices available. For flashing select the `ttyUSB` device. After that the flash section will be available for editing and an image for flashing can be chosen. The button with the three dots opens a file dialog from which a suitable file can be chosen. After selection the screen should look like in figure D.3.

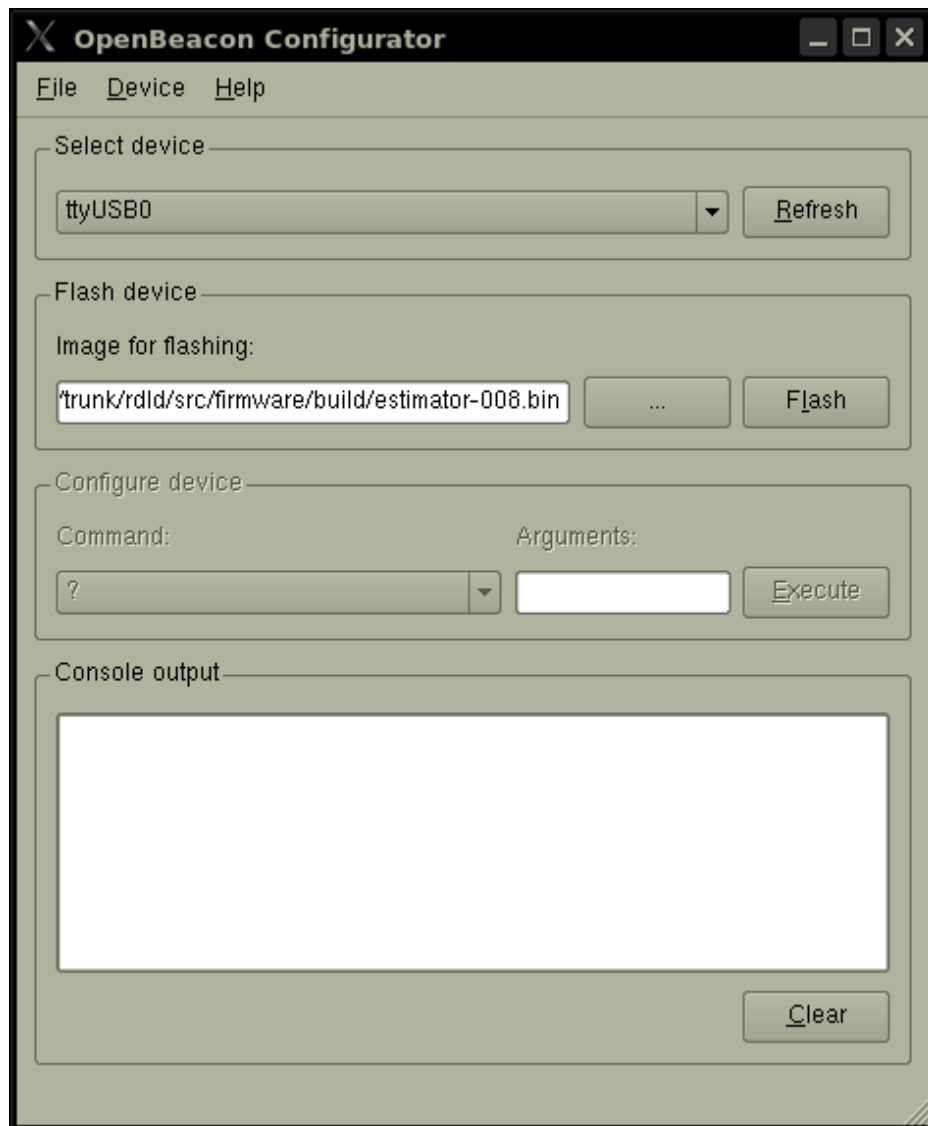


Figure D.3: OpenBeacon Configurator with selected image

When the selection of the image is finished the flashing can be started by pressing the Flash button. The flashing then will interrupt the application until the whole flashing is done. After the flashing the main window should look like in figure D.4.

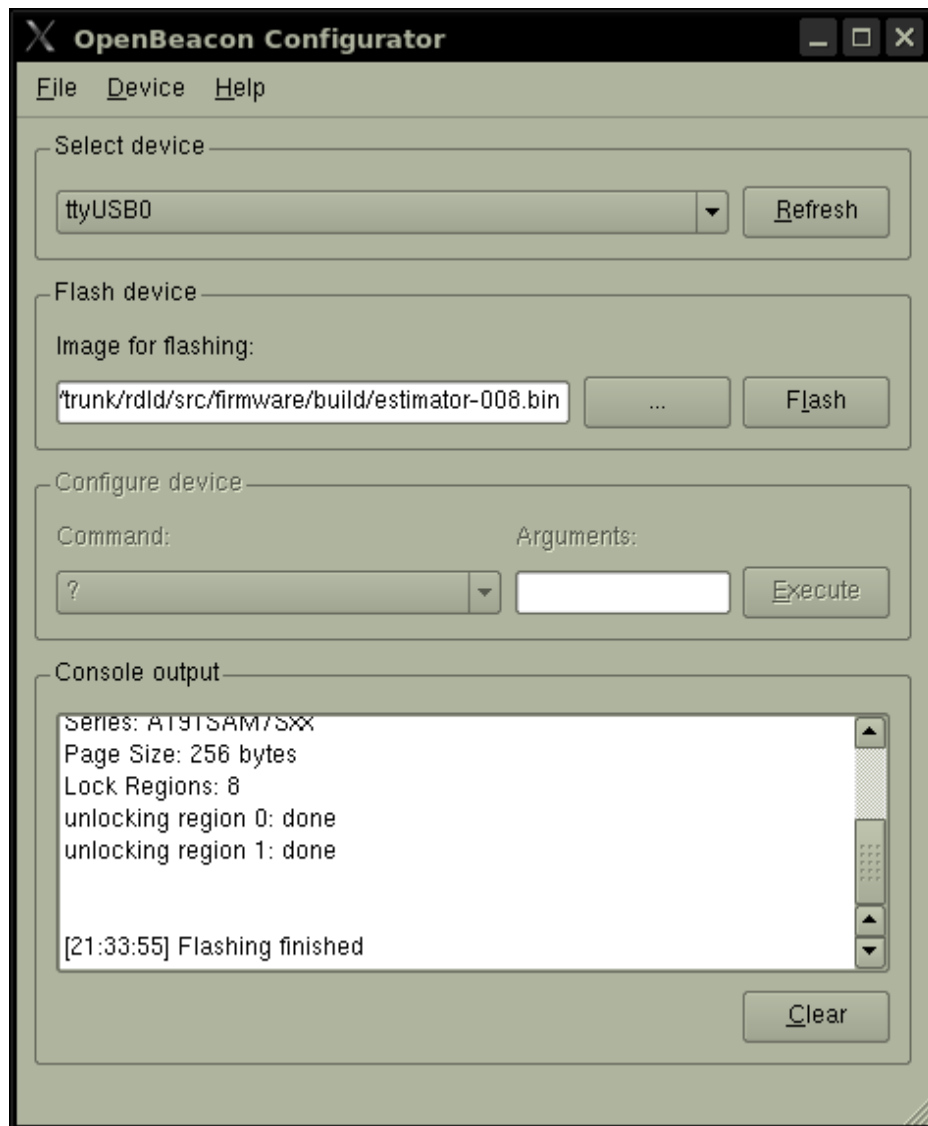


Figure D.4: OpenBeacon Configurator after flashing

D.2.3 Configuring a device

To configure a device select it from the “Select device” pull down menu. An already flashed device is one of the ttyACM ones; if there are only ttyUSB devices then one of the devices needs to be flashed before (see D.2.2). If the correct device is selected the screen looks similar to the following:

