# Contents

# 1 Introduction

This paper will explain the processes followed in designing a burger-flipping robot that is controlled by electromyography (EMG) signals. This is done as part of a project for the BioRobotics minor at the University of Twente. The target of this project is to design a robot that can be used by patients of Duchenne Muscular Dystrophy (DMD) in order to perform physical manual labor.

## 1.1 Casus

Duchenne Muscular Dystrophy (DMD) is a genetic disorder which causes muscular loss. This muscle degeneration occurs progressively over the course of the lifespan of those diagnosed with the disease, which averages to only up to 30 years - although, this number is continuously increasing due to healthcare advancement. This disease mainly affects boys, as the gene for DMD is found on the X chromosome carried by the mother. DMD causes a mutation in the dystrophin encoder gene, which is responsible for keeping muscle cells together.
DMD symptoms can develop in early childhood and begin with lower limb weakness, starting from proximal muscles (muscles close to center of body). This results in a "waddling" gait and difficulties in standing up from a sitting position or jumping. Children with DMD will struggle to keep up with their peers in physical activities. By early teenage years, children diagnosed with DMD will find themselves bound to wheelchairs. Following lower limb dystrophy, upper limb muscles begin to weaken, also starting from proximal muscles. Decline in respiratory and cardiac performance also gradually occurs.
There are also multiple other genetic muscular disorders which are variations of DMD. There is currently no cure for DMD. Due to the physical restrictions of DMD, people affected by it often rely on relatives or caretakers for assistance. To aid in improving the quality of life for people with DMD, a robot will be designed to allow a person diagnosed with DMD to be able to perform a physical manual labor job. Other than increasing the job opportunities for them, the robot will also allow them to rely less on their caretakers in performing certain tasks.

## 1.2 Targeted Challenge

To further define the goals of this project, a task is specified. Prior to picking a task, the group has brainstormed multiple tasks, taking into account the needs of people with DMD who are the target users (See Appendix A).

The goal of this project is to create a device that will allow people with DMD to perform a physical, professional task. This will allow them to work with some degree of independence and doesn't limit them to jobs requiring mental tasks only. The group has chosen to design a burger-flipping robot to enable these people to work in a fast food store, cafe, or restaurant. This robot would not only work in a professional setting, but also gives people with DMD the ability to prepare some basic food at home and make them less dependent on their caregivers. A burger flipper is relatively simple compared to some other cooking devices and can be realized in a 2D-plane robot.

# 2 Function Analysis

## 2.1 Stakeholders

To better define the design of the project solution, it is important to determine the stakeholders, or who all is involved and affected. Below is a list of the groups of people who are considered stakeholders:

- DMD patients
- Patients' caretakers or family members
- Designers/developers
- Government
- Health-care or educational institutes
- Business companies hiring patients (eg. McDonald's)
- Customers of companies

People diagnosed with Duchenne Muscular Dystrophy (DMD) and other similar diseases are the main stakeholders as well as the target users. The goal of the project is to build a robot that is to be controlled by the DMD patient, allowing them to perform manual labor. For this stakeholder, ease of use of the product is important for them, as well as safety.

Caretakers or family members of the DMD patients are also stakeholders, as the product will affect the dynamics in the relationships. The DMD patients will be able to do certain tasks by themselves with the use product, therefore, requiring less assistance from their caretakers or family members. Although, this stakeholder might be the one to help maintain the robot on a daily basis, so a key driver is the low maintenance of the product.

Within the time frame of this project, the designers and developers of the product is considered to be a stakeholder, as they are the ones making the major design decisions.

As a future outlook, other stakeholders are the government, healthcare, and educational institutions, and companies and their customers. The government may have an input on funding and on where to apply the technology. Healthcare and educational institutes can further improve the product through research and find further applications for it as well. Business companies that will hire DMD patients for manual labor is also a stakeholder. The key drivers for this stakeholder is in the cost, efficiency, and safety of the product. The customers of the companies are also a stakeholder as they will be the one to experience the results of the product used by DMD patients. Their key driver would be safety and good service.

## 2.2 Function Definitions

- Pick up burger
- Flip burger
- Move to specified locations
- Measure and analyze EMG signals and use it for controls of movements
- Maintain stability and supports structure
- Ensure safety

## 2.3 Requirements and Specifications

To get a clear view of the task of the robot a list of requirements and specifications has been made.

### 2.3.1 Requirements for Project

- Robot needs to allow people with DMD to do a professional job that requires manual labor.
- The robot must have 2 degrees of freedom.
- The robot must have 2 moving parts (2 bodies).
- The robot must move in a 2D plane.
- The robot should be controlled by 2,3 or 4 EMG inputs
- Robot and its operations need to fit on a 1000x600mm tabletop
- Do not use liquids (safety reasons).
- Load should be less than 200 Grams.
- No sharp moving parts
- No exoskeleton

### 2.3.2 System Specifications

- Needs to be able to flip a burger weighing minimum 30 grams and up to 120.5 grams and possibly more (based on burger sizes of McDonalds).
- Burger needs to land with a 180 degree flip done (to cook other side).
- Burger must not be thrown away (due to flip).
- The spatula needs to be easily detachable (for cleaning).
- Needs to be able to flip a burger more than once.
- Needs to flip either one or two burger at a time
- If more, there could be problems with uneven distribution of heat on hot plate
- Needs to reach any point of a plate of 30 cm by 60cm
- Reasonable size that can fit two rows of burger patties
- As much of the weight in the base as possible
- Designed for burgers on a hot plate
- Placement on the tabletop, not the wheelchair
- For safety
- Must be able to move the burger out of the grilling plate onto a serving plate

# 3 Concepts

## 3.1 Concept Generation

In order to aid in coming up with concepts, a morphological chart was created (See Appendix B).

Three concepts were then chosen and further analyzed using drawings. Feedback was received from peers and experts. These three concepts can be found in the following sections.

## 3.2 Concept 1

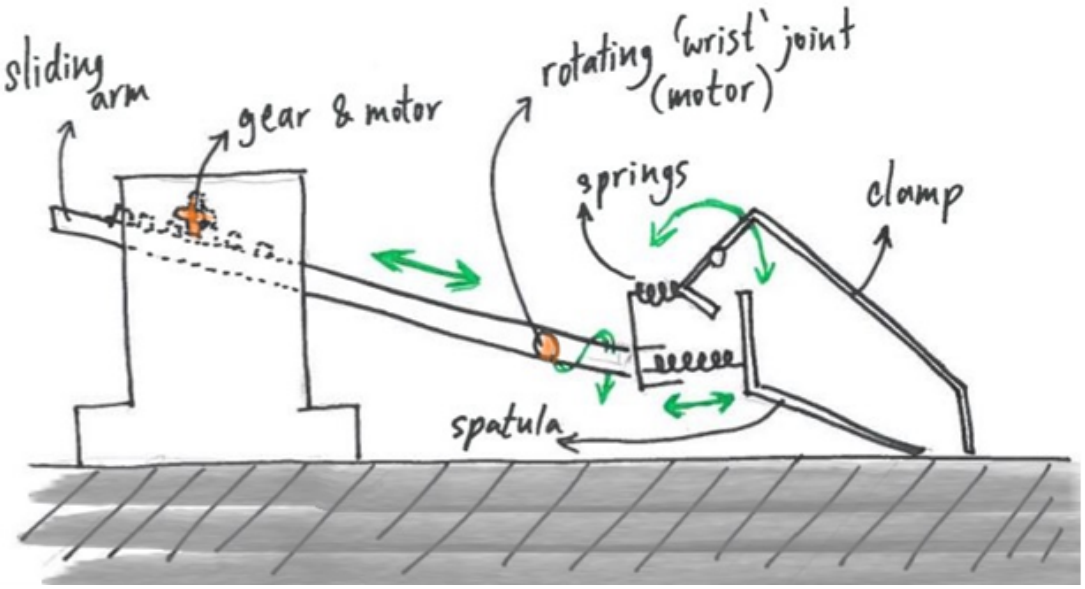


Figure 1: Concept 1 visualization

The first concept we analyzed and design was a simple design, where we have one robotic arm declining on an angle, and has one gear and motor. The gear would make sure for the arm to move in one axis, to make the arm move forward or backward when necessary. Moreover, as seen in the design above there is a rotating joint, including a motor on the edge of the arm, which makes sure to flip the spatula so the burger will fall off.
The spatula itself was designed to be flexible enough to be pressed against a surface, which springs would be ideal for it to move backwards after a force is applied against it. In addition, the spatula will have a clamp, which will after push the burger inside, and in the end the join wrist would rotate so the burger can make a full flip.

The first concept has some pros and cons, a detailed analysis of these upsides and downsides will lead to a better decision in the end.

