Kalle Koskinen

Development of Cost-efficient Embedded Control System for Cryo°Cabin

Metropolia University of Applied Sciences

Bachelor of Engineering

Information and communications technology

Thesis

28 April 2018

Metropolia

Preface

This project was clearly the most challenging technological project I've ever been involved in. In the beginning of this project my knowledge of the techniques and technology was well below sufficient level but the demanding nature of this project made sure that this will not be the case after the finish line was reached. I wish to thank all the people at CTN and CTF who made this project possible, especially Jan Eklund and Juha Yliollitervo. I also wish to thank Sakari Lukkarinen for giving me great tips when composing this thesis.

The smell of a burnt transistor is the smell of technological progression.

28.4.2018
Kalle Koskinen

Author Title Number of Pages Date	Kalle Koskinen Development of Cost-efficient Embedded Control System for Cryo°Cabin 47 pages + 2 appendices 28 April 2018
Degree	Bachelor of Engineering
Degree Programme	Information and communications technology
Professional Major	Health informatics
Instructors	Jan Eklund, Managing Director Sakari Lukkarinen, Senior Lecturer

This thesis was commissioned by Cryotech Nordic AS and its subsidiary Cryotech Finland Oy. The project was triggered by a demand for a simplified cryotherapy system. Thus, the aim of the thesis was to design and develop cost-efficient embedded control system alternative for current Cryo°Cabin cryotherapy system.

The success criterion of the project was to create a control system prototype whose cost-effiency and reliability would allow the creation of a new product at a reduced cost in the future. The prototype should have as much of the features of Cryo°Cabin system as possible.

The project consisted of automation, electrical and software engineering, which were required for ensuring the operation of the system logic, components, devices and software.

The outcome of the project is a prototype which, with its cost-effiency, exceeded the given targets. The development and testing of the prototype continues after this thesis and it will be used in other projects for Cryotech Nordic in the future.

Keywords	Cryotherapy, Embedded control systems, Python, Android

Metropolia

Tekijä Otsikko Sivumäärä Aika	Kalle Koskinen Kustannustehokkaan sulautetun ohjausjärjestelmän kehitys Cryo°Cabin:lle 47 sivua + 2 liitettä 28.4.2018
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Tieto- ja viestintätekniikka
Suuntautumisvaihtoehto	Hyvinvointi- ja terveysteknologia
Ohjaajat	Jan Eklund, Toimitusjohtaja Sakari Lukkarinen, Lehtori

Tämä insinöörityö toteutettiin Cryotech Nordic AS:n ja Cryotech Finland Oy:n toimeksiannosta. Insinöörityön tavoite oli suunnitella ja toteuttaa kustannustehokas vaihtoehto nykyiselle Cryo°Cabin kryoterapialaitteen sulautetulle ohjausjärjestelmälle. Projektin taustalla oli kasvanut kysyntä edullisemmalle kryoterapiajärjestelmälle.

Projektin onnistumisen kriteereinä oli luoda ohjausjärjestelmän prototyyppi, jonka toimintavarmuus ja kustannustehokkuus mahdollistaisi tulevaisuudessa uuden tuotteen luomisen alennetuilla kustanuksilla. Prototyypissä pitäisi olla mahdollisimman paljon jo olemassa olevan kryoterapiajärjestelmän ominaisuuksista.

Projekti sisälsi automaatio-, ohjelmisto- ja sähkösuunnittelua, joita vaadittiin ohjausjärjestelmän toimintalogiikan, komponenttien sekä laitteiden ja ohjelmiston toiminnan varmistamiseksi.

Projektin lopputuloksena syntyi prototyyppi, joka kustannustehokkuudellaan ylitti positiivisesti annetut rajat. Protyypin kehittäminen ja testaaminen jatkuu myös insinöörityön jälkeen ja kyseistä tuotetta hyödynnetään jatkossa muissa Cryotech Nordicin projekteissa.

Avainsanat	Kryoterapia, Sulautettu ohjausjärjestelmä, Android, Python

Metropolia

Contents

List of Abbreviations

ADC	Analogue to digital signal converter
Cron	Time-based job scheduler in Unix-like operating systems
Cryo°Cabin	Cryotherapy system developed by Cryotech Nordic
CTF	Cryotech Finland Oy
CTN	Cryotech Nordic AS
DAC	Digital to analogue signal converter
GPIO	General purpose input output
GUI	Graphical user interface
IDE	Integrated drive electronics
I/O	Input/Output
LED	Light emitting diode
LN_2	Liquid nitrogen
NPN	Transistor type
PC	Personal computer
PV	Process variable
PSU	Power supply
R&D	Research and development
RPM	Revolutions per minute

SBC	Single-board computer
SP	Set Point
Tkinter	Python’s standard GUI package
USB	Universal serial bus
USD	The United States dollar
WBC	Whole-body cryotherapy
WiFi	Wireless Fidelity

1 Introduction

Different kinds of cryotherapy systems and equipment are developed at accelerating rate. This is the due to numerous high profile athletes who have taken cryotherapy as a part of their training routine. (Kemp 2016; Lelinwalla 2015) When it comes to cryotherapy systems, there are two distinctive types of them: large room-sized systems called cryochambers and cryosaunas, which are barrel-like cabins. There are a few aspects that separate these two systems, but one thing that they both have in common, is a rather high price tag starting from around 40000 US dollars (USD) up to over 100000 USD. (Domin 2015; Marshall 2016)

The combined price of cryotherapy system, the coolant and other business-related expenses towards whole-body cryotherapy (WBC), for example the cost of business premises, makes it risky especially for small businesses to consider creating a flourishing cryotherapy business. This is why there has been a demand from the market for a budget alternative of cryotherapy system.

The purpose of this thesis is to develop a cost-efficient alternative for the Cryo°Cabin's current embedded system for Cryotech Nordic. The idea is to decrease the costs of the components used in the embedded system while retaining the same usability and product reliability. The cost of the finished product will be significantly lower compared to the current system thus allowing the creation of completely new model of Cryo°Cabin. The longer term aim is to create a whole new market for this new budget product while the current product remains a high end system.

This thesis contains 8 sections. Following the Introduction, sections 2 and 3 provide the theory around cryotherapy and the control system. The fourth section introduces the company while the fifth section focuses on the requirements for the new product. Section 6 covers the process of design and development of the product. The seventh section has the results of the project while the section 8 contains some final thoughts of the project.

2 Whole-body cryotherapy

The WBC treatment was first used in Japan by Dr. T. Yamauchi in 1978. The idea was to find out how extreme cold temperatures affect rheumatoid arthritis patients when applied on their skin for a short period of time. The conclusion was that the application of the extreme cold on the patient's skin released great amounts of endorphins which eased the pain of the subjects. Since then WBC has been developed in Europe and in the US. Today, WBC is utilized by top-tier athletes all over the world for faster recovery from training sessions as well as patients with different kinds of conditions. (Lombardi et al. 2017)

2.1 WBC systems

There are two main types of WBC systems in the market, that use liquid air or nitrogen (LN_2) as a cooling method: cryogenic chambers (cryochambers) and cryosaunas (Yliollitervo et al. 2017). They both share the same operating principle, but differentiate when it comes to size, operating temperature and the equipment required for the patient. There are also cryogenic chambers that are electrically cooled, but this thesis does not include them, as their operating principle differs significantly from the others.

2.1.1 Operating principle

The main principle of a WBC system is simple: air inside the system is cooled to extreme proportions using either LN_2, which has a boiling point of -196 °C or liquid air, which is a mixture of different gases compressed to liquid form and has a boiling point of around -194 °C (Yliollitervo et al. 2017: 3). If LN_2 is used in WBC system, the system always requires careful surveillance due to treacherous nature of N_2 gas.

There are two ways of inputting the liquid gas to the cryotherapy system: direct input and indirect input. Direct input means that liquid gas is sprayed in small amounts straight into the treatment space where the patient is. This method cools the space quickly, but the downside is, that the extremely cold liquid presents a risk of a frostbite to the patient. (Yliollitervo et al. 2017)

The second method, indirect input, is a method where liquid gas is sprayed into a container, so that the cold liquid evaporates and mixes with air. This mixture of gases is then channeled into the treatment space from the container. Indirect input is not as quick method of cooling the treatment space as direct input, but it is easier to control and has way lower risk to cause a frostbite. (Yliollitervo et al. 2017)

2.1.2 Cryochamber

Cryochambers (Figure 1) are usually large, room-sized, containers where patients wear protective masks, headgear, gloves, shoes and underwear. Protective mask is used to cover the patient's nose and to prevent the cold air causing patients discomfort when breathing. These chambers can usually take multiple persons at a time, although there are also smaller chambers that take only one person at a time.

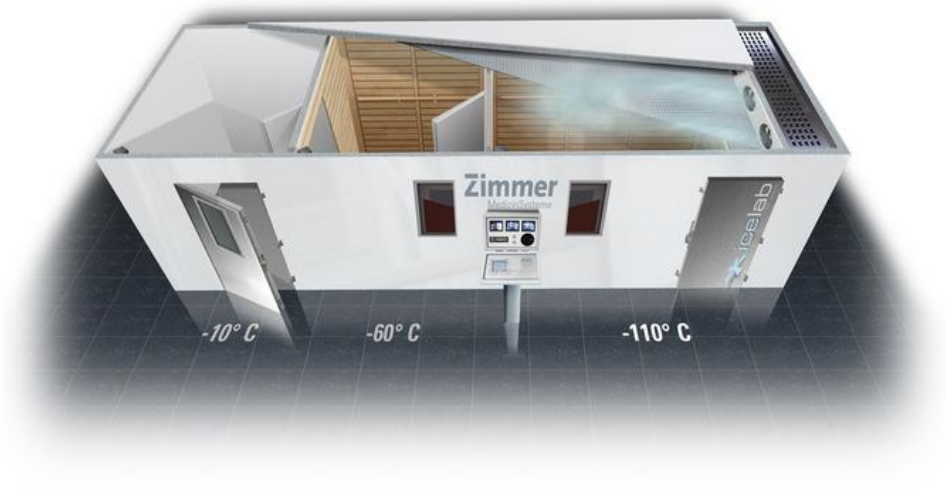

Figure 1. Cryochamber with 2 precool areas using temperatures of -10 °C and -60 °C. The main room operating at -110 °C. (Sequence 2016)

Depending on the model of the chamber, the container can have up to two precool areas like in Figure 1, which are cooled to significantly milder temperatures, around -10 to -60 °C, before the main room, which operates around -100 to -110 °C. Smaller chambers do not have these precool areas. The treatment usually lasts around 2-3 minutes. The patient can move freely while in the cryochamber. (Muhic 2016; Sequence 2016)

2.1.3 Cryosauna

Cryosaunas are 2 to 3 meter tall barrel-like cylindrical cabins made for one person where the patients only wear protective shoes and underwear. The cryosaunas are open from the top so the patient's head is not exposed to the extreme cold, allowing patients to use cryosaunas without headgear or a protective mask. As figure 2 illustrates, no gloves are required since the patient can rest his/her hands on the rim above the cold air mass. As cryosaunas are open from the top, patients suffering from claustrophobia can also take cryotherapy sessions in these systems. (Muhic 2016)