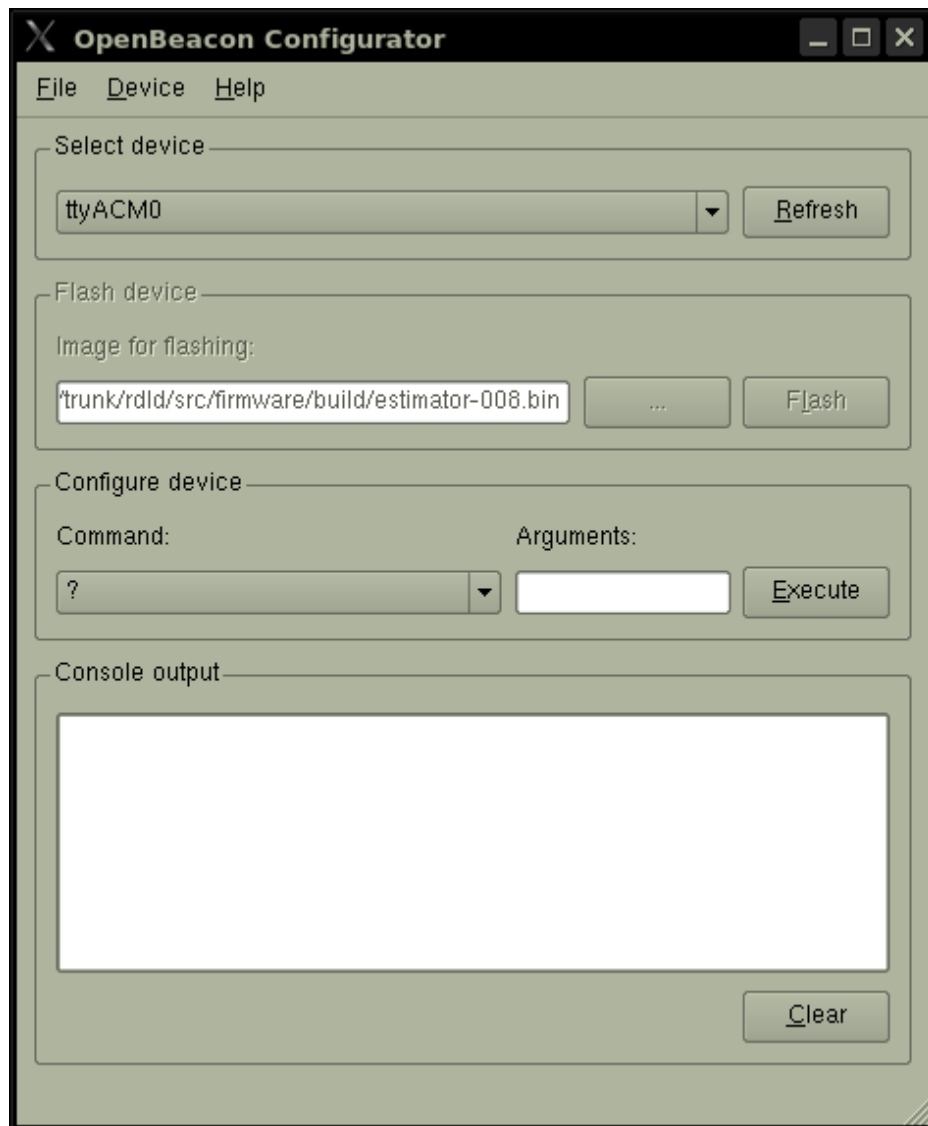


Figure D.5: OpenBeacon Configurator before command execution

The first action we perform is a command without arguments, the “?” command, which will display the help screen of the firmware. Figure D.6 shows the main window with an executed “?” command.

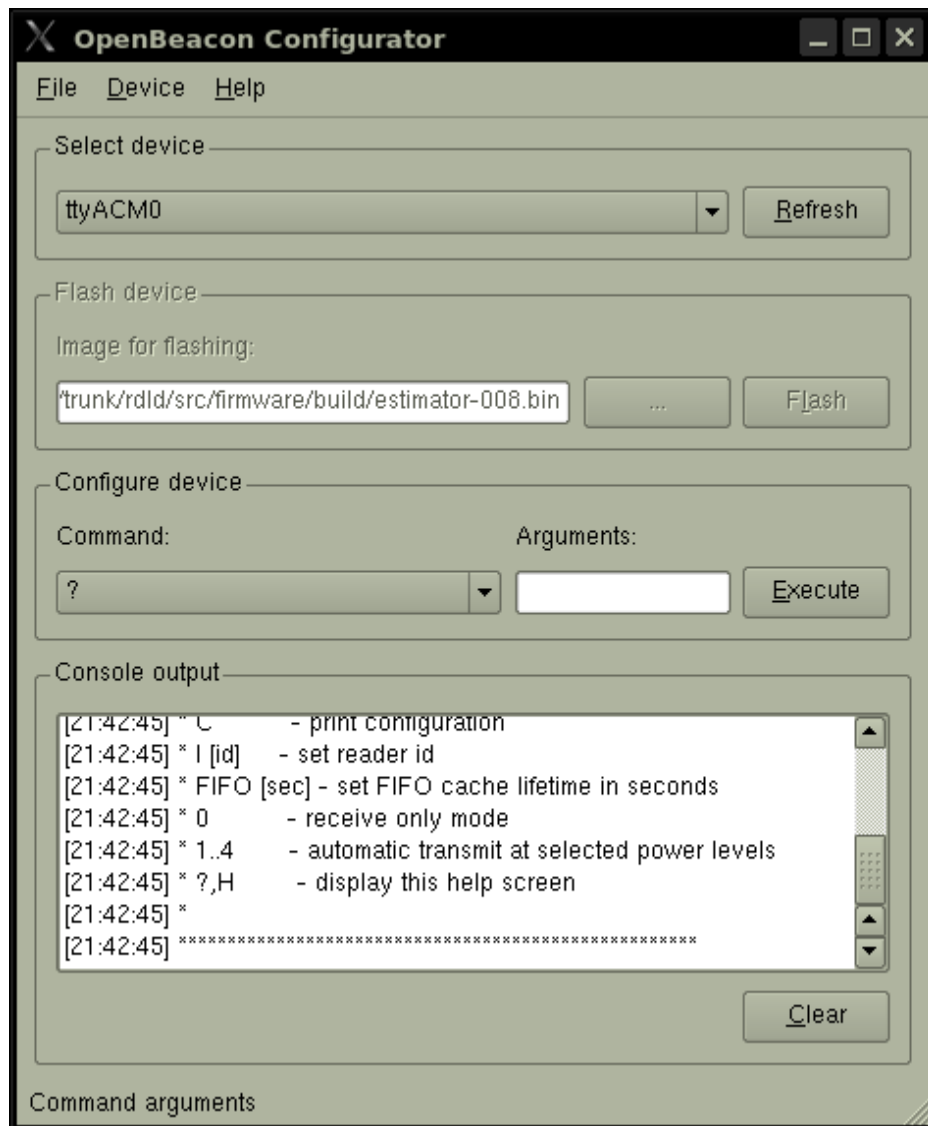


Figure D.6: OpenBeacon Configurator after command execution

The second action is a command with an argument. The “I” command which changes the current id of the OpenBeacon node, it also requires a number as argument, which defines the new id of the node. Figure D.7 shows the execution of this command with the argument “4”.

