#109

BachelorPad

Report

#109

Summary of Bachelor Thesis

Title	BachelorPad
Date	2019-05-20
Author	Harald Geving
Supervisor	Helge Hafting
Employer	Geving IT
Contact Person	Harald Geving
Keywords	Home automation, reverse engineering, unifying solutions
Pages	51
Attachments	1
Availability	Open
Abstract	*"Solving a decade old problem and helping potentially hundreds of people, while at the same time giving back to the open source community and achieving full control over one's house."* That is a very exciting way to describe the application that is the result of this project.

The main goal of this project was to create an app that works like a bridge between the user's xComfort system and their home automation software. Until now, there has been no* good third party alternatives to the 13 year old software one could buy from the manufacturer.
And that software only handled the one system.

The application would speak to the xComfort hardware by using a reverse engineered protocol to talk to the Communication Interface (CI), and have the ability to connect to other systems as well.

By a stroke of luck, there was no need to reverse engineer the protocol, and one of the major third party Home Automation systems released an update that allowed using the message protocol MQTT to talk to the back end, removing the need to develop a custom one.

Also, it was confirmed that generic RF devices DO mostly use industry standards for their communication.

**There is ONE, but it's neither free nor open source.*

Foreword

Why?

This project came to be, because I've always wanted a way to control and monitor all of my devices and appliances from a single place. Or I should rather say; from any place! A few attempts have been made through the last eight or nine years, but have always come short. The biggest reason was a lack of documentation for the main components in my house; the xComfort system. I have tried many applications that control several device types with a single UI, but none of them had support for my xComfort, and therefore didn't work for me. The only way I would ever get the system I wanted was by making it myself!

How?

Judging from the presentation I gave before starting the project, I was clearly over ambitious! Planning to not just have the xComfort part as a module, but also writing the system that would use said module. I also planned to reverse engineer and make a module for 433MHz RF devices, as well as documenting and releasing everything as an open source framework.

I bought several different RF-modules from eBay, and managed to do a fair bit of RF research as well as xComfort coding, before discovering that a system similar to my back end had just had released a massive update that rendered much of my work obsolete. I decided to rewrite the xComfort module to comply with the standards used by the other system, and still end up with a valuable contribution to the open source community.

Who?

There are a few people who are very important to this project:
Mikael Finstad[1], whose early work on reverse engineering the protocol in 2012 inspired me to keep coming back to the topic every once in a while. GitHub user ksya[2] who posted a pdf with the entire xComfort protocol in another project's comment section! I would like to give thanks to my sister, Hanne Geving, for helping me stay focused and seeing this through!

But the most important people, by far, is my lovely wife Eva Gjeset Geving, and my daughters Kristine and Ida! Thank you for all of your support and your sacrifices over the last three years, and especially during these final months of this bachelor thesis!
Thank you!

Harald Geving
HELL, 2019-05-20

[1] "mifi (Mikael Finstad) · GitHub." https://github.com/mifi. Accessed 19 May. 2019.
[2] "ksya (ksya) · GitHub." https://github.com/ksya. Accessed 19 May. 2019.

Case

The original

The original description was very wide and left a lot up to the developer.

In Norwegian:

"Integrere eksisterende hjemmeautomasjons og IoT-systemer med proprietære løsninger inn i ett felles system, som deretter kan knyttes opp mot andre, mer åpne, systemer på markedet.

Av de proprietære systemene er det spesielt xComfort trådløs strømstyring fra Eaton, Arduino-basert custom hardware som kommuniserer med xBee eller wifi, samt generiske 433mhz-enheter som skal kunne leses av og styres ved hjelp at ett enkelt system. Dette systemet skal så kunne kommunisere med f.eks Google Home/Assistant og Amazon Echo/Alexa. Det må også være enkelt å kunne legge til nye enheter i etterkant, f.eks smart dørlås, strømmåler, elbillader eller andre smarte enheter som normalt sett ikke er åpne for integrasjon mot samlesystemer."

There never was an official English text. However, this is translated by the original author:
"Integrating existing home automation as IoT systems with proprietary solutions into a common system, which in turn can be connected to other, more open, systems available on the market today.

Of the proprietary systems, it is especially xComfort wireless power management by Eaton, Arduino-based controllers who communicate using xBee or WiFi, as well as generic 433MHz devices that are to be monitored and controlled by a single, simple system. This system will then be able to communicate with systems such as Google Home/Assistant and Amazon Echo/Alexa. It should also be easy to add new devices later on, such as smart door locks, energy meters, chargers for electric vehicles, or other smart devices that normally aren't possible to integrate."

What I set out to do was to create a master application that would use modules for all communications. The idea being that it would be very easy to expand and improve the system afterwards. I also had plans for at least two modules, three if there was time.

In addition to this, I envisioned developing and documenting a standard for the communication between master and module, ending in a complete open source framework. The planned modules were xComfort and generic 433Mhz RF. Both of which would require reverse engineering of protocol. My starting hypothesis was that many, perhaps even most, manufacturers use well-known industry standards when designing their products, thus making it possible to reverse engineer and replicate their transmissions.

The change

The main reason for not using one of the several other systems available was that they either were too expensive or that there was no feasible way to get them working with xComfort. One such system, OpenHAB2, had shown a lot of promise when I tried it out a few months before, but it didn't have support for xComfort and add-ons had to be written in Java.

In early April, while adding references for citations in the project documentation, I discovered that there had been a major release between submitting the proposal and starting on the project. OpenHAB 2.4 had gotten native support for MQTT. This meant that it was now possible to write an add-on in any language, using MQTT for communication.

Final version

With major changes to the premise for the project, the end goal of the project will naturally also change somewhat. The new goal was to create an application that would meet these criteria:

- Control most xComfort devices using either of the two xComfort Communication Interfaces
  - Specialized industrial components not to be prioritized
- Compatible with OpenHAB 2.4
  - Using MQTT for communication
- Cross-platform
  - Windows and Linux on x86 architectures
  - Linux on ARM architectures
- Open source
- Easy to expand and modify by others

For details, see appendix “Specifications v2.0”

Outline

This project has solved a decade old problem for potentially hundreds of people, including the author himself. Since arriving on the market in 2003, xComfort has quickly become the market leader[3] in home automation and wireless electrical installations on the home front. The products have been marketed as offering everything you need for total home control. While this is technically true, most of these full scale integrations are achieved by adding extra hardware instead of control by software. There exists show cases where total integration using xComfort has been done on a software level. However, these seem to be largely custom programming jobs by the manufacturer.

The idea of home automation is not new, nor is the idea of a single unifying system. There are several applications available today that offer exactly that. However, support for xComfort is lacking in almost all of them, and even though they can be expanded through modules, there weren't written any xComfort modules for any of them.

This project set out to reverse engineer enough of the xComfort protocol to be able to control a home without relying on Binary input-modules or commissioning custom software development. Due to requirements in programming language and/or other factors such as monthly licence fees, developing an addon for one of the existing systems was not an option. Keeping in line with the rest of the home automation scene, the final product would also be module-based. This allows for easy expansion of the system's capabilities, as third parties can develop just the code needed to add a specific feature and not having to worry about things like user management, visualisation, rules management, etc.

In addition to the xComfort module, a module for generic 433MHz devices such as wireless thermometers and remote controlled plugin adapters would be added. The hypothesis being that manufacturers mostly stick to standards instead of "reinventing the wheel" each time they develop a product. If this is the case, it should be feasible to reverse engineer these RF transmissions by analyzing the traffic.

Early on in the process, a major breakthrough was achieved! In the comment section of a GitHub Issue for a related project, someone posted a PDF with the complete specifications of the xComfort protocol! This removed the reverse engineering aspect of the xComfort module, but greatly improved the quality of the final product as it removed any guesswork as well as enabling a basic groundwork for xComfort modules unavailable for testing or research. Even modules not advertised in the Norwegian consumer market have a basic entry in the software.

After acquiring several different RF-modules and from their datasheets learning some technical terms; the hypothesis turned out to be correct! In fact, there are already several Arduino libraries dedicated to the same sort of functionality outlined by the proposed module. There was simply no real research to be done, merely confirming that the libraries and RF-modules did work as described.

The biggest and most influential event occurred in early April, when it was discovered that a major update to one of the most popular multi-manufacturer home automation

[3] "Xcomfort – elektroinstallasjonens revolusjon! - Sinus." 27 Nov. 2007, http://www.sinusmagasinet.no/artikler/2007/november/xcomfort/190. Accessed 20 May. 2019.

systems rendered most of the project obsolete, as the xComfort module could now be adapted to work with that system instead of the one currently being made. Considering the probable increase in overall quality for the end user by switching to a system more than 6 years in the making, backed by more than 20 developers, the decision was made to discard all of the non-xComfort work done so far, and use the remaining time on adding the necessary MQTT protocol and implementing the Homie-convention.

