Rehabilitation Knee Brace

- Electronic Progress Monitoring -

Bachelor Project Report

Jacob Lynge

Aalborg University Esbjerg
Department of Electronics & Computer Engineering
Niels Bohrs Vej 8
DK-6700 Esbjerg

Department of Electronics & Computer Engineering
Niels Bohrs Vej 8
DK-6700 Esbjerg
http://esbjerg.aau.dk

AALBORG UNIVERSITY
STUDENT REPORT

Title:
Rehabilitation Knee Brace

Theme:
Bachelor Project

Project Period:
Fall Semester 2016

Project Group:
ED7-4-E16

Participants:
Jacob Lynge

Supervisor(s):
Simon Pedersen

Copies: 2

Page Numbers: 65

Date of Completion:
January 10, 2017

Abstract:

The focus of this report is to create a functioning knee brace prototype, which can be used to monitor the range of movement of surgically repaired knee. This is done by using a sensor to determine the angle between the lower leg and upper leg. Another focus area of this report is to try and implement motor assisted movement in the brace, since this will aid the recovering knee while it is try to regain control.

This prototype will have a controlling unit, which will have to handle the output from the sensor used for angle monitoring, and the control of the motor used for the motor assisted movement part. For the motor assistance part a control system will be implement.

The prototype knee brace will have to be designed and fabricated personally, which means a review of how this can be done will be performed.

The final prototype ended up being fully functional, however the motor used was not implemented on the brace itself, but on a hinge to show the proof-of-concept. The project status can be viewed as a success, where the areas left as proof-of-concepts showed great promise.

Contents

Preface

I would like to take the opportunity to thank John Smith (Fcubed) on thingiverse.com, for publishing his modular knee brace design under the Creative Commons Attribution 3.0 license, which allowed me to use the design, in this project.

I would also like to thank Simon Pedersen for his guidance as supervisor during my project, this guidance has been greatly appreciated.

This project had the focus on how to monitor a knee while it is going through the rehabilitation period after surgery, with focus on angle monitoring and assisted movement.

This is a seventh semester project that started 1^{st} of September 2016, and is to be handed in the 10^{th} of January. The diploma project is credited at 25 ETCS points, which in time equivalent is almost 690 man hours.

Finally, this work has been carried out as a part of a diploma project of the Electronics and Computer engineering education at Aalborg University, Esbjerg Campus in order to fulfill the requirements from the curriculum of the final project.

Aalborg University Esbjerg, January 10, 2017

Jacob Lynge
<jlynge13@student.aau.dk>

Chapter 1

Introduction

This project idea came about, due to I had an accident while playing American football. In a play near the end zone, in the closing minute of a game I prevented a touchdown by tackling the other team's quarterback, in the process my knee twisted in the moment of contact and the anterior cruciate ligament snapped.

I was then told I had two options by the doctors either get surgery or not be physically active, and the latter was is not an option for me, so I went ahead with the surgery. After the surgery I had four months of rehabilitation with a physiotherapist, to start the healing process of the knee.

During rehabilitation of my knee, one of the goals was to ensure I could fully stretch and bend my knee as fast as possible, since this would show that no scar tissue was building inside the knee and constraining it. The progress was hard to monitor while doing the exercises at home, and it was monitored at the physiotherapist by hand/eye.

These were the circumstances which gave me the idea of an electronic knee brace, that would be able to read the angle in the knee joint and possibly assist with the movement.

Chapter 2

Problem Analysis

This chapter will consist of reviews into different topics relevant to the creation of a prototype, the control of the electronics on the prototype and information relevant to using a motor to assist the movement of the knee.

2.1 Project Description

In this section the subject of the project will be presented, and together with some of the implications that the project might have. Lastly, a set of requirements will be established, so it is possible to evaluate the project in the end.

2.1.1 Project Relevance

In Denmark the total number of Anterior Cruciate Ligament(ACL) reconstructive surgeries totaled to 2584 in 2015 [1], every one that has gone through this surgery, will be assigned to a rehabilitation program in order to gain full control of the knee and leg post-operation. This means that these people will go through nearly the same kind of treatment program, where one the goals is to avoiding scar tissue buildup in the knee, by bending and stretching the knee [2].

During rehabilitation from ACL reconstructive surgery, it is crucial to complete all the recommended exercises given by the physiotherapist, these helps both with regaining control and strength of the knee, they will also help the bending abilities of the knee. These exercises will not only be performed at the physiotherapist where you can get assessments on the progress, but also at home where it can be hard to judge the progress. Furthermore, another thing to keep an eye out for is if the knee swells after the day or workout. This is vital to keep an eye on since it is a sign for over straining your knee, which is not good during rehabilitation [2].

When the physiotherapists needs too be precise, in which angle the knee can be bend to, they use a device called goniometer. A goniometer consists of two plastic bars and where they meet is a circle mount which can rotate and there from the angle can be read [3], an image of a goniometer can be seen in Figure 2.1.

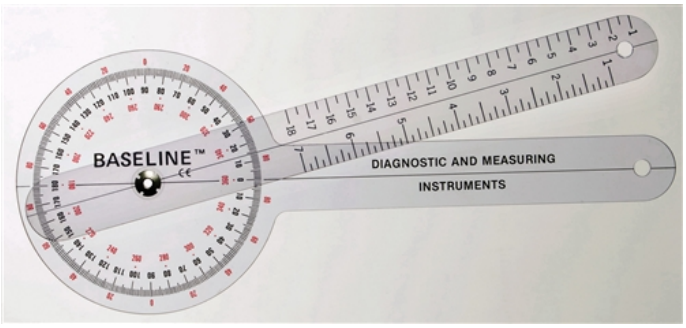

Figure 2.1: Shows a basic goniometer [4].

The goniometer works in the way that you place it, in the case of knee injury, with the baseline at going up the thigh with the moving part at the knee and the other arm going down the lower leg, and then the angle can be read. This goniometer works well and is being used around the world.

A knee brace is designed to give support to the knee during different activities. There are different kind of knee braces for different kind of activities, however currently there are no knee brace with angle measurement that can be used during rehabilitation.

With all of these points in mind, it is clear that a relevance for the project has been established, and a need/marked has been determined where the product can be used.

2.1.2 Project Delimitation

The main focus of this project will be on monitoring the recovering knees bending abilities by making a device, like a knee brace, which has two sets of braces, one on the inner side of the leg and one on the outer side. In these braces on the side of knee will be an angular sensor which will relay the measured angle to a control unit.

The second focus area would be to implement motor assisted movement, this would be beneficiary for the person during rehabilitation, since it would be able to assist the knee in learning the range of movement. This may only be a proof-of-concept, however if this is successful it should only be a matter of scaling the motor so it have the torque required for moving the lower leg, in order to implement it on the knee brace.

Angular measurement and motor assisted movement are the main goals of the project, however if time in the end will allow for further project work, it could be optimal to implement a sensor to monitor the force of which the lower leg is moved with, since this shows an indication of how the rehabilitation is going. In addition to this, during rehabilitation, a way of measuring the swelling at the knee might be a potential improvement, since this could also indicate the progress of the rehabilitation. However, since time is scares the main focus will be as follows:

- Review angular monitoring technologies
- Review the natural movements of the knee
- Development and testing of the prototype

- Implementing motor assistance

To evaluate the project a set of requirements has to be established, and these should be taken into consideration during the information gathering while also be applicable for the prototype.

No.	Requirements	Accomplished
1.	Angle Monitoring:	
a.	Implementable	
b.	High precision measurements	
c.	Does not hinder movements	
2.	Control Unit:	
a.	Handle analog inputs	
b.	Generate PWM signal	
c.	Display data	
d.	User friendly	
3.	Prototype:	
a.	A fabricated knee brace	
b.	Implemented angular sensor	
c.	Implemented motor assistance	
d.	Wearable	
e.	Low cost	

Table 2.1: Requirements

In relation to the allocated time, it has been decided not to pursue a prototype that fulfills laws, regulations and standards that focuses on wearable electronics and medical equipment. However, this area would be something to take into consideration if the prototype should be produced.

2.2 Angle Monitoring

The following sections reviews different technologies that can be used as sensor for detecting the angle in the knee joint. This is done in order to make the educated decision on what sensor to use for the knee brace prototype.

The different kinds of technologies that will be covered in this section is the following:

- Rotary encoders
- Potentiometers
- Motors with position feedback
- Bend sensor

2.2.1 Rotary Encoders

The two kinds of rotary encoders that will be reviewed here is optical and magnetic.

Encoders functions by detecting a change in position, and this can be done in two ways. The first is an incremental encoder which relays a signal that contains the information about the change from prior position, without having any knowledge of what the current position is. The second type is called absolute encoders which relates the precise position (absolute position) after a change in the position, this can be done since the signal is unique for each bit of movement that will happen to the sensor [5]. After this movement, the signal is typically then transferred to a controlling unit, where it is interpreted and from this, it can be calculated how much the disks position has changed.

Rotary encoders comes in a wide range of versions, which is also why it should be possible to find one that would have high enough precision level, and one that would fit on the knee brace. Furthermore, depending on which kind of rotary chosen, there would be implementation challenges with integrating the encoder into the knee hinge, which would demand the hinge being designed specifically for that encoder.

Optical Enconder

One of the more commonly used types of encoders are optical encoders. Looking at Figure 2.2, several components can be seen and these are the bare essentials for an encoder.

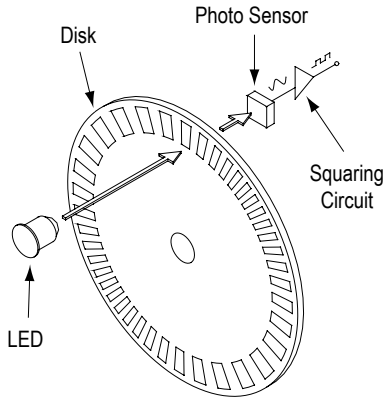

Figure 2.2: The esssentials of an optical encoder, with a squaring circuit [5].

The encoder, seen on Figure 2.2, functions by the LED is constantly emitting light, and depending whether there is a hole in the rotatory disk or not the photo sensor should generate a high or low signal. In this specific circuit seen on Figure 2.2, a squaring circuit (smith trigger) has been attached to the circuit, this makes it easier to identify if the signal is HIGH or LOW. It is then this signal that is transferred to the controlling unit, and based on this information it can determine the movement of the encoder disk. However, the signal can also be combined with other electronics in the encoder such as a counter for

example, then it is this count the will be send to the controlling unit.

Some optical encoders comes with a technology called quadrature, this technology basically if fitted onto an incremental encoder converts it an absolute encoder. What is does is fairly simple, it has two channels which will either send a high or low signal, and based on this it can be determined which way the rotary encoder is rotating and also the speed if wanted [6]. For a graphical overview of how quadrature works see Figure 2.3.

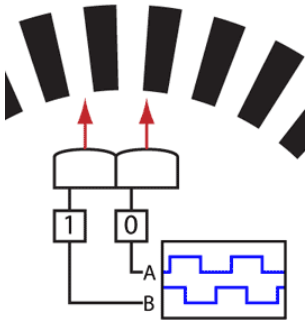

Figure 2.3: Shows an overview of quadrature technology [6].

Figure 2.3 shows that input channel B sends a high signal when hitting a reflective surface on e.g. gear, and when its not hitting an area of reflection (hole in the gear), it returns a low signal. From combining these two signals, it will be possible to tell which way the gear is turning. This would be useful when detecting an angle, since it would assist in tracking the movement of the disc/gear.

Hall Effect Magnetic Encoder

The Hall effect magnetic encoder consists of multiple components that would not be found in an optical encoder, as the name indicates it uses Hall effect. A Hall effect magnetic encoder consist of Hall effect sensors that is placed evenly spaced around the shaft. A Hall effect sensor looks for a change in magnetic field strength or in other words magnetic flux density (see Figure 2.4).

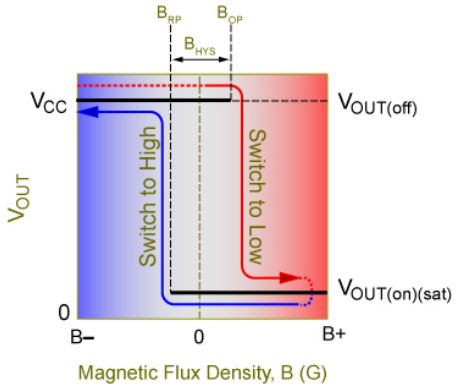

Figure 2.4: Shows the practice of a Hall effect sensor [7].

From Figure 2.4 when there is a high enough positive change in the flux density the sensor will go to a state of low, and when the flux density lowers it produce an output

state of high [7]. Figure 2.4 also shows that an Hall effect sensor has hysteresis built into it, this is done in order to secure smooth transitioning, and not to have the sensor caught in the middle switching state all time due to a low change in magnetic flux density [7].

Besides the Hall effect sensors, a magnetic wheel is also used which goes into a mounting, and it has a certain set of magnetic poles. When the shaft rotates, it results in the Hall effect sensors will change their outputted state a number of times, and based on this number of changes it will be possible to detect a change in position.

2.2.2 Potentiometers

A potentiometer is a type of circuit which typically has three pins, which would be a connection for ground, one for voltage in and one for voltage out. Potentiometers is basically a resistor, where it is possible to change the resistance by turning the shaft, and based on this position and resistance, there will be a specific voltage output. Before, saying that a potentiometer could work as a solution for monitoring the angle between the lower leg and upper leg, there are a few requirements for the potentiometer, if it is chosen.

1. It demands a linear increase in resistance, since the position of the shaft is vital to be able to calculate the angle. Based on the resistance, it should be determinable how many degrees it has been turned from a certain position.
2. It would require that the potentiometer has a high resolution and is gliding smoothly. This is as important as the first requirement.
3. A requirement of its size and shaft length, since it is important to keep in mind that it will have to sit at the side of the knee, where having a big and pointy device sitting would not be optimal. It should be aimed, if the potentiometer is chosen, to acquire a fitting sized device for this type of solution.

These requirements should be met for the potentiometer to be a usable for the purpose.

Figure 2.5: Shows the concept of a potentiometer in a hinge [8].

In Figure 2.5, a potentiometer can be seen mounted in a door hinge, now while this is not the same as a knee brace, the general idea is illustrated. The potentiometer will be

mounted into the hinge, and then depending on how the hinge move it will give a certain output, and this can be used to calculate the current angle. This way of mounting would demand a hole in the knee hinge where the shaft could be fasten, which would in terms be the same principle as Figure 2.5.

2.2.3 Motors with Position Feedback

The idea with using a motor that has position feedback for angle monitoring, is that it will have the motor and angle sensor in one combined unit. This section will only concern angle monitoring, for information on motor assisted movement see Section 2.6-Motor Assisted Movement.

Motors with position feedback basically utilizes one of the previously explained technologies, to give the position feedback, and based on this feedback the angle can be determined. This will be determined by leading the output to an ADC, and make an analysis based on what the ADC outputs.

If the motor has a high resolution i.e. high precision, low error percentage in the feedback, and it is smooth moving it should be possible to use this in combination with the knee brace. Furthermore, the motor should move smoothly, so it do not have an effect on the movement of the knee.

2.2.4 Bend Sensor

A bend sensor is a variable resistors that works on some of the same principles of a strain gauge, depending on how much the sensor is bend, it will then increase or decrease its resistance (see Figure 2.6 and Figure 2.7 for illustration of this). This sensor could be possibly be attached on the side of the knee brace hinge. Figure 2.6 and Figure 2.7 illustrates, the mounting principle discussed at a fully stretched knee and and a knee bend at 90 °, where the brown line illustrates the bend sensors placement this placement will not hinder movements since it is on the exterior of the brace.

