3D printing of optomechanical elements

3D printede optomekaniske komponenter

Master thesis in Physics
Søren Bobach
20104244

Supervisor: Jan Arlt 5. August 2019

Institute of Physics and Astronomi
Aarhus University

I would like to thank the Ultracold Quantum Gases Group at Aarhus University for having me and for all the help during my time in the group. I would particularly like to thank Jan Arlt for being my supervisor and always be available for questions and discussions. I would also like to thank Mick Kristensen for being my daily supervisor and helping me with all the things that needed to explained or fixed. Finally, I would also like to thank my farther and brother for helping me to learn to programme in C/C++ and for being available when I had programming issues.

Institut for Fysik og Astronomi
Aarhus Universitet
Ny Munkegade, Bygning 1520
DK-8000 Aarhus C
Danmark

Abstract

Lasers are widely used in physics laboratories for e.q. cooling atoms and manipulating atoms. To isolate these lasers from their applications, to improve beam quality and to deliver the light where it is needed they are often placed separately. Often the laser beam is at some point coupled into an optical fibre such that the laser beam is transported to where it is needed. The coupling into an optical fibre can be a difficult and tedious work when the coupling efficiency needs to be high. Over time the coupling efficiency drops due to e.g. temperature changes in the mirrors used to couple the laser beam into the optical fibre. This leads to the need for recouple the laser beam often which takes time. This process can be automated if motors are installed on the mirror mounts. There already exist commercial solutions to install motors on the mirror mounts however, the high price of these solutions does not allow them to be installed when many fibres are required. This thesis provide a much cheaper solution to the problem of coupling the laser beam into the optical fibre and maintaining a high coupling efficiency.

Within this thesis I present my realization of an automated kinematic mirror mount where the mirror mount is designed and 3D printed. The mirror mount uses cheap step motors which are controlled by software that I have developed through my work.

First, I present the different mirror mount designs that I have developed and tested through my work. I argue that two of my designs can be used for realizing an automated kinematic mirror mount whereas the last designs have issues that do not make them sufficiently reliable in the present state.

Secondly, I present the two algorithms that I have designed such that the automated kinematic mirror mounts are able to couple the laser beam into the optical fibre with a high efficiency and then maintain the high efficiency over time. This is achieved with two different algorithms that are programmed on an ”Arduino” microcontroller which controls the mirror mounts. I also developed an interface to make it easy to control the mirror mounts manually and activate the two algorithms when needed without the use of a computer.

Finally, I present the testing of the two algorithms and the results from these. The results shows that the algorithms are able to achieve a high coupling efficiency in most of the cases and that the efficiency can be maintained over a longer period than if the algorithm was not activated.

Resumé

Lasere bruges meget udbredt i fysiklaboratorier til fx at køle og manipulere atomer. For at isolere laserne fra deres eksperimenter, for at forbedre kvaliteten og for at kunne levere laserstrålen, hvor den skal bruges er laseren ofte placeret seperat. Ofte bliver laserne på et tidspunkt koblet ind i en optisk fiber så laserstrålen transporteres derhen, hvor den skal anvendes. Koblingen ind i den optiske fiber kan være et svært og besværligt arbejde, når der skal opnås en høj koblingseffektivitet. Over tid falder koblingseffektiviteten på grund af fx temperaturændringer i spejlene, der bruges til at koble laserstrålen ind i fiberen. Dette medfører, at laserstrålen skal rekobles ofte hvilket tager tid. Denne proces kan automatiseres, hvis der installeres motorer på spejlene. Der findes allerede kommercielle løsninger, hvor motorer installeres på spejlene, men den høje pris på disse løsninger gør at det ikke er muligt at anvende, når der kræves mange optiske fibre. Denne afhandling kommer med en meget billigere løsning på, hvordan koblingen af laserstrålen ind i en optisk fiber kan automatiseres og derefter bibeholde en høj koblingseffektivitet.

I denne afhandling præsenterer jeg min realisation af en automatiseret kinematisk spejlholder, hvor spejlholderen er designet og 3D printet. Spejlholderen bruger billige stepmotorer, som styres af software jeg har udviklet under mit arbejde.

Først præsenterer jeg de forskellige design af spejlholdere, som jeg har udviklet og testet under mit arbejde. Jeg argumenterer for, at to af mine designs kan bruges til at realisere en automatiseret kinematiske spejlholder, hvorimod de sidste designs har problemer, som gør at de ikke er tilstrækkelige stabile i deres nuværende tilstand.

Dernæst præsenterer jeg de to algoritmer, som jeg har designet sådan, at den automatiserede kinematiske spejlholder kan koble laserstrålen ind i den optiske fiber med en høj effektivitet og dernæst bibeholder den høje intensitet over tid. Dette opnås med to forskellige algoritmer, som er programmeret på Arduinoen, der styrer spejlholderne. Jeg har også udviklet en brugerflade, for at gøre det nemmere at styre spejlholderne manuelt og aktivere de to algoritmer, når det er nødvendigt, uden brug af en computer.

Til sidst præsenterer jeg testene af de to algoritmer og resultaterne fra disse. Resultaterne viser at algoritmerne kan opnå en høj koblingseffektivitet i de fleste tilfælde og at effektiviteten kan bibeholdes over længere tid end, hvis algoritmen ikke var aktiveret.

Contents

Abstract **i**

Resumé **iii**

Contents **v**

1 Introduction **1**

2 Experimental setup **3**
2.1 Laser . . . 3
2.2 Optical components . . . 5
2.3 Optical isolator . . . 7
2.4 The final setup . . . 9

3 Automated kinematic mirror mounts **13**
3.1 Prototype mirror mount . . . 14
3.2 Second version of the automated kinematic mirror mount . . . 17
3.3 Automated vertical mirror mount . . . 22
3.4 Automated mirror box . . . 28

4 Controlling software and hardware **31**
4.1 Programming . . . 31
4.1.1 Labview . . . 31
4.1.2 Arduino programming . . . 36
4.2 Beam walk algorithm . . . 39
4.3 Constant power algorithm . . . 40
4.4 Interface . . . 41

5 Experimental results **45**
5.1 Beam walk . . . 45
5.2 Constant power . . . 48

6 Conclusion and outlook **51**

Bibliography **53**

A Appendix **55**

1 | Introduction

The realization of the laser in 1960 by Theodore H. Maiman led to new ways of doing experiments and unlocked numerous possibilities to examine new theories. Along with the development of optical elements to control the laser beam, the laser has been used extensively in experimental setups and the optical elements are necessary such that the laser beam can be manipulated in any way scientists wants to do. Laser beams are both useful and powerful and therefore they can be very dangerous to work with. Therefore, in these cases an optical fibre to transport the laser beam, to where it is needed, is desirable. This leads to the complication of coupling the laser beam into the optical fibre. The coupling is very sensitive to misalignment in both angle and position as shown by Wagner and Tomlinson in 1981 [1]. To control this there is a need for using mirrors to secure that the laser beam couples into the optical fibre.

In the past decade 3D printing technology has advanced to be a fast and cheap technique to use when developing new instruments, which has made the research in the field of laser physics more available to the communities in the developing world [2]. With the increase in precision in 3D printing it has got the interest of scientists because it allows for easy and quick access to develop and testing new designs or even print spare parts [3]. This has allowed for the development for making scientific instruments like colorimeters [4], smartphone spectrometers [5] and syringe pumps [6] available to a wider audience. It has also triggered a development in 3D printed optomechanical elements with an on-line database of files available [7]. When a part is needed the file can be downloaded and printed thus saving time waiting on commercial parts being delivered.

Scientists in laser laboratories have been interested in developing 3D printed optical translation stages, optical choppers [7], polarimeters [8] and kinematic mirror mounts [9]. Zhang et al. (2013) have designed and 3D printed a variety of optomechanical components and finds that they have good performance and they have a cost save of 83 % to 99 % as compared to their commercial counterparts. In recent years the precision in 3D printing has made it possible to use 3D parts when working with ultra cold quantum gases [10], where an atom chamber is developed. Also the precision along with high costs have led to the interest in developing a cheap Arduino-based automated kinematic mirror mount based on existing commercial kinematic mirror mounts [11].

The Ultracold Quantum Gases group uses lasers and mirrors to cool and manipulate atoms in all their experiments and a typical experiment involve hundreds of mirrors thus 3D printing offers cost and time saving for the group. The group prepares lasers on one table and transfers the beams through optical fibres to where they are needed which means the group spends lots of time coupling lasers into optical fibres.

This project aims to develop a 3D printed automated kinematic mirror mount based on the previous work of R.Norup who was a former bachelor student in the group working on developing 3D printed kinematic mirror mounts [12]. The project should develop an automated kinematic mirror mount that can be used for optimising the coupling of a laser beam into an optical fibre and maintain a high coupling efficiency over time. Scientists in the group often need to realign the coupling and therefore this project will free up time and show that it can be done at a low cost as compared to the commercial systems.

The thesis is structured as follows:

- In chapter 2 the experimental apparatus and the setup for this project is explained.

- Chapter 3 describes the development of the various versions of the 3D printed models for the automated kinematic mirror mounts. This includes measures of basic performance and difficulties.

- Chapter 4 explains the development of the necessary software for making the automated kinematic mirror mount with the desired abilities. This include the development of programs in both the LabVIEW and Arduino software.

- Chapter 5 describes the experiments made with the automated kinematic mirror mount to investigate the ability to achieve high coupling efficiency and maintaining it, and the results are presented and discussed.

- A conclusion of my work developing the automated kinematic mirror mount is given in chapter 6 along with an outlook.

2 | Experimental setup

The work done towards this thesis was carried out in the preparation lab of the Ultracold Quantum Gases group where all the needed equipment was already set up including a functioning laser system that could provide the needed beam. Therefore, the aim of this chapter is to give an introduction to the equipment used and an explanation of the laser system.

The chapter is structured as the following. First, a presentation and an explanation of the existing laser system that provided the laser beam for the work done towards this thesis is presented in section 2.1. Subsequently, the optical components used to manipulate the laser beam are described in section 2.2. Then the isolator and the issues with it are described in section 2.3. Finally, the experimental setup used for the work of this thesis is described in section 2.4.

2.1 Laser

The laser beam needed in this project was delivered from an existing laser setup that delivers laser beams for three different experimental setups via optical fibres. The laser is based on a commercial laser diode that is optically stabilized by feedback from a diffraction grating like the system described by Ricci et al. (1995) [13]. The laser was built by Henrik Kjær Andersen in his bachelor thesis [14] within the group and the schematics of the mechanical setup is shown in fig. 2.1.

A diode laser is a semiconductor with a p-n junction in which a laser beam is formed. By applying voltage across the laser diode recombination between electrons and holes is allowed which results in emitting a photon. This photon can cause stimulated emission and if the diode is designed properly the gain, when the photons travels around inside the diode, will be greater than the losses due to absorption and incomplete reflection. If the gain is greater than the loss, then the diode starts to lase. The geometry of the optical cavity that surrounds the gain region determines the properties of the laser diode. Many diode lasers have an optical cavity that supports lasing in multi-mode to ensure high power output. If the diode laser is connected to an external cavity new features are possible. The external cavity can reduce the linewidth of the laser, tune the wavelength and be used for mode locking.

The diode laser used to build the laser shown in fig. 2.1 was a 785 nm, 90 mW, Ø5.6 mm diode laser from Thorlabs. This diode laser operates at a wavelength in the interval $\lambda = 775\,\text{nm} - 795\,\text{nm}$ at a temperature of $T = 25\,^\circ\text{C}$.

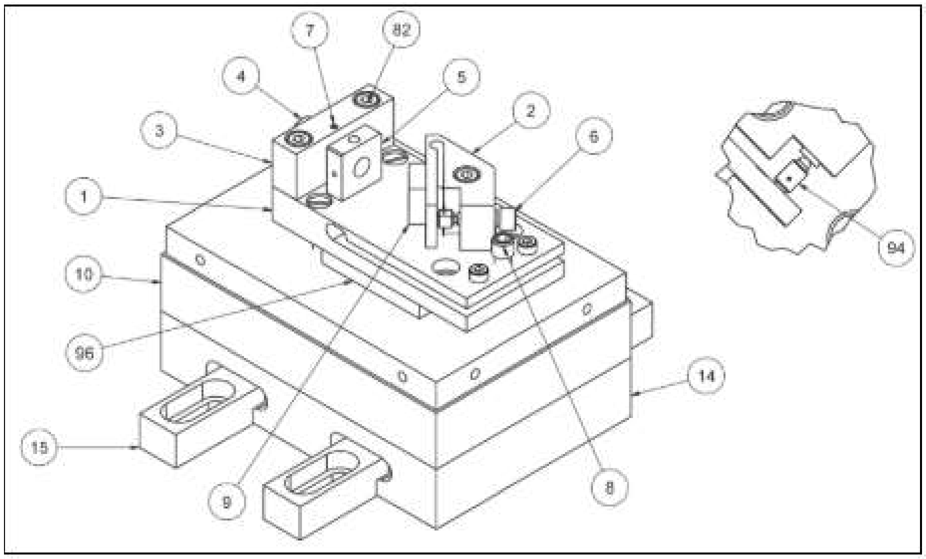


Figure 2.1: Schematics of the mechanical setup of the grating-stabilized diode laser system. The laser was previously designed in the Aarhus group by H.K. Andersen. The figure is from the bachelor thesis [14] and the details of the numbers can be found there.

The grating in fig. 2.1 and the rear back of the diode laser forms an external cavity that is optically coupled to the laser diode cavity. The grating is used to single out the desired operating wavelength of the laser system and make the linewidth narrower. The grating is placed in an external cavity Littrow configuration. The diffraction from a grating is given by:

$$d\left(\sin\theta_i - \sin\theta_m\right) = m\lambda \tag{2.1}$$

Here d is the diffraction constant, θ_i is the angle of the incident beam and θ_m is the angle of the m'th order reflected beam. The grating is placed such that the zeroth order is coupled out at an angle of 90° as compared to the incoming beam giving $\theta_i + \theta_0 \simeq 90$. The first order reflected beam is used as feedback into the diode laser which led to $\theta_i = -\theta_1$. Then according to eq. (2.1) to get a wavelength of $\lambda \simeq 780\,\text{nm}$ a diffraction constant of $d = \frac{1}{1800}$ is chosen.

The temperature poses as a threat to get stabilized laser operation since temperature fluctuations will result in cavity length fluctuations, which then will lead to changes in operating wavelength. That is why a temperature control system was developed to keep the temperature constant [14].

The outcome of the described laser system is a 780 nm laser beam that can be used for experiments. This laser beam is then manipulated with different optical components and then coupled into three different optical fibres. Some of these optical components will be described in the following section.

2.2 Optical components

Mirrors

To be able to control the laser beam it is necessary to use mirrors. It takes two mirrors to control the four degrees of freedom $(x, y, \theta_x, \theta_y)$, that a propagating laser beam has. The mirrors used are dielectric mirrors made for a wavelength of 780 nm.

Dielectric mirrors consist of multiple thin layers of dielectric material, see fig. 2.2. If the type and thickness of the layers are chosen carefully the mirror will have a reflectivity of $R > 99\,\%$ for specific wavelengths. The functioning of the mirrors is based on the interference of light reflected from the different layers of dielectric material with different refractive index. The law of reflection is:

$$\theta_i = \theta_r \tag{2.2}$$

Where θ_i and θ_r are the angle of the incoming and reflected beam. Since the laser beam is also transmitted and this happens in a different media Snell's law gives the relationship between the angle of the incoming and transmitted beam.

$$n_1 \sin\theta_i = n_2 \sin\theta_t \tag{2.3}$$

The electric field of a laser beam can far from the source be described as plane waves, where the field is given by

$$\vec{E}\left(\vec{r}, t\right) = \vec{E}_{\vec{k}} e^{i\left(\vec{k}\cdot\vec{r} - \omega t\right)} \tag{2.4}$$

For a dielectric material the reflection coefficients for s- and p-polarized light can be calculated from the Fresnel equations [15] giving:

$$\begin{aligned} r_s &= \frac{n_1 \cos\theta_i - n_2 \cos\theta_t}{n_1 \cos\theta_i + n_2 \cos\theta_t} \\ r_p &= \frac{n_2 \cos\theta_i - n_1 \cos\theta_t}{n_2 \cos\theta_i + n_1 \cos\theta_t} \end{aligned} \tag{2.5}$$

The thickness of the different layers are chosen to optimise the reflectivity thus giving such a high performance. This is done to make sure that the reflected beams are in phase when interfering with each other. This is achieved by having the path length differences for reflections from different high index layers being equal to an integer multiple of the wavelength wanted. The thickness of the low index layers should ensure that the path length difference for reflections from different low index layers is equal to half a wavelength. This is only half a wavelength since reflections from a low-to-high boundary suffer a 180° phase shift compared to reflections from high-to-low boundaries which can be seen from from eq. (2.5). So the path length difference of half a wavelength ensures that the beams are in phase when interfering.

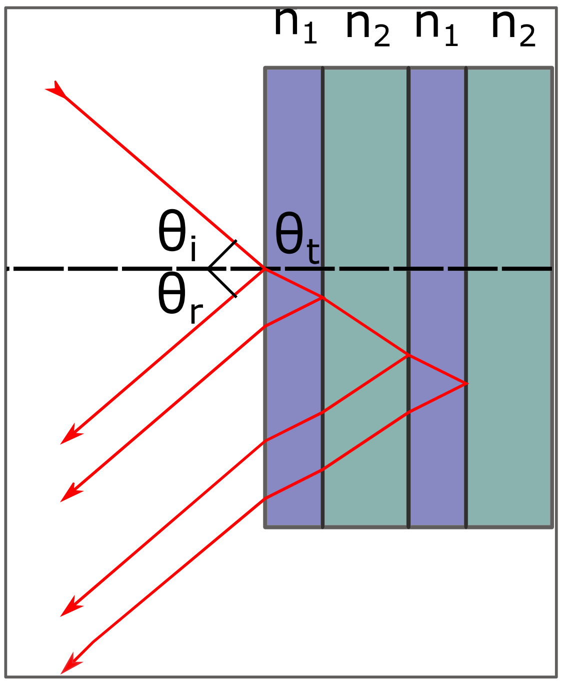


Figure 2.2: Schematics of how the dielectric mirrors work.

Sampler

Another optical component that is based on the Fresnel reflection and is used in this experiment is the optical sampler. The ones used are from Thorlabs with fused silica substrate. The sampler is made such that it only reflects a small portion of the laser beam by taking advantage of eq. (2.5) and eq. (2.3). By using these two formulas and calculating the reflectivity for different incident angles, θ_i, one can see how the optic sampler reflects at different angles. The calculations are done with $n_1 = 1$ and $n_2 = 1.4585$ which represents air and silica, and the results are shown in fig. 2.3.

From the results shown in fig. 2.3 it is easy to see that for an incident angle of $45°$ less than $10\,\%$ of the laser beam will be reflected by the sampler. This allows for selecting a small sample of the beam and use it for other things.

Optical fibre

To transport the laser beam to the experiment used in this project an optical fibre is used. Optical fibres offers a good way to transport the laser beam to where it is needed without posing a danger to the scientists working in the laboratory. The optical fibre used in this project is a single-mode polarizing maintaining optical fibre from Thorlabs.