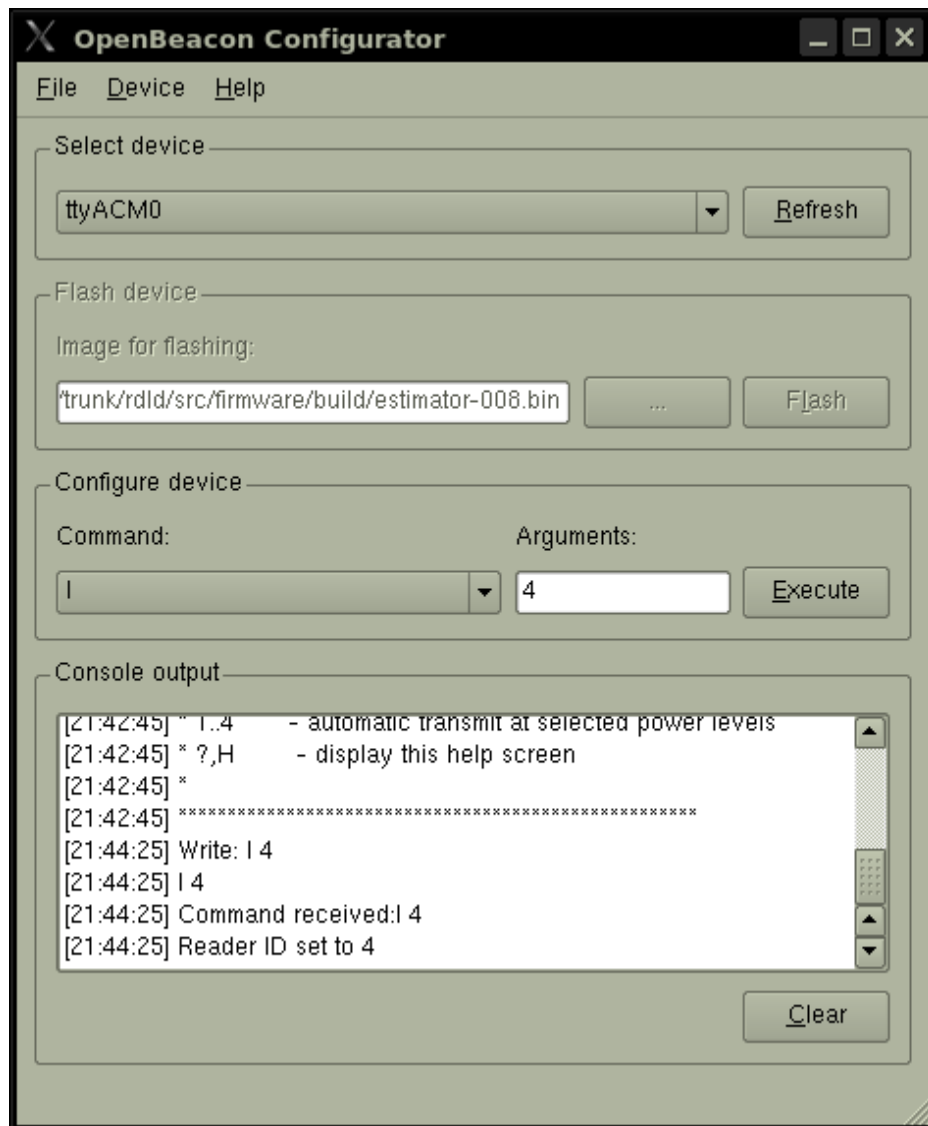


Figure D.7: OpenBeacon Configurator after command execution with an argument

D.2.4 Personalise the configurator

To personalise the “OpenBeacon Configurator” it provides a preference dialog. Figure D.8 shows the initial preference dialog. It is filled with the default values in this case only the help command.

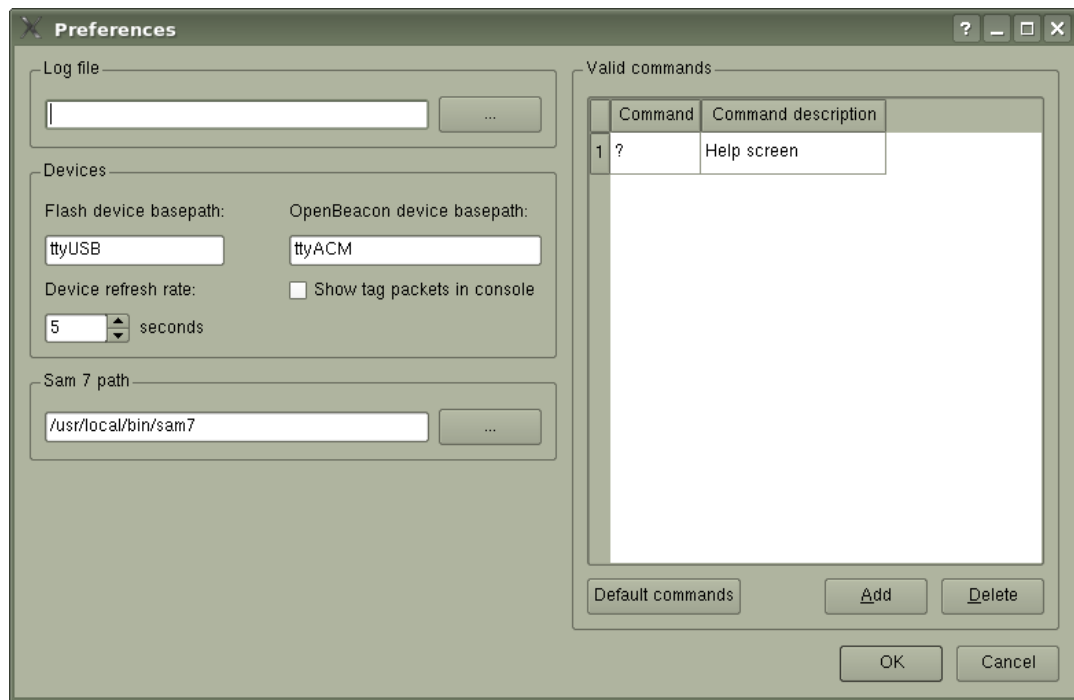


Figure D.8: OpenBeacon Configurator preference dialog

On the left part of the preference dialog are path and GUI specific preferences. The log file path defines the file where the logging should be redirected instead of the console. The sam7 path is the path to the sam7 binary. And the device specific parameter defines the base names of the different devices, the update interval in which the device pull down should be refreshed and a check box with whom the periodic information about the packet loss from the tags can be suppressed.

The right part of the dialog contains the commands that should be available for the “OpenBeacon Configurator”. The “Default Commands” button opens a dialog (see figure D.9) where you can choose which command collection should **replace** the current list.

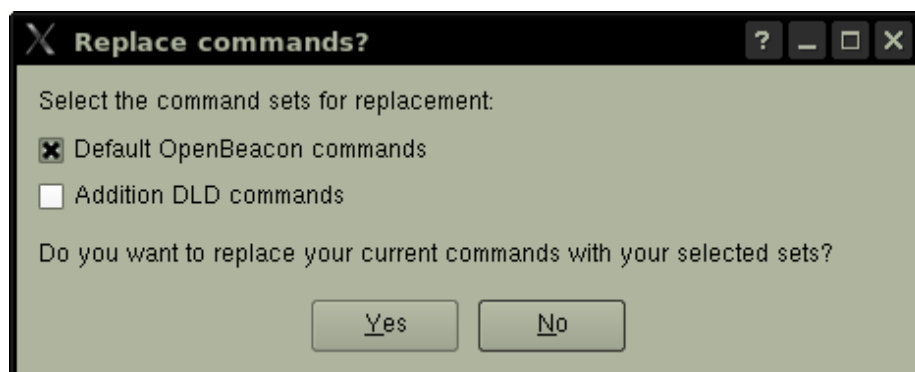


Figure D.9: Default Commands Dialog from the preferences

To add a single command simply click the “Add” button and an empty line will be inserted at the top of the table (see figure D.10).

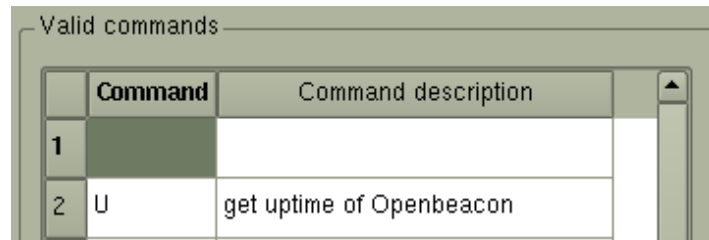


Figure D.10: Empty entry added to command list

After the empty line is added it can be edited by double clicking the cell (see figure D.11). Both fields need to be filled.

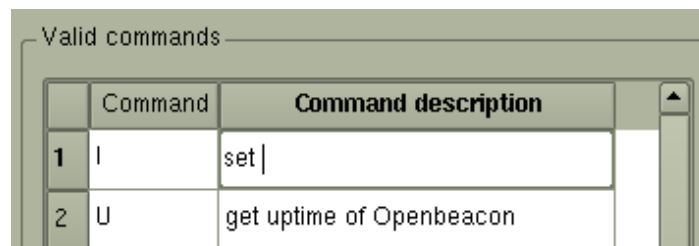


Figure D.11: Recently added entry will be changed

To remove a command simply select a cell from the command table and click the “Delete” button, it will delete the whole entry.

If the preference dialog is closed through the “OK” button all changes will be stored permanently.

D.3 Data gain - Daemon

The data gain daemon gains the strength information from the hardware devices and provides them to other applications.

D.3.1 Configuration of the daemon

The configuration file is located at the following position: `/.config/DLD/Gain data daemon.conf`. Right now it has only one configuration parameter. The *used-*

Parameter name	Description	Possible values (if applicable)
usedDeviceStrategy	describes the hardware strategy that will be used	Hardware Simulator: DLDSimulateStrategy OpenBeacon Hardware: OpenBeaconUSBStrategy

Table D.1: Configuration values for the configuration file

DeviceStrategy parameter is located in the General section of the configuration file.

D.3.2 Command line parameter

Parameter name	Value	Possible values	Description
-v	value	0 1 2 3	The level of verbosity Quiet mode - print nothing Error mode - print errors (default) Info mode - adds info messages Debug mode - adds debug messages
-l	logFile		File name where the messages should be logged to instead of console
-s	strategy	OpenBeaconUSBStrategy DLDSimulateStrategy	Use OpenBeacon USB strategy Use Hardware Simulator

Table D.2: Command line parameters for the data gain daemon

D.3.3 Configuration and start up examples

With configuration and no command line parameters

If the configuration file contains the same as listing D.3 shows, the daemon will start the hardware simulator.

```

1 [ General ]
2 usedDeviceStrategy=DLDSimulateStrategy

```

Listing D.3: Example Configuration file for the data gain daemon

After the configuration is done the data gain daemon can be simply started with the following command:

```
gainData
```

As soon as the daemon is ready it should display the hardware simulator. For further instructions look at the user manual for the hardware simulator in section D.7.