This decision resulted in even further narrowing of the project scope, but increased the value of the end product considerably, as it can now be used as a drop-in solution for anyone already running one of several major home automation systems, as the Homie-convention is supported by many different systems.

Table of contents

1. Introduction

This report consists of six chapters. The first chapter explains why this project has come to be, a brief introduction to the home automation scene, and details how it is intended that this project is going to be solved. Chapter 2 contains relevant theory about the technology aspects touched upon within this report. Chapter 3 focuses on specific technologies and the reasons why these were chosen. Chapter 4 contains the results of the project, detailing how the final product came out. Chapter 5 is the discussion chapter in which certain details are discussed. Chapter 6 is the final chapter, in which the project conclusion and thoughts on the products future is written.

1.1 The general issue

Given the rapid development and ever growing field of home automation, IoT and “smart devices”, it’s becoming increasingly harder to stay loyal to a single brand or ecosystem. This means that one is almost guaranteed to have at least two or more systems with their own app and/or other way of communicating with the user.

This situation can be likened to the living room tables of the 90’s, when there was a dedicated remote control for the TV, VCR, stereo system, satellite or cable decoder, and perhaps even a remote for the air condition. The solution to this problem was a universal remote that could control all of your devices with only one remote.

This project addresses the 2019 equivalent of the many remote controls problem. The goal is to provide a single interface for controlling appliances and systems from multiple brands and technologies. There should be no reason to have to open two or three apps just to turn on the lights.

There are several systems, already available, that aim to remedy this problem of managing several systems in a single home. Many of them do a great job, and have come to support a large amount of devices and appliances. However, almost none of them have native support for xComfort, which is absolutely the most used system in Norway when it comes to wireless home automation with regards to lights and heating. I have only been able to find a single system[4] that has support for xComfort, and that one has a price range of €100 to more than €250.

1.1.1 A brief introduction to xComfort

This project is centered around a specific range of wireless products called xComfort. The main components are wireless push buttons and dimming or switching actuators, but the series includes thermostats, movement detectors and more. Since Moeller released the xComfort line in 2003, they quickly became the number one product in the home automation

[4] "IP-Symcon :: Automation Software." Accessed May 17, 2019. https://www.symcon.de/en/.

segment in Norway. Moeller and Eaton merged in 2011[5], and the xComfort line is now a part of the Eaton family.

xComfort has always been marketed as an innovative and “smart” product, and the system has a long history of offering remote access. This remote access started out as a dialup and GSM modems in 2003 and 2005, before they introduced a USB/RS232 Communication Interface (CI) in 2006. This was a huge selling point, as one could now control the entire home from an ordinary Windows computer. It was the only option for computerized control until they released an ethernet version of the CI in 2013.

In order to use the CI, one also had to have a license for the “Homeputer” software. This software was available in “Standard” and “Studio” edition, where the “Standard” was somewhat limited in its functionality compared to the “Studio” edition, but neither of them could control anything other than xComfort devices. There was also a separate licence for a proprietary web server available, should one want to turn on the lights without using the computer physically connected to the USB interface. These licences were €99 for “Standard”, €149 for “Studio” and an additional €59 for the web server. The price of the USB interface was about €120. All prices eksl. VAT.

1.2 The specific problem

In 2008, I started building the house that my wife and I live in today. Being the technology-interested person I am, a “smart house” was absolutely a goal. After speaking to several electricians, it became clear that xComfort was the only real alternative on the market. So, the house was built with xComfort units in all places. This also includes the thermostats and valves for the in-floor water heating.
For total control of the setup, I purchased a USB interface and a licence for “Homeputer Studio” as well as a licence for their proprietary web server. Unfortunately, this software never provided the level of ease and control that I had envisioned. When xComfort announced their new ethernet based control system in 2013 I was intrigued, but decided not to purchase a new €1000+ hardware unit given my disappointment over the €400+ solution I already had paid for. The idea of writing my own software to handle this problem has been in the back of my mind for almost a decade, but it has never been given priority …
… until now!
This project will be my chance to finally get the control that I desire, as well as making a useful contribution to the open source community and helping out my fellow xComfort owners who also desire a better alternative!

1.3 Limiting the scope

At the time of submission, this project was intended to be a complete system for which others could add functionality by using its API. However, by the middle of December 2018,

[5] "Moeller - Eaton." https://www.eaton.com/Eaton/ProductsServices/ProductsbyName/Moeller/index.htm. Accessed 20 May. 2019.

OpenHAB released version 2.4 of their software[6] which eliminated all of the reasons for not using that software as the backbone of the home. This changed the scope of the project. Instead of making a master system and a support module, it made more sense to focus on making the support module compatible with OpenHAB and perhaps also other systems as well. This will make it much easier for others to reuse the code in their own projects. Many should also be able to simply use the binaries with their own config.

1.4 Hypothesis

The universal remote control was possible because all* of the appliances used the same technology for transferring commands; pulsed IR light.
This is obviously not the case with today's appliances, as they use radio waves instead of pulses of light. However, one might hypothesise that most manufacturers still choose to rely on industry standards instead of doing their completely own thing.

If this is the case, one might be able to reverse engineer the transmissions by first capturing them and then analyzing and comparing them to known or assumed known data such as the command "ON" or the temperature shown on the device's LCD.

**Some readers might be quick to comment that such "universal remotes" weren't necessarily entirely universal after all. If one had a Bang Olufsen TV or stereo, one might experience that many universal remotes weren't able to operate the appliance. This was because BO used a different coding frequency than the rest of the industry for most of their appliances. They still used IR light, but they sent the data in a different manner.*

1.1 Acronyms

Acronym	**Meaning**	**Explanation**
IoT	Internet of Things	Small, usually single task products that are connected to the Internet. Examples: Thermometers, refrigerators, lights.
VCR	Video Cassette Recorder	Machine used for recording and playback of analogue video in the 80's and 90's.
56K	56000 baud	Bandwidth speed of 56kbps (0.056mbps).
GSM	Global System for Mobile Communications	Mobile phone network that preceded 3G.
API	Application Programming	A recipe on how an application or system can be talked to. This includes message formats, addresses, reply

[6] "openHAB 2.4 Release | openHAB." Accessed May 17, 2019. https://www.openhab.org/blog/2018-12-17-openhab-2-4-release.html.

	Interface	structures and more.
IR	InfraRed	Light with a wavelength outside of the visible spectrum.

2. Theory

2.1 Home automation

2.1.1 Definition

In this context, Home automation is remotely or programmatically operating an appliance or device that usually is operated manually.

2.1.2 Brief history

Home automation has been with us in one form or another for many years. One of the earliest and simplest forms of home automation is the timer switch. They have been around since the 1930's[7]. More recent examples of home automation includes "The Clapper"[8], photoelectric switches[9] for controlling lights.

In the late 70's[10] and early 80's, one could see the first products that resemble those solutions we have today: the X-10 system. X-10 modules would allow a user to remotely control appliances such as lamps, sockets, sprinklers and more. X10 modules are still available today.[11]

2.1.3 Extended definition

A very common scenario given when an enthusiast is asked to give an example of home automation is the home theatre:
Imagine sitting yourself down on the sofa and pressing one single button that in turn:

- Dims the lights
- Rolls down the blinds
- Sets the A/C to "silent mode"
- Activates the video projector
- Turns on the surround sound system
- Brings up the movie selection menu on the screen

This example is a good one, because most people can agree that all of these things are nice to have, and usually require operating several switches and buttons on multiple controllers

[7] "US2056400A - Time switch - Google Patents." Accessed May 17, 2019. https://patents.google.com/patent/US2056400A/en.
[8] "US2056400A - Time switch - Google Patents." Accessed May 17, 2019. https://patents.google.com/patent/US2056400A/en.
[9] "US2781477A - Photoelectric control apparatus - Google Patents." Accessed May 17, 2019. https://patents.google.com/patent/US2781477.
[10] "Dave Rye of X10 | HomeToys." Accessed May 17, 2019. https://www.hometoys.com/dave-rye-of-x10/.
[11] "X10.com." Accessed May 17, 2019. https://www.x10.com/.

and devices. This is one of the reasons why Audio/Video integration often is included when speaking of home automation.

2.1.4 Compatibility

The universal remote control from the 90's was possible to create because all of the appliances used the same technology for transferring commands; pulsed IR light[12].

Today, most modern home automation/IoT devices use radio waves for transmitting and receiving data. The data is either processed by the receiver directly, or they are sent to a local hub and/or a server/cloud service. Depending on the transmission type, a gateway might be needed to forward the data to a different medium. These transmission types can be roughly divided into three major categories: "WiFi", "Standardized" and "Other".