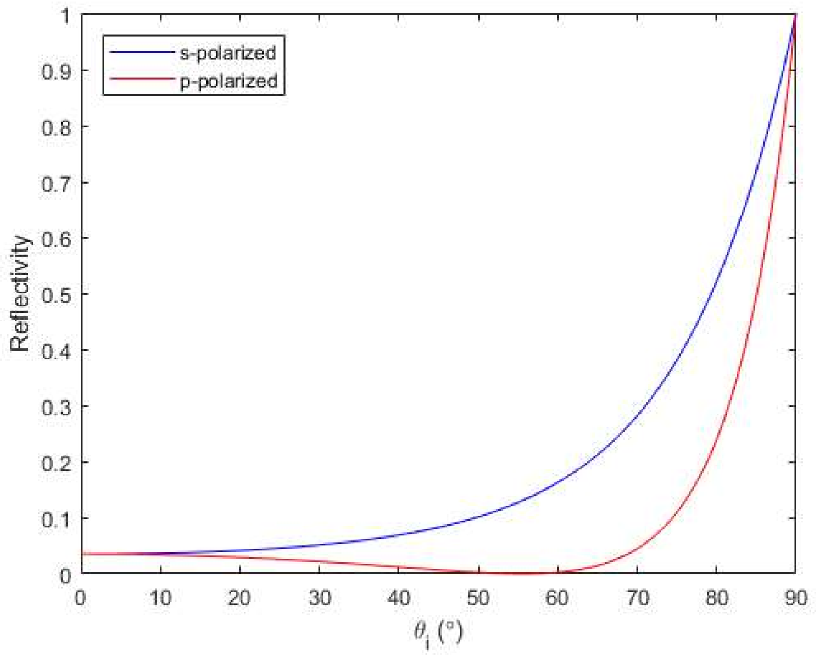


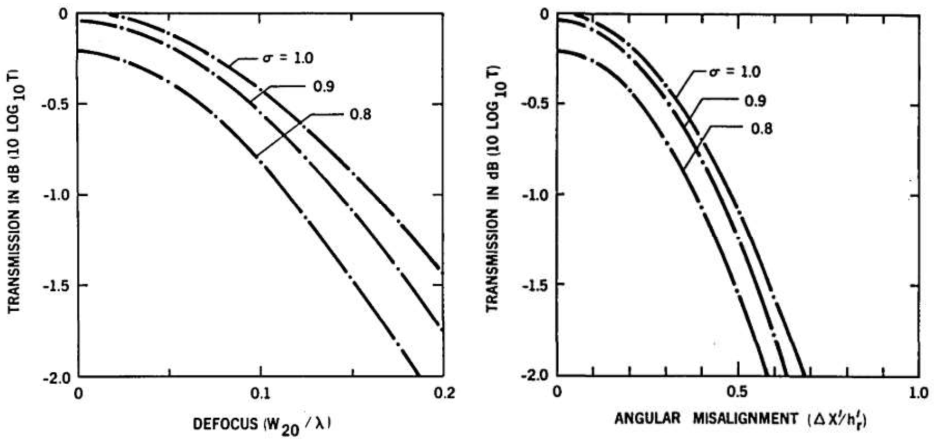
Figure 2.3: Calculations of the reflectivity of the sampler from Thorlabs for different incident angles.

Coupling into optical fibres always exhibits loss due to aberrations and misalignment [1] and therefore one will always loose some of the laser beam in the progress. Linear and angular misalignment leads to rapid losses in coupling efficiency as shown in fig. 2.4 which is from Tomlinson et al. (1981) [1]. The results shown in fig. 2.4 clearly shows that the coupling of a laser beam into an optical fibre is a sensitive task since the efficiency rapidly decrease with both linear and angular misalignment. This shows the premise for the work done in this project and why this can be important work.

2.3 Optical isolator

The last optical component described here is the optical isolator which is a component that only allows transmission of light in one direction. This is used to prevent unwanted feedback into the laser, which would give rise to noise in the signal because it would interfere with the feedback system explained in section 2.1.

The isolators used in this project are polarization dependent isolators which works by utilizing the Faraday effect. The Faraday effect is an interaction between light and the magnetic field in a medium which causes the



(a) (b)

Figure 2.4: How the coupling efficiency of an optical fibre decrease with (a) linear defocus and (b) angular misalignment. The figure is from Tomlinson et al. (1981) [1]. σ in both figures represents the mismatch between the sizes of the laser beam and the optical fibre. W_{20} represents wave-front aberrations occurring from focal shift and field curvature. $\Delta X'/h'_r$ represents the lateral shift of the laser beam distribution occurring when there is an angular misalignment.

polarization of the light to rotate. This happens because left and right hand circular polarized light propagates with slightly different speeds in certain media. Linear polarized light can be decomposed in those two components, thus it is always relevant to look at these components. So the phase shift between these components resulting from the different speeds gives rise to a rotation of polarization. The angle of rotation that the light experiences is given by:

$$\beta = \mathcal{V}Bd \tag{2.6}$$

Where β is the angle of rotation, $\mathcal{V}$ is the Verdet constant which is a property of the material, B is the magnetic field and d is the length of the medium.

The polarizing dependent isolator is build up of three different parts; an input polarizer, a Farady rotator and an output polarizer. The input polarizer makes sure that incoming light is polarized vertically (chosen for simplicity). The Faraday rotator is a medium that rotates the polarization as described above. The length and material of the rotator are chosen such that the polarization of the light is rotated 45°. The output polarizer is placed with its transmission axis with an angle of 45° with the vertical axis, thus ensuring that the forward going light is transmitted. This means that if light is coming back into the isolator the output polarizer ensures that the polarization is at 45°.

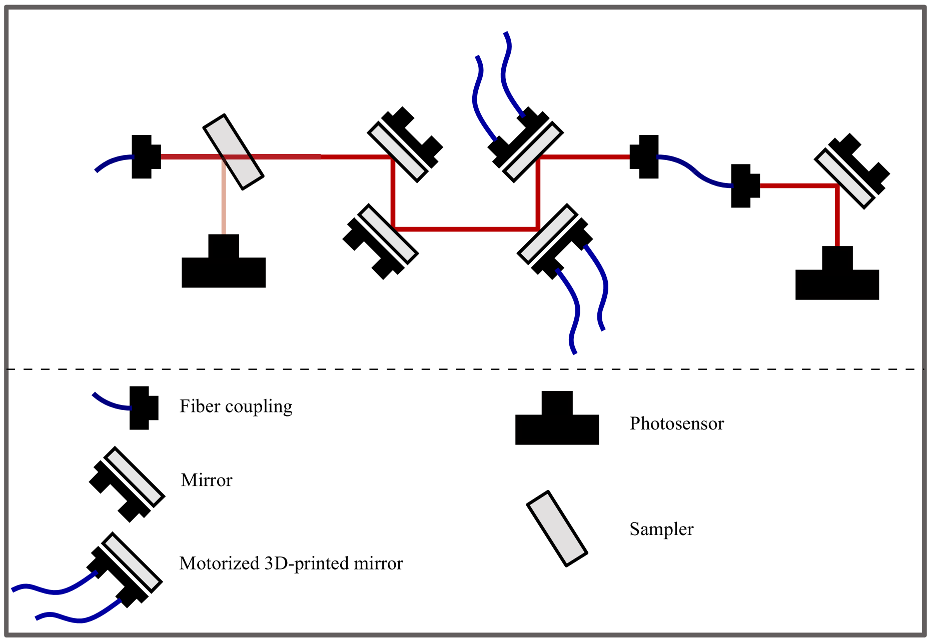


Figure 2.5: The experimental setup for testing and doing experiments with the automated kinematic mirror mounts.

Then the rotator rotates the polarization 45° more, such that the polarization is now horizontal. So when the backward going light hits the input polarizer none of the light is transmitted through. This way the isolator ensures that no feedback is coming through.

2.4 The final setup

The experimental setup for testing and developing the motorized mirror mounts and algorithms is presented in fig. 2.5. The setup is designed such that the laser arrives from an optical fibre to two commercial mirror mounts before arriving at the two automated kinematic mirror mounts. Then it is sent through an optical fibre before reaching a photo sensor. The two commercial mirror mounts are used to control the laser beam before the automated kinematic mirror mounts and the photo sensor works as the source for feedback to the automated kinematic mirror mounts.

During the work of this project there were a lot of noise on the laser signal and it was first late in the process that we were able to find the source of this noise. We found that there were some back reflection from the optical fibres which went back into the laser causing noise on the signal. So to try and fix this we changed the isolator in the existing laser setup, because it was not

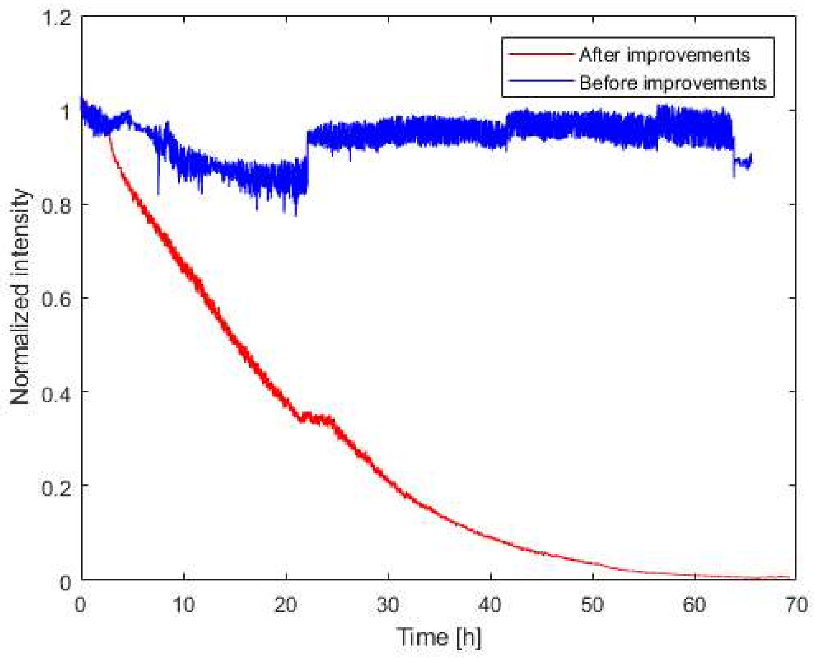


Figure 2.6: Two different measurements which show the improvement made to reduce the noise in the signal.

doing a perfect job securing that no light got back into the laser and disturbing the signal. This helped a little but there were still some noise. Since it was not completely clear were this remaining noise came from I tried to work around this by installing an optical sampler and an extra photo sensor in my setup which is shown in fig. 2.5. As explained in section 2.2 the sampler reflects only a small portion of the laser beam into the photo sensor. By doing this I was able to correct the fluctuations measured on the second photo sensor for fluctuations in the laser beam coming into my setup. This was achieved by dividing the two signals measured. The reduction in the signal noise due to these improvements is shown in fig. 2.6 which shows two measurements taken at different times in the process.

The desire for the implementation of the automated kinematic mirror mounts led to the use of an Arduino as will be explained in chapter 4. This enabled the automation of the mirror mounts without having a computer allocated to this purpose. The motors on the mirror mounts came with ULN2003 driver boards which made the motors easy to control. The driver boards took each four digital inputs, which in total gave a requirement for at least 16 digital outputs on the Arduino. There was also a need for digital outputs to control different buttons, so the requirement was at least 20 digital outputs. To ensure

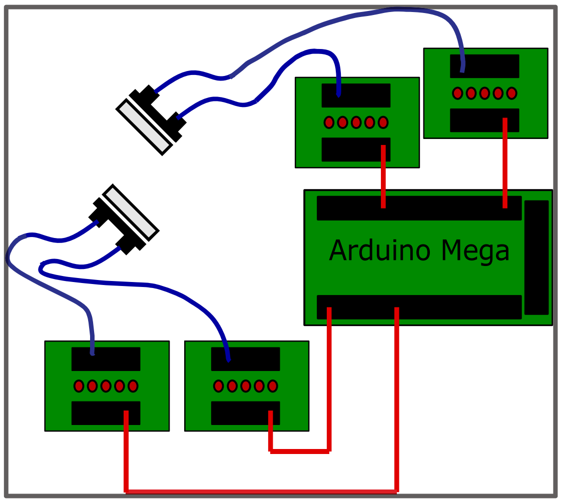


Figure 2.7: How the Arduino was connected to the automated kinematic mirror mounts through the ULN2033 driver boards.

good conditions for the algorithms the Arduino needs to have an ′Analog to Digital Converter′ (ADC) with sufficient bits such that the resolution is high enough. These requirements led to the choice of using an Arduino Mega which has 54 digital outputs and a 10-bit ADC with a working range of 0-5 V which gives a power resolution of $\sim$ 4.9 mV. The Arduino is connected to the automated kinematic mirror mounts through the driver boards as shown in fig. 2.7.

3 Automated kinematic mirror mounts

This chapter starts with giving an overview of the process of 3D printing. In section 3.1 the prototype developed by former bachelor student R. Norup is described, tested and further developed. In section 3.2 the first new design developed in this project is described and various tests are performed to investigate its performance. This leads to the design of a new type of an automated kinematic mirror mount in section 3.3. Finally, this chapter is concluded in section 3.4 with the description of the automated mirror box which is the dream design in this project.

The automated kinematic mirror mounts developed in this project was 3D-printed using a MakerBot Replicator 2 3D printer. It uses the Fused Filament Fabrication (FFF) method where a long plastic filament, 1.75 mm in diameter, is fed through a nozzle with a motor. This small system is called the extruder and it is preheated to 250 °C before it is able to print models. The extruder is connected to rods which are driven by motors such that the extruder can move only in the horizontal plane. The model is printed on a base plated that is able to move up and down. The basic idea is that the extruder liquefy the plastic and prints a layer on the base plate. When one layer is finished the base plate is lowered and the extruder now prints a new layer. In this way the model is built up from below one layer at a time.

The 3D models are designed in a computer-aided design (CAD) program. CAD programs are vector based design tools that allow one to make 3D objects from 2D sketches. There exist many different CAD programs but in this project the Autodesk Inventor was used. To create a 3D object one first draws a 2D sketch from geometric shapes and then extrudes or rotates the sketch into a 3D obejct. The object can then be modified in various ways to look exactly like wanted. After the model is finished it is converted to a format that the MakerBot printer understands using the MakerBot Print software. Finally this file is transferred to the 3D printer using a SD card. As a default the MakerBot Print software converts solid objects into a shell with a honeycomb structure inside it to save both weight, plastic and printing time. The size of the hexagons can be changed or removed in the program. In this project the default setting in the program was used. The material used in this project was polylactic acid (PLA) which is a type of plastic.

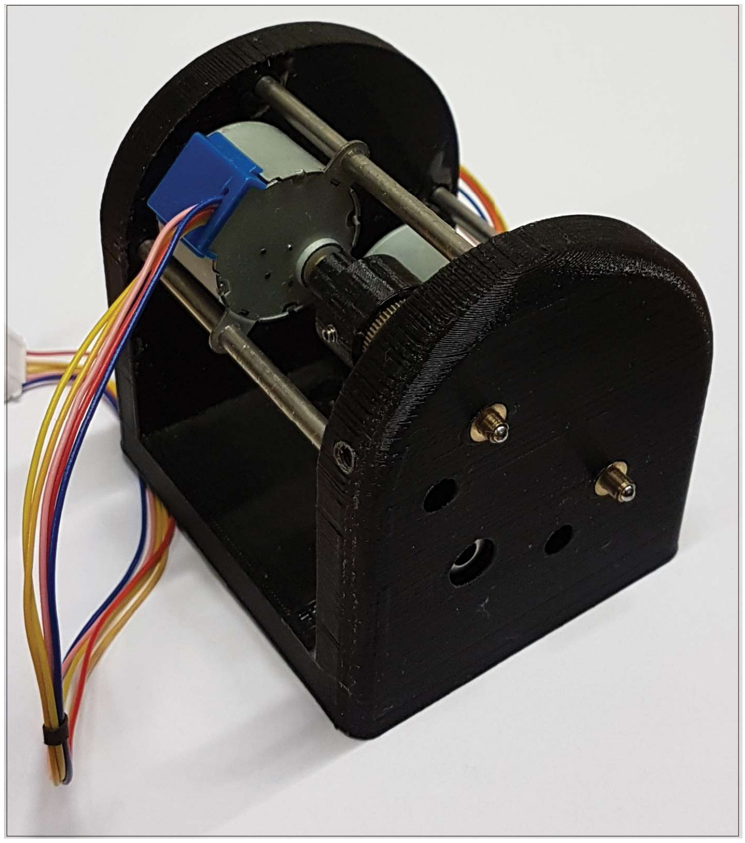


Figure 3.1: The first version of a automated kinematic mirror mount developed by Rasmus Norup [12].

3.1 Prototype mirror mount

The starting point for my work in developing 3D-printed mirror mounts was the previous work of a former bachelor student in the group. In his bachelor project he developed a prototype [12] for a 3D-printed automated kinematic mirror mount using two motors to control the position of the mirror. This design was used as a first version for my work and it is shown in fig. 3.1.

The screws and bushings chosen are ones where the screw travels in the bushing when turned. Thus this automated kinematic mirror mount have the motors hanging on small metal rods so that they are able to move back and forth as the screw travels in and out when the motors are turned. The motors are attached to the screws with some simple adaptors which can be seen in fig. 3.2. The small hole in the adaptor makes it possible to put in a small screw to tighten the grip on the screw that adjust the mirror.

Since the mount was printed in PLA it is possible to mount the mirror

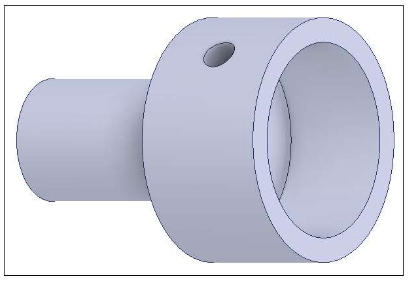
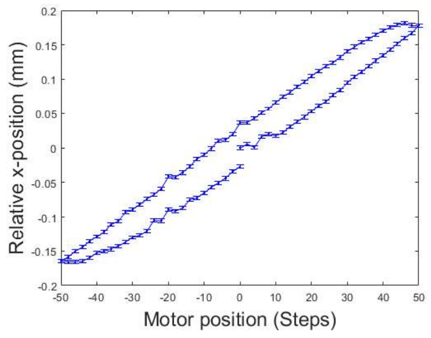
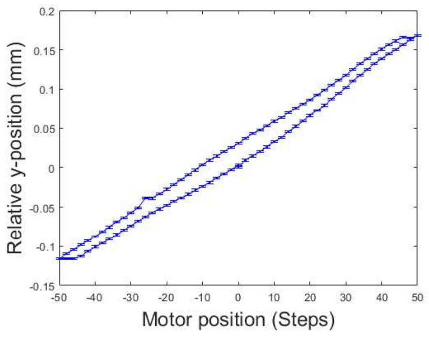


Figure 3.2: The adaptor to the motors such that they are able to turn the screws and thereby changing the position of the mirror.



(a)



(b)

Figure 3.3: The hysteresis for the first version of the automated kinematic mirror mount. (a) The hysteresis in the x-direction for the horizontal motor. (b) The hysteresis in the y-direction for the vertical motor.

between two plates of PLA without scratching the mirror. To hold the mirror tight three M4 nuts and bolts are used as seen in fig. 3.6. The mirror is then attached on the front of the mount with two springes holding it in place.

I started with testing this prototype to find areas of improvement. The first thing I examined was the hysteresis when using the motors to change the position of the mirror. This was done by sending a laser beam onto the mirror on the automated kinematic mirror mount and then onto a CCD camera with a laser beam profiler program such that I was able to measure the laser beam's position on the camera. By turning the motor two steps, measuring the beam position and continuing this for both directions on the motor the hysteresis was measured as shown in fig. 3.3. The errorbars on the data points represent the statistical uncertainty with one σ significance level. which was found by measuring the beam position on the camera 10 times without changing anything in between the measurements.