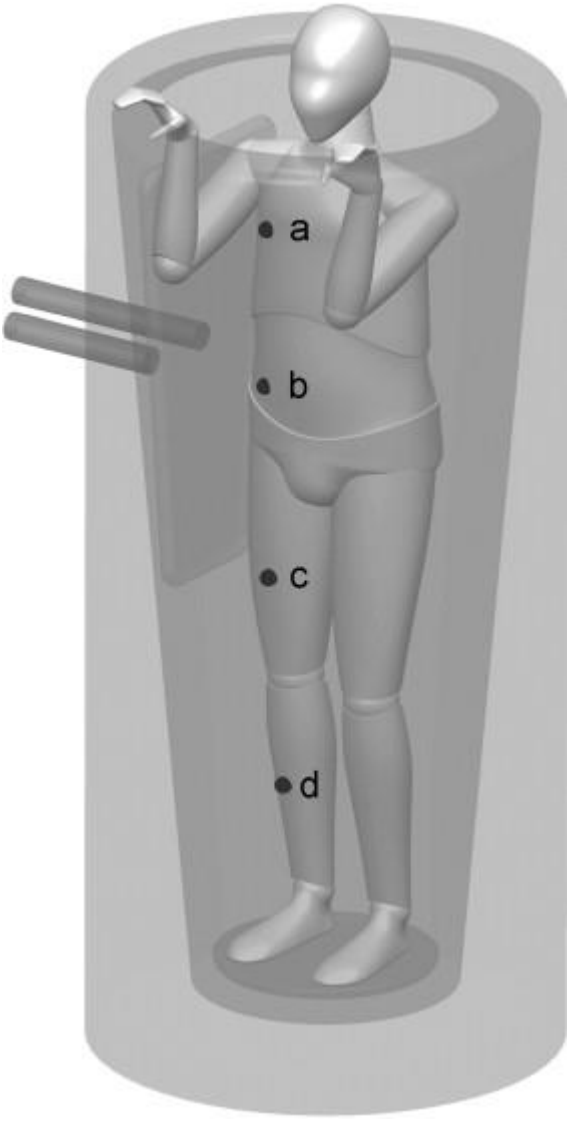

Figure 2. The orientation of the patient during a treatment in a cryosauna. (Savic et al. 2013)

Before treatment, the cryosauna is precooled usually around to 90 % of the operation temperature after which the patient steps into the cryosauna. The treatment usually lasts 2 to 3 minutes and the operation temperature varies from -80 to -140 °C. It is recommended for the patient to move and rotate slowly in cryosauna when the session is in progress. This is due to fact that cold air is usually channeled into the treatment space from one spot, usually from the back of the cryosauna. In Figure 2, the cold air is channeled into the treatment space through the pipes right of the patient. The rotation of the patient ensures that the cold air will not be concentrated to one spot of the body and causing discomfort to the patient. (Eklund 2018)

3 Embedded control system

3.1 Embedded system

The characteristic of an embedded system is a computer-aided system, although it might not be apparent to the user. Embedded systems come in various sizes and a system can be a mixture of multiple components and appliances, or it could just be a single computer. Usually, an embedded system performs a dedicated function or a task, or it processes several smaller tasks. In a large complex unit, such as a modern car, a group of small embedded systems performing different tasks form a network of systems to create the best possible platform for a user to enjoy multiple different features. (Barr 2017)

Embedded systems were created for the Apollo space program in the United States in the 1960's (Tomayko 1988). Nowadays, since the processors have developed to be cheaper, smaller and more powerful, the market for embedded systems, like the figure 3 illustrates, has skyrocketed and today an estimation of 98 % of every manufactured processor unit in the world ends up being a part of an embedded system. (Barr 2017)

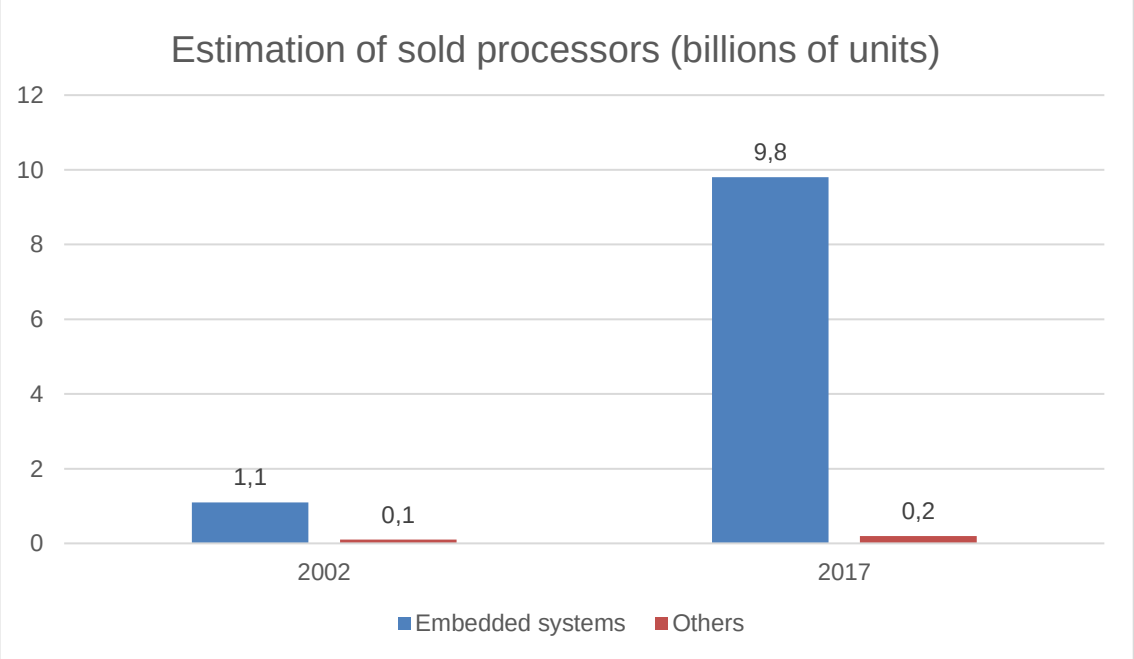

Figure 3. Estimation of sold processors (Barr 2017)

The user of an embedded system might not be aware of the computer that lies inside, because the user interface of the system can be compact with very little options to play with, or it might not exist at all (Forrai 2013). The cruise control in a modern car, for example, is an embedded system. The driver of the car can influence the system by switching it on or off and when it is activated, the driver can set the desired speed to be maintained.

Earlier, every component of an embedded system had to be tailored just for the occasion because of the lack of performance of the used parts. Nowadays, customized components are still being used in different embedded systems, but if the embedded system in question is performing similar actions to regular PC's, like TCP/IP actions, it's more beneficial to use a PC solution for the embedded system, as they offer enough performance with good price. If optimization is needed, it can be done in the software. (Embedded systems 2017)

3.2 Embedded computers

Usually these PC solutions for embedded systems are called embedded computers, industrial computers or embedded box-computers, which name comes from its box-like appearance (Figure 4). An embedded computer differs from a regular PC with its size, durability and cooling. There are often more IO-ports (Input/Output) than usual, that are required for connecting different devices and sensors that might be part of the embedded system. These differences between a regular PC and an embedded computer are the result of the fact that the the embedded computer needs to stay functional under constant stress for long periods of time without maintenance and it has to fit in small-sized systems with ease. In other words, it should be *embeddable* for a system. (Fanton 2014)

Figure 4. Embedded box-computer with metal housing designed as heatsink (Uibx-230-bt-n2/2g-r11 2016)

There are roughly two types of embedded computers: computers with housing and computers without housing, the latter ones being called single-board computers (SBC). In computers with housing the computer circuit is installed inside a durable metal case, which operates as a heatsink for passive cooling. SBC, according to its name, is just a circuit board with the standard PC connections, for example HDMI or VGA-port for a display. Which computer type works best, depends heavily on the application and the system where the computer should be installed. (Fanton 2014)

Most of the time, computers with housing are more powerful and stable for tasks with greater need of performance, thanks to its passive cooling, which makes it possible to install more powerful components on the computer. On the other hand, if the task is simple and the computer has to be small-sized, an SBC is the best option because they are available in a size of a matchbox (Figure 5) (Brown 2016). The price-range of embedded computers is very vast. The least powerful SBC's cost only around 10 to 20 euros, the intermediate SBC's and the cheapest box-PC's around 100 to few hundred euros while the most powerful computers with housing for extremely demanding tasks could cost several thousand euros (Atwell 2015; Uibx-230-bt-n2/2g-r11 2016; Advantech MIC-7900-S5A1E Sulautettu teollisuustietokone 2016).

Figure 5. Matchbox-sized SBC with Ethernet and USB connections (Brown 2016)

3.3 Control systems

A control system controls other devices and systems with control loops. A control system can be mechanic, electrical or computer-aided, ergo *embedded control system*. A simple control system can be just a device or a component, which bases its functionality to physical or chemical reactions of its ingredients of fabrication (a thermostat, for example). This functionality can be represented with two values: SP (Set point) and PV (Process variable). (Forrai 2013: 2; Control system 2017)

SP is a value, which is critical to the control system. It is a pivotal point to the embedded system to base its actions on. For example, the SP value of a thermostat is a temperature where it either activates a device or deactivates it. If the thermostat is mechanical, its materials, e.g. metals or gases, react to the change of temperature and contracts or expands. This gives the connected device a signal to power on or off. The connected device in this case could be a radiator, where hot water is fed when the surrounding air temperature drops below the SP. This continues as long as the surrounding air temperature reaches the SP again and the water feed is closed. An electrical system works in

very similar fashion: the thermostat closes or opens an electrical circuit and the connected device, such as heating resistor, powers on or off (Figure 6). (Control system 2017)

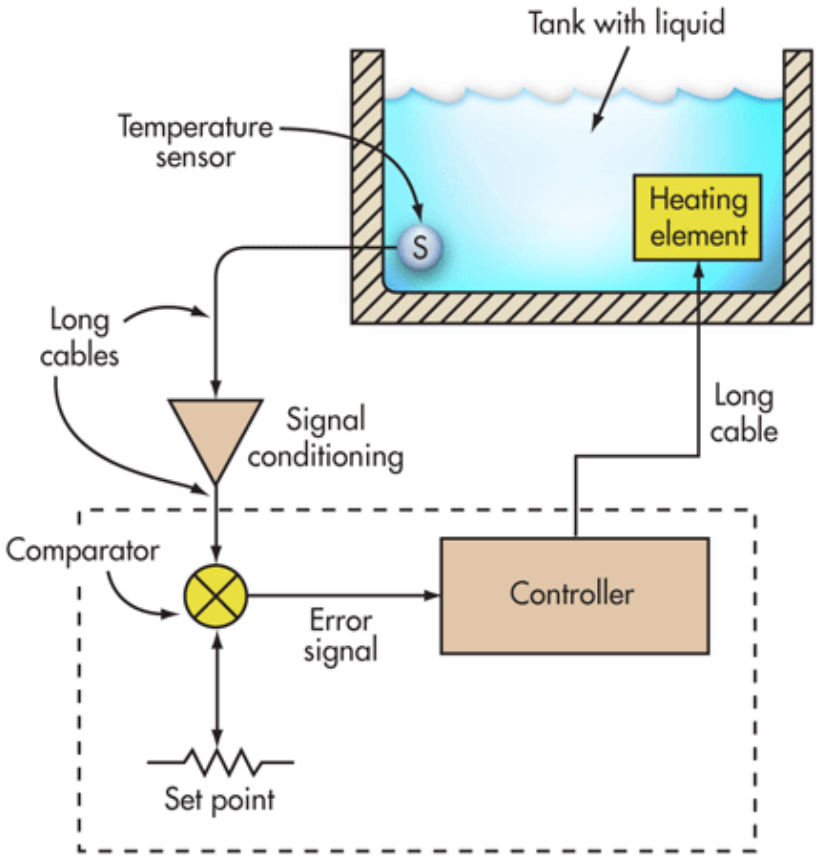

Figure 6. Control system where the temperature of the liquid is maintained using temperature sensor and heating element (Frenzel 2013)