Table 1: Pros and cons of concept 1.

| PROS | CONS |
|---|---|
| Simple design. | Can flip only one burger at a time. |
| Easy to control. | Hard to pick burgers at a specific position. |
| Doesn't take up a lot of space in workplace. | Is not able to move along the grill. |
| Flexible Spatula. | Fixed at one position, and limited reach. |
| - | Picking up the burger might be hard, without any supporting arm. |
| - | The rotating joint will not ensure a full flip. |

### 3.3 Concept 2

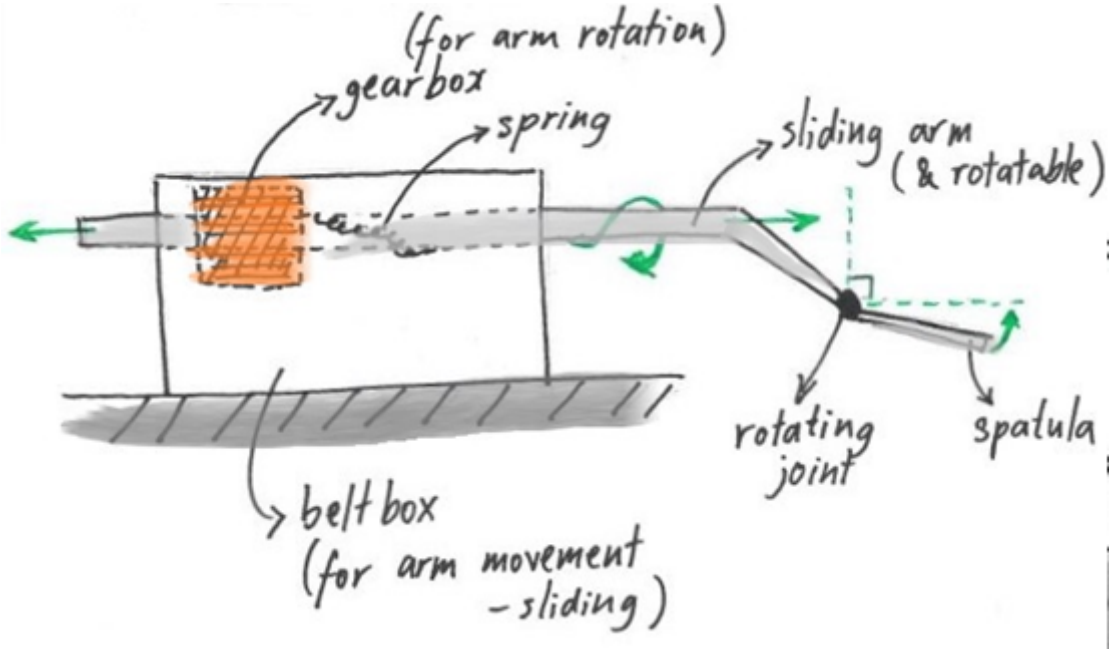


Figure 2: Concept 2 visualization

The idea behind concept 2 was based on a sliding arm that slides forward and backwards, and as in concept one has a rotating joint, which slightly rotates, in addition has a gearbox inside a box that makes sure the arm rotates, moreover, the spring which is connected to the gearbox releases quickly, so the arm can make rotational movements. In this concept a gear box was initially intended to be designed, which makes sure the arm rotates to its original orientation, and afterwards the rotating joint returns to the original angle. However, the gearbox was very hard to implement in real practical work, therefore this design was neglected.

Table 2: Pros and cons of concept 2.

| PROS | CONS |
|---|---|
| Design makes sure for a full 180 flip. | Can flip one burger at a time. |
| Sliding arm is rotatable in whole. | Limited reach on the grill. |
| - | Includes a gearbox. |
| - | Springs might make it hard for the arm to rotate at times. |

Concept 2 was not ideal for our idea, we wanted to make sure that we have enough reach on the grill, and be flexible at any position on the grill. On top of that, make sure we can at least flip two burgers at a time.

### 3.4 Concept 3

After analyzing the previous concepts, the conclusion was that a more improved and more controllable robot has to be designed.
Therefore, the following concept was our final design, and was based on all of the previous designs together, with more reach on the grill, can flip two burgers at a time, and has a movable base to move alongside the grill.

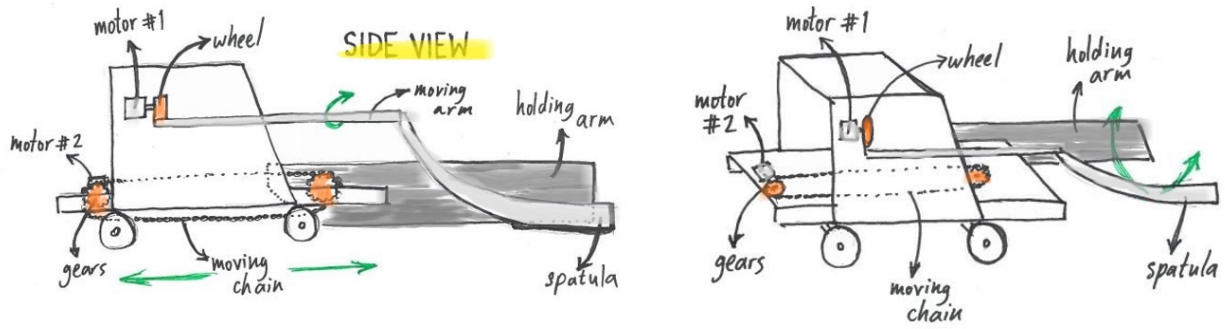


Figure 3: Concept 3 sideview visualization

In the side view we can see two motors that can control one gear, which is responsible for the cart movement alongside the grill, and the other motor makes sure to move the entire spatula at a certain angle, and sweep the burger against the holding surface and make sure the burger will lay on the spatula. After the sweep of the spatula, and burger resting on the spatula, the engine makes sure the spatula will move back to a 90-degree angle, and afterwards the burger can slide off and make a 180-degree flip on the grill surface.

To make sure the burger flips, we designed an edge on the spatula, so while the burger is resting on the spatula, and moves back to its original angle, the burger will start sliding off and make a complete 180-degree flip due to a higher edge on the spatula. The edge on the spatula is shown below.

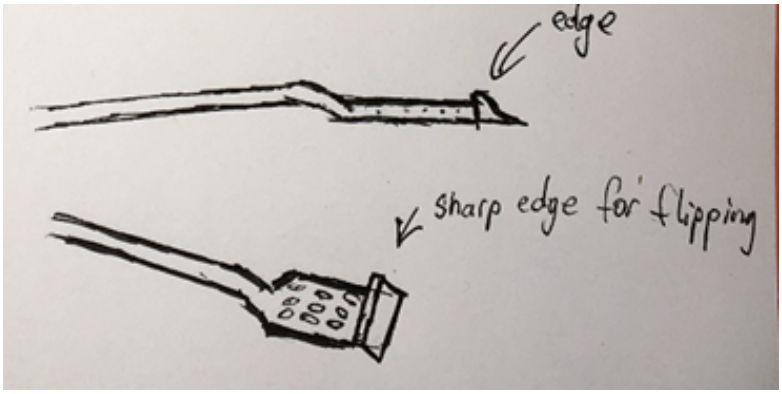


Figure 4: Spatula edge

On top of the design on the edge of the spatula, it was made sure that the spatula is also easily cleanable. So, it was designed that the spatula can be detached easily from the handle, and be cleaned after use.

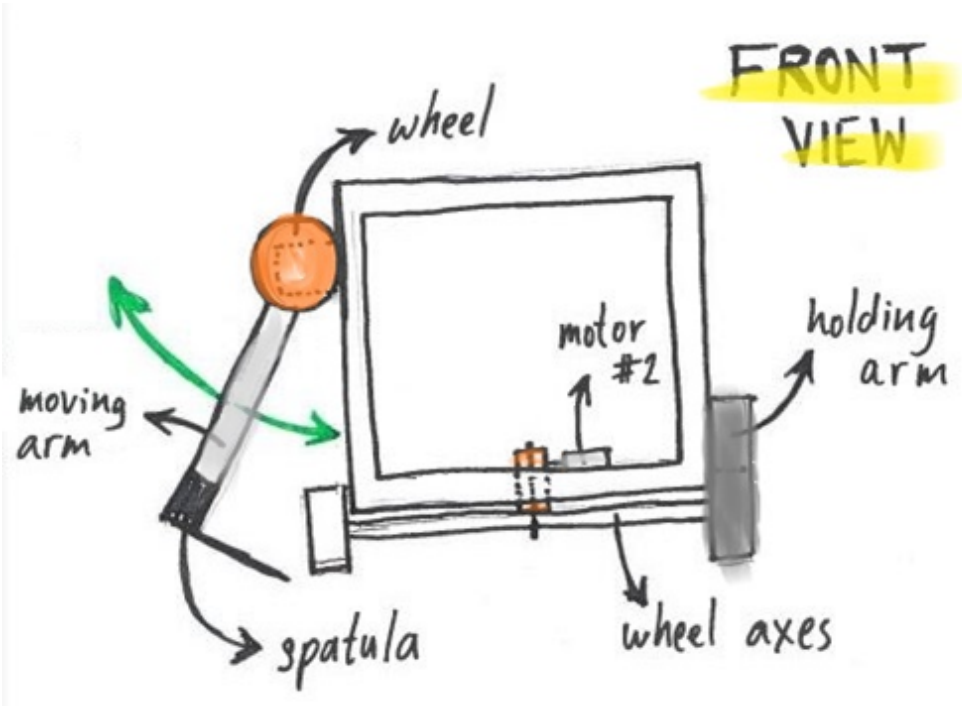


Figure 5: Front view of final concept

The front view gives us also another insight, of how the system will behave. In

addition, to make sure the sweep of the spatula doesn't crash into the blocking arm, we made it possible for the blocking arm to move along the spatula, if excessive force is applied.

The table below shows the pros and cons of this specific design:

Table 3: Pros and cons of concept 3.

| **PROS** | **CONS** |
|---|---|
| Has enough reach on the grill | Counter-weight problems might occur. |
| Can flip two burgers at the same time | - |
| Design makes sure for a full 180-degree flip | - |
| Very controllable design | - |
| Stopping arm makes sure the burger will slide on the spatula | - |
| Detachable spatula | - |

### 3.5 Concept Evaluation

In order to make the best decision, that suits the ideal design to the main idea, weights were put on each requirements to determine its importance. Votes were then made on which system is more feasible in regards to each requirement. The weights were multiplied, and the points were added up (See Appendix C). In conclusion, the third design concept was evaluated to be the best design, although it is only slightly better than the first concept. Through evaluation, the third design was re-evaluated again and changes were made to improve on the weak points. The strong points of the other two concepts were taken into account for the final design.. This will be discussed more in the 'Final Design' section.

