



Norwegian University of  
Science and Technology

# Low Cost Human - Machine Interaction Platform for Bridging the Uncanny Valley, Using Humans as the Control System Input.

**Truls Hjertnes Nygaard**

Master of Science in Mechanical Engineering

Submission date: June 2017

Supervisor: Martin Steinert, MTP

Co-supervisor: Carlo Kriesi, MTP  
Arild Eikefjord, Lærdal Medical

Norwegian University of Science and Technology  
Department of Mechanical and Industrial Engineering





---

# Summary

This master's thesis describes the development of a conceptual prototype of a smooth actuation and control system of the eyes in a healthcare simulation mannequin with the aim to achieve perceived human like eye movement and behavior. The chapters of this thesis identifies, collects, judges and conceptualizes technologies to simulate the eye movement and behavior. The project identifies and develops a system that achieves realistic human-machine interactions, without a feeling of eeriness or unease, as is common with humanoid simulations.

Prototypes are shown that tackle the problems of human-like movement and behavior. The system prototype allows the operator to look through the eyes of the mannequin by displaying the mannequin's view to the operator in the head mounted device. The head mounted device processes and captures the eye movement of the operator and mimics this in mechanical actuation of the eyes of the mannequin. By letting the operator see the environment around the mannequin, the operator is able to interact with medical personnel through the mannequin.

Possibilities for automation of the behavior has been proposed, where the most promising is supervised learning in artificial neural networks. By using deep learning, it may be possible to train an algorithm to mimic human behavior and thus removing the need for operator control.

The system that has been proposed is an alpha prototype that has implemented critical functions for operation and proof of concept. Implementation is dependent on further work and optimization. The system has the potential to provide further dimensions of simulations by introducing nonverbal communication and diagnostic tools that use the eyes.

---

---

---

# Oppsummering

Denne masteroppgaven beskriver utviklingen av en konseptuell prototype av et system for realistisk aktivering og kontroll av øyne i avanserte førstehjelpsdukker. Målet for oppgaven er å kontrollere disse øynene på en måte som er realistisk både i form av bevegelse og oppførsel. De forskjellige kapitlene i denne oppgaven identifiserer, samler, bedømmer og konseptualiserer teknologier for å simulere øynenes bevegelser. Prosjektet identifiserer og utvikler et system som oppnår realistisk menneske-maskin-interaksjoner, samtidig som at det unngår følelser av ubehag eller avsky, som er vanlig ved interaksjon med humanoide roboter.

Prototype-systemet lar en operatør se gjennom øynene til førstehjelpsdukken ved hjelp av et headset. Headsettet viser operatøren det førstehjelpsdukken ser og fanger og prosesserer øyebevegelsen til operatøren mens de ser gjennom øynene til dukken. Mens operatøren observerer omgivelsene, vil øynene til dukken bevege seg slik øynene til operatøren beveger seg. Dette lar operatøren interagere med omgivelsene rundt dukken, på en måte som kan hjelpe diagnostisering og samhandling med medisinsk personell.

Det kan være muligheter for automatisering av styringen av øynene, basert på kunstig intelligens. Det er lagt fram forslag for hvordan det er mulig å etterligne menneskelig oppførsel basert på forskjellige input til systemet.

Systemet som er lagt fram er en alfa-prototype som har implementert kritiske funksjoner for operasjon og beviser dermed konseptet. Implementering avhenger av videre utvikling og optimalisering. Systemet har potensiale til å skape en ny dimensjon for medisinske simuleringer ved å introdusere ikke-verbal kommunikasjon og diagnostiske verktøy som avhenger av øynene.

---

---

---

# Preface

This thesis describes the development of a prototype system for actuation and control of eyes in health care simulators, as requested by Lærdal Medical. It was written to fulfill the requirements of the Product Development and Materials specialization at NTNU's Department of Engineering Design and Materials. I was engaged in this project between January and June 2017.

The task was created as a collaboration between my supervisor Martin Steinert, my supporting coach Carlo Kriesi, representative from Lærdal Medical, Arild Eikefjord and me. The project let me experience health care simulations and their complexity. With the help of Medisinsk SimulatorSenter at St. Olav's Hospital, I got valuable insights into the use of mannequins in health care simulations.

The project has introduced me to the intricacies of biomimetic behavior control, and the challenge has required me to obtain new knowledge of many subjects, such as computer vision and Python programming.

# Acknowledgements

I would like to thank Karen, for enduring with me all these years. I would also like to thank my parents, for helping me all the time, and Erik and Fredrik for being what brothers are supposed to be: funny, helpful and annoying.

Further on, I would like to thank Martin Steinert and Carlo Kriesi for excellent support and guidance during the project. The other Trolls at TrollLABS have also been very helpful, and the workshop session with Lærdal Medical gave me many new ideas for actuation and control of the eyes.

*Truls Hjertnes Nygaard*  
Trondheim 06.06.2017

---

# Table of Contents

<b>Summary</b>	<b>i</b>
<b>Oppsummering</b>	<b>i</b>
<b>Preface</b>	<b>iii</b>
<b>Acknowledgements</b>	<b>iii</b>
<b>Table of Contents</b>	<b>vii</b>
<b>List of Figures</b>	<b>x</b>
<b>1 Introduction</b>	<b>1</b>
<b>2 Background</b>	<b>5</b>
2.1 Project thesis . . . . .	5
2.2 Literature review . . . . .	6
2.2.1 Development methodology . . . . .	6
2.2.2 Diverging and converging . . . . .	7
2.2.3 Agile methods in enterprises . . . . .	8
2.3 Eyes and nonverbal communication . . . . .	9
2.3.1 Uncanny valley . . . . .	10
2.3.2 The eye's movements, and the tracking of these . . . . .	10
2.4 Technology review . . . . .	11
2.4.1 Single board computers . . . . .	11
2.4.2 Programming languages . . . . .	11
2.4.3 Microcontrollers . . . . .	11
2.4.4 Wireless communication . . . . .	12
2.4.5 Near eye displays . . . . .	12

---

<b>3</b>	<b>Prototype Development</b>	<b>15</b>
3.1	Head mounted device . . . . .	17
3.1.1	Prototype 0 . . . . .	17
3.1.2	Prototype 1 . . . . .	18
3.1.3	Prototype 2 . . . . .	19
3.1.4	Prototype 3 . . . . .	20
3.1.5	Prototype 4 . . . . .	21
3.1.6	Looking through the screen . . . . .	22
3.2	Eyes of the mannequin . . . . .	23
3.2.1	Magnet actuation . . . . .	23
3.2.2	Projected eye . . . . .	25
3.2.3	Eye controlled robot . . . . .	26
3.2.4	Camera mounted inside eyes . . . . .	28
3.2.5	Animatronic eyes with actuator rig . . . . .	29
<b>4</b>	<b>System details</b>	<b>31</b>
4.1	Complete system description . . . . .	33
4.2	Usage and operation . . . . .	35
4.2.1	Running the prototype as it is today . . . . .	37
4.3	Hardware . . . . .	37
4.3.1	Head mounted device . . . . .	37
4.3.2	Actuation . . . . .	42
4.4	Software . . . . .	46
4.4.1	Raspberry Pi operating system . . . . .	46
4.4.2	Python . . . . .	46
4.4.3	Arduino . . . . .	52
4.5	Testing and feedback . . . . .	54
<b>5</b>	<b>Discussion</b>	<b>55</b>
5.1	System review . . . . .	55
5.2	System future . . . . .	56
5.2.1	Artificial intelligence . . . . .	56
5.2.2	Ideas for future development . . . . .	57
5.3	Methodology review . . . . .	64
5.3.1	Development as one person opposed to in teams . . . . .	64
5.3.2	Trial and error or wayfaring? . . . . .	65
<b>6</b>	<b>Conclusion</b>	<b>67</b>
	<b>Bibliography</b>	<b>69</b>
<b>A</b>	<b>Code</b>	<b>73</b>
A.1	main.py . . . . .	74
A.2	DetectPupil.py . . . . .	76
A.3	calibrate.py . . . . .	78
A.4	stream.py . . . . .	79

---



---

A.5	Arduino . . . . .	80
A.5.1	Master . . . . .	80
A.5.2	Node . . . . .	82
<b>B</b>	<b>Project Thesis</b>	<b>83</b>

---

---

# List of Figures

1.1	Lærdal's mannequin opened up . . . . .	2
1.2	Lærdal's SimPad . . . . .	3
1.3	Prototypes from the Project Thesis in Appendix B . . . . .	4
2.1	Wayfaring . . . . .	13
3.1	Prototype 0 . . . . .	17
3.2	Prototype 1 . . . . .	18
3.3	Prototype 2 . . . . .	19
3.4	Prototype 3 . . . . .	20
3.5	Prototype 4 . . . . .	21
3.6	Seeing through an LCD . . . . .	22
3.7	The magnet rig. . . . .	24
3.8	The two eye projection prototypes. . . . .	25
3.9	Eye controlled robot . . . . .	27
3.10	Eye mounted camera . . . . .	28
3.11	Animatronic eyes . . . . .	30
4.1	Lærdal's eyes vs the eyes of this prototype . . . . .	31
4.2	The system in operation. . . . .	32
4.3	Functions of the system. . . . .	34
4.4	Smooth pursuit of a finger. . . . .	36
4.5	Oculus Rift . . . . .	39
4.6	Head mounted device's display . . . . .	40
4.7	Laser cut MDF casing . . . . .	41
4.8	Display connection . . . . .	42
4.9	Arduino with nRF24 board . . . . .	43
4.10	Wiring of the IR diodes . . . . .	43
4.11	WIFI camera with wide angle lens . . . . .	44
4.12	Arduino and power management . . . . .	45
4.13	Actuation control . . . . .	45

---

4.14	Looking in different directions, tracking the pupil. . . . .	48
4.15	Manipulation of images . . . . .	50
5.1	Approximation error . . . . .	59
5.2	The nine extreme positions of the actuated eye. . . . .	60
5.3	The elliptical path of the pupil . . . . .	61
5.4	Approximation error corrected . . . . .	61
5.5	Global and local coordinates . . . . .	62

# Introduction

This thesis describes the development of a prototype system for actuation and control of eyes in healthcare simulator mannequins and is based on the early stage development and prototyping of the product as described in Appendix B (Nygaard, 2016). The thesis is written in collaboration with Lærdal Medical (later referred to as Lærdal), who have requested the development of eyes that can be utilized in medical simulations.

Lærdal is an international company that develops advanced healthcare simulation mannequins intended to be used in training of medical personnel. The company has its origins in Stavanger, and started out as a toy manufacturer, but transitioned into production and development of mannequins with the purpose of advancing emergency care and resuscitation.

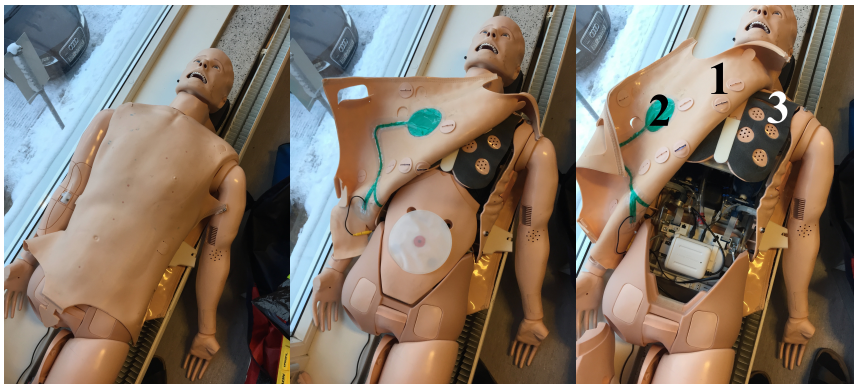
Today, Lærdal manufactures healthcare simulation mannequins (later referred to as just mannequins), which are advanced simulation machines that are able to exhibit complex medical scenarios. The mannequins are used in all kinds of simulation scenarios, both high and low stress. High stress simulations can be emergency care, cardiac arrests or similar, while low stress simulations can be situations where the patient is awake, but experiencing discomfort or symptoms that the simulation participants have to uncover.

Simulations are acted out scenarios that are intended to let medical personnel practice treating conditions that are rare or dangerous to practice on real patients. Some simulations are used to train students in interactions with patients before they are allowed to treat real patients. Other simulations are stress tests for emergency response teams, such as cardiac arrest response teams in hospitals. A simulation revolves around the mannequin, which is an advanced machine that is able to simulate many of the symptoms that a human could show. The mannequin has advanced communication systems, sensors and actuators, and it is able to track its interactions with the medical personnel. It can measure such things as if chest compressions are performed correctly, or if a defibrillation is done when it is actually needed. The mannequin can display mechanical responses, such as pulse or breathing, but also more advanced interactions can be found. The mannequin is able to simulate heart conditions that can be read by an ECG, or the medical personnel can listen to its breathing to discover symptoms affecting the lungs. In Figure 1.1, pictures of the

---

mannequin supplied by Lærdal is shown with the following features:

1. RFID chips embedded in the skin of the mannequin can be seen as small circular extrusions on the inside of the skin. These can be found with a tool that replicates ultrasound equipment and displays images of internal organs as if they are observed at that particular spot.
2. The green area is a patch for simulated defibrillation. To avoid destroying the mannequin, the medical personnel use a special defibrillator and the mannequin responds as if it is a real defibrillator.
3. The rib cage contains microphones that can simulate heartbeat and lung sounds when the medical personnel listen with a stethoscope. The rib cage can be compressed when performing chest compressions.

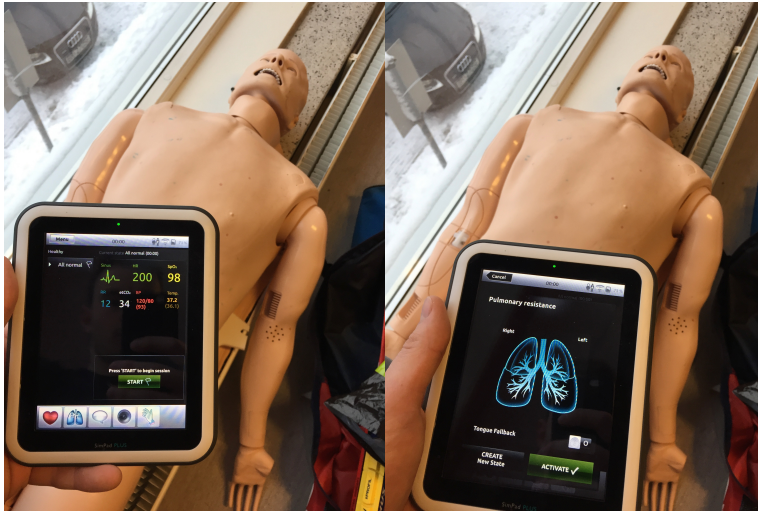


**Figure 1.1:** Lærdal's mannequin opened up

During simulations, an operator will control the situation and the events that occur. The operator is in control of each scenario and can change if needed. The control comes from the simulation software that is available. The simulation can be controlled from a SimPad touch device (Figure 1.2) or from a computer with the correct software.

As described in Appendix B (Nygaard, 2016), a lack of emotional connection between medical personnel and mannequin could be observed in some simulations. The medical personnel did not display an empathetic behavior towards the mannequin, as you would expect towards a human patient. Certain behaviors, such as eye contact when talking to the patient, were uncommon. This could lead to behaviors that would not be expected in a real life scenario. This created the basis for the prototypes developed in Appendix B (Nygaard, 2016). Figure 1.3 shows two of the prototypes from this process.

This master's thesis is sponsored by Lærdal Medical, and the scope is to develop, build, and refine a conceptual prototype of a smooth actuation and control system of the eyes in a healthcare simulation mannequin with the aim to achieve perceived human like eye movement, behavior and interaction. The control system could further be used for training



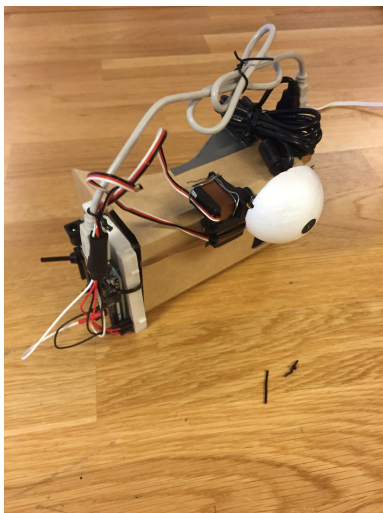
**Figure 1.2:** Lærdal's SimPad with two of the interfaces available

an autonomous control system based on artificial intelligence. This master's thesis is part of NTNU's mechanical engineering master's programme with a specialization in product development and materials, and gives 30 ECTS.

The thesis is written as a research project at TrollLABS at NTNU. TrollLABS is a makerspace (section 2.2.3) and research lab that tries to uncover, understand and leverage early stage engineering design paradigms and push the boundaries of Norwegian product development, creating radical new solutions.



(a) Confirming feasibility of pupil tracking



(b) Testing actuation based on pupil tracking

**Figure 1.3:** Prototypes from the Project Thesis in Appendix B



# Background

## 2.1 Project thesis

This master's thesis was preceded by a project thesis, also known as a premaster thesis, with the same theme (Appendix B (Nygaard, 2016)). The project thesis was part of the mechanical engineering master's programme at NTNU, giving 15 ECTS, equivalent to half of the ninth semester's workload. The goal of the project thesis was to investigate technologies related to the eye's of patient simulator mannequins. This investigation consisted of the following tasks:

- Generating concepts
- Building prototypes
- Building test setups
- Testing and comparing alternatives
- Judging concepts

This was performed over 4 months in 2016, and the conclusions drawn from that work was the basis for this master thesis. The most important conclusion of the project thesis was that in healthcare simulations where a patient simulator mannequin is included, a lack of empathy was observed. Medical personnel were observed to not interact with the mannequin in the same way that is expected when the patient is human. Most significantly, the medical personnel did not look directly at the mannequin when talking to it, and the interactions lacked empathy. Simulation participants reported that there is no observed difference in a mannequin that is awake and one that is dead. Some responses, such as pulse, breath or vocalization where the mannequin is completely still are not easily relatable and in a stressful situation they may be hard to recognize. One incident was reported where medical personnel started CPR on a mannequin that was talking to them.

---

The project thesis explored how the eyes of the mannequin could improve the interactions between medical personnel and the mannequin and how the eyes could be actuated in a way that supports this interaction. The thesis concluded with a need and technological possibility for real time human control of the eyes of the mannequin.

## **2.2 Literature review**

This section reviews the current standing of international academic research in the subjects that this thesis investigates as well as the scientific basis for the development methodology that this thesis utilizes for prototype development.

### **2.2.1 Development methodology**

Smith and Reinertsen (1992) coined the phrase "fuzzy front end" of product development to describe how the early stages of prototyping are unclear, uncertain or "fuzzy". They explained that the early stages of the development offer the best opportunities for large, cheap changes to the product development scope.

The importance of investing in the fuzzy front end is further explained by Leifer and Steinert (2011), who argues that later stage prototypes cost an order of magnitude more in resources, both in time and money, than early prototypes. This difference is due to prototyping materials and environment, and how much is invested in each prototype. Early stage prototyping allow for experimenting with specific ideas or system interactions in cheap materials. Testing simple interactions or critical functions do not need to be created in high resolution in order to get feedback on the idea.

Edelman et al. (2009) explain that low resolution models afford paradigmic shifts, while high resolution models afford parametric shifts. This is why the front end of the development cycle should be inhabited by low resolution prototyping. Paradigmatic shifts come easier when when prototypes are created in cheap materials with little time investment. It is harder to discard a milled aluminium part than a clay model, even though they represent the same shape, only with different resolution.

As explained by Leifer and Steinert (2011), prototypes are built to test a specific idea and or system interaction. This means that in essence, a prototype is an experiment that tries to uncover knowledge. As described by Smith (2007), an experiment allows product developers to investigate new technology and testing its limits without committing to how, or whether, they will use it. This means that prototypes can enable product developers to probe the feasibility of an idea, concept, interaction, simulation or similar to gain knowledge while reducing risk.

When partaking in radical product development, the goal is often unclear. Unknown unknowns (Sutcliffe and Sawyer, 2013) will be uncovered at some point and change the requirements and thus the course of the development. To avoid such paradigmic shifts late in the process, gradually refining prototypes from low resolution to high resolution while testing continuously, will decrease the investment before a potential unknown unknown reveals the need for redesigning. Being aware that the prototype you are working on may have to be thrown away is the basis of the wayfinding that design engineers do, as described by Edelman et al. (2009). Steinert, Martin and Leifer, Larry J. (2012) described

---

the Hunter-Gatherer Model to find one's way when designing products, as opposed to a linear, planning based development process. This model was further refined into the Wayfaring Model (Gerstenberg et al., 2015), which explains that probing ideas is how the unexpected discoveries of unknown unknowns are handled. By prototyping rapidly and being ready to change often, a wider base of knowledge can be captured. This knowledge will support the product into its final form by letting the product developers have wider knowledge of the technical and sociological aspects of the product development challenge. Each probed idea can be explained as an iteration where knowledge is created and tested, and each probe is a tool to discover critical functionalities and subsequent requirements (Kriesi et al., 2016).

The Wayfaring Model (Figure 2.1) describes one such iteration as a probe into a concept, simulation, prototype, idea or similar. A probe creates knowledge in design-build-test cycles, where each cycle consists in generating concepts and converging some of these concepts into a testable prototype. To gain knowledge about the complex system that a prototype often is, the only viable method of testing it, is to build the system (Baldwin and Clark, 2000, p. 273). As knowledge grows about the prototype, it can be refined and remade into higher resolution, culminating in an alpha and beta prototypes.

Alpha prototypes are the first functional prototypes that contain all the critical functions, while not being ready for commercial testing. An alpha prototype may provide a proof of concept while neglecting features that are out of the scope of the prototype. Beta prototypes are ready to be tested on users.

## 2.2.2 Diverging and converging

Generating concepts is a process that can be structured, as discussed by Friesike and Gassmann (2014), who shows different techniques for leveraging creativity in an innovation setting. In a product development process, the generation of new ideas and then choosing the right idea for next development iteration, is important.

There is a difference between generating concepts and converging them into a testable prototype. Eris (2003, 2004) describes how questions contribute to the design process by separating them into two categories. Generative Design Questions (GDQs) are questions that are open to unknown answers where the questions diverge away from the facts, to the possibilities that can be generated. Such questions enable divergent thinking by opening up the solutions space; "How many ways can we ... ?", "Why not ... ?", "Is there another way to ...?".

On the other side are Deep Reasoning Questions (DRQs), which are converging questions that reduce ambiguity, in order to make design decisions. DRQs are seeking a rational and truthful explanation to the question, to reduce the number of proposed design concepts. The questions focus on deliverables and reiterating the goals of the design process. "Why is concept C1 better than concept C2 in terms of ...?", "How long does it take to produce concept C3?".

This thesis has been utilizing principles from the Wayfaring Model for generation of concepts and development of prototypes. While the early stage prototyping, as described in Nygaard (2016), was done with low resolution prototypes and a large amount paradigmatic shifts, the development that this thesis considers has evolved into higher resolution prototypes, culminating in an alpha prototype of the system.

---

The prototypes described in this thesis and in Nygaard (2016) have been developed by the knowledge that has been gathered from following users and probing into needs and technological possibilities that have been found. Leifer and Steinert (2011) discusses the importance of the user and their needs in a product development process. Finding the actual users of the product, their needs and what requirements these needs imply, are important for concept generation, but even more important for converging the concepts into a prototype. The need finding of this thesis was performed mainly in the preceding project thesis (Appendix B (Nygaard, 2016)).

### **2.2.3 Agile methods in enterprises**

The word agile means to be able to move easily and change quickly, and was used by software developers in the Agile Manifesto (Cohen et al., 2003) to explain how traditional, linear development processes are creating more work, more cost and less effectiveness than the agile methods that they proposed. Since then, agile methods have been adopted into product development, and the methods for product development explain that embracing the uncertainty of the product development process is a major point in reducing the cost of rework due to immature decisions (Smith, 2007; Leifer and Steinert, 2011). While the classical way of handling uncertainty has been to improve forecasting of product requirements, Thomke and Reinertsen (1998) show how a flexible management strategy that tries to increase product development agility can outperform projects that use inflexible, forecast-based methods. Wayfaring is one way of staying flexible in the product development process, as agility in development is based on being ready to change and to act rapidly, which are core values of the Wayfaring Model.

As explained by Smith (2007), the flexibility of set-based design principles emphasizes exploring and keeping the design space open, so as not to lock into an option prematurely. The idea of flexibility in the design process calls for making the decision in the last possible moment. If there is no need to choose between two concepts, delay the choice and develop both. If resources demand that the choice be made, choose the best one, based on the current

Fixing requirements early may cause critical aspects to be discovered too late in the process, as accurate information is not yet available. This can lead to high adaption costs to overcome the aspect when it is discovered. Additionally, as product complexity increases, the difficulty of forecasting requirements increases exponentially.

So why do managers choose to operate in a requirement driven product development process? This is mainly thought to be about deliverables and measurability. It is easier to report achievements, deviations and time usage when these can be quantified. When a requirement is fixed, the achievement of this requirement can be easily documented over time, as it will either be fulfilled, partially fulfilled or not fulfilled. Contrarily, the agile product development process is more difficult to quantify, as goals, design paradigms and knowledge about the challenge will change over time. As described by Heck et al. (2016), the quantification of fuzzy front end projects is possible, although the literature that supports this is sparse.

The uncertainty of fuzzy front end engineering makes it difficult to take the step away from regular, easily quantifiable requirement driven product development. As described by Kriesi et al. (2016), it might be beneficial to use the agility of the Wayfaring Model to

---

overcome the uncertainty, and use the generated knowledge that comes from prototyping iteratively, to establish requirements.

### **People factors in product development**

Many companies try to manage early stage product development projects like other projects. This is not optimal, as product development in the fuzzy front end stage is different from other processes where instruments like milestones, stage-gate elements and other process control systems may be needed. While process leadership is important here as well, the key capability in the fuzzy front end is to manage people. Smith (2007) shows that people factors increase labor costs for early stage product development projects ten times more than any other factor.

In the early stages of a product development project, the key capability for a good team is to identify the creative potential of the individuals and give them meaningful work. Teams that are motivated by challenging and interesting tasks are much more successful than those who work for status or money alone (Gassmann and Schweitzer, 2014).

In general, teams benefit from being built up by both generalists and specialists, while early stage product development projects may benefit most by generalists, as they may be more capable of changing design paradigms. While specialists have deep knowledge about one or a few topics, generalists have a general knowledge about many topics. This makes specialists harder to utilize in product development tasks with a broad scope, such as the fuzzy front end has. Still, some people can be categorized both as generalists and specialists which is often described as "T-shaped" people. They have broad knowledge on top and deep knowledge in one or a few topics. T-shaped people are optimal for early stage development teams, as they can contribute in most fields, and may provide insights about their specialist topics that other team-members might not have (Smith, 2007).

### **The importance of makerspaces**

As mentioned in the introduction, TrollLABS is a makerspace and a research lab. Academic makerspaces are education environments for students, with the goal of promoting collaboration, design and manufacturing of prototypes (Wilczynski, 2015). Successful makerspaces promote prototyping and creative confidence by providing access to machines, tools and space, but most importantly through collaboration and communication with other users of the space. By having available educators, design and manufacturing professionals, users can lean on others to gain confidence and ability and thereby prototype more quickly and accurately.

## **2.3 Eyes and nonverbal communication**

In human-human communication, a lot of information is transmitted through nonverbal communication, and the eye region draws significantly more attention compared to any other area of the face and there appears to be dedicated areas of the brain specialized for processing eye information (Hall and Knapp, 2013). Gaze behavior is moderated by

---

culture, age, gender, genetics and more. Still, eye contact and mutual gaze between patient and medical personnel has been found to impact patient-centeredness and medical personnel's awareness about the patient's psychological distress and cognitive functioning (D'Agostino and Bylund, 2014).

### **2.3.1 Uncanny valley**

As human likeness in humanoid robots increases, there will be an increase in how familiar the robot feels, until a certain point called the uncanny valley (Ishiguro, 2007). The uncanny valley is the point where the robot looks and behaves very much like a human, but not quite, which makes it feel strange and disconcerting. This is true for all aspects of the humanoid, including the eyes. The minute details and the vaguely strange behaviors that separate the humanoid from humans is what triggers a feeling of unease with the observer. And the eyes are no exceptions, rather the opposite, as they gather a major part of the attention when interacting with humans or humanoids.

### **2.3.2 The eye's movements, and the tracking of these**

Duchowski (2007) describes the theory and practices around eye tracking methodology. Eyes have three major movement patterns, saccades (rapid shifts in gaze direction), vergence (shifting focal point depth) and smooth pursuit. Identification of these movements can be assumed to provide information of gaze and visual attention.

There are four categories for eye movement measurement, which can be broadly defined as:

- Electro-Oculography - measuring the electropotential differences of the skin around the eyes.
- Scleral contact lenses - contact lenses that mechanically or optically provide location measurements.
- Video-based pupil/corneal reflection - Using infrared light, the reflection of the cornea is measured relative to the center of the pupil.
- Video-Oculography - Video-based detection of the pupil and its position. This is the technique utilized in this thesis.

Duchowski (2007) goes in details to explain how it is possible transform the coordinates of the detected pupil into 3D gaze coordinates or projected onto a 2D plane. The coordinates of the detected pupil can be translated to parametric ray representation of the gaze direction. The book shows several techniques for optimizing tracking of gaze in both 3D and 2D. As explained in section 5.2, this project does not require gaze tracking, as the mechanical actuation can be extrapolated from observed pupil coordinates, either directly or by transforming them. The transformations described in section 5.2 could improve accuracy by accounting for the curvature of the eye ball.

---

## 2.4 Technology review

This section reviews what technologies are available to solve the needs that have appeared during the product development process and explains why the specific technologies have been chosen. Price and availability have been major decision points for most of the purchases. A driving factor in product development is time, and delivery times are important for prototype iterations. This means that some products may solve the challenges to a lesser degree than other products, but have been chosen for their availability.

### 2.4.1 Single board computers

Single board computers (SBCs) are small computers that allow the user to perform tasks that require processing or internet connection on a small form factor and with an operating system. This makes them popular in prototyping Internet Of Things (IOT) applications. The most commonly used SBC for low end prototyping is the Raspberry Pi, which offers a low cost platform for development, with access to a large and active internet community. There are other SBCs that have more processing power, more storage, better GPUs and so on, but there are none that are as accessible in terms of community. There are thousands of projects shared online that use a Raspberry Pi as the core computing unit, which makes it easier to learn for people with less knowledge about computers. The form factor and price of SBCs make them suitable for integration into prototypes and for small scale development projects.

### 2.4.2 Programming languages

In this project, Python has been used as the main programming language. Python is a high level programming language that was developed with simplicity in mind (Rossum, 1995). The language uses indentation to separate blocks, as opposed to braces or keywords, to improve readability. Python uses a syntax that allows users to solve tasks in fewer lines of code than for example C++ or Java.