As seen in fig. 3.3 there was hysteresis when using the automated kinematic mirror mount. Additionally, in fig. 3.3a it seemed like there was also a risk of having plastic deformation since the curve did not come back on itself. These issues were a difficulty when writing the optimising software.

During the initial testing and use of this automated kinematic mirror mount I observed a lot of noise on the signal through the optical fibre that was largest when the motors were changing direction. This led to the thought that the motors were maybe not fastened tightly enough to the mirror mount so that movement of the motor led to vibrations in the whole mount which then led to noise on the signal. If the motors were touched one could easily feel that the metal rods and the attachment holes in the motors did not fit exactly which allowed the motors to move a little sideways especially when changing direction. One way to fix this could be to glue the motors to the metal rods such that there was not any movement but that could potentially give rise to another problem since the screws travel when turned. To investigate if this would be a problem a test to see the step sensitivity of the fibre were made. This was done by finding the mirror position which gave maximum power through the fibre and then turning one of the motors in one direction few steps at a time and measure the power through the fibre to see how quickly the power fell off. This was done for both motors with a distance between the mirror and fibre of (27.5 ± 0.5) cm and the result can be seen in fig. 3.4. These results shows that there was no need to use more than around 30 steps in each direction for each motor. There is a slight asymmetry between the two axis which did not matter much to this project. So to see if these results indicated a problem to the solution of gluing the motors to the metal rods, one needed to know how far the screw would travel when the motor went from one extreme to the other. The screws used in the prototype had 100 threads/inch and the way the program rotates the 28BYJ-48 stepper motors, that are used here, gives the motors 512 steps/rev. It is possible to have higher resolution since one step in the program consists of eight steps where voltage is put on the different coils inside the motor. By doing it this why it is assured that the state of the coils is the same before and after the program has made a step. This means that the screw travels (0.50 ± 0.04) µm/step. To have a small buffer in the system there should be room for the screw to travel 50 µm which is not much. Within the motor itself the shaft can go in and out a little and the screw can travel within the motor adaptor so this should not be an issue.

Based on these measurements, it was decided to glue the motors to the metal rods. This removed a lot of the noise from the signal but also meant that the motors could not come off the mount again and be reused for something else. Because all this happened so early in the process of developing designs I did not take any data to show the noise or the improvement of gluing the motors since the focus was on getting the automated kinematic mirror mount to work properly.

During use of this version of the automated kinematic mirror mount the plastic around the pivot point suffered too high pressure and broke as seen in

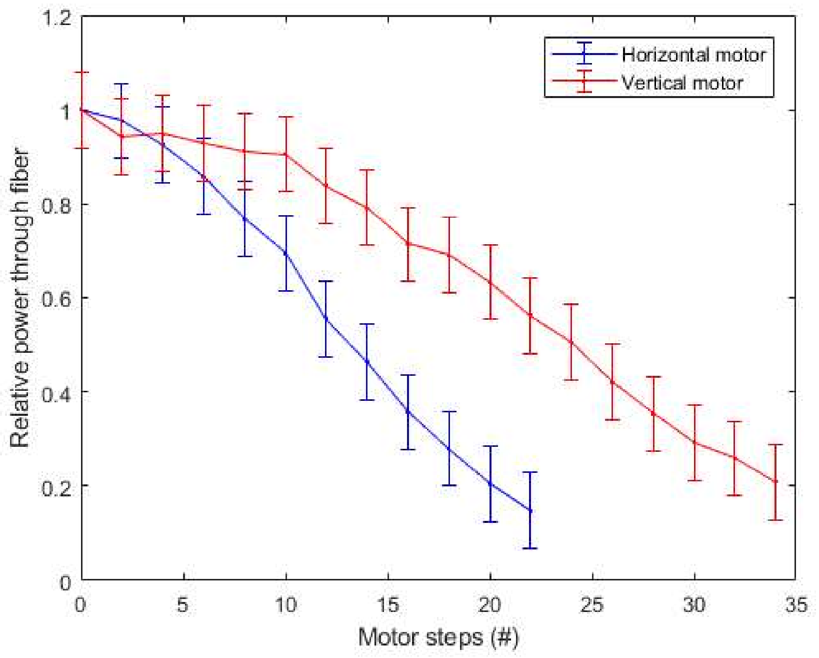


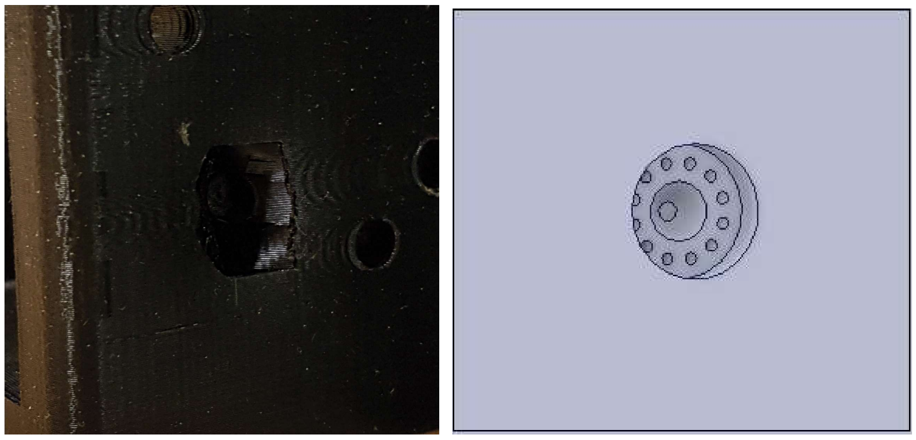
Figure 3.4: The step sensitivity of the fibre with a distance between the mirror and fibre of (27.5 ± 0.5) cm

fig. 3.5a. This issue was fixed by ensuring that the density of plastic in this area was increased. This was done by making small holes in the model which makes the printer use more plastic in the area, see fig. 3.5b. I did a small test to see how much pressure the area was able to take in the new form. The small metal ball that was used as the pivot point was placed as when the mirror mount was assembled. Then heavy lead bricks were placed upon the ball. After experiencing the force from (33 ± 2) kg the plastic was still intact and the test was stopped. The force that the pivot point experienced from two fully extended springs were ~ 10 N [12] which corresponds to the force from ~ 1 kg. This was much less than what the new design could suffice.

3.2 Second version of the automated kinematic mirror mount

Since the motors were glued to the metal rods to minimize vibrations from the motors and were no longer able to move back and forth there were no good arguments for not trying to make the mirror mount better.

There were several considerations to take into account when designing



(a) (b)

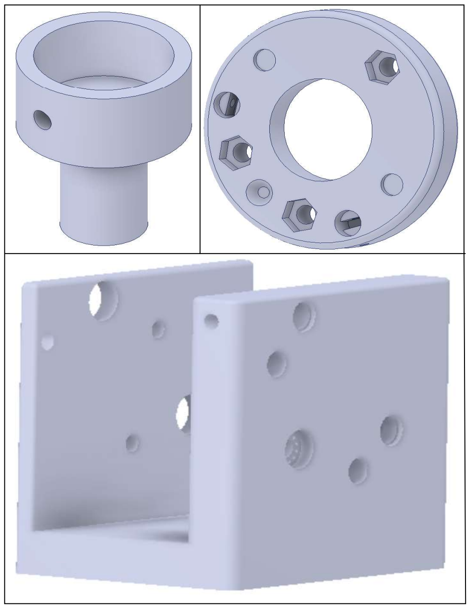
Figure 3.5: (a) Place where the plastic broke around the pivot point and (b) CAD drawing illustrating how this issue was fixed by ensuring higher density of plastic in the area.

the new automated kinematic mirror mount. First, it needed to be small so it could fit into an experimental setup. Second, it should give a good angular resolution when turning the mirror. Third, the motors should be able to move with the screw when turning but be limited to other movements. Fourth, it should endure the force applied by the springs.

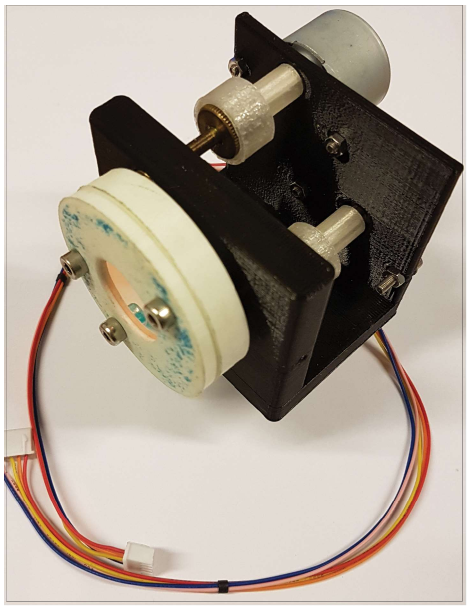
The mirror plates along with the length of the screws used and the length of the motors limited the minimal size of the whole mount. The distance between the front and back plates of the mirror mount should be great enough to enable the screw to get in the bushing. Since the front plate has a certain thickness it was necessary to develop an adaptor to the motor such that it could rotate the screw. The 3D design drawings of the required parts can be seen in fig. 3.6a and an assembly of the finished mirror mount is shown in fig. 3.6b.

As pivot point a small metal ball was located on a washer in the big hole of the front plate seen in fig. 3.6a. To get a high angular resolution as compared with the size of an optical fibre the screws were placed far enough from the pivot point to achieve this.

Since the screws travel as they are turned the design should take this into account. The way this is achieved is two fold. The back plate which hold the motors is thin enough to bend a little which enable the motors to follow the screws as they travel. Furthermore, the adaptor to the motor gives the screw the possibility to slide back and forth while turning. These two features ensured that the screws can travel as they are turned in the required range. The motors are fastened by bolts and nuts to the back plate to limit other



(a)



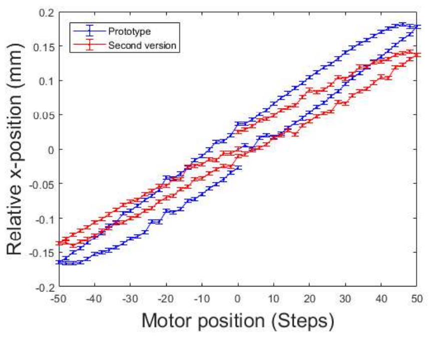
(b)

Figure 3.6: (a) Engineering drawing and (b) The printed and assembled version of the second version of the automated kinematic mirror mount.

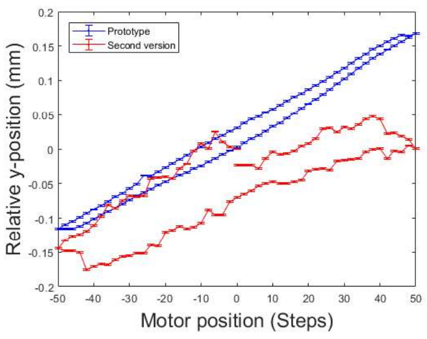
movements than the required ones.

The screws that were used in this design were taken from old 'Radiant Dyes' mirror mounts and have 100 threads/inch. To ensure that the screws can be adjusted when the springs are fully extended the motors need to apply a torque of at least 33 mN m [12]. To meet this requirement 28BYJ-48 stepper motors were chosen. They apply a torque of 34.3 mN m. These motors have 512 steps/rev which in this design results in a minimal step size of 3.42″. One downside of choosing these motors is the fact that they loose steps when switching direction [12]. This means that when the motor is rotating in one direction and the direction is reversed it takes a few steps before the motor is actually rotating again. This had to be compensated for when designing software.

Now that a smaller automated kinematic mirror mount was designed it needed to be tested to see if it worked. The first test made was to measure the hysteresis of this new design and compare it with the old. The hysteresis was measured the same way as described in section 3.1 and the results are shown in fig. 3.7. The hysteresis measurements showed that the hysteresis was worse for the new design than for the prototype and it was worst when using the vertical motor but this design was considerably smaller than the prototype and it was now possible to disassemble and reuse the motors. It

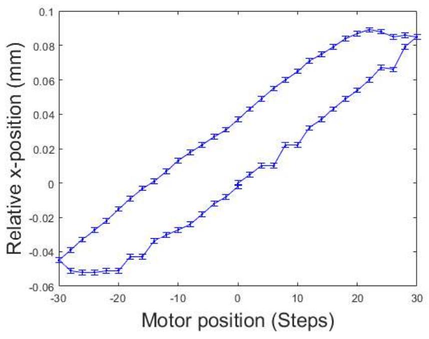


(a)

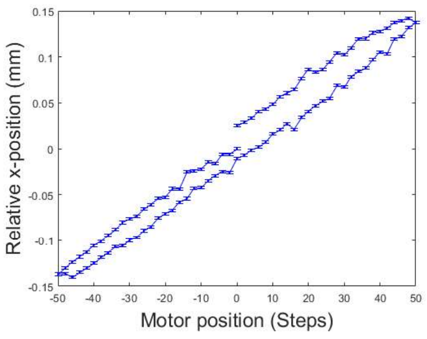


(b)

Figure 3.7: Comparison of the hysteresis for the two current mirror mount designs. (a) The hysteresis in the x-direction for the horizontal motor for the two designs. (b)The hysteresis in the y-direction for the vertical motor for the two designs.



(a)



(b)

Figure 3.8: Comparison the hysteresis in the x-direction for the horizontal motor when turning the motor different amounts in the different directions. In (a) the motor is only turned 30 steps in each direction whereas in (b) the motor is turned 50 steps in each direction.

was then worth testing how the hysteresis looked like if the motors were not turned as much. Therefore the test was repeated with the motors not turned so far in each direction. The results are shown in fig. 3.8. If the motors are limited to move around 30 steps in each direction the hysteresis is nice and there are not any plastic deformations. This gives a limit which had to be taken into account when the software was developed. With this in mind it was reasonable to continue working with this version of the automated kinematic mirror mount.

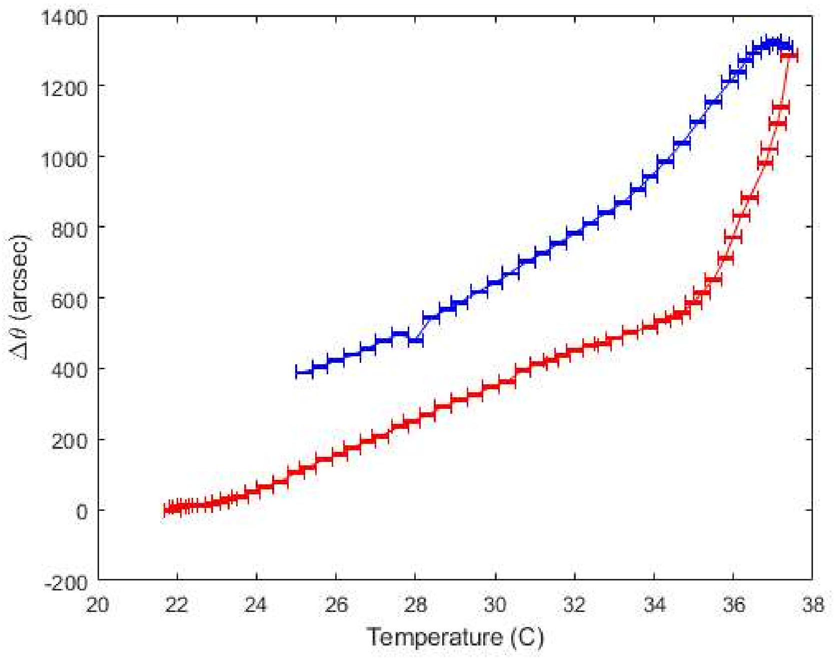


Figure 3.9: Temperature dependence of the mirror position for the automated kinematic mirror mount when it is heated and the cooled back down.

To further test the performance of the automated kinematic mirror mount it was interesting to see how well it handled temperature changes to understand how important a factor that would be when using the mount in a experimental setup. In particular, since PLA is a soft material compared to metal it was reasonable to expect a higher temperature dependence. To test this the mount was placed inside a small-home made oven which was made by the former bachelor student that had being working on this project [12]. The oven had two small holes for the incoming and outgoing laser beam. The outgoing laser beam was directed onto a CCD camera with a beam profiling program. In this way it was possible to measure the movement of the beam when the temperature changed which could be translated to changes in the mirror position.

Figure 3.9 shows that the angular change in the mirror position was large for even small temperature changes. As the temperature rose beyond $\sim$ 36 °C the angular temperature dependence became even greater and it did not recover its original position when cooled back down. This indicates that there was some sort of temperature hysteresis as well. These results points to the importance of having a stable temperature when using this automated kinematic mirror mount in experimental setups. If compared with

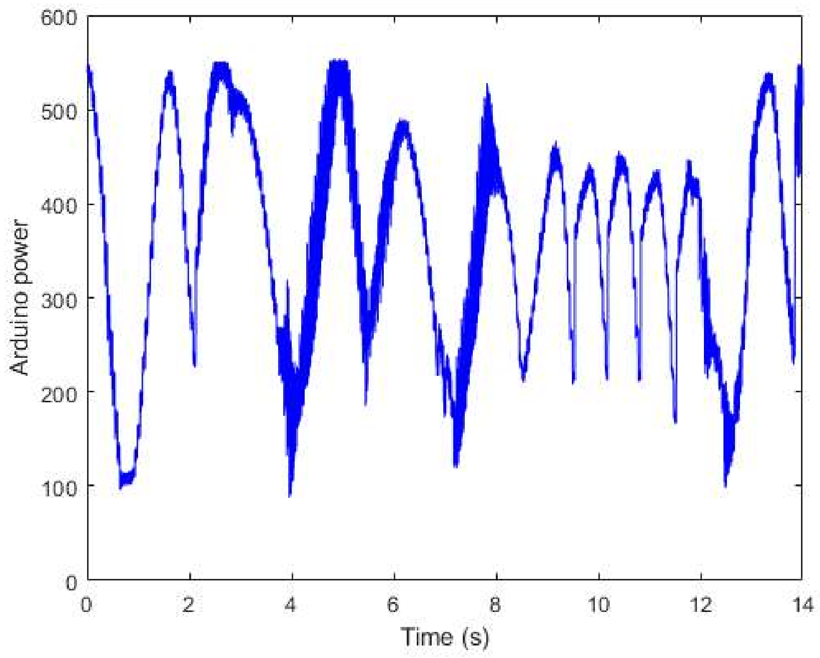


Figure 3.10: The power trace measured by the Arduino through the optical fibre by a first simple program for finding a maximum.

the results of temperature dependence measurements of a Thorlabs KM100 mirror mounts [12], the dependence found in fig. 3.9 were much greater.

To test if this version was able to actually optimise the incoupling of a laser beam into an optical fibre a simple program was developed and used as a test. The program scan a motor for a maximum power through the fibre and stops near it and then does the same for the second motor. The result of this small test is shown in fig. 3.10 where it is seen that it was possible to make this simple optimisation. This showed that it was worth it to continue with development of software with more advanced optimisation schemes.