2.1.4.1 WiFi

Devices that use WiFi for communication obviously can't be controlled by simply repeating a radio signal, nor is it practically possible to read data by listening to the radio signal it transmits. Thus, to control such devices one must analyze the data on a higher level of the OSI model.

2.1.4.2 Standardized

Many manufacturers have chosen to use a well known industry standard like Z-wave or ZigBee for their devices. These devices all use the same protocols, and should therefore be able to interact with each other regardless of manufacturer. One example of this is how Philips Hue light bulbs can be used in conjunction with the Ikea Trådfri system[13].

2.1.4.2 Other

If a device doesn't fall into the aforementioned categories, it's very likely that they use their own data format[14], which they transmit using either OOK or FSK modulation[15] within the ISM-bands[16]. With devices such as switches, thermometers and other sensors, the data sent per transmission is usually rather small. This means that one could utilize such devices in a home automation situation by using a RF transceiver in conjunction with some clever software to help the user identify and replicate the RF device.[17] Unfortunately, such devices

[12] "US5959751A - Universal remote control device - Google Patents." Accessed May 17, 2019. https://patents.google.com/patent/US5959751A/en.
[13] "Compatibility & protocols - IKEA." Accessed May 17, 2019. https://www.ikea.com/gb/en/customer-service/smart-lighting-support/faq-smart-lighting/compatibility-protocols/#1364400184185.
[14] "Oregon Scientific RF Protocol Description." Accessed May 17, 2019. http://www.osengr.org/WxShield/Downloads/OregonScientific-RF-Protocols-II.pdf.
[15] "a comparison between ook/ask and fsk modulation ... - Digikey." http://www.digikey.com.au/Web%20Export/Supplier%20Content/RFM_583/PDF/rfm-an-ook-ask-fsk-comparison.pdf. Accessed 20 May. 2019.
[16] "Frequently asked questions - ITU." Accessed May 17, 2019. https://www.itu.int/net/ITU-R/terrestrial/faq/index.html#g013.
[17] "Decoding NEXA remotes with TellStick Duo | Unix, trix and tools." Accessed May 17, 2019. https://lassesunix.wordpress.com/2013/12/28/decoding-nexa-remotes-with-tellstick-duo/.

are often equipped with transmitters OR receivers, meaning that two way communication is not possible. This makes sense as the use cases mostly lets the user get feedback by observing the result of the transmission. Either by seeing that a light turn on, or that the temperature is being displayed on a screen. However, for a computer it is impossible to know the status of a device. One can only assume that it received and performed the last command sent or observed. This might be OK for some scenarios, but state verification is absolutely desirable in many cases.

2.2 Cross-platform

Writing software without knowing what hardware and/or OS it is going to run on presents a few challenges. It limits your options as a developer, and requires you to think a little extra about the decisions you make during the development process.

2.2.1 Hardware

There will obviously be differences in code written for a 64-bit multi-core CPU with 32GB of RAM and practically infinite storage space as well as dedicated peripherals for processing audio, video, networking and crypto, VS a 8-bit microcontroller with 8K bytes of RAM and 256K bytes of storage. For one thing, the microcontroller would use half of the storage space just to hold the root CA certification package for verifying SSL connections. That's before any code has been written.

Obviously, this example was a very extreme, but it highlights the point; the available hardware is not necessarily known ahead of time, and one needs to code with that in mind. Minimum hardware requirements is one way to ensure that the software will run nicely. In this project, minimum one USB or RS232 port is required, as well as a network connection. However, when it comes to hardware exceeding the minimum requirements, the developer is faced with a choice; to utilize or not to utilize?

An example of this would be how it is possible to increase throughput as well as reduce CPU load by using hardware accelerated crypto[18] instead of doing it all in software. Naturally, this requires more work on the developers part, as there must be functioning code for both cases.

2.2.2 Operating System

Even with the exact same hardware, there might still be huge differences for the developer, as the users might be running different operating systems. Most OS's offer the same functionality to the developer, and sometimes it's enough to just compile the code for the correct OS. This gives one binary for one OS and another binary for another OS, even though no changes were made to the source code. But, often this is not the case. Even though both OS's offer the same functions, the way to get there might be very different.

[18] "Hardware — Cryptographic Accelerator Support | pfSense ...." Accessed May 17, 2019. https://docs.netgate.com/pfsense/en/latest/hardware/cryptographic-accelerator-support.html.

Also, some programming languages can't be compiled for other OS's. This means that software written in such a language would require a complete rewrite in a different language in order to run on a different OS.

2.2.3 Just-in-time-compilers

There are certain languages that allow the developer to write code without much consideration to what hardware or OS the end user has. Such languages mostly rely on a piece of software that sits in between the OS and the application, handling all of the OS specific things in the background. This allows the same application to run on very different devices without needing to do any modifications to the code.

This method comes with some performance loss[19] due to the extra software layer, but for applications that aren't extremely CPU and/or memory intensive: this is not a huge issue.

2.3 Security

Security is important! Especially when it comes to something as private as one's own home. There are sadly many examples of how IoT manufacturers have ignored this aspect of the product.[20] Security may often be hard to implement on home automation devices though:

- Old transmission protocols don't support it
- Limitations in data size
- CPU and memory limitations
- Power consumption
- Price

Fortunately, most of these restrictions only apply to devices that have a limited transmitter range. This means that an attacker needs to be physically close to the victim.
For those devices that rely on the Internet to carry the data, however, the attack could come from any place at any time.

Both REST over HTTP and MQTT are popular protocols for IoT devices. They are easy to understand, easy to implement and they are relatively light weight. Unfortunately, they both support non-encrypted traffic, allowing an attacker to read, and possibly replicate and manipulate data to and from these devices if they are not properly configured[21].

[19] "C++, C# and Java performance review - ToMMTi-Systems :: Hinter den ...." Accessed May 17, 2019. https://www.tommti-systems.de/main-Dateien/reviews/languages/benchmarks.html.
[20] "Breaking Down Mirai: An IoT DDoS Botnet Analysis - Imperva." Accessed May 17, 2019. https://www.imperva.com/blog/malware-analysis-mirai-ddos-botnet/.
[21] "The Fragility of Industrial IoT's Data Backbone - Trend Micro." Accessed May 17, 2019. https://documents.trendmicro.com/assets/white_papers/wp-the-fragility-of-industrial-IoTs-data-backbone.pdf#page=40.

3. Selecting technology and method

3.1 Licensing

It has always been my intention to release this software as open source. There are several reasons for doing this:

3.1.1 Giving back

It's nice to be able to make a contribution that I think will matter to many people. I know that there aren't very many solutions available for xComfort users, and that there are many xComfort users that have mostly given up on ever getting the level of control that they had imagined. Hopefully, this will help change that situation.

3.1.2 Portfolio

For someone who is applying for a job as a software developer, their GitHub is often just as important as their CV. This is absolutely a project I would want to showcase for a potential employer!

3.1.3 Completeness and improvements

There are certain parts of the program that are incomplete and/or untested, due to the lack of hardware on my part. Hopefully, others who have access to those specific components will contribute and complete the code. Also, it would be very arrogant of me to presume that my code could not be improved by someone with more experience and better skills. If someone wants to improve on my work, we both win!

3.2 Choosing a language

I usually program in VB.NET, but since this software is supposed to be cross-platform compatible and also handle rather low level hardware communication, the two best candidates were C# and Java.

I decided to go with C# as there are indications that the performance is better[22] and I have a personal preference for C#.

The cross-platform element has also been important when choosing which libraries to use; that they are Core compatible and will work on all of the major platforms.

Another reason for choosing C# or Java could have been the open source aspect, where far less people would have benefitted from the project had it been written in VB.

[22] "C# .NET Core vs Java - Which programs are faster ... - Debian." Accessed May 18, 2019. https://benchmarksgame-team.pages.debian.net/benchmarksgame/faster/csharp.html.

3.3 Back end choices

As mentioned in chapter 1, the scope was changed dramatically half way through the project. This meant that some difficult choices had to be made:

3.3.1 Continue or switch?

Upon discovering that the new release of OpenHAB had gotten native support for MQTT bindings, I had to reconsider whether or not I should stick to the original plan of making my own back end, or if I should switch to this new version of OpenHAB. The latter would mean discarding many hours of work and research, as it would be rendered useless by OpenHAB.

The decision to switch from my own back end system to OpenHAB 2.4 was finally made because OpenHAB has so much more to offer than what could possibly be created in the time available for this project, and that I would personally prefer to use OpenHAB to my own back end.. At the time, I also believed that it would free up more time to perfect the xComfort module.