A PV is a value, which is constantly measured. PV always exists even if the user of the system is not aware of it. This is usually the case in very simple control systems, where there are no indicators for the PV and the exact value of the PV could be determined with separate tools. User can only determine an estimation of the PV by checking the status of the system and seeing if it's greater or less than SP. It is extremely important that the measure methods of the control system work correctly as corrupted measurements compromise the stability of the system. When choosing the methods for measurements of the control system, the prevailing conditions must be taken into consideration. (Control system 2017)

3.4 Software

An embedded control system requires an operating logic, a computer program, for it to work. The language used in the creation of the software for the system heavily depends on device or computer that controls it. If the device is a PLC (Programmable logic controller), essentially a computer designed for automation processes, the language for the software is chosen from the IEC 61131-3 standard. The standard defines 5 different languages which are used to write software for PLC, for example ST (structured text). If again the device influencing the control system is an embedded computer with operating system like Microsoft Windows or Linux, the language for the software can be chosen from various languages. It is also possible that the control system operates as a mixture of these two and the tasks are shared between these two devices. (John & Tiegelkamp 2010: 99)

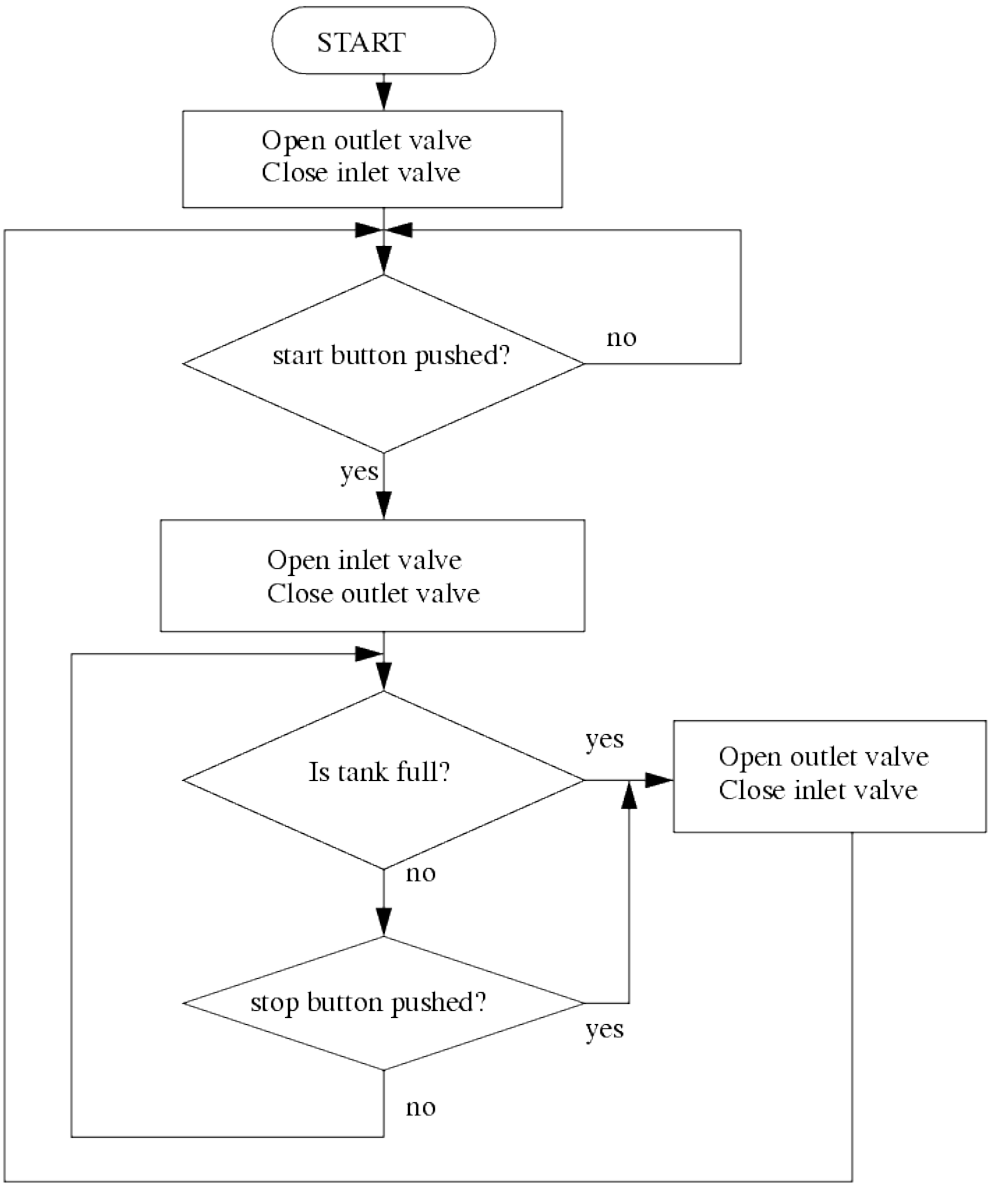

Figure 7. Flowchart illustrating control logic of system controlling a valve (Flowchart based design 2010)

In the beginning of software development, it is extremely important to know what kind of devices and components the control system is dealing with. Also, the operating principle needs to be decided before the development. This includes the information of how the user can interact with system. It is also possible that the user cannot interact with the system at all. The operating principle of the system can be depicted with state diagram or flowchart as seen in figure 7 and appendix 1. (Forrai 2013: 21-30; 242)

4 Cryotech Nordic AS

4.1 Background

Cryotech Nordic AS (CTN) specializes in non-invasive, non-surgical health and aesthetic treatment devices. CTN was founded in 2013 in Harju County, Estonia. Its subsidiary Cryotech Finland Oy (CTF) was also founded in 2013 in Espoo, Finland. Today, the CTN factory in Estonia is still located in Harju County, but the CTF's office and the research and development (R&D) center, also known as the CTN Center, in Finland changed its location to new premises in Vantaa in summer of 2017. The factory in Estonia produces all the frames of the products while the CTN Center in Finland focuses both on the research and product development and the final assembly of the products. At the time of writing, CTN products can be found in more than 30 countries all the way from North America to Australia.

Figure 8. CTN products: Octagon Duo, Cryo°Cabin, Monolith Quattro, Altium Duo. In reality the Cryo°Cabin is twice as tall as the other machines. (Cryotech Nordic 2018)

Besides the Cryo°Cabin, Monolith Quattro, Altium Duo and Octagon Duo are the other products of CTN as seen on the figure 8. Monolith Quattro is a cryolipolysis device for body fat removal. Altium Duo is a High Intensity Focused Ultrasound (HIFU) device for non-surgical face lifting and Octagon Duo is a lipolysis laser, lipolaser, device for bodyscuplting.

4.2 Cryo°Cabin

Cryo°Cabin is a cryosauna-type of a WBC system created by CTN. The first model of the Cryo°Cabin was developed in the end of 2015. The first shipments to customers were made in the beginning of 2016. Its control system relies on Windows platform with the help of high-end PLC. This combination makes it extremely reliable. The newest model of 2018 has some differences when compared to the first model of 2016. The visible changes are mostly aesthetical with some adjustments of the operating principle aimed for better user experience of the Cryo°Cabin. The most important change, however, is the vastly improved usability through some material and structure changes.

Figure 9. Cryo°Cabin model 2018 with brown interior and black exterior (Cryotech Nordic 2018)

The model of 2018 (Figure 9) has 2 different sized monitors for user interaction and temperature monitoring. The 24” touchscreen is used for controlling the device, while the smaller 10” monitor shows current temperature, time elapsed and image of thermal camera to the customer. To control the gas flow of the system, the Cryo°Cabin has a high-quality three-phase induction motor system. There is also a patented Vortex® active gas circulation system, which allows the Cryo°Cabin to consume less liquid coolant. The Cryo°Cabin can be used with both LN_2 and liquid air. The flow of the liquid coolant is controlled with solenoid valve, and the system uses an indirect input of the coolant, as LN_2 or liquid air is sprayed into a container for evaporation.

4.3 Competitors in the market

There are other manufacturers in the market who have managed to build a cryotherapy system, but only few of them have managed to establish themselves to sell cryotherapy systems worldwide. Some of them manufacture cryochambers, while others manufacture cryosaunas. In addition, every notable manufacturer has its own network of distributors which makes it difficult to tell if the company in question is only a distributor or if they manufacture the product themselves. Table 1 illustrates all the companies that advertise a cryotherapy system as a manufacturer. Some of the companies are part of a larger enterprise which are mentioned in parenthesis. The company status in the table is an estimation of the size of the company with all its products and services taken into consideration.

Table 1. Competitors

Company name	**HQ location**	**Product**	**Company status**
BOC (Linde Group)	UK (Germany)	Cryochamber	Established
Cryo innovations	USA	Cryosauna	Growing
Cryomed	Slovakia	Cryosauna	Growing
Cryoness (Asperia Group)	Poland	Cryosauna	Startup
Cryoniq	Slovakia	Cryosauna	Growing
Cryoscience	Poland	Cryosauna	Growing
Grand Cryo	Russia	Cryosauna	Startup
Impact	USA	Cryosauna	Established
Juka	Poland	Cryosauna & -chamber	Established
Krion	Russia	Cryosauna	Established
Kriosystem life	Poland	Cryochamber	Startup
Mecotec	Germany	Cryochamber	Established
Medner medizintech	Germany	Cryosauna	Established
Metrum	Poland	Cryochamber	Established
Revocryo	USA	Cryosauna	Startup
Us cryotherapy	USA	Cryochamber	Growing
Vucuactivus	Poland	Cryosauna	Startup
Zimmer	Germany	Cryochamber	Established

The most notable thing about table 1 is the location of the competitors. Most of them operate from Central Europe, with a few in the USA and Russia, but none in Eastern Asia like Japan, South Korea or China.

5 Requirements for new product

When designing a new product for the health technology market, it is essential to understand the requirements of this business field. The operational reliability has to meet certain standards so that the system is safe to use in every situation. Only trained personnel can operate the Cryo°Cabin and this fact makes the design process slightly easier, since the designer can be sure, the user of the system has a basic knowledge of how the product works.

If the product is aimed for a market in a certain country, it is also extremely important to know the legislation of the aimed destination. The regional differences can be drastic when a country can treat Cryo°Cabins as a non-medical device while another country next to it can classify Cryo°Cabin as a medical device. When a device has been classified as a medical device, the requirements of the product get higher. The components used must have certain seal of approval, for example FCC Declaration of Conformity or CE European Conformity. This can be also the case when the product is not a medical device, but the country of destination requires certain standards for electrical devices.

5.1 Cost reduction

To really make a difference in the current cryotherapy system market, the new system has to be at least 66 % less expensive to manufacture than the current Cryo°Cabin. The percentage could be even higher, up to 75 %. These rates have been found by making a market analysis in the developing countries, where the new product could really make a great impact with its cost-effiency. This drop in price sets limits to the budget where the designer can operate when choosing components for the control system. It is also assumed that the new Cryo°Cabin system will use the same physical frame as the current Cryo°Cabin, with minimum amount of modifications. Manufacturing only one frame type offers savings in mold costs, but limits reducing the total costs to the control system.

5.1.1 Devices

There are over 20 devices connected to the current control system. To reach the 66 % price drop for the new product, some of them have to be removed or replaced with reduced-cost alternatives. The devices that do not affect the basic operating principle of the Cryo°Cabin can be stripped off.

The most important devices for the Cryo°Cabin are the motor unit, the solenoid valve and the temperature sensors. These devices will remain the same for the new system as they are proved to be extremely reliable and their availability from the suppliers is good. Also the small fans of the system will stay the same as their price is already good enough.

The current Cryo°Cabin has a mechatronic lift to easily adjust the patients head above the rim. Although this device is very useful, it is not a necessity and thus costs can be saved if this device is left out from the new product. It also requires a few relays to operate and these can be left out as well. An air compressor for automatic door opening can be removed from the system for it's just a nice addition for the user experience.