3.3 Automated vertical mirror mount

Because the second version of the automated kinematic mirror mount is still large compared with commercial mirrors it was interesting to see if it could be made even smaller and maybe even as small as the commercial ones. Then it would really be of great interest to use in many optics laboratories. To make a new version of the automated kinematic mirror mount that was in about the same size as the commercial kinematic mirrors brought up difficulties because the motors themselves takes up a lot of space. In developing what I call

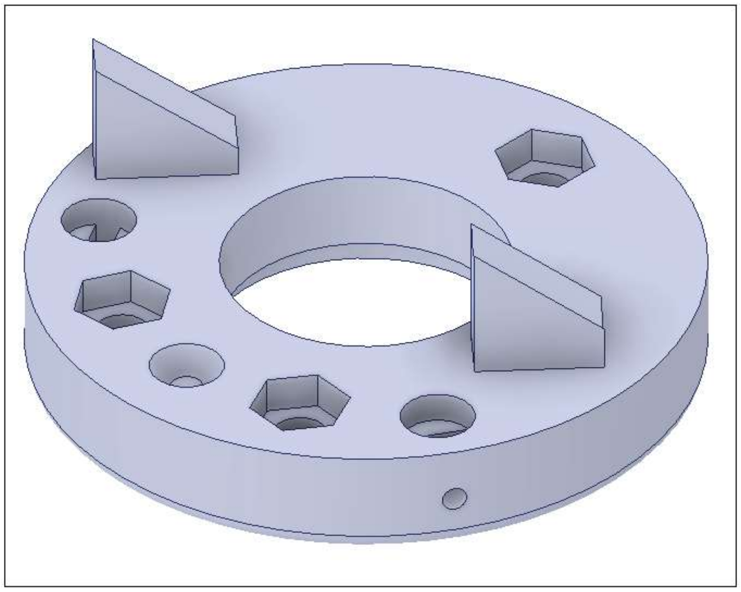


Figure 3.11: The backplate of the plates holding the mirror for the automated vertical mirror mount.

the automated vertical mirror mount I looked at commercial vertical mirror mounts and got inspiration from them. If the screws can be placed vertically and the motors attached on top, which is above the normal mirror height, they do not take up space on the optics tables. That was the starting point for developing the automated vertical mirror mount. To place the screws in a vertical position means that they can not push directly on the mirror holder so another solution had to be developed. The solution chosen for this problem was to have the screws push a ball down a slide that is placed on the backplate of the mirror. The slides on the backplate can be seen in fig. 3.11.

To design the rest of the mount was relatively simple when the decision on how to change the mirror position was made. The design is made such that the mirror is attached in the same way as the previous models. The big difference to this model is that there needs to be some holes where the metal balls can lie and where the slides on the backplate go in. This resulted in the design shown in fig. 3.12.

To assemble the mirror mount the metal balls are placed in the two holes and then the bushings and the screws are inserted in the two holes on top. Then the mirror is attached with two springs and the motors are attached to the screws. To avoid that motors and not the screws turn when applying the program a device to hold the motors together was designed. The idea was to make some new small adaptors for the motors and then a small plate holding both motors such that when one motor was turning the other motor ensures

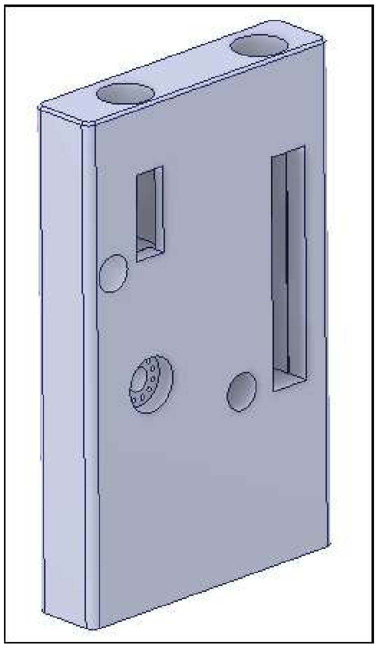


Figure 3.12: The design of the first version of the automated vertical mirror mount.

that it was the screw that was rotating. Figure 3.13 shows the adaptor and the plate holding the motors in place.

When doing some initial test I observed that the hysteresis was worse than the previous models, it experienced plastic deformation, and when the screws travelled up and down the plate holding the motors twisted causing changes and vibrations of the whole mount. This was most noticeable when the motors reversed the direction of rotation.

To really see how the automated vertical mirror mount performed the hysteresis was measured in the same way as previously done with just a small change. For the vertical mount one step of the motor results in a smaller adjustment of the mirror than for the other models, and the range used in measuring the hysteresis was increased accordingly. The difference in range arise because for the vertical mount the screw does not push directly to the mirror, so a smaller travel of the screw results in an even smaller adjustment of the mirror, while there is a $1:1$ relationship for the other designs. The results of the hysteresis measurements are shown in fig. 3.14 where it is clear that the hysteresis is worse if compared with fig. 3.7. The results also indicated that there was plastic deformation when using the automated vertical mirror mount. This constitutes an even bigger challenge if it is used for optimisation.

To try and fix the problem of the movement and vibrations caused by the plate holding the motors a new version of the automated vertical mirror

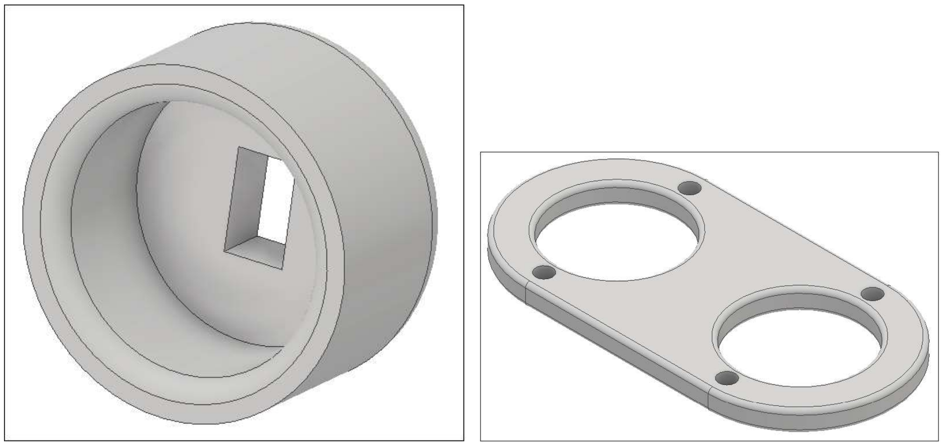
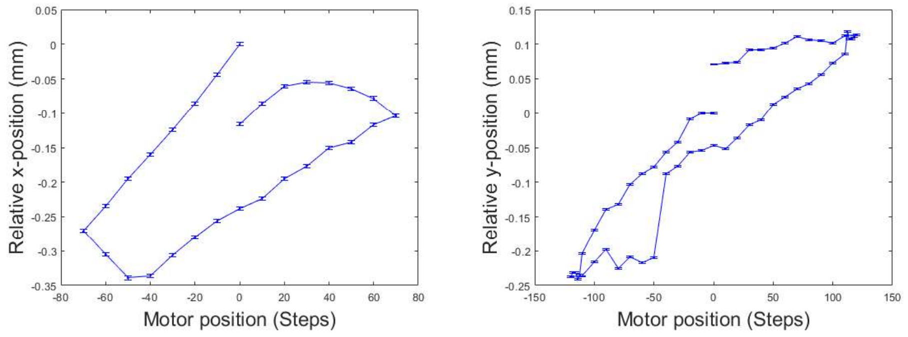

(a) (b)

Figure 3.13: (a) Motor adaptor. (b) The plate holding the motors in place for the automated vertical mirror mount.



(a) (b)

Figure 3.14: The hysteresis for the first version of the automated vertical mirror mount where (a) is the hysteresis in the x-direction for the horizontal motor and (b) is the hysteresis in the y-direction for the vertical motor.

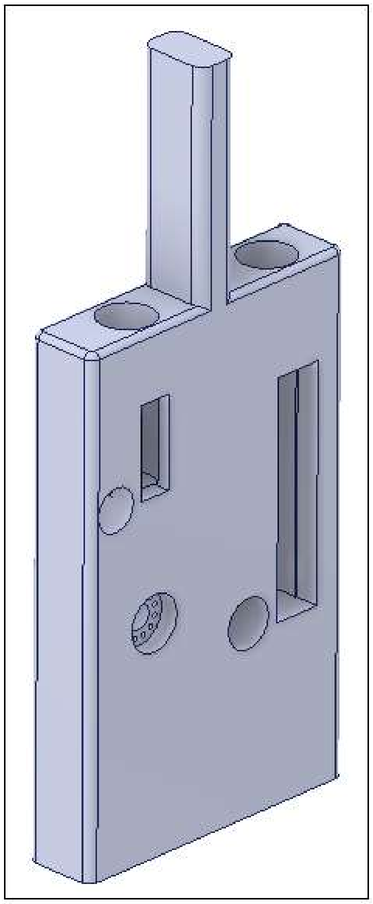
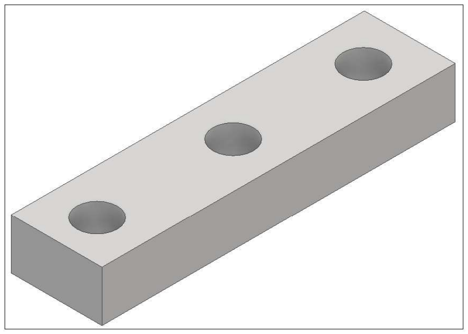


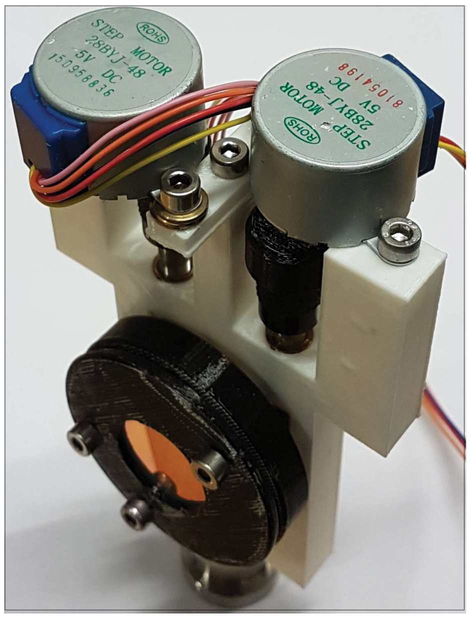
Figure 3.15: The design of the second version of the automated vertical mirror mount.

mount was developed. This version is seen in fig. 3.15 where the difference from the previous version is the pole on top of the mount. The idea with this pole is to create some stabilization to the plate.

Unfortunately the changes to the automated vertical mirror mount did not have the desired effect. The plate holding the motors still twists and moves causing difficulties. This led to the conclusion that the motors somehow needs to be fastened better and probably to the mount itself to give the best possible stability and reduce the movement of the motors. So the design had to be rethought and redesigned to somehow attach the motors to the mount. This was achieved by building up three pillars on top of the existing mount and then create a small add on to fasten the motors. The need for the add on was due to the way the mount was printed, so it was easier to develop this than make some really difficult changes to the mount. If the add on should be



(a)



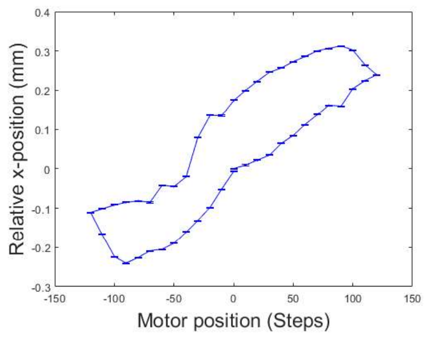
(b)

Figure 3.16: (a) The add on to the mirror mount to hold the motors in place and (b) the assembled third version of the automated vertical mirror mount.

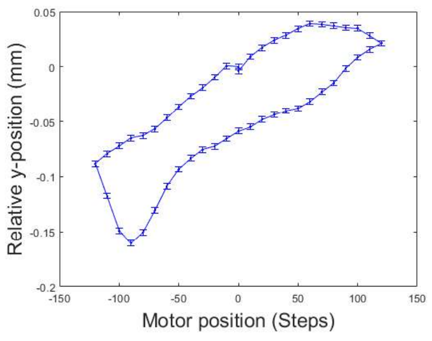
an integrated part of the mount the 3D printer would have needed to built up support to make it possible. The add on to hold the motors in place and an assembled version of the automated vertical mirror mount are shown in fig. 3.16.

To test how well this third version of the automated vertical mirror mount performed the hysteresis was measured to see how difficult a task it would be to use this mount for optimisation and the results are shown in fig. 3.17. It is clear that some improvements have been made compared to the hysteresis found in fig. 3.14. In particular, the plastic deformation is not observed.

To further test this version of the automated vertical mirror mount it was used for an optimisation like the one in section 3.2. Since only one of these mounts were printed it was used in combination with one of the previous models, which were not a vertical mirror mount design. The power trace through the optical fibre for this simple optimisation is shown in fig. 3.18 where the data shows that the automated vertical mirror mount has had difficulties getting high power. Especially if one looks at the peak after around 5 s where the power is much lower than for the peak before and after. This could very well be explained by the large hysteresis shown in fig. 3.17 which means that the mirror could not trace back the way it already had gone and therefore never reached as high power as before. This points to the difficulty



(a)



(b)

Figure 3.17: Hysteresis for the third version of the automated vertical mirror mount where (a) is the hysteresis in the x-direction for the horizontal motor and (b) is the hysteresis in the y-direction for the vertical motor.

in using this mount further on. During the tests it was seen that the automated vertical mirror mount failed more than half the times it tried to do the simple optimisation. Additionally, the power could drop dramatically when the motors changed direction of rotation which was likely caused by vibration of the motor. The reason why it is more clear for the vertical version of the automated kinematic mirror mount is probably because of the design. The vertical version is thinner and probably more unstable and since the motors are on top and vibrating it affects the whole mount more than for the other models. This led to the conclusion that the automated vertical mirror mount was not reliable enough in this state to continue with in the rest of the project. Since there was not enough time to further develop the idea of the automated vertical mirror mount no more versions were further developed.

The best working model to continue with in this project turned out to be the one from section 3.2 and was the one that was used in the remainder of the project.

3.4 Automated mirror box

As a small side project when waiting for the different mounts to be printed or when there were problems with the laser a new model was designed that was hoped to be a final dream version of this project. The idea is to have a small box containing two mirrors inside. With two holes one can send the laser beam into the box and then it comes out aligned on the other side. This way it is possible to make the optimisation setup compact and aesthetic. The box was designed around the same time as the first version of the automated vertical mirror mount so it took its design from that version which later was shown to

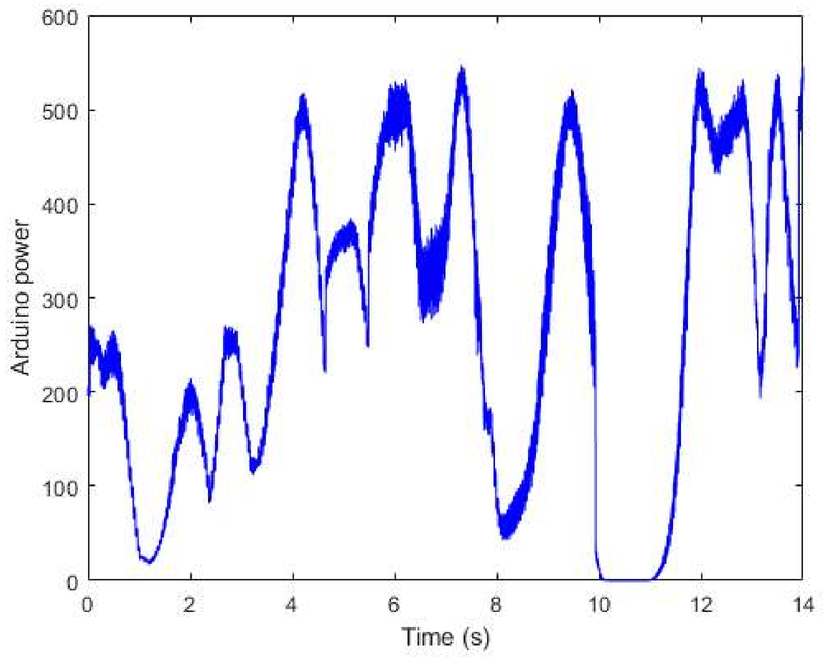


Figure 3.18: The power trace measured by the Arduino through the optical fibre by a first simple program for finding a maximum using one vertical mount and one regular mount.

be sub-optimal for been automatised. A printed and nearly assembled version of the automated mirror box is shown in fig. 3.19. On top of this box four motors were supposed to be attached and do the optimisation, but as shown in section 3.3 that has proven difficult to achieve. Manually one are able to use the mirrors nicely but it was never tried to use it for optimisation since it did not work for the automated vertical mirror mount. Still this shows that it is possible to 3D print a compact setup containing two mirrors that does not takes up significantly more space than commercial mirrors.

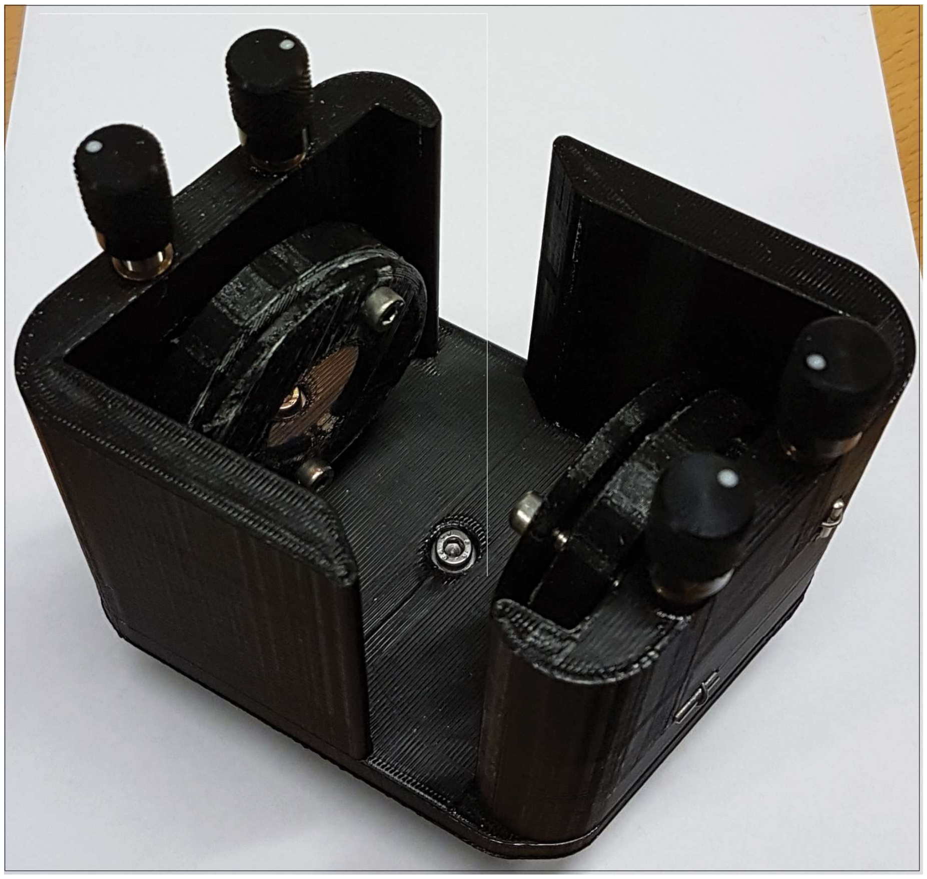


Figure 3.19: The first version of a compact double-mirror mount. It is a small box with two mirrors that can be used for optimisation.

4 Controlling software and hardware

This chapter starts with a brief introduction to why the algorithms needed in this project were developed. The chapter then continues in section 4.1.1 with the development of the first algorithm using the LabVIEW language and an experiment to use as proof of concept. This leads to the programming on the Arduino in section 4.1.2. The algorithms are described in detail in section 4.2 and section 4.3. Finally, section 4.4 explains the interface developed for this project including the design of the menu structure.