Only command line parameters

If no configuration file is available the used strategy will be chosen by command line or if even this is not given the default value (`OpenBeaconUSBStrategy`) will be used.

To start the gain data daemon showing debug messages to the console and use the “Hardware Simulator” strategy, the command is the following:

```
gainData -v 3 -s DLDSimulateStrategy
```

The selected strategy will be stored in the configuration file as soon as the daemon will be shut down.

No configuration and no command line parameters

If the data gain daemon is started without command line parameters nor an existing configuration file is existent, the daemon only shows error messages printed to the console and uses the `OpenBeaconUSBStrategy` as the default hardware strategy. The command for this will be:

```
gainData
```

After the gain data daemon is closed the configuration file will be created and the default strategy will be stored.

Configuration of the “OpenBeacon USB Strategy”

The configuration file is located at the following position:

```
/.config/DLD/obUSBStrategy.conf.
```

The *maxPackagesOnOB* parameter is located in the General section of the config-

Parameter name	Description	Possible values (if applicable)
maxPackagesOnOB	Maximum amount of packages an OB device stores per sec (FIFO)	1-30
id	The nodes id	
x	The X Coordination of the node	
y	The Y Coordination of the node	
z	The Z Coordination of the node	

Table D.3: Configuration values for the OpenBeacon USB Strategy

uration file. The other parameters, *id*, *x*, *y*, and *z*, are located in a section called OpenBeacon-N, where the N is the number of the configured node. The numbers have to start from 0 and every following node has to have an increased number of its predecessor.

Example configuration for the “OpenBeacon USB Strategy”

Listing D.4 shows a configuration example for 3 nodes, who all have a maximum of 30 packages to loose per second.

```

1  [ General ]
2  maxPackagesOnOB=30
3
4  [ OpenBeacon -0 ]
5  id=5
6  x=0
7  y=0
8  z=0
9
10 [ OpenBeacon -1 ]
11 id=24
12 x=-3.5
13 y=0
14 z=0
15
16 [ OpenBeacon -2 ]
17 id=2
18 x=0
19 y=15.5
20 z=0

```

Listing D.4: Example Configuration file for the OpenBeacon USB Strategy

The higher the `maxPackagesOnOB` (the OpenBeacon FIFO value) value the higher the accuracy of the tags will be.

D.4 Generate position - Daemon

The generate position daemon calculates the position of a tag on base of the strength values the data gain daemon is providing. Therefore it only works after the data gain daemon is configured and runs.

D.4.1 Configuration of the daemon

Right now the generate position daemon has no configuration values. At the current state the only available strategy is a two dimensional location detection. But later on it might be possible that a three dimensional location detection can be implemented.

Parameter name	Value	Possible values	Description
-v	value	0 1 2 3	The level of verbosity Quiet mode - print nothing Error mode - print errors (default) Info mode - adds info messages Debug mode - adds debug messages
-l	logFile		File name where the messages should be logged to instead of console

Table D.4: Command line parameters for the generate position daemon

D.4.2 Command line parameter

D.4.3 Example start

```
1 genPos
```

Listing D.5: Example start of the generate position daemon

```
1 genPos -v 3
```

Listing D.6: Example start of the generate position daemon with higher verbosity

Listing D.5 shows a general start up of the client and listing D.5 shows an example start but with a higher verbosity level. The higher verbosity level will enable a lot of messages that will be sent to the console.

D.5 Administrate person data - Application

The person data administration application is used to add persons to the persons database. As pre requirement a database needs to be created. See section 5.3.6 for further information how to create the database.

After the start of the application it presents itself with the following screen:

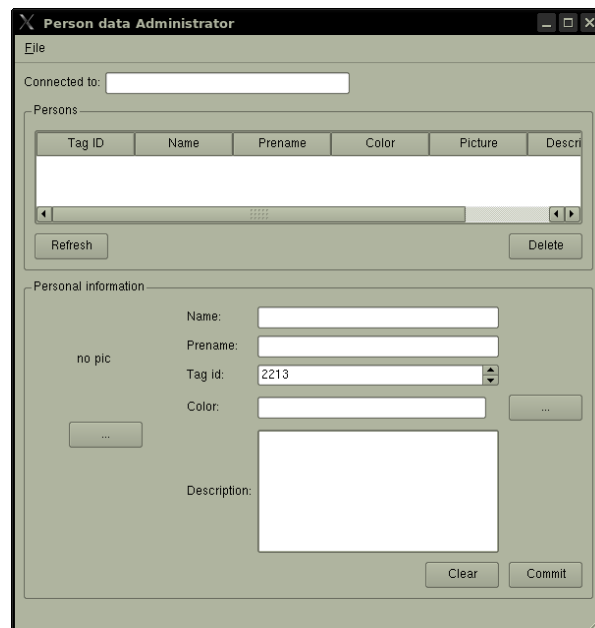


Figure D.12: The main window of the Administrate person data application

Before any of the functionality is useful, the “Person Data Administrator” needs to be connected to the database. To connect to the database press the key combination **CTRL+C** or use the menu like shown in figure D.13.

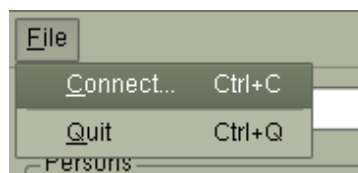


Figure D.13: Select connect from file menu

As soon as connect was selected a dialog for the connection will appear (see figure D.14). The dialog will be refilled by the values that were used in an old session if this is the first time the “Person Data Administrator” is launched all fields will be empty. After the connection was successful the contents of the main screen will change. The topmost field shows the name database, the application is connected to.