Python is an interpreted language with interpreters for many operating systems. The operating system Raspbian Jessie, that runs on the Raspberry Pi, has preinstalled interpreters for both Python 2.7 and Python 3. This availability in combination with the simplicity of the syntax, made Python easier to learn and implement as opposed to for example C++, even though C++ is more efficient.

### 2.4.3 Microcontrollers

Arduino is a company that produces open source single board microcontrollers with software. The Arduino Uno board provides digital and analog input/output (I/O) interfaces and comes with an integrated development environment (IDE) for programming. The Arduino single board microcontrollers are made to allow easy prototyping electrical connections, and to make microcontrollers more accessible for those who have limited electronics experience.

---

As with the Raspberry Pi, Arduino has a large and active online community with many open source projects. This makes the technology easily available and prototyping becomes easier.

#### **2.4.4 Wireless communication**

Some alternatives were considered for communication between the devices described in section 4.3, although the solutions that were chosen were available in TrollLABS during prototyping, and were chosen as a result. The camera that is used is a WIFI camera that streams the video to a local web page. This camera was not in use in the lab, and sufficiently solved the challenge of wireless video streaming. Other options for transmission could be radio, as is commonly used for drones and other remotely operated vehicles.

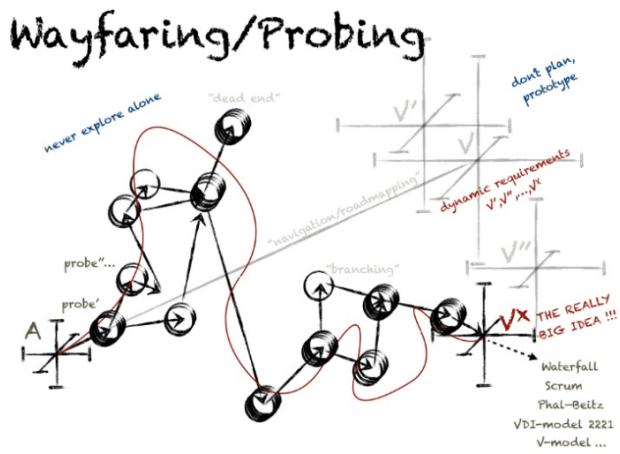
The wireless communication from the Arduino to the actuators was investigated more thoroughly. Arduino has some breakout boards for wireless communication, such as Bluetooth or WIFI, but they are not very easy to set up. In parallel to this project, one group associated to TrollLABS was developing an Arduino library for simple radio frequency communication between Arduino microcontrollers. This library is described in more detail in section 4.4.3 and was chosen for its ease of implementation and available hardware.

#### **2.4.5 Near eye displays**

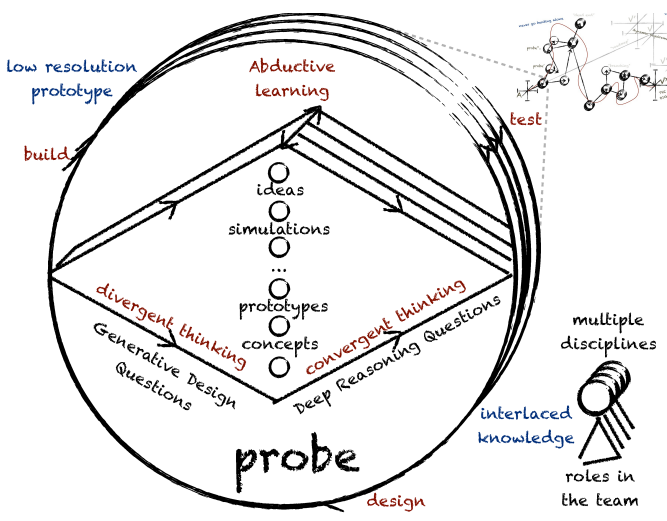
In virtual reality headsets, near eye displays (NEDs) are used along with lenses to display images on a screen that is usually between 3-10 cm away from the eyes. To create a comfortable viewing experience, these displays need to be high resolution and have a high refresh rate. In this project, these factors were considered less important, as the price and availability of small, high resolution screens with a high refresh rate is quite poor. In addition, the Raspberry Pi lack the capability to process video streams with such a high resolution and frame rate.

Display interfaces are not always compatible with the Raspberry Pi. This is true both for the MIPI display serial interface and the HDMI that the SBC has. The display that is used in this project, as detailed in section 4.3.1, was chosen for its short delivery time and low price. The display also has a fairly high resolution and refresh rate, although it is noticeable that it is not optimal. It is possible to see individual pixels while wearing the head mounted device.





(a) Wayfaring



(b) Probe

Figure 2.1: Wayfaring consists of probing ideas to uncover unknown unknowns.

---

---

## Prototype Development

The development of the product described in chapter 4 is based on the prototypes developed in the project thesis (Nygaard, 2016). This chapter describes the prototyping directions that were explored as a result of the findings in the project thesis. The prototypes were first developed in a low resolution before being refined and redesigned to perform the desired functions. This chapter is structured in such a way as show major prototyping milestones, the learnings and conclusions of these milestones, and how the work continued based on this.

The prototypes developed here were developed by way of probing into certain features or functions while trying not to distort the understanding of the whole system. This means to separate aspects that are unimportant from the development probe, and to filter the qualities that are explored (Lim et al., 2008). In effect, this means to make the system architecture modular, where each part, for example the control system for the actuator rig, is made independent and individually modifiable. Being able to transform aspects of the system without having to redesign the whole system is somewhat inspired by set based concurrent engineering of Toyota (Sobek et al., 1999) as described by Smith (2007). This philosophy is based on delaying design decisions to avoid cross-discipline contradictions in a late stage.

Modularity in the design process allows the developer to work with aspects of the system individually, changing and testing these aspects as knowledge grows. The probing that is described in section 2.2.1 can be performed on each module as long as the interfaces between each module allow for changes. In this thesis, the different modules can be broadly defined as:

- Processing
- Displaying
- Capturing eye movement
- Actuation control

---

- Actuation

These modules require interfaces that allow for modular changes. The interface between processing and actuation control was standardized early to be a linear control output represented by byte values. The interface between actuation control and actuation was similarly standardized. This allowed for change in actuators, communication protocols or other inter-modular aspects, and the electromagnet actuator (section 3.2.1), the eye-controlled robot (section 3.2.3), the camera inside the eye (section 3.2.4) and the actuator rig (section 3.2.5) were all controlled with the same interface and only minor changes in code.

The modular approach was also implemented in the code. The control signal output was easier to simulate with a web-camera and a facial recognition algorithm and thus, a conversion function that allowed for different input signals to generate output structured in the same way, was created. This meant that the generation of actuation output could be changed, but the output signal formatting stayed the same. The actuation could be controlled with potentiometers (section 3.2.1), with face recognition or with eye control.

The modularity described here can be compared to the modularity described in Baldwin and Clark (2000), which is describing modular development of computers and software. This book discusses many aspects about modularity, and some of these can be used to explain design methods in the early phase of complex systems. As explained in the text, the essential aspect of modularity lies in the fact that subsets of the design are broken out at the beginning of the design process. Tasks within different modules can then proceed independently, and the resulting designs can be combined in a variety of ways by standardizing the interfaces between each module.

Highly modular systems may result in higher degrees of complexity. Given this complexity, it simply is not possible for designers to know enough about the system to eliminate all uncertainty. Thus each new design is fundamentally an experiment. Its outcome may be guessed, but it cannot be known ahead of time. A different way to attack the problem is to design tests so that the best module can be selected at its own level. To evaluate a module without embedding it in a prototype system requires detailed knowledge about what the module contributes to the whole, as well as how different modules interact. In particular, dysfunctional interactions (like one module transferring heat to its neighbor, or a subroutine failing to return to the calling program) must be understood, so they can be avoided (Baldwin and Clark, 2000).

This thesis has combined the module probes into major milestone prototypes, although they were not developed as one prototype, but rather in module probes. This is done primarily to apply structure to the presentation of the development process and to be able to clearly define the evolution of the prototypes.

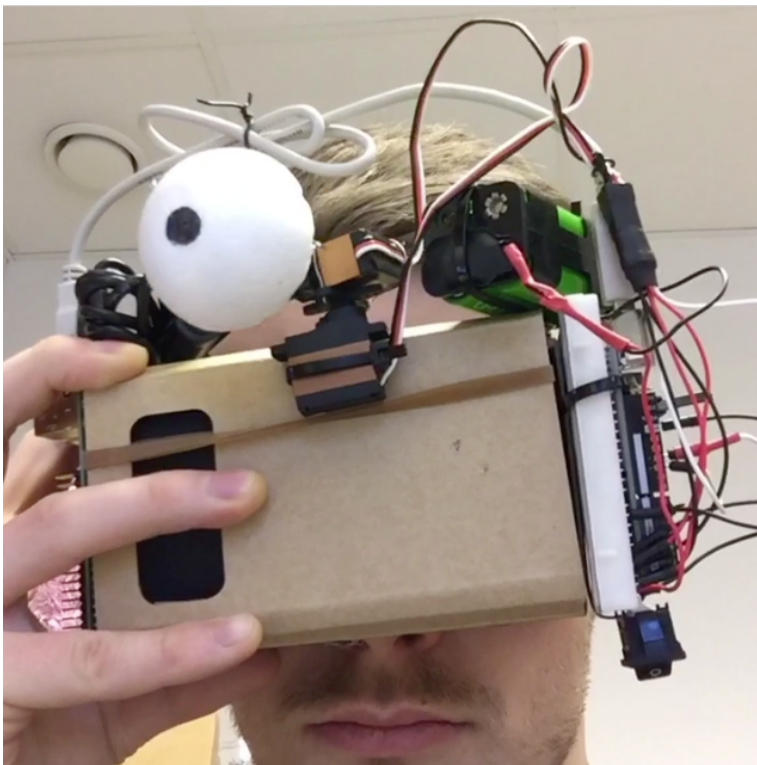
---

## 3.1 Head mounted device

To be able to increase emotional connection between medical personnel and simulator mannequin, a device that can track the eye of the operator while displaying the environment around the mannequin to the operator was investigated. During exploratory prototyping, a head mounted device was tested and managed to prove that it was possible to create actuation based on where the operator was looking.

### 3.1.1 Prototype 0

Nygaard (2016) describes the development of a prototype that showed the possibility of actuating servos with the control input being based on the movement of the operator's eye. This prototype was made from a cardboard casing that is designed to hold your phone while you run a virtual reality app on it. A camera was connected via USB to a Raspberry Pi that was attached to the casing. The Raspberry Pi translated the position of the user's pupil and sent it to an Arduino. The Arduino actuated two servos that moved in two axes, letting a ball simulating an eye move in the same way that the user's eye moved.



**Figure 3.1:** The prototype for the head mounted device from the project thesis. The "eye" moves based on the user's eye movement.

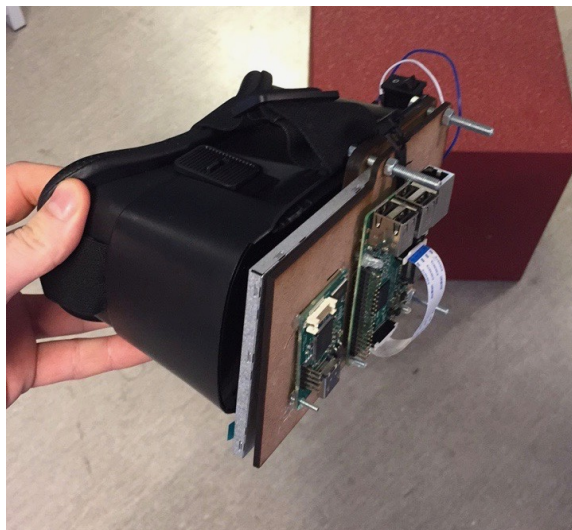
---

From this prototype the key learning was that it is possible to achieve actuation as a response to input from the operator's eyes. However, the actuation of the servos was jerky, there was significant lag, and both the code and the mechanical construction were unstable. Additionally, there was no display inside the casing to display the environment to the operator.

There were several flaws with the prototype, and the tasks for the next prototype iteration were:

- Get a better camera, compatible with the Raspberry Pi
- Find a small, high resolution display compatible with the Raspberry Pi
- Improve the mechanical structure, find a casing that can fit the components and provide a good view of the near eye display.
- Separate the actuation and data gathering, as the actuation would be in the head of the mannequin

### 3.1.2 Prototype 1



**Figure 3.2:** The first prototype for the head mounted device.

The first prototype (Figure 3.2) that was officially a part of this thesis was constructed by using parts of a cheap virtual reality headset. The goal of this prototype was to refine the technological aspects that had been uncovered in the project thesis, and to separate actuation from capturing of data.

The plastic casing made for a sturdier build than the cardboard had offered, and the built in lenses were possible to adjust and focus. A display and camera was purchased and

---

installed in a laser cut casing that was connected to the headset. The laser cut casing also had mounting holes for the Raspberry Pi and the signal boards for the display. Infrared LEDs were installed in the headset to illuminate the left eye. The camera was a Picam NoIR that has no infrared filter, that was mounted straight in front of the left eye of the operator. The display was held in place by the laser cut case, and let the image be displayed only on the right eye.

This prototype showed that it was possible to display an image to the operator and track where the operator is looking. The tracking was still slow and unstable, and the laser cut casing was not optimally designed. There was light leakage and the display was moving around in the frame. The infrared LEDs were running on battery power, which was not a permanent solution, and they did not illuminate the eye sufficiently in the position they were in. The camera offered an accurate view of the eye, and the movement could be tracked from the video stream provided by the camera.

The displayed environment as described in section 4.3.1 created the environment that the operator reacted to. Capturing and processing how the operator reacted to this environment made it possible utilize a human for the control system input. The operator automatically reacted to what they saw, and their gaze subconsciously scanned the environment as it changed, making control output for both micro and macro movements that could be actuated.

The next steps were to optimize the tracking algorithm, increase efficiency of the code and to redesign the frame for the electrical components. The electronic circuit for the infrared LEDs needed to be made permanent, and the positioning of the LEDs needed to be optimized.

### 3.1.3 Prototype 2



**Figure 3.3:** The second prototype for the head mounted device.

---

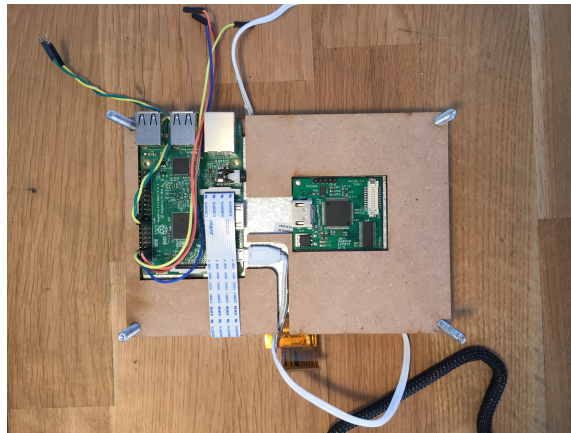
In the second iteration of the head mounted device, the goal was to remake the design from prototype 1. The laser cut frame was improved and the electrical components were better connected to the frame. The infrared LEDs were moved to the left of the users eye, which illuminated the eye better. The LEDs were connected to the 3.3 volt output of the Arduino to be able to run all the time.

In addition to this, threading was introduced to the coding. This means to separate threads (functions calls, tasks) in such a way that the operations can be run in parallel. This allows for the detection algorithm to run separately from the display operations, which makes for a faster detection and displaying.

Further on, the detection algorithm was replaced. Where a Haar Cascade classifier was used in the project thesis (Appendix B, Nygaard (2016)), a Hough Circle detection algorithm made for a much more robust detection. While it was better than the Haar Cascade classifier, the Hough Circle detection needed parameter tuning that would take a lot of testing. The parameters are described in section 4.4.2.

The casing needed to facilitate cable management and to keep everything stable and out of the way. The HDMI cable that was used was too long, and a shorter one was ordered. The circuit boards would also need rearranging to accommodate different wires and to facilitate interaction with the Raspberry Pi in terms of external connections such as keyboard and mouse.

### 3.1.4 Prototype 3



**Figure 3.4:** The laser cut case was too close fitting. Power cable and 40 pin LVDS from the display are interacting.

The third prototype had a goal to encapsulate all components in a case, hidden from view. The prototype enclosed the electronics fully in a laser cut case that was attached to the plastic headset. The infrared LEDs were embedded in the virtual reality headset.

Key learnings of this prototype were that the cable arrangement was sub-optimal. There were cables that interacted mechanically when using the device. The 40 pin LVDS



---

cable was being tugged by the power chord when moving the headset around, as seen in Figure 3.4, which might damage it long term.

Due to the close fit of the laser cut parts, there was some heat build-up, and not all cables fit perfectly. In addition to this, the Raspberry Pi had a tendency to short out due to the lack of insulation between it and the backside of the display, which is made in metal. This was the result of removing one layer of laser cut MDF that seemed unnecessary. Next prototype would demand a restructure of the laser cut case and added insulation between the display and the Raspberry Pi.

### 3.1.5 Prototype 4



**Figure 3.5:** Final prototype of the head mounted device in this project.

The goal of this prototype was to refine the previous iterations into a functional design. To counteract overheating, a silent computer fan was installed, powered with USB power from the Raspberry Pi. The laser cut casing was redesigned to better accommodate cooling and wiring. The casing was properly attached to the headset with a sealing agent that kept light out and held the case securely in place, and a computer fan was installed to cool the components.

The head mounted device was now robustly constructed, operational over longer periods of time, and the code was being optimized for stability, user friendliness and detection accuracy. The prototype had eliminated shortage of the Raspberry Pi, and the mechanical structure of the case was optimized for longer operations. The electronics were organized and kept stable by the case, so that there was no mechanical wear on them.

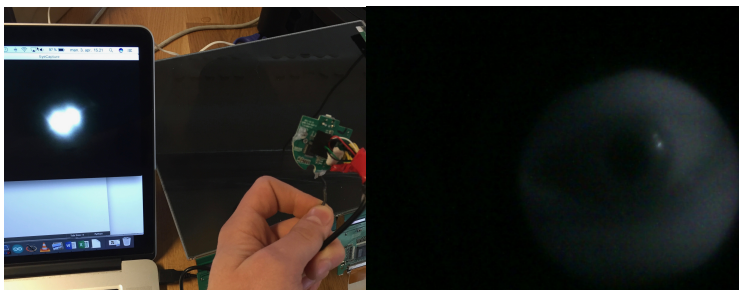
---

Further work on the head mounted device was purely done in terms of software, and this prototype is considered an alpha prototype for the head mounted device. A next iteration could have been built from scratch with better/more efficient electrical components and in a lighter material.

### 3.1.6 Looking through the screen

One idea of looking through the display in the headset was investigated by placing a camera behind a standalone deconstructed display, and detecting infrared through the display. Some rough prototypes were made to confirm that it was indeed possible to detect infrared through the LCD screen, when removing the backlight. When this was confirmed, the screen was mounted in front of the camera, and the screen was put on the headset. The screen was off, with no backlight, and it was possible to detect the pupil of an eye illuminated by infrared through the display, as seen in Figure 3.6.

Although it was possible to see infrared images through the screen when the screen was turned off and had no backlight, there are some major hurdles to overcome before managing to produce a camera and display module that can perform the tasks that are required in this project. For the screen to operate properly, the backlight panels need to be removed where the camera is placed, and there might be a problem with illuminating the display. Further on, the LCD functions by polarizing the crystals in the display matrix, which will reduce the amount of light that shines through, making this even harder to achieve. The feasibility of this module probe is low, but with a high amount of research, it might be possible.



**Figure 3.6:** Seeing through an LCD screen without backlight panels. Left image shows a camera that detects infrared light through the screen. Right image shows an eye as it is observed with the camera through the screen.

---

## 3.2 Eyes of the mannequin

Several ways of actuation were explored during the development work in this thesis were based on the learnings described in Appendix B (Nygaard, 2016). The prototypes for actuation are described in this section, detailing learnings they represent and why they are explored.

### 3.2.1 Magnet actuation

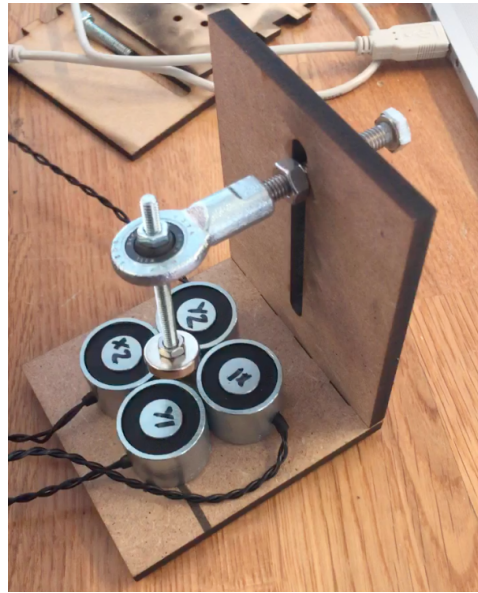
As a result of the prototyping in the project task, a potential for noiseless actuation by way of magnets was discovered and this was further investigated here. A rig was created to be able to actuate a permanent magnet attached to an actuation rod, by controlling the power of electromagnets. The power of the magnets was adjusted with pulse width modulation (PWM) of the power supply. The magnets were controlled by 2 potentiometers and an Arduino. The Arduino generated PWM signals that controlled the power supply to each electromagnet through a MOSFET circuit, as seen in Figure 3.7.

It was possible to move the magnet accurately in two axes using the setup in Figure 3.7. Still, this control could only be exhibited if the actuation rod hung vertically when the magnets were turned off, i.e. the rig needed to be placed flat on the ground.

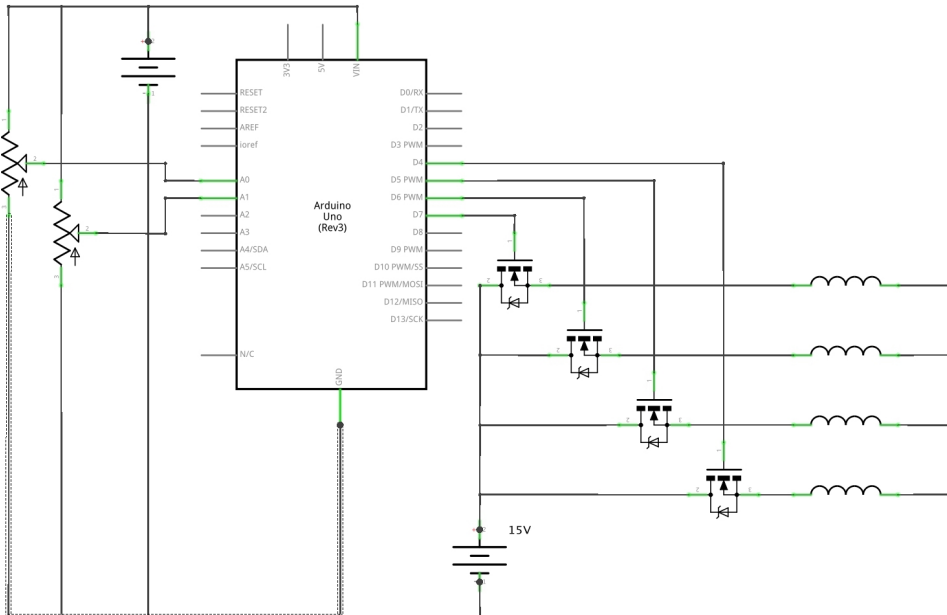
If the rig was tilted, gravity was stronger than the power of the magnets on low PWM cycles. This means that the actuation would only start when the electromagnets managed to overpower gravity. When this happened, the permanent magnet would jerk towards the electromagnet rapidly. Thus it was impossible to tilt the rig without some mechanism of keeping the permanent magnet's default position straight above the center of the electromagnet grid.

One solution was attempted in order to overcome the issue about gravity, namely to connect a spring to the actuation rod that kept it at the default position. This did not solve the problem, as even more power was needed to overcome the force of a spring that was powerful enough to overcome gravity.

In addition to the above difficulties, the fact that noise is minimal with electromagnets is overturned when using PWM. The pulsing signals range in frequency, but the frequencies are in the audible spectrum, and the force of the magnet creates vibrations in the rig and thereby also sound. Further on, the coils in the electromagnets have some resistance, and the magnets eventually started to grow hot when high power was applied.



(a) Magnet rig with a permanent magnet over 4 electro-magnets.



(b) Control Schematic. 4 electromagnets, 4 MOSFETs and 2 potentiometers, controlled by an Arduino Uno.

**Figure 3.7:** The magnet rig.

---

### 3.2.2 Projected eye

While looking into methods of actuation as an alternative to mechanical actuation, projecting an image of the eyes onto a spherical surface was investigated. These prototype iterations had the goal of investigating if this was a viable method for designing eyes. The first prototype was created by vacuum forming a clear plastic half sphere. This half sphere was sanded down to better capture the projected light. A projector was held behind the surface and moved back and forth to simulate movement of the eye.

The prototype showed that the appearance of the projected eye was promising. Still, the sharp light emitted from the projector proved somewhat annoying, and diminished the realism of the prototype.

The second iteration of the prototype added crude eyelids, made the iris larger and added a camera above the eye. This camera was connected to a computer that ran a facial recognition algorithm while displaying the iris. The prototype made the eye look straight at the face of the person that looked at it, if their face was in the frame of the camera.

While the prototype's behavior was impressive and users reported that it felt really cool, the sharp light from the projector made it very unrealistic. Future work would be to develop some way of diminishing the light shining through.



(a) First prototype

(b) Second prototype

**Figure 3.8:** The two eye projection prototypes.

---

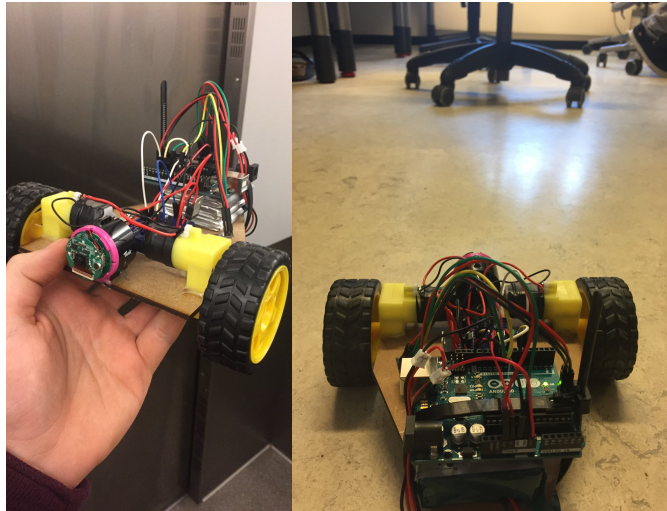
### 3.2.3 Eye controlled robot

While investigating wireless actuation control, a simple robot was used to prototype the transmission and response of signals. This goal of this prototype was not to make a robot, but to rapidly test how it was possible to create wireless control and transmit a video feed wirelessly in a dynamic environment.

This robot was communicating with an Arduino connected to the head mounted device with wireless radio frequency (RF) data transmission to another Arduino. The robot had a wireless Trek AI-Ball WIFI camera that could communicate with the Raspberry Pi in the head mounted device, which in turn can display the image from the camera to the operator.

The robot showed that it was possible to control the actuators wirelessly while displaying the image from the camera to the operator. As the operator looked to one side, the robot would drive to that side. The range for the camera is estimated to around 30 meters, or around 7 meters with a closed door in between. The RF chip's range is much larger and is not a limiting factor. The testing showed the possibilities of the wireless transmission of video and actuation signals, and the concept was adapted into the next prototypes.





(a) Robot from the front

(b) Robot view



(c) Camera view

**Figure 3.9:** The eye controlled robot and what the operator sees inside the head mounted device.

---

### 3.2.4 Camera mounted inside eyes

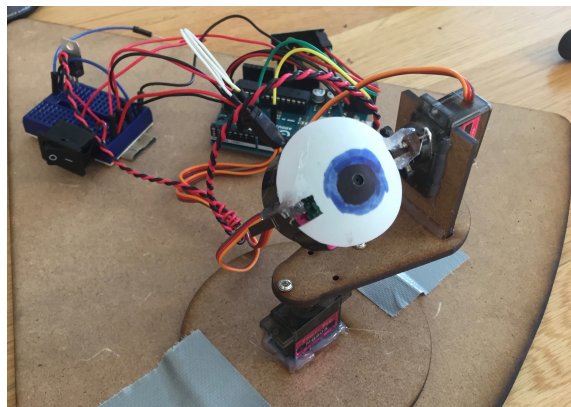
The idea of putting the camera inside one of the eye balls was investigated with the prototype in Figure 3.10. By mounting the camera inside the eyes, the camera would be invisible to the observers, as opposed to a stationary camera outside of the eyes, which would be visible. The goal of this prototype was to investigate how it feels for the operator to have a camera moving along with the eyes of the mannequin.

While a stationary wide angle camera would provide a stationary frame of vision, an eye-mounted camera would move as the operator looks around at the environment, thus shifting the frame of vision. The prototype in Figure 3.10 allowed for the eye to look in the direction the operator looked. The two servos actuate the vertical and horizontal movements.

During testing it was discovered that as the operator looks at one point in the image frame, the camera moves to look at that point, thus changing the frame of vision. When the frame of vision is shifted, the operator will look back at the object, which is now at a different point in the frame, and the frame of vision will shift again. This caused jittering of the prototype eye, and it was strenuous for the operator.

It was found that the jittering could be reduced by introducing a buffer for when to actuate the movement, and also by smoothing the motion through a moving average filter, although these modifications affected the behaviour of the eye in such a way that it was observed to be unnatural. The modifications to the movement also made the eye unable to follow objects in a realistic manner. In addition to this, there is a slight delay in the circuits and program. This means that after the eye movement happens and before the frame is shifted there is about a 200-300 millisecond delay, which contributes to the strain on the eyes of the operator.

The prototype showed that an eye mounted camera was strenuous to operate and the operations that could fix this were affecting the observed behavior of the prototype eye. The feasibility of this method might improve if there is more investment into high end equipment to reduce delay and increase frame rate, though this is not pursued in this project.



**Figure 3.10:** Eye mounted camera



---

### 3.2.5 Animatronic eyes with actuator rig

To be able to better replicate human to human communication, animatronic eyes were ordered from a special effect manufacturer. These eyes are made to look like human eyes, and this may help to accommodate realism in the appearance and behavior of the product, as seen in Figure 3.11.

The eyes were delivered with an aluminium frame and 8 servos to actuate the movement of the eyes and eyelids. The servos were controlled by the Arduino and powered by a 6 volt power source. The eyes are documented in detail in section 4.3.2.