To secure a high fibre coupling efficiency in an experimental setup there is a need for two algorithms. First, there is a need for an algorithm which starts out with low power through the optical fibre and then optimises the fibre coupling efficiency. This algorithm do the job of beam walking. Second, an algorithm to maintain a high fibre coupling efficiency over time is preferable. This algorithm needs to be activated when the power through the fibre falls below some desired value and then optimises the fibre coupling until an acceptable value is reached. Since this project aims to fabricate automated kinematic mirror mounts there is a need for developing controlling software. This software should control all the possible variables for the kinematic mirror mounts. Also the software needs to have all the desired functions that is wanted for the automated kinematic mirror mount. Within the group a lot of the existing software is developed in LabVIEW and because this is already widely used, this was the starting point for me to develop the software for the automated kinematic mirror mount. Since I had no previous experience in programming in LabVIEW I had to start from scratch.

4.1 Programming

4.1.1 Labview

LabVIEW is a platform for designing systems using a visual programming language which is named ”G” [16]. Normally LabVIEW is used for data acquisition, instrumental control and automation which is exactly what I need to do. The programming language, ”G”, is based on data availability and is dataflow programming, which means that if enough data is available for the functions they will execute and the execution flows through the program as the data needed is becoming available. LabVIEW uses graphical programming where functions, data structures etc. have a pictorial representation. To program in this language is to drag and drop the pictures of the things needed

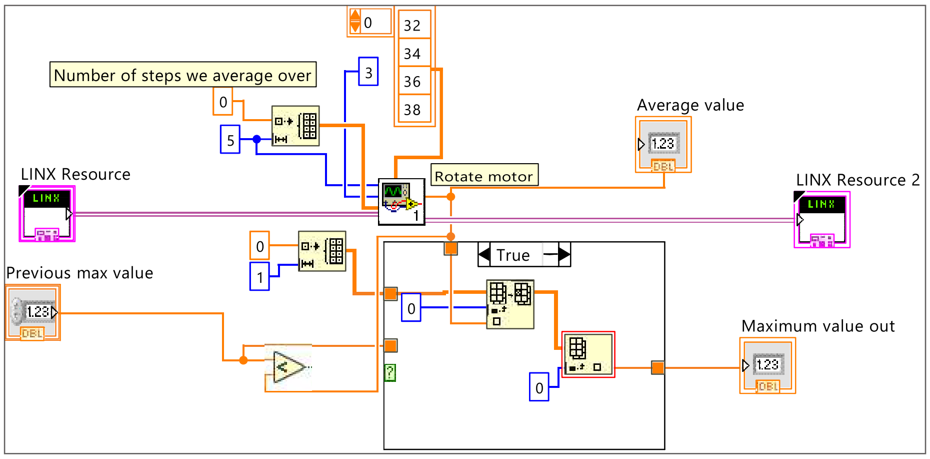
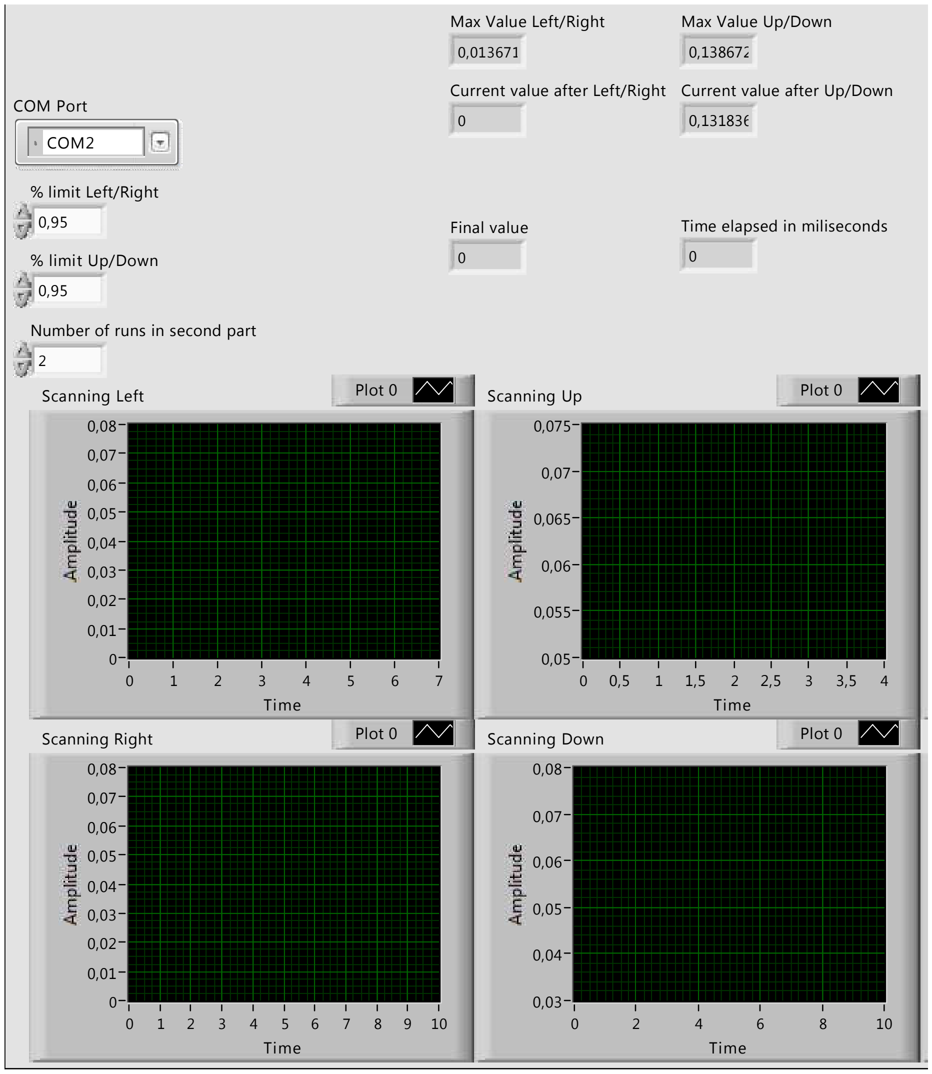


Figure 4.1: How programming in LabVIEW looks like.

and then connect the different parts by drawing wires between them. The wires represent the dataflow between the different parts of the program. This means that data is only available to a function if there is a wire connected which transfers the needed data from a previous section in the program. To better understand how this look see fig. 4.1. A not too detailed explanation of what is shown on fig. 4.1 comes here: The small program shown on fig. 4.1 rotates a motor five steps while measuring the power for each step. After the five steps the program takes the average of the measured power and then compares this power to the previous highest measured power. It then remembers the one that is highest. This small program is one of the basic parts of making a beam walk algorithm, because this small part will work as a scanning program that scans the motor for finding the highest possible power through the fibre. A more detailed explanation of how the beam walk algorithm is achieved in LabVIEW will follow in section 4.1.1.

One of the benefits of using LabVIEW is that it is intuitively to use when using it for the first time. Another benefit is that when programming you automatically develop an interface to control the program which ease the controlling of the software. As seen from fig. 4.2 the interface can consist of different indicators where parameters can be set and results are shown. Also the interface can include graphical representation of desired data. The white displays on the left in fig. 4.2 are the inputs one must give the program to be able to run and this is also where different parameters can be changed. The grey displays will show the chosen results when the program is finished and also the data chosen for a graphical representation will be shown. It is not possible to follow the development of the program on the screen while it is running. A downside of programming in LabVIEW is that when the programs get more complicated than the one shown in fig. 4.1, then it gets



Max Value Left/Right: 0,013671

Max Value Up/Down: 0,138672

COM Port: COM2

Current value after Left/Right: 0

Current value after Up/Down: 0,131836

% limit Left/Right: 0,95

% limit Up/Down: 0,95

Final value: 0

Time elapsed in miliseconds: 0

Number of runs in second part: 2

Figure 4.2: How the interface in LabVIEW looks like.

complicated to follow what is going on. This also makes maintenance and further development more difficult. As an example look at fig. 4.3.

Proof of concept

To prove that it is possible to make automated kinematic mirror mounts that can optimise the coupling into an optical fibre I started writing a first simple version of a beam walk algorithm. Since this is what the project is all about, this needs to be possible to continue with the work. The first hurdle to write a beam walk algorithm was to figure out a way to find the maximum power through the fibre. The solution chosen in this project is based on the assumption that there has to be some power through the fibre before the algorithm starts. Another assumption is that the program does not know anything about where the maximum is or how it looks. These assumption were made because the program should be as basic as possible to hopefully work in as many situations as possible and also the program needed something to start with, such that it did not search in the dark which could take forever. A criteria for the program is that it should be fast and not use much more time than a human will. With these assumptions and criteria the program was started to be written. As mentioned above, the first hurdle was to find a maximum. The solution based on the above assumptions is to chose a motor and let it rotate in one direction while measuring the power through the fibre. For simplicity let us make it specific. First, the horizontal motor rotates left for a given amount of steps while measuring the power and remembering the highest one. Then, the motor starts turning right for a number of steps, that are twice the steps it turned left while still measuring the power and remembering the highest measured power. This is done because there is the assumption that the program does not know where the maximum is, so it has to scan in both directions to be sure that it at some point passes the maximum. When this scanning left and right is finished the program is to the right of the maximum, which make it relatively easy to write the part of the program, that make the program stop at the maximum. I write relatively easy because this proved to be more difficult than expected. To stop at the maximum, the program needs to turn the motor left and stop near the already measured maximum value. This is where the difficulties start. Because how close should the program come to the maximum before it stops? Since this is a program to make a proof of concept test the chosen limit is 95 %. As shown in chapter 3 the automated kinematic mirror mounts have hysteresis which proved, a problem which sat a limit on how high the stopping limit can be put. It was observed many times that when the program tried to stop near the maximum it did not reach high enough power to come above the stopping limit. That is why a limit of 95 % is chosen. This gives a high chance of succeeding. These experiments are done with the first version of the prototype of the automated kinematic mirror mount shown in section 3.1 and there are a lot of noise on the signal though the fibre. To make it a little smoother the program was changed such that the motor rotated 5 steps and averaging the power. This gives a less noisy signal but it also reduces the resolution of the power measurements. In the end it gave better results.

Now the program can take a motor and find the position with maximum power through the fibre, so to complete the beam walk algorithm, the other mirror has to be 'walked'. The final version of the beam walk algorithm is shown in fig. 4.3. The first part of the final program is optimising both motors on the second mirror as described above. The second part is that it starts with

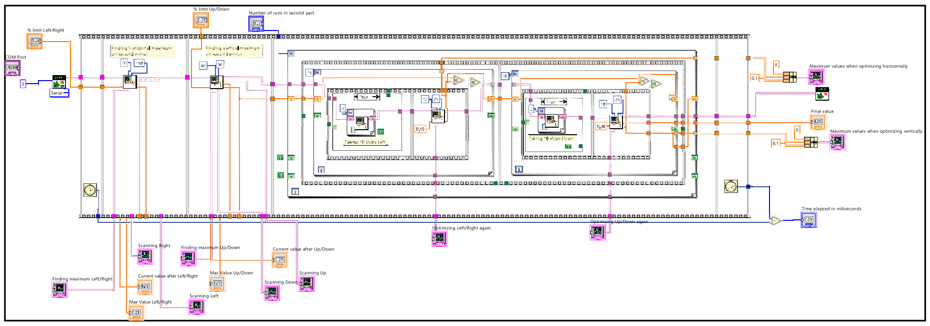
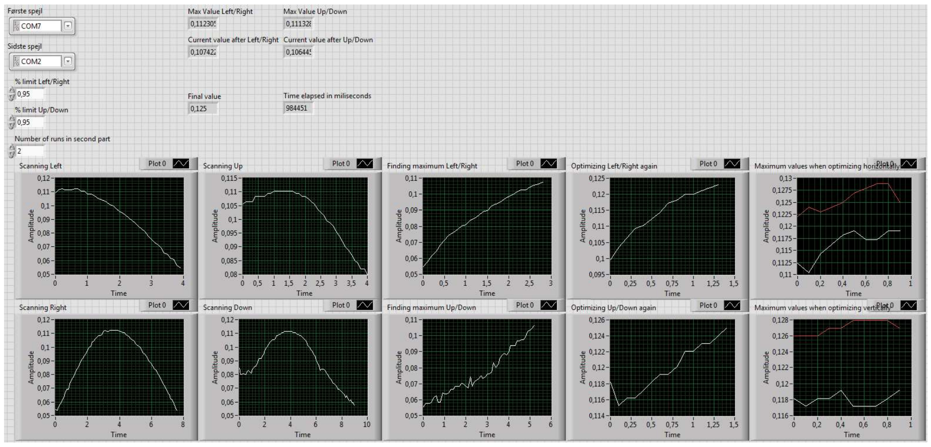


Figure 4.3: Beam walk algorithm in the LabVIEW language.


Scanning Left
Scanning Up
Finding maximum Left/Right
Optimizing Left/Right again
Maximum values when optimizing horizontally
Scanning Right
Scanning Down
Finding maximum Up/Down
Optimizing Up/Down again
Maximum values when optimizing vertically
Plot 0
Amplitude
Time

Første spejl
COM7

Sidste spejl
COM2

% limit Left/Right
0,95

% limit Up/Down
0,95

Number of runs in second part
2

Max Value Left/Right
0,11230

Max Value Up/Down
0,11132

Current value after Left/Right
0,10742

Current value after Up/Down
0,10644

Final value
0,125

Time elapsed in miliseconds
984451

Figure 4.4: Result of the LabVIEW beam walk algorithm where the second part was done twice.

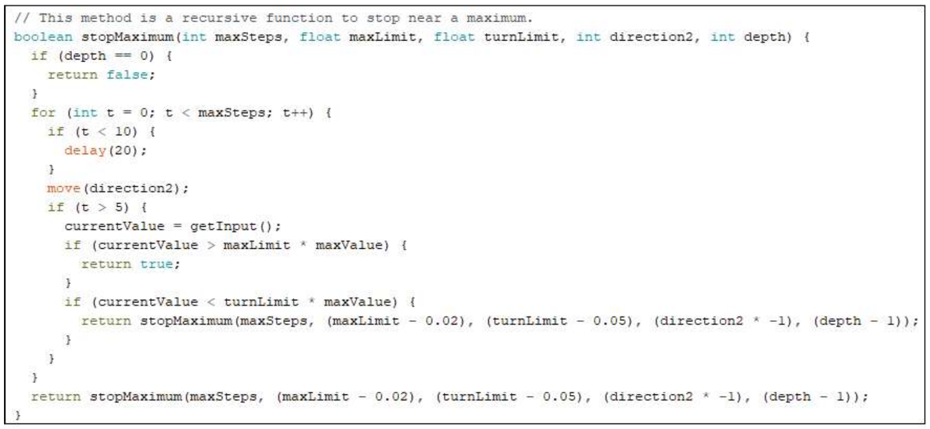
the horizontal motor on the first mirror and moves 10 steps to the left and then optimises the horizontal motor on the second mirror. If this yields a higher value, the motor on the first mirror rotates 10 steps left again. If not, the motor rotates 10 steps right. This is continued 10 times before the same is done for the two vertical motors. The limit of 10 times is set to reduce the risk of the algorithm to fail and also reduce the time it takes. This is done because the results first appears when the program is finished. The program has the option to put a number of repeats on the second part of the program to see if it improves with more attempts. The results of one of the initial experiments are shown in fig. 4.4 where the second part of the algorithm is repeated twice.

To understand the results shown in fig. 4.4 it is best to look at the graphs. The two to the left show how the power develops when the horizontal motor scans left and right in the first part of the algorithm. The next two graphs show the same but for the vertical motor. The next two graphs show how the power developes when the motors in the first part tries to stop near the maximum. This is shown such that one is able to see how this part of the process went, which helps pinpointing issues. The two graphs to the far right are the most interesting ones. These are the ones showing how the beam walking went. The white curve is for the first run of the second part in the program and the red is the second run of the second part. The data shown in these are the maximum measured values for each beam walking step in the second part, which was moving the first motor and then optimising the second motor. For the algorithm to succeed, the power should increase for each step in this progress. Over time there is a small increase in power but the largest improvement happens between the two runs of the second part. This can be explained by the lack of decoupling between the axis on the automated mirror mount, meaning that when turning the horizontal motor it might affect the vertical position as well. All in all these results clearly proves that it is possible to make automated kinematic mirror mounts that can optimise the coupling into an optical fibre.

There were also things that I found very complicated to do. When using many case-structures such as if- while- and for-loops things got very complicated to navigate in in LabVIEW. In these cases I found that LabVIEW seemd very rigid to work with and it was difficult to follow the dataflow which made it very hard to maintain. Another issue with using LabVIEW was that it was very slow because the runtime environment had to be opened every time which was a large and slow task. The results shown in fig. 4.4 show that the program took $\sim 16.5\,\text{min}$ which is a long time. One final issue that I found using LabVIEW was that to run the programs, I needed to have a computer connected to the Arduino that supported the mirror mounts. This is not a flexible solution, so after taking these things into account I chose to go in another direction. Since I already used an Arduino for the setup I looked into how it was possible to write code directly to the Arduino which could be saved directly on the Arduinos memory.

4.1.2 Arduino programming

The Arduino has 256KB flash memory to store and run code and it uses a C/C++ based language which make it easy to programme [17]. Since I had no previous experience in programming in either C og C++ I had to learn a new programming language and syntax. Both the C and C++ language are text based programming languages which allows for creating functions and using logical statements. One of the differences between C and C++ is that in C++ there is the possibility to write programs as object oriented programming which enables the creation of classes [18]. This relates to the language of JAVA which I have a little previous experience in writing. For me to be able to write programs to the Arduino I had to learn some new stuff but not start from scratch. To get an idea of how to write functions and see the syntax for the Arduino code see fig. 4.5.



```
// This method is a recursive function to stop near a maximum.
boolean stopMaximum(int maxSteps, float maxLimit, float turnLimit, int direction2, int depth) {
  if (depth == 0) {
    return false;
  }
  for (int t = 0; t < maxSteps; t++) {
    if (t < 10) {
      delay(20);
    }
    move(direction2);
    if (t > 5) {
      currentValue = getInput();
      if (currentValue > maxLimit * maxValue) {
        return true;
      }
      if (currentValue < turnLimit * maxValue) {
        return stopMaximum(maxSteps, (maxLimit - 0.02), (turnLimit - 0.05), (direction2 * -1), (depth - 1));
      }
    }
  }
  return stopMaximum(maxSteps, (maxLimit - 0.02), (turnLimit - 0.05), (direction2 * -1), (depth - 1));
}
```

Figure 4.5: Small example from the code written for the Arduino.

Table 4.1: Most basic functions used in creating the Arduino algorithms.

Basic programming functions
move(direction)
getInput()
findMaximum(...)

To develop the two needed algorithms for this project some basic functions has to be developed, which are the foundation for the algorithms. Those functions can be seen in table 4.1.

Because it has already been proved that it is possible to make some algorithms that can optimise the coupling into the fibre, this time the algorithms needs to be more sophisticated. The idea of how to do this in C/C++ is to create a 'Motor Class' with the functions mentioned in table 4.1 along with some other support-functions. This way the functions become simpler and the functions are encapsulated. One of the benefits of doing it this way is that the four motors can be created as objects and then easily manipulated individually.