## 4 Design

### 4.1 Final Design

Following the choice of Concept 3 and further evaluation, the following changes were made to improve the design (and satisfy the set functions and requirements):

- The spatula is now detachable, allowing for cleaning and therefore, increasing health safety;
- The robot cart now moves side to side rather than forwards and backwards, allowing for burgers to be re-flipped, as well as precise control of which burgers to flip;
- Support for the arms is increased;
- And hinges are included to ensure a fluid scooping motion by the spatula.

With these changes implemented to the design, a SolidWorks model was created.

## 4.2 Modeling

### 4.2.1 Solid Works

Below are the SolidWorks model created. More clear visuals of the SolidWorks models can be found in Appendix D. The labeled components are as follows:

1. Spatula Plate
2. Spatula Stick
3. Spatula Connection
4. Spatula Arm
5. Holding Arm
6. Top Plate
7. Side Mount
8. Wheels
9. Bearings
10. Gears
11. Side Plate
12. Motor (for Holding Arm)
13. Motor (for Cart Movement)
14. Hooks for wheel rod
15. Space for controllers and shields

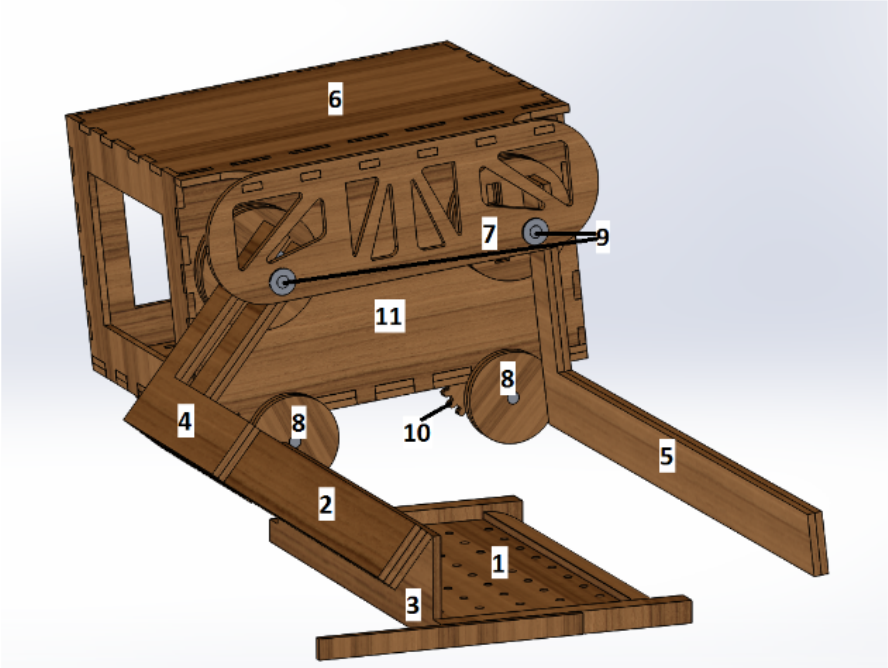


Figure 6: Front view of SolidWorks model

The numbers represent each component which was used to construct the model.

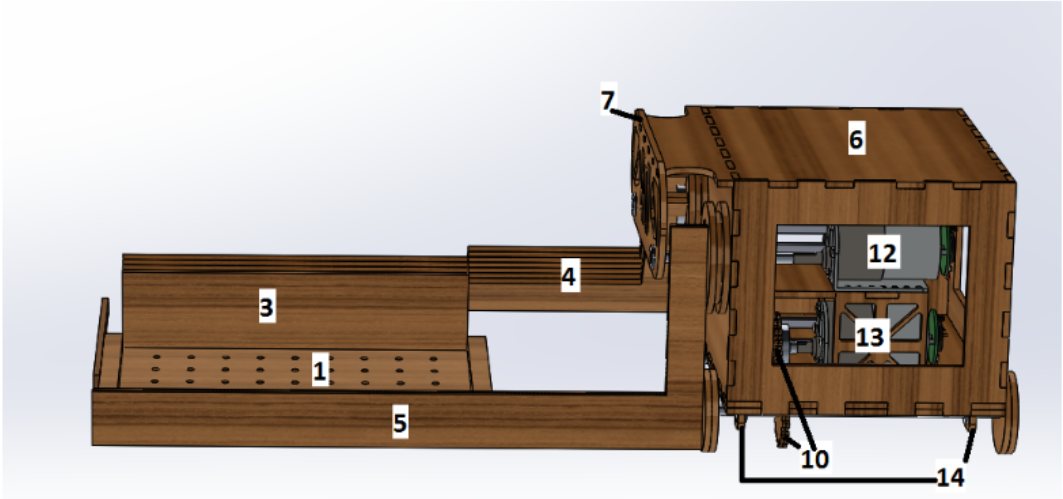


Figure 7: Side view of SolidWorks model

The side-view shows the spatula and the stopping arm, including the motors which lay inside of the cart. Each of the motor is responsible for the movement of the cart and the sweeping movement of the spatula. While the stopping arm, is designed to withheld enough force, so the arm won't break in any overshoot.

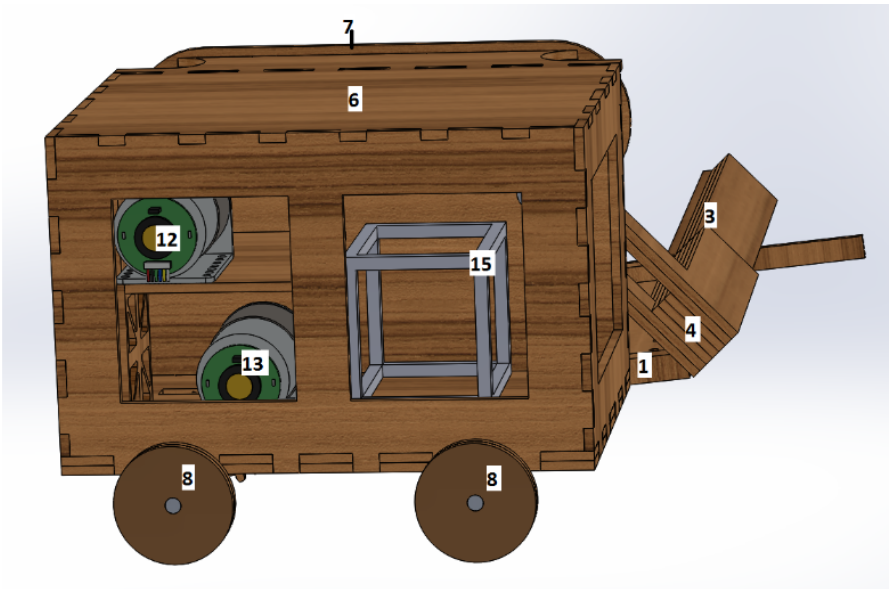


Figure 8: Back view of SolidWorks model

The back-view shows a better visualization of the positioning of the motors inside the card. The above motor is screwed on a wooden piece, and it controls the spatulas movement, while the lower motor is connected to a gear, which moves the wheels of the card, making the cart movable alongside rails.

### 4.2.2 Dynamics

To assess the behavior of the robot a physical model should be made, using this model the controller can also be tuned. The model of the robot is split into two part, the moving of the cart and the swinging of the spatula. The torque and inertia relation is given below. Finding the inertia is described in section[8.5.1]

$$T = J_{tot}\ddot{\theta} + D\dot{\theta} \tag{1}$$

At This point the model is not complete as the motor still needs to be added. The motor can be simplified to a system with an voltage input, a resistor and a coil which changes current into a torque. The relationship between the torque and current is given by the motor constant $K_m$ see appendix[8.5.2]. This torque is applied to the motor and then through gears to the rest of the robot. The equation can now be written as. To simplify the model only the damping of the motor will be taken into account which is $Km^2/R$. Taking this into account the response of the system can be shown as the following transfer function. A simulink model has also been made from this transfer function (see fig[9]).

$$\frac{\theta(s)}{v(s)} = \frac{1}{\frac{J_{tot}R}{K_m}s^2 + k_m s} \tag{2}$$

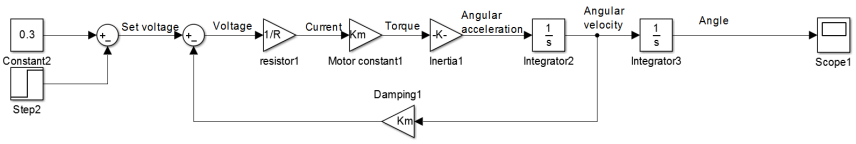


Figure 9: Simulink model of the robot

### 4.2.3 Filter design

With the final design the number of distinct EMG signals that we need to read is minimally 4, moving forward and backward for the cart and clockwise and counterclockwise turning for the spatula. Initially the idea was to use only the left and right biceps and try to get two levels of intensity with which to initiate the directions of movement. Because of difficulty of fine-tuning the EMG filter however we decided to add a third signal which could be placed at either the triceps or the lower arm. As for the filters applied we went for a basic filter group, High-pass filter → Notch filter → Rectification → Low-pass filter. This filter group was chosen as it seemed to be a lot easier to program in C++ than a MOVAG or RMS filter. As for the details of the filter, the Shield-EKG-Emg already has integrated filters however the high-pass filter is put at 0.16Hz while according to research useful EMG signals start at around 10Hz. As such we put the high-pass filter at 10Hz to remove noise that could in no way be part of the EMG. This also makes sure that there is no baseline drift. Aside from the high-pass filter at 0.16Hz there is also a so-called "Besselworth" filter which is a combination of a bessel filter and a butterworth filter however this filter is mostly used to prevent aliasing in a 10-bit AD-converter and does not add much to the filtering that we need to do. The Notch filter that we apply is put at 50Hz to remove the mains hum. With all the noise removed the signal is rectified to make the signal easier to analyse. While testing our filter we also found that we almost always had the biggest increase in voltage during muscle activity at around 11-12Hz and as such we decided to also add a 4dB peak gain at 11Hz to amplify parts of the signal that we knew for sure to be useful. Lastly the low pass filter is applied at 6Hz to determine the EMG envelope. The filter could have been placed at a lower frequency than 6Hz but this might cause too much loss of data which could make the analysis harder.