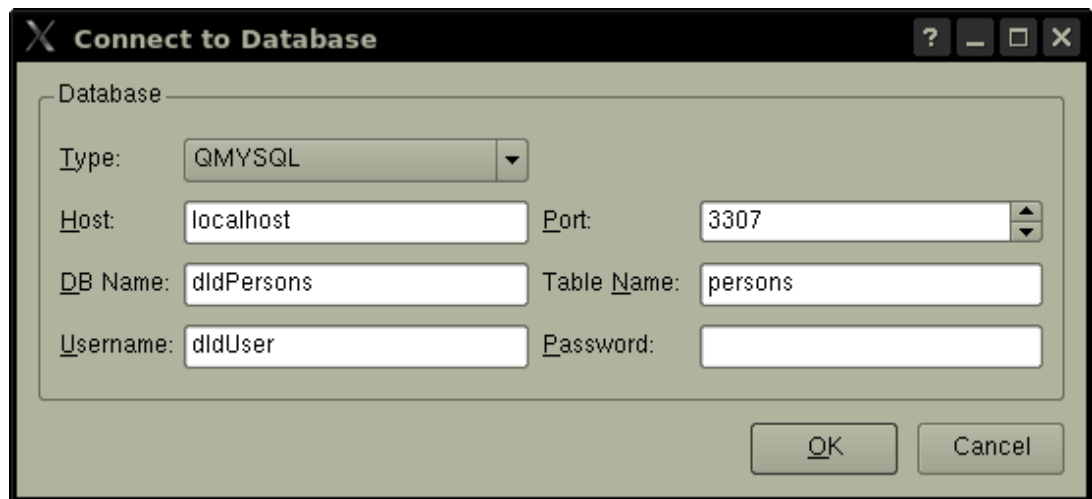


Figure D.14: Database connection dialog

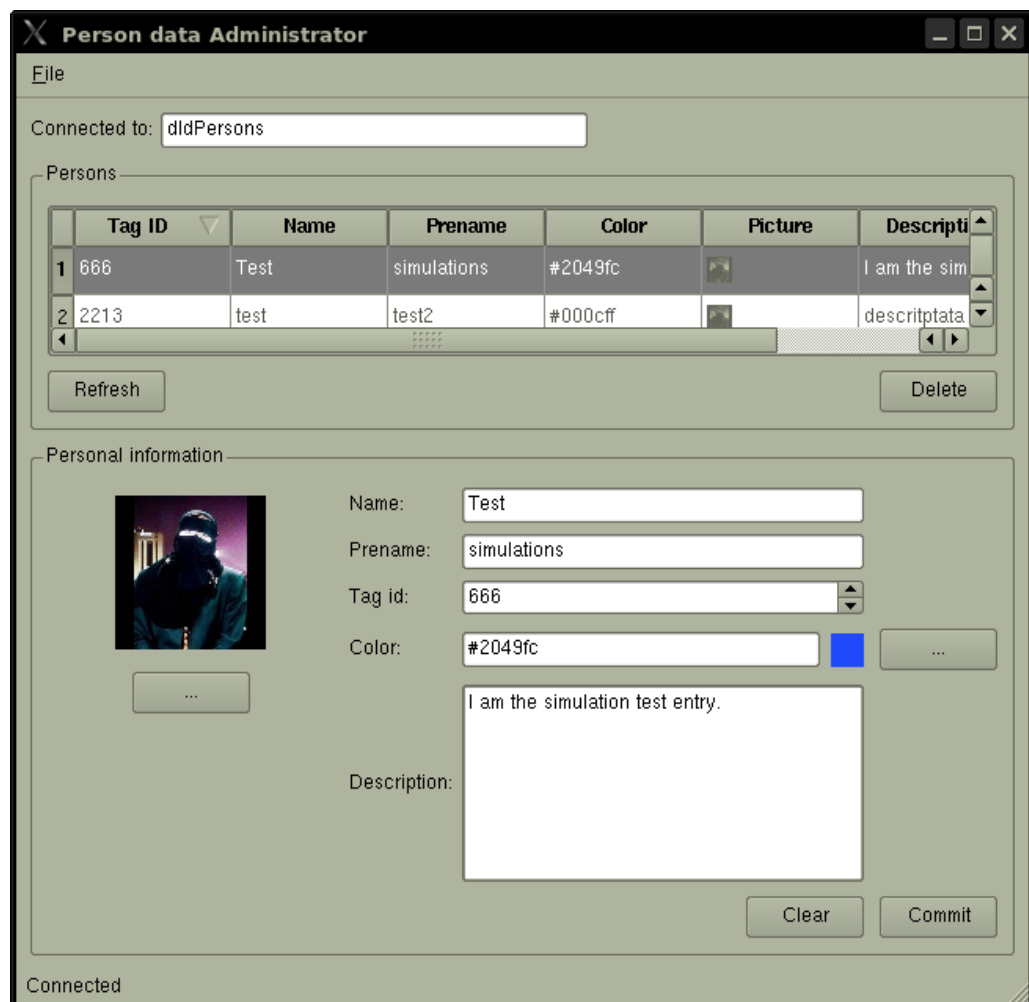


Figure D.15: Main screen with selected entry

Below is a list of the entries that are stored in the database and the lowest group box (Personal Information) stores a detailed view of the selected entry. If an entry

is selected the data can be changed in this group box as well. To store the changes the “Commit” button needs to be pressed.

To create a new entry the Personal Information group box needs to be filled with new data, the tag id will be used as the key value, therefore only the tag id must be different from the other entries. To clear all fields (i.e. you want to create a new entry) press the clear button. This will not delete an entry it just clears all the fields. The “...” button right next to the colour field will open a dialog (see figure D.16) which allows the selection of the colour.

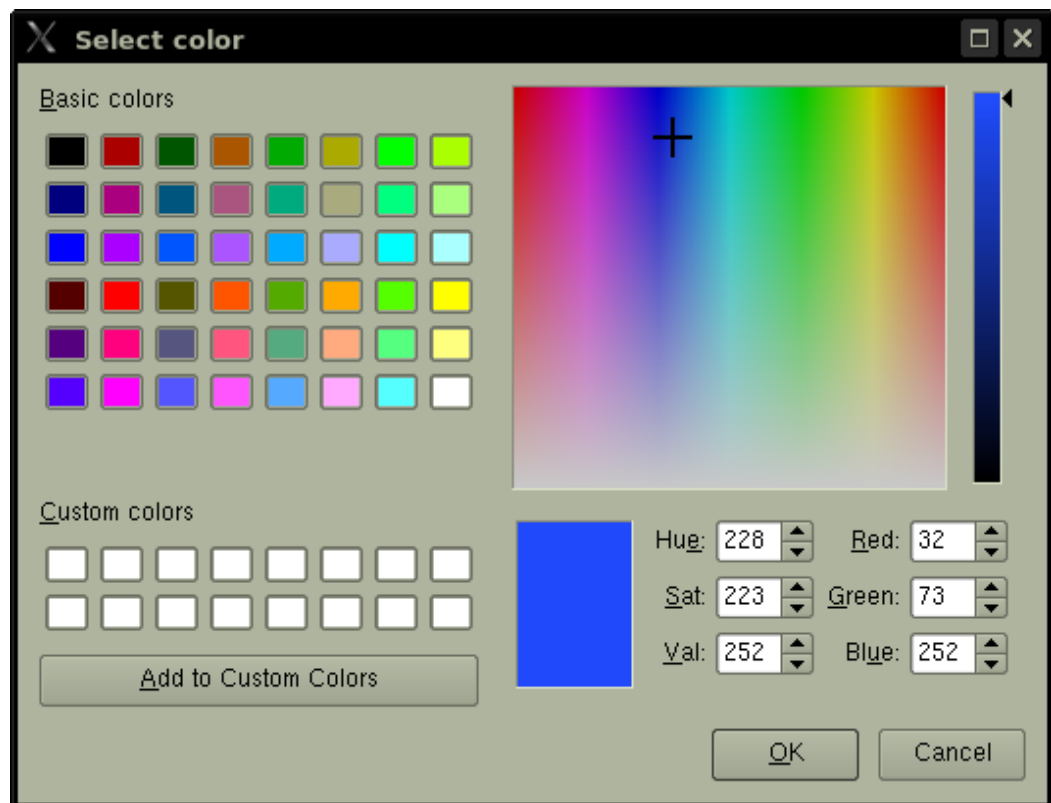


Figure D.16: Colour selection

The “...” button below the picture on the other hand opens a dialog (see figure D.17) to select an image which will represent the person.

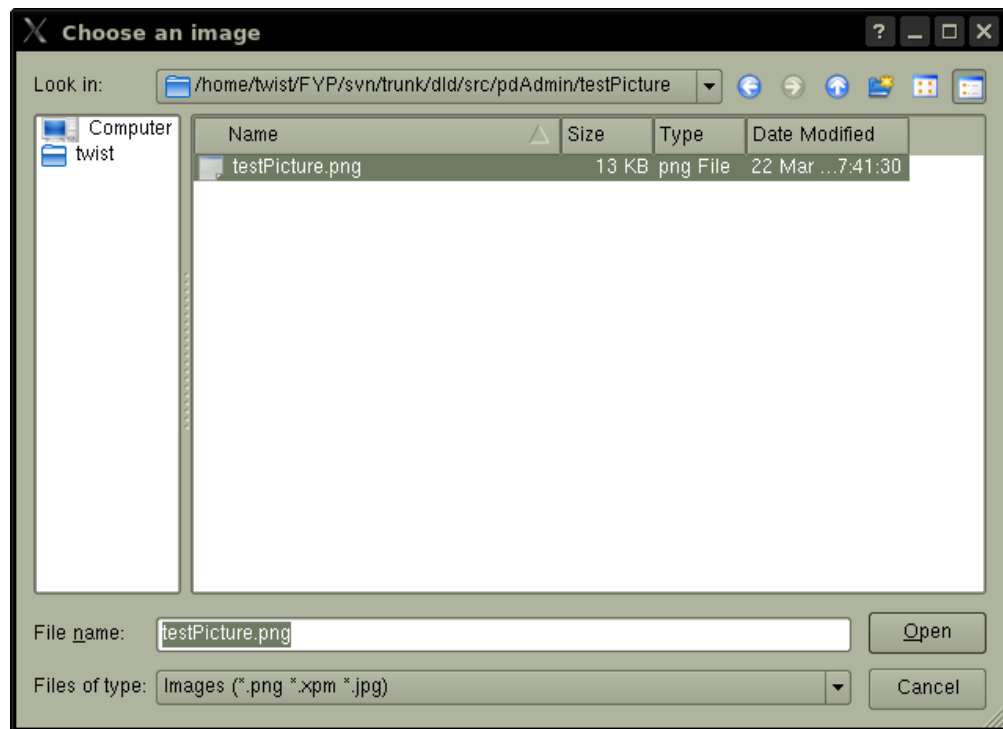


Figure D.17: Picture selection

If an entry should be deleted the entry needs to be selected and then a click on the delete button will remove the entry permanently from the database.

D.6 Show position - Application

Right after the “Show position” application is launched it presents itself like shown in figure D.18.