3.3.2 MQTT, but how?

When the decision to switch to OpenHAB was made, another choice awaited: Using MQTT was a given, but what structure to use?
The MQTT message structure allows for a lot of freedom[23], which is nice in many situations. OpenHAB supports several different conventions[24], including “Homie”, “HomeAssistant” and a generic one. The latter is much easier to implement than the former, but the former allows for auto discovery by OpenHAB. This is a great feature for the end user, saving them a lot of time on configuring each device individually.

After looking at the specifications for both “Homie” and “HomeAssistant”, I chose “Homie” because it was very thorough and completely manufacturer-independent. It was, however, also the most complex convention to implement.

3.3.3 What to keep?

At this point in the project, a lot of existing code had to be re-written or removed, because it didn’t serve any purpose for the Homie format. One example of this is how the Homie format uses fewer data types than xComfort does, rendering code for preserving these data types obsolete and thus ready for removal.

However, because this meant to be an open source project, I decided that it would be better to leave most of the redundant and unused code in the project, as it makes a good basis for developing a secondary convention to choose from. If someone wants to use my code in a project that doesn’t use the Homie convention, it would be helpful to have the old code ready instead of having to rewrite it from scratch.

[23] "MQTT - Bindings | openHAB." https://www.openhab.org/addons/bindings/mqtt/. Accessed 20 May. 2019.
[24] "MQTT Things and Channels - Bindings | openHAB." Accessed May 18, 2019. https://www.openhab.org/addons/bindings/mqtt.generic/.

3.4 Tools

For the development of this project, I decided to use Visual Studio Enterprise 2017 as this is the most common tool for developing C# applications. Testing was done on a Raspberry Pi 3 because it is absolutely one of the most popular single board computers on the market today. And the fact that it runs Linux on an ARM processor, makes it a very good candidate for making sure that my application actually is cross-platform compatible.

3.5 Method

Following a strict development method, as described in the literature, seems somewhat artificial when one has both the role of customer and the role of project manager, as well as single handedly making up the entire development team.
The project consisted largely of three parts:

- Research
- Development
- Documentation

This is also the order in which the parts were mainly carried out. There were, of course, some overlap. The Vision document was prioritized before any real research was started. Boilerplate code and rough outlines were also programmed early on, due to logistics and shipping of the RF-modules needed for some of the research.

This doesn’t mean that the development process has been completely ad-hoc and random, but rather more in the lines of a “super agile” method with an “Instant iterative process”, where very little time is wasted on debates between developer and customer.

4. Results

4.1 Scientific results

4.1.1 Hypothesis

Most generic wireless devices today do operate in the EMI frequencies, using either OOK or FSK modulation. The data packets do not necessarily adhere to any standard format, but the encoding usually does. For devices that have a limited set of operations or range of data, such as thermometers and remote controlled sockets, the concise changes in small packets helps making the reverse engineering easier.

4.1.2 Discoveries

There are already a few well-defined conventions for MQTT message layout in the home automation/IoT community.
The Open Assistant standard, developed for the Open Assistant software is now supported and to an extent encouraged by Open Assistants biggest competitor; OpenHAB.

4.2 Engineering results

4.2.1 Goal achievement

These are the user stories from the document “Specifications v2.0”:

*… I want to configure parameters for my Communication Interface (CI)
so that it will work with my hardware.*
This criteria has been met by giving access to all settings through both the menu system as well as storing them in a local file called “settings.json”.

*… I want to import the list of data points that I already have
so that I don’t need to spend much time setting up the system.*
This is done by placing the datapoints.txt file in the application folder, or specifying a new location in the settings.

*… I want to configure parameters for my MQTT broker
so that it will work with my custom setup.*
This is handled by using the menu or by editing “settings.json”

*… I want to add xComfort support to my existing home automation system
so that I can include xComfort devices in my rules and automation.*
This is possible as long as the existing system has support for devices using the Homie convention for MQTT messages.

… I want to use any of my devices as a host for this application
so that I can choose hardware and OS freely, without thinking of compatibility.
By writing the application in .NET Core 2.2, it is fully cross-platform capable. It has been tested on 64-bit Windows 10 and 32-bit Linux on a single board computer with an ARM processor.

… I want to have access to the source code
so that I can verify, modify and improve for my own needs.
The source code is already available on GitHub[25]. The specific open source licence model has yet to be determined, but is expected to be in place by the end of May.

4.3 Administrative results

4.3.1 Hours and target

This is a table showing the different estimates and the actual timesheet for hours spent.

Source	**Hours**	**Deviation from target**		**Within tolerances**
Guidelines	501.5	--	--	--
GANTT v1.0	472	29.5	5.88%	Yes
GANTT v1.0	492	9.5	2.01%	Yes
Timesheet	481	20.5	4.17%	Yes

[25] "BachelorPad · GitHub." https://github.com/Geving/BachelorPad. Accessed 20 May. 2019.

4.3.2 Distribution

When grouping the entries by category, it becomes clear that there has been an increase in the number of hours spent on the documentation, while a somewhat similar decrease in the number of hours spent on research.

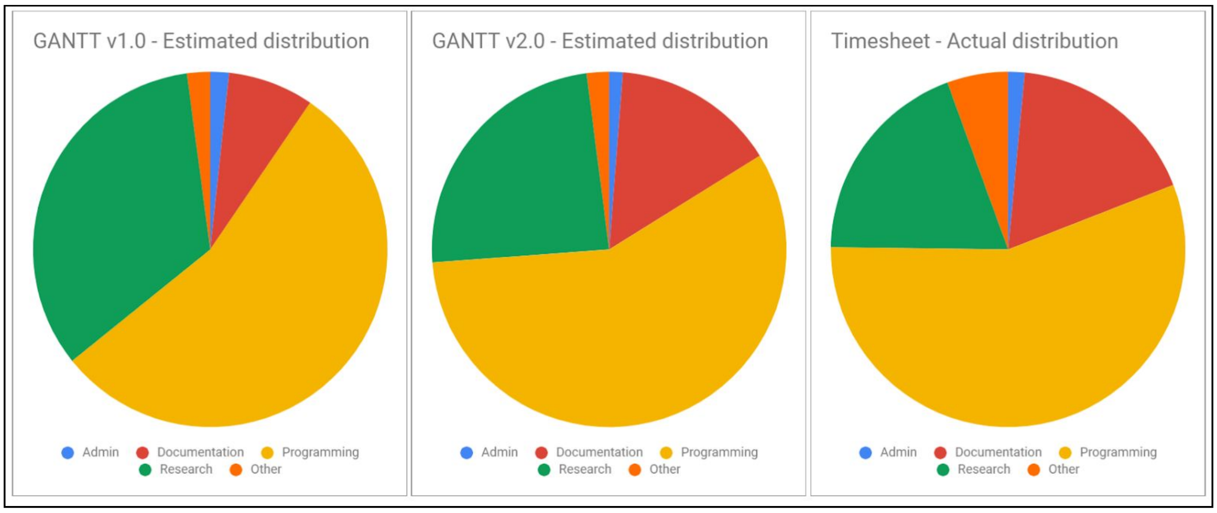


Fig 1) Show the differences in estimated and actual distribution of hours by category

5. Discussion

5.1 Scientific

5.1.1 Hypothesis

There are already several libraries[26] available for the Arduino platform, as well as decoding[27] and DIY[28] instructions for generic 433MHz devices. Unfortunately, this was not something I was able to discover before I had started doing research, as it wasn't until I added certain phrases from the RF-module's datasheets to my search queries that I found the overwhelming amounts of prior research into the topic.
This discovery was both good and bad at the same time:

- It was a good discovery, because it meant that it would absolutely be possible to achieve the customer's goal: controlling generic 433MHz devices.
- It was a bad discovery, because it meant that I no longer had a reasonable hypothesis to research for my thesis, as well as it would make no real sense to write from scratch that which already is available several places on GitHub. It turned the entire module development into an academic exercise where one has access to the answer but pretends not to.

Discovering this after having spent a lot of time and money on the different RF-modules and how to use them, ended up being more of a setback than an advancement.

5.2 Engineering

5.2.1 The delivered product

Did the customer get what was ordered?

Considering the literal description given at the start of the project; absolutely not! The delivery described in the first set of documents is way more elaborate than the final product.

However, considering that the earliest documents were written with the premise that it was impossible to utilize existing solutions, one might see that the delivered product in conjunction with a system like OpenHAB is in fact a better solution for the end user than what could have been achieved in the allotted time.

Also, the customer was definitely a part of the decision to redefine the scope half way through the project. And that new scope is much closer to the delivered product.

There are, though, some shortcomings with the final product:

[26] "sui77/rc-switch - GitHub." https://github.com/sui77/rc-switch. Accessed 20 May. 2019.
[27] "Poor man's 433MHz RF Sniffer - Codrey Electronics." 7 Jun. 2018, https://www.codrey.com/electronic-circuits/433mhz-rf-sniffer/. Accessed 20 May. 2019.
[28] "OOK | Hackaday." https://hackaday.com/tag/ook/. Accessed 20 May. 2019.