### 4.2.4 Controller

The goal of both motors is different. The motor moving the cart only needs to have a constant velocity and overshoot is not a big problem as the effecting scooping area is very big. The motor for the spatula has a very different task to that of the cart this motor needs precise velocity control with low overshoot as otherwise the robot could destroy itself. Taking each task into account the motor driving the Cart will not use a controller embedded in the software. As for this motor the person operating it is the controller himself choosing when to turn the voltage applied to the motor on or off using EMG signals.

The motor to the spatula will need a controller. The controller chosen for this is a PID controller. PID was chosen as it adds damping and reduces the steady state error. The risk of picking PID is that it could lead to an unstable system, to avoid this risk the tuning rules will need to be followed (see section 8.6.1). To correctly tune the PID controller a crossover frequency must be set, this has to be less than 1/10 of the sampling frequency, the sampling frequency used is 1000 Hz so the chose sampling frequency of 15 rad/sec fits this requirement. The response of the controller can be viewed by the following transfer function.

Transfer function PID

$$K(s) = \frac{P(sT_Z + 1)(sT_I)}{sT_I(sT_P + 1)} \tag{3}$$

To implement the values found for the transfer function into the software it has to be rewritten into the following form of the PID controller.

$$C(s) = K_P + \frac{K_I}{s} + \frac{K_D s}{s\tau + 1} \tag{4}$$

Rewriting gives the following equations and values see appendix[8.6.2].

$$K_P = \frac{T_z + T_I}{T_P} = 30 \tag{5}$$

$$K_I = \frac{P}{T_I} = 3.51e - 05 \tag{6}$$

$$K_D = \frac{T_Z P}{T_Z + T_I} = 4.93e - 06 \tag{7}$$

$$\tau = T_P = 0.021 \tag{8}$$

Combing the controller with the model shows how the system will respond to certain inputs, figure 10 show the simulink model of this system. The step responses and bode plots of both motors with their respective controls can be found in section [19].

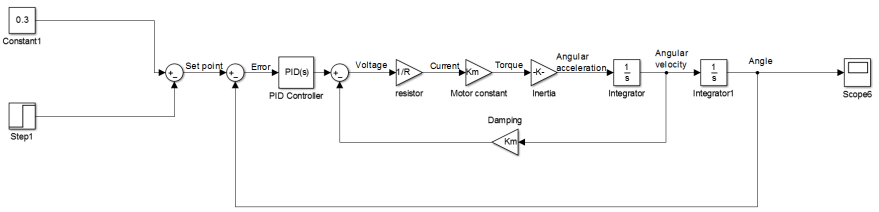


Figure 10: Simulink model of the spatula

## 4.3 Realization

### 4.3.1 Mechanics

With the SolidWorks model, plywood was cut into desired parts with a laser cutter. When building and putting the parts together, some changes were made in order to accommodate for unaccounted factors; thus, improving the performance of the robot.The changes made to the mechanical design are stated in this section, as well as the reasoning behind the changes.
The corners of both the holding arm and the spatula arm that are adjacent to the wheels of the robot were filed. This is to reduce the chance of the arms scraping or hitting the wheels, which is not good for neither the swinging motions of the arms nor the positional movements of the robot itself.
The position of the arms were also switched around, so instead of the holding arm being attached to the motor, the spatula arm is attached. This change was made in order to improve the burger-scooping system and to increase the stability of the robot throughout the motion.
A factor that was not expected was that the plywood that was used in building the robot would not be completely flat. This resulted in the gears that was cut out to be slightly bent, therefore, it did not properly fit the slit that was cut out in the bottom plate for the gears. So as the robot moved and the gears rotated, at certain points, the gears would scrape against the side of the slit. To solve the problem, the slit was filed to create a slightly larger hole.
Problems were also faced with the spatula and scooping system due to the bending of the material. Due to the slight bend, the spatula would not properly slide along the surface of the table (comparable to a hot plate for cooking burgers). This resulted in difficulties in picking up the burger from the middle parts of the spatula, as it would hover slightly above the table and slide into the side of the burger rather than under it. Although, the material was not the only reason for the problems with the spatula, as the change in how the spatula plate is attached to the stick also contributed. It was decided that the spatula plate would be attached directly to the spatula stick, rather than have an extra piece of wood to connect it. This is because the surface of the spatula and the thinness of the connecting wood did not allow for stable gluing. By removing the connecting piece of wood, the weight on the arm is also reduced, allowing for better swinging and scooping movement.
To fix the problems with the spatula during the scooping movements, springs were added to make sure the spatula stays open and that the edges slides along the table surface properly. But due to the springs, the spatula does not fold itself in easily when coming down from the swing to scoop the burger, causing it to hit the table surface and get stuck. This problem is then fixed by filing the edges of the spatula plate that is attached to the stick, so that the spatula plate is not at a fixed angle.
Another unexpected factor was with the glue not being strong enough for the spatula arm. Gluing the spatula arm to the rod was not stable enough, especially with all the weight. Therefore, a mounting hub was used to mount the spatula arm to the rod instead. By doing so, the stability of the arm during rotation is increased and the arm is also prevented from sliding against the side of the box.
Because the mounting hub was not included in the design model, the space needed to

include the hub was more than expected. To create space, the side mount plate was cut in half, and thus the bearings on the side mount for the spatula arm was also removed. This spatula arm was still stable even without the second bearings. The side mount plate could not be removed completely as it is needed for the stability of the holding arm. The reduction of the side mount plate also allows for an increase range of motion for the spatula arm.
Other than the changes mentioned above, an aspect that was included to allow for the realization of the mechanical design is the string system that coordinates the movement of the two arms. This stringed system was not shown in the modeling, but was taken into account by the design of both arms. By attaching strings in a certain position on the arms, it was possible to coordinate the holding arms to be pulled in when the spatula arm swings down. This movement helps to push the burger patties onto the spatula.
Apart from the mechanical realization of the robots, rails were also added to the table. These rails are there to make sure that the robot moves in a straight line, as the wheels, like the gears, are slightly bent, due to the material. The rails also restrict the movements of the robot to only within the table surface, thus preventing the robot from falling off.

### 4.3.2 Construction

After following each step of the SolidWorks model, the final model was constructed piece by piece. As seen in the picture below, the model is working accordingly to the initial idea. The picture, explains the steps of picking up the burger, to flipping the burger in a 180 degree manner.

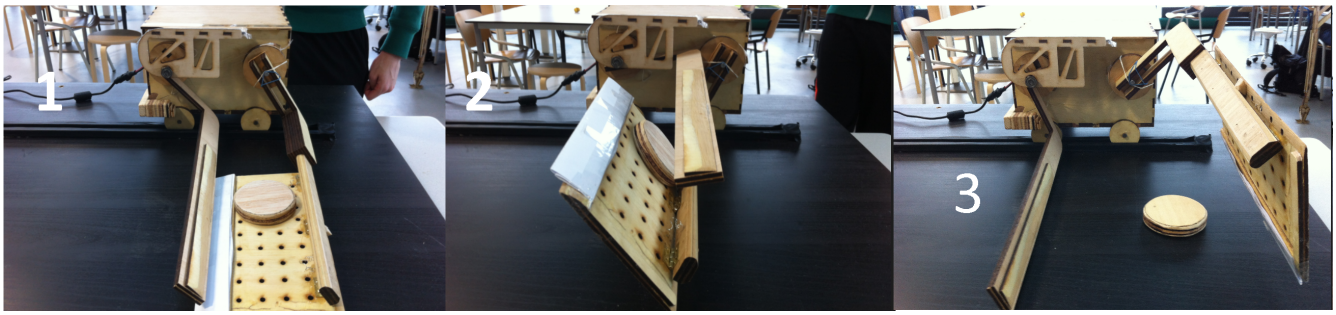


Figure 11: Visualization of the process

To conclude, different angle of the final product will be included below. More images of the final product can be found in Appendix H.