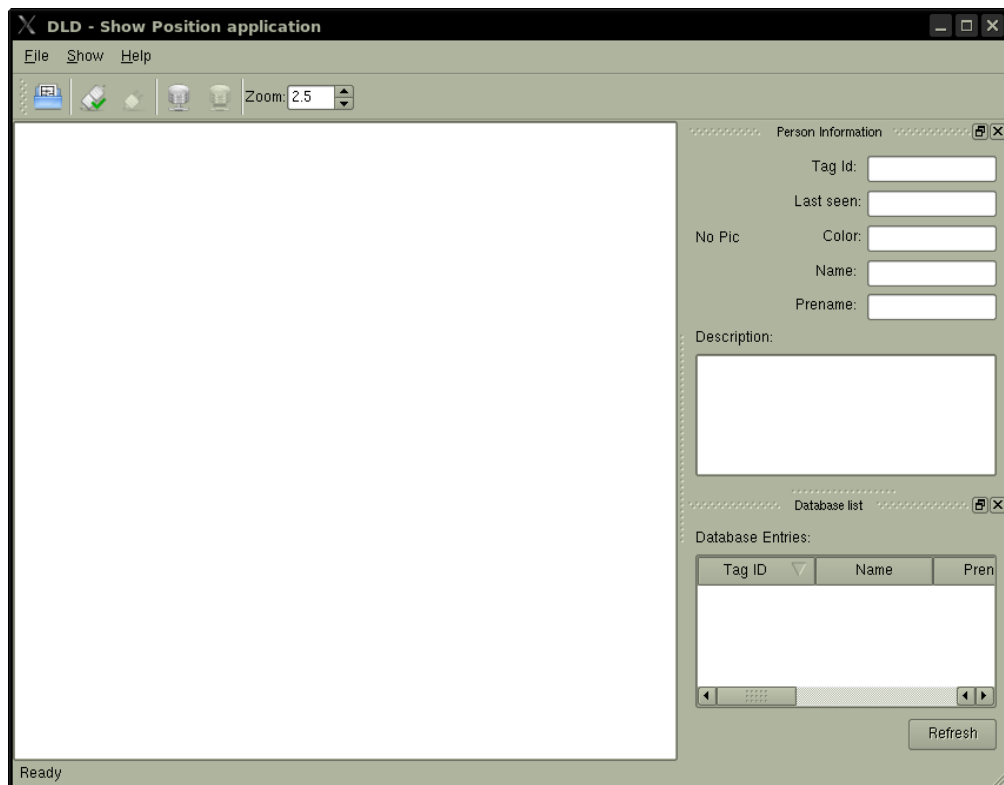



Figure D.18: Main window of show position application

The centre widget will be used to display the map and the tags. On the right side are two dock widgets, one shows the person information of the current selected tag and below is a widget which displays the entries that are loaded from the database.

At first the “show position” application should be connected to the database (if it is available), this action can be performed through three different ways. the first one is to click the “Connect to Database” toolbar button , the menu: **File->Connect to database**, or the key combination **CTRL+S**. After requesting the connection it will open the same database connection dialog like it was used in the “Administrative person data” application (see figure D.14). After the connection is built up the “Database entries” widget contains a list of the entries (see figure D.19).

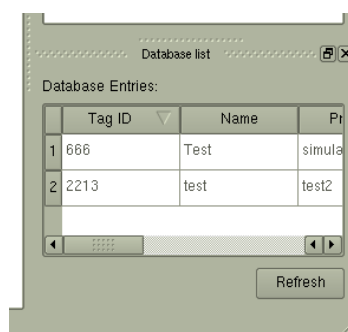



Figure D.19: Database entries widget filled with a few entries

After the connection to the person information database is built up the “show position” application needs to be connected to the “generate position daemon”. To perform this action there are again three ways to do it, a button  at the toolbar, the menu: **File->Connect to Generate Position** or the key short cut **CTRL+C**. This action will pop up a connection dialog where you have the choice how you want to connect to the “generate position daemon”, through D-Bus or through SSL. SSL is not working at the moment because it was not implemented, so the only working choice is the D-Bus connection.

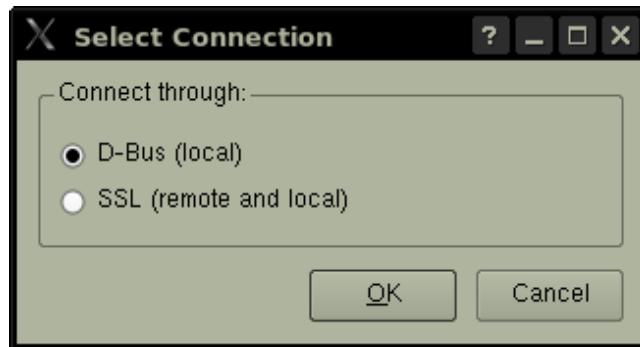


Figure D.20: Connect to Generate position Dialog

Right after the connection to the generate position daemon is built up, the main screen will show a coordination system and if some tags are near the nodes, they will show up as well. The following figure (D.21) shows this state with a blue tag.

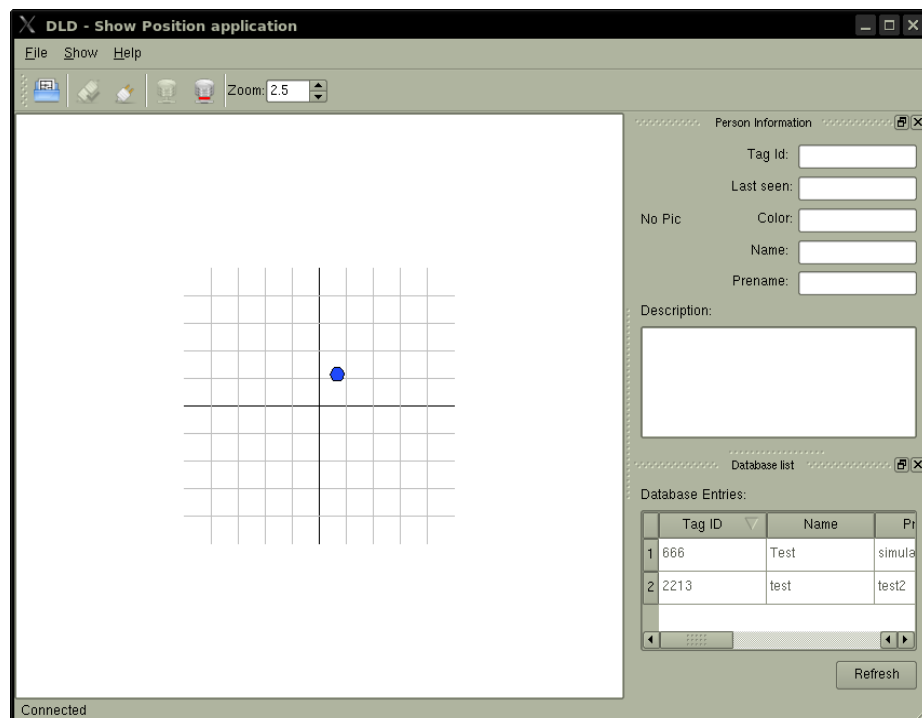


Figure D.21: Main window with connections build up and a visible tag

If the mouse will be moved over the blue tag, the person information that is stored in the database will be displayed in the person information widget as well as on a widget that appears right next to the mouse cursor. Figure D.22 shows this widget with the entry for tag id 666.

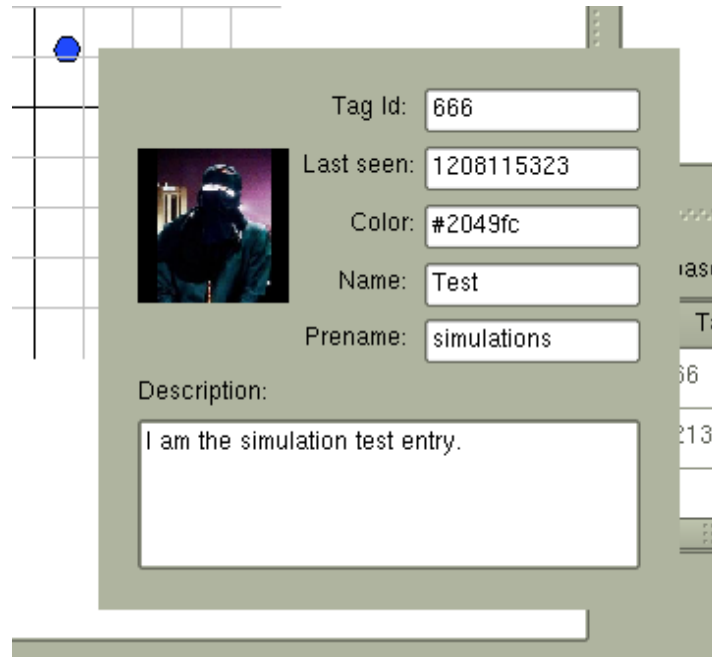


Figure D.22: Widget that appears on mouse over

To show or hide the widgets the check boxes in the **Show** menu can be selected or unselected.