Not all device types are completely supported.

Even though the entire protocol specifications were available, there are still some device types that weren't prioritized. Devices such as Rosetta Routers, Dimplex thermostats, and BOSCOS (Bed Occupancy Sensor / Chair Occupancy Sensor) modules are only handled enough that they won't cause a problem should one ever be added into the system.

Ideally, every device type should be fully supported. And this might happen someday, but it was not a logical thing to prioritize for this release. Also, without the physical hardware to test and confirm that it works, it's even more reason to deprioritize it.

RS232 unconfirmed

For some unknown reason, it was not possible to verify that the RS232 implementation is working as it should. The CI module did not comply with the specifications in the protocol, as it skipped the last byte of the transfer, and it changed the starting byte as well.

However, it is believed to be working, as when the definition of the starting byte was changed to match the value given by the CI, it worked. This did not solve the problem with the missing byte at the end of each transmission though. This can only be verified by testing with a second CI unit.

Method and result

It is impossible to claim that this project has adhered to a strict development scheme. This is one of the inherent dangers of having both the role of the customer and that of the developer in the same project. The proposed time table was jostled several times by unexpected discoveries, both in research and by pure accident. However, the estimates were rather descent up to those events.

Also, after restructuring, the process has been about as expected, though the time estimates for both rewriting the documentation and retrofitting the code was a bit too optimistic. This resulted in rather long writing sessions at the end.

There is no doubt that this project would have been more structured if there had been a decision in place to simulate the customer/developer/manager interactions. However, this would most likely result in poor morale and disdain for the project. This was the reason why such a path was not taken.

5.3 Administrative

5.3.1 The bigger picture

Choosing to discard a large part of the original case and rewriting the xComfort module to function with other systems was a big decision. On the one hand, it would result in a lot of wasted time, effort and money. It would also make result in quite a lot of extra work on the documentation side, as all references to the RF research and original specs would have to be removed and/or rewritten.

But on the other hand, it would clearly make the end user experience much better by allowing them to choose between several systems that all are more mature and with greater functionality and user base than what could have been achieved by sticking to the original case.

Hopefully, this improvement will allow more users to take advantage of the end result, and also inspire others to improve and expand upon the code provided.

5.3.2 Ethics

There is one document referenced again and again in this report, but is not included nor linked to anywhere in the documentation, and that is the CI protocol specifications. This is done because it is unclear whether or not it would be legal to distribute the file. The original comment where it was found has since been edited so that it no longer contains the file.

As a compromise between the duty to include all references and staying within the law, the first four pages containing the front page, the change log and the table of contents are added to this report.

6. Conclusion and future work

6.1 Conclusion

With such an overwhelming amount of prior art examples, one must conclude that the main hypothesis indeed turned out to be true.

As for the final product; all of the requirements given in the vision document have been either met or exceeded. The project was completed within the time frame given, and within the expected amount of hours worked.

6.2 Future work

Since this application will be used daily by its author, further development and maintenance is inevitable. There are still xComfort modules which implementation can be improved, and it might be a good idea to add more MQTT conventions as well. That will further increase the amount of users who can benefit from this work.

6.3 Advice

Based on the experiences made during this project, a piece of advice to the next person who should wish to undertake such a task:

“Do your research first, and when you are about to start; do it again! Because things many things may happen in the period between submitting a proposal and starting on the actual job.”

References

A list of all footnotes in the document:

1. "mifi (Mikael Finstad) · GitHub." https://github.com/mifi. Accessed 19 May. 2019.
2. "ksya (ksya) · GitHub." https://github.com/ksya. Accessed 19 May. 2019.
3. "Xcomfort – elektroinstallasjonens revolusjon! - Sinus." 27 Nov. 2007, http://www.sinusmagasinet.no/artikler/2007/november/xcomfort/190. Accessed 20 May. 2019.
4. "IP-Symcon :: Automation Software." Accessed May 17, 2019. https://www.symcon.de/en/.
5. "Moeller - Eaton." https://www.eaton.com/Eaton/ProductsServices/ProductsbyName/Moeller/index.htm. Accessed 20 May. 2019.
6. "openHAB 2.4 Release | openHAB." Accessed May 17, 2019. https://www.openhab.org/blog/2018-12-17-openhab-2-4-release.html.
7. "US2056400A - Time switch - Google Patents." Accessed May 17, 2019. https://patents.google.com/patent/US2056400A/en.
8. "US2056400A - Time switch - Google Patents." Accessed May 17, 2019. https://patents.google.com/patent/US2056400A/en.
9. "US2781477A - Photoelectric control apparatus - Google Patents." Accessed May 17, 2019. https://patents.google.com/patent/US2781477.
10. "Dave Rye of X10 | HomeToys." Accessed May 17, 2019. https://www.hometoys.com/dave-rye-of-x10/.
11. "X10.com." Accessed May 17, 2019. https://www.x10.com/.
12. "US5959751A - Universal remote control device - Google Patents." Accessed May 17, 2019. https://patents.google.com/patent/US5959751A/en.
13. "Compatibility & protocols - IKEA." Accessed May 17, 2019. https://www.ikea.com/gb/en/customer-service/smart-lighting-support/faq-smart-lighting/compatibility-protocols/#1364400184185.
14. "Oregon Scientific RF Protocol Description." Accessed May 17, 2019. http://www.osengr.org/WxShield/Downloads/OregonScientific-RF-Protocols-II.pdf.
15. "a comparison between ook/ask and fsk modulation ... - Digikey." http://www.digikey.com.au/Web%20Export/Supplier%20Content/RFM_583/PDF/rfm-an-ook-ask-fsk-comparison.pdf. Accessed 20 May. 2019.
16. "Frequently asked questions - ITU." Accessed May 17, 2019. https://www.itu.int/net/ITU-R/terrestrial/faq/index.html#g013.
17. "Decoding NEXA remotes with TellStick Duo | Unix, trix and tools." Accessed May 17, 2019. https://lassesunix.wordpress.com/2013/12/28/decoding-nexa-remotes-with-tellstick-duo/.
18. "Hardware — Cryptographic Accelerator Support | pfSense ...." Accessed May 17, 2019. https://docs.netgate.com/pfsense/en/latest/hardware/cryptographic-accelerator-support.html.
19. "C++, C# and Java performance review - ToMMTi-Systems :: Hinter den ...." Accessed May 17, 2019. https://www.tommti-systems.de/main-Dateien/reviews/languages/benchmarks.html.
20. "Breaking Down Mirai: An IoT DDoS Botnet Analysis - Imperva." Accessed May 17, 2019. https://www.imperva.com/blog/malware-analysis-mirai-ddos-botnet/.
21. "The Fragility of Industrial IoT's Data Backbone - Trend Micro." Accessed May 17, 2019. https://documents.trendmicro.com/assets/white_papers/wp-the-fragility-of-industrial-IoTs-data-backbone.pdf#page=40.
22. "C# .NET Core vs Java - Which programs are faster ... - Debian." Accessed May 18, 2019. https://benchmarksgame-team.pages.debian.net/benchmarksgame/faster/csharp.html.
23. "MQTT - Bindings | openHAB." https://www.openhab.org/addons/bindings/mqtt/. Accessed 20 May. 2019.
24. "MQTT Things and Channels - Bindings | openHAB." Accessed May 18, 2019. https://www.openhab.org/addons/bindings/mqtt.generic/.
25. "BachelorPad · GitHub." https://github.com/Geving/BachelorPad. Accessed 20 May. 2019.

26. "sui77/rc-switch - GitHub." https://github.com/sui77/rc-switch. Accessed 20 May. 2019.
27. "Poor man's 433MHz RF Sniffer - Codrey Electronics." 7 Jun. 2018, https://www.codrey.com/electronic-circuits/433mhz-rf-sniffer/. Accessed 20 May. 2019.
28. "OOK | Hackaday." https://hackaday.com/tag/ook/. Accessed 20 May. 2019.

Appendices

I) Vision document v1.0

Describes the original vision for the project.

II) Specifications v2.0

Describes the updated specifications for the application

Separate file:

Project handbook v1.0

Contains GANTT diagrams and timesheets documenting the planned and the actual progress through the different phases of the project process.

#109

Appendix I

Vision

v1.0

Revision history

Date	Version	Description	Author
2019-03-01	1.0	Initial version	Harald Geving

#109

Table of contents

#109

1. Introduction

This document describes the high-level needs and desires of the stakeholders and how BachelorPad will address them. Details on how each need is fulfilled is described in the use-cases and the supplementary “Specifications” document.