5.1.2 Control system

The current control system of Cryo°Cabin consist of a switchboard and a computer unit with two monitors. The high-quality components of the switchboard are somewhat costly. The system include PLC, motor control unit, 2 power supplies, 60 terminal blocks, relays, residual current device and the attachment panel.

A motor control unit is essential when using a three-phase motor unit. Therefore it has to remain untouched when creating the new control system. Also, the attachment panel itself will remain the same since there is no need for a change: it's easier for the company to only order attachment panels of one type which will fit both versions of the Cryo°Cabin.

To really make the control system cost efficient, the embedded computer unit must be replaced with a cost-efficient alternative. The two monitors of the unit, the computer itself with a Windows license can be easily replaced with Linux based system and the two monitors are not a necessity for the system to operate.

5.2 Product reliability

To ensure the safe use of the Cryo°Cabin, the devices and the control system must meet a certain level of reliability. Also, the user must feel secure when operating the cryotherapy system and he/she has to be able to trust the system. This can be achieved by testing the created system and analyzing the data from the current system. Especially the data gathered from the current system will show what kind of technical problems the Cryo°Cabin might face in the future.

The second part of the reliability is the software. It must be designed in a way that it can either recover from errors it might face, or if it cannot recover from an error, it never compromises the health of the patient. This can be achieved by making a risk analysis that shows the most vulnerable parts of the system. The current software bases its operations on a signal data gathered from the sensors mounted inside the cabin. If this data is corrupted in any way, the software must recognize the situation and adapt to it. If the situation is unadaptable, the system needs to stop its operations and if possible, inform the user about the error it encountered. Also, the devices connected to the control system have to be set up in such a way that a possible software crash will not cause danger to the user or to the patient.

The combination of the devices, the switchboard and the software define the operational reliability of the whole product. The reliability can be measured for example counting the failures per year or failures per runtime. A failure is counted when any part of system dysfunctions. The current Cryo°Cabin has really high operational reliability. The new product must meet these reliability standards.

6 Design & development

This chapter covers the process of production from the first prototype to the fourth and the final version. The chapter is divided to parts and one part covers the phase of the production in question with detailed information about changes in the project. This thesis covers the production part of the project from the start at the end of March, 2017 to the end of December, 2017.

6.1 Design decisions

When the designing process begun, the first thing was to determine what could be left out of the product and still maintain the usability of the product. Windows platform itself is a rather expensive operating system and it was clear that it needed to go. Linux based operating systems are usually completely free and their compatibility with other devices is great. This fact made it easy to choose a Linux based computer for the new embedded system. The only problem with changing Windows to Linux was that the current Cryo°Cabin software could not run on Linux platform so it would need a brand new software that was designed just for this application.

The next step of the process was to decide what kind of embedded computer would run the control system. The debate was between Arduino Uno and Raspberry Pi (Figure 10), since both of them are very small sized, they have pretty good availability from the vendors, their performance is sufficient to run the Cryo°Cabin software and they both have numerous I/O-ports to control other devices. Raspberry Pi was chosen because it had bluetooth and WiFi (Wireless Fidelity) connections, it has the option for graphical windowed user interface and it was familiar to the developer from earlier school projects.

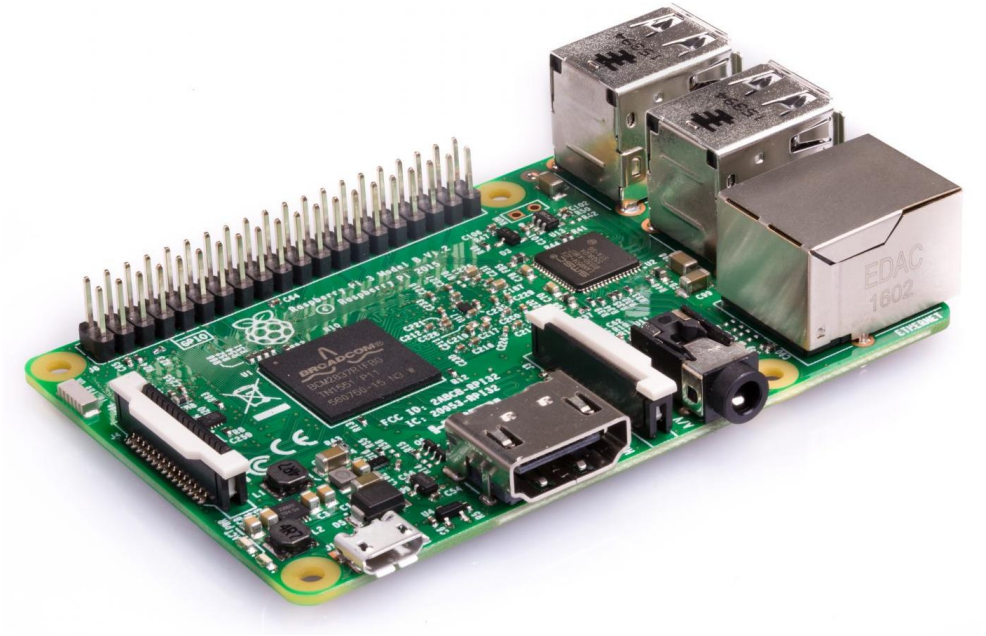

Figure 10. Raspberry Pi 3 model B, SBC with numerous connections. (Raspberry Pi Model 3 B 2018)

The language of the new software was a difficult decision to make since there were many options to choose from. The languages that were tested before the decision were C++, a combination of PHP and Javascript, Java and Python. The three last mentioned options were the strongest languages for the developer and the final decision was to create the software with Python, as it would offer great tools to operate the I/O-ports. What is more, software development with Python is quite simple and fast.

The last decision to make before the prototyping phase could start was to decide how the cabin would be operated and what devices the control system would control. If the monitors of the current system would be left out, it would drop the price of the system significantly but it also would mean that the way the user interacts with the system and vice versa would change considerably. Also the other devices like the lift, door pusher, heater, drying fan and thermal camera were under consideration to be removed from the system. They provide a great additional value for the product, but their significance to the basic operational principle is small. The decision was to remove the monitors and replace them with a simple push button that would control the system. A single led light would be sufficient to give the user information of the system status. All unnecessary devices would be removed at first, but taking them back could be taken into consideration, if the situation needed so.

6.2 Pre-prototype phase

The design process started from absolute scratch. No components were ordered for the start of the project and every component used in the first stages of the project were spare parts used in previous CTN projects. The Raspberry Pi was lent from Metropolia.

6.2.1 Control system

At first, there were only two components in the system: the Raspberry Pi and the solenoid valve. The question was, how to operate the valve when it requires 24 V of supply voltage and the programmable Raspberry Pi GPIO pins can only output 3,3 V of supply voltage? The solution was to use solid state relay which could be activated with 5 V supply voltage. The Pi has two 5 V non-programmable GPIO pins and one of them worked as the power supply for the relay. To control the relay, a NPN transistor had to be installed on the ground side of the relay to work as a switch. The relay activated when 3,3 V signal from a GPIO pin was inputted to the base of the transistor. This is depicted in figure 11.

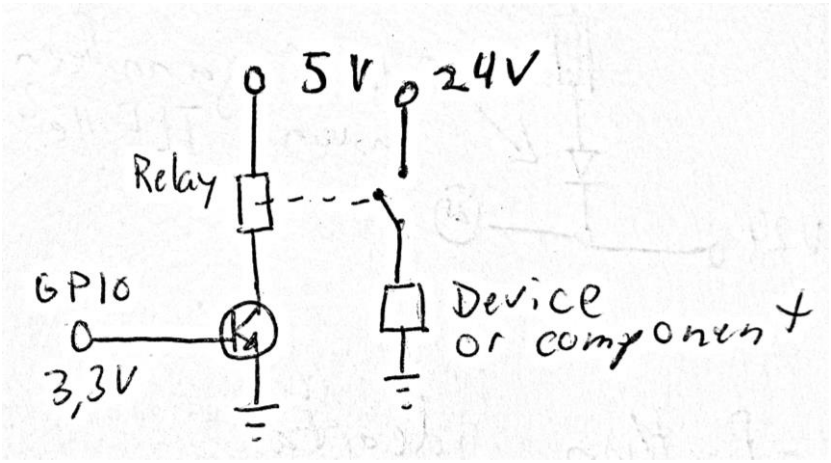

Figure 11. The transistor-relay solution controlling a 24 V device

Now that the relay could be controlled, only a 24 V power supply (PSU) was needed to give sufficient voltage to the valve. This breakthrough also meant, that any device or component could be controlled with the solution and it was an important step since almost every device on the system requires at least 10 V of supply voltage. Also, the transistor-relay solution was fairly inexpensive to create which suited the agenda of this project.

To make the creation of the first prototype possible, the missing important devices needed to be introduced to the system. These were the temperature sensor, the control button and the motor control unit. The control button was straight forward, as it did not require any other components to work. It got its voltage from one of the non-programmable 3,3 V GPIO pins and the ground side was directed to one of the programmable GPIO pins. If the button was pressed, the destination GPIO port's value changed from 0 to 1.

The motor control requires two signals: a 24 V start signal, which has to be activated to operate the motor and a motor speed signal, which can vary from 0 V to 10 V and where the inputted voltage signal corresponds the RPM (Rounds per minute) value of the motor. The start signal was simply handled with the transistor-relay solution but the speed signal needed a little bit extra thinking. To operate the cabin properly, it requires at least 3 different speeds for the motor and the voltage of the speed signal should not exceed the 10 V. The different speeds are a full speed with value of 10 V, a medium speed with value of 7,6 V and a low speed with value of 6,7 V.

Since the PSU supplied 24 V of voltage, a full voltage could not be inputted to the speed signal port. The supply voltage needed to be divided properly to achieve the 10 V maximum signal voltage. The solution was to use two resistors with values of 1,4 kΩ and 1 kΩ on a connected line to create a voltage divider. The 1,4 kΩ resistor was connected to the supply of the PSU and the 1 kΩ resistor to the ground. Then from the middle of the line, between the resistors, a line could be taken which supplied the 10 V of voltage. This 10 V was then directed to a group of connected terminal blocks, where one line was connected to the speed signal port (Figure 12). This connection worked as the full power signal.

Other two ports of the terminal block group were connected to two different relays. The first relay was connected with a line and 1 kΩ resistor and the output side of the relay was grounded. This same method was used on the second relay but with resistor of 2 kΩ. Both relays were controlled with the transistor-relay solution. Now, if the both of the relays were not activated, the motor got 10 V signal and full power. If the first relay were activated, the value of the speed signal dropped to the 6,7 V which was the low speed. Then if the first relay were deactivated and the second relay were activated, the speed signal got value of 7,6 V which was the medium speed (Figure 12). With this solution the motor could be controlled. Also, when the number of the relays got higher, a 5 V PSU was introduced to the system to supply the needed voltage to the components.