D.7 Hardware Simulator - Application

The hardware simulator is launched by the data gain daemon. It will open a user interface that emulates the existence of three nodes and the strengths each of them is receiving. In addition it shows the current data in a graphical representation. Figure D.23 shows the main screen of the hardware simulator. The whole simulator consists of this one screen. The upper group box contains the coordinations and the strength data (radius) of nodes. The box below the coordinates contains general information. The update interval is the time (in seconds) the data nodes emit the data about their radii. The “Maximum Node range” is the length of one axis in one direction, the change of this value will take effect after the restart of the application. The lowest widget displays the different radii of the nodes (red for the first node, green for the second one and blue for the third node). Through the zoom value the view can be enlarged or shrunken.

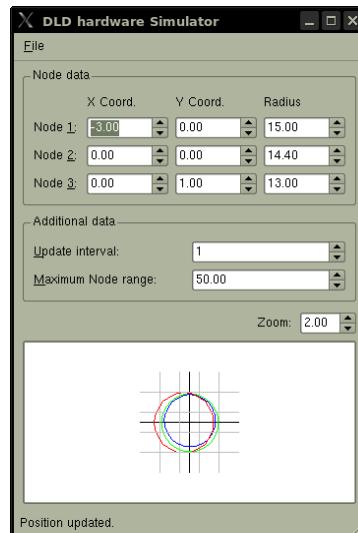


Figure D.23: hardware simulator main screen

The hardware simulator can be closed by clicking **File -> Quit** or pressing the key combination **CTRL+Q**, this will also close the data gain daemon.

E Logbook

Level 3 CDS and BCIT

Project Logbook

Notes on use of the project logbook

1. The student and supervisor must arrange regular supervisory meetings to review progress and make plans for the project. It is the purpose of the Project Logbook to document these meetings and therefore build up a record of the student's progress throughout the project;
2. The student should prepare for the supervisory meeting by deciding which questions he or she needs to ask the supervisor and what progress has been made since the last meeting (if applicable) and noting these in the relevant sections of the sheet, effectively forming an agenda for the meeting;
3. The business of the meeting should be noted briefly as items in the relevant section of the sheet. There will be one sheet for each supervisory meeting and the actions on the student (and perhaps the supervisor) which should be carried out before the next meeting should be noted briefly in the relevant section of the sheet;
4. The Project Logbook is one of the deliverables from the project and is an important record of the student's organisation and learning experience. The student should ensure that it is handed in as an appendix of the report, with sheets dated and numbered consecutively to show a consistent record of the supervisory meetings;

Logbook from week 2

Student's family name:	Student's other names:	E-mail/user-name:
Schäfer	Simon Philipp	su002988
Project title :		
RFID domestic location detection		
Items for discussion (noted by student before supervisory meeting): 1. How to reference other products? 2. Do I need a preface in my project report? 3. Do I need an abbreviation directory?		
Record of discussion of supervisory meeting: 1. Discussed my questions 2. Which ethics form should I use?		
Action list (to be attempted or completed by student by the next supervisory meeting): 1. Create a Gantt chart 2. Do some research about methodologies		
Further notes by supervisor or student (if applicable): Supervisor next week not available		

Logbook from week 4

Student's family name:	Student's other names:	E-mail/user-name:
Schäfer	Simon Philipp	su002988
Project title :		
RFID domestic location detection		
Items for discussion (noted by student before supervisory meeting): 1. Project layout question: What should be the maximum depth of the table of contents? 2. Handed in Ethics form		
Record of discussion of supervisory meeting: 1. Should use short ethics form, supervisor will email me a link. 2. Table of content should display what should be of interest but not too detailed 3. Methodology, what I have in mind, which I plan too choose. 4. Gantt chart, forgot to bring it to the meeting, have to mail it 5. I should write a small conclusion at the end of each research item		
Action list (to be attempted or completed by student by the next supervisory meeting): 1. Choosing a Methodology 2. Mail Gantt-chart to supervisor 3. Do more research for project		
Further notes by supervisor or student (if applicable):		

Logbook from week 5

Student's family name:	Student's other names:	E-mail/user-name:
Schäfer	Simon Philipp	su002988
Project title :		
RFID domestic location detection		
Items for discussion (noted by student before supervisory meeting): 1. Does the Logbook need to be handwritten? 2. Do I need more detail in my Gantt chart?		
Record of discussion of supervisory meeting: 1. Gantt chart and how to break it down, also add the dates of the assignments and exams 2. Discussion about my modified version of the waterfall methodology		
Action list (to be attempted or completed by student by the next supervisory meeting): 1. Detailed Gantt chart (mailed to supervisor) 2. First draft of the project report regarding the research stage		
Further notes by supervisor or student (if applicable):		

Logbook from week 6

Student's family name:	Student's other names:	E-mail/user-name:
Schäfer	Simon Philipp	su002988

Project title :
RFID domestic location detection

<p>Items for discussion (noted by student before supervisory meeting):</p> <ol style="list-style-type: none"> 1. Layout: 2cm as top and bottom border, does this mean to text or to header/footer? 2. Referencing: If I write a whole section from the knowledge of one or more books, how do I do the reference of it. 3. Logbook: Do I need to use the original form for the logbook or may I make my own one? 4. Report: Reference & bibliography? Difference and do I need the bibliography too, may I announce wikipedia in bibliography? 5. Style: How to paraphrase he/she in something like "The customer..., he" or something like "he/she" or "he or she"? 6. Appendices: Is there a fixed order for the appendices. Does the logbook need to be in appendix D? 7. Gantt: Is the Gantt chart OK. 8. XML: If I want to use XML to exchange information, or may I take the knowledge as granted?

<p>Record of discussion of supervisory meeting:</p> <ol style="list-style-type: none"> 1. Layout: To text not to header. 2. Referencing: At the beginning a short line explaining where the knowledge for this section comes from. 3. Logbook: I may use my own form. 4. Report: No bibliography. But I should reference to wikipedia as well, as long as I also use books to reference. 5. Style: No he or she, just repeat "the customer" 6. Appendices: No preset order, but it should make sense. 7. Gantt: Gantt is OK but as soon as I enter the other stages I have to put more detail on them. In critical evaluation I have to mention the gantt charts. 8. XML: Just say why and show the alternative, and why it would be better with XML. 9. Question from Carolin: Leftovers for this stage?: half of XP methodology to write, one more tech (ZigBee) and the XML mentioned above
--

<p>Action list (to be attempted or completed by student by the next supervisory meeting):</p> <ol style="list-style-type: none"> 1. Send 1. draft if it is finished.
--

<p>Further notes by supervisor or student (if applicable):</p>

Logbook from week 7

Student's family name:	Student's other names:	E-mail/user-name:
Schäfer	Simon Philipp	su002988
Project title :		
RFID domestic location detection		
Items for discussion (noted by student before supervisory meeting): 1. Referencing: How to reference a website if there are no details on it? 2. Should I describe how I installed and configured additional helping software like svn or trac?		
Record of discussion of supervisory meeting: 1. Try to find a date at the bottom of the page, and as title use the title of a the Webpage 2. This should be done at the implementation stage.		
Action list (to be attempted or completed by student by the next supervisory meeting): 1. creating new Gantt chart with a more detailed Analysis stage and updated dates		
Further notes by supervisor or student (if applicable): No comment to the project report by now, because the supervisor had no time to read it		

Logbook from week 8

Student's family name:	Student's other names:	E-mail/user-name:
Schäfer	Simon Philipp	su002988
Project title :		
RFID domestic location detection		
Items for discussion (noted by student before supervisory meeting): 1. What do we have to deliver till christmas, for the second accessor?		
Record of discussion of supervisory meeting: 1. Depends on the second accessor. As soon as I have got assigned to one I need to find out what I need. 2. Project report: In front of each section a small paragraph about why it is important for my project and what it is for. 3. Project report: To each methodology pros and cons in general and regarding to my project 4. Project report: perhaps an overview of the methodologies at the end of the Methodology section (table?) 5. Project report: DB/Webserver sections need more details for each 6. Project report: Conclusions need much more details, justification of tables etc.		
Action list (to be attempted or completed by student by the next supervisory meeting):		
Further notes by supervisor or student (if applicable):		