Figure 2.6: Illustrates the mounting idea of a bend sensor.

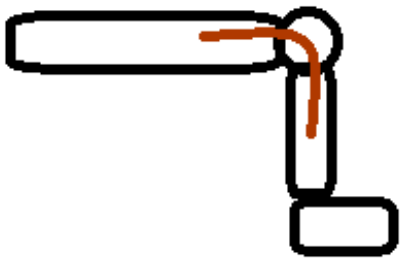

Figure 2.7: Shows the bend sensor bend 90 °with the knee.

Depending of what kind of bend sensor is chosen, the way to measure the angle will have to be fitted to that specific sensor. However, common for nearly all of the bend sensors, is that they can be placed as a resistor in a circuit, and the a V_{out} can be measured

right after the bend sensor, and this will be led to an ADC to be analyzed. Based on the ADCs output it will be possible to calculate the current angle.

2.2.5 Fulfillment of Requirements

In Section 2.1.2-Project Delimitation three requirements for the sensor used for angle monitoring, were established and these can also be seen in the listing below. This section contains a walk through of each of the technologies fulfillment of the requirements, and reasoning why it does or does not fulfill the requirements will be presented.

The requirements for an useful angular sensor is:

1. Implementable
2. High precision measurements
3. Does not hinder movements

Requirement 1 is going to be evaluated based on a scale from 1-5 on how difficult it would be to implement the sensor in/on the knee brace prototype, where 1 indicates low level of difficulty and time spent on modifying the standard hinge to fit the sensor. Where 5 would indicate high difficulty, and show that it would take a great amount of time to implement the sensor in/on the prototype. The grading is solely based on personal opinion and visions for the implementation.

Rotary encoders are evaluated with a 5, this is due to the knee hinge must go through extensive remodeling, in order for it to accommodate the encoder. Potentiometers are evaluated with a 3, this is due to it will present a challenge and will take some time to make a model for the implementation of the sensor. Motors with position feedback is not evaluated, since it will not take any extra time to implement the angle sensor, ones the motor has been fitted to knee hinge, since it is in the motor, and therefore it does not demand extra attention for this specific purpose.

Lastly, the bend sensor has been evaluated at 1, since it will be of light difficulty to implement and it will not demand any remodeling of the knee hinge. Furthermore, it will be mounted on the exterior only two holders will have to be created (see Figure 3.3 in Section 3.1) .

For evaluation purposes the four technologies has been split into the following two groupings:

- Group 1: Rotary encoders, Potentiometers and Motors with position feedback
- Bend sensors

Requirement 2, high precision measurements, group 1 shows great promise of fulfilling this requirement. Since, the technologies comes in a wide product range to choose from, it will be possible to find one precise enough for the prototype, therefore group 1 fulfills this requirement. Moving on to the bend sensor, after observing which kinds of bend sensor that are on the marked, one thing that stands out is the % deviation which varies a lot depending on the specific sensor. However, the deviation problem is something to deal with if the bend sensor is chosen as sensor for angle monitoring. Therefore,

due to the unknowns of the bend sensor it will be evaluated as it might fulfill requirement 2.

Now for requirement 3: does not hinder movement, this requirement is crucial since if the movement of the knee is strained in anyway it might hurt the recovering knee. All of the sensors should be able to be implemented in a way, which would not hinder the natural movement of the knee.

The evaluation table, presented below, is filled based on the previous paragraphs.

Requirements	**Implementable**	**High Precision Measurements**	**Does not hinder movement**
Rotary Encoders	5	✓	✓
Potentiometers	3	✓	✓
Motors with position feedback	N/A	✓	✓
Bend Sensor	1	≈	✓

Table 2.2: Compliance of Requirements

Table 2.2 shows the fulfillment of requirements where a ✓ shows fulfillment, a ≈ shows that it might fulfill the requirement, and a scale of 1-5 on how hard it will be to implement the sensor as angle monitor sensor, where 5 is the hardest.

2.3 Controlling Unit

In this section there will be a look into different kinds of controlling units and based on the criteria's in the problem delimitation, the simplest and smallest one that fits the requirements will be used. The two controlling units (CUs) that will be reviewed are:

- Arduino Uno
- Raspberry Pi

These two CUs has been chosen due to; familiarity with them from past semester projects, and they should be manageable to fit on a knee brace without disturbing the function of the brace and the knee. The two CUs, were also chosen due to, they are present on the university campus and would not cost anything to acquire for the project.

2.3.1 Arduino Uno

The Arduino Uno that might be used for this project is a revision type 3, which is the newest in the line of Arduino Unos [9]. Looking at Figure 2.8 it can be seen that the board contains a lot of different components, and the core component is the microcontroller ATmega328P. Also in Figure 2.8 it can be seen that the board has sections of pins namely, power pins, six analog pins, and 14 digital pins. Some of these pins has multiple purposes E.G. digital pin 3 can be used for normal operations, PWM generations and can handle interrupts, this is also marked on Figure 2.8.

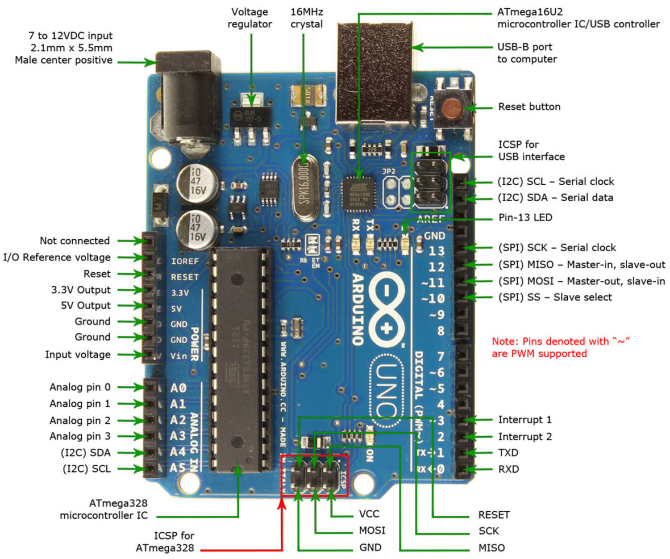

Figure 2.8: An schematic overview of the Arduino Uno [10].

The Arduino Uno does not in itself have a network accessing possibility, however on of the best things about the Arduino Uno, besides the simplicity, is that it has the possibility of choosing from a variety of add-ons or "shields" as they are normally referred to. This essentially means that it would be possible, through a shield, to enable network access or mount a display for a visual output [9].

The Arduino Uno can be programmed by using the Arduino IDE which, also launched with the first revision of the Arduino. The IDE is using the programming language C [9].

2.3.2 Raspberry Pi

The Raspberry Pi is the version 3 model B. Looking at Figure 2.9 it can be seen that the Pi 3 has 40 general purpose input/output pins, some of these pins might also have a specific purpose such as a SPI function. It can also be seen in Figure 2.9, that it has different utilities such as; 1.2 *GHz* Qaud Core CPU, 4 USB ports, HDMI output, Micro SD card slot, 1 *GB* Ram and an Ethernet port. The Pi3 also has the ability like the Arduino Uno Rev 3, to do hardware PWM and it can also act on hardware interrupts [11]. Furthermore, in the Pi3, it has on board Bluetooth and Wi-Fi utilities enabling wireless communication with out further addons.

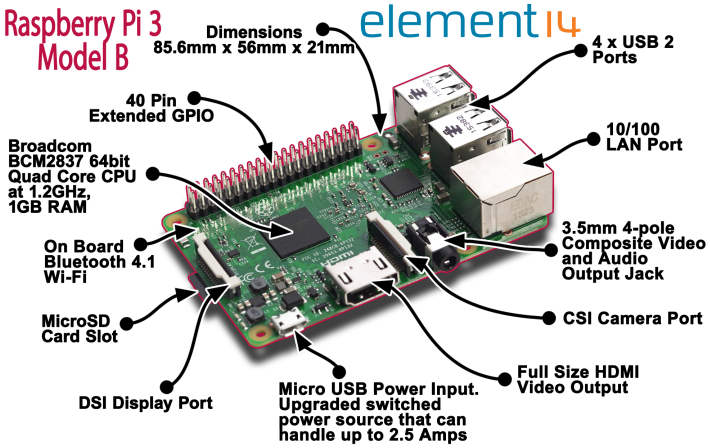

Figure 2.9: An schematic overview of the Raspberry Pi 3 model B [12].

The Pi3 is a little more complex to work with than an Arduino Uno, and this is due to it being a computer with the operating system Raspbian, which is based on Debian. This means it can run different at programs at ones, and not just one program as the Uno.

Programming on the Raspberry can be done in bunch of different languages such as Python, Java and C. However, depending on which context the code is needed to be used in wrappers might be needed, to facilitate better/faster execution or access to hardware on the board.

2.3.3 Fulfillment of Requirements

In this section, there will be an evaluation on whether the two CUs fulfill the pre-established requirement in Section 2.1.2-Project Delimitation. They are the following requirements:

1. Handle analog inputs
2. Generate PWM signal
3. Display data
4. User friendly

These four requirements will be aid in deciding which to choose for the project. Requirement 1 to be able to handle analog inputs, the Arduino Uno does poses pins specific for analog inputs, with a 10bit ADC, so it fulfill this requirement. However, this is not the case with the Pi3, due to the Pi3 does not contain an ADC it cannot process analog inputs with out an external circuit running. This essentially means requirement 1 is not possible, however with the ability to attach an external circuit it is still a might work, therefore the requirement for Pi3 will be marked as might fulfilled.

Requirement 2, which is to be able to generate PWM signal. Both the Uno and the Pi3 does have the ability to generate a PWM signal, which means they fulfill this requirement.

When thinking of measuring data like angles, it would be optimal to have a way of displaying the data to the user of the knee brace, and this is why requirement 3 was established. The Pi3 does not have an inbuild screen on the board, however it does contain an HDMI port, which could be used to display data. The Arduino can display data through a shield mounted onto it. Thus, both the CU candidates posses the ability to display data.

Requirement number 4 is, being user friendly, this is important as it might not be the most tech-savvy people using the brace. The Uno does not require the user to start the program on it, with the only necessary task powering the Uno, which would launch the program, thus this unit is user friendly. It is a bit different with the Pi3 since it a computer, as it requires the user input to run the program, while also requiring powering. While it may not be a very difficult task to start these files, it cannot be guarantied everyone will be able to do this, thus the Pi3 fails at this specific requirement.

After going through each of the requirements with the Pi3 and the Uno, the following table can be filled with marks on whether or not they fulfill the pre-established requirements for the CU.

Requirements	Handle analog inputs	Generate PWM signal	Display data	User friendly
Pi3	≈	✓	✓	✘
Uno	✓	✓	✓	✓

Table 2.3: Compliance with requirements.

Table 2.3 shows the fulfillments of requirements where a ✓ shows fulfillment, a ≈ shows that it maybe that the technology can fulfill the requirement, and a ✘ shows that it does not fulfill the requirement.

2.4 The Knee

When developing a product, it is important to remember the environment the product is going to be used in, and with a knee brace it is no different. In order to develop a prototype/product that does not hurt the knee while in use, it is important to take a look at the built and function of the knee, while also considering the natural movements that the knee can perform with the brace on.

2.4.1 The built of the knee

The leg consists of three major bones which joins together at the knee joint, where one of them is the femur which is the bone that goes from the knee to the hip socket. The other one is the Tibia which goes from the knee down to the angle, and the last one is the Fibula which runs along the Tibia. Where the Femur and the Tibia meets is the knee joint, and it is here the ligaments that hold these two bones together resides. In Figure 2.10 the structure of the knee can be seen.

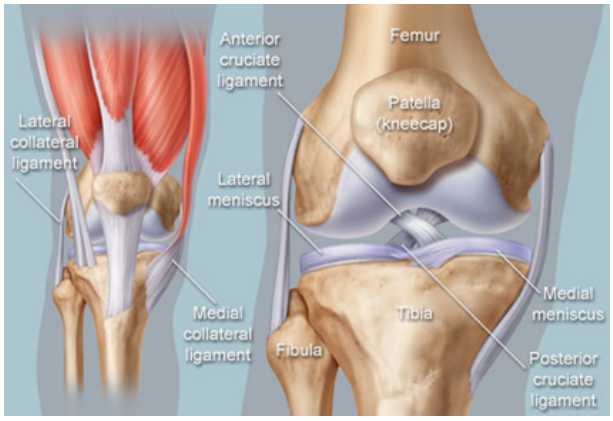

Figure 2.10: An overview of the structure of the knee [13].

In Figure 2.10 it can be seen that there are four ligaments, one on each side of the knee this prevents the knee from slipping vertically out of the joint, where the Anterior Cruciate Ligament (ACL) and the Posterior Cruciate Ligament (PCL) holds the femur and tibia together. The layer of meniscus is on the top of the tibia which ensures, together with the articular cartilage, that there are no grinding bone against bone. Also visible on the Figure 2.10 is the kneecap (patella) and it is suspended by the quadriceps and the patella tendon over the front of the knee. The patella, and tendons attached to it, guides the knee when it is moving and it also makes it less likely that femur will slip of the tibia and tear something in the knee.

Therefore, it is very crucial not to make a brace that will ruin the function of this complex joint, the brace should stabilize not destabilize the joint.

2.4.2 Movements of the Knee

A knee going through rehabilitation, will not be acting like an uninjured knee which have been used for years. This is due to an uninjured knee will not be as wobbly and it would not tend to over extend as much as an injured knee might.

The knee will be moving forward and backwards as it is supposed to, however since some of the control might be missing from the knee, there are a wide range of movements that might happen during this process.

From personal experience, getting reconstructive surgery to the anterior cruciate ligament, also meant losing control over the motor coordination of the knee joint area. After the surgery, a wobbly feeling in the knee showed during rehabilitation, which sometimes meant the knee would act unintentional. Also, the knee would tend to over extending, due to lose of control with the femoral muscle. Due to this lose of control, the knee would move in the ways which can be seen in Figure 2.11.

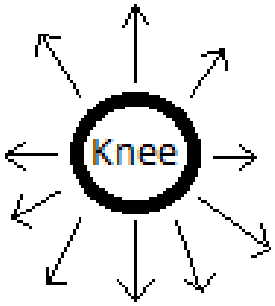

Figure 2.11: An overview of expected movement directions in early rehabilitation.

The movement range seen in Figure 2.11, if not kept to a minimum during rehabilitation, will have a negative effect to the healing knee. Thus, when designing a knee brace it would be a good idea to incorporate support in the brace, to assist the knee in avoiding this range of unwanted movements. However, it clearly has to be noted that an knee brace might hinder the unwanted movements or try to, but when the rehabilitation process is finished and control is regained the expected movements of the knee can be seen in Figure 2.12.

Figure 2.12: An overview of expected movement directions after completed rehabilitation.

Figure 2.12 shows the normal function of a knee, which is the movement forward and backwards. This is also the wanted range of movement during rehabilitation, since other movement ranges might cause harm. However, at some point when it is time to take the brace of, the knee should be trained in movements in all directions as can be seen in Figure 2.11.

2.5 Knee Brace Materials

There are a lot of different designs for knee braces, whether it is a brace made from some foam and a neoprene sleeve, or it is a brace made from metal and/or plastic. The materials vary a lot from brace to brace, therefore it should be possible to find a brace that suits the project.