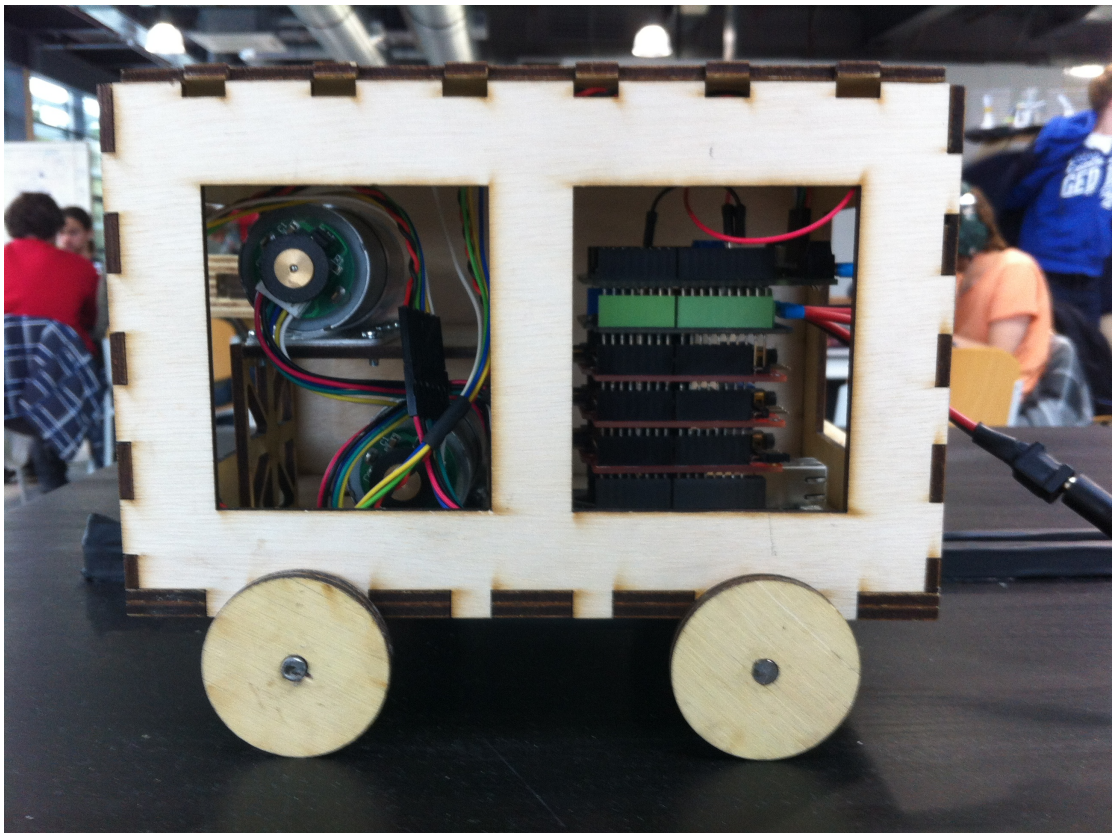

Figure 12: Side-view of constructed model

### 4.3.3 Filter Application

In the graph below 4 signals can be seen.


Figure 13: EMG signals

Channel 0 shows the raw EMG signal that is acquired with the electrodes. Channel 1 shows the signal after the high pass filter and the notch filter have been applied, already making the signal a lot more recognizable. In channel 2 the signal has been rectified and the peak gain at 1Hz has been applied and in channel 3 an envelope is put over that signal with the low pass filter.
This filtered signal alone would not be enough to determine if an action should be taken or not. Determining whether or not an action should be taken is done by first calibrating. By starting a calibration we can check for both the biceps and one other muscle, either the triceps or the extensor muscles in the lower arm, what the maximum value is that can be achieved by contraction. After this we express a percentage of this maximum contraction as the line where an action should be taken. In theory several lines could be taken above which different actions can be taken however this was not to successfully control the robot and as such was not implemented. With three EMG signals we can now give the minimum of 4 different signals to the controller.

### 4.3.4 Control

While testing the mechanics of scooping it was observed that that the overshoot after turning off the motor can be nullified by quickly turning the motor the other way for a brief period of time before turning off. Using this technique made the PID controller obsolete as this gave a solution to the problems the PID controller would fix without is cons such as the chance of getting an unstable system. Both motors will use this method to reduces overshoot. Thus the control used by both motors will be the person operating in combination with the software that quickly makes the motor stop.

### 4.3.5 Programming

This is the first project for which we have used C++ to create a program, and it is also our first experience with embedded programming. We split up the programming part into engine control and EMG filtering, and subsequently put these parts together.

**Pulse Width Modulation**

The motor directions are determined by the value of a digital pin (1 or 0). On the top of the script pins 4 and 7 are defined to be m1direction for motor 1 and m2direction for motor 2 using DigitalOut. The power to the motor is provided using a separate power source, that receives power when another digital pin becomes 1. We have defined the digital 5- and 6 pins to power the motors. These pins support Pulsed Width Modulation (PWM). Using PWM we can set the (average) power through a digital pin by altering between 1 and 0 . The Duty Cycle (the percentage of a period that the PWM pin is 0) was set by writing the value to the motor's respective PWM name (motorpwm1 and motorpwm2 respectively). We used the buttons on the board for testing this code before adding it to the EMG-related code.

**Encoder**

The motor has a quadrature encoder which can be read out using the QEI library. The encoder function can record the amount of pulses transmitted by the encoder using the getPulses() function. We use the encoder to get the position of the arm (provided the program starts with the arm in the starting position). The encoder is essential for the spatula-arm, as the robot can destroy itself when the arm moves into the structure. However, for the translational movement, the encoder does not deliver much added value, as the burger will not be in a predetermined position after the first flip, and the robot is protected from running off the table by the rails that bound it.

**Stopping**

We found during testing that the motors do not instantly stop when the signal goes to zero. This means that especially for the spatula, we need to take two safety zones into account (at the end of a scoop and with the spatula held up to the highest point), in which an inadvertent muscle input cannot accidentally destroy the robot. We defined two constants to check if the spatula is in any such region and prevent the motors from activating into the dangerous direction if this is the case. The highest spatula position is chosen as the zero count.

To assist the motor in stopping quickly, and to reduce lengths of the safety-distances, we created a function superstop() that helps the selected motor to slow down more quickly. It is based on our observation that a fully spinning motor can still reverse practically immediately after cutting the power. The "superstop" function quickly alternates the

direction of the motor right after cutting power.

### 4.3.6 Combining Control and Filter

The EMG part was made using the multicolor LED as output for testing purposes. The program compares the filtered measured signal with the highest filtered signal encountered during calibration. If it surpasses a certain percentage of that maximum, a light associated with that sensor (Red for right biceps, blue for the left biceps, and green for the forearm) is activated.
The activation part of our code was added to the parts where the green and blue colors would activate. Red signal (right biceps) was used for switching between the motor for the spatula and for the horizontal movement (by making variable modesw one or zero respectively). In order to prevent a one second contraction to switch the modesw 1000 times, we created a counter (timercount) that counts with every iteration of the function from a 1000 to 0, and resets to 1000 if red signal is received (as well as switching modesw between one and zero) the other two were used to activate the selected motor into a specific direction (per sensor respectively). If no signal surpassed the threshold, superstop() is executed to ensure the motor stops as soon as possible. The full code can be viewed at appendix I.

# 5 Evaluation

## 5.1 Technical evaluations

The table below goes through the system specifications and evaluates them one by one. As can be seen, most of the system specifications are fulfilled. The table has been separated in two parts.

| Specifications | Evaluation |
|---|---|
| Robot needs to be able to flip a burger weighing minimum 30 grams and up to 120.5 grams and possibly more (size of McDonald's burgers). | Two models of burgers were created - one weighing 35g, the other weighing 38g. The robot was able to flip both of the burger models, either one at a time or both at once. |
| Burger needs to land with a 180-degree flip done (to cook other side). | The robot was able to successfully flip the burger at a success rate of around eighty percent. For the other twenty percent when the robot fails at flipping the burger, the reason is because the burger was not properly scooped up onto the spatula due to the bent material (as discussed in the realization section). Sometimes, due to the wooden material that the model is made out of, the burger model would bounce against the table and perform a double flip. |
| Burger must not be thrown away (due to flip). | Flipping does not result in the burger being thrown away. |
| The spatula needs to be easily detachable (for cleaning). | The spatula was made to be detachable and doing so did not affect the stability of the robot. |
| Robot needs to be able to flip a burger more than once. | With the stringed system, at a certain point, both the spatula arm and holding arm are lifted above the table surface. The space at which these arm hovers over the table is just enough for the robot to move over the burgers, therefore, allowing for the ability to choose which burger to flip, as well as the possibility of flipping a burger multiple times. |
| Robot needs to flip either one or two burgers at a time | With the size of the spatula, the robot can flip either one or two burgers at once. Even though the number of burgers that the robot is flipping does change the stability of the robot, the size of the spatula has already contributed to making the robot less stable. The design of the spatula could have been designed with better support, as right now it is attached and held up at only one end. |

Figure 14: Specifications and Evaluations

| | |
|---|---|
| Robot must be able to reach any point of a plate of 30 cm by 60cm | The robot was simulated on a presentation table in which the spatula covers an area of 25 cm by 97 cm. This means that the robot would satisfy the length of the plate, but it does not satisfy the width of the plate. The robot would not be able to reach some of the points on the plate and therefore the spatula would need to be made longer. Although, the extra space the robot can cover across the length of the plate, could allow for the removal of burgers from the hot plate onto a serving plate. |
| As much of the weight in the base as possible | Even though from the SolidWorks design, it may look like most of the weight is in the front due to the size of the spatula, most of the weight is still kept in the base due to the weight of the motors, controllers, and shields. |
| Robot is designed for burgers on a hot plate | With the current material of plywood, the robot would not satisfy this requirement and would burn down. To satisfy this requirement, the materials would need to be made more suitable for working with heat. |
| Placement of the robot must be on the tabletop; not on the wheelchair, for safety | The robot is placed on a tabletop and not on the wheelchair. With rails added as well, the safety is more than ensured. This also allows for the mobility of the robot, as well as that of the user. Although the user must make sure that he is moving along with the robot, or longer wires should be used. |
| Robot must be able to move the burger out of the grilling plate onto a serving plate | This is possible, as the range of motion that the robot is capable of is longer that the specified length of the grilling plate. Although, a special serving tray would have to be used, as there is no precise control for the landing of the flipped burger. |
| The robot must make use of 2 motors. | Two motors are incorporated into the robot to allow the robot to move in a 2D plane. One motor is used to move the cart from side to side, while the other motor is used to rotate the spatula arm. |
| The robot should be controlled by 2,3 or 4 EMG inputs | Three EMG inputs are used to control the robot. There is input from the bicep of both the left and right arm. The third input may come from either a triceps or forearm. The biceps are used to control the rotation (clockwise or anti-clockwise) of the motor and the third input is used to control which motor will be used (for rotating arm or for moving cart). The speed of the motor is fixed. |