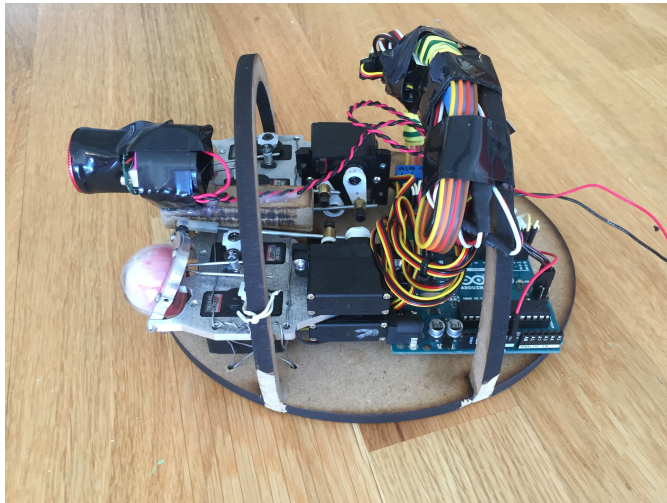
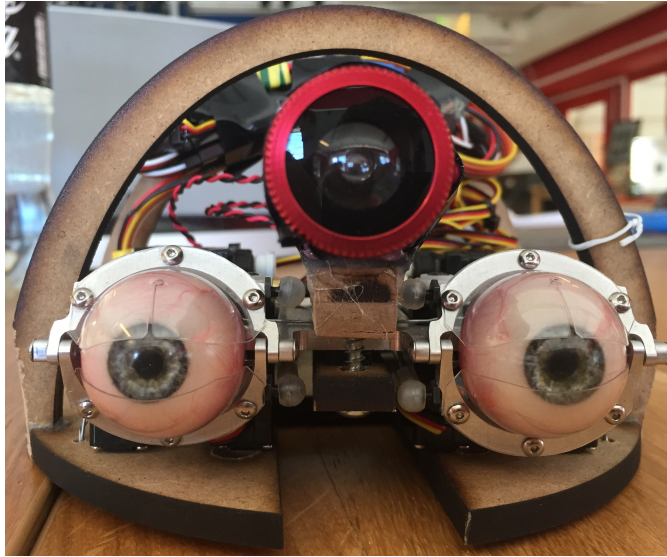
The animatronic eyes were installed into a laser cut frame that was designed to fit into the head of the mannequin. The frame was intended to hold the components in place inside the head. Two screw holes are included in the frame for connecting the frame to the head of the mannequin through the mouth, and the top of the frame is pressed against the top of the head. The frame holds the eyes in the center of the eye holes in the mannequin's head.

The electronic components were installed in the frame behind the animatronic eye rig. The electronics are detailed in section 4.3.2. The actuation control, power management and Arduino communication was possible to incorporate with the only wires being for power.

To mount the wide angle camera, an adjustable holder for the camera was attached to the aluminium frame. This holder clamps down on the frame and is held in place by friction. Two screws connect an upper and lower laser cut part and pull them together. The camera is attached to the upper part.

The camera module is designed to protrude the forehead of the mannequin's head. This will allow for a wide, stationary view of the environment around the mannequin, while the general appearance of the prototype may suffer. There are smaller cameras than the one utilized in this project, that would be less visible, but can provide the same field of view.

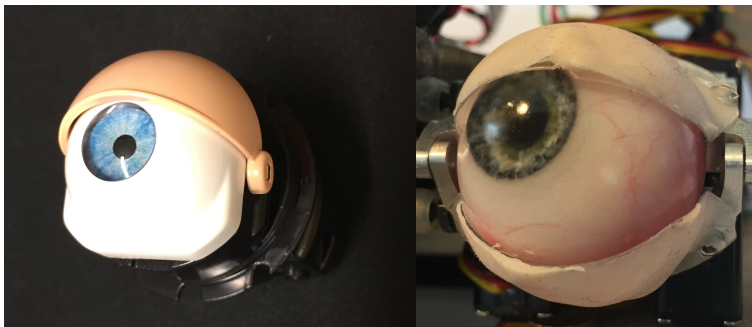
This prototype is considered the final prototype of the actuation and actuation control modules, and is described completely in section 4.3.2. The prototype provided a solid platform for interaction and actuation. The largest drawback of the rig was the noise from the servos, although this was not prominent when installed inside the head of the mannequin and tested in a workshop environment.



**Figure 3.11:** Animatronic eyes with a stationary wide angle camera, electronics are installed behind the eyes.

# Chapter 4

## System details



**Figure 4.1:** The difference in appearance between the eyes of Lærdal’s mannequin, and the prototype described here.

The goal of the development of this alpha prototype is to create human like movement and behavior of the eyes in the mannequin. By using a human as the control system input, it may be possible to bridge the uncanny valley (Ishiguro, 2007) by removing the tiny differences that separates natural from unnatural. This product may be useful in scenarios where communication is important and it may be used to diagnose certain diseases. A simple test, like following an object, may give insight towards the condition of the patient’s brain. In addition, non verbal communication and eye contact is an important factor in a patient-doctor relationship, and this product may provide a better training experience for the medical personnel that participates in the simulation.

This chapter describes the purpose and details of the complete system as well as each subsystem. The details included are essential for the function of the system as it is. Each subsystem has evolved through a thorough development process, and the results are described here.



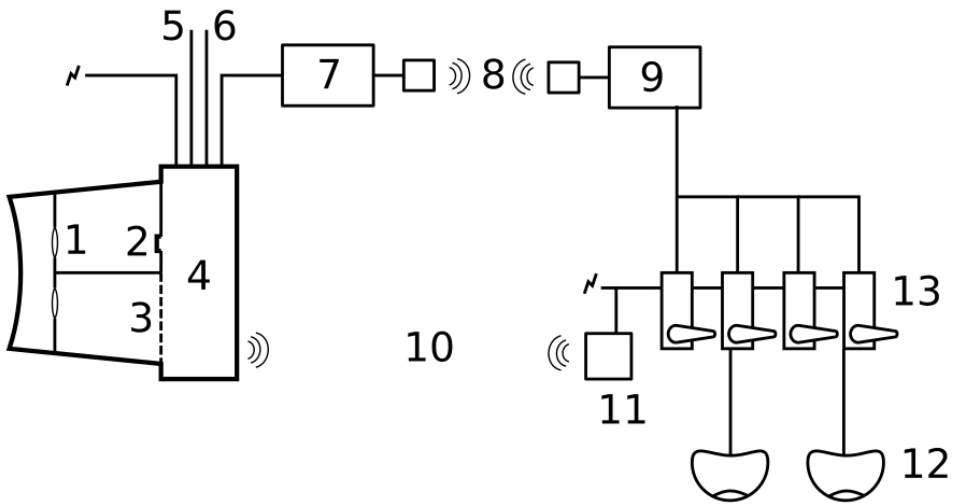
**Figure 4.2:** The system in operation.

---

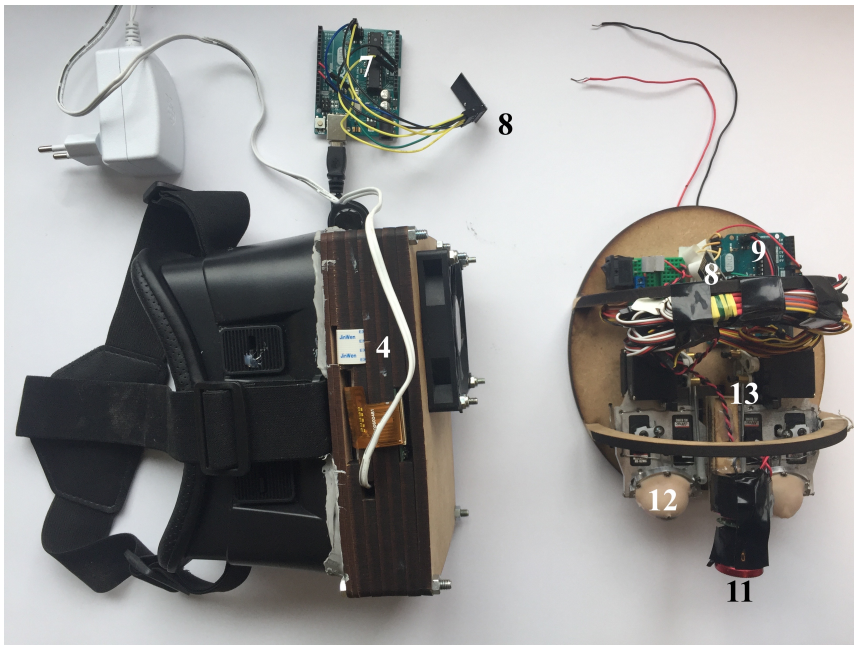
## 4.1 Complete system description

Figure 4.3 describes the functions of the system. The functions or components of the system are listed below, with corresponding numbers in Figure 4.3. The details of the components and functions can be found in sections 4.3 and 4.4.

1. Lenses
2. Eye tracking camera, Picam NoIR
3. Display
4. Raspberry Pi and display signal boards
5. Computer Keyboard
6. Computer Mouse
7. Arduino Master, attached to HMD
8. nRF24 communication boards
9. Arduino Node, attached to actuation rig
10. WIFI communication between camera and Raspberry Pi
11. Trek AI-Ball WIFI camera
12. Eyes actuated by servos
13. Servos



(a) Functional map of the system



(b) Picture of the system, only visible elements are numbered.

**Figure 4.3:** Functions of the system.

---

## 4.2 Usage and operation

The system, as described in detail in section 4.3 is intended to be used by an operator and consists of a head mounted device that is worn by the operator, and an actuator rig that is installed in the head of the mannequin. The system lets the operator see through the eyes of the mannequin and the mannequin's eyes will look where the operator is looking.

One usage scenario may be a simulation where the patient is conscious, can talk and look around. The medical personnel approach the patient and get eye contact. The patient is meeting their eyes and talking back. Initial questioning reveals that the patient has fallen and hit their head. The medical personnel asks the patient to follow their finger back and forth. The test reveals no sign of anomalies, but the medical personnel assess that the patient is in distress based on rapid shifts in gaze, unwillingness to maintain eye contact and excessive blinking.

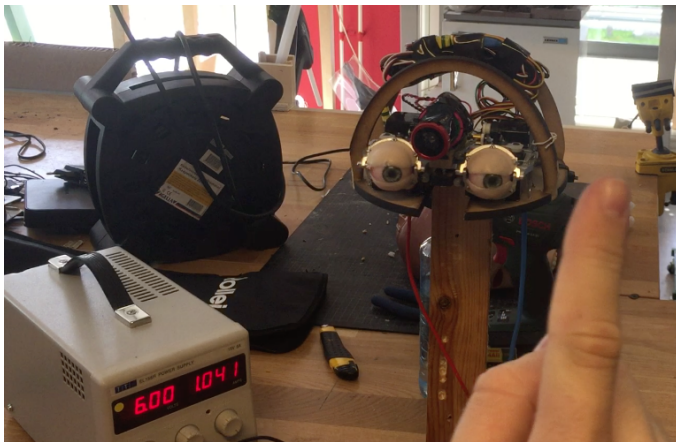
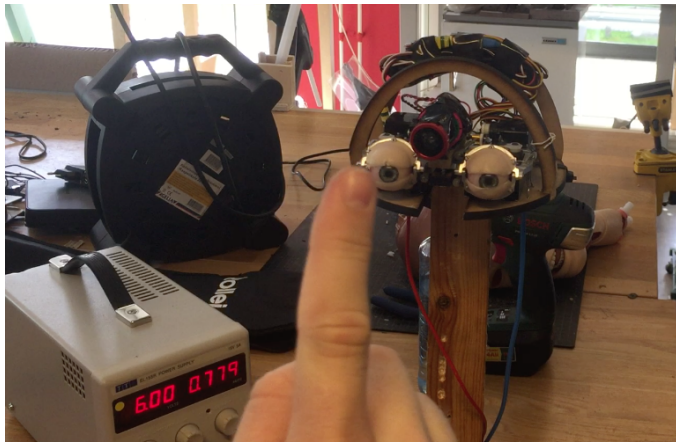
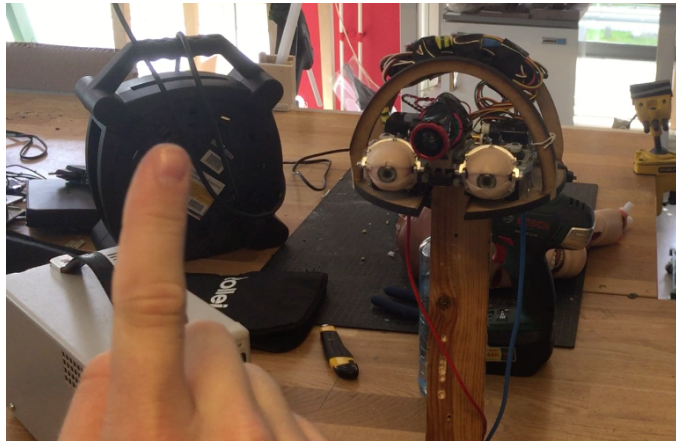
In the above scenario, the eyes would be controlled by an operator that sits in adjacent room. They are looking through the eyes of the mannequin and are acting as if they are the patient. This allows them to control the situation in terms of communication and eye movement. Other symptoms may be simulated in the same way as they normally would.

The presented system provides many new aspects of interactions between mannequin and medical personnel. The system allows the operator to interact directly with the simulation participants through the mannequin's eyes. Some diagnostic methods rely on observation of eye movement and behavior in dynamic tests, such as the test described above. Following objects in a smooth pursuit, as well as rapid shifts in visual direction can be telling towards neurological disorders and mental state. Focusing on a dynamic object might be more difficult if the patient is under the influence of drugs. Pictures from smooth pursuit can be seen in Figure 4.4.

Further on, the system introduces another aspect of communication with the mannequin. The main focus of visual attention during human-human interaction is the eyes and the region around the eyes. Implementing human like behavior and movement of the eyes, may give medical personnel a more empathetic view of the mannequin, and it will be easier to talk directly to the mannequin when eye contact is mutual.

The eyes are often the first point of contact for medical personnel, as the eyes are telling of the medical condition of the patient. The eyes can, at a glance, answer questions such as: "Is the patient awake?", "Are they apathetic?", "Are the eyelids droopy, or do they struggle to focus?". The eyes contribute to the immediate impression of the well being of a patient.





**Figure 4.4:** Smooth pursuit of a finger.



---

### 4.2.1 Running the prototype as it is today

The main goal of the prototype is human-machine interactions, which means that there has not been a focus on developing a user interface. The current prototype is controlled from the Raspberry Pi operating system Raspbian Jessie and the program is run through the Terminal by keyboard input. The exact commands are described in section 4.4. After making sure that the Raspberry Pi is connected to the WIFI network of the Trek AI-ball camera that captures the mannequin's view, the program can be initiated. This will display the wireless video stream of the mannequin's view to the operator, inside the head mounted device. At the same time, the head mounted device will start tracking the movement of the eyes of the operator and send the actuation signals to the actuators in the eyes. The program can be shut down by pressing q on the keyboard.

## 4.3 Hardware

The hardware of the system has evolved into two separate systems that communicate wirelessly. The prototype that was developed in Appendix B (Nygaard, 2016) consisted of a single system that showed the possibilities that the available technology presented. Further development of the system required the system to split into two subsystems, the head mounted device, to be worn by the operator, and the actuator rig that is placed inside the head of the mannequin.

### 4.3.1 Head mounted device

For an operator to control the eyes, the operator needs to see what the eyes of the mannequin "see" (this view is later referred to as the mannequin's view, or in the code: environment). There are several possibilities for both capturing and displaying the mannequin's view. To be able to create a one-to-one behavior of the eyes of the mannequin and the eyes of the operator, in such a way that the mannequin can interact with medical personnel in a meaningful manner, the mannequin's view must be displayed to the operator in a way that feels natural.

#### Raspberry Pi

Development of the system is done with a Raspberry Pi as the core processing unit, using available communication protocols and interfaces on the single board computer to control and interact with the hardware that is used. The computer was chosen for its ease of use, processing power and available online community for support and help. The Raspberry Pi runs the operating system Raspbian Jessie and the programming language that is used is Python. The Raspberry Pi can be seen in Figure 4.7 (b), (c) and (d) as well as in Figure 4.8.

The Raspberry Pi model 3 was chosen because it has a quad-core Cortex-A53 processor that can provide high performance when using threading of the processes. This prototype uses the following of the Raspberry Pi's ports:

- MIPI camera serial interface (CSI), for Raspi NoIR camera.

- 
- HDMI port connected to the MIDAS MCIB-14 board
  - General Purpose Input/Output (GPIO) pins:
    - 5 volt output to MIDAS MCIB-14 board
    - 3.3 volt output to infrared diodes
    - Ground to MIDAS MCIB-14 board and infrared diodes
    - I<sup>2</sup>C Serial Data Line (SDA)
    - I<sup>2</sup>C Serial Clock Line (SCL)
  - Micro-USB port for power supply. The power supply can deliver 5 volts at up to 2.5 amperes.
  - USB ports:
    - Cooling fan
    - Arduino Uno
    - Computer keyboard
    - Computer mouse

### **Display and lenses**

Displaying the mannequin's view to an operator could be done in several ways, and there are positive and negative aspects to all of them. There are ways of tracking a person's gaze both with head-mounted gear or with stationary cameras, and the same is true for displaying the mannequin's view.

Displaying the view on a PC desktop monitor would be the simplest solution in terms of hardware, although this would create difficulties with tracking the gaze of the operator. There are libraries available for tracking a person's gaze with a stationary camera, but they require a lot of processing power, and are prone to being inaccurate.

Near eye displays (NEDs) are getting more and more common, as virtual reality tools become more available and powerful. An Oculus Rift headset can be seen in Figure 4.5. NEDs allow for compact form factors and with small, high resolution screens, they can display an image that looks as sharp as with a desktop monitor.

In Appendix B (Nygaard, 2016) it was shown that capturing eye movement with a camera close to the eyes was possible. Putting both the display and the camera inside a head mounted device makes both displaying and capturing is easier, as head movement no longer affects either. To accommodate both capturing of the operator's eye movement and the display, the mannequin's view is only displayed on one eye, as shown in Figure 4.6.

The display that is used for displaying the image on the right eye has a resolution of 512 x 600 pixels. The lenses in the head mounted device allows the operator to focus on the screen. While a higher resolution would be preferable on a near eye display, this screen costs, with all accessories, around 600 NOK, making it very affordable. The resolution is sufficient to be able to recognize faces from a large distance.

The screen is connected to the Raspberry Pi via HDMI. While the Raspberry Pi offers a MIPI display serial interface, displays that can be connected to this port are mostly larger



**Figure 4.5:** Oculus Rift virtual reality headset, gathered from (Walton, 2016)

displays with lower resolution and touch screen. This display is a thin film transistor (TFT) liquid crystal display (LCD) with a low voltage differential signaling (LVDS) interface. This technology offers low cost, but to communicate with the Raspberry Pi, the LVDS signals needs to be converted to HDMI. This conversion is performed in two steps, as shown in Figure 4.8. HDMI is converted to 20 pin LVDS interface with the MIDAS MCIB-14. This board requires I<sup>2</sup>C communication of data and system clock as well as a 5 volt power supply with a connection to ground. This is provided through the colored wires in the figure. The 20 way LVDS cable connects the MDAS MCIB-14 to the MIDAS MCIB-16. The MIDAS MCIB-16 converts the 20 pin input to the 40 pin output that the display requires.

### **Arduino**

To actuate the mechanical eyes, as described in section 4.3.2, it is not possible to use only the Raspberry Pi, as the single board computer does not have the hardware to drive the pulse width modulated servos. It is possible to run one servo with the available pins of the Raspberry Pi, but to be able to control more, a micro-controller is needed. Arduino is a company that produces single board micro-controllers for prototyping, that are programmed using a dialect of C and C++.

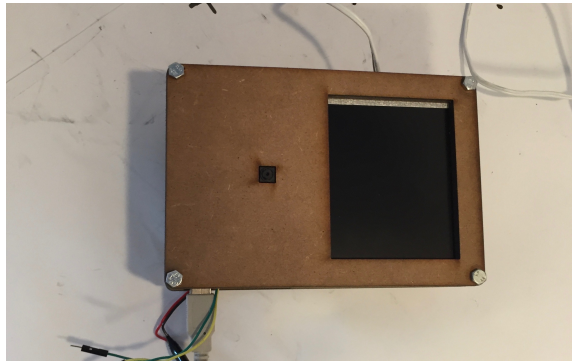
An Arduino Uno microcontroller is connected to the Raspberry Pi via USB, to allow for power and serial communication through a single cable. Further on, the Arduino Uno uses a nRF24-chip which is an ultra low power 2Mbps radio frequency transceiver integrated circuit for the 2.4GHz band. This chip allows for wireless communication to one (or more) external Arduino over significant distances. The RF communicates to an Arduino Uno placed in the actuator rig as described in section 4.3.2.

### **Camera**

A Picam NoIR is used to capture images of the operator's left eye. The Picam NoIR has no IR-filter, hence the name. This camera was chosen in order to be able to see inside the head mounted device. The camera is engineered to be used with the Raspberry Pi, and connects to its MIPI camera serial interface port. This connection allows for high speed



(a) The head mounted device with lenses installed.



(b) The head mounted device without lenses, the camera protrudes to the left.

**Figure 4.6:** The head mounted display displays only an image to the right eye of the operator, while the left eye is tracked with a camera.

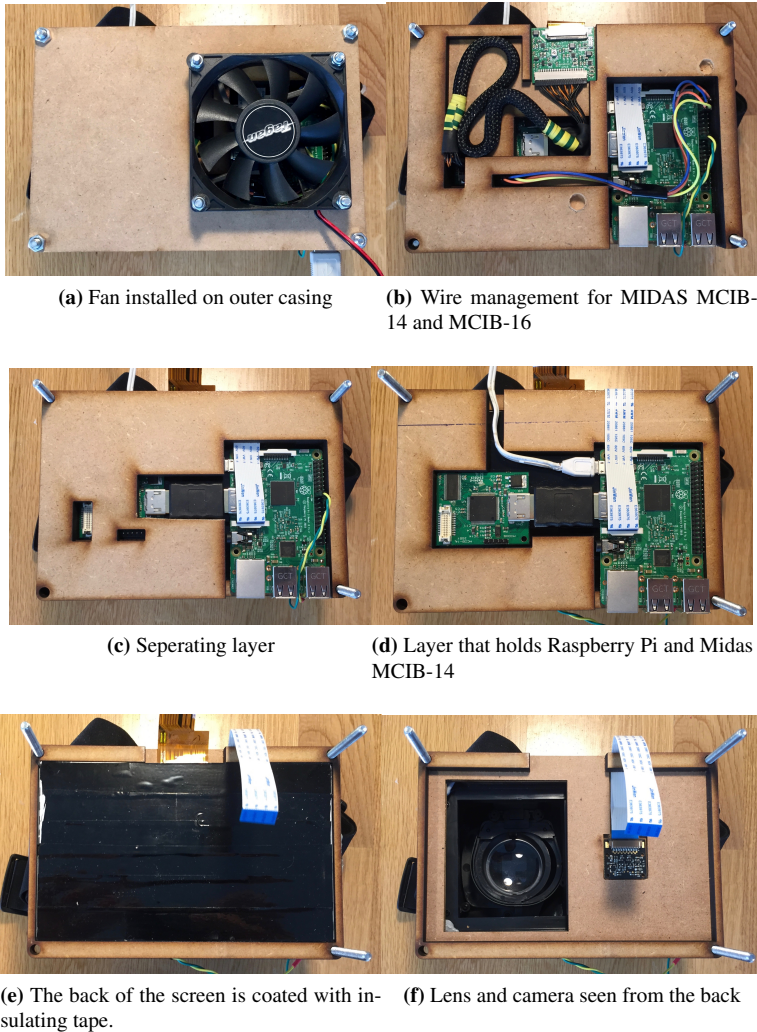
capturing of images. The camera has the ability to capture still images of up to 2592 x 1944 pixels, while at 640 x 480 pixels, it can capture up to 90 frames per second. The camera can be observed as it is installed in the casing i Figure 4.7 (f).

### **Infrared lighting**

To be able to see the eye of the operator when the prototype is worn on the head, infrared lighting has been used. This makes it possible to see the operator's eye without them noticing, as it is completely dark on the left eye. The prototype has installed 3 infrared diodes just left of the left lens as seen in Figure 4.6 (a), and they are connected to the Raspberry Pi's 3.3 volt output pin and ground as shown in Figure 4.10.

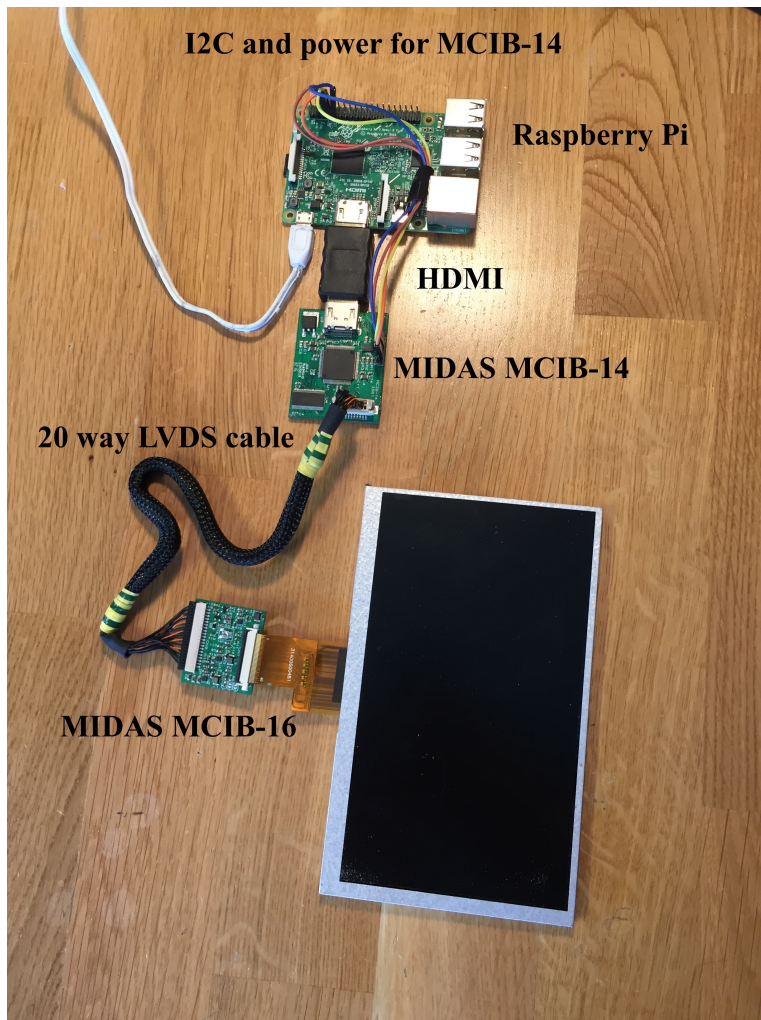
### **Cooling**

As the Raspberry pi is running processes that are very demanding, the temperature of the processor will rise over time. To avoid overheating, a computer fan was installed. The



**Figure 4.7:** The casing for the processing, screen and camera is divided into layers to accommodate wiring, cooling and robustness.

computer fan runs on 5 volts from the Raspberry Pi's USB port, and has no noticeable noise. To be able to run the fan from the Raspberry Pi, the USB port was chosen, as it can deliver a higher current than the available 5 volt pins. The fan creates an air flow that runs through the housing, cooling all the components. The fan can be seen in Figure 4.7 (a).

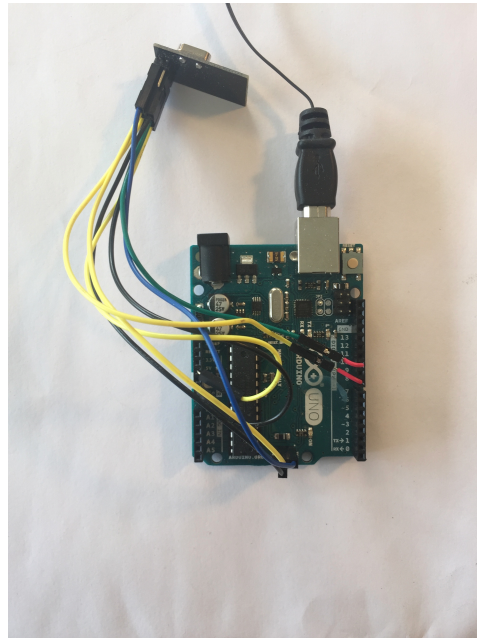


**Figure 4.8:** The connection between display and Raspberry Pi. Communication goes through HDMI and is converted to the LVDS interface of the display in two steps.

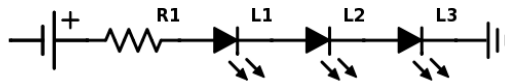
### 4.3.2 Actuation

The actuator rig is designed to be installed inside the mannequin's head. The design does not consider existing technology that is already embedded in the head of the mannequin. It is designed to operate only connected to a power supply, communicating over radio frequency and WIFI to the head mounted device. This allows for the rig to operate completely independently in the mannequin, given that the power supply is able to supply sufficient current.





**Figure 4.9:** Arduino with nRF24 board connected.



**Figure 4.10:** How the infrared diodes are wired. R1 = 100 ohm.

### Eyes and eyelids

The mechanical eyes, as well as the aluminium frame with mounted servos, was produced by Dan Thomson, who runs [www.animatronicparts.com](http://www.animatronicparts.com). He creates high-end products that attempt to visually replicate human appearance. The product offers lifelike eyeballs that are hand made, and the eyelids are made to be modified. The prototype was fitted with silicone eyelids of a skin color.

A high-end product was chosen for the prototype to better be able to illustrate human like behavior. This was decided after testing revealed that low-end prototypes did not provoke the feeling of realism that was desired. In Figure 3.10, a low-end prototype can be seen, and it is clear how different the appearance looks and feels compared to the prototype seen in Figure 4.2.

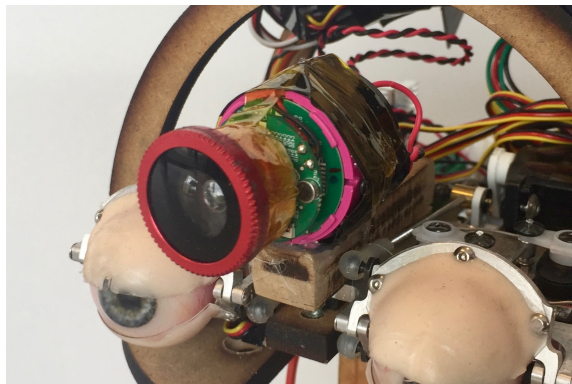
---

## Camera

The wireless camera is a Trek AI-ball WIFI camera. This camera generates a WIFI network that the Raspberry Pi connects to automatically. The camera then generates a continuous video stream that is run on a local website on the network. This stream can be read by the Raspberry Pi.