However, as the in the requirements from the project delimitation states that the implementing motor assistance and angle measurements, it would be beneficiary to build

a frame of plastic or metal that would look like Figure 2.13, so it could be reviewed and altered as desired.

Figure 2.13: Example of a knee brace design [14].

The knee brace seen in Figure 2.13 is from the producer Donjoy and is of the model Legend, and this brace provides supports for injuries to the different cruciate ligaments [15]. This brace consists of; two support beams running down by the side of the knee, a bar going over the thigh, a bar going behind the lower leg, and straps. At the side of the knee joint is stabilizers to avoid the knee sliding sideways.

It would be ideal to pick materials which are light weight and be comfortable during use of the brace. There are two options for the prototype in regards to the support beams they are, plastic and metal.

Being able to 3D print the brace, would allow for customizing the brace so it can have a near perfect fit on every leg in theory. A crucial thing to consider, if this method is chosen, is how high density level is necessary so that the plastic frame doesn't break from twisting the knee. Furthermore, plastic might not be very good at handling the stress and twists an assisting motor. However, this would depend on the choice of plastic, and since it is possible to get plastic which should be as strong as metal [16], this will be something to take into consideration while implementing a motor.

Metal is also an option for a very stiff frame and depending on the thickness of the frame it will way less and less the thinner it gets. However, metal would provide excellent support if the to counter instabilities in the knee to prevents shifts in the knee joint. A metal frame should also be able to handle the stress from an assisting servomotor.

When looking at metal vs. plastic, in regards to the project when it comes to production, it would be easier to get a designed 3D printed plastic frame made, since there are a lot of 3D printing options on university campus. Furthermore, it would be easier to get parts for the brace reproduced if something breaks during testing it is just to print a new one. Whereas, if the custom metal frame breaks it might require a new metal frame piece being made from a metal worker.

2.6 Motor Assisted Movement

The second focus of the project, after angle measurement, is to implement movement assisted by a motor. This is due to that a freshly operated knee joint will quickly become less powerful, and therefore it could be beneficiary with a motor to assist with the movement of the lower leg during rehabilitation. However, it is vital to construct proper boundaries for the motor, since unwanted forced movement of the knee, might damage the knee.

There are many different kinds of motors that could be used for this purpose such as; stepper motors, servo motors, and AC/DC motors. One of the important things when wanting to use a motor to assist with the movement of the lower leg, is that the motor must be able to move in both directions (clockwise and counterclockwise), if the motor is unable do this it can not be used in this context. Another thing that is important to note, when choosing a motor, is that it should be able to be placed at the side of the knee. Thus, it may not be too heavy as this may hinder proper movement of the leg/knee.

For the prototype a motor, such as a servo motor, with a limited range of movement of $0° \rightarrow 90°/180°$, would be a usable option.

2.6.1 Controlling the Assisted Movement

There are different kinds of control systems such as; open loop system, closed loop system. Common for both of these system is that there are some physical action that is being modeled, based on this model the controller can be made.

Figure 2.14: Shows basic components of a control system.

Figure 2.14 shows the basic components of a continuous time control system, however depending on which kind of system that is going to be used, there might be additions to these blocks and or replacements of some of the blocks. The blocks seen in Figure 2.14 are a input block, a PID-Controller block, transfer function block and a output block. The transfer function is based upon data reads from the actual system, and it is used to describes the behavior of this system in simulations.

The reason why a PID-controller is present here even-though there are a lot of different controller types, is due to PID-controllers are the most common in industrial control systems with feedback [17]. The PID-controller consist of a special setup of blocks, which can be seen in Figure 2.15 below.

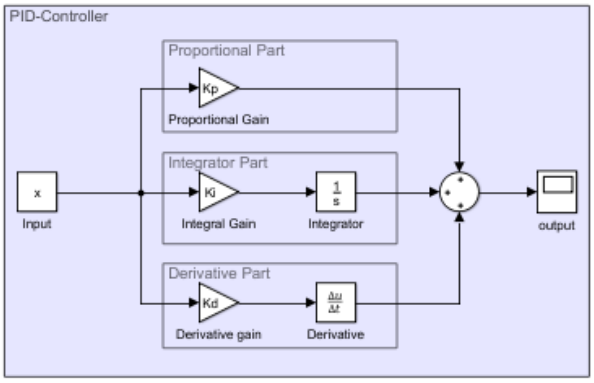

Figure 2.15: An overview of the components that makes a PID-Controller.

Figure 2.15 shows the three parts that make up a PID-controller namely, the proportional part, the integrator part and the derivative part. It can be seen in Figure 2.15 that the proportional part is just a gain added to the incoming data. It also can be seen that the integrator part consist of a gain and an integral block, and the derivative part is consisting of a gain and a derivative block. Finally, these are summed together and the passed on to the plant.

Depending on what gain is altered in the PID controller it will have an effect on the rise time, the overshoot, the settling time, and steady-state error of the step response for the plants model. The following table explains how the different parts of the of the PID controller effects these properties mentioned above.

Response	**Rise time**	**Overshoot**	**Settling time**	**Steady-state error**
Proportional part	Decrease	Increase	Small Change	Decrease
Integral part	Decrease	Increase	Increase	Eliminate
Derivative part	Small Change	Decrease	Decrease	no

Table 2.4: An overview of the effects from increasing the different PID parts [18].

Table 2.4 shows the general effect the different parts have on the signal, however it must be clearly stated that these part are not independent when they are put together, since changing one of them, may affect the other parts impact and in the end PID controller output. Thus, this table is only for consulting use, to aid the process of identifying the different gains [18].

The part that typically follows a controller is the plant, this is the model of the physical system (see Figure 2.16).

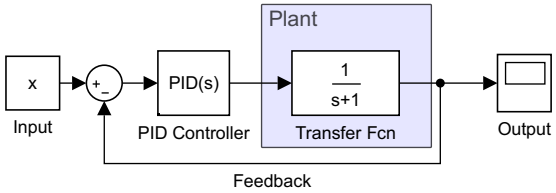

Figure 2.16: Shows the plant in a closed loop control system.

Figure 2.16 illustrates the where the plants is positioned in a control system, and it can seen that it is a transfer function model, however the type of model can vary depending what kind of overall model is needed. Commonly used is the transfer function model, which is estimated based on dataset of input, output and sample time, in other words how the system response is to a specific set of inputs. In the computer program MATLAB the transfer function can be found using the function called ident. This toolbox, can be used to estimate the best fitting transfer function, based on the wanted number of poles and zeroes. The output of this estimation is a transfer function, where all its properties can be inspected such as pole placements and fit percentage.

In general, a low fit percentage does not mean that the transfer model is non-usable, however the fit should be as close to 100% as possible.

Open loop control systems, are often referred to non-feedback system, since it does not compare the input with the actual output. In an open loop system some input is given to the PID controller, the controller will then based on the given input, produce an output to the plant which then will give the real output. The difference from a close loop system to an open loop system is the feedback loop. Figure 2.17 below shows how an ordinary closed loop systems looks like, where the system output is used as feedback, which is then compared with the input, this then yields the error between the input and the output of the system.

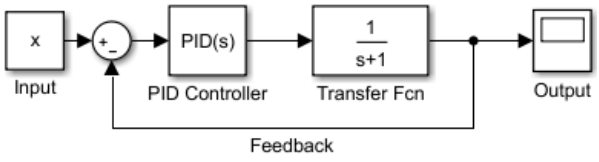

Figure 2.17: An overview of a Closed loop Control system

Depending on how the system is going to be implemented, what is going to be the input and what is going to be the output, is to be considered when building a supervisor for the control system. A supervisor is a function which takes one or more inputs, and based on these inputs it will take the programmed actions and relate the correct output. An example of what a supervisory control system could look like can be seen in Figure 2.18.

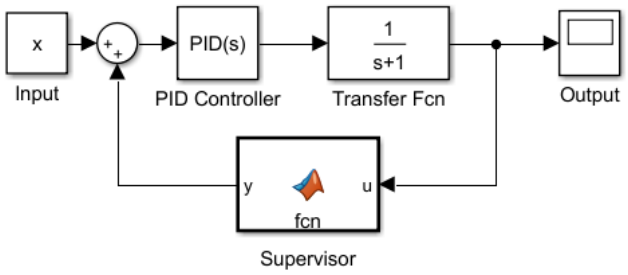

Figure 2.18: Shows a supervisory closed loop controller

In Figure 2.18 it can be seen that the supervisor is providing the feedback for the system, based on what the output of the system is, the role in this setup could be as simple as rounding or limiting the feedback. This setup is only one of many ways to configure a supervisory control system.

2.7 Conclusion of Problem Analysis

Through the problem analysis knowledge has been gained, and in this section a decision on which technologies and concepts will be chosen for use in solution and build of the prototype will be made.

In Section 2.2-Angle Monitoring an investigation into different kinds of technologies that could be used for the prototype was performed. Encoders, variable resistors and motors with position feedback was covered. A comparison table (see Table 2.2) was compiled, and it showed that not one of the proposed technologies directly fulfilled the requirements for angle monitoring. However, based on the data obtained from Table 2.2 it was chosen to use the bend sensor as it fulfills most of the requirements and should take the least amount of extra time to implement.

In Section 2.3-Controlling Unit the control unit for the project was analyzed, and the choice was between the Raspberry PI 3 model B, and the microcontroller of the type Arduino Uno Rev. 3. The decision is based on the comparison (see Section 2.3.3-Fulfillment of Requirements), which showed that the Arduino Uno would be the preferred control unit for the project.

In Section 2.4-The Knee, a review of how the knee is build and how the movements are was carried out. This section provided information about what to keep in mind when creating a prototype that revolves around a complex joint such as the knee.

One of the goals of the project is to create a knee brace prototype and, in Section 2.5-Knee Brace Materials, an analysis into the different kinds of materials that could be used for the knee brace was conducted. This ended up with a choice between metal and plastic, where the decision came down to being able to modify the design easily after it has been made in, if unforeseen problems would arrive. This is why plastic was chosen, since it is an option to 3D-print locally on campus.

Section 2.6-Motor Assisted Movement covered the topic of using a motor as assistance

to moving the leg, and it was discussed the control of the assistance using a control system. However, a specific control system type will be chosen in the solution as it will depend on what motor is chosen, and how it reacts in initial tests. Two requirements was set for a motor, first it must be able to move in both directions, and second it should be able to fit on the side of the knee hinge. A servo motor was chosen due to it fits these requirements, and that a useable one was present on university campus.

Chapter 3

Solution

This chapter will contain all the relevant information on hardware, software, testing and modeling, which has been used to make the prototype. The focus of this chapter will be to create and present the chosen setup, and how it can be used to solve the task at hand.

3.1 Knee Brace

One of the requirements from the 2.1.2-Project Delimitation is to have a working prototype, and in the 2.7-Conclusion of Problem Analysis it was chosen to pursue a knee brace made from plastic. This means it would be possible to use 3D-printers on campus to print the prototype, and any replacement parts or add-ons that would be required in order to create a working prototype. Hence, the first task would be to produce a 3D Cad model of a knee brace, so it could be printed while other testing were carried out.

This task seemed manageable within the timeframe of the project. However, after one week of working on the design, the only model was nowhere near done. At this point, it was clear that the man hours needed to create a model of a brace was greatly underestimated. This resulted in scraping the model and looking for a finished model, which could be slightly altered.

A model was found on "Thingiverse.com" which could be used with some alterations. This model called "Modular Knee Brace" created by John Smith [19], can be seen in Appendix A-Unedited Brace Model, came with a guide on how to take the correct measurements and where to input these measurements into the assembly file in order to custom fit the brace for the specific leg. This model was published under Creative Common Attribution 3.0 license [20], which means it can be used for personal purpose and alterations. This is under two terms, the author must be credited for use of the model and all alterations should be clearly indicated [20]. Furthermore, there should be a link to the license, and this link is going to be together with a depiction of the model in Appendix A-Unedited Brace Model.

Finding this model put the project back on track, since the deadlines was starting to slide due to trouble creating the brace. The guide was followed, on how to make brace fit a knee, so it could be wearable prototype.

3.1.1 Experimental Work: Knee Brace

In this section all the experimental work done that has to with the print of the initial prototype, and small mountings for the flex sensor will be presented. The total printing time for the brace with the appropriate settings is around 96 hours, and this is the best case scenario. However, the 3D printing process is far from perfect and this means it would take longer than this amount. Some alterations were made to the lower and upper crossbar, which meant the printing time ended up being around 72 hours total.

During the printing process it was noted that, even though anti warping support was enable, some of the parts would warp/contort at crucial point, which meant restarting the print. After this had happen several time, flexible tape was applied to the printing bed, which solved this problem. In the end the knee brace was assembled, and the result can be seen just below in Figure 3.1.

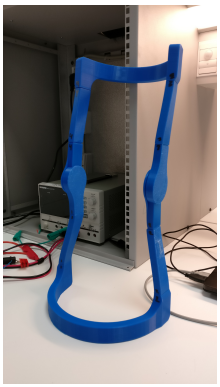

Figure 3.1: A picture of the printed Knee Brace

After the print of the knee brace prototype, the next thing was to attach some Velcro tape, so it could be tightly fitted the knee. This was done and then the knee brace could be mounted on the leg.

During the printing process, it was reviewed how the bend sensor chosen could be mounted, and Figure 3.2 shows that it is the ends of the bend sensor which should be moved.

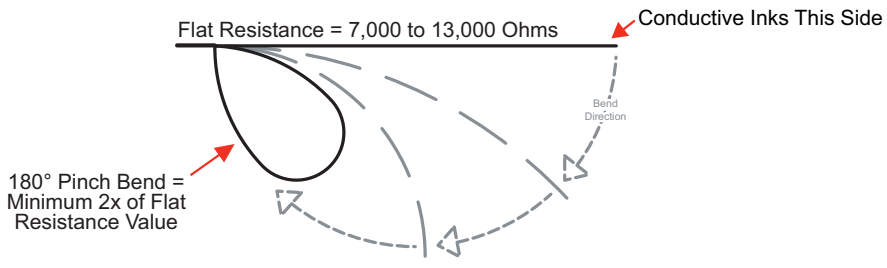

Figure 3.2: Shows how the bend sensor should function.

This, meant creating two holders for the bend sensor that would be glued onto the existing knee brace hinge which can be seen in Figure 3.3.

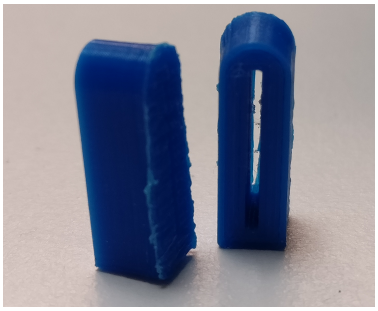

Figure 3.3: Is a picture of the Flex sensor mounts attached to a test knee hinge

3.2 Arduino Uno

In the Section 2.7-Conclusion of Problem Analysis it was decided to move forward with using the Arduino Uno Rev. 3 as the controlling unit of the prototype. The structure of the Uno is rather familiar, since it has been used for several projects before. Normal customs when writing a program for the Uno, is to follow the structure set in the blank template, having this structure on will help the understanding by others in the review of the code or trouble shooting, since this is how it is normally structured.

The Uno board itself has a power requirement of $5V$ to operate, which can be produced from the USB connection which is also used to transfer the code to the micro-controller. However, this might not be enough to power the whole board when powering the board from an external power source, therefore Arduino recommends to supply the board with at least $7V$ and a maximum of $12V$, since this should ensure normal operations [9]. Therefore the board will be powered with $7V$ while testing.