1.1. References

“Specifications” document

2. Summaries - Problem and product

2.1. Problem summary

The problem with	each manufacturer having their own way of controlling their smart appliances
affects	residents who are unable or unwilling to use only products from a single manufacturer
which results in	limited options for choosing appliances or having to use several control methods, usually mobile apps, to control the home as a whole.
A successful solution	allows users more freedom in choosing appliances for the home, as well as reducing the number of apps required to operate the home.

2.2. Product summary

For	residents and users of “smart homes”
who	desire freedom to choose appliances from many manufacturers and/or want to control the smart home without having to switch between many apps
Bachelor Pad	is a system that communicates with, and controls, appliances from several manufacturers using only one interface.
Unlike	most manufacturers applications
Bachelor Pad	● can communicate with systems from several different manufacturers and technologies ● is expandable by third-party modules ● doesn’t require the installation of an app* ● can be utilized by guests* ** Depending on the configuration set by administrator*

3. Executive description of stakeholders and users

3.1. Summary of stakeholders

The only real stakeholder in this project is the developer. However, one might still identify some

Stakeholder	Represented by	Role in project
Customer	Geving IT	Placeholder for project customer
Developer	Harald Geving	Develops the product ordered by the customer
Users	Harald Geving and his family	End users of the final product, deployed in the family home
Other users and/or developers	Not represented	Users of the product after the project has ended, and the system has been released as open source.

3.2. Summary of users

Title	User type description	Represented by	Group(s)
“The geek” / ”The engineer”	Is the one implementing and maintaining all smart appliances in the home. Very forgiving if/when technology fails. Im	Harald Geving	Everyone, Residents, Adults, High tech users
“The wife”	Is willing to employ new solutions as long as a certain minimum of reliability and user friendliness is achieved. Has the ability to veto any new solution. Is not very forgiving when technology fails or just fails to meet expectations.	Eva Geving	Everyone, Residents, Adults, Low tech users
“The kid”	Has an open mind and is quick to grasp new technologies as long as a certain level of user friendliness is met. Has some tolerance for technology failures. Holds no direct power over technology	Kristine Geving, Ida Geving	Everyone, Residents, Kids, Low tech users

	implementation in the home.		
“The guest”	Has no training, no prior knowledge, no expectations. Demands high levels of intuitiveness from the solution. Has zero forgiveness for complicated and/or failing technology.	Not represented	Everyone, Guests, Low tech users

3.3. User environment

The primary user environment is a residential house where most of the lights and heating controls are from the xComfort series by Eaton, and a few lights are from IKEAs Trådfri product line. Three plug-in actuators without any brand name are seasonally in use for decorative lighting. All of these lights are operated by manipulating switches/buttons or other physical remote controls. The Trådfri products are controllable by a mobile app, given that the user is connected to the same network as the Trådfri Gateway, has installed the Trådfri app on their Android or iOS phone, and has configured the app with the correct gateway serial number. Some monitoring systems are non-branded stand alone appliances that display real-time readings such as temperature or power consumption and/or transmit such readings to a database. Future user environments will probably have appliances from other manufacturers, but would in essence be similar to the current setup described above.

3.4. Summary of users needs

Need	Priority	Affects	Current solution	Suggested solution
Operate lights by using regular wall mounted switches	High	Everyone	Works 100%	Allow users to control any appliance regardless of the switch’s manufacturer.
Operate lights by using a mobile phone	High	Adults	Works for IKEA lights when the user is connected to the same LAN as the lights.	Allow users to control all lights from their mobile phone regardless of manufacturer and/or connection method.
Check status of lights	High	Adults	Some xComfort light statuses can be read from a display on the Home Manager unit. IKEA Trådfri	Allow users to view the status of all lights from their mobile phone regardless of manufacturer and/or connection method.

			light status available in app if user is connected to the same LAN as the lights.	
Check status of heating and temperatures	High	Adults	xComfort based readings available from Home Manager display. Stand-alone systems readings available through basic web interface.	Allow users to view the status of all temperatures and heating valves from their mobile phone regardless of manufacturer and technology.
See summary status of the heating and lighting on a unifying dashboard web page.	Medium	Everyone	No such functionality exists today	Can provide live data for a third party dashboard application or other visualization tools..

4. Product alternatives

There are quite a few existing products that offer similar functionality to BachelorPad. This is a short list of the most popular ones, with their biggest pros and cons as they relate to the user environment described in chapter 3.3.

Product	Pros	Cons
OpenHAB 2.0	● Free ● Huge library of addons ● Open source	● Addons must use Java ● Hard to configure ● End user UI is unintuitive
Home Assistant	● Huge library of addons ● Addons can be written in any programming language ● Open source	● $5/mo for cloud (required for Google Assistant)
HomeSeer	● Plug-ins are .NET EXEs	● Not free ● Limits number of simultaneous plugins ● Only some plugins are free ● Closed source
Symcon	● Support for xComfort	● Not free ● Subscription based ● Primarily German speaking users and support forums

4.1. The products role in the user environment

The product will primarily augment the existing light switches by giving the user a second way to control the lights in the house. It will also allow for cross manufacturer control by allowing a light switch from one manufacturer to control lights or a smart appliance made by a different manufacturer. Lastly, the product will allow the users to interact with all lights and smart appliances through a single interface instead of loading each manufacturer's mobile app or moving to a dedicated control device for that appliance.

4.2. Dependencies

This product requires a local computer that can operate continuously in the home, connected to the network and preferably the Internet. Ideally, this would be a dedicated micro computer such as the Raspberry Pi 3 or similar. With its small form factor, low power consumption and a large selection of case designs, it should be possible to place almost anywhere in a modern home. The product can also run on a standard desktop or laptop PC, but this might be less desirable depending on the usage and/or uptime of the machine. A home server or home theatre PC would likely be the best candidate.

Depending on the third party systems that one wishes to communicate with, extra hardware might be necessary:

- For communicating with xComfort, a USB Communication Interface is required. Either the CKOZ-00/03 or the CKOZ-00/14 can be used.
- For IKEA Trådfri there is no extra hardware to connect directly to the computer, but an IKEA Trådfri Gateway is required to control the lights with the IKEA Trådfri app for Android or iOS, and thus also required for this product.
- Most wireless technologies require a special radio transmitter device of some sort. Common frequencies include 433MHz, 868Mhz and 2.4GHz.

5. The products functional properties

1. Expose status of any light through a secure API
2. Expose control of any light through a secure API
3. Expose status of any temperature reading through a secure API
4. Allow user to control any relevant light with less than three presses/clicks/taps
5. Allow the user to get a situation status with less than three presses/clicks/taps
6. Allow the administrator to easily add new devices to the solution
7. Provide a way to add new device types with as little work as possible
8. Allow the system to be controlled by Google Assistant

6. Non-functional properties and other requirements

6.1 Functionality

- The system must be expandable with plugins or other third party programs.
- The system must be multi platform capable (Does not apply where specialized hardware is involved)
- Basic maintenance tasks should be available in an administrator menu.
- The system must be usable on most Windows, Android or iOS devices less than 5 years of age.

6.2 Usability

- The products language shall be English, but must allow for Norwegian characters in names and descriptions.
- The user should not need to perform more than two button presses/clicks/taps to perform the most used actions. (This does not include waking/accessing the phone/tablet/other control device.)
- UI must be optimized for touch input.
- Using the system should be as easy or easier than using the physical switch mounted on the wall.

6.3 Reliability

- Uptime should be at least 99% of the uptime of the host OS.

6.4 Performance

- There shall be no noticeable performance hit with at least 10 simultaneous users accessing the system. (Given that the system has near exclusive access to the resources on a Raspberry Pi 3 or equal hardware)

6.5 Supportability

- It must be possible to select which parts of the system are available for a given user.
- It must be possible to allow access to parts of the system without a login.
- It must be possible to change parts of the configuration without noticeable performance hits.
- It must be possible to communicate with the system through an API.

7. References

Title	Description	More information
Google Assistant	Voice controlled virtual assistant	https://assistant.google.com/
xComfort	Product line for wireless control of lighting and other basic home automation.	http://www.xcomfort.com/
CKOZ-00/03	xComfort Communication Interface	https://datasheet.eaton.com/datasheet.php?model=104928
CKOZ-00/14	xComfort Communication Stick	https://datasheet.eaton.com/datasheet.php?model=168549
Home Manager	xComfort programmable central unit required for “smart” automation of the system	http://www.xcomfort.co.uk/shop/phased-out-products/48-chmu-00-02-home-manager.html
IKEA Trådfri	Product line with wireless light bulbs and remote controls	https://www.ikea.com/gb/en/products/lighting/smart-lighting/
IKEA Trådfri Gateway	Required to control Trådfri products from the Trådfri app	https://www.ikea.com/gb/en/products/lighting/smart-lighting/tr%C3%A5dfri-gateway-white-art-20337807/
Raspberry Pi 3	Micro Computer board	https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/
Openhab	Alternative to BachelorPad	https://www.openhab.org/
Home Assistant	Alternative to BachelorPad	https://www.home-assistant.io/
HomeSeer	Alternative to BachelorPad	https://homeseer.com/
Symcon	Alternative to BachelorPad	https://www.symcon.de/en/

All links verified working on 2019-04-29.