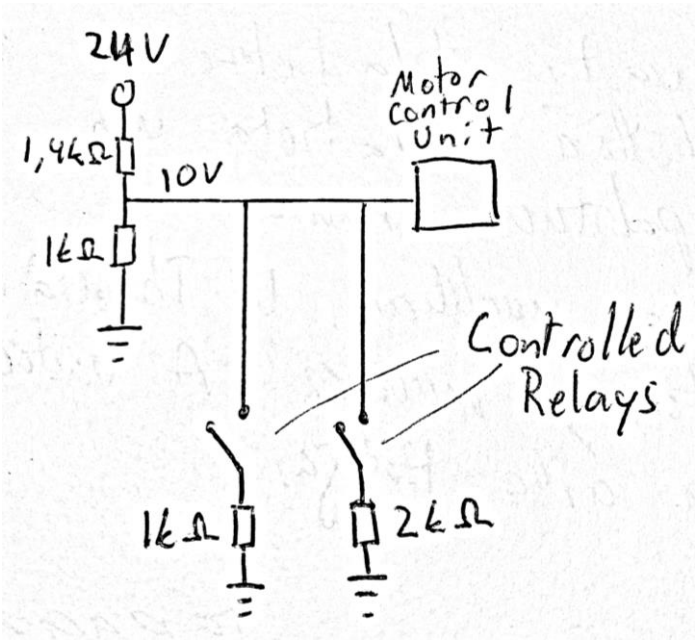

Figure 12. Motor control schematic

The last device was the temperature sensor. The problem with it was that it was an analogue temperature sensor and the GPIO pins of the Pi understand natively only digital input signal. An analogue to digital converter (ADC) was needed and luckily there was a USB ADC available. With this ADC, the temperature sensor could be connected to the system. The sensor got its voltage from 3,3 V GPIO pin and the output was directed to the input port of the ADC. Also, a 1 kΩ resistor was connected from the same input port to the ground, so that any noise voltage would be grounded and it would not cause inaccurate measurements of the temperature. (Figure 13)

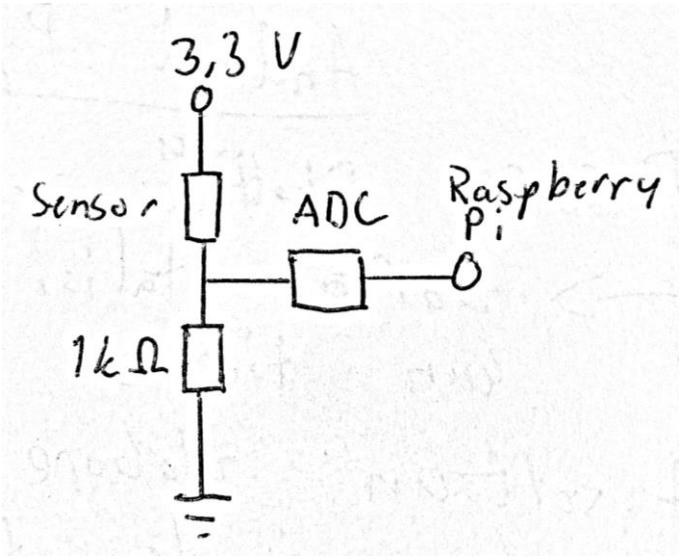

Figure 13. The temperature sensor circuit

6.2.2 Python

Before the creation of the software logic could begin, modules for GPIO pins and ADC needed to be installed and the newly installed modules needed to be tested. To control the GPIO pins, the GPIO module needed to be initialized. After that, a setup for the desired pin needed to be made. In this setup, the direction, input or output, of the pin was determined. When the setup was complete the pin was ready to be either controlled or listened depending on if the direction was output or input.

The software logic for the output signals works as follows: the pin has two states, true or false and the state determines if the pin works as a 3,3 V voltage supply or a ground. In this situation the controlled output devices were the transistors connected to the relays.

The temperature measurements were made with an ADC which meant the data gathered from the device were in the form of bits. The range of the bits varied from 0 to 1024. To convert these bits to corresponding temperature, a mathematical formula was needed and it was created by making a table, where the measured bit value was matched with a temperature, which was measured with a separate thermometer. From this table, a graph could be drawn and with that the formula could be determined.

6.3 Prototype I

The first prototype was installed on a cabin in the end of May, 2017. This prototype was different from the others as it had a Python software with graphical user interface (GUI). This meant it required a monitor to fully operate it. Although this was the first fully running prototype, no tests with a patient were ran with this setup.

6.3.1 Control system

A couple of devices needed to be introduced to the system to make it a real prototype: a led light and a boost fan, which was essentially a 24 V fan used in PC computers. The led's purpose was to be the indicator of the state of the system. The idea was to program it to switch on or off or blink when the systems state changes. The led was connected to a GPIO pin and a ground.

The boost fans logic in the system was as follows: always on, but with half-power and at certain times full-power. To control the boost fan, a 3-port terminal block and a transistor-relay solution was needed. 24 V of voltage was inputted to one of the ports with a 56 Ω power resistor, which dropped the voltage to 12 V and this was the half-power voltage. The second port of the terminal block was used to bypass this 12 V. It was connected to a relay with 24 V input and which was controlled by the Pi. If the relay were activated, the voltage of the fan, which was connected to the third port, went to 24 V which was the full-power. To make the prototype complete, a main power switch was added to easily power up the system. Figure 14 illustrates all the devices used in the first prototype.

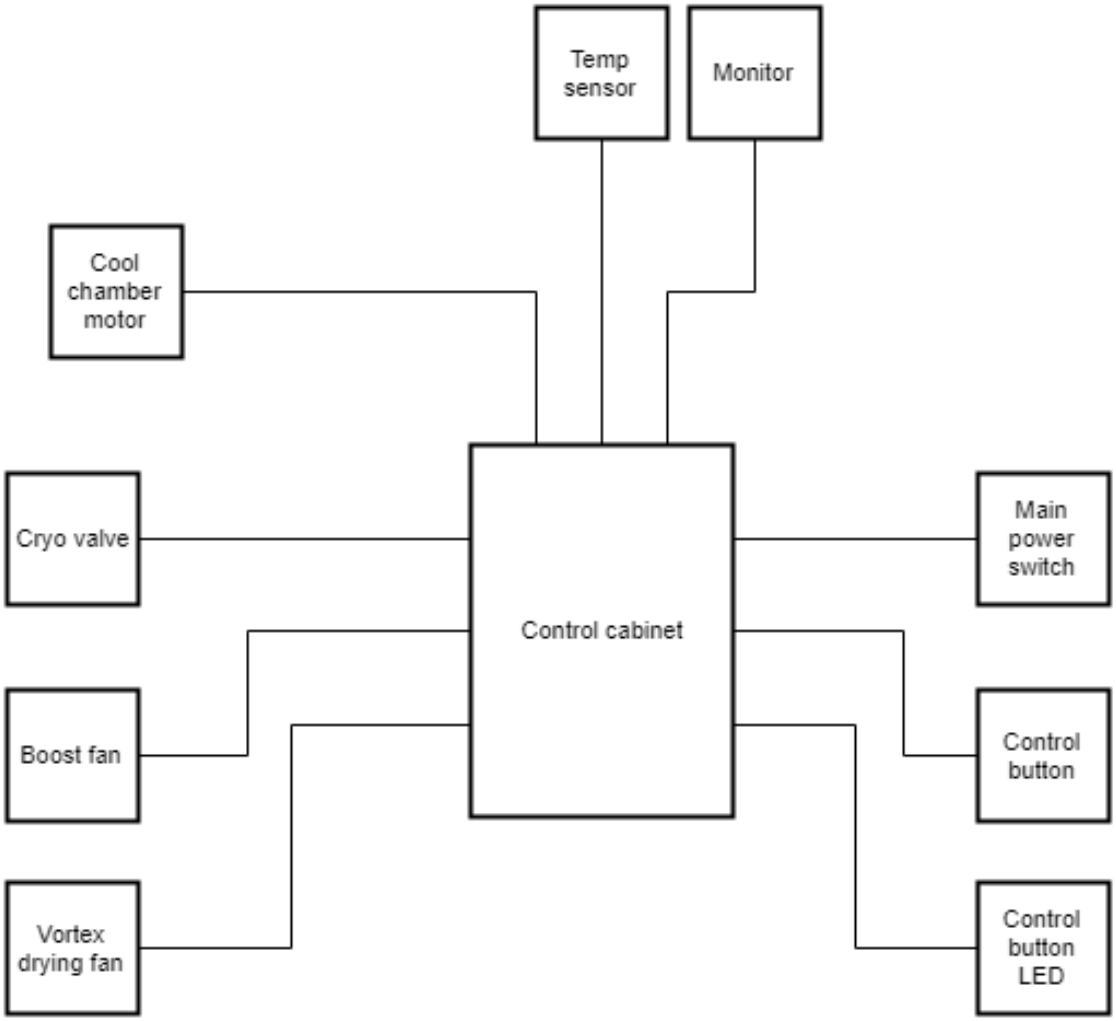

Figure 14. Cabling layout of the control system with connected devices

The control system was very messy at the time (Figure 15). The connections between the Raspberry Pi and the devices were made with a 40 pin IDE cable and terminal blocks. The other end of the cable was plugged to the pins of the GPIO while the connector of the other end was removed and the desired wires were separated and plugged to the terminal blocks.

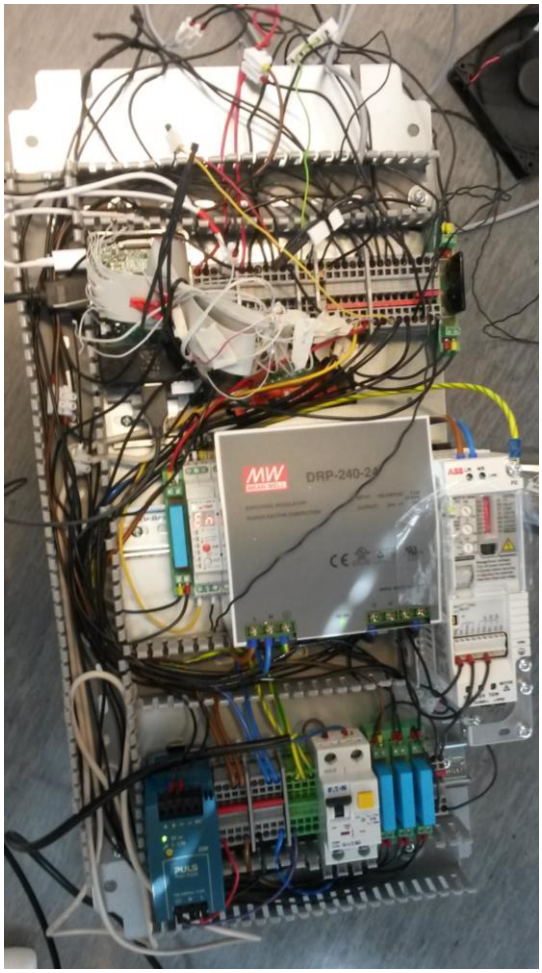

Figure 15. Control system's layout of the first prototype

At that time, there were some concerns about the reliability of the Pi. The question was: if the software crashes in the middle of the session, will it close the solenoid valve if it's in open state? This was a situation where the time and repetition could only give the answer for this reliability issue and since the development process was in the early stages, an alternative solution was needed. The solution was a time-relay, which could be programmed to only activate for a period of time on a given signal. To make things a little bit complicated, the signal strength for the activation needed to be 24 V so a transistor-relay solution was needed for this one as well. As shown in the figure 15, a considerable amount of terminal blocks was needed to build the first prototype. This combined with the 5 solid state relays and the time-relay, the price of the control system slowly started to rise.

6.3.2 Python

The idea of the logic running on the Pi is fairly simple: measure temperature and time and check the status of the control button. These three parameters were the only things affecting the status of the system. The status of the control button could be true or false and when the button was pressed this status changed. Also, if a time limit in a session was reached, the status of the button went from true to false. The logic is illustrated in the appendix 1.

There are two types of sessions: precooling sessions and main sessions for the patients. The differences between these two sessions from the aspect of the logic are very clear. Precool sessions can last significant amount of time, even over 10 minute sessions can occur when the ambient air is very hot or the coolant input systems pressure is getting low. The main session on the other hand will never exceed the desired treatment time which is usually 2-3 minutes.