In order to be able to view the angle measurements, while the Uno board is powered externally, it was decided to use a LCD keypad Shield, on top of the Arduino. The specifics of the shield will be covered in the next section. The pins which are in use on the Arduino can be viewed in Appendix D-Arduino PinOut, however this table does not include the pins that are used for the lcd display.

3.2.1 LCD KeyPad Shield

This shield is from the maker DFrobot, and has a 2 line 16 slot LCD screen, in combinations with five usable buttons, which are connected to the $A0$ pin of the Arduino. This shield can be seen in Figure 3.4, and it is attached directly on top of the Arduino.

Figure 3.4: Graphical overview [21]

.

The Shield is going to be used for displaying the read outs from the flex sensor, and the buttons on the shield will be used for tuning the angle measurement in the start of the code. Example code, for the shield, provided by DFrobot, provided a blueprint for testing the buttons and also showed how to print text out on the screen. The LCD screen can be written to by using the *LiquidCrystral* library, which is made for the purpose of writing to LCD screens. The only configuration needed, is to define at which pins the LCD screen is connected, and the size of the screen.

3.2.2 Experimental Work: Arduino

The Arduino has been present at nearly all the experimental work that had to do with electronics, therefore there are no specific for it. However, as for the LCD keypad shield a function test was carried out of the screen and the buttons.

This test showed that the keypad and screen was working correctly. However, while testing it was discovered that there were no function in place for clearing a line on the display, which then was developed.

3.3 Flex Sensor

For monitoring the angle of the bend knee, it was in Section 2.7-Conclusion of Problem Analysis decided to use a bend sensor. "Spectra Symbol" are the producer of the chosen bend sensor (flex sensor for future references), which is a variable resistor that will alter resistance based on how bend it is, can be seen in Figure 3.5. The flex sensor ordered for the project has a length of $\approx 11.4\,cm$ and the active part of the sensor is $\approx 9.75\,cm$ long [22].

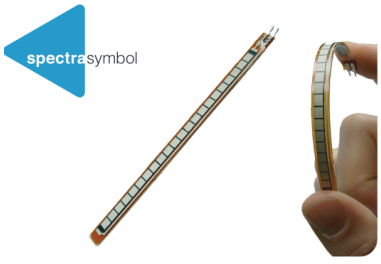

Figure 3.5: Illustration of the flex sensor

The flex sensor has a base resistance of $10\,k\Omega$, which it can deviate from with as much as ±30 %, and when pinched at 180 °, it will provide the double amount of resistance from what the baseline resistance is [22], and its operating temperature range is between -35 ° Celcius and 80 ° Celcius.

This sensor will be place at the side of the knee hinge, where the middle part of the flex sensor is going to be placed right at the pivot of the knee, see Figure 3.6.

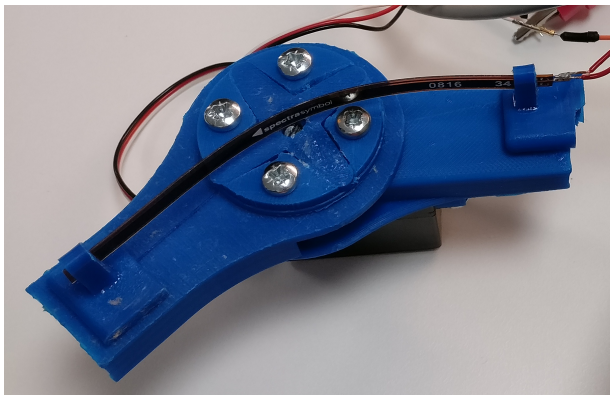

Figure 3.6: Is a picture of the flex sensor mounted on a test knee hinge

.

This position should give correct readouts of the angle of the knee, however the tolerance of ±30 % should be kept in mind.

Flex Sensor circuit

It was decided to create a circuit which could be used in connection with the flex sensor, so a proper voltage read out could be made. This circuit diagram can be seen in figure 3.7.

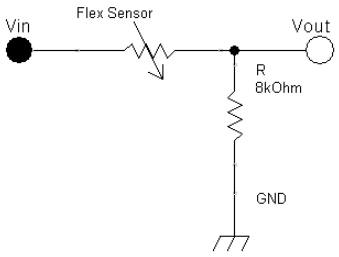

Figure 3.7: Drawing of the flex sensor circuit

The circuit output seen in Figure 3.7, is the output that goes to the Arduino Unos $A1$ pin. This $A1$ pin leads to the 10bit ADC, which can take up to $5\,V$ in and output a corresponding ADC-value. Since the ADC is 10bit it can detect the difference of ≈ 0.0049mV, this is due to 10bit = 1024 decimal values, and this correspondence between ADC-value and voltage can be seen in Table 3.1 just below.

Voltage	ADC value
$0V$	0
$0 < V < 5$	$1 - 1022$
$5V$	1023

Table 3.1: ADC step comparison

3.3.1 Software

The software in this section was written with the purpose of calculating the angle, based on the V_{out} seen in Figure 3.7, and then saving this value in a variable. This was done with 2 getter functions and 3 other functions. The getter functions will be covered through the walk through of the 2 main functions.

void startCalibration() This function is executed when the micro-controller is powered, and it has been created in order to calculate a correct angle for the project. The activity diagram of the function can be seen in Figure 3.8.

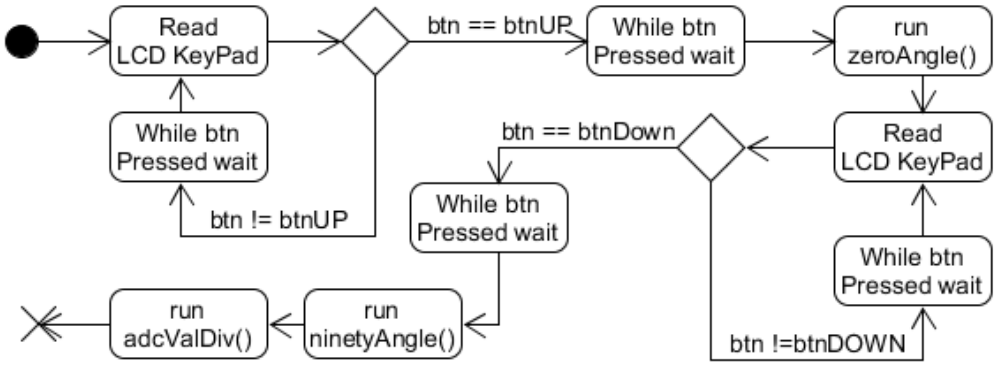

Figure 3.8: Illustrates the activity diagram of startCalibration().

The following startup sequence has been defined, when the leg is completely stretched press up, and then when the leg is bend at 90 °press down. This is the way it has been decided to do the calibrations, and it can only be done in this manor.

Figure 3.8 shows the following process; wait for btnUP to be pressed, wait for it to be let go, run "zeroAngle()" which takes an ADC measurement and saves it into the variable zeroDegADCval, wait for btnDOWN to be pressed and let go, run "ninetyAngle()" which again takes an ADC measurement and saves it into the variable ninetyDegADCval, after this it runs "adcValDiv()" and then terminates. The last function "adcValDiv()" calculates the following equation and saves the result into a variable, which is then used in the "currentAnlgeCalc()".

$$adcValprDeg = \frac{ninetyDegADCval - zeroDegADCval}{75}$$

The reason why 75 is a hard-coded constant in this equation, is due to the real movement span of the knee hinge is between 15 °and 90 °, since the hinge does not stretch fully.

The only thing that hasn't been taken care off in ''void startCalibration()" is if the "zeroAngle()" gets a misread which might happen and becomes bigger than "ninetyAngle()" ADC value, and this would ruin the calibration. However, this problem was solved with the following bit of code in the "void setup()" function.

```
1 while(zeroDegADCval > ninetyDegADCval || zeroDegADCval <= 50 ||
    ninetyDegADCval<= 50){
2     clear_LCD_line(0);
3     clear_LCD_line(1);
4     lcd.setCursor(0,0);
5     lcd.print("redo cali.");
6     startCalibration();
```

This code goes in and compares the values with first each other, to see if the zeroDegADCVal is bigger than the ninetyDegADCval. Furthermore, it looks for if one of the values is less than or equal to 50, if no voltage is on the analog port it will give a very little reading, this situation can be caused by a loose connection or not enough power. If any of this is true it will clear the screen of the LCD display, print redo cali., and then start the calibration process allover again.

void currentAngleCalc() This function is used for getting the current angle of the bend flex sensor, and what it essentially does is take an ADC measurement, do a comparison and then based on the comparison perform the angle calculation or set the angle to 3. The activity diagram of the function can be seen in the following Figure 3.9.

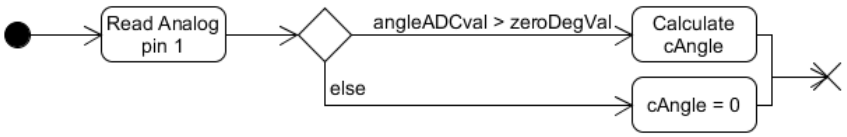

Figure 3.9: Shows the activity diagram of currentAngleCalc().

The calculation that happens in the function can be seen in the following equation

$$cAngle = \frac{angleADCval - zeroDegADCval}{adcValprDeg}$$

This is the current measurement (angleADCval) which to get the delta value from the lowest angle gets the lower bound(zeroDegADCval) subtracted, and then this divided with the conversion variable (adcValprDeg), and then the calculated angle is save in the variable cAngle.

3.3.2 Experimental Work: Flex sensor

Two test were preformed to solely test the flex sensor, it was the initial function test of the sensor and the proof of concept test of the sensor. The flex sensor was also used to be able to make the transfer function for the servo motor, however that will be covered in the experimental work Section 3.4-Servo Motor. Finally, the flex sensor was used in the final system test but that will be covered in Section 3.6-Unifying the Prototype.

Function Test

When the flex sensor was received, the functionality was tested to confirm compliance with the specifications in the datasheet. This meant setting up a multimeter and building the previously mentioned circuit in a breadboard, see Figure 3.10 to see test setup.

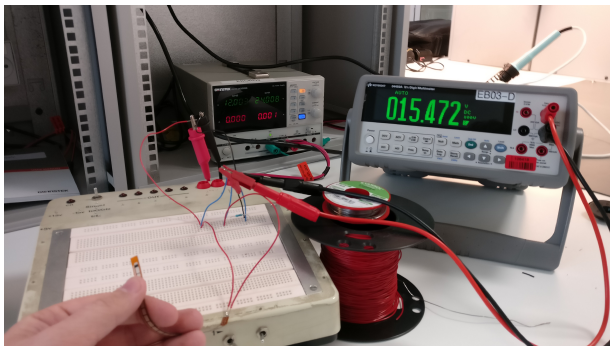

Figure 3.10: Shows the test setup used for the function test

When the test was set up it, voltage was applied and the flex sensor was bend to see if the output voltage would change. The functionality of the flex sensor was confirmed based upon this conducted test.

Proof-of-Concept Test

In order to do this test properly, the flex sensor was mounted onto the knee hinge, which then would be bend at different angles to see if the theory would work with it mounted on a knee hinge. Furthermore, in order to do the proof-of-concept test, it would be required to get an angle read out from the setup, so the software, see Section 3.3.1, was implemented in the microcontroller. The setup of the proof-of-concept test can be seen in the Figure 3.11.

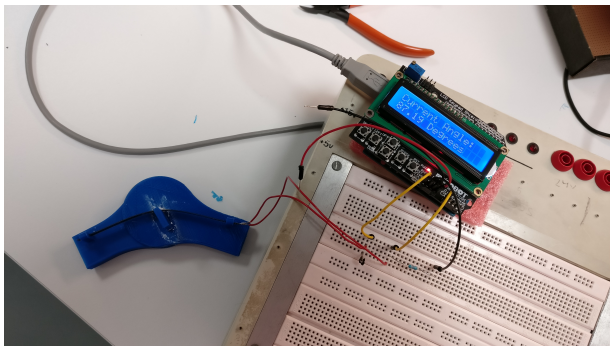

Figure 3.11: Presents an overview of test setup.

This test was carried out successfully, as the obtained angles based on the resistance from the flex sensor was correctly corresponding to the angle of the knee hinge. In Figure 3.11 the angle on the LCD screen does not correspond with the angle of the hinge, however this error should be ignored, since this was a bug in the code , this error has since been fixed so it displays the correct angle.

3.4 Servo Motor

In Section 2.1.2-Project Delimitation it was stated that the second focus of the project would be to implement some form of motor assisted movement. Which in Section 2.7-Conclusion of Problem Analysis was decided would be a servo motor.

Initially a servo motor which was present on campus was tested, however as this servo could only could move in one direction, it was not an option for the project. Therefore, another servo was chosen for the project, a Parallax Standard Servo which has a limited range of movement from $0\,°$to $180\,°$, and the servo operates on $4\,V - 6\,V$ of power [23]. This power range made it possible to operate it from the Arduinos $5\,V$ power pin, with an alternative power source being an external power supply.

3.4.1 Software

The software for the servo is actually fairly simple it is only commands that are allready present via the *Servo* library that is used, combined to create test software. In this library the following functions servo.attach(), servo.write(). What the attach function does is actually just to designate which pin the servo signal pin is on the Arduino. The ”servo.write()” takes in the argument which is the angle set point, and when it is called it will give the fitting PWM signal to the motor, which will then move the servo horn to that setpoint.

This PWM signal generated by the *Servo* library, is a form of software generated PWM. This is used since the uno can not produce a clean 50Hz PWM signal as hardware generated PWM, which means the software generated PWM is used even though it might vary a bit depending on the software running on the Uno.

3.4.2 Experimental Work: Servo

The servo motor is probably the component that has been through the most tests, in order to show the compatibility of it for the prototype and the project. This resulted in a function test, a proof-of-concept test, a "freewheeling" test, and a test that would help with identifying the system behavior. The last mentioned test was carried out so that modeling of the motor was possible. The final system test will be covered in Section 3.6-Unifying the Prototype.

Function Test

When the servo arrived the first thing to do, were to see if the servo would act as described in the datasheet. This was done by attaching it to the Arduino directly and using the *Servo* library, this library took care of setting the PWM, which meant the input to the motor would be an angle set point and the library would convert this into a PWM signal, which then would be passed through to the signal pin of the motor.

The servo was tested with big steps of $20° - 45°$ for a long amount of time, and after this it was tested to move $1°$ at the time for $5\,min$. These tests showed that it should be

possible to get the motor to at least follow the measurement from the flex sensor.

Creation of the Servo Hinge

Before being able to do the proof-of-concept test with the servo, it needed to be attached to the knee hinge in some way. This meant altering the design for one of the knee hinges so it would look like Figure B.1 in Appendix B-Modified Knee Brace Model. This process started out with just making slight alterations to the knee hinge models, making a centered hole of the size of the servo gear. It also meant creating a motor mount for the, from the dimension data in the datasheet of the servo, the 3D model of the motor mount and technical drawings can be seen in Appendix C-Modified Servo Hinge Parts in Section C-Motor Mount.

After the mounting for the servo motor had been created, it was clear that alterations to the knee hinge were required. It was necessary to extract the servo gear through the hinge and then fasten it to the other side of the hinge. The model of this extraction and the mounting for the servo can be seen in Appendix C-Modified Servo Hinge Parts in Section C-Servo Axle Extension and Section C-Servo Axle Extenstion Holder. Due this alteration, the printed hinge needed a centered hole of 9mm, so it could fit the servo axle extension, which was drilled. The parts put together can be seen in Figure 3.12.