#109

Appendix II

Specifications

v2.0

Revision history

Date	Version	Description	Author
2019-03-01	0.1	Initial version	Harald Geving
2019-03-19	2.0	New scope, rewritten almost everything	Harald Geving
2019-05-20	2.1	Touch ups before submitting	Harald Geving

#109

Table of contents

1. Introduction

This document details each function required from the product in such a way that anyone can easily understand what is to be delivered and how it should work. This information is presented as user stories in this format: *As a <role> I want to <perform task> so that <goal achieved by task>.* This makes it easy to determine whether or not a requirement has been fulfilled by asking the question: *“Can <role> do <task> so that <goal> is achieved?”*
The target audience for this document is assumed to have some experience setting up their chosen third party home automation software, as well as some knowledge of the xComfort[1] system by Eaton[2].

2. User stories

2.1 As an Administrator ...

… I want to configure parameters for my Communication Interface (CI)
so that it will work with my hardware.

… I want to import the list of data points that I already have
so that I don't need to spend much time setting up the system.

… I want to configure parameters for my MQTT broker
so that it will work with my custom setup.

… I want to add xComfort support to my existing home automation system[3]
so that I can include xComfort devices in my rules and automation.

… I want to use any of my devices as a host for this application
so that I can choose hardware and OS freely, without thinking of compatibility.

… I want to have access to the source code
so that I can verify, modify and improve for my own needs.

2.2 As a User ...

… I want to control xComfort units in the same way that I control other smart devices
so that I don't have to use more than one interface.

3. Domain model

3.1 Current situation

No communication with third party systems is possible. All actions and events are confined within the xComfort system.

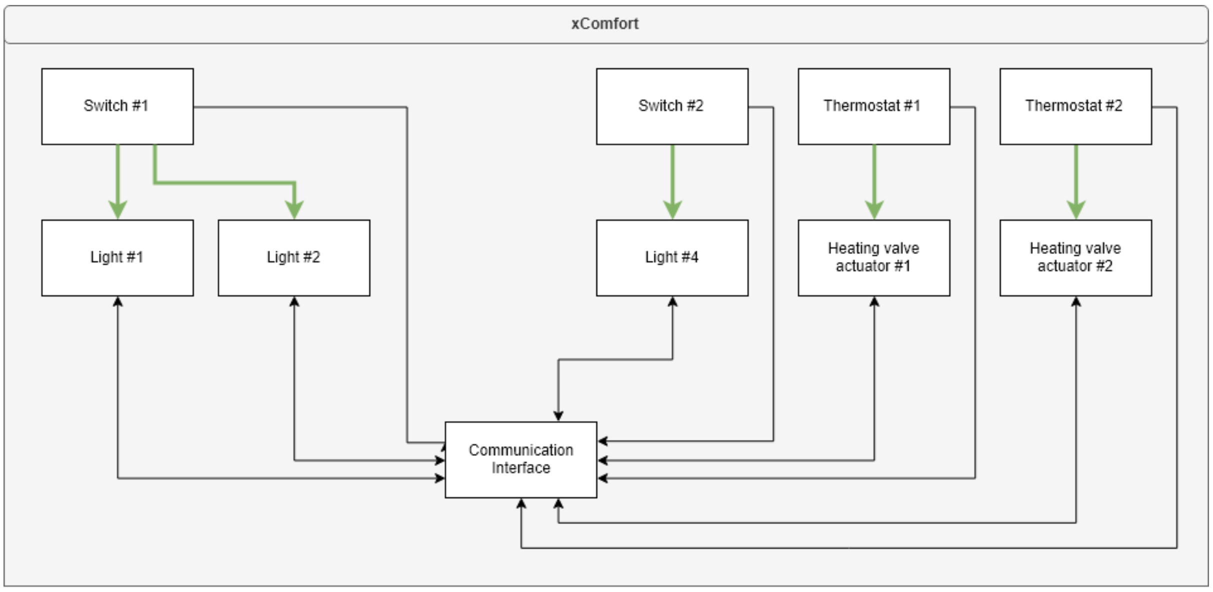


Fig. 1) Internal data flow for xComfort system

3.2 New structure

By adding xComfort Wingman as middleware between the xComfort system and a third party home automation system such as OpenHAB 2.4, Home Assistant or others, actions and events can be transported into and out of the xComfort system. This enables an action in one system to affect a device in the xComfort system and vice versa.

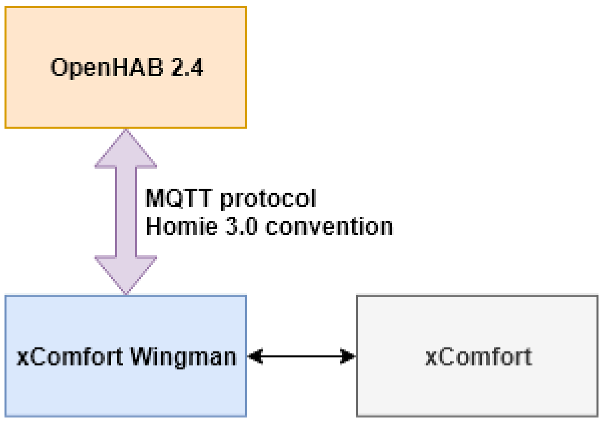


Fig. 2) Simplified view of data flow between systems and their respective Wingman modules