The camera is run by the same power supply as the servo motors, but the voltage is reduced from 6 volts to 3.3 volts through a voltage regulator. The camera requires around 500 milliamperes of current and is included in the circuit as seen in Figure 4.13.

The camera was mounted with a wide angle camera lens to give a larger field of view. The wide angle lens has a field of view of about 160 degrees. The wider field of view allows a better interaction with the environment around the mannequin, as more is visible.



**Figure 4.11:** WIFI camera with wide angle lens

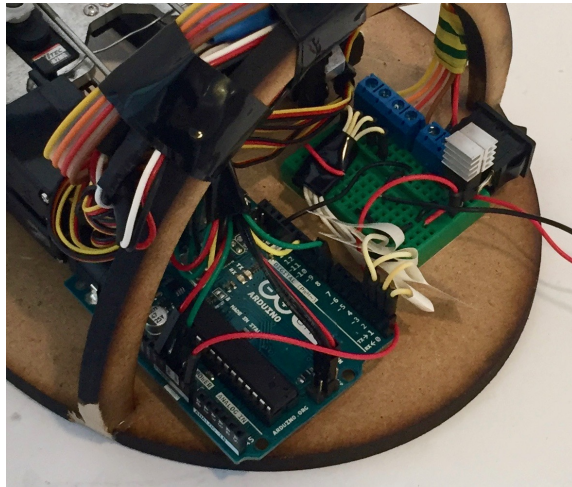
## Actuation control

Actuation is controlled through the Arduino Uno as shown in Figure 4.13. The control input is received through the nRF24 chip, which is sent by the Arduino that is connected to the Raspberry Pi. The actuation is controlled by assigning an angle to the servo. This angle corresponds to where the operator is looking at in the mannequin's view. The complete control system code can be read in Appendix A.5.

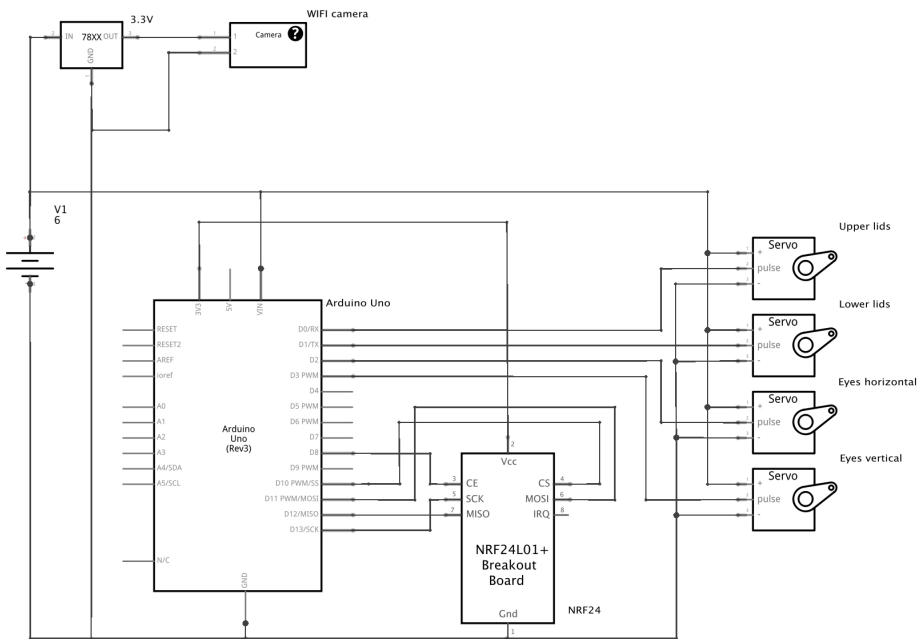
The eyes and eyelids are actuated by 8 digital servos that operate in pairs. The pairs are controlled by 4 signals, and they control upper lids, lower lids, vertical movement of the eyes and horizontal movement of the eyes, respectively. The servos are mounted on an aluminium frame, and are connected to the actuated parts via mechanical arms.

The servos are connected to the Arduino Uno as shown in Figure 4.12 and 4.13. The servo pairs are represented as single servos in the schematic, as they are operated with the same signal, power and ground. They are controlled using pulse width modulation (PWM), which means that a specific pulse width represent 0 degrees, and increasing this will increase the angle of the servo.





**Figure 4.12:** Arduino and power management, as described by schematics in Figure 4.13



**Figure 4.13:** Schematics of the actuation control

The servos share a power supply and ground connection and the required current peaks at around 4 amperes. This means that for the camera to not loose power, the actuator rig needs to have available a minimum of 4.5 amperes. Another solution may be to have

---

individual power supplies for the motors and the camera.

## Mounting rig

The mounting rig is constructed to contain all the components that are mounted inside the head, and to position the eye balls in the center of the eye openings in the mannequin's head. The rig has supports for the top of the head and holds the correct shape of the skin of the head. The rig is made from laser cut medium density fiber (MDF) plates, 6 mm thick and wood. The electrical components are glued to the surface platform, while the actuators are connected with steel wire through holes in the MDF platform and through the aluminium frame that holds the servos, as seen in Figure 3.11.

## 4.4 Software

This section details important aspects of the software of this prototype. The code described in this section can be found in Appendix A.

### 4.4.1 Raspberry Pi operating system

The Raspberry Pi runs the operating system Raspbian Jessie, and it has been configured to display the screen on the near eye display mounted in the head mounted device. The display configuration is done by editing the boot file `/boot/config.txt`. The following alterations have been added to the configuration file to make the display work correctly and only show an image on the visible half of the screen:

```
hdmi_timings=1024 0 16 44 100 600 0 10 3 10 0 0 0 60 0
    51000000 7
config_hdmi_boost=4
hdmi_group=2
hdmi_mode=87
start_x=1
gpu_mem=128
dtparam=spi=on
dtparam=i2c_arm=on
```

The Raspbian Jessie operating system comes with a Python 2.7 and Python 3 interpreter. In this project, Python 3 is being used. Other features that have been changed in the operating system are to enable the I<sup>2</sup>C ports and to activate the MIPI CSI interface.

### 4.4.2 Python

Python is a high level programming language that is designed to be easier to read and learn, and lets the user solve tasks in fewer lines of code than for example C++ or Java. Python can be used for both small scale and large scale programming, and has a large amount of modules that can be downloaded and imported when needed. More details about Python is described in section 2.4.2

---

The open source computer vision library OpenCV (Bradski, 2000) can be imported as a module in Python after it is downloaded on the computer. It contains methods for manipulating images and for detecting a wide range of objects. Along with the scientific computing module NumPy, OpenCV allows the user to manipulate images effectively and with a wide range of options.

The code for this project is divided into 4 Python scripts and uses threading to utilize more of the processing power of the Raspberry Pi. The code can be read in full in Appendix A. The scripts may be run individually, but are designed to be run from the main script, main.py.

Separating the detection and wireless streaming operations from the main script increases the frame rate of the displayed images. This is due to the display functions being independent from the image manipulation operations. These operations are demanding in terms of processing, and they may be slower than the frame rate of the video stream. By running detection and wireless streaming in separate threads from the main script, the displayed images will have a much higher frame rate and at the same time the detection of the pupil will be much faster.

### Interacting with DetectPupil.py

The script DetectPupil.py contains a class for detecting pupils in an image that is read from the Picam, and sends these coordinates to the Arduino via serial communication. The class DetectPupil can be imported from the file DetectPupil.py by running

```
from DetectPupil import DetectPupil
```

and when the script is initiated from main.py by running

```
pupil = DetectPupil()
```

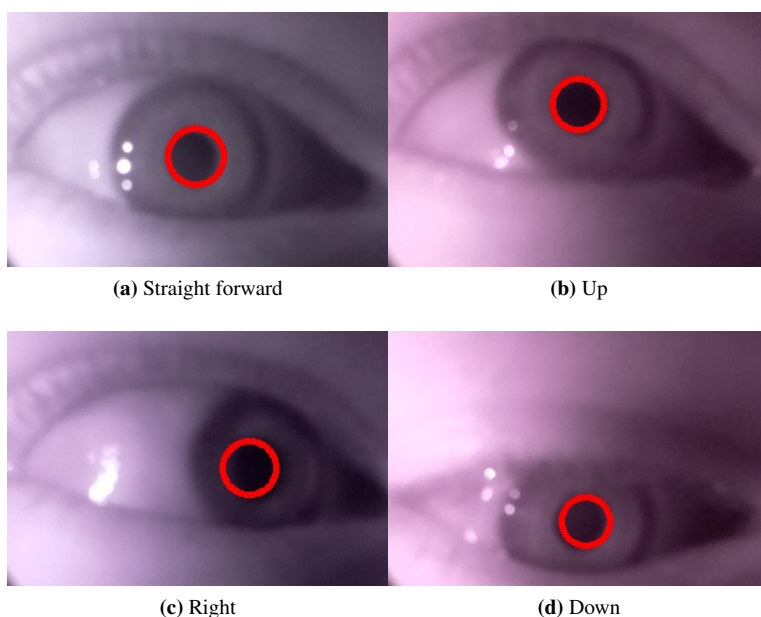
the `__init__()` function defines the required parameters, initiates the camera, and if specified, initiates Arduino communication. The detection class is named pupil for simplicity. After initiation, a processing thread is started by running

```
pupil.start()
```

which starts a thread that runs the `update()` function. This function will run continuously in this processing thread until it is explicitly told to stop. The function finds the pupil, translates the position into bytes and sends them to the Arduino before updating the image frame.

### How DetectPupil.py works

In this prototype, a Hough Gradient Method (OpenCV, 2017) is used for detection of circles, utilizing OpenCV's function `cv2.HoughCircles()`. This method finds gradients and determines the positions of gradients that are connected, creating edges. This is done using OpenCV's `Canny()` edge detector. These edges can be investigated for whether they are shaped like a circle. In Figure 4.14, the detected pupil is shown, as the eye is looking in different directions.



**Figure 4.14:** Looking in different directions, tracking the pupil.

By tuning the parameters parameters below, it is possible to improve the robustness of the Hough Gradient Method. The goal of the tuning has been to reduce the amount of false positives, ensure that the pupil is detected if indeed present, and to make the observed position of the pupil to be as precise as possible.

The function `cv2.HoughCircles()` takes in a grayscale image and applies the Hough Gradient Method to detect circles in the image. The parameters that are tuned in the detection algorithm are:

- First method-specific parameter. This value sets the higher value of the Canny() edge detector threshold, while the lower value will be half of this value. If this is set too high or too low, the edges may not be detected, as the gradient might be completely outside the set threshold.
- Second method-specific parameter. This parameter is the accumulator threshold for detection of circles. The lower it is, the higher chance is for false positives.
- Minimum distance between the centers of the detected circles. If this value is too low, multiple circles may be detected from the same edges, while if it is too large, some circles may not be detected. This value has been set to 1000 pixels, to ensure that no other circles than the pupil are detected.
- Minimum circle radius has been set to 20 to avoid small circles detected in the image's noise and shadows.

- 
- Maximum circle radius has been set to 30 to avoid larger circles being detected, such as the iris or some of the surrounding shadows.

The precision of the method can be improved further by utilizing image manipulations such as blurring and thresholding before applying the Hough Gradient Method. In Figure 4.15 are examples of how the image frame is manipulated using the following code. The last line shows the object "circles", which is a list of tuples in the form of (x,y,r) describing the position of each circle in x and y, as well as radius. In this line, we see the parameters that were tuned, as described above.

```
for f in videostream:
    frame = f.array
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    blur = cv2.GaussianBlur(gray, (radius, radius), 0)
    (minVal, maxVal, minLoc, maxLoc) = cv2.minMaxLoc(blur)
    low = minVal+20
    _, threshold = cv2.threshold(blur, low, 255, cv2.
    THRESH_BINARY)
    circles = cv.HoughCircles(gray, cv2.HOUGH_GRADIENT, 1,
    minDist=1000, param1=40, param2=15, minRadius=15, maxRadius
    =30)
```

The function `cv2.cvtColor()` converts the image to grayscale, before the function `cv2.GaussianBlur()` applies a Gaussian blur to the image. This makes the image flatter by blurring out sharp edges and evening out contrasts, which helps the edge detector to find the correct edges. The sharper the image is, the more false edges can be found. Lines 5 through 7 make a threshold image which is only white and black, with the pupil as a black circle and the rest of the image white. These transformations can be seen in Figure 4.15, and each of the steps improve the accuracy of the detection algorithm. Each of the image transformations have parameters that affect the degree of transformation and thus the detection accuracy.

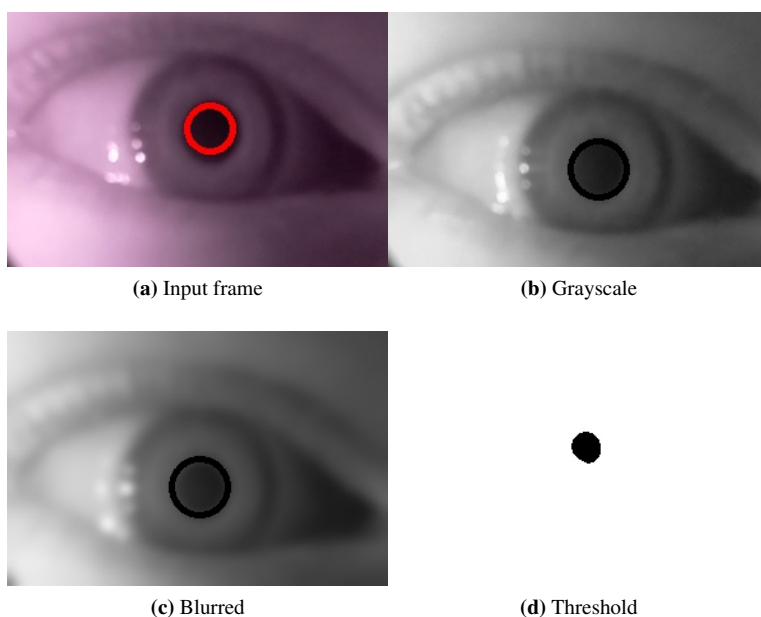
The current image frame, as well as the coordinates of the pupil, can be read from the `update()` loop at any time by the `read` command, ran in `main.py`

```
x, y, x_conv, y_conv, r, frame = pupil.read()
```

The values `x` and `y` are the coordinates of the pupil as detected in the image frame of the eye. `x_conv` and `y_conv` are the same values, translated in order to display the gaze unto the mannequin's view. That is, the coordinates of where the operator is looking. `r` is the radius of the detected pupil, and `frame` is the image frame of the eye.

The coordinates of the center of the pupil, `x` and `y`, are sent to the Arduino by the following lines of code:

```
if self.write:
    x_pos= int(self.x/(self.right-self.left)*255)
    y_pos= int(self.y/(self.upper-self.lower)*255)
    self.ser.write(str.encode('%3d%3d%1d' % (x_pos, y_pos,
    self.blink)))
```



**Figure 4.15:** Manipulating the image of the eye to increase detection accuracy

This section of the code maps the integer values of  $x$  and  $y$  into the arbitrary range of 0-255. This is part of the standardization that allows for modularity in the development process. By having input to Arduino and output from Python standardized to a 0-255 range, it is easy to change either parts of the code, as long as the output or input has this range.

The last line above encodes the integers into padded string values that can be sent via serial communication to the Arduino. Padding the string to 3 digits (“%3d”) means that a 2 digit integer will be preceded by one 0 in the string. This means that if the coordinates of the pupil is (480,200) in a 512 x 600 frame, the converted values will be (239,85). Thus, the encoded string will be “2390850”, assuming that `self.blink` is 0.

The DetectPupil thread is closed by running

```
pupil.stop()
```

### **stream.py**

This script contains a class that is structured the same way as DetectPupil, with functions with the same names: `__init__()`, `update()`, `start()`, `read()` and `stop()`. The main difference is the content of the `__init__()` and `update()` functions. The class is called `stream`, and in `main.py`, the class is initiated with

```
from stream import stream
```

---

```
Environment = stream()
Environment.start()
```

Here, the `__init__()` function in `stream()` defines the necessary values, while the `update()` thread continuously reads the streamed video feed from the Trek AI-Ball camera as mentioned in section 4.3.2. The streamed image can be read at request with the command

```
environment = Environment.read()
```

where "environment" is the image frame while "Environment" is the `stream()` class.

### **calibrate.py**

To be able to override the predefined image cropping variables as they are defined in `main.py`, there is a possibility to enable manual calibration of the image frame with the `calibrate` class that is found in `calibrate.py`. To be able to capture the gaze correctly, the pupil should be positioned approximately in the middle of the image frame, and the whole eye should be visible.

When the calibration is activated (by parsing the argument `-cb 1` as described below), 4 trackbars allow the operator to manipulate the cropping of the frame in such a way that it is possible to position the eye to the requirements above.

### **main.py**

The main script is named `main.py`, and this script imports, initiates and calls the other scripts. The script is initiated through the Terminal on the Raspberry Pi. The script runs an argument parser that allows the user to change the run commands, which is useful for prototyping and troubleshooting. This means that the script can be run without an Arduino connection by running

```
python main.py -a 0
```

or to set another resolution for the Raspi NoIR camera with for example

```
python main.py -x 320 -y 240
```

See Appendix A.1 to see all the available arguments to parse.

The script loops over the code below. The main operations are:

- Read the current image frames in the `DetectPupil()` and `stream()` threads
- Check that the image frames are not empty, or "None"
- Display the image frame
- If the key `q` is pressed, exit the loop and stop the parallel threads.

```
while True:
    x, y, x_conv, y_conv, r, frame = pupil.read()

    if not args["display"]:
```

---

```

environment = Environment.read()
if environment is not None:
    cv2.imshow("Display", environment)

elif args["display"]:
    if frame is not None:
        cv2.circle(frame, (x, y), r, (0, 0, 255), 3)
        cv2.imshow("Display", frame)

if cv2.waitKey(1) & 0xFF == ord('q'):
    pupil.stop()
    if not args["display"]:
        Environment.stop()
    break

```

### 4.4.3 Arduino

The Arduino programming language is a dialect of C and C++ that uses the same syntax and has its own functionality that is specifically designed to work with electromechanical sensors and actuators that connects to the Arduino microcontroller. A script must be compiled and uploaded to the Arduino board, and the most used method for this is to use the Arduino integrated development environment (IDE).

#### TrollBot

To use the nRF24 breakout board, as seen in Figure 4.13, some libraries need to be installed, particularly NRF24.h and NRF24network.h. In addition, a project group at Troll-LABS, NTNU, has recently developed a library for easy implementation of RF communications between Arduinos, using these libraries. Their library is called TrollBot (Trollbot, 2017).

The TrollBot library allows for easy wireless control of actuators and sensors. The library lets the user have one master Arduino to control almost limitless numbers of slave Arduinos. The master Arduino contains all the custom code, and should be the only one where code is modified. The slaves, or nodes, are only coded to respond to the master's requests. In this project, only one slave Arduino is used. The Arduino node's circuit is described in Figure 4.13.

#### Master Arduino code

The complete code can be read in Appendix A.5. The main operations of the code is:

- Read byte signal from Python
- Translate into integers and map the values to the corresponding servo actuation signal
- Send actuation signal to the servos



---

The Arduino code is intended to actuate the servos in response to the observed pupil position, as detected in `DetectPupil.py` in section 4.4.2. In order to calibrate the actuation to respond correctly to every new user that uses it, a dynamic calibration algorithm maps the historical maximum and minimum values as the corresponding maximum and minimum values for the mapping. This means that if the user positions the eye differently inside the image frame, the calibration will use the historical values to define the actuation range. The largest drawback to this method is that the first few movements will be somewhat exaggerated and thereby inaccurate. As the user looks around, the actuation will be more and more calibrated.

In this script, `valx` and `valy` are raw byte values representing the pupil position as read from Python. These values are read from the serial connection to the Raspberry Pi. The code below waits for a serial signal and reads 7 bytes. The byte sequence of 7 bytes is read into 3 strings, `strx`, `stry` and `blinkstr`. These are then converted to integers to be actuated.

```
while (Serial.available() > 0) {  
  
    Serial.readBytes(vals, 7);  
    for (int i = 0; i < 3; i++) {  
        strx += vals[i];  
    }  
    for (int i = 3; i < 6; i++) {  
        stry += vals[i];  
    }  
    blinkstr += vals[6];  
  
    // Converting strings to ints. Range: 0-255  
    int valx = strx.toInt();  
    int valy = stry.toInt();  
    int blinking = blinkstr.toInt();
```

Below, the calibration and mapping is shown. The historical maximum and minimum values, `xmax`, `xmin`, `ymin` and `ymin`, are defined if the new value of `x` and `y` are higher than the historical maximum. To filter out noise or falsely positive detections, the new value is unable to be too large, as restricted by the if-statements. EV means Eyes Vertical, EH means Eyes Horizontal, LL means Lids Lower, LU means Lids Upper. The mapped values are actuation values that are relative to the maximum and minimum historical values.

```
// Dynamic calibration  
if (valx > xmax && valx/xmax < 1 + (255-xmax)/xmax) {  
    xmax = valx;  
}  
if (valy > ymax && valx/xmax < 1 + (255-xmax)/xmax) {  
    ymax = valy;  
}  
if (valx < xmin && valy/xmin > (255-(255-xmin))/(255-xmin))  
{  
    xmin = valx;
```

---

```
}
if (valy < ymin && valy/xmin > (255-(255-xmin))/(255-xmin))
{
    ymin = valy;
}

// Map calibrated values to servo values
int EVval = map(valy, ymin, ymax, EVmin, EVmax);
int EHval = map(valx, xmin, xmax, EHmax, EHmin);
int LUval = map(valy, ymin, ymax, LUmax, LUmin);
int LLval = map(valy, ymin, ymax, LLmin, LLmax);
```

In addition to controlling the gaze direction of the mechanical eyes, the script also controls blinking. Blinking occurs when no pupil is detected in `DetectPupil`. The actuation of a blink is performed by closing the eyelids when an absence of pupil is detected. The gaze direction will not change, something that makes the blinking look natural and realistic.

The eyelids are also actuated when they are not blinking. The actuation is performed in such a way that the eyelids cover the edge of the iris as the eyeballs move with the changing gaze. This is similar to how the eyelids move in humans.

## 4.5 Testing and feedback

The different prototypes have been tested on people in most of the development stages. The alpha prototype described in this chapter has been tested as well. These tests have shown that the prototype is able to gain eye contact, follow people and objects and to blink in such a way that people have reported that it looks almost completely natural. The feedback of the latest prototypes have not mentioned eeriness or unease, as is a common way to describe feelings associated with the Uncanny Valley (Ishiguro, 2007).

A typical test setup has been to place the eyes on the table and interact with people remotely. The eyes have also been put inside the mannequin's head and people could interact with it. The natural movement and blinking of the alpha prototype cause the observers to not think about the fact that the eyes are mechanical. In addition to this, the realism of the eyes, which is a result of using high-end special effect eyes, makes the feedback be mainly about behavior and movements. Most negative feedback has been about bugs in the control system that have occurred during prototyping and testing, such as suddenly jittering violently or similar.

Further on, the servo motors can be quite noisy when doing rapid actuation, such as when blinking, which has led to some feedback on the noise level of the prototype. The noise is reduced when the rig is installed into the mannequins head.

# Discussion

This chapter reviews the system as it is, and how it may be improved or developed further. The methodology of the project is reviewed in a perspective of the methods presented and performed and their effectiveness and efficiency.

## 5.1 System review

The prototype that is presented in this thesis is an alpha prototype, as described in section 2.2.1. This means that the prototype shows a proof of concept, fulfilling some of the needs that were discovered in the early stages of the development. Some features have intentionally been kept at a low resolution, as they are not important to the function of the product. One example is the laser cut case for the processing and wire arrangement in the head mounted device. This case is heavy, big and not optimized in other aspects than the critical functions of stability and cooling. These are things that will not affect the performance of the device, although it may affect the comfort of the operator.

This product shows the potential of what can be achieved with small investments into the technologies of the eyes. The alpha prototype shows that interactions between mannequin and medical personnel can be improved with the use of mechanical eyes and operator control of the mannequin. Simulations were observed to lack natural communication and empathy, and this product shows how these challenges may be overcome. The product has the potential to improve health care simulations by letting medical personnel practice diagnostics on the eyes as well as interactions with patients that are awake and aware. Additionally, this can solve concrete problems that were observed in the simulations such as the mannequin being awake, but participants do not notice, or similar.

This thesis had the scope of developing, building, and refining a conceptual prototype, as described in section 3.1, of a smooth actuation and control system of the human eyes in a healthcare simulation mannequin with the aim to achieve perceived human like eye movement, behavior and interaction.

The mechanical actuation has been solved to a high degree, as it delivers high precision and speed. The servos can be controlled accurately, and they can replicate movements

---

rapidly. The biggest drawback of the actuators is noise, but this has been overlooked to some degree in this thesis. Solutions for reducing noise were investigated in Appendix B (Nygaard, 2016).

The electrical system was created to accommodate modularity in the development process by standardizing module interfaces. Power management and signal transmission has been a large factor in the electrical design of the prototype.

The mechanical structure of the prototypes are primarily made in laser cut MDF or other materials that allow for rapid prototype generation and quick iterations. MDF has the added positive aspect of being robust while being easily modified.

The Python scripts have been split into several modules to allow threading on the Raspberry Pi. This allows the processes to run in parallel to achieve more computation speed. This also accommodates the modularity that has been discussed. The Arduino control system allows for wireless transmission and for rapid actuation while being easily modifiable.

All in all, the system fulfills the thesis' scope to a high degree, and delivers a robust, efficient prototype system that may be able to give healthcare simulations a new aspect of interaction between medical personnel and mannequin.

## 5.2 System future

How can development of this prototype continue in the future? The prototype proves that it is possible to create human like movement of the eyes of the mannequins, and to increase the degree of non-verbal communication that occurs in simulations. This may lead to a higher degree of reported empathy towards the mannequins, and may also provide a tool for diagnostics.

It is possible to refine the system into an implementable module. The control system could be improved by the implementation of diseases of the eyes. The system could be made compatible with the existing system of Lærdal's mannequins, and the head mounted device could be made into a more efficient product.

The system is created as a low-cost prototype with sub-optimal capabilities in terms of processing, and it could be improved by creating an integrated processing unit which does not require an operating system or user control. There are several possibilities for future development of this prototype. It might be possible to further develop the software to allow for the development of artificial intelligence.

### 5.2.1 Artificial intelligence

Artificial intelligence (AI) is a term that describes computational tools that use advanced statistics to predict outcome or expected behavior based on big data sets of observed inputs. Neural networks are AI methods that are "black box" methods that tries to learn behavior from given input and categorize this behavior.

CAFFE (Jia et al., 2014) is a deep learning framework that allows learning of visuo-motor control from video streams. For example, a robot can learn how to perform simple tasks, like moving an object from A to B purely by observing a video stream of a person moving this object. Similarly, it would be possible to define parameters for training

---

behavior of the eyes based on inputs from the video stream. As described in chapter 4, the control system output is a 7 digit byte sequence that stems from an image input. This image can be manipulated and objects can be detected in the image. This means that it would be possible to train a control system with the use of neural network AI to react to image inputs such as the actual image, faces detected in the image, objects detected (finger held up, flashlight into the eyes, etc.) or sounds.

If the AI system could be trained to behave like a human would, in the situations it was trained to face, the control of the eyes could be decoupled from the head mounted device, and the operator could continue operating the simulations like they do today, only through the simulation software.

## 5.2.2 Ideas for future development

If the system is further developed, there are several areas where improvement can be made. The most prominent is how the product should be used in a simulation setting. In the case of AI, a control system would only require control input from the simulation software to change the behavior. These alterations to the behavior could be sluggish behavior, inability to focus on objects or similar symptoms.

With no AI, the system would demand operator control, as it is designed for today. This would also require incorporation into the existing simulation software, so the operator could choose to activate the headset and take control over the eyes. It might be possible to implement idle movements that the mannequin could make when the operator does not take control over the eyes, or the eyes could default back to the functions that the eyes have in mannequins today. This could be behavior like looking straight forward while being able to blink.

Further development into streamlining the head mounted device would lead to re-designing it for weight and comfort. Major improvements could be achieved by changing the design into a custom made case, optimizing electronics in terms of size and weight and processing capabilities.

It is possible to use other gaze tracking methods to allow the operator to have a display on both eyes. Some alternative tracking methods were discussed in section 2.3.2, such as infrared reflection of the cornea or electrodes measuring the electric potential differences in the skin around the eyes. The implementation of other methods would allow a more pleasant viewing experience with a screen on both eyes. Some virtual reality headsets have implemented eye tracking with other methods.

The viewing experience could further be improved simply by upgrading the hardware of the display module. Investing in a display with a higher resolution and refresh rate would provide smoother observed motions and less strain on the eyes.

For implementation of vergence actuation and for a better one to one representation of the eyes of the operator, it might be necessary to implement a two-camera detection method where the cameras could be placed below the screen. The implementation of such a system could provide more diagnostic tools for the simulations.

For vergence to be accurately captured, a 3D environment would have to be displayed to the operator. This could be done by displaying a different video stream to both eyes. This might be achievable by having two cameras. It might be possible to have a camera inside each of the eyes, but not like in section 3.2.4, where the camera moves with the

---

eye. One idea was suggested where a stationary camera was mounted inside of the eye in such a way that the eye can move independently of the camera, and the camera provides a stationary field of view. This might be achievable by using a one way, perforated film that the camera can see through, but from the outside it looks like an eye. Another solution could be to have cameras installed in glasses that the mannequin could wear. Otherwise it might be possible to have the cameras mounted near the eyes in some way, even though this might affect the appearance of the mannequin.

### **Improving eye detection and actuation**

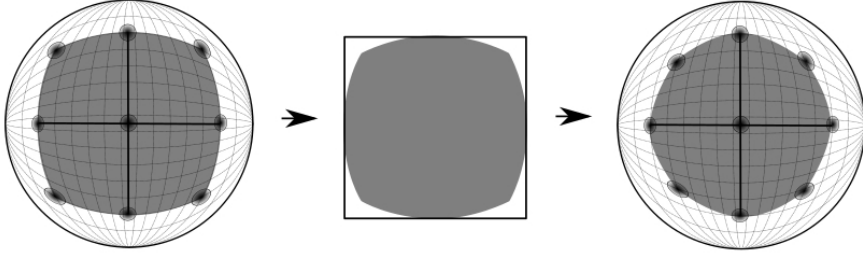
As mentioned in section 2.3.2, the gaze direction can be accurately calculated by transforming the two-dimensional coordinates of the pupil detection algorithm into spherical angles. This is useful when trying to determine where in space someone is looking, either in a 3D-space and tracking both eyes, or on a 2D screen. Gaze direction tracking is not used in this project, as the gaze can be actuated by assuming that the coordinates extracted from the pupil detection algorithm can be transformed linearly into actuation of the servos. This assumption can be used for an approximations as long as the camera that is used for detection is placed directly in front of the eye and as long as the operator's visual attention is concentrated roughly in the center of the display.