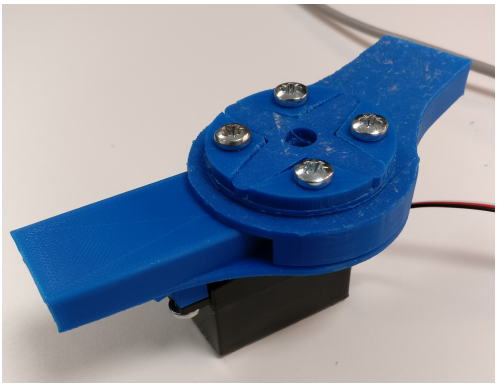

Figure 3.12: Shows the servo hinge assembly from the side.

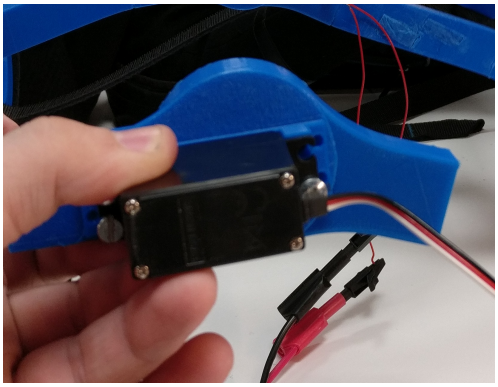

Figure 3.13: Shows the servo hinge assembly from the top of the servo.

Proof-of-Concept Test

Now with the servo motor attached to the knee hinge, it would be possible to carry on with the proof-of-concept test. This test shall show that the servo mounted on the knee hinge should be able to follow an angle setpoint from the flex sensor, however instead of the flex sensor being attached for this test it would follow coded path.

The test outcome showed that the motor still could follow a set point while being attached to the knee hinge. This is proof-of-concept of that it will work if it was attached to the brace, this is due to it would only be a question of scaling the servo motor so it had enough torque to move the leg, and since it can follow an angle input, should be able to lead based on angle measurements also.

Freewheeling Test

In order to make the motor run loose after it has hit a certain angle, meaning it would not keep the brace at a specific angle and break down trying to work against the legs motion. The detach function in the previously mentioned servo library was called and correctly it would detach, however the detach function also caused not to execute in order.

Due to the behavior when the servo is detached, it was decided after some trouble shooting not to use the function due know instabilities. However, this also meant that it would not be possible to have the motor attached to the prototype in the final system test, but rather on a separate hinge.

3.5 System Identification and Modeling

It has been decided to try and implement a supervisory reference point control system. This decision is based on the wish of wanting to implement motor assisted movement, as discussed in 2.6-Motor Assisted Movement. The assisted movement needs to guide the leg hinges in the direction, that the leg is moving, and this should happen automatically. This is the reason a supervisor, which based on the programmed logic, sets the next reference point for the motor, which will enable guidance of the leg.

To create the supervisor it was decided to use Simulink, a part of the MATLAB software suite, to simulate this control system. This simulation of the servo system, should end up saving time when coding the supervisor, since every time an alterations is made to it, a simulation can be made instead of needing the physical setup to test a small alteration.

Therefore, a model of the motor needs to be established, this model will have to be based on experimental data from the servo motor and in combination with the flex sensor as verification on if the set point is reached.

3.5.1 Supervisor Description

The supervisor will be a central part of the control system for the servo. Therefore, before starting the modeling part and creation of the supervisor in the simulation software, it is a good idea have requirements for the supervisor, which then also would ease the implementation into the simulation software.

The way the supervisor is going to be implemented is that it should act based on the following set of points:

- If the difference in angle measured is less than ±5 °ignore it and set latest setpoint again.
- Sudden chance greater than ±25 °, set the setpoint to the current position.
- The supervisor should make the servo guide the leg/brace.

These was the initial requirements for the supervisor when the project first started. From testing it was visually noticed that the servo test hinge in itself does not have problems with reaching a setpoint, with the little stress the motor is under.

It should be stated that the first requirement, is to judge if the leg is moving or in a steady state. Thus, if the difference is less than ±5 °, it will be seen as the controller as not moving. However, once it has started moving the supervisor will have to ignore this, until a new steady state has been reached.

3.5.2 System Identification Test

This test was conducted solely with the purpose of modeling the motor. The setup for the test can be seen in the Figure 3.14. As well as the flex sensor which has been mounted on the servo hinge, and both the servo and flex sensor has been connected the Uno. The Uno is powered from the USB port, where the servo is powered from an external power supply.

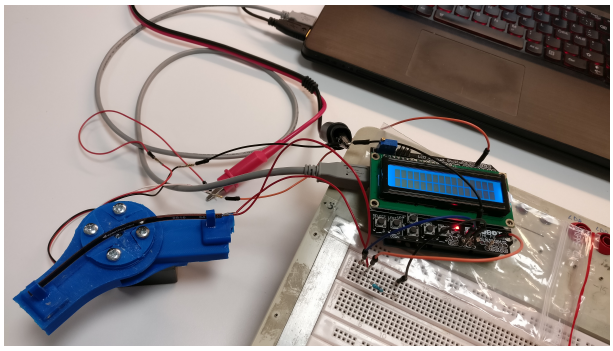

Figure 3.14: Is an image of the used setup

The code that ran in the main part of the program can be seen in Appendix E-System Identification Test Code. The test is fairly simple, but in order to perform it some initial testing was required. To ensure sampling at fast enough frequency, a test was conducted to show time it would take for the motor to move the largest step possible for the test. This was done by altering a delay function in the code until was not able to hit the desired angle before receiving a new input. It was determined through testing to be 450ms for it to travel from a position of 15 °to a position of 90 °.

Following Nyquist sampling theorem, which is $f_{sample} > 2 \cdot f_N$, and this is atleast twice [24]. This means that the sample rate should at least be 225ms, however this is the maximum step and with smaller steps it was decided to sample at 30 times the maximum time, which means a sample time of 15ms. To ensure every sample was taken after a 15ms the following comparison to take calculation time into consideration.

```
1    currentTime = millis();
2    while (currentTime - timea - calcTime < sampleTime){
3       currentTime = millis();
4    }
```

In the whileloop a comparsion is made 3 variables (currentTime, timea, calcTime), and the sampleTime of 15ms. the three variables are all unsigned long ints, and they are

populated using the millis(), which gives a specific amount of *ms* from a specific point in time. The result from subtract these variables with each other is then compared to the sample time and if it is less than the sampleTime constant (15 *ms*) then it will update the currentTime variable until it reaches a point where this statement is not true, which would mean that 15 *ms* has passed and it is time for another sample of the flex sensor.

The data that was gathered from the test was the set point for the servo motor and the read out from the flex sensor. Since the knee hinge when fully stretched is bend at an angle of 12 °, this has been corrected by adding 12 to the data gathered.

3.5.3 Creation of Servo Model

The first to this process is to get the collected data into the MATLAB workspace.

With the data in the current workspace, the MATLAB command ”ident” or ”systemIdentification” is called. This open the system identification toolbox (see Figure 3.15), which will be assisting in the process of generating a model based on estimations.

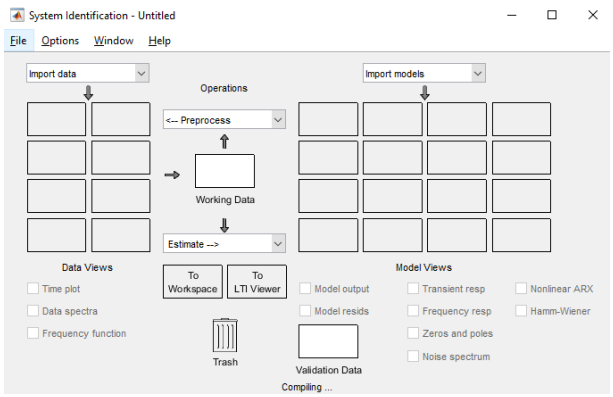

Figure 3.15: Shows the look of the System Identification toolbox.

When the toolbox seen in Figure 3.15 is present on the screen, the next thing was to import the data, this is done by clicking in the import data tap in the upper left corner of the window, and select the type of data to be imported.

The data for the modeling of the motor is time domain data, so this tab was chosen. This action prompts a window to pop-up this is the import data window, which can be seen properly configured in Figure 3.16. In this window is where the condition for the data is configured before the data is imported. As input for the model of the servo motor, is the set point angle for it, and the output is based on a reading form the flex sensor since the servo chosen did not have any position feedback signal.

The last thing to configure in this window is the start time (0) and the Sample time (15 *ms*) it is important to remember both of these time as the standard unit of seconds. The window configured for importation of the collected data can be seen in Figure 3.16 just below.

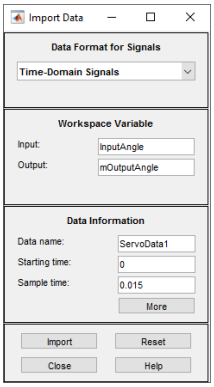

Figure 3.16: Shows the Import data window configured to import he collected data properly.

When the import button in the lower left corner of the import data window has been pressed, the data should then appear in the system identification tool box. At this point the next thing to do was to confirm that the data was imported correctly, this was done by selecting the data in the tool box and then clicking time plot, which is under the data views options. This prompted a pop-up which would consist of 2 graphs, the output over time and the input over time. This would show the step response of the servo compared with the input, see Figure 3.17.

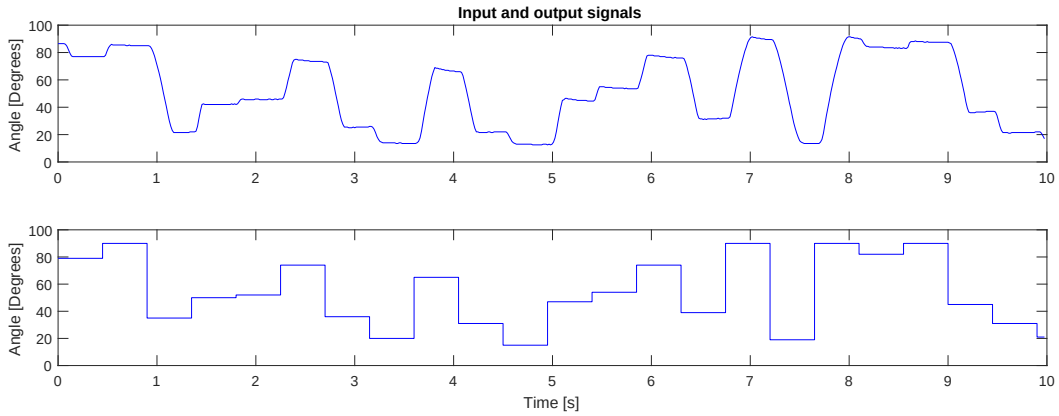

Figure 3.17: Shows the step response and the step.

In Figure 3.17 the top graph is the measured output with the flex sensor, and the bottom graph is the given input. It can also be seen that the measured angle sometimes is not the inputted value, this is due to the fluctuations of the flex sensor. However, it can also be seen that it keeps stable after stabilizing after a step, which is very important, since an high amount of fluctuations at this point would mean, the flex sensor would be completely useless.

When building a control system some kind of plant is needed to emulate the system (servo), in this case a system model. There are different models used worldwide, and commonly used are transfer function model. Furthermore, other than tf models for the servo a process model was also chosen to see what would be the best fit for simulating the servo. The following 4 models was chosen to evaluate with:

- tf1: a transfer function model with 1 pole and 0 zeroes

- tf2: a transfer function model with 2 poles and 0 zeroes
- tf3: a transfer function model with 2 poles and 1 zeroes
- P1: a process model with 1 pole and 0 zeroes

These four models has been chosen based on they should provide a simple function with a usable fit. The model estimations can be seen just below.

$$tf1 = \frac{8.084}{s + 8.414}$$

$$tf2 = \frac{101.8}{s^2 + 11.2\,s + 107.3}$$

$$tf3 = \frac{6.098\,s + 22.83}{s^2 + 6.478\,s + 24.15}$$

$$P1 = \frac{0.96075}{1 + 0.11915 \cdot s}$$

They will be compared based on fit to the original data, step response and pole-zeromap. The functions has been estimated using the "systemidenticfication toolbox", and can be seen in Figure 3.18, together with the measured output.

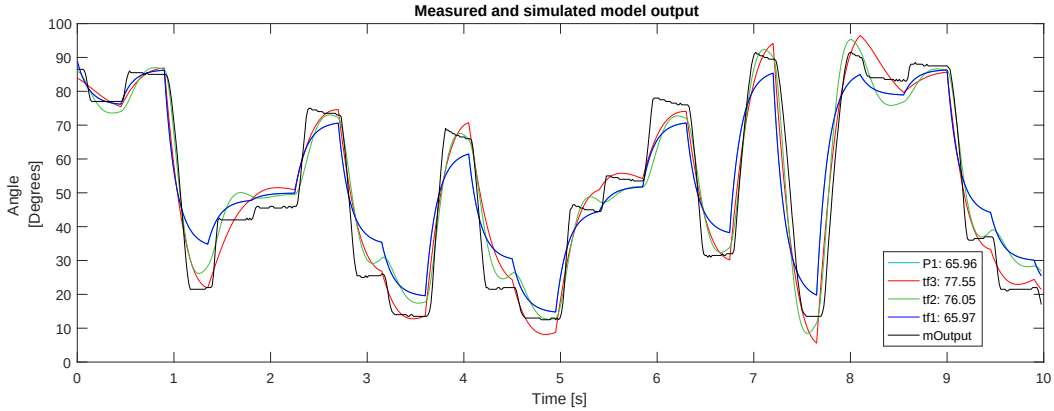

Figure 3.18: Shows a comparison of the different fits.

In Figure 3.18 is a graphical over the fits, where the different models fits can be followed, in correspondence with the black line which is the measured output of the system, the fits percentage vise can be seen. In Figure 3.18, it might be hard to spot P1, this is due to the fit is almost identically to tf1's fit. It can also be noted, in this graph, that all the models does not follow the measured output very well. This behavior is explained by the poor fit percentages of the models, where the most precise is tf3 with 77.55%.

When designing a model of a system it is not only the fit percentage that is important, but also the step response of the model, since if a fit might have huge over shoot where the system requirements does not allow overshoots. In Figure 3.19 the step responses of the different models can be seen, this response is to a step from 0 to 1.

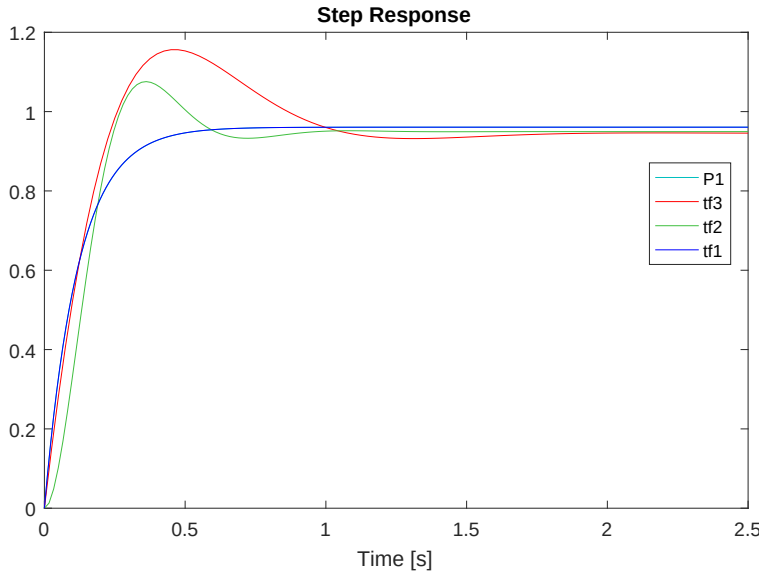

Figure 3.19: Shows a comparison of the different step responses.