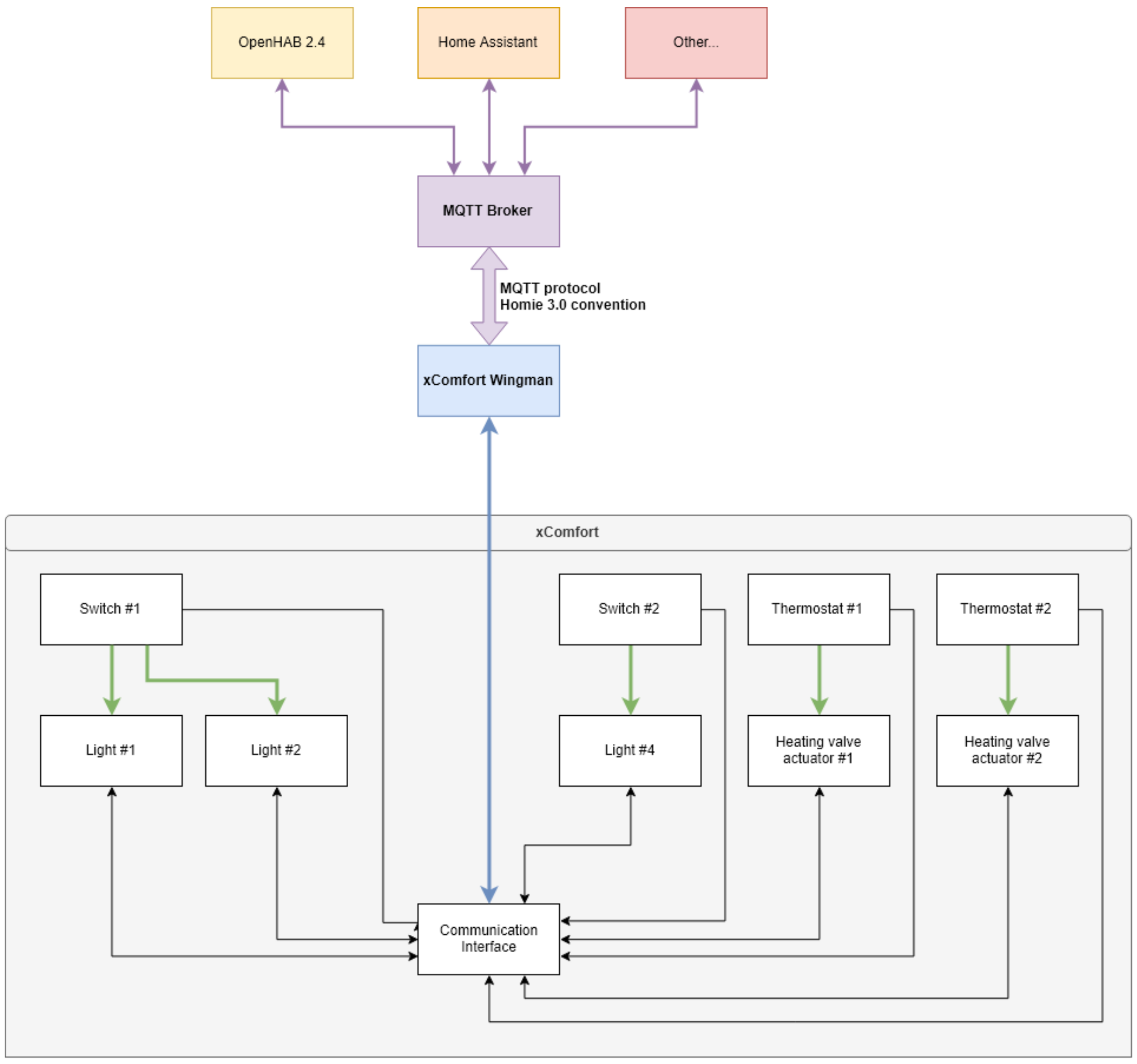


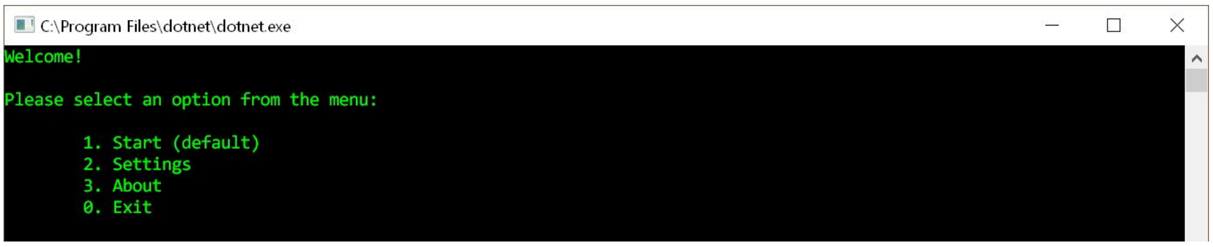
Fig. 3) Data flow between xComfort and several possible back end systems using MQTT

4. Prototypes

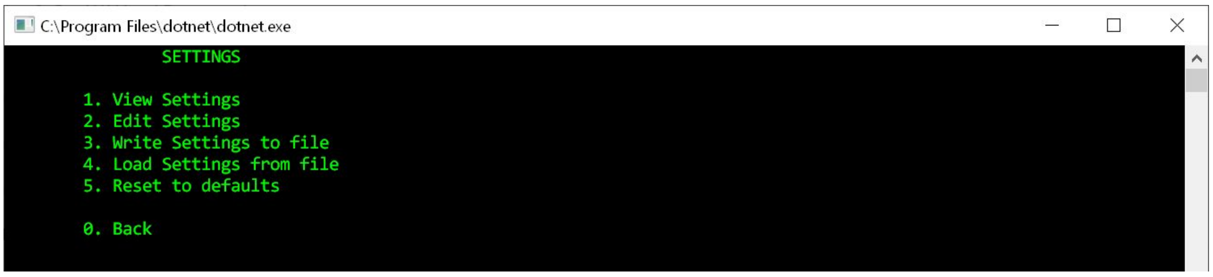
The only part of this application that will have any interface is the setup menu. Everything else is done within the third party home automation software.

4.1 Menu

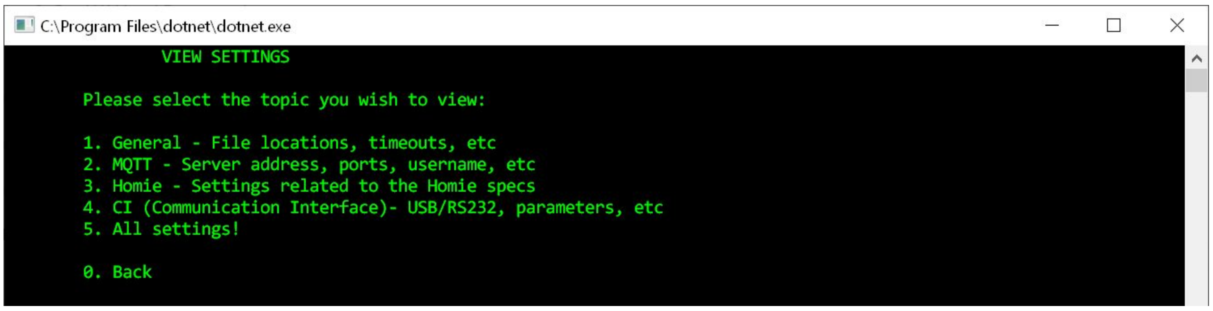
These are cropped screenshots that show how the menu can look like.



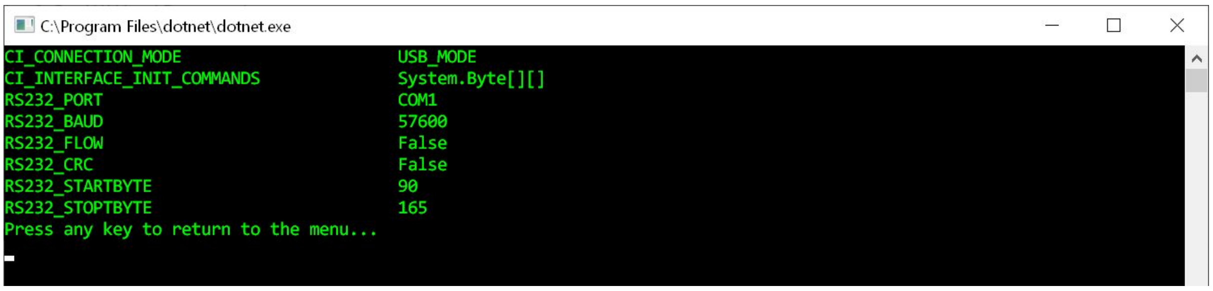
```
C:\Program Files\dotnet\dotnet.exe
Welcome!

Please select an option from the menu:

        1. Start (default)
        2. Settings
        3. About
        0. Exit
```



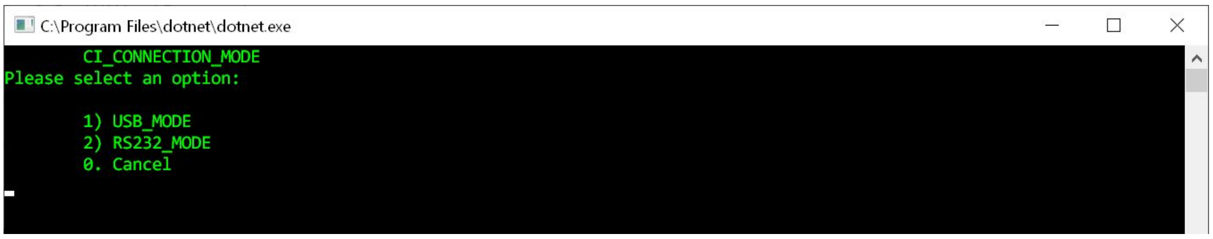
```
C:\Program Files\dotnet\dotnet.exe
                SETTINGS

        1. View Settings
        2. Edit Settings
        3. Write Settings to file
        4. Load Settings from file
        5. Reset to defaults

        0. Back
```



```
C:\Program Files\dotnet\dotnet.exe
                VIEW SETTINGS

        Please select the topic you wish to view:

        1. General - File locations, timeouts, etc
        2. MQTT - Server address, ports, username, etc
        3. Homie - Settings related to the Homie specs
        4. CI (Communication Interface)- USB/RS232, parameters, etc
        5. All settings!

        0. Back
```



```
C:\Program Files\dotnet\dotnet.exe
CI_CONNECTION_MODE                      USB_MODE
CI_INTERFACE_INIT_COMMANDS              System.Byte[][]
RS232_PORT                              COM1
RS232_BAUD                              57600
RS232_FLOW                              False
RS232_CRC                               False
RS232_STARTBYTE                         90
RS232_STOPTBYTE                         165
Press any key to return to the menu...
_
```



```
C:\Program Files\dotnet\dotnet.exe
        CI_CONNECTION_MODE
Please select an option:

        1) USB_MODE
        2) RS232_MODE
        0. Cancel
_
```

5. References

1. "Xcomfort – elektroinstallasjonens revolusjon! - Sinus." 27 Nov. 2007, http://www.sinusmagasinet.no/artikler/2007/november/xcomfort/190. Accessed 20 May. 2019.
2. "Moeller - Eaton." https://www.eaton.com/Eaton/ProductsServices/ProductsbyName/Moeller/index.htm. Accessed 20 May. 2019.
3. "openHAB 2.4 Release | openHAB." Accessed May 17, 2019. https://www.openhab.org/blog/2018-12-17-openhab-2-4-release.html.