The temperature also affects these sessions. Both of the sessions will do solenoid cycles, opening the solenoid for a period of time and then closing it, until the desired temperature is reached. If the precool session's target temperature is reached, the session finishes and the systems is ready for the patient to step in and start the main session. This is not the case in the main session, where the temperature only tells if the system needs to do more solenoid cycles.

Of course several error recognizing algorithms were implemented to the system in addition, for example if the temperature sensor broke or got disconnected from the system during a session, the session would stop and sessions could not be started before a proper temperature measurement could be made.

The Python GUI was made for the development purposes using Tkinter module which allowed creating dialogs for user interaction. The reason for this was to monitor the temperature measurements during the sessions and to allow I/O port activations when the system was idle, which meant that the manual activations of the I/O ports were disabled during sessions. For example the motor could be operated with the I/O signals to ensure the correct RPMs of the system (Figure 16).

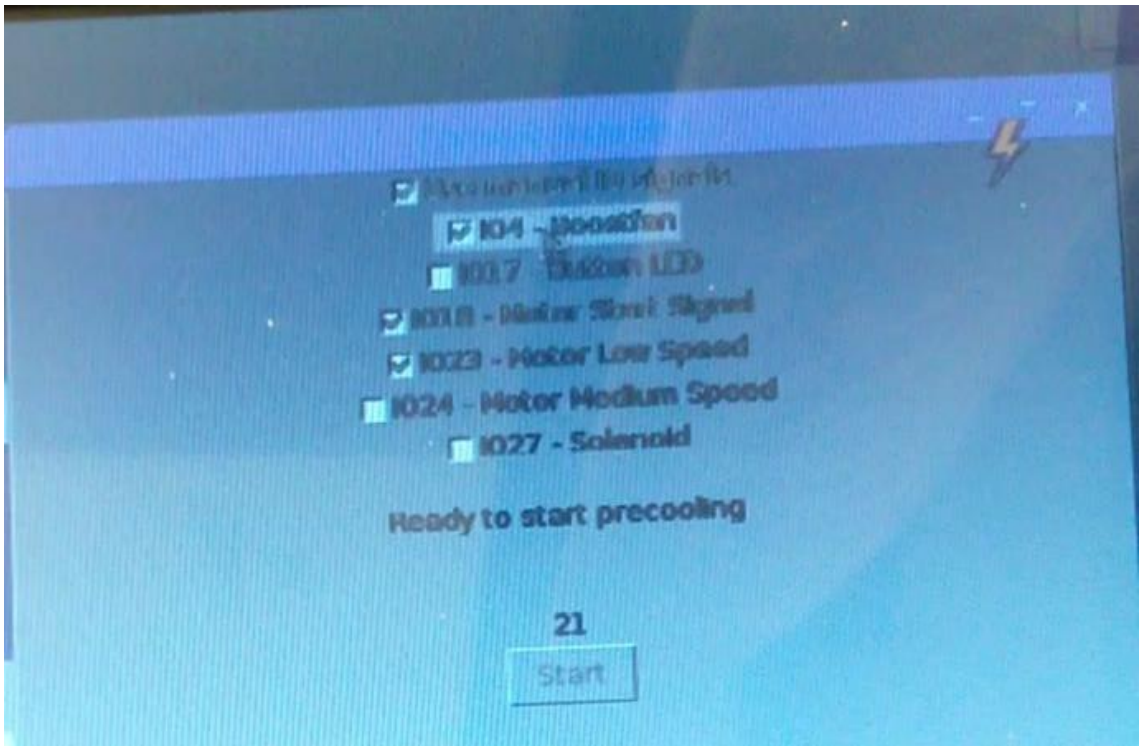

Figure 16. Python software with GUI

6.4 Prototype II

There were some major software changes made for the second prototype. The biggest of them was the introduction of the mobile software, which allowed the remote control and monitoring of the system via bluetooth connection. This prototype was the first to have tests with a patient. The second prototype was installed on a cabin in the mid of June, 2017. It did not have a big impact on the control system itself. The most notable change was that the monitor was removed from the system. This meant that the system was the first time completely automated, with the indicator led giving signals of the status of the system.

6.4.1 Raspbian

To control the system via bluetooth, the Raspbian system needed to be set up properly. A bluetooth module had to be installed and some important changes were needed to make before the system was running in a desired way. One of them was the use of cron. This method allows scheduling of tasks and it was used to start the Python software when the Raspbian booted. This way also the system-ready time got better, as the time

required from the power up to the situation where a session could be started, significantly decreased.

The other setting was the bluetooth setup. Because there were no monitor in the system to accept device pairing queries, an automation was needed. The system needed to be set up to accept all incoming pairing attempts and also advertise itself to the devices nearby so it could be found. Also, the system needed a name so it could be identified from the other devices nearby. Fortunately these settings were part of the system and there was no need for a custom script.

6.4.2 Python

The software of the second prototype faced significant changes. The introduction of the bluetooth remote control and monitoring system meant, that the data gathered from the system needed to be distributed to the Android device. Also, commands from this device must be listened and executed. The bluetooth connection has a few ways to be set up. In this system, the bluetooth connection was basically a traditional serial connection, where there is a port for transmission and a port for reception. This meant that the data moving between the devices was in the form of bytes, which could be interpreted as ASCII characters, essentially as plain text.

The hierarchy of the bluetooth connection needed to be determined before the connection could be made. The pair of devices, in this case the Raspberry and the Android device, had to have a server and a client. Since Raspberry collects all the data and controls the system, it was the natural choice to be the server of the system, while the Android device worked as a client.

The incoming data from the device were commands of different kind. The most important command was the start signal. It needed to work beside the control button. The solution was to create own listener to the start command from the device and to make it manipulate the same variable which the control button controls. This way the sessions could be started and stopped from both locations.

Now that the monitor was removed from the system, a new way of controlling manually the I/O ports needed to be made. The solution was a two-phase command listener. This was created to make sure the commands for the I/O ports were accurate and no errors

would occur. In the first phase, the received command for the I/O ports prepared the system for the I/O port control and the second phase listened which ports should be activated or deactivated.

The outgoing data included the temperature information, status of the system and the state of the session. The temperature data were sent to the Android device just for monitoring purposes, but the other two were critical for the synchronisation of the system.

6.4.3 Android

The idea of the Android application was that it would function as extension for the Python software. The system could operate without the Android application, but the use of the remote device would significantly improve the user experience. With this in mind, the software should be easily connectable and disconnectable without interruption, even during a session.

The application would have two states: a disconnected and a connected state. In the disconnected state, the user could search for new devices or connect to already known devices. The connected state would have two views: a main view for the session control and monitoring and I/O signal view for controlling the I/O ports. From the main view the user could start and stop sessions, monitor the temperature and elapsed time of sessions and set the session temperature and duration. The I/O signal view would list all the available I/O devices with an activation button next to them.

The development of the Android software begun from setting up the Android development software, Android studio. Usually when creating an app for Android, the Android studio offers real time app simulation on an emulator, where the app can be tested before installing it on an Android device. This is not the case if the app requires bluetooth connection. The emulator cannot simulate bluetooth connection, thus the app under development has to be tested directly on an Android device (Figure 17). To do this, the Android device has to be set on developer mode (Figure 17). This way the Android studio can update the app on the device fluently and the development process speeds up massively.

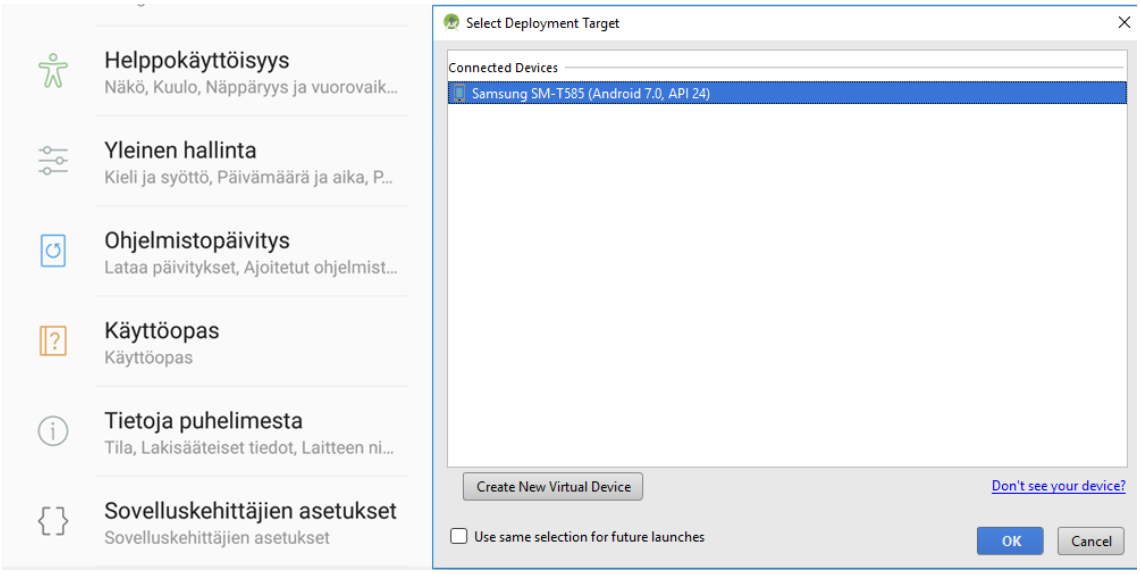

Figure 17. Developer mode activated in Android device which is indicated by the last option in the menu with the braces icon (left). Android studio recognizing external device (right)

The bluetooth connection has a few steps which has to be met before the connection can be established. The first one is to find the counterpart, which can done by starting a search. The Raspberry was set up in a way that it would always broadcast itself to nearby devices and this made it possible to the Android device to find it. When the correct device is found, a request to connect can be made. Usually this means that a paring between the two devices is made and a confirmation from both devices is needed to complete this. The Raspberry accepted all the incoming pairing attempts which made it easy to pair the devices up. Now that the devices were paired, the connection could be established. The device pairing also meant, that in the future no search was needed before the connection because the Android device saves the paired devices.

The disconnected state (Figure 18) was created with 4 elements: two buttons, text variable and a list. The first button was for the new devices which could be searched pushing this button. The second button was for the existing connections, the already paired devices. The list was a blank element before any action would be made. The found new devices or the already paired devices were listed on this list after the desired command. If the user clicked a listed device, the Android tried to create connection to it. The text variable was for the Bluetooth connection status. If the bluetooth was not activated from the Android device or the connection could not be established, the app notified the user of it.

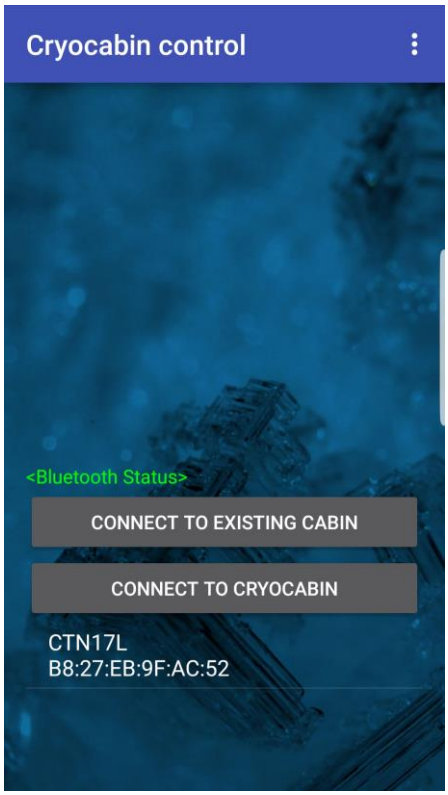

Figure 18. Disconnected state of the mobile app

The main view of the connected state had 6 elements: 4 different text variables, start/stop button and options button. The first text element had the information of the current state of the system, the second were only active if a session was in progress as it indicated the elapsed time of the session, the third indicated the current temperature of the system and the fourth indicated the connection status. The start/stop button sent a command to the Raspberry on click. Its status, start or stop, depended on the state of the system. The options button opened options menu, where the I/O signal view could be opened. The menu had also two other options: a view for setting session temperature and a view for setting session duration (Figure 19).