From Figure 3.19 it can be seen that the step response response of tf1 and P1 are practically the same with no oscillation in the process of converging at a value. Tf2 can be seen having oscillations before it converges at a value, and finally tf3 is seen to be the quickest to hit the value, however it can also be seen to have the biggest overshoot, a small amount of oscillations and it takes the longest time to converge.

Figure 3.20 is a pole-zero map of the models poles and zeroes, where the placement of the elements can be used to describe the behavior of the step response.

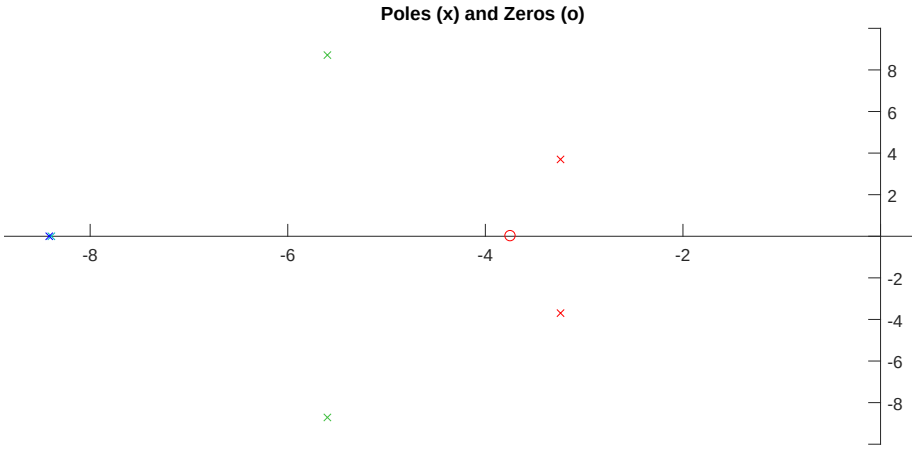

Figure 3.20: Shows a comparison of the different place of the poles and zeros from the tranferfunctions.

In Figure 3.20 it can be seen all the poles and zeros are located in the left half plane, this indicates that all the functions are stable. It can also be seen that tf2 and tf3 has a complex conjugated pole pair which is located of the the real axis, thus because they are not residing on the real axis oscillations will happen before convergence, which also can be confirmed by look at the respective step responses in Figure 3.19.

The placement of the poles on a PZ-map determines how fast it will converge at a specific value; far to the left of origo will cause it to converge fast, closer to origo while still being in the left half plane will cause the convergence to be slower, if the pole/zero are located at origo or on the imaginary axis it will never converge nor diverge it will have a stable oscillation based on the initial conditions, and lastly the further away from the origo into the right half plane the more the oscillations will spiral out of control. Based on this it is also why a system with poles/zeroes in the right half plane is classified as an unstable system.

The model that is going to be used for the system simulation is tf1. This decision is made on having a model that does not contain any overshoot, since having a overshoot in a control system with an injured knee might cause harm due to over extension. Thus the better fitting models tf2 and tf3 was not an option. Furthermore, tf1 has a slightly better fit than P1, and this is why P1 was ruled out. For the remainder of this report tf1 will be referred as the following

$$TF_{servo} = \frac{8.084}{s + 8.414}$$

3.5.4 Validation of Model

In this section a validation of the servo model will take place. The validation of the servo model, is going to be based upon the response of the transfer function when it is given a specific set of input data, which also is compared with the output of the real servo response to the same specific set of input data. The response of the system can be seen in Figure 3.21.

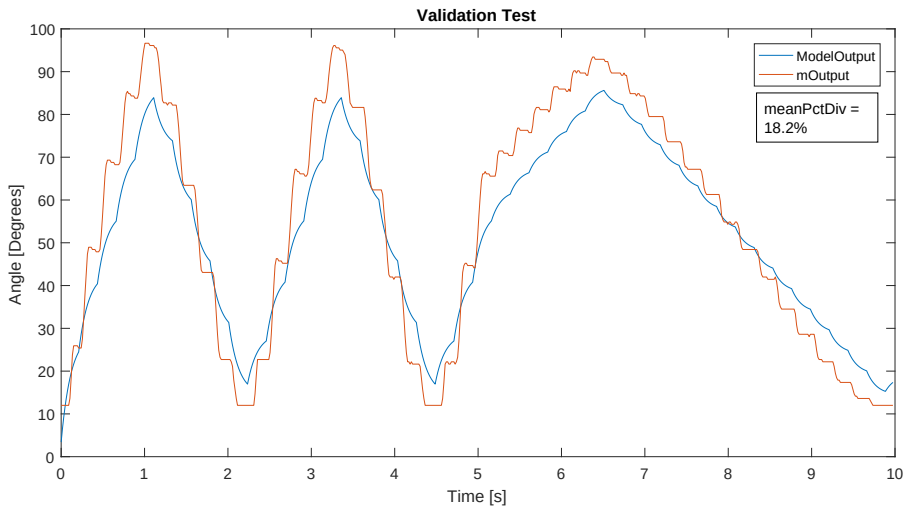

Figure 3.21: Shows the real response and model response to an input set.

From Figure 3.21 it can be seen that there are some difference in how the two systems acts based on the same input and the mean pct deviation between the measured output and the model output. This number is calculated, in MATLAB, by using the following set of equations

$$deltaOutput = abs(modelOutput - mOutput);$$
$$percentageDiffference = deltaOutput./mOutput;$$
$$meanPctDiv = mean(percentageDiffrence) * 100;$$

The following calculations results in a deviation of ≈ 18.2% in how much the modelOutput average deviate from the measuredOutput.

Recalling that the transfer function model has a fit of 65.97% to the original data on which the model was created. This fit is not that great and deviation from the real system behavior should be expected. Furthermore, it should also be kept in mind that the measured output is based on readings from the flex sensor, which has a deviation of ±30%, thus they are also deviating from the correct system output. However, this data will have to do, but if a more precise sensor were to be used for data gathering the deviation both in model fit and validation test should decrease.

The system identification test was done multiple times, and the best data set was used. This meant that the transfer function model TF_{servo} had the best fit, and even-though the model has this deviation, it will still be used to assist in the creation/modifying the supervisor.

3.5.5 Supervisor Creation

In order to create a supervisor based on simulation, a Simulink model was created to be used for this purpose. The transfer function that will be used to resemble the system is the following.

$$TF_{servo} = \frac{8.084}{s + 8.414}$$

These values are plugged into the transfer function block in the Simulink model that can be seen in Figure 3.22.

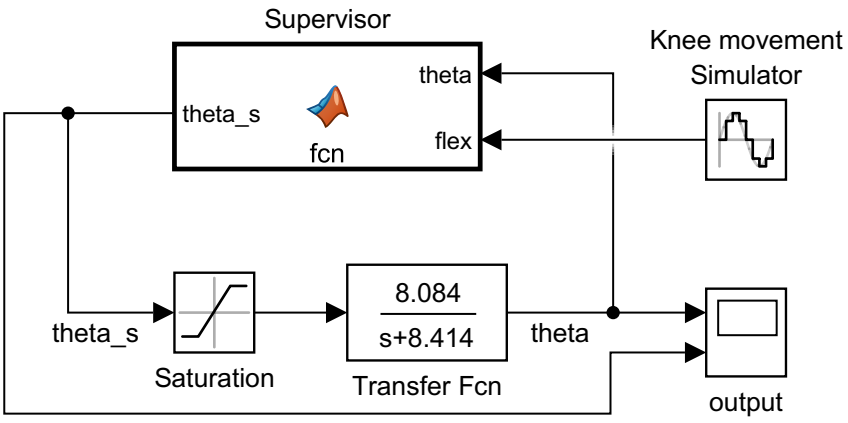

Figure 3.22: Shows the simulation model made to tune/configure the Supervisor.

From looking at Figure 3.22 it can be seen that the supervisor receives feedback, from the transfer functions output, while also receiving an input from the flex sensor. This

input from the flex sensor is generated with a sine wave generator which has 90 °peak-to-peak and has a offset of 45, which means it alters between $0 - 90\,°$. The reason this has been chosen to simulate the flex sensor reads, is that in principle a knee should move like a sine wave does, and the brace only function in this workarea. It can also be seen in Figure 3.22, that the supervisor sets output theta_s and this is the summed together with the input from the flex sensor. This signal is then saturated with an upper boundary of 90 and lower of 15 since this is the area the knee brace is operating in.

In this control model there is a classical component missing, this is the actual controller part. However, there is an explanation why this, possible PID-controller, has been left out of the model, this is due to during testing it was noticed that it was not necessary to implement a controller, since test results showed that the servo must have an internal controller. This means that the transfer function listed in the model actually also covers that controller. A model of this scenario has been created and it can be seen in Figure 3.23 below.

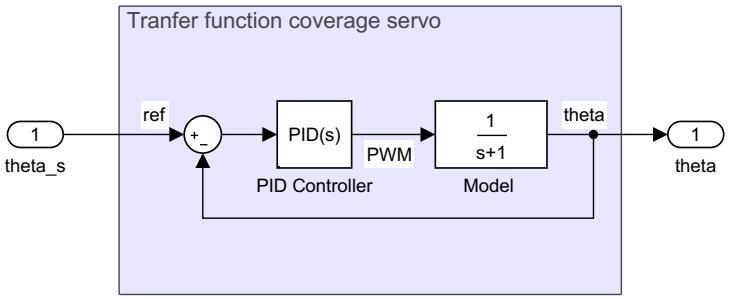

Figure 3.23: Shows the expected internal servo model system.

Figure 3.23 shows what the servo transfer function is covering, during the process of generating this transfer function, the servo was treated as a black box, due to its unknown the exact internal logistics this model has been generated on what is observed from the test of the servo.

Tuning the Supervisor

After the model of the supervisor was created it, simulation could then begin and the tuning of the supervisor could happen.

In the process of coding the supervisor, it was discovered that the code for the system needed some extra inputs based on the written code, in Figure 3.24 a revised simulation model can be seen.

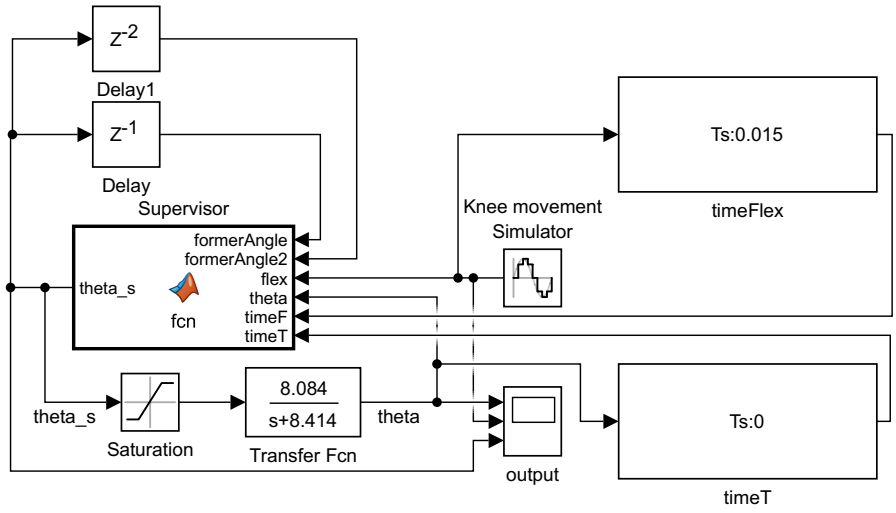

Figure 3.24: Shows the revised simulation model.

In Figure 3.24, it can be seen the supervisor takes two former angle setpoints as inputs, and to sampling times. Furthermore, theta can be seen as an input, however in the final supervisor code it was not used for anything.

The tuning proceed after these modifications to the model, and this in the end yielded the behavioral result which can be seen in Figure 3.25.

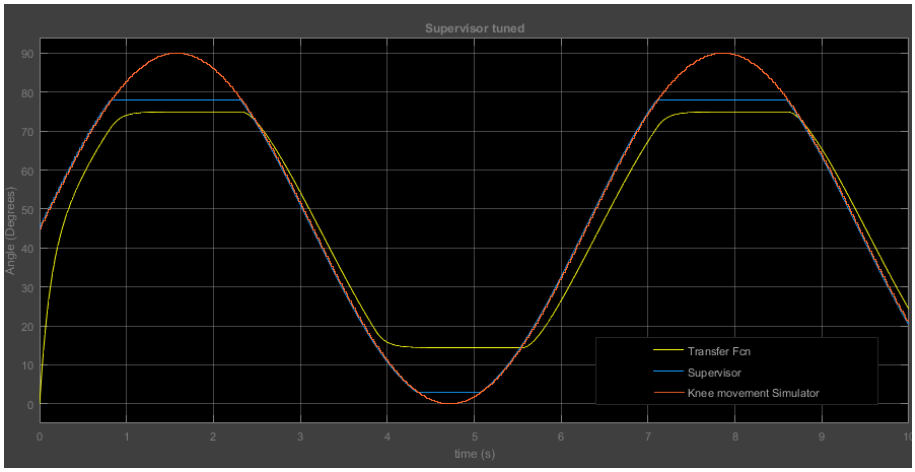

Figure 3.25: Shows the revised simulation model.

Figure 3.25 it can be seen that the supervisor follows the movement of the knee movement simulator within certain bounds, which has been set based on the prototype. It can also be seen that transfer function model is not behind the setpoint, while also not hitting the set values.

It has to be stated that the supervisor was not tested, based on a sudden angle change of ±25°, however if this sudden change happens withing 3 measurements it will set the reference point to that value. This is done inorder to not try and force movement, because this situation might indicate something bad happen to the knee.

The full code of the supervisor in its matlab implementation can be seen in Appendix F-MATLAB Supervisor Code. This code was converted nearly problemless directly into C-code, with only one small alteration needed. The tuning of the supervisor in a simulation model, ended up saving a lot of time.

3.6 Unifying the Prototype

This is the section where the full system test of the prototype will be covered. The servo motor will be left as a proof-of-concept, and will not be attached to the prototype. Since, it can not be controlled with HW PWM and the motor can not be freewheel with the software implemented PWM(see Section 3.4.1-Software and Section 3.4.2-Freewheeling Test). Furthermore, the servo motor will not have the strength to move the brace with a leg in.

The full system test will have the prototype mounted on a leg, with the flex sensor on its side. The servo motor will be left on the servo hinge, and will be fixated in a way so it will resemble a knee. Finally, the arduino with LCD keypad shield will be mounted by the side of the servo hinge, since this will make it easier to compare the current angle which will be outputted to the display.

The unification also means combining all the code to one functioning program for the full system test, the method diagram os this program can be found in Appendix G-Method Diagram, and it contains a declaration of all constants and functions used in the FST program. Furthermore, the code can be found on the CD attached in Appendix H-CD, or the uploaded Appendix H zip file.

In order to give a clear overview of the flow in the test program, an activity diagram was created and can be seen in Figure 3.26.