To describe how the two algorithms work the basic functions need to be addressed first. The first function gives the ability to move the desired motor in the chosen direction. The second function tells the Arduino to measure the power from the photo sensor after the optical fibre. It does this by measuring the power 100 times with 1 ms delay in between and then taking the average. This is chosen to compensate for small and fast oscillating disturbances in the signal and since measuring the power takes on the μs level it is extremely fast compared with during it in LabVIEW. This allow the program to maintain the full power and step resolution and still smooth out the power. The third function is the most complicated of the three and is pretty similar to the one described in section 4.1.1 with the same basic assumptions but there are important differences. This function uses one motor to find the best possible position for this motor. As shown in fig. 3.8 and fig. 3.9 the hysteresis and temperature dependence makes this a more difficult task. As stated before, the motors looses steps when switching direction of rotation. The function needs to work around these obstacles to be able to succeed in finding the best position for the motor. There are some basic assumptions behind this function. First, there has to be a signal through the fibre such that the function has something to start with. The signal has to have a sufficiently high signal/noise ratio such that noise does not interfere with the algorithms ability to perform. Second, the function assume that the motor is not able to back trace the beam position when switching direction due to the hysteresis. This together with the loss of steps does that the function cannot just remember its position for the highest measured value and then returning to this when it does not find any position that is better. So to overcome these difficulties the function works like this: First, the motor rotates in one direction and always stores the highest measured power. It keeps moving until the power falls below 70 % of the maximum measured value. This is to save time compared with the version written in LabVIEW. Then, the motor starts rotating in the other direction while it always measures the power and checks whether the power is higher than the previous highest measured. It always remembers the highest power measured. The motor again continues until the power falls below 70 % of the maximum measured power. Every time the motor switches direction it does not measure the power for the first five steps, because when the motor switches direction it causes vibrations and it looses steps, so this tries to compensates for this and secure that the program does not get 'wrong' measurements that are influenced by too much noise. Now the program knows how high power it is capable of achieving and the function now tries to get close to this value. The motor changes direction again and continues until the power is above 98 % of the maximum measured power. If this never happens because of hysteresis and the power starts dropping way below the maximum measured value, the motor changes direction and try to find the maximum again. But now the requirement to stop is lowered to 96 %. If it again fails, it changes direction again but now with 94 %. This continues for several tries where the requirement is lowered 2 % for each time. The requirement is lowered to compensate for the risk that the hysteresis and deformation makes it impossible to reach the same power as in the beginning. The way this is written in code can be seen in fig. 4.5 where the stopMaximum(...) function is the one trying to stop near the maximum. This is a recursive function that calls itself with lower limits if it fails in stopping near the maximum. As a parameter it, as an example, takes the first stopping limit but also how many tries it should maximum use before giving up.

4.2 Beam walk algorithm

Now that the basic functions are clear the algorithms can be explained. The **BeamWalk** algorithm is created to optimise the fibre coupling efficiency by doing a beam walk like the ones usually performed in laboratories. The algorithm falls in four stages where in the first stage the algorithm optimises in the horizontal plane. In the second stage the algorithm optimises in the vertical plane. These two stages are repeated as stage three and four. Each stage consist of five steps explained here and shown in fig. 4.6. The first step is to optimise mirror 2 using the findMaximum(...) function explained earlier. The maximum value measured when using the function is stored in a variable called 'previousValue'. Then, in the second step mirror 1 is turned until the power falls below 90 % of the maximum measured value in step one. In the third step mirror 2 is again optimised using the findMaximum(...) function. In the fourth step the maximum measured value in step three is compared against the value stored in previousValue. After the comparison the maximum value measured in step 3 is now stored in previousValue. Step five is the most complicated part of the algorithm since it is here the decisions are made. The decision process is as follows and is also shown in fig. 4.6. If the maximum measured value in step three was equal to the value stored in previousValue the algorithm is finished with this stage and continues with the next stage. If the maximum measured value in step three was higher than the value stored in previousValue mirror 1 has turned in the correct direction and the algorithm remembers this by setting a variable called 'Correct direction' as true. Then steps 2-5 is repeated. If the maximum measured value in step three was lower than the value stored in previousValue then two things can happen. The first thing is if mirror 1 has already turned in the correct direction, because this means that the algorithm has turned mirror 1 past the optimal position and therefore the algorithm needs to stop this stage and continue with the next stage. The second thing is if mirror 1 has not already turned in the correct direction. Then the direction of rotation is reversed for mirror 1 and the algorithm has a counter that is counted up once. Then step 2-5 is repeated. If the counter reaches five the algorithm stops this stage and continues with the next stage. This is done because the algorithm and mirrors can experience a situation where the maximum measured value in step three is still lower than the previous measured maximum value even though mirror 1 is rotated in the opposite direction. To make sure that the algorithm does not continue with something that is going in the wrong direction it stops. These situations occur because of the hysteresis and deformation explained earlier.

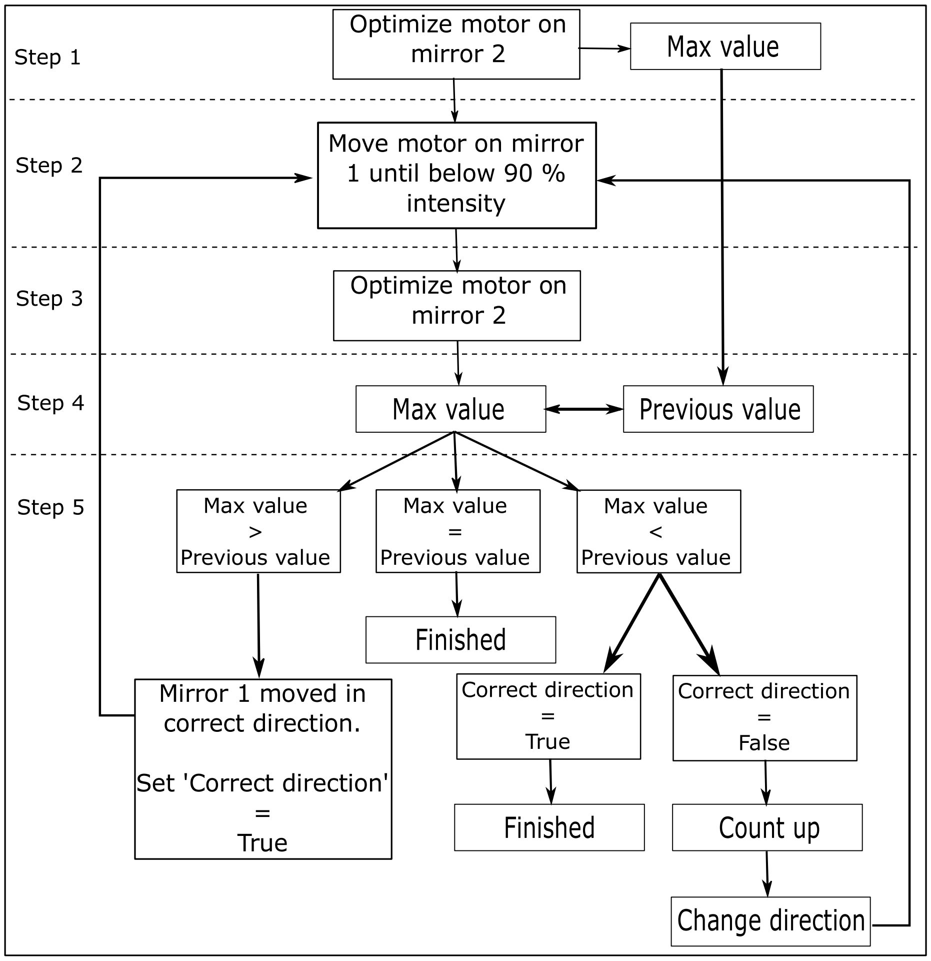
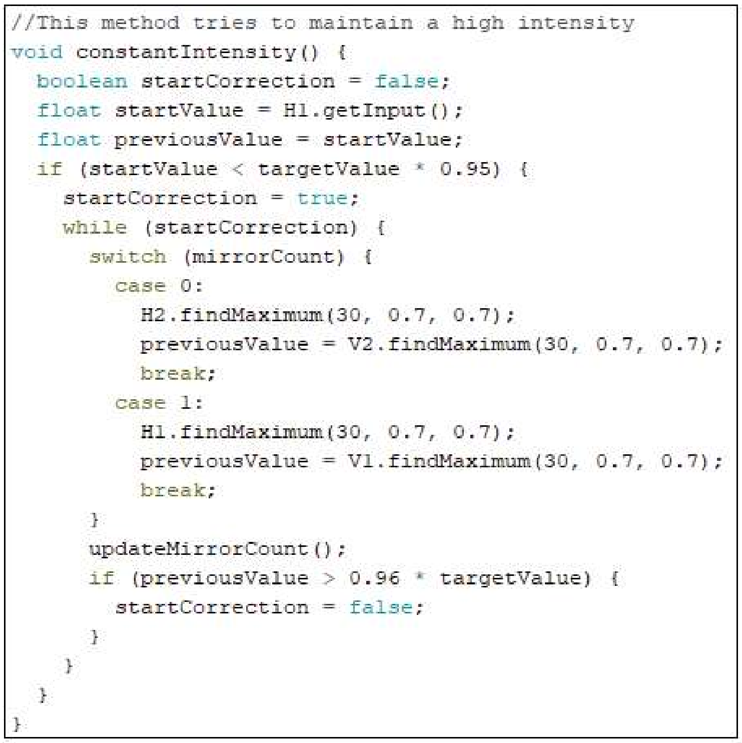

Step 1
Optimize motor on mirror 2
Max value
Step 2
Move motor on mirror 1 until below 90 % intensity
Step 3
Optimize motor on mirror 2
Step 4
Max value
Previous value
Step 5
Max value > Previous value
Max value = Previous value
Max value < Previous value
Finished
Mirror 1 moved in correct direction.
Set 'Correct direction' = True
Correct direction = True
Correct direction = False
Finished
Count up
Change direction

Figure 4.6: The five steps in each stage of the beam walk algorithm.

4.3 Constant power algorithm

The second algorithm developed in this project is called the **Constant power** algorithm. This algorithm aims to maintain a high power through the fibre. To use this algorithm the fibre coupling efficiency already needs to be high and a certain target value has to be set. The target value is the power the algorithm should try to maintain within the limits of the algorithm which is described in the following. When the target value is set and the algorithm is activated, the algorithm always observe the power through the fibre. If the power falls below 95 % of the target value the algorithm starts to make corrections to the mirrors positions. The algorithm does this by optimising the position of one of the mirrors by using the findMaximum(...) function on both the horizontal and vertical motor of the mirror. When the chosen mirror



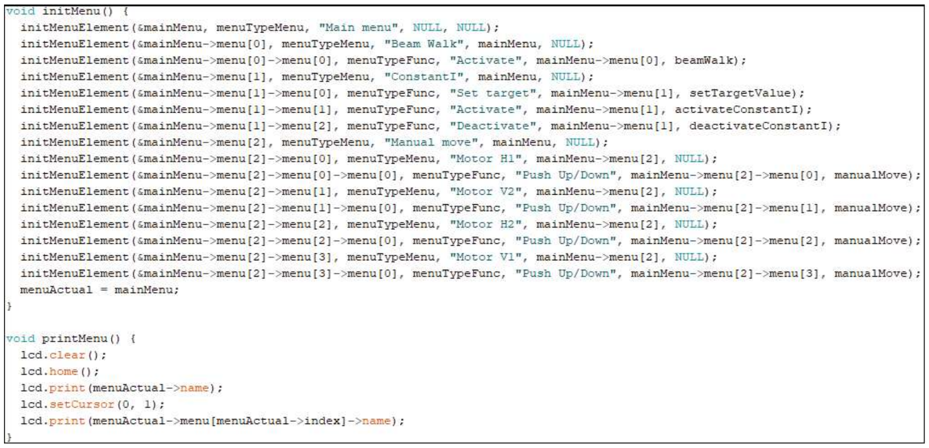
```
//This method tries to maintain a high intensity
void constantIntensity() {
  boolean startCorrection = false;
  float startValue = H1.getInput();
  float previousValue = startValue;
  if (startValue < targetValue * 0.95) {
    startCorrection = true;
    while (startCorrection) {
      switch (mirrorCount) {
        case 0:
          H2.findMaximum(30, 0.7, 0.7);
          previousValue = V2.findMaximum(30, 0.7, 0.7);
          break;
        case 1:
          H1.findMaximum(30, 0.7, 0.7);
          previousValue = V1.findMaximum(30, 0.7, 0.7);
          break;
      }
      updateMirrorCount();
      if (previousValue > 0.96 * targetValue) {
        startCorrection = false;
      }
    }
  }
}
```

Figure 4.7: Code of the 'Constant power' algorithm

is optimised the algorithm checks if the power is above 96 %. If this is the case then the algorithm is finished with making corrections and goes back to observing the power. If the power is not above 96 % then the algorithm tries to optimise the other mirror and then checks if it is then good enough. It continues in this way until the power is above 96 %. The code for this algorithm is simple and is shown in fig. 4.7.

4.4 Interface

To be able to use the two different algorithms easily, along with the desire to be able to manually adjust the mirrors as well, there are a need for some kind of interface to achieve this. For the beam walk algorithm the only thing needed is some way to activate this algorithm. The constant power algorithm has greater demands. For this algorithm to work it should be possible to set the target value, activate the algorithm and deactivate the algorithm. To manually move the mirrors by using the motors it should be possible to chose in which direction the motors should turn. To handle all these desires an interface with a LCD-display along with four buttons (up, down, select and back) will be



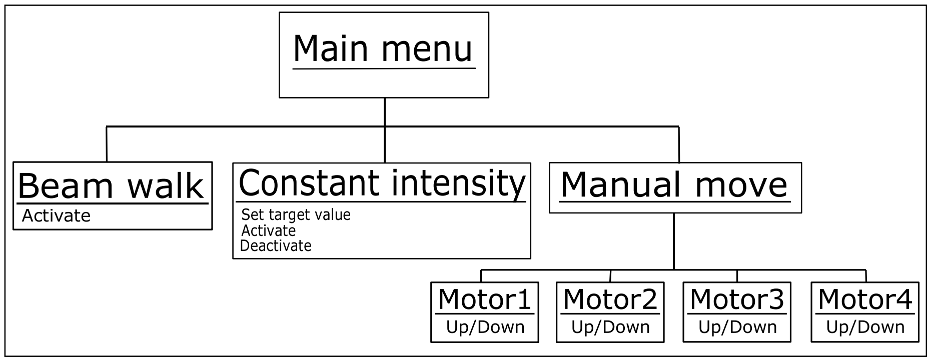
```
void initMenu() {
  initMenuElement(&mainMenu, menuTypeMenu, "Main menu", NULL, NULL);
  initMenuElement(&mainMenu->menu[0], menuTypeMenu, "Beam Walk", mainMenu, NULL);
  initMenuElement(&mainMenu->menu[0]->menu[0], menuTypeFunc, "Activate", mainMenu->menu[0], beamWalk);
  initMenuElement(&mainMenu->menu[1], menuTypeMenu, "ConstantI", mainMenu, NULL);
  initMenuElement(&mainMenu->menu[1]->menu[0], menuTypeFunc, "Set target", mainMenu->menu[1], setTargetValue);
  initMenuElement(&mainMenu->menu[1]->menu[1], menuTypeFunc, "Activate", mainMenu->menu[1], activateConstantI);
  initMenuElement(&mainMenu->menu[1]->menu[2], menuTypeFunc, "Deactivate", mainMenu->menu[1], deactivateConstantI);
  initMenuElement(&mainMenu->menu[2], menuTypeMenu, "Manual move", mainMenu, NULL);
  initMenuElement(&mainMenu->menu[2]->menu[0], menuTypeMenu, "Motor H1", mainMenu->menu[2], NULL);
  initMenuElement(&mainMenu->menu[2]->menu[0]->menu[0], menuTypeFunc, "Push Up/Down", mainMenu->menu[2]->menu[0], manualMove);
  initMenuElement(&mainMenu->menu[2]->menu[1], menuTypeMenu, "Motor V2", mainMenu->menu[2], NULL);
  initMenuElement(&mainMenu->menu[2]->menu[1]->menu[0], menuTypeFunc, "Push Up/Down", mainMenu->menu[2]->menu[1], manualMove);
  initMenuElement(&mainMenu->menu[2]->menu[2], menuTypeMenu, "Motor H2", mainMenu->menu[2], NULL);
  initMenuElement(&mainMenu->menu[2]->menu[2]->menu[0], menuTypeFunc, "Push Up/Down", mainMenu->menu[2]->menu[2], manualMove);
  initMenuElement(&mainMenu->menu[2]->menu[3], menuTypeMenu, "Motor V1", mainMenu->menu[2], NULL);
  initMenuElement(&mainMenu->menu[2]->menu[3]->menu[0], menuTypeFunc, "Push Up/Down", mainMenu->menu[2]->menu[3], manualMove);
  menuActual = mainMenu;
}

void printMenu() {
  lcd.clear();
  lcd.home();
  lcd.print(menuActual->name);
  lcd.setCursor(0, 1);
  lcd.print(menuActual->menu[menuActual->index]->name);
}
```

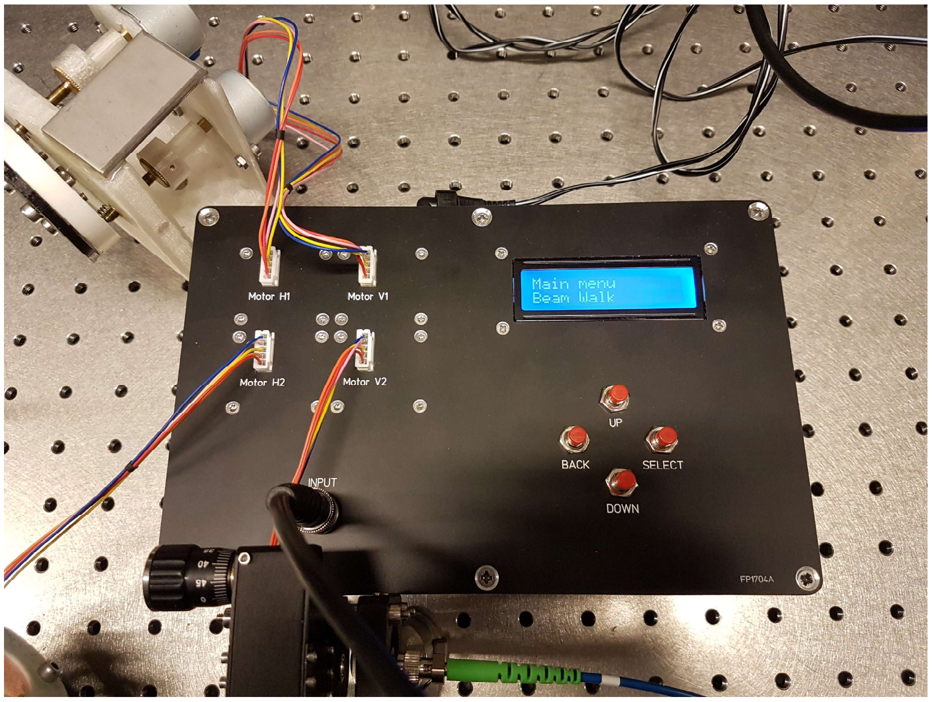
Figure 4.8: Code that creates the menu and displaying it as described by the flow diagram in fig. 4.9a.

sufficient. To develop this interface an electronic box containing the Arduino, driver boards, the electronics and the LCD-display is developed. With the LCD-display it is possible to develop a menu structure where the buttons can be used to navigate and select the desired functions. A flow diagram showing the intended menu structure of all the desired functions are shown in fig. 4.9a. The way this menu structure is written in code is to use the 'struct' type in C/C++ to create a main menu with three sub menus. Each sub menu containing different functions to call. The code to create the main manu along with the sub menus and functions is shown in fig. 4.8 along with the code that display the menu on the LCD-display.