Figure 15: Specifications and Evaluations pt.2

### 5.2 Performance evaluation

For physical manual labor work, the robot should be able to perform very well, as it can flip burgers in batches. Along with the ability to control which row of burger needs to be flipped or re-flipped, the robot allows for the user to work efficiently. The small size of the robot also allows for mobility and it can be easily moved to be used in multiple settings.
Due to the batching process and the lack of precision when flipping the burger, the robot cannot yet properly put the burgers onto a plate. Another disadvantage due to batching is that it controls flipping of burgers over rows rather than one by one. This means that when flipping a row of burgers, some burgers may not be ready yet.

## 6 Conclusions

### 6.1 Suitability for Targeted Challenge

The robot can be considered to be suitable for people with Duchenne, although supervision during use is recommended.
The controls for this robot is very simple and intuitive. Even so, the user would have to learn how to use the robot properly and become accustomed to its controls, as there are slight delays. Although the delay is very small, timing is very important for the burger to be properly flipped. The delay should not be a large problem as there is a large margin of error for the scooping, thus making it easy to scoop even with the slight delay.
In a real situation, the user would most likely need assistance in setting up the robot and the grilling plate. Assistance would also be needed in cleaning and turning off the grilling plate when done. Supervision is also recommended as the robot is moving separately from the user, while both the robot and the user are connected via wires. For safety, supervision could help ensure that wires do not get tangled.

### 6.2 Recommendations

As this robot is only a prototype, there are many points of improvement, from materials to design. This section discusses the things that could be done to upgrade the robot.
The design could be altered slightly to provide more support for the spatula and the holding arm. Doing so could also provide the option to make the spatula larger, allowing for more burgers to be flipped. For the robot to be used in performing a manual labor job, this is an efficient option. The increase in support and the change to a more durable and suitable material could also make the robot more time-enduring.
To increase the mobility of the robot and to ensure safety, a wireless system could be implemented for the cart.
Aside from changes to the robot, a system could be implemented into the environment in which the robot will be used. This system should allow for an efficient way to serve the burger patties on buns. The system for turning the grilling plate on and off should also be made suitable for people with Duchenne.

# 7 References

1.DA, Duchenne Muscular Dystrophy, 2007 https://www.mda.org/disease/duchenne-muscular-dystrophy

2.limex Ltd., Shield-ekg-emg bio-feedback shield user?s manual, 2011, https://www.olimex.com/Products/Duino/Shields/SHIELD-EKG-EMG/resources/SHIELD-EKG-EMG.pdf

3..Grosse, M.J. Cassudt, P.Brown, EEG-EMG, MEG-EMG and EMG-EMG frequency analysis: physiological principles and clinical applications, 2011, http://homes.mpimf heidelberg.mpg.de/ mhelmsta/pdf/2002

4.ololu Corporation, Pololu Corporation,last seen (8-11-2016), 131:1 Metal Gearmotor 37Dx57L mm with 64 CPR Encoder, 2016, https://www.pololu.com/product/1447/specs

# 8 Appendices

## 8.1 Appendix A

### 8.1.1 Targeted Challenge

Below is the brainstorming done to come up with a specific task for the robot that is to be built in this project. Following the brainstorm session, each group member had to vote for two of their favorite concepts. The concept with the most vote would then be the specified task for the robot that is to be built for this project.

END-TARGET: professional job that requires manual labor.

- Ice-cream man - scoop ice cream
- Cafeteria person - serving scoops of food
- Chain restaurant fryer
- Table cleaner 2
- Cashier
- Watering plants - gardener 2
- Cake decorator
- Mailman
- Sorting papers - office job
- **Burger flippers 3**
- Portrait Photographer 1
- Fruit weigher 2
- The muur - wall organizer

## 8.2 Appendix B

### 8.2.1 Morphological Chart

| Task | Solutions | | | |
|---|---|---|---|---|
| **Moving on the plate** | Moving Robot | Moving 2 arms (base stays still) | Move 1 arm, Base rotates | - |
| **Put burger on spatula** | Movement of spatula | Stopping arm on side | Stopping arm that comes down | Wall |
| **Flip burger** | Rotating arm | Rotating spatula | Upward moving spatula | Upward moving arm |
| **Motor placement** | Base | 1 base, 1 an arm | - | - |
| **Maintain stability and supports structure** | Rail | Add extra weights | Put most mass in the base | - |
| **Amount of EMGs** | 2 | 3 | 4 | - |
| **Filter type** | Low pass | High pass | Notch | Peak gain |
| **Controller** | P | PD | PID | PI |

Figure 16: Specifications and Evaluations

## 8.3 Appendix C

### 8.3.1 Concept Evaluation

To evaluate the three concepts and decide on one for the final design, each concept was evaluated by each requirement set for the robot. Each requirement has different weightings, depending on its importance. The weightings can be found in bolded numbers in front of the requirements, as can be seen below. Following the list, the table shows the scoring chart for each of the concept designs.

**Requirements for design burger flipping (Bolded numbers = weighting):**

- **2** Needs to be able to flip a burger weighing minimum 30 grams and up to 120.5 grams and possibly more (size of mcdonald's burgers).
- **3** Burger needs to land with a 180 degree flip done (to cook other side).
- **3** Burger must not be thrown away (due to flip).
- **1** The spatula needs to be easily detachable (for cleaning).
- **2** Needs to be able to flip a burger more than once.
- **2** Needs to flip either one or two burger at a time If more, there could be problems with uneven distribution of heat on hot plate
- **2** Needs to reach any point of a plate of 30 cm by 60cm which is a reasonable size to fit two burgers at once.
- **3** As much of the weight in the base as possible
- **3** Designed for burgers on hot plate
- **3** Placement on the tabletop, not the wheelchair for safety
- **2** Must be able to move the burger out of the grilling plate onto a serving plate

Table 4: Concept evaluation

| **Reqs (xMultiplier)** | **Design 1** | **Design 2** | **Design 3** |
|---|---|---|---|
| **1 (x2)** | 10 | 10 | 10 |
| **2 (x3)** | 15 | 15 | 15 |
| **3 (x3)** | 15 | 6 | 12 (+1) |
| **4 (x1)** | 3 | 5 | 4 |
| **5 (x2)** | 10 | 10 | 10 |
| **6 (x2)** | 6 | 6 | 8 |
| **7 (x2)** | 10 | 10 | 10 |
| **8 (x3)** | 12 | 15 | 12 |
| **9 (x3)** | 15 | 15 | 15 |
| **10 (x3)** | 15 | 3 | 15 |
| **11 (x2)** | 10 | 10 | 10 |
| **Final score** | **121** | **105** | **122** |

## 8.4 Appendix D

### 8.4.1 SolidWorks Models

Here we include all of our screen-shots of our SolidWorks model. To provide the best view possible for our model, there are different angles of the view included.