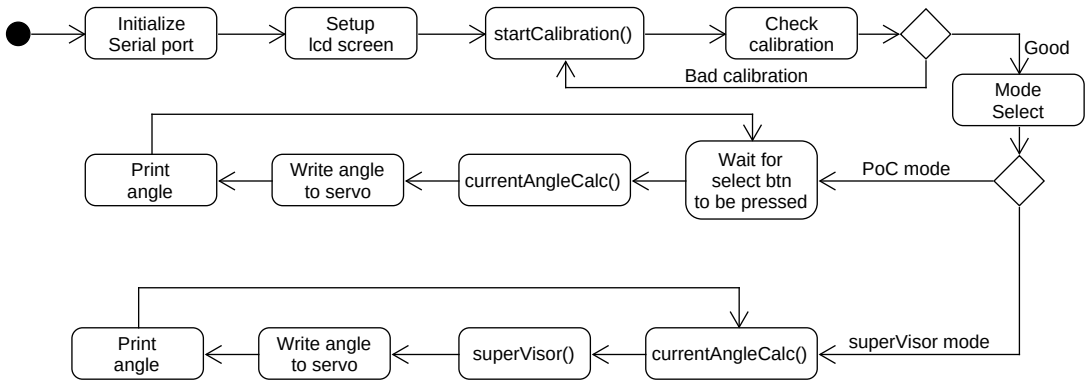

Figure 3.26: Shows a activity diagram of the test program.

In Figure 3.26, it can be seen that when the program starts it initializes the serial port first, setups the LCD screen and then runs the *startCalibration*(). If the calibration is deemed good it will contentious to mode select, and pending this selection one of the two mode is set to execute. These are the two modes, which later in this section will be tested.

If PoC mode is chosen, the program will wait until the select button is pressed, then it will calculate the current angle, write this angle directly to the servo, print the angle to display, and return to wait until the select button is pressed again. This is nearly the same with the superVisor mode, however this mode executes continuously. The current angle is calculated, then passed to the superVisor(), which takes care of converting the input to the next angle that will be written to the servo, and printed on the LCD screen.
In Figure 3.26, it can be seen that when the program starts it initializes the serial port first, setups the LCD screen and then runs the *startCalibration*(). If the calibration is deemed good it will contentious to mode select, and pending this selection one of the two mode is set to execute. These are the two modes, which later in this section will be tested. If PoC mode is chosen, the program will wait until the select button is pressed, then it will calculate the current angle, write this angle directly to the servo, print the angle to display, and return to wait until the select button is pressed again. This is nearly the same with the superVisor mode, however this mode executes continuously. The current angle is calculated, then passed to the superVisor(), which takes care of converting the input to the next angle that will be written to the servo, and printed on the LCD screen.

3.6.1 Final System Test

The final system test will be two parted, one showing the proof-of-concept for the prototype, where upon request a reading from the flex sensor will be taken, and then the calculated angle will be written to the servo, making it move to that specific angle and displaying the angle on the LCD screen.

The second part will be a demonstration of the supervisor implemented. The supervisor is based on the requirements in Section 3.5.1-Supervisor Description, and will act upon the measurements from the flex sensor. This test should demonstrate the motor will try and lead the leg, based on an angular velocity calculation.

Based on the success of the final system test it will be possible to evaluate the prototype.

FST: PoC Mode

The first part test was a success, it was possible by the press of a button, to get the motor to move to the angle at that corresponds to the prototypes current angle.

This shows the proof-of-concept of the prototype, without a supervisor implemented.

FST: superVisor Mode

This test showed that the implementation of a supervisory reference point control system was successful, with modifications.

The servo motor showed that it was possible to ”lead” the prototype, which should mean it could assist the movement of the knee/leg. During this test, the servo would fluctuate a bit when leading the leg, however this most likely due to the flex sensor imprecision. Therefore, it was decided not to troubleshoot this.

The test showed proof-of-concept of implementing a supervisory reference point control system. However, it should only be a matter of tuning and scaling for it to be implementable on the prototype.

A video of this part of the final system test, can be viewed by following the link in reference [25].

3.7 Conclussion of Solution

Based on the results of Section 3.6-Unifying the Prototype, which showed a working prototype of a knee brace, where the flex sensor despite its imperfections was able to be used as sensor for angle monitoring. Unfortunately due to a couple of reasons, the working prototype did not have the servo motor implemented, however via the servo hinge a successfully demonstration of proof-of-concept was made. Furthermore, a supervisory reference point control system was successfully implemented in the second part of the final system test.

Chapter 4

Discussion

Throughout the preparation of the report and the development of the prototype, there has been multiple things that have effected the project, these impacts will be discussed in this chapter. Furthermore, the project will be evaluated against the goals set in Section 2.1.2-Project Delimitation. Table 4.1 depicts the evaluation for each of the requirement and the results will be discussed on the following pages.

To start off by will be an table containing the evaluation of each requirements and then a discussion of these results.

No.	Requirements	Accomplished
1.	Angle Monitoring:	≈
a.	Implementable	✓
b.	High precision measurements	✘
c.	Does not hinder movements	✓
2.	Control Unit:	✓
a.	Handle analog input	✓
b.	Generate PWM signal	✓
c.	Display data	✓
d.	User friendly	✓
3.	Prototype:	≈
a.	A fabricated knee brace	✓
b.	Implemented angular sensor	✓
c.	Implemented motor assistance	✘
d.	Wearable	✓
e.	Low cost	✓

Table 4.1: Requirements Evaluation

Requirement 1: Detect angles, can be marked overall as a success since the flex sensor is able to ensure angle detection. Requirement 1.b is not fulfilled, since the flex sensor in itself is not precise enough. The flex sensor was chosen based on it would fulfill the requirements for a sensor used for angle monitoring. While it was known that the tolerance was ±30 %, it was not know that this was constantly fluctuating, and not a linear deviation from the base value. This deviation might have scewed the results for the modeling of the servo. However, it fit on the knee in compliantly brace and did not

cause any trouble, which is good, else than being precise. This also means if a more precise flex sensro can be acquired it would be a good choice. However, due to it was not precise requirement 1.b is marked as failed, but 1.a and 1.c can be marked as a success.

Requirement 2: Implement a Control Unit, can also be considered a success. The controlling unit chosen Arduino Uno rev. 3 was implemented as CU for running the reads from the flex sensor on its analog input, it handled generating PWM so a position could be related to the servo motor, and the 10bit ADC on the Uno was used to convert the analog signal from the flex sensor to an angle that could be presented. Finally, it could display data via a shield and it is user friendly. Thus, all of the sub requirements for requirement 2 has been achieved and can be called a success.

However, before moving on the prototype requirement a comment must be made about the Arduino Uno. The Arduino Uno's microcontroller ATmega328p, caused some trouble during the testing of the servo motor. This was mainly due to a wish of running hardware generated PWM which was a possibility for the Uno/ATmega combo, however due to the servo motor that was chosen required a 50 Hz signal, the hardware generated PWM could not be used [26]. The ATmega328p can not produce a 50 Hz PWM signal, other frequencies close to 50 Hz were tested, but these did not work. Therefore, the PWM ended up being software generated, which is not as precise as hardware PWM. For future work a microcontroller that is able to generate the PWM frequency at which the motor is working on basis of hardware would be a requirement.

Moving on to the status of requirement 3 in Table 4.1, which is a prototype. This requirement is marked as partly achieved, since one of the core components is not met. Starting with sub requirement 3.a, which is a success the knee brace prototype was printed and can be used. However, if it would be possible to get a 3D printer with dissolvable material, since some of the hinges sturdiness was taken away in the assembly process when grinding away plastic so the hinges could be forced into place. Requirement 3.b was to implement an angular sensor on the prototype, and this is definitely a success.

Requirement 3.c was not achieved due to motor assisted movement was not implemented on the prototype, and this is due to a couple of reasons. Firstly, when the servo had reached its set point correctly, it should be disengaged until a new set point would signaled. However, this was not possible because when the "servo.detach()" was used it was observed that the order in the main loop of the code would not be kept. The second reason for this decision, was that it was unknown how big of a servo motor it would require to be able move the lower leg and the knee brace. This issue was left unexplored and should be investigated if further work is to proceed on the project. The proof-of-concept of the servo motor indicated that motor assistance is possible on the full scale, and the implementation of a supervisory reference point control system was a success on the proof-of-concept scale. Even-though, that 3.c was not achieved on prototype scale, it showed great promise on the proof-of-concept scale, which is also a success.

Requirement 3.d: wearable, can be called a success due to that what was achieved in the final system test. This was achieved by using Velcro tape to mount on a leg, and it maintained the position throughout the test. The last sub requirement for the prototype was that it should be low cost. Low cost is a relative term, however it is marked as a

success, since compared to knee braces on the marked it is in the lower end cost vise. The cost of the prototype can be seen in Table 4.2, the prices noted are prices found online and should be taken as an estimate.

Part	Cost (Euros)
1kg PLA filament	≈ 30
Arduino Uno Rev. 3	≈ 24
Flex Sensor	≈ 12
Servo Motor	≈ 12
Velcro 10m	≈ 24
Total Cost	≈ 102 *Euros*

Table 4.2: Cost of prototype

Table 4.2, some of the cost indicated covers larger amount materials than actually used, such as the Velcro tape and the PLA filament, furthermore the design of the knee brace is not depicted. The total cost of around 102 euros is a low cost, it is basically the same as a low-end knee brace such as a Donjoy Reaction Knee brace which cost around 100 euros, and it is definitely lower than a high-end ones that cost around 10 times as much [27].

The prototype itself should be used as a tool for monitoring the process of the rehabilitation of the recovering knee. This means when the recovering knee reaches a specific place in the rehabilitation process, the prototype should not be used anymore. This is the point where the knee is healed enough to start training movements in directions such as sideways or a mix of sideways and forward, this is around the same point where the knee does not have trouble bending and stretching any more. Another important thing to note about the use of the prototype/knee brace, is that it should not build focus on just bending and stretching the leg, since this might move focus away from the rehabilitation program. The prototype, if fully developed, should be used as a tool of guidance, however it is important to keep focus on what the physiotherapist is advising, since they are the trained medical professionals.

In this project, the prototype is modeled on a borrowed blue print, based on dimensions taken by hand which caused some in precisions in the brace. One way to make the brace fit more precisely is to 3D-scan the leg a week after surgery. The dimensions could then be extracted from the scan. This would also ensure a knee brace which would precisely fit every knee it went on, however it would make the prototype's price increase.

The flex sensor and the servo motor has been the focus points during this project. Where the accuracy of the flex sensor could have been corrected, if the used servo motor would have had a feedback signal. This would mean having redundancy in the system with feedback on the angular position from two different sensor. This will improve the system as these signals can be compared to get a more precise position reading, if both of the sensors are very precise. It would also mean that the system would still be able to disclose the angle if one of the sensors would fail during use. For future work a bend sensor more precise and a servomotor which has position feedback would be required.

Software wise the project was not that complicated, however there are some key points

to note in this area also. The calibration function right now demands the knee to be fully stretched and bend at 90 °. However, a freshly operated knee will not be able to do at least the fully stretched position. This was not thought of during the creation of the software, but the way it calibrates currently is based on a calculation where the amount of degrees between these two points is know, so it would be possible to have other position for calibration and still being able to get angle measurement, it would only take an alteration of that one constant more or less to solve this.

The implementation of supervisory reference point control, showed that the supervisor works as intended on the modified code based on the transfer function which was acquired from the system identification of the motor 3.5-System Identification and Modeling. It worked on the proof-of-concept servo hinge, and it should also work on a motor on the prototype, however with a different motor comes a different behavior and it should be expected that alterations are necessary to the system model, if the supervisor should need further development. One of the good things with using Simulink and a system model for the motor, is that every time a alteration has been made to the supervisor, the system can be simulated and this can show if it works as wanted. The simulation of the supervisor saves time testing every individual alteration, and it might also reveal some undiscovered behavior which might not be seen during normal testing.

Chapter 5

Conclusion

Based on the main focus area of this project, monitoring the recovering knees bending abilities, was completed at a satisfactory level, this project is concluded to be a success. Furthermore, while the second focus area, motor assisted movement, was only completed on a proof-of-concept level, it showed great potential and it should only be a matter of scaling for it to be fully implemented on the prototype.

In section 2.1.2-Project Delimitation a specific set of requirements was set for the project over all, and these was evaluated in 4-Discussion. From this evaluation it could be seen that nearly all of the requirements was achieved, and that only two was not completed. The flex sensor used in this project did not have high enough precision and should be replaced with a more accurate sensor, to add the required precision to the prototype.

In section 3.5-System Identification and Modeling showed the process of creation and implementing a supervisory reference point control system. Even-though, the supervisor was not complicated, it showed the potential of what a supervisor could bring to a control system. It was implemented successfully on a proof-of-concept scale. Time was saved by modeling the system, tuning the supervisor in the simulation model, and then converting without any troubles to the used programming language.

In Chapter 2-Problem Analysis different sensors and controlling units were reviewed, in order to make an educated choice for the solution. This chapter also included a review of the knee with focus on its structure and movement, and also a dive into what materials could be used to make a knee brace.

Chapter 6

Perspective

This chapter is dedicated towards the possible future of the project, it will contain a section dedicated towards making sure the requirements for the project is achieved, and a section dedicated towards the future of the prototype.

6.1 The Next Steps

The project ended with two not achieved requirements, high precision angle monitoring and implementing motor assisted movement. These two requirements would be the first issues to attend to if the project was to be continued.

The problem with not having a sufficient enough precision, this should be eliminated by acquiring a more precise sensor. The idea of the flex sensor fit well into the prototype, and if it would be possible to acquire something similar to the flex sensor this would be desirable. The replacement sensor could preferable also be wireless, since this will limit the amount of wires running across the knee brace.

A company called Biometrics Ltd, is a British company that has develop an electronic goniometers which can either wirelessly or wired be fasten to a brace or the leg, and will then relate to the current angle. However, this product is very expensive and this combined with the late discovery was a reason for it not being implemented in the project. However, it would be a candidate for a replacement of the flex sensor if the work was continued.

As for the other part that was not achieved, the servo motor would need to be replaced. First of all an investigations into how much torque is needed for the motor to be able to assist proficiently enough with moving the leg. The next impact-full point would be to ensure that the motor chosen can be disconnected or ”freewheel”. This could be ensured by using HW generated PWM which could be turned off. It could also be solved by implementing a relay, which could turn of the power to chosen motor, the state on the relay board should be known by the supervisory control system, in order to generate a bumb-less startup of the motor, meaning when it is powered on it would smoothly increase the speed until it has reached the current setpoint. With these two points taken care of, it should just be a matter of fitting the motor onto the knee brace, and slightly

altering the code so it would fit the new motor.

6.2 Future Improvements

There are a lot of option to where this project could go from its current state/after the work mentioned in Section 6.1-The Next Steps, and this is what this section will be concerning.

One of the gadgets that could be a good addition to the current prototype would be a sensor which could measure the force of which the lower leg is moving. The implementation of a sensor that could do this would be a good improvement, because that an increase in how much force at which the lower leg can be moved at would show that the knee is progressing well during the rehabilitation.

Another good addition to the prototype would be the ability to measure the circumference of the thigh. This could reveal how much swelling comes from the rehabilitation exercise, and give an indication about the exercises have been too much for the recuperating knee, or if the intensity can increased the next time. This could possibly be done by having a ring of distance sensors measuring just above the knee, the distance would then be read as point that could be plotted and connected, and from this a circumference could be calculated. Another way this might be possible is by knowing the length from the of the Velcro that is securing the brace to the leg, and at the end of this would be some form of sensor, which based on a distance from a certain point on the brace, could be used to calculate the size of the thigh. However, these two mentioned possibilities are just speculations and a way to do this will have to be researched, as these two are not guaranteed to work.