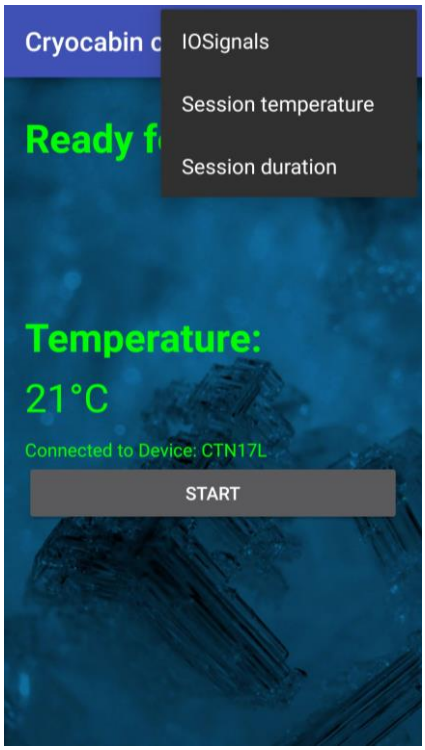

Figure 19. Connected state of the mobile app

The current cryocabin system has the duration and temperature setting on the main screen and this was the goal for the Android software as well. This could not be done on the first Android version as the element used for choosing the value was too large for the main view and there were no space for the two of them.

6.5 Prototype III

The biggest change made to the third prototype was the introduction of ADC microchip. This microchip would replace the USB ADC which was used for temperature measurements. Also, the solid state relays would be replaced with two relay modules which consisted 4 relays each. The third prototype was installed on a cabin in the beginning of August 2017.

6.5.1 Control system

The main reason for the ADC microchip introduction was clear: the price. The microchip's price was almost 80% lower than the price of the USB module. Of course the USB module had all the ports for component connections built-in, while the microchip needed a

circuit board where it could be installed (Figure 20), but all that taken into consideration, the microchip would still be more cost-efficient. Another reason for replacing the USB ADC was the low polling rate of the device, which caused some problems when measuring temperature continuously for long periods of time.

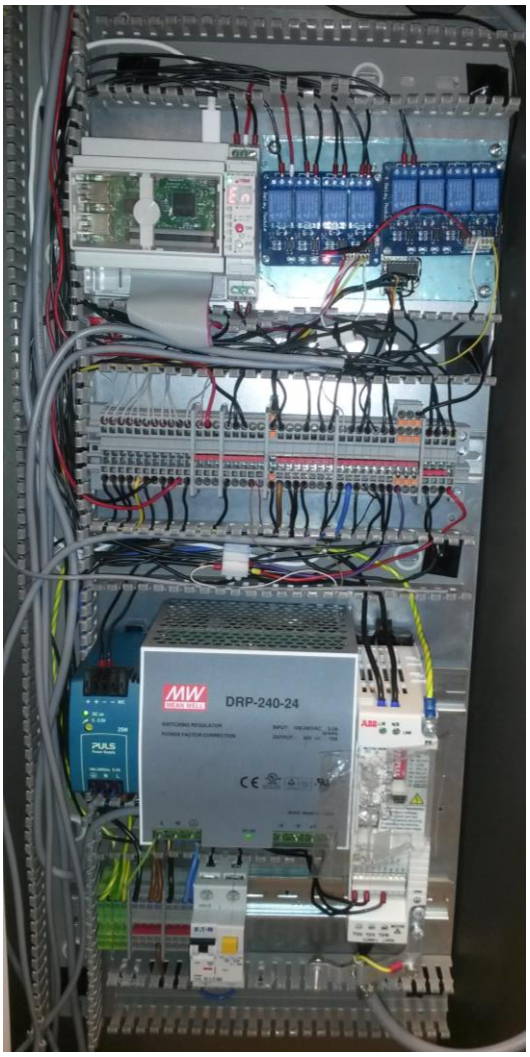

Figure 20. Layout of the third prototype

The price was also the reason for replacing the solid state relays with the relay modules. The relay module cost as much as one solid state relay, which made the decision very easy when the system required 5 solid state relays and the module had 4 relays on it. Also, the relay modules were compatible with the Raspberry GPIO which meant that the transistors were no longer required to activate the relays (Figure 20). The new relays were active low, which meant that the logic to control them was the opposite as when the transistor-relay solution was in place. To activate them, the control pin for the specific

relay in the module needed to be grounded, not input a 3,3 V signal, and this could be done through the GPIO pins as they can operate both ways.

6.5.2 Python

The changes made to the control system had some impacts on the logic software. The operating principle of the new relays meant that the whole logic for controlling the relay-controlled devices needed to be inverted: what earlier was true was now false. Although this change meant that some time consuming coding had to be implemented to make the new relays operate, the task was fairly straight forward.

Also the new ADC microchip required changes to the way the temperature data was acquired. While the USB ADC required 3-step data acquisition, a method where the data request command is first sent to the device, then the answer from the device is read and the value is saved to a variable which then can be modified for further use; the data from the new ADC could be gathered with just one function call. The new method also meant that the device was more responsive and the polling rate was higher. This meant that more temperature data could be gathered for more precise results. In addition, the data collected from the new ADC was in the same form as in the USB ADC which meant that the old formulas for calculating the temperature was still usable.

6.6 Prototype IV

The final prototype of the control system was installed on the cabin in the middle of September, 2017. It introduced a digital to analogue signal converter (DAC) microchip for stepless and more precise motor RPM control. This change required a circuit board where both ADC and DAC could fit with sufficient amount of GPIO and PSU connections. During this time the Android app also faced major changes when the UI was made more similar to the current Cryo°Cabin Windows software with adjustment circles for session duration and temperature.

6.6.1 Control system

The introduction of the DAC microchip was inevitable. A motor control system where only 3 possible settings were available, was bad for optimizing and testing the system. Also, if minor changes were needed for the motor control, the implementation of these changes were clumsy and time-consuming. With this component, the need for 5 different relays decreased to 3 which meant the system only required one 4-relay module to operate (Figure 21). The system also required less terminal blocks since most of the connections were made on the circuit board. Although the DAC was easily implemented to the system since it was compatible with the Raspberry Pi, it required an amplifier to reach the voltage levels the motor control unit required. The DAC outputted voltage signals from 0 to 3,3 V and the required levels were 0 to 10 V. To reach these levels the amplifier would need to boost the voltage levels of the incoming signal 3 times higher than the original signal.

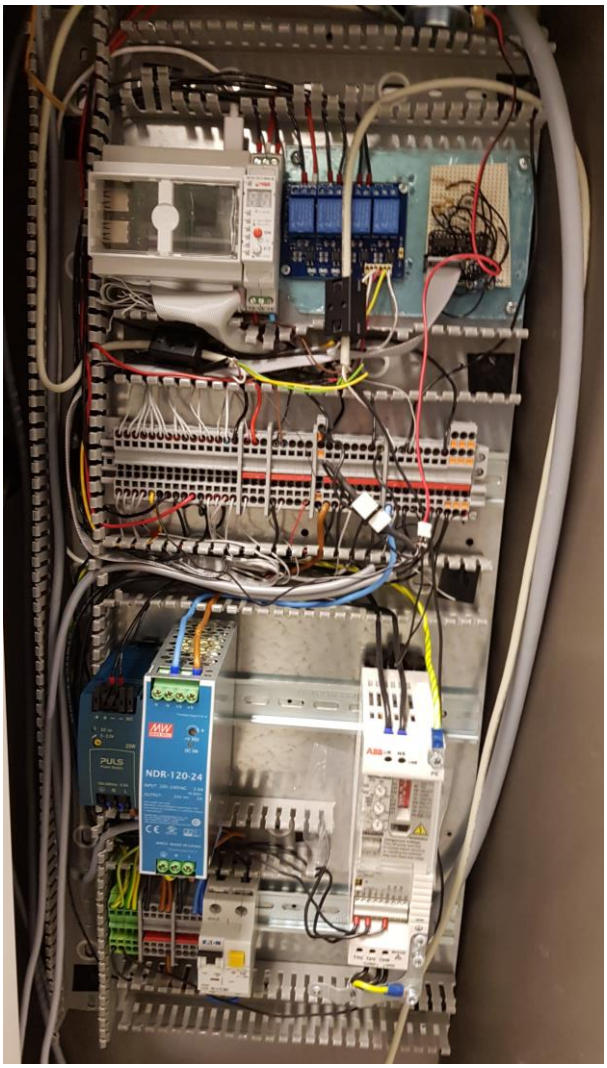

Figure 21. Layout of the fourth prototype

Now that there were 3 microchips: ADC, DAC and amplifier; a larger circuit board was required. The connectivity of the board should be good with some room for additional analogue components such as additional temperature sensors. The answer was a 16 pin IDE which offered the required amount of connections for easy installation. It could carry all the supply voltages, the GPIO connections, the analogue input signals and the motor control signal. The supply voltages of the circuit board were 3,3 V, 5 V, and 24 V. The reason for this were the microchips which required different levels of supply voltage (Appendix 2). The old 24 V PSU was also changed for smaller profile 24 V PSU (Figure 21).

6.6.2 Python

The Python software of the fourth prototype had to be adjusted to the changes made for the control system. The old logic where the motor RPM was controlled with the relays was now replaced with the DAC. The microchip itself required a software module to be controlled. The earlier method where the relay state defined the motor speed was Boolean type, true or false; after the change the motor was controlled with function calls where an integer type of parameter was given and this numerical value set the voltage inputted to the motor. The determination of the correct numerical values for the motor speed presets was very easy since the control voltage of the speed vary from 0 to 10 V which meant that any speed from that range is the percentage value of the range divided by 10. For example, the low speed preset is 6,7 V which means it was 67 % of the full speed.

The changes made to the Android software did not affect the Python software greatly but since the session temperature and duration selectors were changed a new operating logic was needed. Earlier, if the user selected the duration or the temperature option from the options menu, a bluetooth command were sent to the Python to listen and parse the next value for the new temperature or duration. User could cancel this by leaving the view and there were no time limit for this action. The new system also had this two-step system where a preparation command was sent for the Python, but since the preparation command stops the logic for the time period it listens the new value, a time-out was needed for this action. If the user did not set the new value within this time-out time the session parameter in question was returned to its earlier value and a warning message was sent to the Android device.

6.6.3 Android

In the previous prototype versions the Android app was run on a phone which meant that the size of the screen was tiny compared to the computer screen of the current Cryo°Cabin. This fact made it almost impossible to recreate the UI of the Windows software with all the same control buttons. To make the UI recreation possible, a tablet was purchased. Although the screen size of a tablet is still small compared to the computer monitor, it’s roughly three times larger than a phone screen size.

The control buttons in the Windows version of the Cryo°Cabin software are all circular shaped (Figure 22). This type of layout requires a lot of space because the radius of the buttons has to be large enough to contain the data inside the button. The positive side of the circular button is that it can fit a slider control its edge and this is used for example setting the session temperature. To recreate this in the Android app, the main view of the connected state required 3 circular buttons and a text element for the temperature data. Also two slider controls were needed for the temperature and duration control.