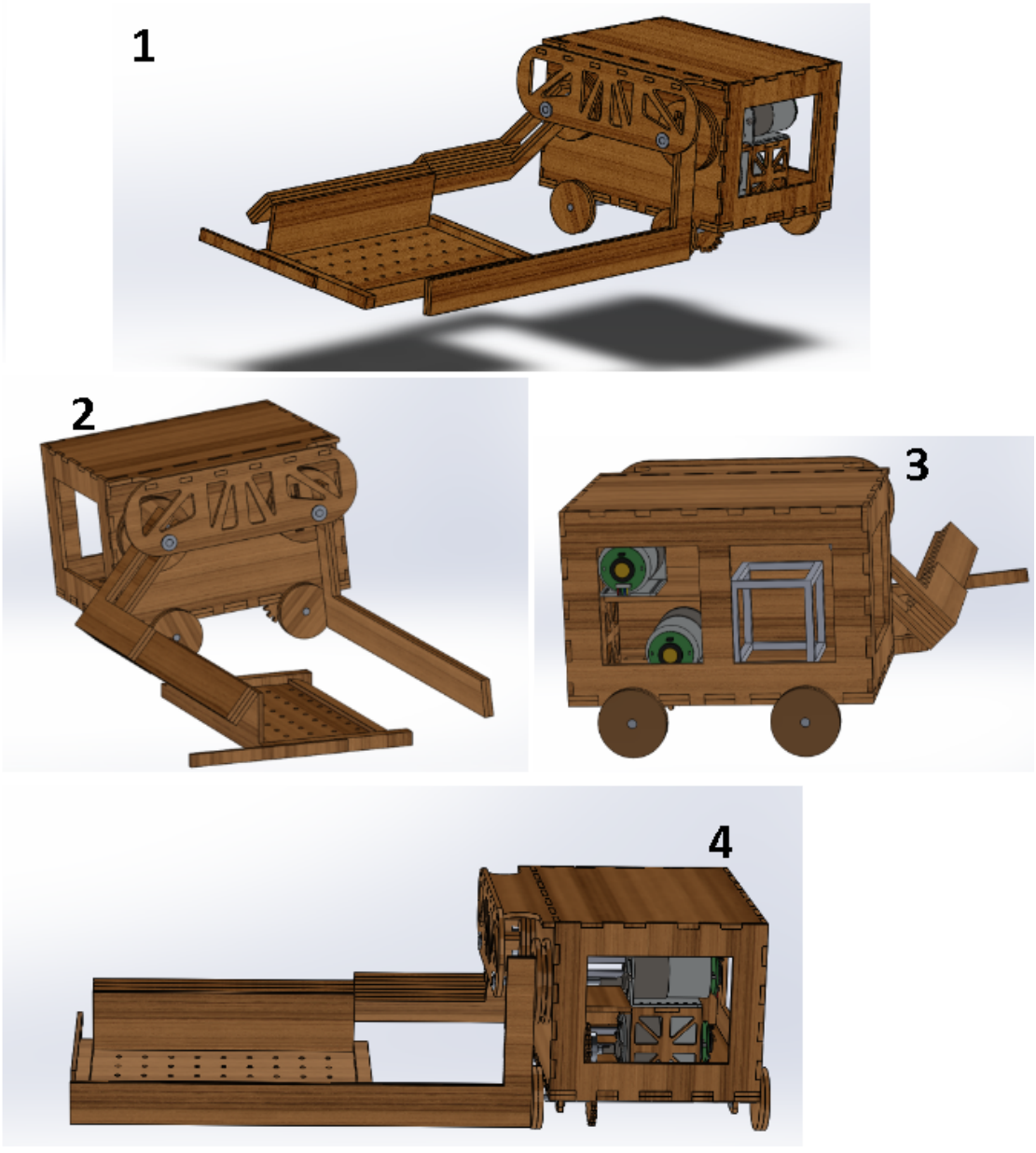


Figure 17: Different views of the SolidWorks model

## 8.5 Appendix E

### 8.5.1 Finding Inertia

The inertia on the motors is the sum of the motor inertia and the inertia of everything moved by the motor. The inertia where found using Solidworks. $J_{Base} = (5.95e-04)Kgm^2$ $J_{Spatula} = (2.08e-04)Kgm^2$. An estimation was made for the inertia of the motor taking its size and weight into account $J_{motor} = (3.0e-0.8)Kgm^2$. As the inertia of the of the base and spatula are after the gears it will need to be multiplied by $\frac{1}{131^2}$.
This gives the final inertia values of
$J_{basetotal} = J_{motor} + \frac{J_{Base}}{131^2} = (6.4672e-08)Kgm^2$
$J_{spatulatotal} = J_{motor} + \frac{J_{spatula}}{131^2} = (3.4662e-08)Kgm^2$

### 8.5.2 Finding motor value

To find the motor constant by dividing the torque over current. To be able to do this the current and torque need to be converted to the following SI units, N-m and A respectively. This gives a motor constant, $K_m$ of 0.4.
To find the resistance of the motor the voltage should be divided by the stall current. This shows that the resistance in the motor,R, is 2.4Ω 2

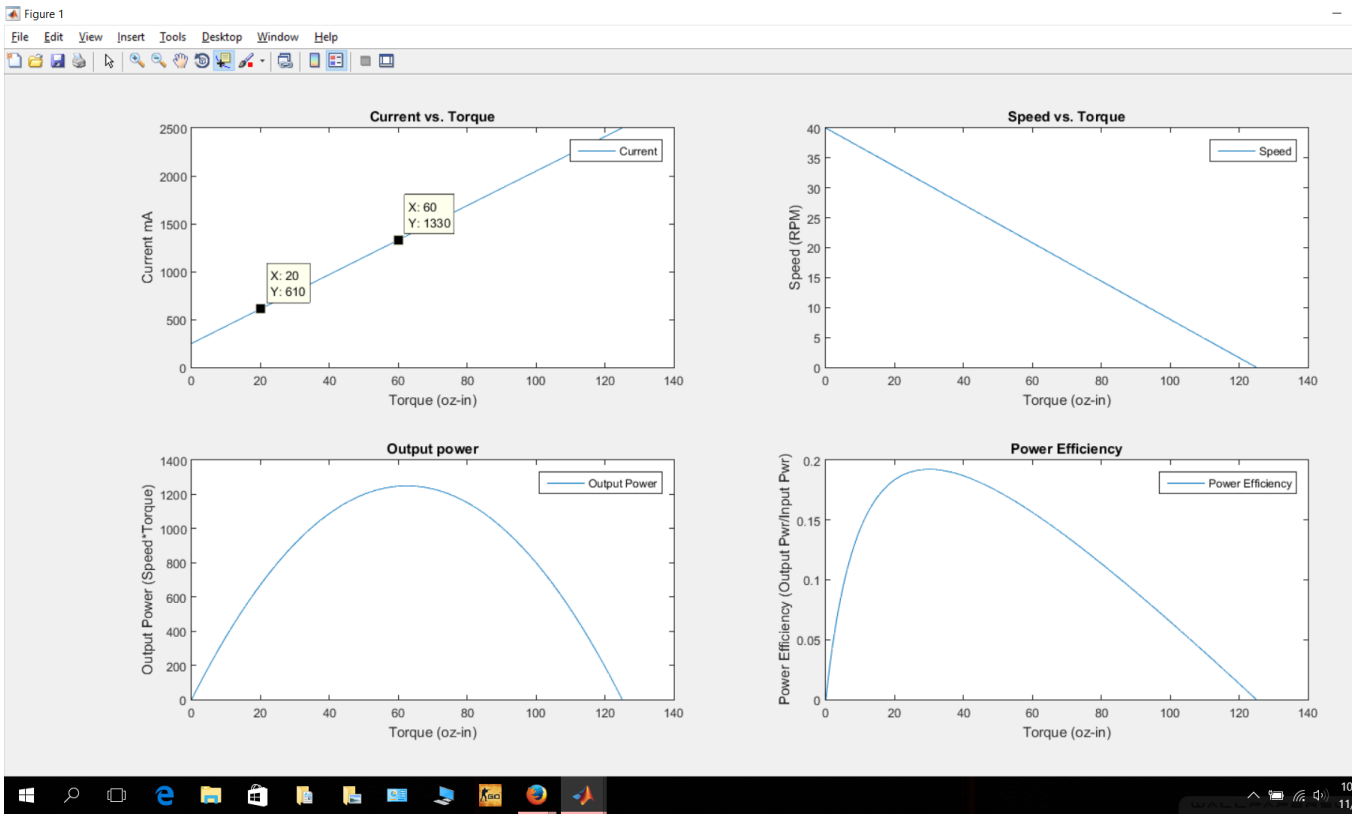


Figure 18: Behavior of the motor

## 8.6 Appendix F

### 8.6.1 Tuning rules

To find the values for the PID transfer function(k(s)) the tuning rules need to be followed.

$$K(s) = \frac{P(sT_Z + 1)(sT_I)}{sT_I(sT_P + 1)} \tag{9}$$

$$T_Z = \frac{\sqrt{\frac{1}{\alpha}}}{\omega_c} \tag{10}$$

$$T_P = \alpha T_Z \tag{11}$$

$$T_I = \beta T_Z \tag{12}$$

$$K_P = \frac{m_{eq}\omega_c^2}{\sqrt{\frac{1}{\alpha}}} \tag{13}$$

Where $\alpha = 0.1$, $\beta = 0.2$ and $m_{eq} = \frac{J_{tot}R}{K_m}$ [**?**]

### 8.6.2 Converting transfer function

Converting the transfer function values of the PID into PID controller values that the code can read. To do this both equations need to be of the same form.

Transfer function PID:

$$K(s) = \frac{P(sT_Z + 1)(sT_I)}{sT_I(sT_P + 1)} \tag{14}$$

PID controller:

$$C(s) = K_P + \frac{K_I}{s} + \frac{K_D s}{s\tau + 1} \tag{15}$$

Rewriting the PID controller such that the whole equation is in one fraction:

$$C(s) = \frac{K_P\tau s + K_P}{s\tau + 1} + \frac{K_I\tau + \frac{K_I}{s}}{s\tau + 1} + \frac{K_D s}{s\tau + 1} \tag{16}$$

This formula can be rearranged to:

$$C(s) = \frac{(K_P\tau + K_D)s + K_I\tau + K_P + \frac{K_I}{s}}{s\tau + 1} \tag{17}$$

Now the PID transfer function needs the same form as the rewritten control PID.