Lastly, implementing 3D scanning of the leg, on which the dimensions of the knee brace could be based on would be a way of ensuring a custom fit for the user. This process is not usually that cheap, but it would end up improving the fit of the brace, which would be beneficial for the rehabilitation process of the knee.

References

[1] Dansk Korsbånds Rekonstruktions Register. *Årsrapport 2016*. [Online]. Available from: https://www.sundhed.dk/content/cms/0/4700_dkrr_aarsrapport_2016.pdf, 2016. Accessed 4th October 2016.

[2] Egrabin @ Sports Knee Therapy. *ACL Surgery Recovery Timeline*. [Online]. Available from: http://sportskneetherapy.com/acl-surgery-recovery-timeline/. Accessed 17th December 2016.

[3] TECH-FAQ. *What is a Goniometer?* [Online]. Available from: http://www.tech-faq.com/goniometer.html. Accessed 17th December 2016.

[4] GNR Systems. *Baseline 12P̈lastic Goniometer 360 degrees*. [Online]. Available from: https://www.gnrcatalog.com/Baseline-12-Plastic-Goniometer-360-p/gfa121000.htm. Accessed 4th October 2016.

[5] Encoder Products Company. *The Basics Of How An Encoder Works*. [Online]. Available from: http://encoder.com/core/files/encoder/uploads/files/WP-2011_Basics%20of%20How%20an%20Encoder%20Works.pdf. Accessed 12th October 2016.

[6] Creative Robotics Ltd. *What are Quadrature Encoders*. [Online]. Available from: http://www.creative-robotics.com/quadrature-intro. Accessed 10th December 2016.

[7] Allergro MicroSystems. *Latching Switch Hall-Effect IC Basics*. [Online]. Available from: http://www.allegromicro.com/en/Design-Center/Technical-Documents/Hall-Effect-Sensor-IC-Publications/Latching-Switch-Hall-Effect-IC-Basics.aspx. Accessed 13th October 2016.

[8] Nathan Hurst. *Simple way to make a position measuring hinge*. [Online]. Available from: http://njhurst.com/blog-files/01356332288/measuring-hinge.jpg. Accessed 17th December 2016.

[9] Arduino. *ArduinoGenuino UNO*. [Online]. Available from: https://www.arduino.cc/en/Main/ArduinoBoardUno. Accessed 30th October 2016.

[10] ROBOMART.com. *Arduino Uno R3 Board*. [Online]. Available from: https://www.robomart.com/image/catalog/RM0058/01.jpg. Accessed 19th December 2016.

[11] Raspberry PI Foundation. *Raspberry Pi 3 Model B*. [Online]. Available from: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/. Accessed 30th October 2016.

[12] element14.com. *Raspberry pi 3 model b technical specifications*. [Online]. Available from: https://www.element14.com/community/servlet/JiveServlet/showImage/102-80899-15-252356/Pi3+Breakout+Feb+29+2016.png. Accessed 19th December 2016.

[13] WebMD. *Picture of the Knee*. [Online]. Available from: http://www.webmd.com/pain-management/knee-pain/picture-of-the-knee. Accessed 28th October 2016.

[14] Physioroom. *Picture of a Donjoy Legend*. [Online]. Available from: http://www.physioroom.com/images/products/38206_image.jpg. Accessed 30th October 2016.

[15] Physioroom. *Donjoy Legend Knee Brace*. [Online]. Available from: http://www.physioroom.com/product/Donjoy_Legend_Knee_Brace/2034/38206.html. Accessed 30th October 2016.

[16] American Friends of Tel Aviv University. *Steel-Strength Plastics and Green, Too* [Online]. Available from: http://www.aftau.org/news-page-environment--ecology?&storyid4703=1375&ncs4703=3. Accessed 10th December 2016.

[17] Caltech. *PID Control*. [Online]. Available from: http://www.cds.caltech.edu/~murray/books/AM08/pdf/am06-pid_16Sep06.pdf. Accessed 18th December 2016.

[18] University of Michigan. *Introduction: PID Controller Design*. [Online]. Available from: http://ctms.engin.umich.edu/CTMS/index.php?example=Introduction&section=ControlPID. Accessed 10th December 2016.

[19] John Smith aka. fcubed @ Thingiverse.com. *Modular Knee Brace*. [Online]. Available from: www.thingiverse.com/thing:1509833. Accessed 11th December 2016.

[20] CC licensing. *Creative Commons Attribution 3.0 Unported*. [Online]. Available from: https://creativecommons.org/licenses/by/3.0/. Accessed 11th December 2016.

[21] DFRobot. *File:LCD&KeyPad Shield.jpg*. [Online]. Available from: https://www.dfrobot.com/wiki/index.php/File:LCD%26KeyPad_Shield.jpg. Accessed 6th January 2017.

[22] SpectraSymbol. *Flex Sensor*. [Online]. Available from: http://www.spectrasymbol.com/wp-content/themes/spectra/images/datasheets/FlexSensor.pdf. Accessed 12th December 2016.

[23] RS Components. *Parallax Inc, Servomotor, 140 mA, 4 to 6 V*. [Online]. Available from: http://dk.rs-online.com/web/p/servomotorer/7813058/?sra=pmpn. Accessed 13th December 2016.

[24] National Instruments. *Acquiring an Analog Signal: Bandwidth, Nyquist Sampling Theorem, and Aliasing*. [Online]. Available from: http://www.ni.com/white-paper/2709/en/. Accessed 27th December 2016.

[25] Jacob Lynge. *Supervisory Reference Point Control System*. [Online]. Available from: https://www.youtube.com/watch?v=nBVRv8qXeWM. Accessed 9th January 2017.

[26] Arduino. *Adjusting PWM Frequencies*. [Online]. Available from: http://playground.arduino.cc/Main/TimerPWMCheatsheet. Accessed 21th December 2016.

[27] Sport112. *Reaction Knee Brace*. [Online]. Available from: https://www.sport112.dk/reaction-knee-brace/?options=2707446,2707087&gclid=Cj0KEQiAyuPCBRCimuayhb3qqvwBEiQAgz62kcWaUniHi3Gbs0WhKGPqdS_42X7WjxOvI9pwxOOFUjUaAuRv8P8HAQ. Accessed 21th December 2016.

Appendix A

Unedited Brace Model

Original layout for the "Modular Knee Brace" made by the user Fcubed on thingiverse.com

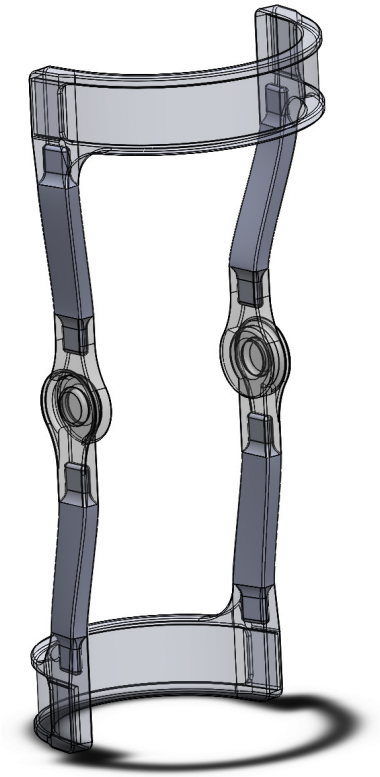

Figure A.1: Unedited modular knee Brace model

Since I am using the model here are the attribution card, as required by the license, just below in figure A.2.

Figure A.2: Attribution Card

Appendix B

Modified Knee Brace Model

This is a model of how the knee brace prototype would look. like with both flex sensor and servo motor attached to it, without velcro straps.

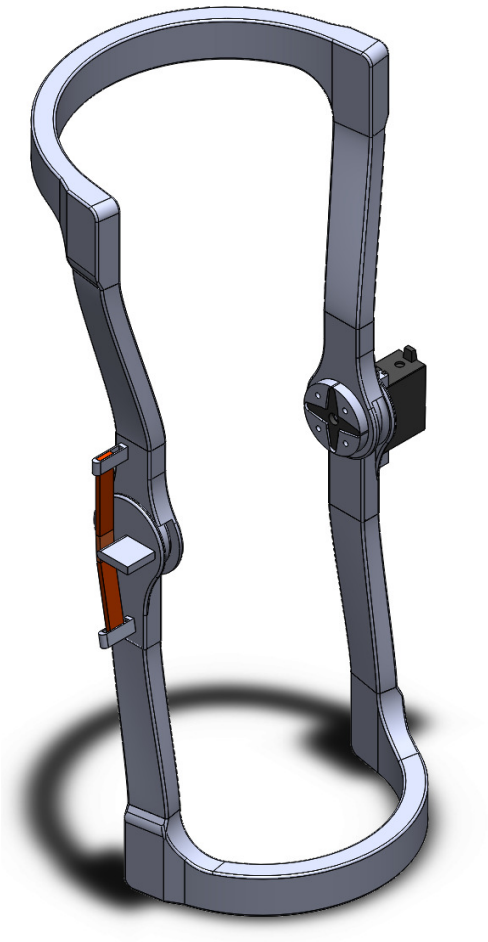

Figure B.1: Illustrates how the prototype looks

The knee hinge where the flex sensor is located has been altered a bit and now only has the two end holders, and do not have bit at the middle of the hinge which can be seen on B.1.

Appendix C

Modified Servo Hinge Parts

This appendix will contain design drawings for the servo hinge, all of the measurements on the drawings is in millimeters, and ISO standard has been used for sizing of the holes for screws.

Motor Mount

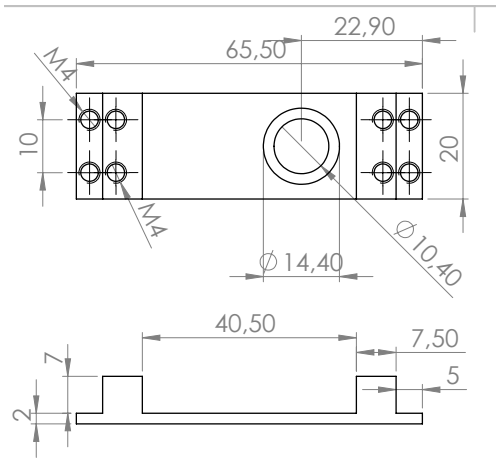

Figure C.1: Design drawing of the motorholder

Servo Axle Extension

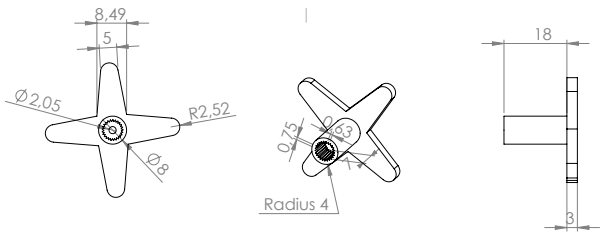

Figure C.2: Design drawing of the servo axle extension

Servo Axle Extenstion Holder

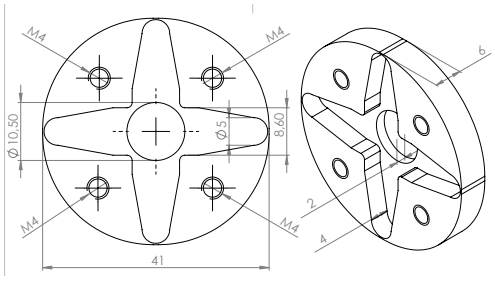

Figure C.3: Design drawing of the servo Axle Extenstion holder

Appendix D

Arduino PinOut

The following table shows the used externally used pins on the Arduino, however the pins used for controlling the LCD display has been left out.

Arduino Pin	Purpose
A0	Keypad Buttons
A1	Flex Sensor Input
Vin	7V external power Input
5V Out	Servo Power Supply
D3	PWM Signal Servo
GND	GND from signal
GND	GND Out

Table D.1: Arduino Pin Out

Appendix E

System Identification Test Code

This is the code which was used for the system identification test of the servo motor.

```
void loop() {
  // put your main code here, to run repeatedly:
  while(o <=625){
    timea = millis();
    if (i == 30){
     sAngle = random(90);
        if (sAngle < 3){
          sAngle = 3;
        } else if(sAngle>78){
         sAngle = 78; // true 90
        }
     kneeServo.write(sAngle);
     i = 0;
    }
    currentAngleCalc();
    Serial.print(sAngle);
    Serial.print(" ");
    Serial.println(cAngle);

   i = i + 1;
   o = o + 1;
   timeb = millis();
   calcTime = timeb - timea;
   //Serial.println(calcTime);
   currentTime = millis();
   while (currentTime - timea - calcTime < sampleTime){
      currentTime = millis();
   }
  }
}
```

Appendix F

MATLAB Supervisor Code

This is the final code segment of the supervisor function, which was used in the simulation model.

```
function theta_s = fcn(formerAngle, formerAngle2, flex, theta,
   timeF, timeT)

if formerAngle == flex && formerAngle2 == flex
    moving = false;
else
    moving = true;
end

if abs(formerAngle - flex) < 5
    if moving == false
        theta_s = formerAngle;
    else
         angularVelocity = (flex - formerAngle)/(timeF - timeT);
         timeD = timeF - timeT;
         theta_s = flex + angularVelocity*timeD;

    end
    else if abs(formerAngle - flex) > 25
      theta_s = flex;

        else
       angularVelocity = (flex - formerAngle)/(timeF - timeT);
       timeD = timeF - timeT;
       theta_s = flex + angularVelocity*timeD;
    end
end

    if theta_s > 78
     theta_s = 78;
        else if theta_s < 3
         theta_s =3;
        else
```

```
33          theta_s = double(theta_s);
34
35          end
36      end
37  end
```

Appendix G

Method Diagram

This is a method diagram of the code used for the final system test

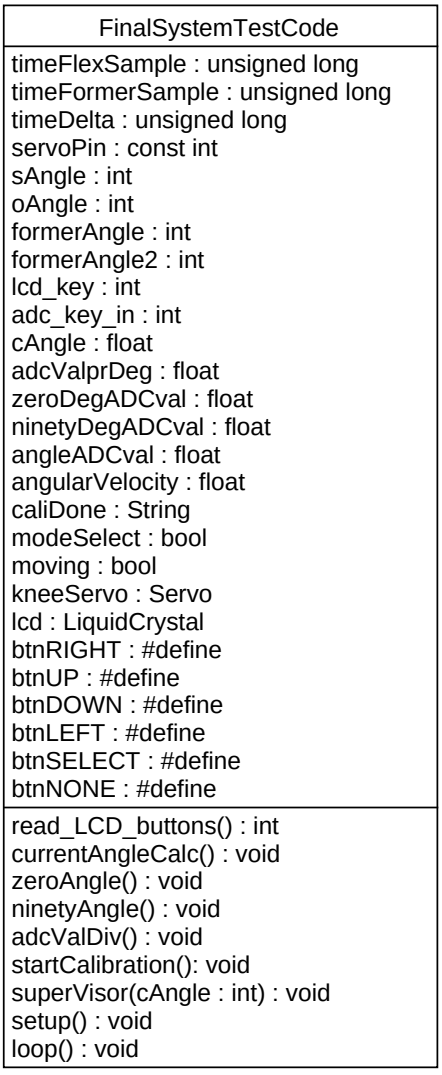

Figure G.1: Shows a method diagram of the test program.

Appendix H

CD

This appendix have an CD physically attached to it which contains a .zip file with full final system test code, and the report as a digital copy. Furthermore this .zip file will be uploaded together with the report submission, so it is also download able.