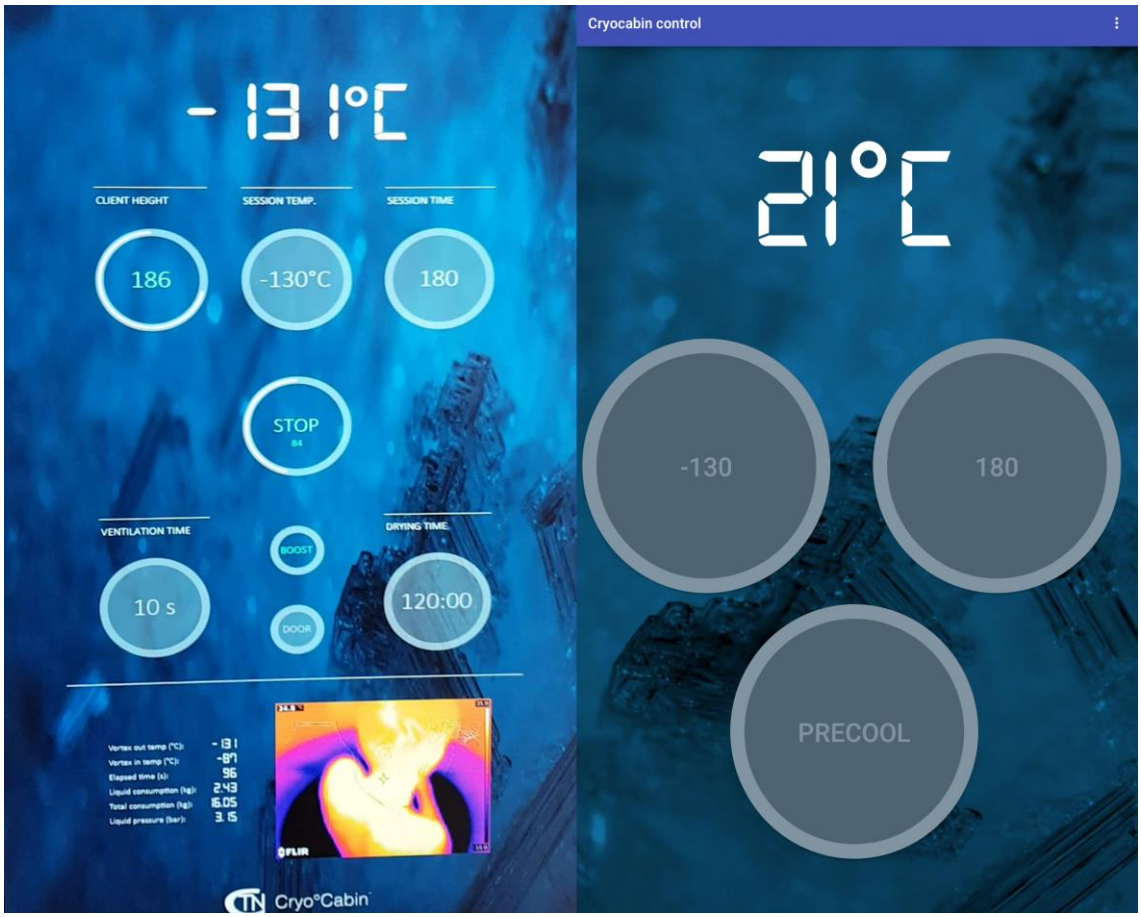

Figure 22. The Windows Cryo°Cabin software (left). The Android app of the fourth prototype running on tablet (right)

By default, the slider controls are hidden from the main view, but the user can activate them by clicking the desired session parameter button (Figure 22). This sends a preparation command for the Python which then listens the set value the user inputs with the slider control. If this action takes too long, the earlier value is restored, the slider control is hided and a warning message is displayed for the user. To avoid conflicts in the set values of the session parameters, two slider controls cannot be active at the same time. If one slider control is activated the other slider control cannot be activated before the

changes for the other are fully complete. If the user tries to activate the other slider control while on is already in use, a warning message is shown to the user.

The last change for the Android software was the cleanup of the I/O-view. The true-false boxes were replaced with a slider control to set the motor speed steplessly. Also 4 radio buttons were tied to the slider; with these radio buttons the user could choose preset values of 0, the low, the medium and the full speed. Clicking a radio button would move the slider to the position of the desired speed.

7 Results

The development process which was conducted in this thesis started at the end of March, 2017 and ended at the end of December, 2017. During this period of time major changes in the control system and in the software took place. The experimental nature of the first prototype turned into more industrial type of a prototype in the fourth version. The software written for this project improved with each iteration as more features were added for better user experience.

7.1 Cost reduction

To make a fair comparison between the costs of the previous model and the developed lower-cost model, the features and devices which were replaced and not removed are mainly taken into account. The monitor of the current system, the computer and the PLC were replaced with the Raspberry Pi, a circuit board, a relay module and a time-relay. The Android app is optional, but it greatly improves the user experience of the system. When comparing the costs of these components with the assumption that the future customer already has an Android device, the manufacturing cost of the new control system is 90 % lower than the current control system. If the price of the Android device included in the control system, the cost saving still reaches almost 80 %.

If the new simplified Cryo°Cabin is viewed as a complete unit without the existing cabin's non-crucial features, the price of the system would be significantly lower. This is due to fact that the most expensive components can be replaced with lower cost alternatives or removed completely. The savings made with these changes easily exceed the 66 % threshold of the cost-efficiency rating determined in the beginning of the project. When looking at these numbers, it is safe to say that this project has achieved the goal of developing a cost-effective alternative for the previous Cryo°Cabin. This gives the company a great opportunity to start manufacturing a lower cost model of the Cryo°Cabin.

7.2 Product reliability

The new system with its software and the mobile app were tested throughout the development process. The electrical part of the control system did not show any signs of malfunctioning during this time. Not a single system crash or equivalent major error were recorded during the time. The Python software also endured quite well as the most notable errors recorded usually were bugs after an introduction of new feature to the system. Most of the bugs were easy to fix. This same applied to the Android app which usually had some bugs after changes in the code.

These results were really promising although no real standardized tests were run on the system. To really compare the reliability of the two control systems, these standardized tests must be completed and feedback from users must be collected. This requires at least 6 to 12 months of testing time. Until this data is gathered no real reliability rating can be assessed for the system.

8 Summary

The goal of this project was to create a cost-efficient and reliable option for the current control system of the Cryo°Cabin. It is safe to say that this goal was reached and exceeded with a great margin. Although it is yet to be decided if the simplified Cryo°Cabin will ever be included in the product range of CTN, the control system itself appeared very promising. The same control system will also be used for other CTN projects and products in the future.

The circuit board created for this project has been further developed since it was first installed on the prototype IV and at the time of writing it is a full I/O extension card with multiple functions for Raspberry Pi named CRKKV-18; the name comes from the words: CTN Raspberry Kalle Koskinen Version 2018. Also, the development of an Apple version of the mobile app has been started.

Cryotherapy is still a quite unknown treatment amongst people. One reason for this is the fact that it is such a new method and the information about it has reached mostly the sports and wellbeing enthusiasts only. The second reason is the status of the treatment which can be viewed falsely as an “elitist treatment”. To really raise the awareness of people, cryotherapy should be affordable for any customer. This project aimed for a cost-efficient outcome to lower the costs of cryotherapy systems and hopefully in the future these cost-efficient systems will serve customers of all kinds.

References

Advantech MIC-7900-S5A1E Sulautettu teollisuustietokone. 2016. Web material. Elkome. <https://shop.elkome.com/fi/mic-7900-s5a1e-embedded-computer.html> Read 4.12.2017.

Atwell, Cabe. 2015. Top 10 single board computers from 9$ to 569$. Web material. <https://www.eetimes.com/document.asp?doc_id=1328008&page_number=1> Read 4.12.2017

Barr, Michael. 2017. Embedded systems glossary. Web material. <https://barrgroup.com/Embedded-Systems/Glossary-E> Read 4.12.2017.

Brown, Eric. 2016. World's smallest quad-core Linux SBC starts at $8. Web material. <http://linuxgizmos.com/worlds-smallest-quad-core-sbc-starts-at-8-dollars/> Read 4.12.2017.

Control system. 2017. Web material. Wikipedia. <https://en.wikipedia.org/wiki/Control_system> Read 4.12.2017.

Cryotech Nordic. 2018. Web material. Cryotech Nordic. <https://www.cryotechnordic.com/> Read 10.4.2018

Domin, Martin. 2015. Floyd Mayweather using £40,000 Cryosauna chilled to -115C at 4am to aid his recovery for Manny Pacquiao fight. Web material. Daily mail. <http://www.dailymail.co.uk/sport/article-3048710/Floyd-Mayweather-using-4-000-Cryosauna-chilled-115C-4am-aid-recovery-Manny-Pacquiao-fight.html>

Eklund, Jan. 2018. Research & Development Director, Cryotech Finland, Vantaa. Interview. 21.2.2018.

Embedded systems. 2017. Web material. Wikipedia. <https://en.wikipedia.org/wiki/Embedded_system> Read 4.12.2017

Fanton, Derek. 2014. What is an embedded computer? Web material. <https://www.logicsupply.com/explore/io-hub/embedded-computer/> Read 4.12.2017.

Flowchart based design. 2010. Web material. Engineer on a disk. <http://engineeronadisk.com/V2/book_PLC/engineeronadisk-77.html> Read 19.4.2018

Forrai, Alexandru. 2013. Embedded control system design. Springer.

Frenzel, Lou. 2013. The industrial control model. Web material. Electronic Design. <http://www.electronicdesign.com/industrial/industrial-control-model-0> Read 18.4.2018

John, Karl-Heinz & Tiegelkamp, Michael. 2010. IEC 61131-3: Programming industrial automation systems. Springer.

Kemp, Rob. 2016. Cryotherapy: the cutting edge technology that could be the secret to Leicester City's success. Web material. The Telegraph. <http://www.telegraph.co.uk/health-fitness/body/cryotherapy-the-cutting-edge-technology-that-could-be-the-secret/>

Lelinwalla, Mark. 2015. How Cryotherapy Works And Why Star Athletes Like Kobe Bryant And Floyd Mayweather Jr. Love It. Web material. Tech Times <http://www.techtimes.com/articles/61392/20150618/cryotherapy-works-why-star-athletes-love.htm>

Lombardi, Giovanni; Ziemann, Ewa & Banfi, Giuseppe. 2017. Whole-Body Cryotherapy in Athletes: From Therapy to Stimulation. An Updated Review of the Literature. Web material. NCBI. <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5411446/>

Marshall, Niko. 2016. How much does a cryotherapy machine cost? Web material. Quora. <https://www.quora.com/How-much-does-a-cryotherapy-machine-cost>

Muhic, Elvira. 2016. Whole-body cryotherapy. Web material. Physiopedia. <https://www.physio-pedia.com/Whole-body_cryotherapy>

Raspberry Pi 3 Model B. 2018. Web material. <https://www.raspberrypi.org/products/raspberry-pi-3-model-b/> Read 19.4.2018

Savic, Miroslav; Fonda, Borut & Sarabon, Nejc. 2013. Actual temperature during and thermal response after whole-body cryotherapy in cryo-cabin. Web material. ScienceDirect. Read 17.4.2018. <https://www.sciencedirect.com/science/article/pii/S0306456513000193>

Sequence. 2016. Web material. Zimmer. <http://whole-body-cryotherapy.com/ice-lab/sequenc/> Read 13.2.2018

Tomayko, James E. 1988. Computers in Spaceflight: The NASA Experience. Web material. NASA <https://history.nasa.gov/computers/Ch2-5.html> Read 4.12.2017

Uibx-230-bt-n2/2g-r11. 2016. Web material. Ipc2u. <https://ipc2u.com/catalog/uibx-230-bt-n2_2g-r11> Read 4.12.2017

Yliollitervo, Juha; Eklund, Jan & Martins, Jean-Patric. 2017. Fluid circulation system for a cryocabin arrangement and related cryocabin arrangement. Young & Thompson.

Control logic flowchart