When using this technique, the actuation will be more and more inaccurate as the operator looks towards the corners of the display, as explained by Figure 5.1. From left to right in the figure, the approximation error can be explained:

1. Left: The movement space of the operator's pupil as observed by the camera, as the operator looks upon the screen, is not rectangular, but a four sided area of a sphere where the corners are 90 degrees.
2. Middle: The actuation control system on the Arduino represents the observed space as a Cartesian space with maximum horizontal and vertical movements in a rectangular shape. As observed, the control space does not accurately represent the movement space of the pupil.
3. Right: The actuated movement, based on the Cartesian control space. As seen here, the corners are not actuated accurately.

As explained above, the 2D coordinates that represent the detected pupil, are obtained through the detection algorithm, but needs to be mapped to overcome the curvature of the eye. Figure 5.1 shows how the curvature distorts the coordinates that are detected by the eye tracking. This phenomenon can be observed in Figure 5.2. To the camera, the path that the pupil follows will be shaped like an ellipse when following a line on the display due to how the gaze plane intersection with the sphere of the eyeball is projected to the camera.

The elliptical path distortion can be overcome by applying the following transformation of observed coordinates  $x'$  and  $y'$  into actuation coordinates  $X$  and  $Y$ . The equations are derived from the geometry described in Figure 5.3. The radius of a human eyeball averages around 12 mm, which is needed for the calculations.  $X$  and  $Y$  needs to be represented by  $x'$  and  $y'$ .



**Figure 5.1:** Representation of the approximation error in the pupil actuation control system.

$$(x', y') \rightarrow (X, Y)$$

The formula of the ellipse that the pupil will follow (Figure 5.3) when looking at an arbitrary horizontal line is described as

$$\frac{x'^2}{12mm^2} + \frac{y'^2}{Y^2} = 1$$

which can be solved for Y

$$Y = \sqrt{\frac{y'^2}{(1 - \frac{x'^2}{12mm^2})}}$$

and by the formula

$$\frac{x'}{y'} = \frac{X}{Y}$$

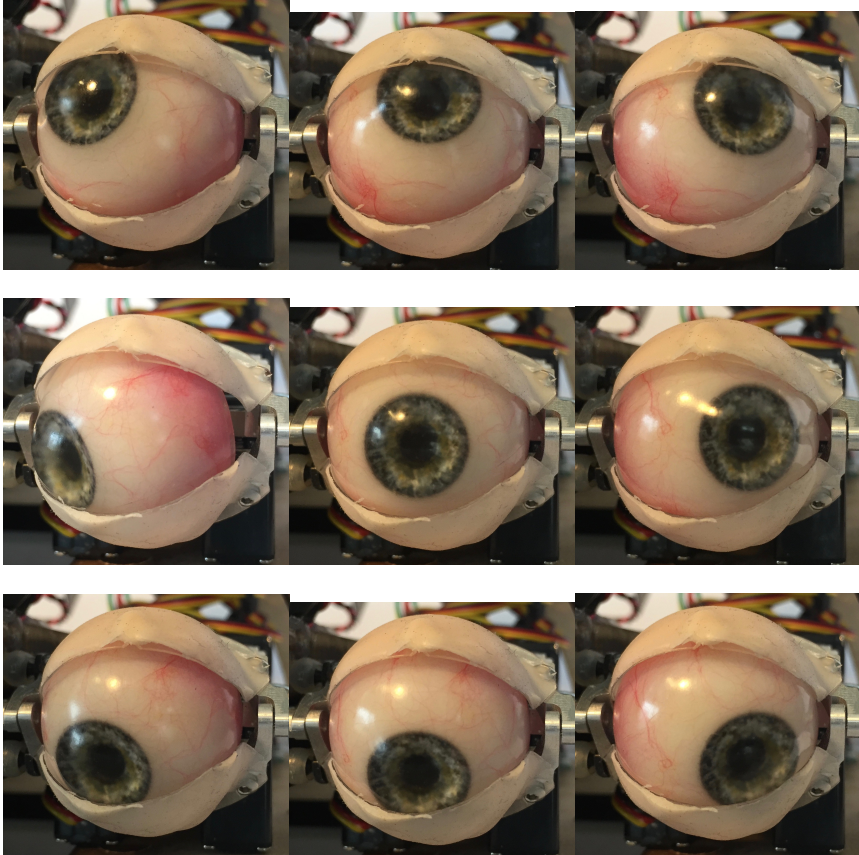
the corrected actuation coordinates can be found:

$$(X, Y) = \left( \frac{x'}{y'} * \sqrt{\frac{y'^2}{(1 - \frac{x'^2}{12mm^2})}}, \sqrt{\frac{y'^2}{(1 - \frac{x'^2}{12mm^2})}} \right) \quad (5.1)$$

The above equation is only valid if  $x'$  is smaller than  $y'$ . If this is not the case, the following equation is derived similarly, but with ellipses with a vertical main axis.

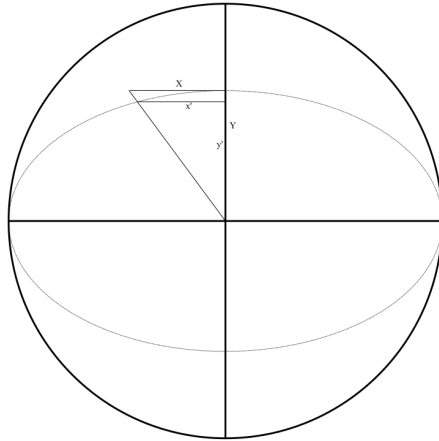
$$(X, Y) = \left( \sqrt{\frac{x'^2}{(1 - \frac{y'^2}{12mm^2})}}, \frac{y'}{x'} * \sqrt{\frac{x'^2}{(1 - \frac{y'^2}{12mm^2})}} \right) \quad (5.2)$$

This method is currently not implemented, but could be applied in the Arduino code. The discrepancy was discovered late in the process, but feedback has not shown that the inaccuracy was noticeable.

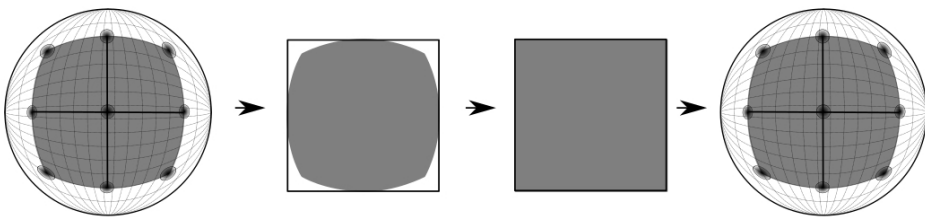


**Figure 5.2:** The nine extreme positions of the actuated eye.

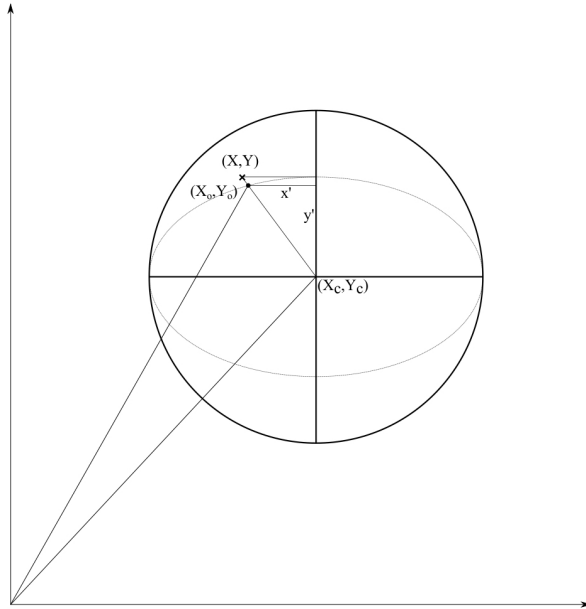




**Figure 5.3:** How the elliptical path of the pupil can be used to calculate the corrected actuation coordinates.



**Figure 5.4:** How the approximation error is corrected by equation (5.1) and (5.2)



**Figure 5.5:** Global and local coordinates

Equations (5.1) and (5.2) are not generalized to handle if the eye is placed in an arbitrary position in a coordinate system, but defined when the center of the eye ball is in (0,0).

It is possible to position Figure 5.3 as a local coordinate system in a global coordinate system, if the center position of the eye ball is known, as seen in Figure 5.5. To generalize for all cases, equation (5.3) can be derived from positioning the local coordinates  $(x', y')$  in the global coordinate system. Below, it is assumed that eye ball center is known, and referred to as  $C$ , and  $(X, Y)$  are the calibrated coordinates relative to global origin. We define the raw observed coordinates relative to origin, as  $(X_o, Y_o)$ .

$$\begin{aligned}
 O &= (X_o, Y_o) \\
 C &= (X_c, Y_c) \\
 (X, Y) &= C + f(x', y') \\
 f(x', y') &= \begin{cases} \text{Equation (5.1)} & \text{if } x' < y' \\ \text{Equation (5.2)} & \text{if } x' > y' \end{cases} \\
 (x', y') &= (X_o - X_c, Y_o - Y_c)
 \end{aligned}$$

---


$$(X, Y) = \begin{cases} \text{if } X_o - X_c < Y_o - Y_c: \\ (X_c, Y_c) + \left( \frac{X_o - X_c}{Y_o - Y_c} * \sqrt{\frac{(Y_o - Y_c)^2}{1 - \frac{(X_o - X_c)^2}{12mm^2}}}, \sqrt{\frac{(Y_o - Y_c)^2}{1 - \frac{(X_o - X_c)^2}{12mm^2}}} \right) \\ \text{if } X_o - X_c > Y_o - Y_c: \\ (X_c, Y_c) + \left( \sqrt{\frac{(X_o - X_c)^2}{1 - \frac{(Y_o - Y_c)^2}{12mm^2}}}, \frac{Y_o - Y_c}{X_o - X_c} * \sqrt{\frac{(X_o - X_c)^2}{1 - \frac{(Y_o - Y_c)^2}{12mm^2}}} \right) \end{cases} \quad (5.3)$$

Equation (5.3) can be implemented into the Arduino code as is.  $(X_o, Y_o)$  are the raw byte values sent from Python to the Arduino. These values can also be used to approximate  $(X_c, Y_c)$  over time, as mentioned in section 4.4.3. This means that  $(X, Y)$  can be derived from only  $(X_o, Y_o)$ , using the assumptions above.

To find the center of the eye ball, it is possible to use similar dynamic calibration method as is currently implemented in the Arduino code for calibration. With the camera positioned straight in front of the eye, it may be assumed that the observed extreme values of vertical and horizontal eye movement are symmetrical in relation to C. Thus, C can be approximated by averaging the historical maximum and minimum values of  $X_o$  and  $Y_o$ . This gives us  $(X_c, Y_c)$  and by observing  $(X_o, Y_o)$ . It is possible to accurately actuate the servos with the values  $(X, Y)$  by implementing Equation (5.3) in the actuation control.

Further accuracy in the actuation could be achieved by implementing a two camera setup, as described in section 5.2.2, where each eye has an embedded stationary camera. This could allow the operator to actually look through the eyes of the mannequin, and each eye could have been controlled by one of the operator's eyes. This would allow for actuation of vergence (focal point shifts), blinking with one eye and more.

---

## 5.3 Methodology review

The development method that has been utilized in this thesis has been inspired by the concept of probing in the Wayfaring Model (Gerstenberg et al., 2015), as well as concepts from set-based design (Smith, 2007), as explained in section 3. These exploratory early phase tools have been used in a modular development strategy that has allowed for rapid paradigmatic shifts in each of the modules, while not sacrificing the functionalities of the rest of the system. This has created a high efficiency in the development, and it lessens rework.

This modular approach has been possible due to the complexity of the system that was to be developed. The project has consisted in programming, mechanical actuation and electronics, and all the subsystems, or modules, have required some kind of interface. These interfaces have been electrical, mechanical and digital. Separating the project into modules that allow radical changes while retaining the same interface, has expanded the ability to test rapidly and gain feedback on individual modules.

This modular independence is best described with an example: when testing the control system for the actuation, instead of building the actuator as it was thought to be, a simpler module could be employed. This simpler module was the eye controlled robot in section 3.2.3, which had all the same interfaces, while being a much simpler design. Thus, it was possible to get a general response to the control system output instead of a specialized response that required more development.

Similarly, when testing actuators for viability in terms of smoothness of motion, instead of running the incomplete pupil tracking software, a much simpler facial recognition program was sufficient to create the same control signals. This allowed for development in parallel of both control system of the actuators and the mechanical design of the actuators.

### 5.3.1 Development as one person opposed to in teams

This project was performed by one person. Although the community in and around Troll-LABS is helpful and available, the project and its work has been performed mainly by the author. This has lead to some reflections about working with a product development project as one person as opposed to working in teams, when applying team based methods such as wayfaring, agile, set-based and other techniques mentioned in the thesis.

Working as one person as opposed to working in teams affects the development process in several ways. The effects have been observed to be mainly detrimental to the efficiency of the development process. This is thought, by the author, to be due to unknown unknowns and how they may be discovered through Generative Design Questions (GDQs) and Deep Reasoning Questions (DRQs) (Eris, 2003, 2004) in team-work. GDQs are essential generating concepts in teams, as they are open ended questions that are looking for unknown answers or unknown questions. These unknowns may remain unknown longer when externals are not included in the development process, both to answer the GDQs and to pose their own questions.

This is also true when it comes to DRQs and their ability to exclude non-viable ideas based on answers that are enlightening to unknown aspects of the product and its use. This means that a team-member may be able to see some restrictions about the prototype that might make in non-viable for the requirements, that you do not see yourself.

---

However, the mindset of using these types of questions might still be beneficial in a solo-development setting. By being aware of how GDQs create openness towards new ideas, and how they open up the solution space, it may be possible to create more ideas than if one is not aware of them. The same can be said for DRQs and how they may help to converge the solution space.

Personal factors, such as general and specialist knowledge will affect the efficiency of the development, as new subjects will have to be explored when entering a new area of development. A diverse team will have a broader knowledge base than one developer.

Sjøvold (2006) describes how members of an efficient team will continuously change roles in the team. In a solo-development process, the developer will not be able to fill several roles at the same time, to create a similar dynamic. Further on, people of different disciplines may have different ways of solving problems and understanding challenges, which can help development in multidisciplinary teams.

One positive side to developing alone could be that the modular approach that this project has employed may be harder to utilize when working in teams. The knowledge that is required about each subsystem in order to be able to change the subsystem radically while retaining the interface, might be harder to communicate and coordinate across teams, and especially so across multidisciplinary teams. This could be easier when the developer has to accumulate specialist knowledge about each module. Of course this requires a developer with the ability and available time to be able to learn each subject matter thoroughly.

### **5.3.2 Trial and error or wayfaring?**

Trial and error is the process of trying to accomplish goals by trial, or in the terms of product development, by prototyping. When conducting trial and error development, a goal has been set in terms of functions that should be fulfilled, and this goal is pursued in a manner that consists of finding a solution, make the prototype, test the prototype, and abduct knowledge from the trial. After this, you reconsider the design and you either probe further, or shift concept.

What, then, separates trial and error from Wayfaring and the design-build-test cycle that it explains? Mainly, the difference comes in the form of diverging and in the amount of ideas generated and investigated. With both mindset and methods for generating a large amount of ideas, or even better, tangible prototypes, wayfaring tries to uncover a large solution space and probe into the solutions that seem most viable. Being agile in the development and delaying decisions of which concept to choose allows that more ideas can be probed in design-build-test cycles. This will create more abductive learning, and more unknown unknowns may be uncovered.

In conclusion, trial and error is the default way of problem solving, which has proved an effective method as long as problem solving has been done. Wayfaring is a mindset that encompasses methods for creativity, divergence, agility in development and probing into ideas or concepts. Wayfaring tries to manage product development projects where critical knowledge may yet be uncovered, where the uncertainty is large and the reward of success is high. The methods for trying to manage such projects need to be agile and ready for change, and they need to try to capture as much knowledge as possible to overcome the uncertainty of the fuzzy front end of product development.

---

## Conclusion

This master's thesis describes the development of eyes for healthcare simulator mannequins, that can be used in medical simulations. The eyes were developed to overcome an observed lack of emotional connection and non-verbal communication between medical personnel and the simulation mannequin. Gaze behavior and eye contact between medical personnel and patient has been found to impact patient-centeredness and medical personnel's awareness of a patient's psychological distress and cognitive functioning (D'Agostino and Bylund, 2014).

Discussion of the current academic standings in topics such as early stage product development methodologies, agile methods in enterprises, humanoid robots and eye movement and behavior constitutes the academic background of the product. The technological background of the product is discussed in terms of Lærdal Medical's current products' abilities. Technical possibilities are explored by benchmarking and investigation of the newest discoveries and technological advances.

This thesis describes the development process of the alpha prototype as described in Chapter 4, which details the product as it is in the last prototyping iteration. The methods used, their efficiency and effectiveness, have been reviewed and investigated.

In conclusion of this master's thesis, the alpha prototype proves that the concept of human control of a mannequin's eyes is possible, and demonstrates one way that this is achievable. The thesis itself presents alternatives to the current solution, as well as possibilities for further development and improvements. The current prototype can be characterized as a low-cost human-machine interface system to bridge the Uncanny Valley, which can allow improvements in health care simulations by introducing nonverbal communication and diagnostic tools related to the eyes.

This thesis details possibilities for nonverbal communication and diagnosing diseases based on eye movement and behavior. This allows for a new dimension of diagnostic possibilities, as both psychological and physical trauma can affect the behavior of the eyes, and certain diagnoses can create apathy and non-responsiveness while the patient is awake. In today's mannequin, this is not possible to simulate, but the presented system creates an opening for implementation of such symptoms and diagnostics.

---

The product described here is still a prototype, but it provides insights into several topics and gives an idea of how to solve this particular challenge. In addition to this, the thesis reviews the methods used, presents the results and discusses the efficiency and effectiveness of the methods. The primary conclusions about the methods used for this early stage product development project is that developing a complex system as one person requires adaptation and learning quickly to be able to develop in diverse subjects. This resonates with the idea of agility in development projects, and is further supported by Wayfaring Model. Further on, applying modularity in the development to overcome the complexity of the system, grants the developer a higher understanding of the project. In addition, it provides a flexibility in the development process as it lets modules be completely re-designed without having to redesign the complete system.

To sum up, this project describes the development of an alpha prototype of a system that may improve healthcare simulations by allowing nonverbal communications and eye movement diagnostics.



# Bibliography

- Baldwin, C. Y., Clark, K. B., 2000. Design rules: The power of modularity. Vol. 1. MIT press.
- Bradski, G., 2000. OpenCV. Dr. Dobb's Journal of Software Tools.
- Cohen, D., Lindvall, M., Costa, P., 2003. Agile software development. DACS SOAR Report 11.
- D'Agostino, T. A., Bylund, C. L., 2014. Nonverbal accommodation in health care communication. *Health communication* 29 (6), 563–573.
- Duchowski, A., 2007. Eye Tracking Techniques. In: *Eye Tracking Methodology*. Springer London, pp. 51–59, doi: 10.1007/978-1-84628-609-4\_5.  
URL [http://link.springer.com/chapter/10.1007/978-1-84628-609-4\\_5](http://link.springer.com/chapter/10.1007/978-1-84628-609-4_5)
- Edelman, J. A., Leifer, L., Banerjee, B., Sonalkar, N., Jung, M., Lande, M., 2009. Hidden in plain sight: affordances of shared models in team based design. In: *DS 58-2: Proceedings of ICED 09, the 17th International Conference on Engineering Design*, Vol. 2, Design Theory and Research Methodology, Palo Alto, CA, USA, 24.-27.08. 2009.
- Eris, Ö., 2003. Asking generative design questions: a fundamental cognitive mechanism in design thinking. In: *DS 31: Proceedings of ICED 03, the 14th International Conference on Engineering Design*, Stockholm.
- Eris, O., 2004. Effective inquiry for innovative engineering design. Vol. 10. Springer Science & Business Media.
- Friesike, S., Gassmann, O., 2014. Leveraging creativity. In: *Management of the Fuzzy Front End of Innovation*. Springer, pp. 159–177.
- Gassmann, O., Schweitzer, F., 2014. Managing the Unmanageable: The Fuzzy Front End of Innovation. Springer International Publishing, Cham, pp. 3–14.  
URL [http://dx.doi.org/10.1007/978-3-319-01056-4\\_1](http://dx.doi.org/10.1007/978-3-319-01056-4_1)

- 
- Gerstenberg, A., Sjöman, H., Reime, T., Abrahamsson, P., Steinert, M., 2015. A simultaneous, multidisciplinary development and design journey—reflections on prototyping. In: *International Conference on Entertainment Computing*. Springer, pp. 409–416.
- Hall, J. A., Knapp, M. L. (Eds.), 2013. Nonverbal communication. No. 2 in *Handbooks of communication science*. De Gruyter Mouton, Berlin ; Boston.
- Heck, J., Rittiner, F., Steinert, M., Meboldt, M., 2016. Iteration-based performance measurement in the fuzzy front end of pmps. *Procedia CIRP* 50, 14–19.
- Ishiguro, H., Jan. 2007. Scientific Issues Concerning Androids. *The International Journal of Robotics Research* 26 (1), 105–117.  
URL <http://ijr.sagepub.com/content/26/1/105>
- Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., Darrell, T., 2014. Caffe: Convolutional architecture for fast feature embedding. arXiv preprint arXiv:1408.5093Deep visuomotor control: [https://docs.google.com/presentation/d/1UeKXVgRvvxg9OUdh\\_UiC5G71UMscNPlvArsWER41PsU/edit#slide=id.g11018250fb\\_1055\\_0](https://docs.google.com/presentation/d/1UeKXVgRvvxg9OUdh_UiC5G71UMscNPlvArsWER41PsU/edit#slide=id.g11018250fb_1055_0).
- Kriesi, C., Blindheim, J., Bjelland, Ø., Steinert, M., 2016. Creating dynamic requirements through iteratively prototyping critical functionalities. *Procedia CIRP* 50, 790–795.
- Leifer, L. J., Steinert, M., 2011. Dancing with ambiguity: Causality behavior, design thinking, and triple-loop-learning. *Information Knowledge Systems Management* 10 (1-4), 151–173.
- Lim, Y.-K., Stolterman, E., Tenenberg, J., 2008. The anatomy of prototypes: Prototypes as filters, prototypes as manifestations of design ideas. *ACM Transactions on Computer-Human Interaction (TOCHI)* 15 (2), 7.
- Nygaard, T. H., 2016. Technologies to control the eyes of patient simulators, appendix B, Project thesis.
- OpenCV, 2017. Feature detection. Accessed: 2017-06-04.  
URL [http://docs.opencv.org/2.4/modules/imgproc/doc/feature\\_detection.html?highlight=houghcircles#houghcircles](http://docs.opencv.org/2.4/modules/imgproc/doc/feature_detection.html?highlight=houghcircles#houghcircles)
- Rossum, G., 1995. Python reference manual. Tech. rep., Amsterdam, The Netherlands, The Netherlands.
- Sjøvold, E., 2006. Teamet: Utvikling, effektivitet og endring i grupper. Universitetsforlaget.
- Smith, P. G., 2007. Flexible product development: building agility for changing markets. John Wiley & Sons.
- Smith, P. G., Reinertsen, D. G., 1992. Shortening the product development cycle. *Research-Technology Management* 35 (3), 44–49.

- 
- Sobek, D. K., Ward, A. C., Liker, J. K., 1999. Toyota's principles of set-based concurrent engineering. *Sloan management review* 40 (2), 67.
- Steinert, Martin, Leifer, Larry J., 2012. Finding Ones Way: Re-Discovering a Hunter-Gatherer Model based on Wayfaring. *International Journal of Engineering Education* 28 (2), 251–253.
- Stutcliffe, A., Sawyer, P., 2013. Requirements elicitation: Towards the unknown unknowns. In: *Requirements Engineering Conference (RE), 2013 21st IEEE International*. IEEE, pp. 92–104.
- Thomke, S., Reinertsen, D., 1998. Agile product development: Managing development flexibility in uncertain environments. *California management review* 41 (1), 8–30.
- Trollbot, Jun. 2017. Trollbot.h.  
URL <https://www.ntnu.no/wiki/display/TrollLABS/TrollBOT>,  
<https://github.com/trollllabs/2017-trollbot>
- Walton, M., Oct. 2016. PSVR vs. HTC Vive vs. Oculus Rift vs. Gear VR: Which VR headset should you buy?  
URL <http://arstechnica.com/gaming/2016/10/best-vr-headset-2016-psvr-rift-vive/>
- Wilczynski, V., 2015. Academic maker spaces and engineering design. In: *American Society for Engineering Education*. Vol. 26. p. 1.

---

---

Appendix **A**

Code

---

## A.1 main.py

```
1 import argparse, time, cv2
2 from DetectPupil import DetectPupil
3 from calibrate import calibrate
4
5 #construct the argument parser and parse the arguments
6 ap = argparse.ArgumentParser()
7 ap.add_argument("-x", "--resx",           type=int, default=640,
8               help="resolution in x")
9 ap.add_argument("-y", "--resy",           type=int, default=480,
10              help="resolution in x")
11 ap.add_argument("-a", "--arduino",        type=int, default=1,
12              help="connect to Arduino? 1/0")
13 ap.add_argument("-cb", "--calibrate",     type=int, default=0,
14              help="calibrate? 0/1")
15 ap.add_argument("-d", "--display",        type=int, default=0,
16              help="display? eye = 1, environment = 0")
17 ap.add_argument("-n", "--num",           type=int, default=1,
18              help="The first picture has this number when taking snapshots. ")
19 args = vars(ap.parse_args())
20
21 num = args["num"]
22
23
24 # _____Calibration _____
25
26 left = 250
27 right = 500
28 upper = 300
29 lower = 100
30
31 #Override the above values if -cb = 1
32 if args["calibrate"] == 1:
33     c = calibrate(left, right, lower, upper)
34     c.start()
35     left, right, lower, upper = c.calibrate()
36     c.stop()
37
38 # _____Initializing_____
39
40 #Pupil detection
41 time.sleep(0.2)
42 pupil = DetectPupil((args["resx"], args["resy"]), 32, args["arduino"], left,
43                   right, lower, upper)
44 pupil.start()
45 time.sleep(0.2)
46
47 #Display positioning
48 cv2.namedWindow("Display", cv2.WINDOW_NORMAL)
49 cv2.moveWindow("Display", 0, 0)
50 cv2.resizeWindow("Display", 512, args["resy"] * (1024/2) / args["resx"])
51
52 #Environment display
53 if not args["display"]:
54     from stream import stream
55     Environment = stream()
56     Environment.start()
57
58 # _____Running_____
59
60 while True:
61     #Read position and size of pupil, and the current frame.
62     #(x,y) is for "frame", (x_conv,y_conv) is for environment.
63     x, y, x_conv, y_conv, r, frame = pupil.read()
64
```

---

```
65 #Display the stream image if not otherwise specified
66     if not args["display"]:
67         environment = Environment.read()
68         if environment is not None:
69             cv2.moveWindow("Display",0,0)
70             cv2.imshow("Display", environment)
71
72 #Show the eye if -d = 1
73     elif args["display"]:
74         if frame is not None:
75             cv2.circle(frame, (x,y), r, (0,0,255), 3)
76             cv2.imshow("Display", frame)
77
78 #Stop the threads and exit the loop if "q" is pressed
79     if cv2.waitKey(1) & 0xFF == ord('q'):
80         pupil.stop()
81         if not args["display"]:
82             Environment.stop()
83         break
84
85 #If "a" is pressed, take a snapshot of the displayed image.
86     if cv2.waitKey(33) & 0xFF == ord('a'):
87         if not args["display"]:
88             cv2.imwrite("/home/pi/images/environment%d.jpg" % num
89             ,environment)
90         elif args["display"]:
91             cv2.imwrite("/home/pi/images/eye%d.jpg" % num,frame)
92         num = num + 1
93 cv2.destroyAllWindows()
```

---

## A.2 DetectPupil.py

```
1 from picamera.array import PiRGBArray
2 from picamera import PiCamera
3 from threading import Thread
4 import cv2, serial
5
6 class DetectPupil:
7     def __init__(self, resolution, framerate,arduino,left, right, lower,
8         upper):
9         self.arduino = arduino
10        self.r = 7
11        self.camera = PiCamera()
12        self.camera.resolution = resolution
13        self.camera.framerate = framerate
14        self.rawCapture = PiRGBArray(self.camera, size=resolution)
15        self.stream = self.camera.capture_continuous(self.rawCapture, format="
16        bgr", use_video_port=True)
17        self.resx,self.resy=resolution
18        self.upper = upper
19        self.lower = lower
20        self.left = left
21        self.right = right
22        self.frame = None
23        self.x = 0
24        self.y = 0
25        self.x_conv = 0
26        self.y_conv = 0
27        self.radius = 0
28        self.stopped = False
29        self.blink = 1
30        if self.arduino:
31            self.write = True
32            self.connected = False
33            self.ser = serial.Serial("/dev/ttyACM0",9600,rtscts=1)
34            while not self.connected:
35                serin = self.ser.read()
36                self.connected = True
37        else:
38            self.write = False
39
40    def start(self):
41        Thread(target=self.update, args=()).start()
42        return self
43
44    def update(self):
45        for f in self.stream:
46            frame = f.array
47            frame = cv2.flip(frame,0)
48            frame = frame[self.lower:self.upper,self.left:self.right]
49            gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
50            gray = cv2.GaussianBlur(gray, (self.r, self.r), 0)
51            circles = cv2.HoughCircles(gray,cv2.HOUGH_GRADIENT,1,1000,param1=20,
52            param2=15,minRadius=20,maxRadius=30)
53
54            if circles is not None:
55                self.blink = 0
56                for i in circles[0,:]:
57                    self.x = i[0]
58                    self.y = i[1]
59                    self.x_conv = int(i[0]*640/(self.upper-self.lower))
60                    self.y_conv = int(i[1]*480/(self.right-self.left))
61                    self.radius = i[2]
62                    if self.write:
63                        x_pos= int(self.x/(self.right-self.left)*255)
64                        y_pos= int(self.y/(self.upper-self.lower)*255)
65                        self.ser.write(str.encode('%3d%3d%1d' % (x_pos,y_pos,self.
```



---

```
        blink)))
63     elif circles is None:
64         self.blink = 1
65         if self.write:
66             x_pos= int(self.x/(self.right-self
        .left)*255)
67             y_pos= int(self.y/(self.upper-self.lower)*255)
68             self.ser.write(str.encode('%3d%3d%1d' % (x_pos,y_pos,self.blink)
        ))
69
70     self.frame=frame
71     self.rawCapture.truncate(0)
72
73     if self.stopped:
74         self.stream.close()
75         self.rawCapture.close()
76         self.camera.close()
77         if self.arduino:
78             self.ser.write(str.encode('%3d%3d%1d' % (90,90,1)))
79             self.ser.close()
80         return
81
82 def read(self):
83     return self.x,self.y,self.x_conv,self.y_conv,self.radius,self.frame
84
85 def stop(self):
86     self.stopped = True
```