Inside the electronic box are the Arduino connected to the driver boards that control the motors, the input of the power through the fibre, the LCD-display, the four buttons and the power supply. To make all this work there are some connectors into the electronic box. There is a BNC plug for getting the power measure available to the Arduino, a 12 V power supply, a USB plug for connecting a computer to the Arduino and four plugs for the motors which are labelled such that the motors are correctly connected to the Arduino. Most of this can be seen from fig. 4.9b. Inside the box are also the electronics that tells the Arduino if a button is pushed and the electronics that ensure power for the motors and the Arduino.



(a)



(b)

Figure 4.9: (a) Flow diagram of the menu structure for the interface and (b) shows the final electronic box.

5 | Experimental results

This chapter describes how the two algorithms developed in chapter 4 were tested. First, testing of the beam walk algorithm is described in section 5.1 along with the results of this experiment. Then section 5.2 describes the testing of the constant power algorithm along with the result from this test.

Now that the automated kinematic mirror mount and the two algorithms have been developed it was time for them to be tested in an experimental setup to see if they were capable of optimising the coupling of a laser beam into an optical fibre and maintain a high coupling efficiency. The experimental setup was as described in chapter 2 where the second version of the automated kinematic mirror mount was used as argued in chapter 3.

5.1 Beam walk

To illustrate how the algorithm works fig. 5.1 shows one stage of the beam walk algorithm. Figure 5.1a shows how the power through the fibre developed as the motors were turning. Figure 5.1b and fig. 5.1c show the motor position of the 'V1-motor' and 'V2-motor' respectively. The 'V1-motor' was turned in one direction which is then followed by the 'V2-motor' finding the maximum power just as described earlier in section 4.2. To clearly see how the algorithm works one can see that the first motor changes position while the power drops. The motor then stops and the second motor starts turning in one direction until the power reaches a minimum. The motor then reverses direction and turns while the power rises and falls again until the motor stops. Finally the motor reverses direction again and continues until the power comes close to the maximum. This is continued until it stopped at one of the different possible situations described in section 4.2 ia met.

To test how well the beam walk algorithm performed there was a need to develop some sort of standard way of testing this. The way this was done in this experiment was to develop a new algorithm that misaligned in some 'standard' way such that the results were comparable. The algorithm to achieve this picked one motor randomly and turned it in a random direction until the power dropped below 95 % of the initial power. Then the algorithm picked another motor randomly and turned it in a random direction until the power dropped below 95 % of the power before that motor started turning. This was continued until the power dropped below 30 % of the power before the algorithm started. Thus the mirrors were misaligned in a random way each time but the power was always below 30 %, so the starting point for comparing each run of the beam walk algorithm was similar. This was chosen

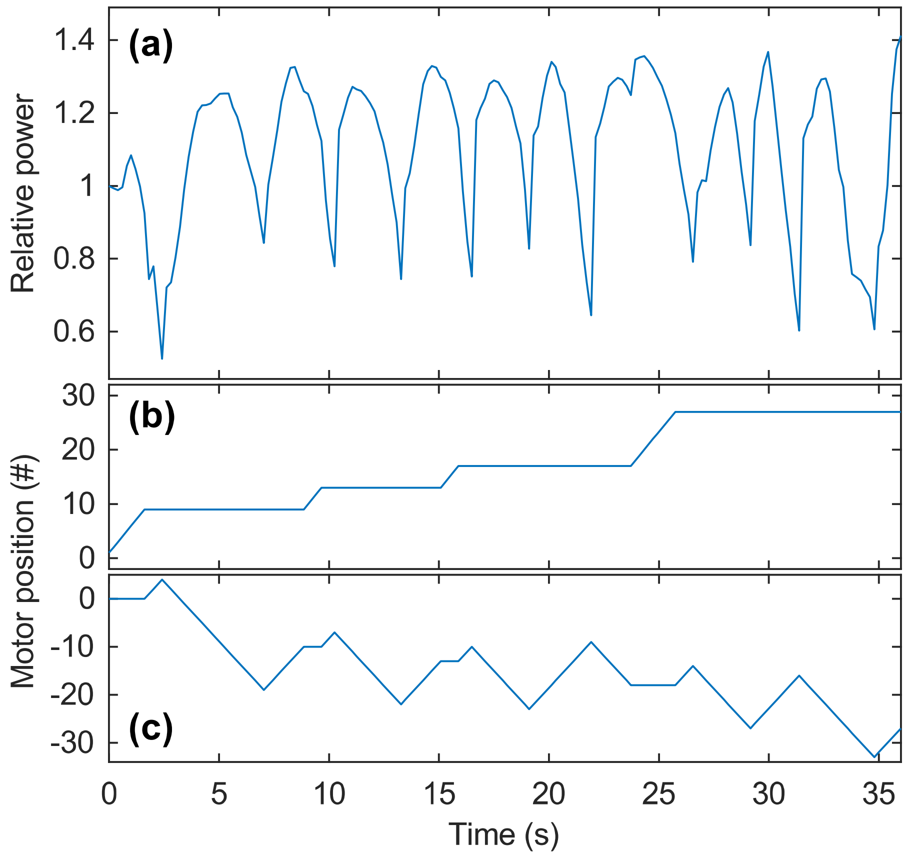
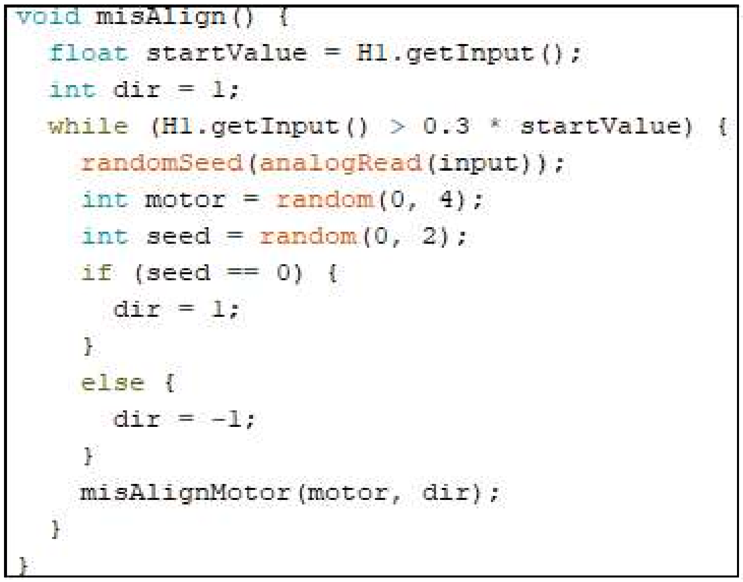


Figure 5.1: Single step of the beam walk algorithm along with the motor positions. (a) How the power developed as the motors were turned and (b) and (c) show the motor position for the vertical motor on mirror one and two respectively.

to try to simulate the situations in real experimental setups where a laser beam is coupled into an optical fibre. As explained in chapter 4 the algorithm needed enough power so that the signal/noise ratio was high enough and hence 30 % was chosen. The code for the misalign algorithm is shown in fig. 5.2.

The way the algorithm was tested was to manually beam walk the laser beam using the commercial mirrors in the setup until finding the maximum power. This power was then measured as a reference point for the algorithm and used to normalize the data. Then the misaligning algorithm was run before the beam walk algorithm was activated. This was done multiple times to obtain a large amount of data. The results are shown in fig. 5.3 where each single run and a histogram of the final intensities are shown. As seen from



```
void misAlign() {
  float startValue = H1.getInput();
  int dir = 1;
  while (H1.getInput() > 0.3 * startValue) {
    randomSeed(analogRead(input));
    int motor = random(0, 4);
    int seed = random(0, 2);
    if (seed == 0) {
      dir = 1;
    }
    else {
      dir = -1;
    }
    misAlignMotor(motor, dir);
  }
}
```

Figure 5.2: Code for the misalign algorithm that was used to test the beam walk algorithm.

fig. 5.3a the algorithm in most cases quickly obtains an power above 80 % and thereafter the improvement was more slow. The blue lines represents the runs where the final power was above 90 % and the red ones are the ones where the final power was below 90 % In most of the cases the intensities increased to above 90 % but in some cases the intensities dropped below again and did not recover. In these cases it was most likely deformation or vibrations from the motors that resulted in the lack of ability to recover intensities above 90 % again. Figure 5.3b summarizes the final intensities of the runs in fig. 5.3a. It is clearly shown that in most cases the power was above 90 % and this showed that the algorithm was capable of securing a high coupling efficiency. In few cases the power exceeded 100 % but when looking at the data they were less than 1 % above. This was likely due to the human optimisation that was not completely optimal or small changes in the power in the system during the collection of the data. As described in chapter 3 the automated kinematic mirror mount had problems with hysteresis and deformations and that was why the tests sometimes failed. This was also the case with the beam walk algorithm, so the data shown in fig. 5.3 does not indicate all the runs taken. In around 20 % of the runs the algorithm failed and all signal through the optical fibre was lost. This often happened when a new motor should start turning which probably resulted in vibrations and then signal loss. So there is room for improvement to achieve a higher stability for the algorithm.

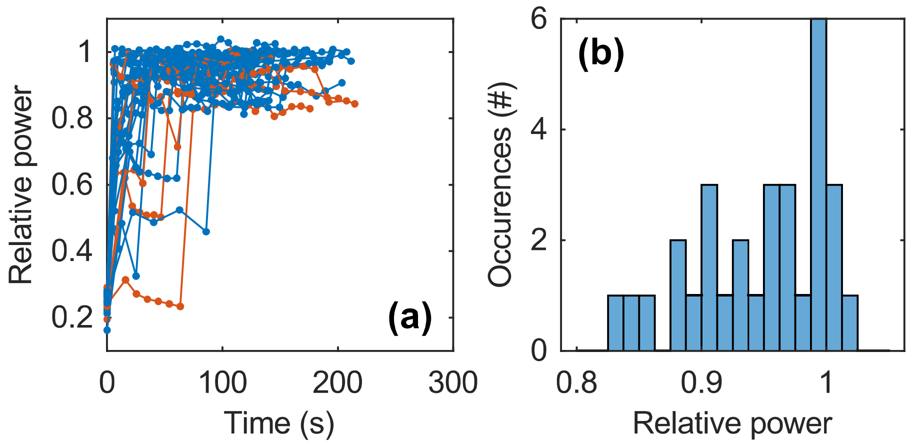


Figure 5.3: Performance of the beam walk algorithm where (a) shows the single runs and how the power through the fibre developed over time and (b) shows a histogram over the final intensities of the runs measured.

5.2 Constant power

To test how well the constant power algorithm performed the system was optimised and then the target value was set. The algorithm was then activated and the algorithm was adjusted so it checked the power once every minute. Then the data was recorded for a period of around 18 h. To see if the power had a different development when the algorithm was not activated, the power was measured without the algorithm as well. The results of these measurements are shown in fig. 5.4 where the power developments with and without the algorithm are shown in fig. 5.4a and a histogram of the measured intensities in the period where the algorithm was activated is shown in fig. 5.4b. It is clear from the figures that the algorithm was able to maintain a higher and more stable power over time since fig. 5.4a clearly shows that the power decays over time when the algorithm was not activated. The power with the algorithm activated suffers from large power fluctuations over time which are probably caused by the temperature fluctuations in the room which as shown in section 3.2 had an impact on the positioning of the mirrors. From fig. 5.4b it is clear that with the algorithm activated the power lies above 97 % for most of the time. The black circle in fig. 5.4a represents the time were the algorithm made corrections to the mirrors. This was done once during the 18 h and the correction took ~ 7 s. These results show that the algorithm did what it was intended to do and that it kept the coupling efficiency high while it only took few seconds to make corrections to the mirrors when necessary.

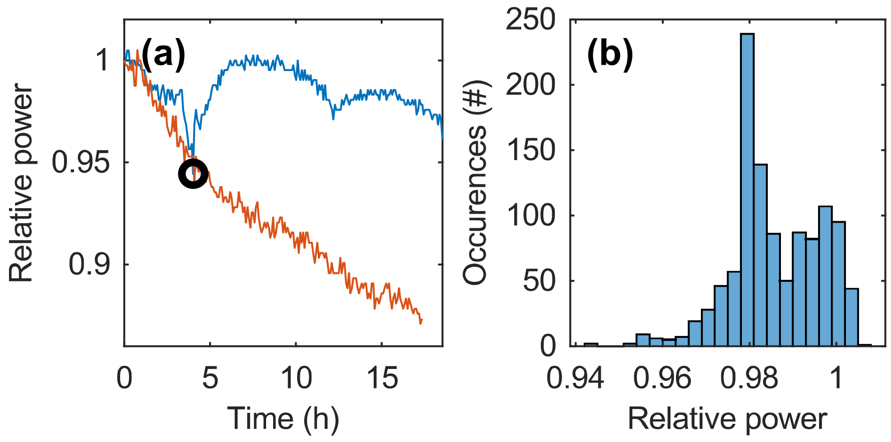


Figure 5.4: Ability of the algorithm to retain a certain value.

6 | Conclusion and outlook

The main result within this thesis is the realization of a 3D printed automated kinematic mirror mount with two different algorithms that are able to first obtain a high coupling efficiency through an optical fibre and second an algorithm that maintain a high coupling efficiency over time.

During the work towards this thesis multiple versions of the automated kinematic mirror mount were developed and chapter 3 presented the design and performance of the different versions. Chapter 3 also showed that the first two versions were the ones that were most reliable whereas the vertical mirror mount suffers too much from hysteresis and plastic deformation to be used in the optimisation process.

Chapter 4 presented the algorithms to realize the automated kinematic mirror mount. A proof of concept was written in LabVIEW that showed that it was possible to automated the coupling into an optical fibre. However, the process was slow and problematic. The beam walk algorithm took several minutes and it needed a computer to run the algorithm. This led to the development of algorithms written directly to the Arduino that supported the mirror mounts. This reduced the time to around a minute and opened up for a variety of options. Thus it was possible for me to develop an interface for the whole system which included a LCD-display that show the menu structure of the interface.

Finally chapter 5 showed the experiments to test how well the algorithms worked in an experimental setup. The results showed that the beam walk algorithm was able to achieve a high coupling efficiency in most of the cases. However, sometimes the algorithm failed and the power dropped without that the algorithm was able to correct this. This shows that there is still work to be done to secure higher reliability. The results for the constant power algorithm showed that it was able to maintain a high coupling efficiency over a long period as compared to the situation where the algorithm was not activated. This was achieved while only small corrections was made by the algorithm. Since the group often need to realign their coupling into optical fibres this could save them for some tedious work.

As presented in this thesis there is still a lot of work that can be done in the future to improve this field. I have presented the idea for a vertical mirror mount and the idea to create a small box containing two mirrors in a compact way. These two solutions need some work to reduce the hysteresis and plastic deformation such that they become more reliable and can be used for automated coupling. In my opinion this needs to be done if 3D printed automated mirror mounts should get at place in real experimental setups.

As future work it would be interesting to test the mirror mount solution presented by Salazar et al. (2018) [11] with the algorithms developed in this thesis. Since the solution is based on commercial mirror mounts the hysteresis is better and the risk of deformation is minimal. This could lead to improvements in the algorithms that I have written since they are limited by the hysteresis and risk of deformation. The use of the automated kinematic mirror mount presented in this thesis is not limited to coupling into an optical fibre but can also be used for controlling laser beams positions in other situations. This is an area that can pursued in future work as well. Thus work automated kinematic mirror mount could hopefully be more widely used in experiments since the solution presented in this thesis is much cheaper than the commercial solutions available.

Bibliography

[1] W. J. Tomlinson, R. E. Wagner, T. S. Stakelon, T. W. Cline, and R. B. Jander, “Coupling efficiency of the optics in single-mode and moltimode fiber components”, in Integrated optics and optical fiber communication, Vol. 21, 15 (1981), TUL2.

[2] J. Gwamuri and J. M. Pearce, “Open source 3-d printers: an appropriate technology for building low cost optics labs for the developing communities”, in Etop 2017 proceedings (2017), 104522S.

[3] T. Baden, A. M. Chagas, G. Gage, T. Marzullo, L. L. Prieto-Godino, and T. Euler, “Open Labware: 3-D Printing Your Own Lab Equipment”, PLOS Biology **13**, e1002086 (2015).

[4] G. C. Anzalone, A. G. Glover, and J. M. Pearce, “Open-source colorimeter”, Sensors (Switzerland) **13**, 5338–5346 (2013).

[5] A. J. S. McGonigle, T. D. Pering, J. R. Willmott, T. C. Wilkes, and J. M. Cook, “Low-cost 3D printed 1 nm resolution smartphone sensor-based spectrometer: instrument design and application in ultraviolet spectroscopy”, Optics Letters **42**, 4323 (2017).

[6] B. Wijnen, E. J. Hunt, G. C. Anzalone, and J. M. Pearce, “Open-source syringe pump library”, PLoS ONE **9**, 1–8 (2014).

[7] C. Zhang, N. C. Anzalone, R. P. Faria, and J. M. Pearce, “Open-Source 3D-Printable Optics Equipment”, PLoS ONE **8**, edited by A. G. de Brevern, e59840 (2013).

[8] S. Bobach, A. Hidic, J. J. Arlt, and A. J. Hilliard, “Note: A portable rotating waveplate polarimeter”, Review of Scientific Instruments **88**, 036101 (2017).

[9] L. J. Salazar-Serrano, J. P. Torres, A. Valencia, J. P. Torres, A. Valencia, J. P. Torres, and A. Valencia, “A 3D Printed Toolbox for Opto-Mechanical Components”, PLOS ONE **12**, edited by L. Carretero, e0169832 (2017).

[10] R. Saint, W. Evans, Y. Zhou, T. Barrett, T. M. Fromhold, E. Saleh, I. Maskery, C. Tuck, R. Wildman, F. Oručević, and P. Krüger, “3D-printed components for quantum devices”, Scientific Reports **8**, 8368 (2018).

[11] L. J. Salazar-Serrano, G. Jiménez, J. P. Torres, L. José Salazar-Serrano, G. Jiménez, J. P. Torres, L. J. Salazar-Serrano, G. Jiménez, and J. P. Torres, “How to automate a kinematic mount using a 3D printed Arduino-based system”, arXiv:1803.06429v1, 1–9 (2018).

[12] R. S. Norup, “3d printed mirror mounts for use in laser laboratories”, Bachelor thesis, 2017, Aarhus University.

[13] L. Ricci, M. Weidemüller, T. Esslinger, A. Hemmerich, C. Zimmermann, V. Vuletic, W. König, and T. Hänsch, "A compact grating-stabilized diode laser system for atomic physics", Optics Communications **117**, 541–549 (1995).

[14] H. K. Andersen, "Opbygning af et rubidium mot lasersystem", Bachelor thesis, 2003, Aarhus University.

[15] Wikipedia, *Fresnel equations*, (Visited 2nd August 2019) `https://en.wikipedia.org/wiki/Fresnel_equations`.

[16] Wikipedia, *Labview*, (Visited 30th July 2019) `https://en.wikipedia.org/wiki/LabVIEW`.

[17] Wikipedia, *Arduino ide*, (Visited 30th July 2019) `https://en.wikipedia.org/wiki/Arduino_IDE`.

[18] Wikipedia, *C++*, (Visited 30th July 2019) `https://en.wikipedia.org/wiki/C++`.