Logbook from week 9

Student's family name:	Student's other names:	E-mail/user-name:
Schäfer	Simon Philipp	su002988
Project title :		
RFID domestic location detection		
Items for discussion (noted by student before supervisory meeting):		
1. Second accessor? What does he/she want to see?		
Record of discussion of supervisory meeting:		
1. Have to look in the FYP system and send an email to second accessor.		
2. General progress, a little bit of polish of report + 1/3 of analysis		
Action list (to be attempted or completed by student by the next supervisory meeting):		
1. some more polishing at the methodology section		
2. some more work at the analysis section		
Further notes by supervisor or student (if applicable):		

Logbook from week 10

Student's family name:	Student's other names:	E-mail/user-name:
Schäfer	Simon Philipp	su002988
Project title :		
RFID domestic location detection		
Items for discussion (noted by student before supervisory meeting):		
1. Does the second accessor need any papers in advance?		
Record of discussion of supervisory meeting:		
1. Not necessarily.		
2. Talked about analysis (thought I had written all). Missed the target. What I have written so far can be used in the design stage but not in analysis		
Action list (to be attempted or completed by student by the next supervisory meeting):		
1. Rewrite the analysis section		
Further notes by supervisor or student (if applicable):		

Logbook from week 11

Student's family name:	Student's other names:	E-mail/user-name:
Schäfer	Simon Philipp	su002988
Project title :		
RFID domestic location detection		
Items for discussion (noted by student before supervisory meeting):		
Record of discussion of supervisory meeting: 1. How is my projet going on compared to the Gantt chart? Ok, but for now more work is put into assignments		
Action list (to be attempted or completed by student by the next supervisory meeting): 1. Contact second accessor to make up an meeting date		
Further notes by supervisor or student (if applicable):		

Logbook from week 12

Student's family name:	Student's other names:	E-mail/user-name:
Schäfer	Simon Philipp	su002988
Project title :		
RFID domestic location detection		
Items for discussion (noted by student before supervisory meeting): 1. What is the gradex entry and do I need it? 2. Not able to write conclusions for all the topics, but extended them so they got a reason. 3. Do I need a ethical statement somewhere?		
Record of discussion of supervisory meeting: 1. Not necessarily because I am an exchange student. 2. Ok 3. somewhere at the beginning after short proposal 4. Need more detail in Methodology conclusion 5. Analysis should be like the "Pflichtenheft" but much shorter 6. The technical stuff should be moved from Analysis to Research		
Action list (to be attempted or completed by student by the next supervisory meeting): 1. Get research and analysis done		
Further notes by supervisor or student (if applicable):		

Logbook from week 13

Student's family name:	Student's other names:	E-mail/user-name:
Schäfer	Simon Philipp	su002988
Project title :		
RFID domestic location detection		
Items for discussion (noted by student before supervisory meeting):		
Record of discussion of supervisory meeting: 1. Talking about how the EOS interview was and what was criticised. 2. Current status of the project.		
Action list (to be attempted or completed by student by the next supervisory meeting): 1. Try to catch up with the Time schedule 2. Fix the criticised parts of the report		
Further notes by supervisor or student (if applicable):		

Logbook from week 14

Student's family name:	Student's other names:	E-mail/user-name:
Schäfer	Simon Philipp	su002988
Project title :		
RFID domestic location detection		
Items for discussion (noted by student before supervisory meeting):		
1. How to reference pics? 2. How is analysis and design looking so far 3. 10pt or 12pt for the report 4. do I need Copyright or Trademark signs for the specific products?		
Record of discussion of supervisory meeting: 1. like normal text, direct and indirect 2. Analysis should have a diagram like desgin has, design ok 3. 12pt 4. No 5. Current state of the project regarding to the gantt, should have finished analysis but not yet done, instead some design done.		
Action list (to be attempted or completed by student by the next supervisory meeting): 1. finish writing analysis and write some researched topics down into the Research section		
Further notes by supervisor or student (if applicable):		
Need to skip the meeting in the 4th or 16th week		

Logbook from week 15

Student's family name:	Student's other names:	E-mail/user-name:
Schäfer	Simon Philipp	su002988
Project title :		
RFID domestic location detection		
Items for discussion (noted by student before supervisory meeting):		
Record of discussion of supervisory meeting: 1. Testplan has to be derived from analysis 2. Analysis needs a plan like the design 3. write down the restrictions and features of the system.		
Action list (to be attempted or completed by student by the next supervisory meeting): 1. do the fixes from above and finish design		
Further notes by supervisor or student (if applicable):		

Logbook from week 17

Student's family name:	Student's other names:	E-mail/user-name:
Schäfer	Simon Philipp	su002988
Project title :		
RFID domestic location detection		
Items for discussion (noted by student before supervisory meeting):		
1. State of the project		
Record of discussion of supervisory meeting: 1. Finished designing and started implementation 2. setup trac and subversion 3. OpenBeacon Configurator finished (implementation+documentation)		
Action list (to be attempted or completed by student by the next supervisory meeting): 1. correct the SSL protocol in design 2. setup the compiler environment for the firmware 3. do the adjustment of the firmware		
Further notes by supervisor or student (if applicable):		

Logbook from week 18

Student's family name:	Student's other names:	E-mail/user-name:
Schäfer	Simon Philipp	su002988
Project title :		
RFID domestic location detection		
Items for discussion (noted by student before supervisory meeting):		
Record of discussion of supervisory meeting: 1. State of the project: SSL protocol done, firmware done, started with the gain data daemon 2. Some talk about the time thats left.		
Action list (to be attempted or completed by student by the next supervisory meeting): 1. finish the gain data daemon 2. catch up to write the digitalized logbook entries		
Further notes by supervisor or student (if applicable):		

Logbook from week 19

Student's family name:	Student's other names:	E-mail/user-name:
Schäfer	Simon Philipp	su002988
Project title :		
RFID domestic location detection		
Items for discussion (noted by student before supervisory meeting):		
1. Logbook entries are digitalized		
Record of discussion of supervisory meeting: 1. State of the project		
Action list (to be attempted or completed by student by the next supervisory meeting):		
Further notes by supervisor or student (if applicable):		

Logbook from week 21

Student's family name:	Student's other names:	E-mail/user-name:
Schäfer	Simon Philipp	su002988
Project title :		
RFID domestic location detection		
Items for discussion (noted by student before supervisory meeting):		
Record of discussion of supervisory meeting: 1. Current state of the project, done, undone and problems.		
Action list (to be attempted or completed by student by the next supervisory meeting):		
Further notes by supervisor or student (if applicable):		

Logbook from week 22

Student's family name:	Student's other names:	E-mail/user-name:
Schäfer	Simon Philipp	su002988
Project title :		
RFID domestic location detection		
Items for discussion (noted by student before supervisory meeting):		
Record of discussion of supervisory meeting: 1. What was done during the easter holiday 2. Big problem with the hardware that occurred during testing		
Action list (to be attempted or completed by student by the next supervisory meeting): 1. Creating a simulation for the hardware to presentate the project and its implementations		
Further notes by supervisor or student (if applicable):		

Logbook from week 23

Student's family name:	Student's other names:	E-mail/user-name:
Schäfer	Simon Philipp	su002988
Project title :		
RFID domestic location detection		
Items for discussion (noted by student before supervisory meeting):		
1. When and where will the presentation be. 2. What should be presented and how.		
Record of discussion of supervisory meeting: 1. State of the project 2. Hardware simulator done		
Action list (to be attempted or completed by student by the next supervisory meeting): 1. Send project report to supervisor to have some advice if there is any.		
Further notes by supervisor or student (if applicable):		

Logbook from week 24

Student's family name:	Student's other names:	E-mail/user-name:
Schäfer	Simon Philipp	su002988
Project title :		
RFID domestic location detection		
Items for discussion (noted by student before supervisory meeting):		
<ol style="list-style-type: none"> 1. Presentation date set to 6th of May 2. Are there other ways to hand in the report (not only ring) 		
Record of discussion of supervisory meeting:		
<ol style="list-style-type: none"> 1. Presentation time to 2pm 2. There is a special facility which does the ring stuff 3. Too few references in the research part 4. Should mention used testing strategies 		
Action list (to be attempted or completed by student by the end of the project):		
<ol style="list-style-type: none"> 1. Ask Graham if 2pm is ok 2. Fixing some minor tasks in the report 		
Further notes by supervisor or student (if applicable):		
Student: Well the project is nearly over, and I have to say: "The time went by really fast"		