---

## A.3 calibrate.py

```
1 from picamera.array import PiRGBArray
2 from picamera import PiCamera
3 from threading import Thread
4 import cv2
5 import numpy as np
6 import time
7
8 class calibrate:
9     def __init__(self, left, right, lower, upper):
10        self.camera = PiCamera()
11        self.camera.resolution = (640,480)
12        self.r = 45
13        self.camera.framerate = 32
14        self.rawCapture = PiRGBArray(self.camera, size=self.camera.
resolution)
15        self.stream = self.camera.capture_continuous(self.rawCapture,
format="bgr", use_video_port=True)
16        self.resx, self.resy = self.camera.resolution
17        self.upper = upper
18        self.lower = lower
19        self.left = left
20        self.right = right
21        self.frame = None
22        self.x = 0
23        self.y = 0
24        self.radius = 0
25        self.stopped = False
26        self.font = cv2.FONT_HERSHEY_SIMPLEX
27        self.avglen = 50
28        self.xstorage = np.zeros((self.avglen,), dtype = np.int)
29        self.ystorage = np.zeros((self.avglen,), dtype = np.int)
30        cv2.namedWindow("trackbar")
31        cv2.resizeWindow("trackbar", 256, 240*(1024/2)/640)
32        cv2.createTrackbar("left", "trackbar", self.left, self.resx, self.
nothing)
33        cv2.createTrackbar("right", "trackbar", self.right, self.resx, self.
nothing)
34        cv2.createTrackbar("lower", "trackbar", self.lower, self.resy, self.
nothing)
35        cv2.createTrackbar("upper", "trackbar", self.upper, self.resy, self.
nothing)
36
37    def update(self):
38        for f in self.stream:
39            frame = f.array
40            if frame is not None:
41                self.left = cv2.getTrackbarPos("left", "trackbar")
42                self.right = cv2.getTrackbarPos("right", "trackbar")
43                self.lower = cv2.getTrackbarPos("lower", "trackbar")
44                self.upper = cv2.getTrackbarPos("upper", "trackbar")
45                frame = cv2.flip(frame, 0)
46                if self.left is not -1:
47                    frame = frame[self.lower:self.upper, self.left:self.
right]
48
49            self.frame = frame
50            self.rawCapture.truncate(0)
51
52            if self.stopped:
53                self.stream.close()
54                self.rawCapture.close()
55                self.camera.close()
56
57    def start(self):
58        Thread(target=self.update, args=()).start()
```

---

```

60     return self
61
62     def nothing(x,y):
63         pass
64
65
66     def calibrate(self):
67         while True:
68             if self.frame is not None:
69                 cv2.imshow("trackbar",self.frame)
70                 if cv2.waitKey(33) & 0xFF == ord(' '):
71                     self.stop()
72                     print(self.left,self.right,self.lower,self.upper)
73                     break
74
75             return self.left,self.right,self.lower,self.upper
76
77     def stop(self):
78         self.stopped = True
79         cv2.destroyAllWindows()

```

## A.4 stream.py

```

1  import numpy as np
2  import cv2
3  import urllib
4  from threading import Thread
5
6  class stream:
7      def __init__(self,URL="http://192.168.2.1/?action=stream"):
8          self.stream = urllib.urlopen(self.URL)
9          self.byte = ""
10         self.stopped = False
11         self.image = None
12
13     def update(self):
14         while True:
15             self.byte += self.stream.read(1024)
16             a = self.byte.find("\xff\xd8")
17             b = self.byte.find("\xff\xd9")
18             if a!=-1 and b!=-1:
19                 jpg = self.byte[a:b+2]
20                 self.byte = self.byte[b+2:]
21                 self.image = cv2.imdecode(np.fromstring(jpg, dtype=np.
22                 uint8),1)
23                 if self.stopped:
24                     return
25
26     def start(self):
27         Thread(target=self.update, args=()).start()
28         return self
29
30     def read(self):
31         return self.image
32
33     def stop(self):
34         self.stopped = True
35         cv2.destroyAllWindows()

```

---

## A.5 Arduino

### A.5.1 Master

```
1 #include "RF24Network.h"
2 #include "RF24.h"
3 #include "RF24Mesh.h"
4 #include <SPI.h>
5 #include <Servo.h>
6 #include <TrollBot.h>
7
8 char vals[6];
9 String strx;
10 String stry;
11 String blinkstr;
12
13 int LUmax = 110; // Fully open
14 int LUmin = 80; // Fully closed
15 int LLmax = 110; // Fully open
16 int LLmin = 80; // Fully closed
17
18 int EVmax = 110; // Fully down
19 int EVmin = 70; // Fully up
20 int EHmax = 110; // Fully left
21 int EHmin = 70; // Fully right
22
23 int xmax = 127;
24 int xmin = 127;
25 int ymax = 127;
26 int ymin = 127;
27
28 TrollBot Master('M');
29 TrollBot Albert('A');
30
31 void setup() {
32     Serial.begin(9600);
33
34     // Confirm Serial Connection to Python:
35     Serial.write('1');
36
37     Master.setup_setNodeID(0);
38
39     Albert.servo_attach(1, 0); // Eyes Vertical
40     Albert.servo_attach(2, 1); // Lids Lower
41     Albert.servo_attach(3, 2); // Lids Upper
42     Albert.servo_attach(4, 3); // Eyes Horizontal
43
44     Albert.servo_write(1, 90);
45     Albert.servo_write(2, 90);
46     Albert.servo_write(3, 90);
47     Albert.servo_write(4, 90);
48
49     delay(500);
50 }
51
52 void loop() {
53
54     Master.loop();
55
56     // Reading Serial Input into strings strx and stry, each 3 bytes, and
57     // blink, 1 byte
58     while (Serial.available() > 0) {
59         Serial.readBytes(vals, 7);
60         for (int i = 0; i < 3; i++) {
61             strx += vals[i];
62         }
63     }
```

---

```

63     for (int i = 3; i < 6; i++) {
64         stry += vals[i];
65     }
66     blinkstr += vals[6];
67
68     // Converting strings to ints. Range: 0-255
69     int valx = strx.toInt();
70     int valy = stry.toInt();
71     int blinking = blinkstr.toInt();
72
73     // Dynamic calibration
74     if (valx > xmax) {
75         xmax = valx;
76     }
77     if (valy > ymax) {
78         ymax = valy;
79     }
80     if (valx < xmin) {
81         xmin = valx;
82     }
83     if (valy < ymin) {
84         ymin = valy;
85     }
86
87     // Map calibrated values to servo values
88     int EVval = map(valy, ymin, ymax, EVmin, EVmax); // valy ymin-ymax,
89     EV 70-110
90     int EHval = map(valx, xmin, xmax, EHmax, EHmin); // valx xmin-xmax,
91     EH 110-70
92     int LUval = map(valy, ymin, ymax, LUmax, LUmin); // valy ymin-ymax,
93     LU 110-80
94     int LLval = map(valy, ymin, ymax, LLmin, LLmax); // valy ymin-ymax,
95     LL 80-110
96
97     // Truncate the strings, to make room for new serial input
98     strx = "";
99     stry = "";
100    blinkstr = "";
101
102    // If not blinking
103    if (blinking == 0) {
104        Albert.servo_write(1, EVval); // EV
105        Albert.servo_write(2, LLval); // LL
106        Albert.servo_write(3, LUval); // LU
107        Albert.servo_write(4, EHval); // EH
108    }
109    // If blinking
110    else if (blinking == 1) {
111        Albert.servo_write(1, EVval);
112        Albert.servo_write(2, LLmin);
113        Albert.servo_write(3, LUmin);
114        Albert.servo_write(4, EHval);
115    }
116 }

```

---

---

## A.5.2 Node

```
1 #include <TrollBot.h>
2
3 TrollBot Albert('A');
4
5 void setup() {
6     Albert.setup_setNodeID(0);
7 }
8
9 void loop() {
10    Albert.loop();
11    receive_send('A');
12 }
```

Appendix **B**

Project Thesis

---

---

# Technologies to control the eyes of patient simulators

---

---

PROJECT TASK DESCRIBING THE DEVELOPMENT OF AN  
OPERATOR CONTROLLED SYSTEM FOR SIMULATING EYE MOVEMENTS  
IN PATIENT SIMULATOR MANNEQUINS

TRONDHEIM, 2016

WRITTEN BY

**TRULS NYGAARD**

SUPERVISED BY

**MARTIN STEINERT  
CARLO KRIESI**

*Norwegian University of Science and Technology  
Trondheim*



**NTNU**

2016



---

# Summary

This project task identifies, collects, judges and conceptualizes technologies to simulate eye movement and behavior in healthcare simulator mannequins. The project identifies techniques for controlling the behavior of the eyes of mannequins in a realistic manner, and explores technologies for realistic movement.

Prototypes are shown that tackle the problems of human-like movement and behavior. A system is proposed for further development and potential research, and challenges and possibilities with this system are considered. The system prototype processes and captures the eye movement of an operator and mimics this in a mechanical eye. By letting the operator see the environment around the mannequin, they can look around and the mannequin will replicate this movement.

Several possibilities for automation of the behavior has been proposed, where the most promising is supervised learning in artificial neural networks. By using deep learning, it may be possible to train an algorithm to mimic human behavior and thus removing the need for operator control.

The system that has been proposed is the current iteration in a product development cycle and is far from a perfect solution. Implementation is dependent on further work and optimization. The system has potential, and it has to be further explored and developed.

---

# Preface

This project thesis describes the development of a prototype system for actuation and control of eyes in health care simulators, as requested by Lærdal Medical. It was written to fulfill the requirements of the Product Development and Materials specialization at NTNU's Department of Engineering Design and Materials. I was engaged in this project between August and December 2016.

The task was created as a collaboration between my supervisor Martin Steinert, my supporting coach Carlo Kriesi, representative from Lærdal Medical, Arild Eikefjord and me. The project let me experience health care simulations and their complexity. With the help of Medisinsk SimulatorSenter at St. Olav's Hospital, I got valuable insights into the use of mannequins in health care simulations.

I would like to thank Martin Steinert and Carlo Kriesi for excellent support and guidance during the project.

Truls Nygaard  
Trondheim 12.12.2016

# Table of Contents

<b>Summary</b>	<b>i</b>
<b>Preface</b>	<b>ii</b>
<b>Table of Contents</b>	<b>iv</b>
<b>List of Figures</b>	<b>v</b>
<b>1 Introduction</b>	<b>1</b>
<b>2 Literature and Technology Review</b>	<b>5</b>
2.1 Literature review . . . . .	5
2.1.1 Product development methodologies . . . . .	5
2.1.2 Humanoid robots . . . . .	6
2.1.3 Eyes . . . . .	6
2.2 Technology review . . . . .	7
2.2.1 Simulator eyes . . . . .	7
2.2.2 Animatronic eyes . . . . .	7
2.2.3 Computer vision and processing . . . . .	8
2.2.4 Virtual reality tools . . . . .	8
<b>3 Development</b>	<b>11</b>
3.1 Users and usage . . . . .	11
3.2 Prototyping . . . . .	12
3.2.1 Prototypes . . . . .	12
<b>4 Proposed solution</b>	<b>19</b>
4.1 System . . . . .	19
4.1.1 Environment recording . . . . .	20
4.1.2 Displaying the environment . . . . .	21
4.1.3 Recording eye movement of the operator . . . . .	21

---

4.1.4	Filtering and processing of blinking . . . . .	22
4.1.5	Processing the eye movement . . . . .	22
4.1.6	Actuation . . . . .	22
4.2	Usage . . . . .	23
4.2.1	Scenario: Medical students in simulation . . . . .	23
<b>5</b>	<b>Further work</b>	<b>25</b>
5.1	Training of scenarios . . . . .	25
5.2	Testing . . . . .	26
5.3	Actuation . . . . .	26
5.4	Challenges . . . . .	26
5.5	Research possibilities . . . . .	27
<b>6</b>	<b>Conclusion</b>	<b>29</b>
	<b>Bibliography</b>	<b>31</b>
	<b>Appendices</b>	<b>35</b>

# List of Figures

1.1	Lærdal's mannequin opened up . . . . .	2
1.2	Lærdal's SimPad . . . . .	2
1.3	Lærdal's mannequin with blinking . . . . .	3
2.1	Wayfaring model . . . . .	6
2.2	Eye of Lærdal's patient simulator mannequin . . . . .	8
2.3	Animatronic eyes (Discovery, 2011) . . . . .	8
2.4	Haar-like features used for detecting objects (Viola and Jones, 2001) . . . . .	9
2.5	Oculus Rift virtual reality headset, gathered from (Walton, 2016) . . . . .	9
3.1	Simple pupil dilation prototype . . . . .	13
3.2	Face recognition . . . . .	14
3.3	Mechanical eye, as it tracks a face . . . . .	14
3.4	Air muscle . . . . .	15
3.5	Fishing line muscle holding 500g . . . . .	16
3.6	Magnet control . . . . .	16
3.7	PiezoMotor's LEGS technology (PiezoMotor, 2016) . . . . .	17
3.8	Canon's Nano USM (Canon Imaging Plaza, 2016) . . . . .	17
4.1	Illustration of the flow of information . . . . .	20
4.2	The camera that records the environment, along with the mechanical eye . . . . .	21
4.3	Left eye is recorded by a camera and the pupil is located . . . . .	21
4.4	Simple mechanical actuation with two servo motors. . . . .	23

---

---

# Chapter 1

## Introduction

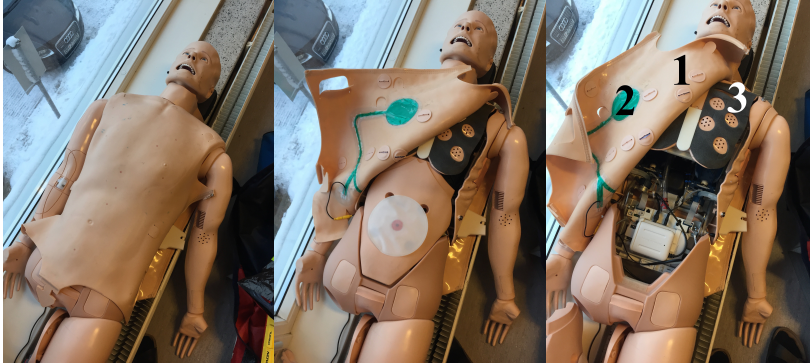
Lærdal Medical (later referred to as Lærdal) is an international company that develops advanced healthcare simulation mannequins. The company has its origins in Stavanger, and started out as a toy manufacturer, but transitioned into production and development of mannequins with the purpose of advancing emergency care and resuscitation.

Today, Lærdal manufactures advanced patient simulators (later referred to as mannequins) that have a broad spectrum of features and functions. They are able to simulate medical conditions that may occur too rarely, or be too life threatening to practise on real life scenarios. Conditions that rarely occur are hard to experience or practice on real patients, but mannequins can provide the necessary training for these situations.

The mannequins can be controlled to exhibit complex medical conditions, or the conditions can be more subtle. There are mannequins made for many different scenarios. Some can simulate heart conditions, breathing problems or similar. Some are in the shape of adults, while others are made to simulate a woman giving birth or an infant. The simulations are controlled by an operator that is trained to use the simulator software that often has clinical experience in healthcare. They control the mannequin's responses and outputs remotely through the simulation software.

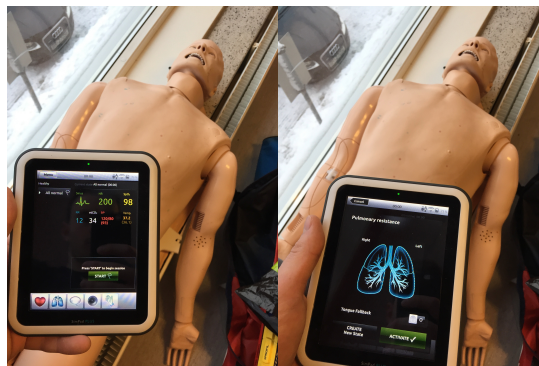
A mannequin was provided for this project. In figure 1.1 the mannequin is shown both closed and open. Some functions that can be seen in the figure are:

1. RFID chips embedded in the skin of the mannequin can be seen as small circular extrusions on the inside of the skin. These can be found with a tool that replicates ultrasound equipment and displays images of internal organs as if they are observed at that particular spot.
2. The green area is a patch for simulated defibrillation. To avoid destroying the mannequin, the participants use a special defibrillator and the mannequin acts as if it is a real defibrillator.
3. The ribcage is embedded with microphones that can simulate heartbeat and lung sounds when the participants listen with a stethoscope. The ribcage can be compressed when performing chest compressions.



**Figure 1.1:** Lærdal's mannequin opened up

These are only examples of the many functions this mannequin has. The mannequin and its functions are controlled by an operator through a simulation software. The Lærdal SimPad, as shown in in figure 1.2, can control the functions with an intuitive interface. It operates with a touch screen and can change lung sounds, heart rates, breathing patterns and much more.



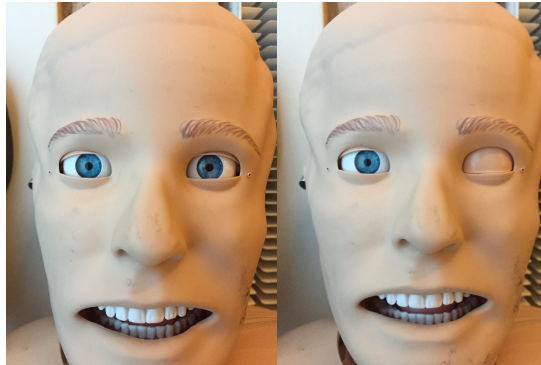
**Figure 1.2:** Lærdal's SimPad

One of the main challenges with the mannequin is noise. Mechanical noise from movement of the ribcage, friction between skin and mechanical components, compressor and air flow are examples of noise sources in the mannequin. When the doctor listens for heart sounds with a stethoscope, they can hear many of these sounds. This can impact the way they interact with the mannequin and in the worst case scenario, they are unable to identify crucial information in the heartbeat or lung sounds due to the interfering noise.



---

The eyes of the mannequins today have either few responses or no response. Some have blinking capabilities and pupil dilation which provide some information to the doctor. These functions can assist the doctor in determining some diagnoses. Figure 1.3 shows the blinking feature in one of Lærdal's mannequins.



**Figure 1.3:** Lærdal's mannequin with blinking

To improve interaction between doctors or other medical personnel who participate in the simulation (later referred to as participants) and the mannequin, Lærdal wants to re-design the eyes of the mannequin in such a manner that they can replicate human behavior and movement. By replicating these features, there can be an increased flow of information to the participant. Non-verbal communication is a large factor in how humans interact (see section 2.1.2), and can be an important way to determine if a person is acting normal or is suffering from some kind of disease or discomfort.

During simulations, an operator can speak through the mannequin's internal speakers to simulate conversations. This can help participants to train on building rapport with the patient, and it can help to communicate discomforts, pain or other information. While verbal communication through a microphone may assist the operator to convey a message or meaning to the participants, it has been shown that eye movements and the gaze can have a social and emotional function as mentioned in section 2.1.2. This means that we can transfer information about our emotional state as well as our feelings towards other individuals, just with our eyes.

There are also mechanical features to look for in the eyes. The eye has voluntary and involuntary movements that can be affected by medical conditions or drugs. The voluntary movements have 3 different modes of operation as mentioned in section 2.1.3: smooth pursuit, vergence shifts and saccades. There are also involuntary movements, such as pupil dilation, saccadic jerks, ocular flutter or reflexes. All of these movements may provide the doctor with crucial information about medical conditions.

This project is sponsored by Lærdal, and the scope is to explore technologies that can enhance simulations and communication between participant and mannequin by using the eyes of the mannequins. The main dimensions of interest are to replicate human physiol-

---

ogy, i.e the movement of the eye, and to mimic the behaviour of the eye. The project task consists of:

- Generating concepts
- Building prototypes
- Building test setups
- Testing and comparing alternatives
- Judging concepts

The execution of this product development task was performed in the fuzzy front end of the development process. Wayfaring is used as a product development methodology in this project. A wayfaring model is a non-linear development process with no fixed goals. The main objective is to change often and iterate rapidly. This is far less costly in the early stages of the product development process, and can provide better solutions in the end, as more dimensions have been explored. This is further explained in section 2.1.1

# Literature and Technology Review

## 2.1 Literature review

### 2.1.1 Product development methodologies

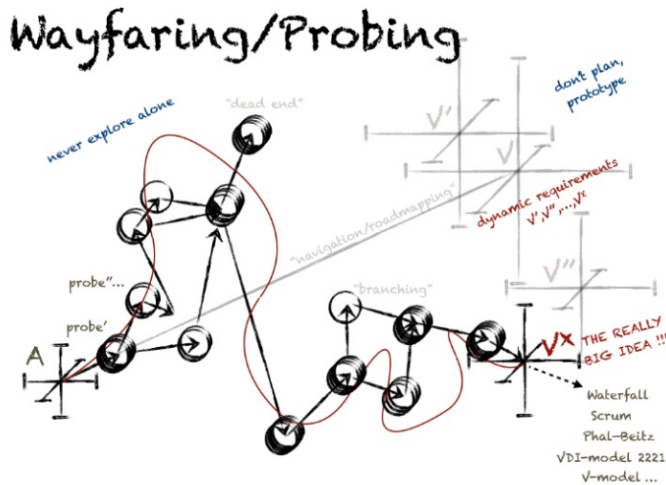
This project utilizes a wayfaring model in the fuzzy front end of the product development cycle. To find the needs of the users, observation of user cases and interviews have been conducted.

Smith and Reinertsen (1992) coined the phrase "fuzzy front end" of product development. They explained that the early stages of the development offer the best opportunities for large, cheap changes to the product development cycle. They call it the fuzzy time between idea and large investment of resources. Changing product idea or concept in this phase is inexpensive and easy, while later stage change can be costly.

Addressing the right problems in the right way is a large part of the fuzzy front end (Gassmann and Schweitzer, 2014). This is not always easy, as it is impossible to know what the best solution is before you start. This is why a wayfaring model can be introduced in the fuzzy front end (Steinert, Martin and Leifer, Larry J., 2012; Gerstenberg et al., 2015). A wayfaring model can be seen in figure 2.1, and it is based on the idea that radical innovations are outside of what is possible to plan and execute linearly. When using a wayfaring model, you are aware that at any point it might be necessary to change your design radically, throwing away a lot of what you have done so far. When developing in the fuzzy front end of the development cycle, radically changing a design is not costly.

In (Leifer and Steinert, 2014), need finding is defined as being central to the early stage development. To establish the needs of users, one can borrow anthropological methods for immersing oneself in the situation of the user. Needs are often not spoken, or the spoken needs do not reflect the true needs. To establish an understanding of how the problem can be tackled, it is important to both observe and participate. This allows the product developer to obtain a feeling of the problem that otherwise could be lost.

To reduce effort in product development, it is possible to utilize a Wizard of Oz method to simulate complicated systems (Dow et al., 2005). By simulating the proposed system



**Figure 2.1:** Wayfaring model

with manual control, one can get feedback and avoid rework. This means that if you have a system that requires a lot of software development, it might be a good idea to manually control parts of the system, and iterate on the feedback you get from user testing.

### 2.1.2 Humanoid robots

Non verbal communication in a health care setting has been found to be very important. Both patients and doctors are affected by the counterpart's gaze (Hall and Lloyd, 1990; Hall and Knapp, 2013). Eye contact can greatly increase the patient's attitude towards the doctor, and it can function as a reassurance and be calming to the patient. However, there has also been conducted research to establish how this effect relates to humanoid robots.

Is there a difference in how we interpret non verbal communication from a human and that of a humanoid robot? As human likeness in humanoid robots increases, there will be an increase in how familiar the robot feels, until a certain point called the uncanny valley (Ishiguro, 2007). When the robot becomes very similar to a human, there will be little more than subtle differences that makes the humanoid differ in appearance from a real human. These differences can be small, but they will be enough to create unease with the observer. A sensation of strangeness can occur, as the differences can feel intangible and eerie.

### 2.1.3 Eyes

Wong (2008) provides extensive details about the movements of the eyes, both voluntary and involuntary, and explains medical conditions that can affect or induce different movements. The most important information for this projects is that is controlled by 6 muscles,

---

and has 6 degrees of freedom. The main voluntary movements are smooth pursuit (tracking objects), vergence shifts (binocular focusing) and saccades (rapid shifts in gaze and focus). Human eyes perform approximately 10.000 saccades each day. In diagnosing of medical conditions, these three voluntary movements can provide important information, and can be examined by simple tests. Tracking a finger from left to right will show if the patient is capable of smooth pursuit. Tracking a finger that moves from a short distance away, towards the nose, will show if the patient is capable of vergence shifts. Saccades can be tested by having the patient move their gaze from one point on the left to another, one the right.

Involuntary movements are numerous and complex. Wong (2008) explains the complexity of involuntary movements, and how they can be used in clinical diagnosing. An example of an involuntary movement, can be Nystagmus. Involuntary oscillations of the eye initiated by slow eye movements drives the eye away from the target. The oscillations appear differently based on different medical conditions, and can be induced by different types of movements. Some conditions cause the eyes to move in sinusoidal patterns, while others cause linear jerking patterns. Some are presented while head is still, while others appear only with head movement.

To track the eyes, there are many possibilities. The need for vision tracking has been present for some time and there are two types of eye tracking: tracking motion relative to the head, and tracking orientation in space (Duchowski, 2007). The most common technique relies on placing electrodes around face to measure the skin's electrical potential differences. One technique uses infrared light reflecting of the eye into a sensor and another technique uses video detection of the pupil. Video detection tracking is the easiest to implement, as it requires little calibration or signal interpretation. This method tracks the eye's orientation in space, not relative to the head.

## **2.2 Technology review**

### **2.2.1 Simulator eyes**

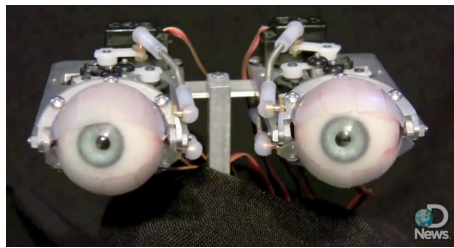
The eyes used in one of Lærdal's simulators can be seen in figure 2.2. These eyes are immobile and have a blinking mechanism. They also have the option to include pupil dilation functionality. The pupil dilation is reacting to light conditions, like in the human eye. The blinking is controlled by the operator, through the simulation software, and is actuated with an electromagnetic motor.

### **2.2.2 Animatronic eyes**

Animatronics is the technique of making and operating lifelike robots. Animatronic projects have made realistic eyes both in movement and appearance. These can move in a manner that seems fluid and natural. They are controlled either by code, or manually in real time by an operator and usually actuated by servos. This makes them look natural, but there is usually some noise when actuated. An example of the animatronic eyes can be seen in figure 2.3.



**Figure 2.2:** Eye of Lærdal's patient simulator mannequin



**Figure 2.3:** Animatronic eyes (Discovery, 2011)

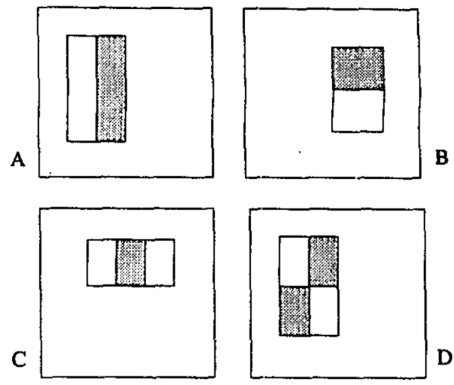
### 2.2.3 Computer vision and processing

In this project, processing is done in python with serial communication to an Arduino. By utilizing powerful computer vision tools, it is possible to process real time video streams and utilize feature recognition software to control the output to the Arduino.

OpenCV is a python library that handles computer vision and can perform feature recognition. By using machine learning techniques on Haar-like features in images (Figure 2.4), OpenCV can be trained to detect objects (Bradski, 2000; Viola and Jones, 2001). The classifier is called a Haar cascade classifier, and in the OpenCV library there are cascades for, amongst others, eyes and faces. Training a Haar cascade classifier to identify a new object would require around 10 000 positive and negative images to achieve an acceptable accuracy. A positive image in this case is an image that contains the object you want to detect, while a negative image does not.

### 2.2.4 Virtual reality tools

Virtual reality has boosted technology surrounding images displayed close to the eyes. An example of a virtual reality headset can be seen in figure 2.5. Lenses and headsets can come in all shapes and price ranges. Some are cheaper than ever and easy to work with while other technologies are kept as trade secrets. Software is available for processing and



**Figure 2.4:** Haar-like features used for detecting objects (Viola and Jones, 2001)

distortion of the image. Small high resolution screens with high frame rates are becoming more common and inexpensive and can be easily implemented.



**Figure 2.5:** Oculus Rift virtual reality headset, gathered from (Walton, 2016)

---

---



# Chapter 3

## Development

### 3.1 Users and usage

Through observation of simulations, participation and interviews, two main users and their different needs were identified. Simulation and interviews were conducted in St. Olav's hospital with base in their simulator center. Observed simulations were emergency care of an infant arriving in ambulance, conducted in the emergency room, as well as a cardiac arrest case where a doctor was found lifeless in a hallway.

There are two main users of the patient simulator mannequins: simulation participants, which can be a doctor, medical student or others, and the operator, who controls the mannequin and the simulation scenario. The users have different ways of interacting with the mannequin. The participants are training in a simulation environment to obtain experience that might be difficult to get from real life scenarios. The operator controls the mannequin and interacts with the system controls and the participants if needed. The operators are in the background and oversee the simulation.

The two main users have different needs that have to be fulfilled. For the operator, the mannequin, as well as the software and mechanical functions have to be operational at any time. There should be no downtime or delays. The operator needs to be able to alter scenarios in real time to create a fluid environment that adapts. The participants need to be able to extract information from the patient simulator to generate meaning. This meaning can be deduced from mechanical feedback, sound or visual cues. This meaning should lead to a conclusion or diagnose, and the participants should learn from the experience. Was the diagnose correct or wrong, and why?

Since participants in the simulations often do not know about the mannequin's full capabilities, they are prone to misinterpreting a lack of response from the simulator. They are not sure whether no response is a result of limited capabilities of the simulator, or if it is indeed a part of the simulation. This creates an artificial environment in which the doctor is very aware that it is indeed a simulation and contacts the operator to gather information, instead of the patient. This leads to behaviors that are unusual or non-intuitive for the simulated situation.

---

Some participants reported that they changed their mindset when approaching a simulation. They report that when participating in a simulation, they no longer think in the same way as when they work with real patients, and they enter a "simulation mindset". Thus, they are training to become very good simulation participants, but can struggle to make the training relevant to real cases.