A | Appendix

The complete programming code is presented here:

```
// include the library code:
#include <LiquidCrystal.h>

// initialize the library by associating any needed LCD interface pin
// with the arduino pin number it is connected to
const int rs = 22, en = 24, d4 = 26, d5 = 27, d6 = 28, d7 = 29;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

// Define the buttons used
#define downButton 48
#define upButton 49
#define selectButton 47
#define backButton 46

// Define the analog input channel
#define input 0

// Define some menu types
#define menuTypeFunc   'F'
#define menuTypeMenu   'M'

struct menu_s *menuActual;
float targetValue;
float valueAfterOptimization;
int motorCount;
boolean activateConstantIntensity;
int direction;
int direction1;
int direction2;
float currentValue;
float maxValueAfterOptimization;
int count;
boolean doBeamWalk;
boolean motorMove;
int mirrorCount;

// Define a Motor class.
class Motor
{
    // Class memeber variables
    int cha1;
    int cha2;
    int cha3;
    int cha4;
    int pause;
    float maxValue;
```

```
  // Constructor
public:
  Motor(int pin1, int pin2, int pin3, int pin4, int wait)
  {
    cha1 = pin1;
    cha2 = pin2;
    cha3 = pin3;
    cha4 = pin4;
    pause = wait;
    maxValue = 0;
  }

  // Class methods:

  // This method moves the motor one step in the specified direction.
  void move(int dir)
  {
    int IN1;
    int IN2;
    int IN3;
    int IN4;
    if (dir < 0) {
      IN1 = cha1;
      IN2 = cha2;
      IN3 = cha3;
      IN4 = cha4;
    }
    else if (dir > 0) {
      IN1 = cha4;
      IN2 = cha3;
      IN3 = cha2;
      IN4 = cha1;
    }
    for (int i = 0; i < 9; i++) {
      switch (i) {
        case 0:
          digitalWrite(IN1, LOW);
          digitalWrite(IN2, LOW);
          digitalWrite(IN3, LOW);
          digitalWrite(IN4, HIGH);
          break;
        case 1:
          digitalWrite(IN1, LOW);
          digitalWrite(IN2, LOW);
          digitalWrite(IN3, HIGH);
      digitalWrite(IN4, HIGH);
     break;
  case 2:
      digitalWrite(IN1, LOW);
      digitalWrite(IN2, LOW);
      digitalWrite(IN3, HIGH);
      digitalWrite(IN4, LOW);
     break;
  case 3:
      digitalWrite(IN1, LOW);
      digitalWrite(IN2, HIGH);
      digitalWrite(IN3, HIGH);
      digitalWrite(IN4, LOW);
     break;
  case 4:
      digitalWrite(IN1, LOW);
      digitalWrite(IN2, HIGH);
      digitalWrite(IN3, LOW);
      digitalWrite(IN4, LOW);
     break;
  case 5:
      digitalWrite(IN1, HIGH);
      digitalWrite(IN2, HIGH);
      digitalWrite(IN3, LOW);
      digitalWrite(IN4, LOW);
     break;
  case 6:
      digitalWrite(IN1, HIGH);
      digitalWrite(IN2, LOW);
      digitalWrite(IN3, LOW);
      digitalWrite(IN4, LOW);
     break;
  case 7:
      digitalWrite(IN1, HIGH);
      digitalWrite(IN2, LOW);
      digitalWrite(IN3, LOW);
      digitalWrite(IN4, HIGH);
     break;
  default:
      digitalWrite(IN1, LOW);
      digitalWrite(IN2, LOW);
      digitalWrite(IN3, LOW);
      digitalWrite(IN4, LOW);
     break;
}
delay(pause);
```

```
      }
    }

    // This method updates the maxValue if the given value is higher than
the previous.
    void updateMaxValue(float value) {
      if (value > maxValue) {
        maxValue = value;
      }
    }

    // This method returns the maxValue obtained
    float getMaxValue() {
      return maxValue;
    }

    // This method reads the analog input averaged over 100 measurements.
    float getInput() {
      float total = 0;
      for (int i = 0; i < 100; i++) {
        total += analogRead(input);
        delay(1);
      }
      total /= 100;
      return total;
    }

    // This method resets the maxValue.
    void resetMaxValue() {
      maxValue = 0;
    }

    // This method is a recursive function to stop near a maximum.
    boolean stopMaximum(int maxSteps, float maxLimit, float turnLimit, int
direction2, int depth) {
      if (depth == 0) {
        return false;
      }
      for (int t = 0; t < maxSteps; t++) {
        if (t < 10) {
          delay(20);
        }
        move(direction2);
        if (t > 5) {
          currentValue = getInput();
          if (currentValue > maxLimit * maxValue) {
            return true;
          }
          if (currentValue < turnLimit * maxValue) {
            return stopMaximum(maxSteps, (maxLimit - 0.02), (turnLimit - 0.
05), (direction2 * -1), (depth - 1));
          }
        }
      }
      return stopMaximum(maxSteps, (maxLimit - 0.02), (turnLimit - 0.05),
(direction2 * -1), (depth - 1));
    }

    // This method finds the maximum and stops near it.
    float findMaximum(int maxSteps, float leftCutoff, float rightCutoff) {
      boolean findMaximum = false;
      boolean activateLimit = false;
      currentValue = getInput();
      updateMaxValue(currentValue);
      direction2 = -1 * direction1;
      for (int i = 0; i < maxSteps; i++) {
        if (i < 10) {
          delay(20);
        }
        move(direction2);
        currentValue = getInput();
        updateMaxValue(currentValue);
        if (currentValue < leftCutoff * maxValue) {
          break;
        }

      }
      direction2 *= -1;
      for (int i = 0; i < 2 * maxSteps; i++) {
        if (i < 10) {
          delay(20);
        }
        move(direction2);
        currentValue = getInput();
        if (currentValue > 0.7 * maxValue) {
          updateMaxValue(currentValue);
          activateLimit = true;
        }

        if (activateLimit) {
          if (currentValue < rightCutoff * maxValue) {
            break;
            }
          }
        }
        direction2 *= -1;
        if (stopMaximum(maxSteps, 0.98, 0.6, direction2, 8)) {
          return currentValue;
        }
        else {
          return 0;
        }
      }
};

// Create the 4 Motor objects.
Motor H1(37, 36, 35, 34, 3);
Motor V2(41, 40, 39, 38, 3);
Motor H2(45, 44, 43, 42, 3);
Motor V1(33, 32, 31, 30, 3);

// Initialize the menu structure
struct menu_s {
  struct menu_s *menu[4];
  int index;
  char type;
  struct menu_s *menuParrent;
  char name[20];
  void (*func)();

};

// Function to initialize menu elements
void initMenuElement(struct menu_s **menu, char type, char* name, struct
menu_s *menuParrent, void*func) {
  *menu = malloc(sizeof(struct menu_s));
  memset(*menu, 0, sizeof(struct menu_s));
  (*menu)->type = type;
  strcpy((*menu)->name , name);
  (*menu)->index = 0;
  (*menu)->menuParrent = menuParrent;
  (*menu)->func = func;

}

struct menu_s *mainMenu;

// This method optimizes the intensity by doing a beam walk
```

```
void beamWalk() {
  valueAfterOptimization = 0;
  float maxValueOptimizing = 0;
  float valueOptimizingBefore = 0;
  boolean optimize = true;
  boolean correctDirection = false;
  int count = 0;
  int correctCount = 0;
  H1.resetMaxValue();
  V1.resetMaxValue();
  H2.resetMaxValue();
  V2.resetMaxValue();
  H2.findMaximum(40, 0.7, 0.7);
  maxValueOptimizing = H2.getMaxValue();
  valueOptimizingBefore = H2.getMaxValue();
  while (optimize) {
    while (H1.getInput() > 0.90 * maxValueOptimizing) {
      H1.move(direction1);
    }
    H2.resetMaxValue();
    H2.findMaximum(40, 0.7, 0.7);
    if (H2.getMaxValue() < maxValueOptimizing) {
      if (correctDirection) {
        if (correctCount > 0) {
          optimize = false;
        }
      }
      if (H2.getMaxValue() < valueOptimizingBefore) {
        direction1 *= -1;
      }
      count++;
      if (count == 5) {
        optimize = false;
      }
    }
    else if (H2.getMaxValue() > 1.01 * maxValueOptimizing) {
      maxValueOptimizing = H2.getMaxValue();
    }
    else if (H2.getMaxValue() > maxValueOptimizing && H2.getMaxValue() <= 1
01 * maxValueOptimizing) {
      optimize = true;
      maxValueOptimizing = H2.getMaxValue();
    }
    else if (H2.getMaxValue() == maxValueOptimizing) {
      optimize = false;
    }
    if (H2.getMaxValue() > valueOptimizingBefore) {
      correctDirection = true;
      correctCount++;
    }
    valueOptimizingBefore = H2.getMaxValue();
  }
  optimize = true;
  correctDirection = false;
  count = 0;
  correctCount = 0;
  V2.resetMaxValue();
  V2.findMaximum(40, 0.7, 0.7);
  valueOptimizingBefore = V2.getMaxValue();
  maxValueOptimizing = V2.getMaxValue();
  while (optimize) {
    while (V1.getInput() > 0.90 * maxValueOptimizing) {
      V1.move(direction1);
    }
    V2.resetMaxValue();
    V2.findMaximum(40, 0.7, 0.7);
    if (V2.getMaxValue() < maxValueOptimizing) {
      if (correctDirection) {
        if (correctCount > 0) {
          optimize = false;
        }
      }
      if (V2.getMaxValue() < valueOptimizingBefore) {
        direction1 *= -1;
      }
      count++;
      if (count == 5) {
        optimize = false;
      }
    }
    else if (V2.getMaxValue() > 1.01 * maxValueOptimizing) {
      maxValueOptimizing = V2.getMaxValue();
    }
    else if (V2.getMaxValue() > maxValueOptimizing && V2.getMaxValue() <= 1
01 * maxValueOptimizing) {
      optimize = true;
      maxValueOptimizing = V2.getMaxValue();
    }
    else if (V2.getMaxValue() == maxValueOptimizing) {
      optimize = false;
    }
    if (V2.getMaxValue() > valueOptimizingBefore) {
      correctDirection = true;
      correctCount++;
    }
    valueOptimizingBefore = V2.getMaxValue();
  }
  optimize = true;
  correctDirection = false;
  count = 0;
  correctCount = 0;
  H2.resetMaxValue();
  H2.findMaximum(40, 0.7, 0.7);
  valueOptimizingBefore = H2.getMaxValue();
  maxValueOptimizing = H2.getMaxValue();
  while (optimize) {
    while (H1.getInput() > 0.90 * maxValueOptimizing) {
      H1.move(direction1);
    }
    H2.resetMaxValue();
    H2.findMaximum(40, 0.7, 0.7);
    if (H2.getMaxValue() < maxValueOptimizing) {
      if (correctDirection) {
        if (correctCount > 0) {
          optimize = false;
        }
      }
      if (H2.getMaxValue() < valueOptimizingBefore) {
        direction1 *= -1;
      }
      count++;
      if (count == 5) {
        optimize = false;
      }
    }
    else if (H2.getMaxValue() > 1.01 * maxValueOptimizing) {
      maxValueOptimizing = H2.getMaxValue();
    }
    else if (H2.getMaxValue() > maxValueOptimizing && H2.getMaxValue() <= 1
01 * maxValueOptimizing) {
      optimize = true;
      maxValueOptimizing = H2.getMaxValue();
    }
    else if (H2.getMaxValue() == maxValueOptimizing) {
      optimize = false;
    }
    if (H2.getMaxValue() > valueOptimizingBefore) {
      correctDirection = true;
      correctCount++;
    }
    valueOptimizingBefore = H2.getMaxValue();
  }
  optimize = true;
  correctDirection = false;
  count = 0;
  correctCount = 0;
  V2.resetMaxValue();
  V2.findMaximum(40, 0.7, 0.7);
  valueOptimizingBefore = V2.getMaxValue();
  maxValueOptimizing = V2.getMaxValue();
  while (optimize) {
    while (V1.getInput() > 0.90 * maxValueOptimizing) {
      V1.move(direction1);
    }
    V2.resetMaxValue();
    V2.findMaximum(40, 0.7, 0.7);
    if (V2.getMaxValue() < maxValueOptimizing) {
      if (correctDirection) {
        if (correctCount > 0) {
          optimize = false;
        }
      }
      if (V2.getMaxValue() < valueOptimizingBefore) {
        direction1 *= -1;
      }
      count++;
      if (count == 5) {
        optimize = false;
      }
    }
    else if (V2.getMaxValue() > 1.01 * maxValueOptimizing) {
      maxValueOptimizing = V2.getMaxValue();
    }
    else if (V2.getMaxValue() > maxValueOptimizing && V2.getMaxValue() <= 1
01 * maxValueOptimizing) {
      optimize = true;
      maxValueOptimizing = V2.getMaxValue();
    }
    else if (V2.getMaxValue() == maxValueOptimizing) {
      optimize = false;
    }
    if (V2.getMaxValue() > valueOptimizingBefore) {
      correctDirection = true;
      correctCount++;
    }
     valueOptimizingBefore = V2.getMaxValue();
  }
}

// This method resets the count
void resetCount() {
  count = 0;
}

// This method sets the target value
void setTargetValue() {
  targetValue = H1.getInput() * 0.90;
}

// This method clears the target value
void clearTargetValue() {
  targetValue = 0;
}

// This method activates the constant intensity function
void activateConstantI() {
  activateConstantIntensity = true;
}

// This method deactivates the constant intensity function
void deactivateConstantI() {
  activateConstantIntensity = false;
}

// This method updates the motor count
void updateMotorCount() {
  motorCount = (motorCount + 1) % 4;
}

// This method updates the mirror count
void updateMirrorCount() {
  mirrorCount = (mirrorCount + 1) % 2;
}

//This method tries to maintain a high intensity
void constantIntensity() {
  boolean startCorrection = false;
  float startValue = H1.getInput();
  float previousValue = startValue;
  if (startValue < targetValue * 0.95) {
    startCorrection = true;
    while (startCorrection) {
      switch (mirrorCount) {
        case 0:
          H2.findMaximum(30, 0.7, 0.7);
          previousValue = V2.findMaximum(30, 0.7, 0.7);
          break;
        case 1:
          H1.findMaximum(30, 0.7, 0.7);
          previousValue = V1.findMaximum(30, 0.7, 0.7);
          break;
      }
      updateMirrorCount();
      if (previousValue > 0.96 * targetValue) {
        startCorrection = false;
      }
    }
  }
}

// This method enables us to move the laserbeam manually by using 2 button
for direction
void manualMove() {
  int i = menuActual->menuParrent->index;
  int dir = 0;
  if (digitalRead(downButton) == HIGH) {
    dir = 1;
  }
  else if (digitalRead(upButton) == HIGH) {
    dir = -1;
  }
  else if (dir == 0) {
    return;
  }
  if (i == 0) {
    H1.move(dir);
  }
  else if (i == 1) {
    V2.move(dir);
  }
  else if (i == 2) {
    H2.move(dir);
  }
  else if (i == 3) {
    V1.move(dir);
  }
```

```
}

// This method creates the menu
void initMenu() {
  initMenuElement(&mainMenu, menuTypeMenu, "Main menu", NULL, NULL);
  initMenuElement(&mainMenu->menu[0], menuTypeMenu, "Beam Walk", mainMenu,
NULL);
  initMenuElement(&mainMenu->menu[0]->menu[0], menuTypeFunc, "Activate",
mainMenu->menu[0], beamWalk);
  initMenuElement(&mainMenu->menu[1], menuTypeMenu, "ConstantI", mainMenu,
NULL);
  initMenuElement(&mainMenu->menu[1]->menu[0], menuTypeFunc, "Set target",
mainMenu->menu[1], setTargetValue);
  initMenuElement(&mainMenu->menu[1]->menu[1], menuTypeFunc, "Activate",
mainMenu->menu[1], activateConstantI);
  initMenuElement(&mainMenu->menu[1]->menu[2], menuTypeFunc, "Deactivate",
mainMenu->menu[1],  deactivateConstantI);
  initMenuElement(&mainMenu->menu[2], menuTypeMenu, "Manual move",
mainMenu, NULL);
  initMenuElement(&mainMenu->menu[2]->menu[0], menuTypeMenu, "Motor H1",
mainMenu->menu[2], NULL);
  initMenuElement(&mainMenu->menu[2]->menu[0]->menu[0], menuTypeFunc, "Pus
Up/Down", mainMenu->menu[2]->menu[0], manualMove);
  initMenuElement(&mainMenu->menu[2]->menu[1], menuTypeMenu, "Motor V2",
mainMenu->menu[2], NULL);
  initMenuElement(&mainMenu->menu[2]->menu[1]->menu[0], menuTypeFunc, "Pus
Up/Down", mainMenu->menu[2]->menu[1], manualMove);
  initMenuElement(&mainMenu->menu[2]->menu[2], menuTypeMenu, "Motor H2",
mainMenu->menu[2], NULL);
  initMenuElement(&mainMenu->menu[2]->menu[2]->menu[0], menuTypeFunc, "Pus
Up/Down", mainMenu->menu[2]->menu[2], manualMove);
  initMenuElement(&mainMenu->menu[2]->menu[3], menuTypeMenu, "Motor V1",
mainMenu->menu[2], NULL);
  initMenuElement(&mainMenu->menu[2]->menu[3]->menu[0], menuTypeFunc, "Pus
Up/Down", mainMenu->menu[2]->menu[3], manualMove);
  menuActual = mainMenu;
}

// This method prints the menu to the LCD-display
void printMenu() {
  lcd.clear();
  lcd.home();
  lcd.print(menuActual->name);
  lcd.setCursor(0, 1);
  lcd.print(menuActual->menu[menuActual->index]->name);
}
```

```
void setup() {
  pinMode(30, OUTPUT);
  pinMode(31, OUTPUT);
  pinMode(32, OUTPUT);
  pinMode(33, OUTPUT);
  pinMode(34, OUTPUT);
  pinMode(35, OUTPUT);
  pinMode(36, OUTPUT);
  pinMode(37, OUTPUT);
  pinMode(38, OUTPUT);
  pinMode(39, OUTPUT);
  pinMode(40, OUTPUT);
  pinMode(41, OUTPUT);
  pinMode(42, OUTPUT);
  pinMode(43, OUTPUT);
  pinMode(44, OUTPUT);
  pinMode(45, OUTPUT);

  pinMode(46, INPUT);
  pinMode(47, INPUT);
  pinMode(48, INPUT);
  pinMode(49, INPUT);
  initMenu();
  // set up the LCD's number of columns and rows:
  lcd.begin(16, 2);

  printMenu();
  Serial.begin(115200);

  // Define initial values
  targetValue = 0;
  activateConstantIntensity = false;
  motorCount = 0;
  direction = 1;
  direction1 = 1;
  direction2 = 1;
  currentValue = 0;
  valueAfterOptimization = 0;
  maxValueAfterOptimization = 0;
  count = 0;
  doBeamWalk = false;
  motorMove = false;
  mirrorCount = 0;
}
```

```
void loop() {
  if (digitalRead(backButton)) {
    if (menuActual->menuParrent != NULL) {
      menuActual = menuActual->menuParrent;
      printMenu();
    }
  }
  else if (!strncmp(menuActual->name, "Motor", 5)) {
    manualMove();
    motorMove = true;
  }
  else if (digitalRead(downButton)) {
    do {
      menuActual->index++;
      menuActual->index %= 4;
    }
    while (menuActual->menu[menuActual->index] == NULL);
    printMenu();
  }

  else if (digitalRead(upButton)) {
    do {
      if (--menuActual->index < 0) {
        menuActual->index += 4;
      }
    }
    while (menuActual->menu[menuActual->index] == NULL);
    printMenu();
  }
  else if (digitalRead(selectButton)) {
    if (menuActual->menu[menuActual->index]->type == menuTypeFunc) {
      menuActual->menu[menuActual->index]->func();
    }
    else {
      menuActual = menuActual->menu[menuActual->index];
      printMenu();
    }
  }
  if (activateConstantIntensity) {
    constantIntensity();
  }
  if (!motorMove) {
    delay(100);
  }
  motorMove = false;
}
```