## **3.2 Prototyping**

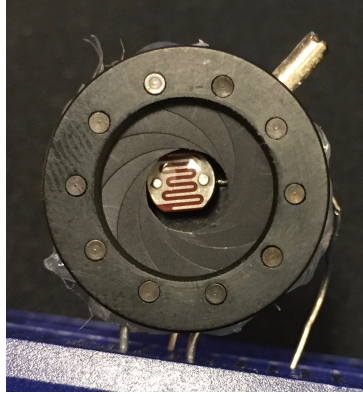
To incorporate wayfaring into the fuzzy front end of this development process, iterations and rapid prototyping has been heavily weighted. By prototyping in dimensions of interest, it is possible to learn from the process of development and testing (Erichsen et al., 2016). This can enlighten new dimensions of interest or needs that can be further explored. In a non-linear product development process, like this one is, it is important to not get locked into solutions.

Prototypes should be developed in certain dimensions of interest. This project tries to fulfill the needs of the users by focusing on generating meaning from the mannequin eyes. By introducing behavior and movement to the mannequin eyes, it may be possible to simulate human non verbal communication.

### **3.2.1 Prototypes**

To develop meaningful behaviour of the eyes, it is important to look at how the human eye moves. The human eye tends to look at faces and objects, while also performing micro-movements that scan the surrounding environment (Cerf et al., 2009). The meaning these movements create is based on their responsiveness, range and accuracy. To explore these factors during simulation, participants should be able to communicate with and observe the patient, and they can perform tests to search for different faults in the patient's vision or movement of the eye.

Some features are easier to replicate than other, such as pupil dilation. One simple example of pupil dilation is shown in figure 3.1. This prototype utilizes a light-dependent resistor to determine the amount of light that enters the "eye" through an iris diaphragm, and the diaphragm closes to adapt to the amount of light.

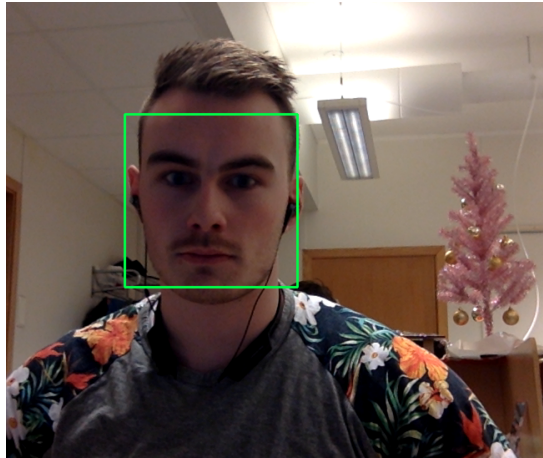


**Figure 3.1:** Simple pupil dilation prototype

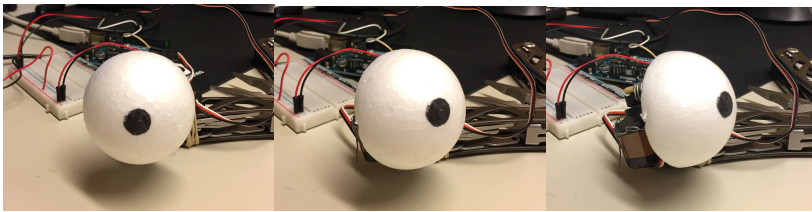
The implementation of movement and behavior in a mannequin is a complex problem. The problem can be split into two parts: actuation, which replicates the movement, and environment response, which replicates behavior. The development of both actuation and environment response was conducted in parallel.

### **Environment response**

The scope of the first environment response prototypes was to autonomously act upon an observed environment. The prototypes were developed in Python 3.6 with communication to Arduino through serial. The code for the face recognition can be seen in Appendix A. By capturing and scanning each frame of the video stream of the mannequin's environment, searching for faces with OpenCV's Haar cascade classifiers (Bradski, 2000), as seen in figure 3.2, it is possible to control the mannequin eyes. A camera sends a video stream to the processor, and the processor sends a control signal to an Arduino, which in turn controls the movement of the mechanical eye, as seen in Figure 3.3.



**Figure 3.2:** Face recognition



**Figure 3.3:** Mechanical eye, as it tracks a face

When there are multiple faces present in the video stream, there are several possibilities for choosing the correct one. This can be chosen randomly, or by directional audio, if someone is speaking. If someone is speaking, it might also be possible to determine this with movement of the mouth. To tackle this problem, a directional microphone was used to record audio in parallel with the face recognition. To choose which face to look at, a test setup was developed. By recording the audio, as well as the position of the faces in view, it is possible to use a machine learning algorithm to choose which face to look at. In Appendix B, a python code is shown which records all data and prepares it for machine learning. The data will consist of an audio file and a position file. The labels of the data tell us which person is speaking. The audio file can be processed to extract uniform features per audio chunk, such as volume, pitch, difference between channels or similar.

The test setup is to have two or more faces present in the environment, and have one person speaking. The data is then labelled, and it is possible to use supervised learning algorithms to create a classifier than can choose the right face in real time. A suggested machine learnign algorithm for a problem like this is Support Vector Machines (SVM) (Mohri et al., 2012).

---

Creating coded behaviour that appears natural and not eerie, as mentioned in section 2.1.2, is extremely difficult. It requires incredible amounts of coding to cover all possible outcomes. To circumvent this problem, a Wizard of Oz-inspired prototype (See section 2.1.1) was developed and implemented in the proposed system as described in Chapter 4. The original idea with this prototype was to teach the system where to look and to use the prototype to obtain user feedback. It turned out that to implement an operator controlled eye, might provide better results than a coded/programmed controlled eye.

### **Actuation**

Actuation of the mechanical eye is an important factor in order to avoid the uncanny valley. To create fluent, fast, precise and silent movement can be incredibly difficult. Some solutions offer one or more of these dimensions, but few offer all of them.

Air muscles as shown in figure 3.4 are strong linear actuators that expand radially to create a linear pulling force. These can be controlled quickly by applying air pressure, but they operate in a binary fashion if you can't control the pressure precisely. If you are able to control pressure, the muscles can be positioned fairly accurately. When releasing pressure, there is significant noise. However, they are cheap and easy to make and can be created in many sizes.



**Figure 3.4:** Air muscle

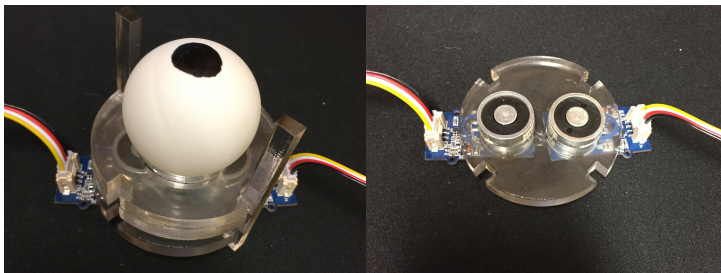
Heat controlled muscles can be made from fishing lines by applying a constant axial force and then twisting the line until it creates a coil. This coil has the property of contracting when heated, and some versions of these muscles can contract up to 49% (Haines et al., 2014). This muscle can be controlled precisely, silently and quickly, if you are able to control the temperature. By putting the coil inside a flexible tube, it is possible to feed warm or cold water to the muscle, and the muscle will contract based on the temperature of the water.

Magnet control is viable as an actuation method, if there is sufficiently low friction in the bearings of the eye ball. An experimental setup is shown in figure 3.6, where the eye can be controlled in one axis between the two magnets. The electromagnets interact with a permanent neodymium magnet attached to the eyeball, and by regulating the relative voltage between the magnets, it is possible to control the positioning of the magnet very accurately. Each magnet can be controlled with pulse width control (PWC) between 0 (off) and 255 (maximum power). This means that the position of the neodymium magnet can be



**Figure 3.5:** Fishing line muscle holding 500g

regulated between the electromagnets. By setting the magnets' power to (LEFT,RIGHT) = (255,0), the magnet will be in the extreme left position. (255,255) is center, and (0,255) is the extreme right. This means that we can control the position very accurately in the available movement space. Drawbacks of this actuation is that the magnets produce some sound and can get hot over time.



**Figure 3.6:** Magnet control

Servo motors can be a cheap way to actuate both rotation or linear motion. With a precise, fast response, they are easily controllable. They are small and can produce significant forces. There are however few silent servo motors, and they produce significant amounts of noise. To reduce the noise, it is possible to lubricate the gears. Digital servos also produce more noise than analog ones, as digital servos have constant position regulation.

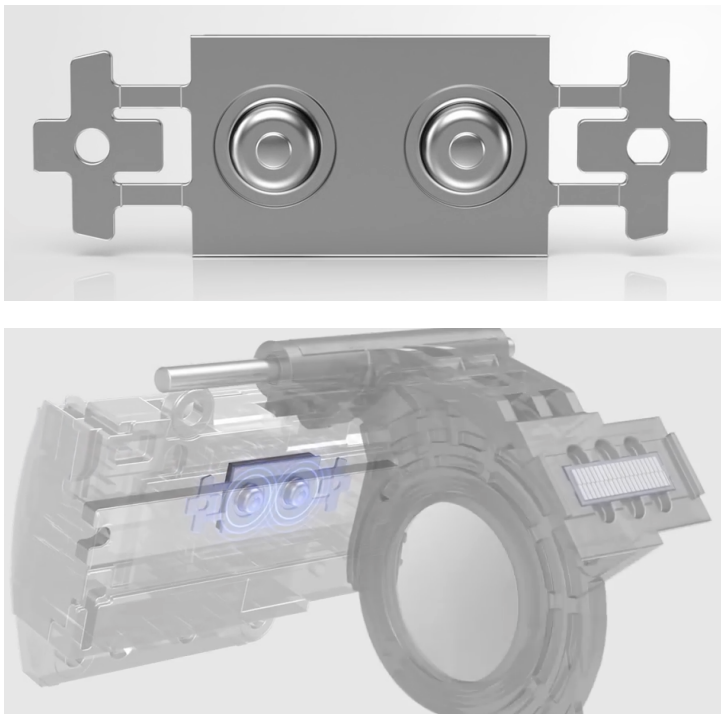
Piezoelectric motors are motors that are controlled by sending specialized electric signals to a piezoelectric ceramic. This ceramic actuates a certain movement based on the

---

electrical signal and the shape of the ceramic. This allows the piezoelectric ceramic to perform movements that are similar to walking (PiezoMotor, 2016). They perform stepwise movements with very precise control. Many piezoelectric motors offer sub-micrometer movement, and some even sub-nanometer movement. The actuation speeds range between 1 and 200 mm/s with varying positioning precision. Piezoelectric motors are highly customizable, and can be very small. Canon produces DSLRs that have a small piezoelectric motor that drives the autofocus of the lens. This operates on ultrasonic frequencies, and is completely inaudible. It has a fast motion that is very precise (Canon Imaging Plaza, 2016).



**Figure 3.7:** PiezoMotor's LEGS technology (PiezoMotor, 2016)



**Figure 3.8:** Canon's Nano USM (Canon Imaging Plaza, 2016)

---

---



# Chapter 4

## Proposed solution

This project describes the early stages of product development of patient simulator eyes. This implies that a proposed solution is temporary. Constant learning and testing creates new solutions that changes the product. The proposed solution is only the current iteration in the prototyping process.

### 4.1 System

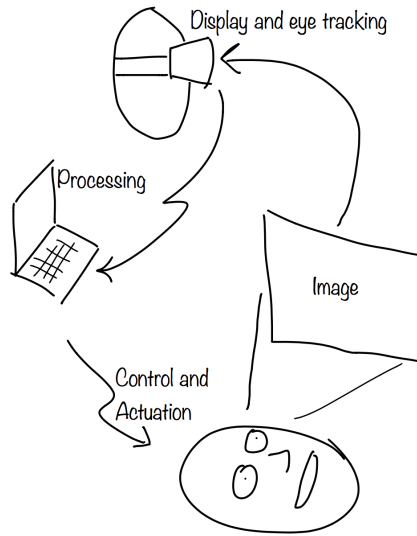
When creating a control system for the eyes of the patient simulator, natural movement and behaviour is important. To create an interaction between patient and doctor, there has to be actual interaction. There has to be emotion and communication through the eyes, and the behaviour of the eyes has to induce meaning with the doctor.

To facilitate this communication, the proposed solution is to mimic the eye movement of the operator, as they look through the eyes of the mannequin. The operator can put on a headset, similar to a virtual reality headset, where they can see a video stream of what the mannequin sees. In real time, the headset tracks the eye movement of the operator, and mimics this movement in the mannequin. This means that if a person enters the mannequin's field of view, and the operator looks at this person, the mannequin also looks at this person.

To manually control the eyes of the mannequin during all simulations is not feasible. This is why further work will include attempts to record emotions, behaviours and health conditions. By learning patterns and movements that are distinct for each situation, these situations can be played out without manual operation.

The proposed system is developed to create natural communication between the operator and the patient. The subsystems have to be further developed and optimized to enhance the communication. The subsystems of the product are:

- Environment recording
- Displaying of the environment to the operator



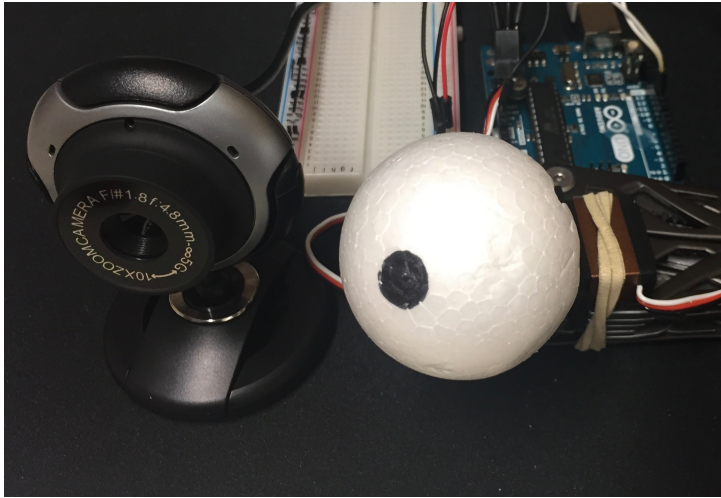
**Figure 4.1:** Illustration of the flow of information

- Recording eye movement of the operator
- Filtering and processing of blinking
- Processing the eye movement and blinking, and translating into actuation control
- Actuation

The prototype is built using 2 cameras, one display, a virtual reality headset, an Arduino, a computer and two servo motors. The video streams are interpreted with python 3.6 and the computer vision library OpenCV 3.1 and processed on the computer. The code can be found in Appendix C. The display used in the prototype is an iPhone that displays the video from the camera that records the environment. The other camera is placed in the headset and records the left eye of the operator. The actuation of the servo motors is controlled by an Arduino that receives the coordinates through serial communication.

#### 4.1.1 Environment recording

To be able to translate what the mannequin "sees", it is necessary to record the environment. The mannequin already records sounds, and by putting a camera on the doll that records its field of view, it is possible to feed this image to the operator or to an algorithm as will be discussed in section 5.1. By avoiding advanced image processing and relying on human processing of the images, a more natural behaviour can be implemented or trained. The camera that records the environment should have a wider angle lens than the current one, as the field of view is narrower than that of a human.



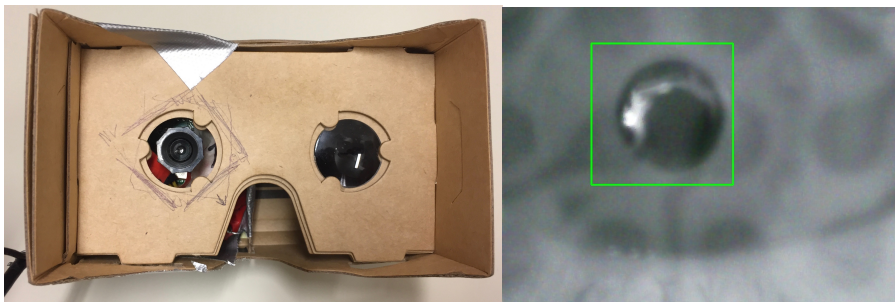
**Figure 4.2:** The camera that records the environment, along with the mechanical eye

### 4.1.2 Displaying the environment

When the operator has the headset on, a display inside the headset provides a real time video feed from the mannequin. This is displayed close to the eye with a lens that lets the operator focus on the screen. When showing the environment to the operator, they will instinctively start to look around and try to make sense out of everything that is displayed. This means that the operator shows us a natural way of looking at the environment.

When using lenses to display a screen close to the eye, it is important to have a high resolution screen. The pixels can be very easy to see when looked at up close. There is also some distortion due to the curvature of the lens, which has to be counteracted.

### 4.1.3 Recording eye movement of the operator



**Figure 4.3:** Left eye is recorded by a camera and the pupil is located

---

The camera in the headset records the left eye of the operator. With 3 infrared diodes and an infrared camera chip, it is possible to detect the pupil even with no light present. The infrared light creates no reflection on the eyeball. Combined with the fact that the headset blocks a large amount of visible light from entering, it is possible to get accurate readings without being affected by lighting conditions. In figure 4.3, there is some reflection, as the headset is not closed fully.

In the current iteration, the camera that records the eye has a defect that affects the image it records, as seen in figure 4.3. This lowers the accuracy of the detection, but this can be counteracted by using a camera without defects.

#### **4.1.4 Filtering and processing of blinking**

In eye tracking, blinking is usually considered to be noise that has to be filtered out. In this application, it is possible to utilize blinking as an actuation tool. When the operator blinks, there will be 1-2 frames in the video stream where no eye is detected. This can be interpreted as a blink, meaning that it is possible to communicate to the Arduino that blinking has occurred. If the operator closes their eyes, or removes the headset, there will also be no eyes detected. The difference can be detected easily through coding, as there will be larger amount of frames with no eye detected.

#### **4.1.5 Processing the eye movement**

The video stream from the headset is processed with the OpenCV library and the position of the pupil is found by utilizing a Haar cascade classifier trained to find eyes. The classifier is not optimized to find pupils, but it is possible to train a new Haar cascade classifier to achieve a higher accuracy, as mentioned in section 2.2.3.

If the classifier finds an eye, the eye is recorded with an values  $(x,y,w,h)$  which describes the middle point of the pupil and the width and height. When the eye's position is found, this is translated and sent to the Arduino through a serial port. The Arduino controls actuator with the coordinates that are received. The actuators are controlled with an adjusted value based on the eye's position in the video stream. If the video has resolution  $600 \times 480$ , and the pupil is found in  $(x,y) = (300,400)$  the Arduino will receive 2 bytes with values between 0 and 255. This means that x value received will be 127 and the y value will be 212. The Arduino can then control the actuators with these values. If the actuators are 180 degree servos, the Arduino has to map these values between 0 and 179.

In optimization of the accuracy of the tracking, it is possible to have the recorded eye position displayed to the operator. In effect, the operator is shown where the program thinks they are looking and it is possible to calibrate this until the eye tracking is correct.

#### **4.1.6 Actuation**

Actuation of the eyes is a complicated matter. To achieve a natural behaviour of the eye, it is important that the movement is fluid and that it is silent. Possibilities for optimal actuation is considered in section 3.2.1 and 5.3. In the current prototype, servos are used to illustrate the movement that it is possible to achieve. While not being optimal, they serve prototyping purposes.



**Figure 4.4:** Simple mechanical actuation with two servo motors.

One proposed solution is to use piezoelectric, ultrasonic motors. They make no audible sound and can control the movement rapidly and precisely. Ultrasonic piezoelectric motors are used in silent autofocus motors in DSLR cameras, and can have sub-micrometer precision at speeds up to 200 mm/s. Due to their complexity and price, they have to be manufactured to order. The largest drawbacks are that they are expensive and require large amounts of engineering. They also require external driver electronics that should be engineered to the particular motor and use. Still, they can be customized to be extremely compact and specialized.

For further development, prototyping samples of linear piezoelectric motors have been requested by PiezoMotor, a Swedish piezomotor manufacturer.

## 4.2 Usage

The proposed system is designed to be used manually in live simulations and also in training of scenarios that can be put in a library. The use is intended to further enhance patient-doctor interactions and to simulate conditions that require communication. Some conditions require thorough examination of the eyes, which can be simulated with this solution.

### 4.2.1 Scenario: Medical students in simulation

Medical students with limited experience are often put into low stress simulations in order to get used to interacting with patients. Some scenarios may be emergency care patients with non life threatening conditions, or nursing of post-operation patients. Communication is a tool that can help diagnosing certain conditions, like strokes. Movement of the eyes

---

may also contribute to eliminating the feeling of a simulation, as the mannequin appears more human.

In this scenario, the operator can assume the role of the patient, and the participants can interact with them by talking and observing. Certain conditions may be revealed through the way a person blinks, how their field of view is or how quickly they respond to visual cues. All of this is possible to implement by manipulating the image displayed to the operator, or in the mechanical response of the eye.

# Chapter 5

## Further work

Continuing development will be done in a couple dimensions of interest. To create a responsive actuation, the hope is to acquire prototyping samples of piezoelectric linear actuators. Precise control of these can provide the desired movement of the eyes.

Another dimension of interest is to optimize eye detection and calibration. By creating a Haar cascade for the pupil, it could be possible to achieve better tracking. Another way to increase accuracy is to get better calibration from testing.

To achieve better environment response and the implementation of medical conditions, it will be interesting to explore machine learning, and to conduct user tests where an operator can control the eyes while a participant interacts with them.

### 5.1 Training of scenarios

Since the operator has other tasks than just operating the eyes, the mannequin should be able to be taught how to act during certain specified scenarios. A large opportunity for development is the training of scenarios. By recording certain behavior that is specific to one condition, it might give the operator the freedom to execute pre-trained programs that can act out conditions such as panic, pain, strokes etc..

There are a number of medical conditions that can affect the way a person sees their environment. Some conditions can give the patient blind areas in their field of view. Drugs can affect the movement of the eyes and the way a person blinks. Some drugs make the eyes move very rapidly, while others make them slow. Certain conditions can make the eyes flicker and move inconsistently, as mentioned in section 2.1.3.

The training environment might be based on position recording of the eye, as well as blinking and observed faces from the environment. The training can either be recorded and played out directly, or it can be trained by machine learning and respond to inputs from the environment. A third alternative is a combination of the two. By using recorded behavior and add coded or trained behavior as a response to certain environment changes, such as bright lights, faces or voices.

---

A feasible machine learning approach may be artificial neural networks. By using supervised deep learning neural networks (Bengio, 2009), it could be possible to replicate the behavior of the operator based on inputs from the environment. This can create smart behavior that mimics the operator. This has the potential of creating an artificial intelligence (AI) that can make the need for operator control obsolete.

To be able to execute a neural network training, the pupil recording has to be more precise. A deep learning framework that fits the needs of this project may be Caffe (Jia et al., 2014). This framework is made for expression, speed and modularity by the Berkeley Vision and Learning Center. The framework has a Python interface and has been used for advanced visuomotor tasks before (Levine et al., 2015). It has among other things been used to teach a robot to screw a cork on to a bottle.

A possible outcome of the neural network, based on the accuracy and extent of training that is performed, is the training of emotional states. It might be possible to train the algorithm to exhibit stress or discomfort.

## **5.2 Testing**

User testing has not yet been performed. The current prototype has been made from observed needs of operator and simulation participants. Testing will provide crucial feedback on the system and on the design. The goal is to implement simple user tests with facilitators at St. Olav's Hospital when the accuracy of the eye tracking is sufficient.

By performing user tests it is possible to obtain feedback on the system as a whole, as well as comfort, ease of use and their general thoughts. The testing is expected to reveal unforeseen problems and concerns.

## **5.3 Actuation**

Further work on the actuation will hopefully include piezoelectric motors, if they are obtainable. Otherwise, the easiest solution would be to obtain small, analog servos that can be lubricated to limit noise. A more challenging solution would be to implement the heat controlled fishing line muscle, as the technology is really promising in terms of speed of actuation and strength. Magnet control is also an alternative that is promising and can be further explored. For testing, there should be a complete actuation system available, preferably with 2 eyes, including blinking. This can help to get user feedback when testing.

## **5.4 Challenges**

The main challenges of the implementation of this system are to create lifelike movement and to make the behavior accurately represent the operator's behavior. There might be some problems with processing power, depending on the camera resolution. The current programs are running on a 2.5 GHz Intel core I7, but it is still a demanding process.



---

Creating a Haar cascade classifier for the pupil might increase the accuracy of the detection algorithm as mentioned in section 2.2.3. This requires a large amount of preparation of the images, but it should be possible to create. Calibration of eye position to the recorded environment as well as the actuated eye might be a challenge. The camera observing the environment might have to be a wide angle lens in order to capture a sufficient view. The calibration is also important for creating effective communication between the participant and the mannequin.

Another challenge is to implement medical conditions that alter the mechanical movement of the physical prototype. As discussed in section 2.1.3, there are large amounts of involuntary movements and movement patterns that can be affected by different medical conditions. Achieving accurate representation of the most important and decisive movements may depend on the actuator accuracy and speed.

## **5.5 Research possibilities**

In section 5.1, a neural network is proposed for mimicking human environment response. Research on gaze and eye tracking is mainly done with respect to classification of diseases and to analyse where a person's gaze is focused. In Campana et al. (1999) they use neural networks to classify identify patients with Schizophrenia.

The possibility of creating a behavior of the eyes in the mannequins that mimic human behavior through neural networks may prove to be an interesting study. An alternative is to make a comparative study that compares human control against recorded behavior, coded behavior and neural net control, and performing Turing tests.

---

---

# Chapter 6

## Conclusion

This project has had the task of identifying, collecting, judging and conceptualizing technologies to simulate human eye movement and behavior. Through a wayfaring model, this task has been performed and the iterations and prototypes have been shown in this paper. The proposed system is put together from a set of subsystems that have been generated throughout the project, and is the current iteration in both actuation and environment response.

The system provides some solutions to the needs that have been identified through user observations and interviews. At this time, the system is at an early prototyping stage. To venture forward, there are possibilities and challenges that must be addressed. The system needs to be user tested when the prototypes are mature enough to be tested. The viability of the proposed system is dependent on further development and testing.

Further dimensions of development have been proposed in order to be able to leverage this technology in a simulation setting. This includes optimization and calibration as well as the creation of a training environment, when it comes to behavior.

Actuation must be implemented in the complete system, and not as stand-alone components. The environment response system is capable of sending control signals that have to be acted out. Further work on the actuation must be performed so that the mechanical system can operate flawlessly in combination with the rest of the mannequin.

By using the operator as the control for the eyes, the system becomes a lot more fluid and natural. The coding also becomes less complex and requires less processing power. The code that mimics the operator's behavior is very simple, as opposed to one that has to account for every possible scenario that could happen. The implementation of taught behavior based on recorded scenarios, may be an effective way to simplify the operator's experience. A library of predefined conditions may offer a powerful communication.

As mentioned in section 5.1, it might be possible to train an algorithm that can control the behavior of the mannequin. If this done in a sufficient manner, it will be interesting to perform a Turing test (Harnad, 2003). A Turing test asks humans to try to distinguish a machine from a human. If they are indistinguishable, the machine has passed. In this case, the test would be to determine if the eyes are controlled by an operator or by the neural

---

net.

All in all, the system as it is now, shows the possibilities of what may be achieved with further development. The system may provide a powerful platform for communication that could improve learning in simulation scenarios. Human non verbal communication is a crucial component in how we interact, and this may be a tool for introducing non verbal communication to health care simulations.

# Bibliography

- Bengio, Y., 2009. Learning Deep Architectures for AI. *Foundations and Trends in Machine Learning* 2 (1), 1–127.  
URL <http://www.nowpublishers.com/article/Details/MAL-006>
- Bradski, G., 2000. Dr. Dobb's Journal of Software Tools.
- Campana, A., Duci, A., Gambini, O., Scarone, S., Jan. 1999. An Artificial Neural Network That Uses Eye-Tracking Performance to Identify Patients With Schizophrenia. *Schizophr Bull* 25 (4), 789–799.  
URL <http://schizophreniabulletin.oxfordjournals.org/content/25/4/789>
- Canon Imaging Plaza, 2016. ACTUATOR NANO USM "EF-S18-135mm F3.5-5.6 IS USM" (CanonOfficial).  
URL <https://www.youtube.com/watch?v=8YUOUB96YBk>
- Cerf, M., Frady, E. P., Koch, C., Nov. 2009. Faces and text attract gaze independent of the task: Experimental data and computer model. *Journal of Vision* 9 (12), 10–10.  
URL <http://jov.arvojournals.org/article.aspx?articleid=2122098>
- Discovery, 2011. Animatronic Eye Mechanism Explained.  
URL <https://www.youtube.com/watch?v=p3BMrt3PD6A&t=33s>
- Dow, S., MacIntyre, B., Lee, J., Oezbek, C., Bolter, J. D., Gandy, M., Oct. 2005. Wizard of Oz support throughout an iterative design process. *IEEE Pervasive Computing* 4 (4), 18–26.
- Duchowski, A., 2007. Eye Tracking Techniques. In: *Eye Tracking Methodology*. Springer London, pp. 51–59, doi: 10.1007/978-1-84628-609-4\_5.  
URL [http://link.springer.com/chapter/10.1007/978-1-84628-609-4\\_5](http://link.springer.com/chapter/10.1007/978-1-84628-609-4_5)

- 
- Ericksen, J. A. B., Pedersen, A. L., Steinert, M., Welo, T., 2016. Learning in Product Development: Proposed Industry Experiment Using Reflective Prototyping. *Procedia CIRP* 50, 454–459.  
URL <http://www.sciencedirect.com/science/article/pii/S2212827116303821>
- Gassmann, O., Schweitzer, F., 2014. Managing the Unmanageable: The Fuzzy Front End of Innovation. In: Gassmann, O., Schweitzer, F. (Eds.), *Management of the Fuzzy Front End of Innovation*. Springer International Publishing, pp. 3–14, doi: 10.1007/978-3-319-01056-4\_1.  
URL [http://link.springer.com/chapter/10.1007/978-3-319-01056-4\\_1](http://link.springer.com/chapter/10.1007/978-3-319-01056-4_1)
- Gerstenberg, A., Sijman, H., Reime, T., Abrahamsson, P., Steinert, M., 2015. A Simultaneous, Multidisciplinary Development and Design Journey Reflections on Prototyping. In: Chorianopoulos, K., Divitini, M., Baalsrud Hauge, J., Jaccheri, L., Malaka, R. (Eds.), *Entertainment Computing - ICEC 2015*. Vol. 9353. Springer International Publishing, Cham, pp. 409–416.  
URL [http://link.springer.com/10.1007/978-3-319-24589-8\\_33](http://link.springer.com/10.1007/978-3-319-24589-8_33)
- Haines, C. S., Lima, M. D., Li, N., Spinks, G. M., Foroughi, J., Madden, J. D. W., Kim, S. H., Fang, S., Andrade, M. J. d., Gktepe, F., Gktepe, ., Mirvakili, S. M., Naficy, S., Lepr, X., Oh, J., Kozlov, M. E., Kim, S. J., Xu, X., Swedlove, B. J., Wallace, G. G., Baughman, R. H., Feb. 2014. Artificial Muscles from Fishing Line and Sewing Thread. *Science* 343 (6173), 868–872.  
URL <http://science.sciencemag.org/content/343/6173/868>
- Hall, J. A., Knapp, M. L. (Eds.), 2013. Nonverbal communication. No. 2 in *Handbooks of communication science*. De Gruyter Mouton, Berlin ; Boston.
- Hall, T., Lloyd, C., 1990. Non-Verbal Communication in a Health Care Setting. *British Journal of Occupational Therapy* 53 (9), 383–386.  
URL <http://search.ebscohost.com/login.aspx?direct=true&db=ccm&AN=104217292&site=eds-live>
- Harnad, S., 2003. Minds, Machines and Turing. In: Moor, J. H. (Ed.), *The Turing Test*. No. 30 in *Studies in Cognitive Systems*. Springer Netherlands, pp. 253–273, doi: 10.1007/978-94-010-0105-2\_14.  
URL [http://link.springer.com/chapter/10.1007/978-94-010-0105-2\\_14](http://link.springer.com/chapter/10.1007/978-94-010-0105-2_14)
- Ishiguro, H., Jan. 2007. Scientific Issues Concerning Androids. *The International Journal of Robotics Research* 26 (1), 105–117.  
URL <http://ijr.sagepub.com/content/26/1/105>
- Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., Darrell, T., 2014. Caffe: Convolutional architecture for fast feature embedding. arXiv preprint arXiv:1408.5093.

- 
- Leifer, L. J., Steinert, M., 2014. Dancing with Ambiguity: Causality Behavior, Design Thinking, and Triple-Loop-Learning. In: Gassmann, O., Schweitzer, F. (Eds.), Management of the Fuzzy Front End of Innovation. Springer International Publishing, pp. 141–158, doi: 10.1007/978-3-319-01056-4\_11.  
URL [http://link.springer.com/chapter/10.1007/978-3-319-01056-4\\_11](http://link.springer.com/chapter/10.1007/978-3-319-01056-4_11)
- Levine, S., Finn, C., Darrell, T., Abbeel, P., Apr. 2015. End-to-End Training of Deep Visuomotor Policies. arXiv:1504.00702 [cs]ArXiv: 1504.00702.  
URL <http://arxiv.org/abs/1504.00702>
- Mohri, M., Rostamizadeh, A., Talwalkar, A., 2012. Foundations of machine learning. MIT Press, Cambridge, MA, oCLC: 809846149.  
URL <http://public.eblib.com/choice/publicfullrecord.aspx?p=3339482>
- PiezoMotor, 2016. Piezo LEGS Walking Drive Technology.  
URL <http://www.piezomotor.com/how-it-works-2/>
- Smith, P. G., Reinertsen, D. G., Jun. 1992. Shortening the Product Development Cycle. Research Technology Management 35 (3), 44.  
URL <http://search.proquest.com/docview/213814030/abstract/F75E0EFCEE4D2FPQ/1>
- Steinert, Martin, Leifer, Larry J., 2012. Finding Ones Way: Re-Discovering a Hunter-Gatherer Model based on Wayfaring. International Journal of Engineering Education 28 (2), 251–253.
- Viola, P., Jones, M., 2001. Rapid object detection using a boosted cascade of simple features. In: Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001. Vol. 1. pp. I–511–I–518 vol.1.
- Walton, M., Oct. 2016. PSVR vs. HTC Vive vs. Oculus Rift vs. Gear VR: Which VR headset should you buy?  
URL <http://arstechnica.com/gaming/2016/10/best-vr-headset-2016-psvr-rift-vive/>
- Wong, A. M., 2008. Eye Movement Disorders. Oxford University Press, Cary, US.  
URL <http://site.ebrary.com/lib/alltitles/docDetail.action?docID=10273145>

---

---



# Appendices



---

## A Face recognition

### Python code

```
1 import numpy as np
2 import cv2, serial
3
4 #Open Face detection classifier and initiate video
5 cascPath = "/Users/truls/opencv/data/haarcascades/
   haarcascade_frontalface_alt.xml"
6 eyeCascade = cv2.CascadeClassifier(cascPath)
7 eyeCapture = cv2.VideoCapture(1)
8
9 #Capture camera resolution
10 resx = eyeCapture.get(3)
11 resy = eyeCapture.get(4)
12
13 #Connect to Arduino
14 connected = False
15 ser = serial.Serial('/dev/cu.usbmodem1411', 9600)
16 while not connected:
17     serin = ser.read()
18     connected = True
19
20 while True:
21     # Open one frame of the video stream and mirror it
22     _, frame = eyeCapture.read()
23     frame = cv2.flip(frame, 1)
24     #Detect faces in the current frame
25     gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
26     eyes = eyeCascade.detectMultiScale(
27         gray,
28         scaleFactor=1.1,
29         minNeighbors=5,
30         minSize=(50, 50),
31         maxSize=(250,250),
32     )
33     #If faces are present, write its position to the Arduino
34     if len(eyes)>0:
35         xstr = np.zeros((len(eyes)), dtype=np.int)
36         ystr = np.zeros((len(eyes)), dtype=np.int)
37         xstr=(int((eyes.item(0,0))/resx*255))
38         ystr=(int((eyes.item(0,1))/resy*255))
39         ser.write(b"%3d%3d" % (xstr, ystr))
40     # Draw a rectangle around the faces
41     for (x, y, w, h) in eyes:
42         cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)
43     #Display the frame with a rectangle around the face
44     cv2.imshow('EyeCapture', frame)
45
```

---

```
46 if cv2.waitKey(33) & 0xFF == ord('a'):
47     break
48 ser.close()
49 eyeCapture.release()
50 cv2.destroyAllWindows()
```

## Arduino code

```
1 #include <Servo.h>
2 Servo Servo1;
3 Servo Servo2;
4 int rng = 179;
5 char vals[6];
6 String strx;
7 String stry;
8
9 void setup() {
10     Serial.begin(9600);
11     Serial.write('1'); // Connect to Python
12     Servo1.attach(10); // Set Servos to center position
13     Servo1.write(90);
14     Servo2.attach(11);
15     Servo2.write(90);
16 }
17
18 void loop() {
19     while (Serial.available() > 0) { // While the Arduino receives a serial
20         Serial.readBytes(vals, 6); // input the code writes the x-value
21                                     // to one servo, and the y-value
22                                     // to another
23
24         for (int i = 0; i < 3; i++) {
25             strx += vals[i];
26         }
27
28         for (int i = 3; i < 6; i++) {
29             stry += vals[i];
30         }
31
32         int valx = strx.toInt();
33         valx = map(valx, 0, 255, 0, 179);
34         int valy = stry.toInt();
35         valy = map(valy, 0, 255, 0, 179);
36
37         Servo1.write(179-valx);
38         Servo2.write(valy);
39
40         strx = "";
41         stry = "";
42     }
43 }
```

## B Face recognition and audio preparation

### Python code

---

```

1 from sys import bytearray
2 from array import array
3 from struct import pack
4 import numpy as np
5 import cv2, sys, serial, pyaudio, wave, csv, struct, matplotlib
6
7
8 connected = False
9 ser = serial.Serial('/dev/cu.usbmodem1411', 9600)
10 while not connected:
11     serin = ser.read()
12     connected = True
13
14
15 print('input audiofile FILENAME: ')
16 FILENAME = input() #Get filename inputs
17 print('input positioning FILENAME: ')
18 POSFILE = input()
19 print('input audio - csv FILENAME: ')
20 AUDCSV = input()
21
22 #audio
23 THRESHOLD = 100
24 CHUNK_SIZE = 8192
25 FORMAT = pyaudio.paInt16
26 RATE = 44100
27 CHANNELS=2
28 CHANGE_RATE = 1
29
30 #video
31 cascPath = "/Users/truls/opencv/data/haarcascades/
32     haarcascade_frontalface_alt.xml"
33 faceCascade = cv2.CascadeClassifier(cascPath)
34 video_capture = cv2.VideoCapture(1)
35 resx = video_capture.get(3)
36 resy = video_capture.get(4)
37
38 def normalize(snd_data):
39     #Average the volume out
40     MAXIMUM = 16384
41     times = float(MAXIMUM)/max(abs(i) for i in snd_data)
42
43     r = array('h')
44     for i in snd_data:
45         r.append(int(i*times))
46     return r
47
48 def trim(snd_data):
49     #Trim the blank spots at the start and end
50     def _trim(snd_data):
51         snd_started = False
52         r = array('h')
53
54         for i in snd_data:
55             if not snd_started and abs(i)>THRESHOLD:
56                 snd_started = True
57                 r.append(i)

```

---

---

```

57
58         elif snd_started:
59             r.append(i)
60         return r
61
62     # Trim to the left
63     snd_data = _trim(snd_data)
64
65     # Trim to the right
66     snd_data.reverse()
67     snd_data = _trim(snd_data)
68     snd_data.reverse()
69     return snd_data
70
71 def add_silence(snd_data, seconds):
72     #Add silence to the start and end of 'snd_data' of length 'seconds' (
73     float)
74     r = array('h', [0 for i in range(int(seconds*RATE))])
75     r.extend(snd_data)
76     r.extend([0 for i in range(int(seconds*RATE))])
77     return r
78
79 def start_record():
80     p = pyaudio.PyAudio()
81     counter = 0
82     stream = p.open(format=FORMAT, channels=CHANNELS, rate=RATE,
83                     input=True, output=True,
84                     frames_per_buffer=CHUNK.SIZE)
85     r = array('h')
86     position = []
87
88     while True:
89         # Audio recording
90         snd_data = array('h', stream.read(CHUNK.SIZE, exception_on_overflow
91         = False))
92         if byteorder == 'big':
93             snd_data.byteswap() #wav format endianness is "little"
94         r.extend(snd_data)
95         #Face recognition and display video
96         ret, frame = video_capture.read()
97         gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
98         faces = faceCascade.detectMultiScale(
99             gray,
100             scaleFactor=1.1,
101             minNeighbors=5,
102             minSize=(30, 30),
103         )
104         for (x, y, w, h) in faces: # Draw a rectangle around the faces
105             cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)
106
107         if len(faces):
108             xstr = np.zeros((len(faces)), dtype=np.int)
109             ystr = np.zeros((len(faces)), dtype=np.int)
110             xstr=(int((faces.item(0,0))/resx*255))
111             ystr=(int((faces.item(0,1))/resy*255))
112             ser.write(b"%3d%3d" % (xstr, ystr))

```

---

---

```

112     #writing positioning and timing
113     temp_pos = [0]*len(faces)
114     xstr = [x[0] for x in faces]
115     ystr = [y[1] for y in faces]
116     for i in range(0,len(faces)):
117         temp_pos[i] = [int(xstr[i]/resx*255),int(ystr[i]/resy*255)]
118     position.append(temp_pos)
119
120     cv2.imshow('Video', frame) # Display the resulting frame
121     if cv2.waitKey(33) & 0xFF == ord('a'):
122         break
123
124     print('*done recording')
125     sample_width = p.get_sample_size(FORMAT)
126     stream.stop_stream()
127     stream.close()
128     p.terminate()
129
130     r = normalize(r)
131     r = trim(r)
132     r = add_silence(r, 0.5)
133
134     video_capture.release()
135     cv2.destroyAllWindows()
136     return sample_width, r, position
137
138 def record_to_file(path):
139     #Records from the microphone and outputs the resulting data to 'path'
140
141     sample_width, data, position = start_record() #position
142     coordinates: [X0 X1 X2... , Y0 Y1 Y2...]
143
144     data = pack('<' + ('h'*len(data)), *data)
145
146     wf = wave.open(path, 'wb')
147     print('*saving recording and closing')
148     wf.setnchannels(2)
149     wf.setsampwidth(sample_width)
150     wf.setframerate(RATE*CHANGE_RATE)
151     wf.writeframes(data)
152     wf.close()
153     return position
154
155 def pcm_channels(FILENAME):
156     """Given a file-like object or file path representing a wave file,
157     decompose it into its constituent PCM data streams.
158
159     Input: A file like object or file path
160     Output: A list of lists of integers representing the PCM coded data
161     stream channels
162     and the sample rate of the channels (mixed rate channels not
163     supported)
164     """
165     stream = wave.open(FILENAME,"rb")
166
167     num_channels = stream.getnchannels()
168     sample_rate = stream.getframerate()

```

---

---

```

166     sample_width = stream.getsampwidth()
167     num_frames = stream.getnframes()
168
169     raw_data = stream.readframes( num_frames ) # Returns byte data
170     stream.close()
171
172     total_samples = num_frames * num_channels
173
174     if sample_width == 1:
175         fmt = "%iB" % total_samples # read unsigned chars
176     elif sample_width == 2:
177         fmt = "%ih" % total_samples # read signed 2 byte shorts
178     else:
179         raise ValueError("Only supports 8 and 16 bit audio formats.")
180
181     integer_data = struct.unpack(fmt, raw_data)
182     del raw_data # Keep memory tidy (who knows how big it might be)
183
184     channels = [ [] for time in range(num_channels) ]
185
186     for index, value in enumerate(integer_data):
187         bucket = index % num_channels
188         channels[bucket].append(value)
189
190     return channels, sample_rate
191
192 def write_pos_to_csv(position):
193
194     print ('Saved',
195           len(position),
196           'positions to file',
197           POSFILE)
198
199     with open(POSFILE, "w") as f:
200         writer = csv.writer(f)
201         writer.writerow(position)
202
203 def write_channels_to_csv(channels):
204
205     print ('Saved',
206           len(channels),
207           'channels with',
208           int(len(channels[0])/CHUNK_SIZE),
209           'audio chunks of size',
210           CHUNK_SIZE,
211           'to files',
212           FILENAME,
213           'and',
214           AUDCSV)
215
216     with open(AUDCSV, "w") as f:
217         writer = csv.writer(f)
218         writer.writerow(channels)
219
220 position = record_to_file(FILENAME)
221 channels, sample_rate = pcm_channels(FILENAME)
222 ser.close()

```

---



---

```
223 write_pos_to_csv(position)
224 write_channels_to_csv(channels)
```

## C Eye recognition, displaying and actuation

```
1 import numpy as np
2 import cv2, serial
3 import math as m
4 import matplotlib.pyplot as plt
5
6
7 img = cv2.imread("/Users/truls/Pictures/pattern.jpg")
8
9 #video
10 cascPath = "/Users/truls/opencv/data/haarcascades/haarcascade_eye.xml"
11 eyeCascade = cv2.CascadeClassifier(cascPath)
12 eyeCapture = cv2.VideoCapture(0)
13 mannequinEye = cv2.VideoCapture(1)
14 resx = eyeCapture.get(3)
15 resy = eyeCapture.get(4)
16 resx2 = mannequinEye.get(3)
17 resy2 = mannequinEye.get(4)
18
19 connected = False
20 ser = serial.Serial('/dev/cu.usbmodem1411', 9600)
21 while not connected:
22     serin = ser.read()
23     connected = True
24
25 gamma = 1
26
27 #plot initiate
28 plt.ion()
29 plt.hold(False)
30
31 blinklen = 0
32
33 def adjust_gamma(image, gamma):
34     # build a lookup table mapping the pixel values [0, 255] to
35     # their adjusted gamma values
36     invGamma = 1.0 / gamma
37     table = np.array([((i / 255.0) ** invGamma) * 255
38         for i in np.arange(0, 256)].astype("uint8"))
39
40     # apply gamma correction using the lookup table
41     return cv2.LUT(image, table)
42
43
44 while True:
45     #Face recognition and display video
46     eyeFPS = eyeCapture.get(5)
47     mqFPS = mannequinEye.get(5)
48
49
50     ret, frame = eyeCapture.read()
51     frame = cv2.flip(frame, 1)
```

---

```

52
53     frame = adjust_gamma(frame, gamma)
54     frame = cv2.subtract(frame, img)
55     a, frame2 = mannequinEye.read()
56     gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
57     eyes = eyeCascade.detectMultiScale(
58         gray,
59         scaleFactor=1.1,
60         minNeighbors=5,
61         minSize=(50, 50),
62         maxSize=(250,250),
63     )
64
65
66     for (x, y, w, h) in eyes: # Draw a rectangle around the faces
67         cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)
68         x_circ = int((x+w/2)*resx2/resx)
69         y_circ = int((y+h/2)*resy2/resy)
70
71         cv2.circle(frame2, (x_circ, y_circ), 30, (0, 255, 0), 2)
72
73     if len(eyes)>0:
74         xstr = np.zeros((len(eyes)), dtype=np.int)
75         ystr = np.zeros((len(eyes)), dtype=np.int)
76         xstr=(int((eyes.item(0,0))/resx*255))
77         ystr=(int((eyes.item(0,1))/resy*255))
78         ser.write(b"%3d%3d" % (xstr, ystr))
79
80         plt.plot(int(x+w/2), resy-int(y+h/2), 'r.', ms=100)
81         plt.axis([0, resx, 0, resy])
82         plt.show()
83
84         blinklen = 0
85     elif len(eyes) == 0 & blinklen < 12:
86
87         plt.annotate(s='BLINK', xy=[resx/2, resy/2])
88         blinklen = blinklen + 1
89     else:
90         plt.annotate(s='EYE SHUT', xy=[resx/2, resy/2])
91
92     cv2.imshow('EyeCapture', frame) # Display the resulting frame
93     cv2.imshow('MannequinEyes', frame2)
94     if cv2.waitKey(33) & 0xFF == ord('a'):
95         break
96
97     ser.close()
98     eyeCapture.release()
99     mannequinEye.release()
100    cv2.destroyAllWindows()

```

THE NORWEGIAN UNIVERSITY  
OF SCIENCE AND TECHNOLOGY  
DEPARTMENT OF ENGINEERING DESIGN  
AND MATERIALS

**PROJECT WORK FALL 2016  
FOR  
STUD.TECHN. Truls Hjertnes Nygaard**

**Lærdal Medical stand alone (Wireless) baby mannequin that can be used in medical teaching**

Identify, collect, judge, and conceptualize technologies to simulate human physiology – e.g. eye movement/follow, blinking, pupil dilation to light eye communication, or reactions to external influences through the skin.

- generate concepts,
- build prototypes
- build test setups
- test and compare alternatives
- judge concept

Also, it is expected to contribute to one or more scientific publications during the project/master thesis.

The supporting coach is Carlo Kriesi, the contact at Lærdal Medical is Arild Eikefjord. Also the student shall collaborate with Henrik Aasheim and Knut Åsland who are conducting the second Lærdal challenge.

**Formal requirements:**

Students are required to submit an A3 page describing the planned work three weeks after the project start as a pdf-file via "IPM DropIT" (<http://129.241.88.67:8080/Default.aspx>). A template can be found on IPM's web-page (<https://www.ntnu.edu/ipm/project-and-specialization>).

Performing a risk assessment is mandatory for any experimental work. Known main activities must be risk assessed before they start, and the form must be handed in within 3 weeks after you receive the problem text. The form must be signed by your supervisor. Risk assessment is an ongoing activity, and must be carried out before starting any activity that might cause injuries or damage materials/equipment or the external environment. Copies of the signed risk assessments have to be put in the appendix of the project report.

No later than 1 week before the deadline of the final project report, you are required to submit an updated A3 page summarizing and illustrating the results obtained in the project work.

Official deadline for the delivery of the report is 13 December 2016 at 2 p.m. The final report has to be delivered at the Department's reception (1 paper version) and via "IPM DropIT".

When evaluating the project, we take into consideration how clearly the problem is presented, the thoroughness of the report, and to which extent the student gives an independent presentation of the topic using his/her own assessments.

The report must include the signed problem text, and be written as a scientific report with summary of important findings, conclusion, literature references, table of contents, etc. Specific problems to be addressed in the project are to be stated in the beginning of the report and briefly discussed. Generally the report should not exceed thirty pages including illustrations and sketches.

Additional tables, drawings, detailed sketches, photographs, etc. can be included in an appendix at the end of the thirty page report. References to the appendix must be specified. The report should be presented so that it can be fully understood without referencing the Appendix. Figures and tables must be presented with explanations. Literature references should be indicated by means of a number in brackets in the text, and each reference should be further specified at the end of the report in a reference list. References should be specified with name of author(s) and book, title and year of publication, and page number.

Contact persons:

At the department	Martin Steinert, Carlo Kriesi
From the industry	Arild Eikefjord



Martin Steinert  
Supervisor



**NTNU**  
Norges teknisk-  
naturvitenskapelige universitet  
Institutt for produktutvikling  
og materialer

NTNU



HIMS

## Kartlegging av risikofylt aktivitet

Utlærbeidet av	Nummer	Dato
HMS-avd.	HMSRV2601	22.03.2011
Godkjent av		Erstatter
Rektor		01.12.2006

**Enhet:** Department of engineering design and materials

**Linjeleder:** Torgeir Welo

**Deltakere ved kartleggingen (m/ funksjon):** Carlo Kriesi, veileder./ Martin Steinert, veileder./ Truls Nygaard, student.  
(Ansv. veileder, student, evt. medveiledere, evt. andre m. kompetanse)

**Kort beskrivelse av hovedaktivitet/hovedprosess:** Prosjektoppgave student Truls Nygaard.

**Er oppgaven rent teoretisk? (JA/NEI):**

risikovurdering. Dersom «JA»: Beskriv kort aktiviteteten i kartleggingskjemmet under. Risikovurdering trenger ikke å fylles ut.

**Signaturer:** Ansv. veileder: Martin Steinert


Student: Truls Nygaard

**Dato:** 2015.09.03



ID nr.	Aktivitet/prosess	Ansv. veileder	Eksisterende dokumentasjon	Eksisterende sikringsiltak	Lov, forskrift o.l.	Kommentar
1	Bruk av Trolllabs workshop.	DM	Romkort	Romkort		
1a	Bruk av roterende maskineri	DM	Maskinens brukermanual	Ukjent	Ukjent	
1b	Bruk av laserkutter	DM	Maskinens brukermanual	Ukjent	Ukjent	
1c	Bruk av 3D printer	DM	Maskinens brukermanual	Ukjent	Ukjent	
1d	Bruk av skjæreverktøy	DM	Ukjent			
1e	Bruk av sammenføyningsmidler (lim og lignende.)	DM	Produktets brukermanual og datablad	Datablad	Ukjent	
2	Tilstedeværelse ved arbeid utført av andre.	Andre	Andres HMSRV2601	Andres HMSRV2601	Prosessåvhengig	

NTNU	Kartlegging av risikofylt aktivitet	Utarbeidet av	Nummer	Dato	
		HMS-avd.	HMSRV2601	22.03.2011	
HMS		Godkjent av		Erstatter	
		Rektor		01.12.2006	

3	Ekperimentelt arbeid	DM	Egen risikovurdering- må gjøres for hvert enkelt eksperiment		Prosessavhengig	
---	----------------------	----	--	--	-----------------	--



NTNU	Risikovurdering	Utarbeidet av	Nummer	Dato	
		HMS-avd.	HMSRV2601	22.03.2011	
HMS		Godkjent av		Erstatter	
		Rektor		01.12.2006	

ID nr	Aktivitet fra kartleggings-skjemaet	Mulig uønsket hendelse/belastning	Vurdering av sannsynlighet (1-5)	Vurdering av konsekvens:				Risiko-Verdi (menneske)	Kommentarer/status Forslag til tiltak
				Menneske (A-E)	Ytre miljø (A-E)	Øk/materiell (A-E)	Om-dømme (A-E)		
1	Bruk av Trolllabs workshop.								
1a-i	Bruk av roterende maskineri	Stor kuttskade	2	D	A	A	D	2D	Sørg for at roterende deler tilstrekkelig sikret/dekket. Vær nøye med opplæring i bruk av maskineri.
1a-ii		Liten kuttskade	3	B	A	A	A	3B	Vær nøye med opplæring i bruk av maskineri. Ikke ha løse klær/tilbehør på kroppen.
1a-iii		Klemskade	2	D	A	A	C	2D	Vær nøye med opplæring i bruk av maskineri. Ikke ha løse klær/tilbehør på kroppen.
1a-iv		Flygende spon/gjenstander	3	C	A	A	B	3C	Bruk øyevern og tildekk hurtig roterende deler (Fres og lignende.)
1a-v		Feil bruk-> ødelagt utstyr	3	A	A	C	A	3C	Vær nøye med opplæring i bruk av maskineri
1b-i	Bruk av laserkutter	Klemskade	2	D	A	A	C	2D	Vær nøye med opplæring i bruk av maskineri. Ikke ha løse klær/tilbehør på kroppen.
1b-ii		Brannskade	3	B	A	A	A	3B	Vær nøye med opplæring i bruk av maskineri. Bruk hansker ved håndtering av varme materialer.

 NTNU HMS	<b>Risikovurdering</b>	Utarbeidet av	Nummer	Dato	
		HMS-avd.	HMSRV2601	22.03.2011	
		Godkjent av		Erstatter	
		Rektor		01.12.2006	

1b-iii		Øyeskade-laser	2	D	A	A	C	2D	Bruk øyevern! Skru av laser når maskinen ved oppsett.
1b-iv		Brann	2	B	A	D	C	2B	Vær nøye med opplæring i bruk av maskin. Ha slukkeutstur tilgjengelig
1c-i	Bruk av 3D-printer	Brannskade	3	B	A	A	A	3B	Vær nøye med opplæring i bruk av maskin.
1c-ii		Innhaling av plast/printemateriale	5	A	A	A	A	5A	Bruk åndedrettsvern/ vernebriller
1c-iii		Feil bruk-> ødelagt maskineri	3	A	A	C	A	3A	Vær nøye med opplæring i bruk av maskin.
1d-i	Bruk av skjæreverktøy	Stor kuttskade	2	D	A	A	D	2D	Bruk skapre verktøy og riktig skjæreunderlag
1d-ii		Liten kuttskade	3	B	A	A	A	3B	Bruk skapre verktøy og riktig skjæreunderlag
1e-i	Bruk av samentøyningsmidler (lim og lignende.)	Eksposering på øyet	2	D	A	A	B	2D	Bruk øyevern, ha datablad tilgjengelig
1e-ii		Eksposering hud	4	A	A	A	A	4A	Bruk hansker, ha datablad tilgjengelig
1e-iii		Eksposering åndedrett	4	A	A	A	A	4A	Bruk åndedrettsvært/ god ventilasjon. Ha datablad tilgjengelig.
1e-iv		Søl	4	A	B	A	A	4A	Ha papir/ rengjøringsmateriell tilgjengelig. Ha datablad



NTNU	<b>Risikovurdering</b>	Utarbeidet av	Nummer	Dato	
		HMS-avd.	HMSRV2601	22.03.2011	
HMS		Godkjent av		Erstatter	
		Rektor		01.12.2006	

									tilgjengelig.
2	Tilstedeværelse ved arbeid utført av andre.	Se andres risikovurdering om sikkerhet betviles.	3	C	C	C	C	3C	Hold et øye med hva som foregår rundt deg.
3-i	Eksperimentelt arbeid	Vann-drukning	1E	A	A	A	D	1E	Bruk redingsvest i båt og lignende.
3-ii		Elektrisitet- strøm	3	B	A	A	A	3V	Typisk lite energi involvert. Bruk isolerte verkøty

NTNU	Risikovurdering	Utarbeidet av	Nummer	Dato	
		HMS-avd.	HMSRV2601	22.03.2011	
HMS		Godkjent av		Erstatter	
		Rektor		01.12.2006	

### Sannsynlighet vurderes etter følgende kriterier:

Svært liten 1	Liten 2	Middels 3	Stor 4	Svært stor 5
1 gang pr 50 år eller sjeldnere	1 gang pr 10 år eller sjeldnere	1 gang pr år eller sjeldnere	1 gang pr måned eller sjeldnere	Skjer ukentlig

### Konsekvens vurderes etter følgende kriterier:



Gradering	Menneske	Ytre miljø Vann, jord og luft	Øk/materiell	Omdømme
<b>E</b> Svært Alvorlig	Død	Svært langvarig og ikke reversibel skade	Drifts- eller aktivitetsstans >1 år.	Troverdighet og respekt betydelig og varig svekket
<b>D</b> Alvorlig	Alvorlig personskade. Mulig uførhet.	Langvarig skade. Lang restitusjonstid	Driftsstans > ½ år Aktivitetsstans i opp til 1 år	Troverdighet og respekt betydelig svekket
<b>C</b> Moderat	Alvorlig personskade.	Mindre skade og lang restitusjonstid	Drifts- eller aktivitetsstans < 1 mnd	Troverdighet og respekt svekket
<b>B</b> Liten	Skade som krever medisinsk behandling	Mindre skade og kort restitusjonstid	Drifts- eller aktivitetsstans < 1uke	Negativ påvirkning på troverdighet og respekt
<b>A</b> Svært liten	Skade som krever førstehjelp	Ubetydelig skade og kort restitusjonstid	Drifts- eller aktivitetsstans < 1dag	Liten påvirkning på troverdighet og respekt

### Risikoverdi = Sannsynlighet x Konsekvens

Beregn risikoverdi for Menneske. Enheten vurderer selv om de i tillegg vil beregne risikoverdi for Ytre miljø, Økonomi/materiell og Omdømme. I så fall beregnes disse hver for seg.

### Til kolonnen "Kommentarer/status, forslag til forebyggende og korrigerende tiltak":

Tiltak kan påvirke både sannsynlighet og konsekvens. Prioriter tiltak som kan forhindre at hendelsen inntreffer, dvs. sannsynlighetsreducerende tiltak foran skjerpet beredskap, dvs. konsekvensreducerende tiltak.

NTNU	<b>Risikomatrise</b>	utarbeidet av	Nummer	Dato	
		HMS-avd.	HMSRV2604	08.03.2010	
HMS/KS		godkjent av		Erstatter	
		Rektor			

### MATRISSE FOR RISIKOVURDERINGER ved NTNU

<b>KONSEKVENSENS</b>	Svært alvorlig	<b>E1</b>	<b>E2</b>	<b>E3</b>	<b>E4</b>	<b>E5</b>
	Alvorlig	<b>D1</b>	<b>D2</b>	<b>D3</b>	<b>D4</b>	<b>D5</b>
	Moderat	<b>C1</b>	<b>C2</b>	<b>C3</b>	<b>C4</b>	<b>C5</b>
	Liten	<b>B1</b>	<b>B2</b>	<b>B3</b>	<b>B4</b>	<b>B5</b>
	Svært liten	<b>A1</b>	<b>A2</b>	<b>A3</b>	<b>A4</b>	<b>A5</b>
		Svært liten	Liten	Middels	Stor	Svært stor
		<b>SANNSYNLIGHET</b>				

Prinsipp over akseptkriterium. Forklaring av fargene som er brukt i risikomatrisen.

Farge	Beskrivelse
Rød	Uakseptabel risiko. Tiltak skal gjennomføres for å redusere risikoen.
Gul	Vurderingsområde. Tiltak skal vurderes.
Grønn	Akseptabel risiko. Tiltak kan vurderes ut fra andre hensyn.