Transfer function PID:

$$K(s) = \frac{P(sT_Z + 1)(sT_I)}{sT_I(sT_P + 1)} \tag{18}$$

$$K(s) = \frac{T_Z T_I P s^2 + P(\frac{T_z}{T_I})s + P}{sT_I(sT_P + 1)} \tag{19}$$

Divide this by $sT_I$:

$$K(s) = \frac{T_Z P s + P(\frac{T_Z}{T_I}) + \frac{P}{T_I s}}{T_P + 1} \tag{20}$$

Now both equations now have the same form thus making it possible to compare them with each other. Comparing gives the following equations.

$$K_P \tau + K_D = T_Z P \tag{21}$$

$$K_I \tau + K_P = P(\frac{T_Z}{T_I} + 1) \tag{22}$$

$$K_I = \frac{P}{T_I} \tag{23}$$

$$\tau = T_P \tag{24}$$

Solving these equations for the unknowns in the PID controller gives:

$$K_P = \frac{T_z + T_I}{T_P} \tag{25}$$

$$K_I = \frac{P}{T_I} \tag{26}$$

$$K_D = \frac{T_Z P}{T_Z + T_I} \tag{27}$$

$$\tau = T_P \tag{28}$$

### 8.6.3 Step response and Bode plot

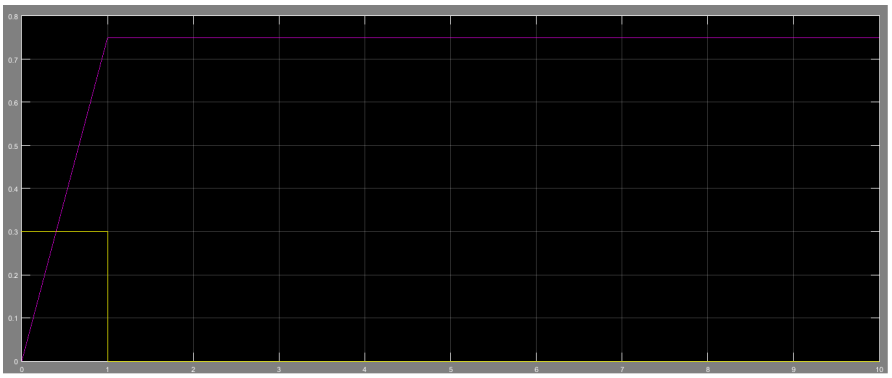

Figure 19: Voltage step response of the cart
yellow = input, pink = output

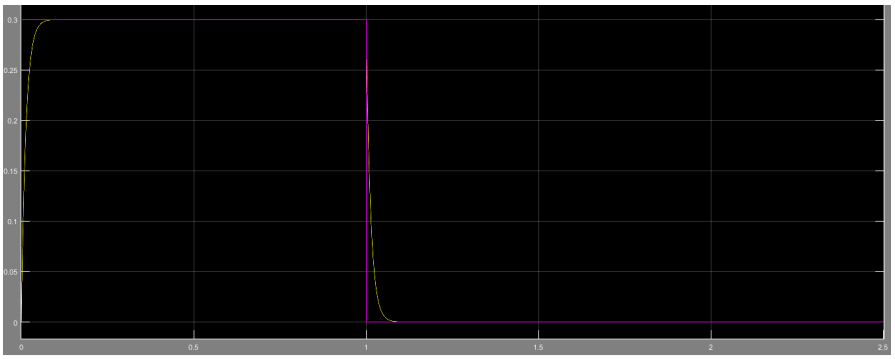

Figure 20: positional step response of the spatula
pink = input, yellow = output

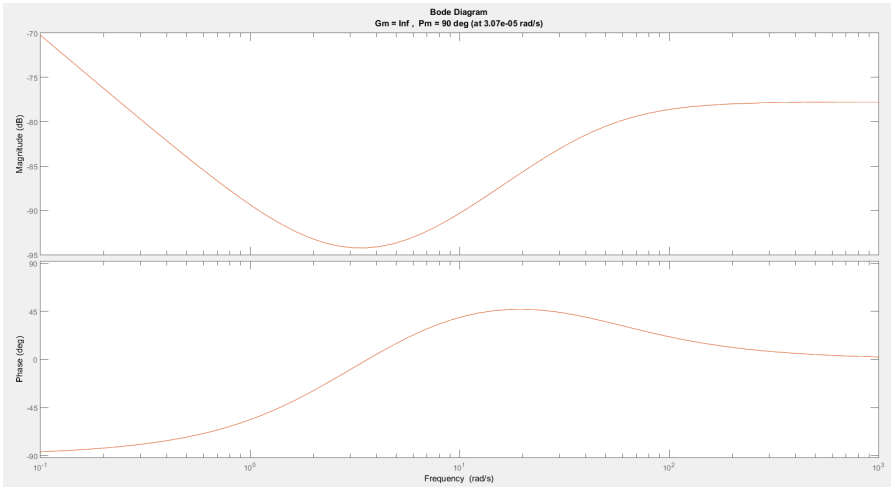


Figure 21: Openloop Bode plot of the cart

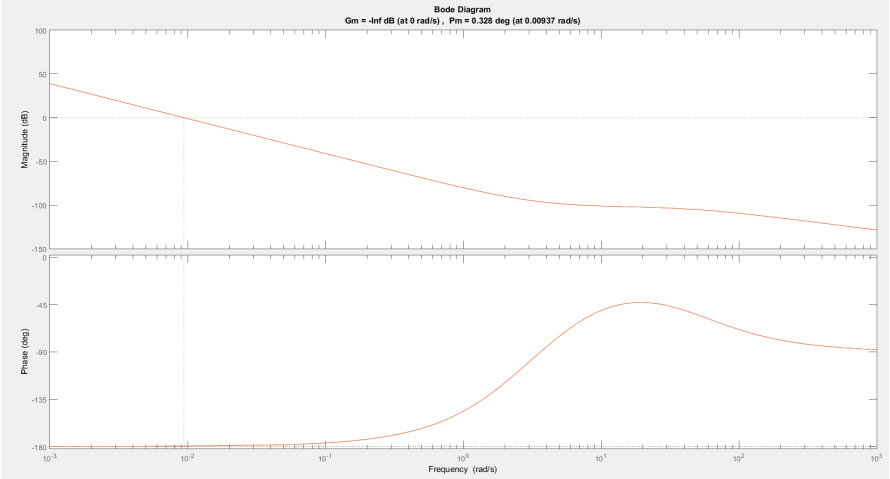


Figure 22: Open loop Bode plot of the spatula + PID controller

## 8.7 Appendix G

### 8.7.1 Laser Cutting

The following images show the files used for the laser cutting.

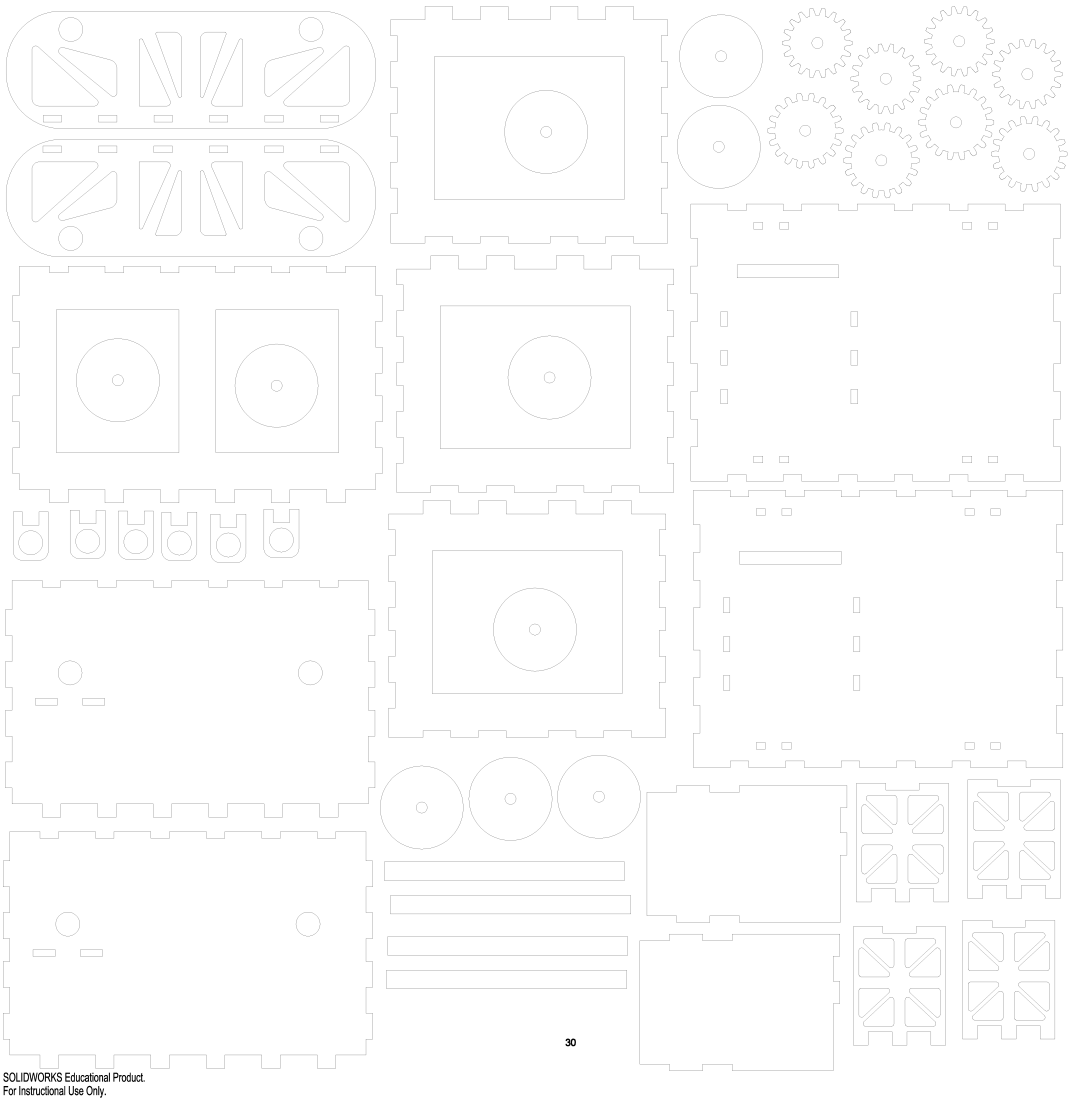


Figure 23: DFX files

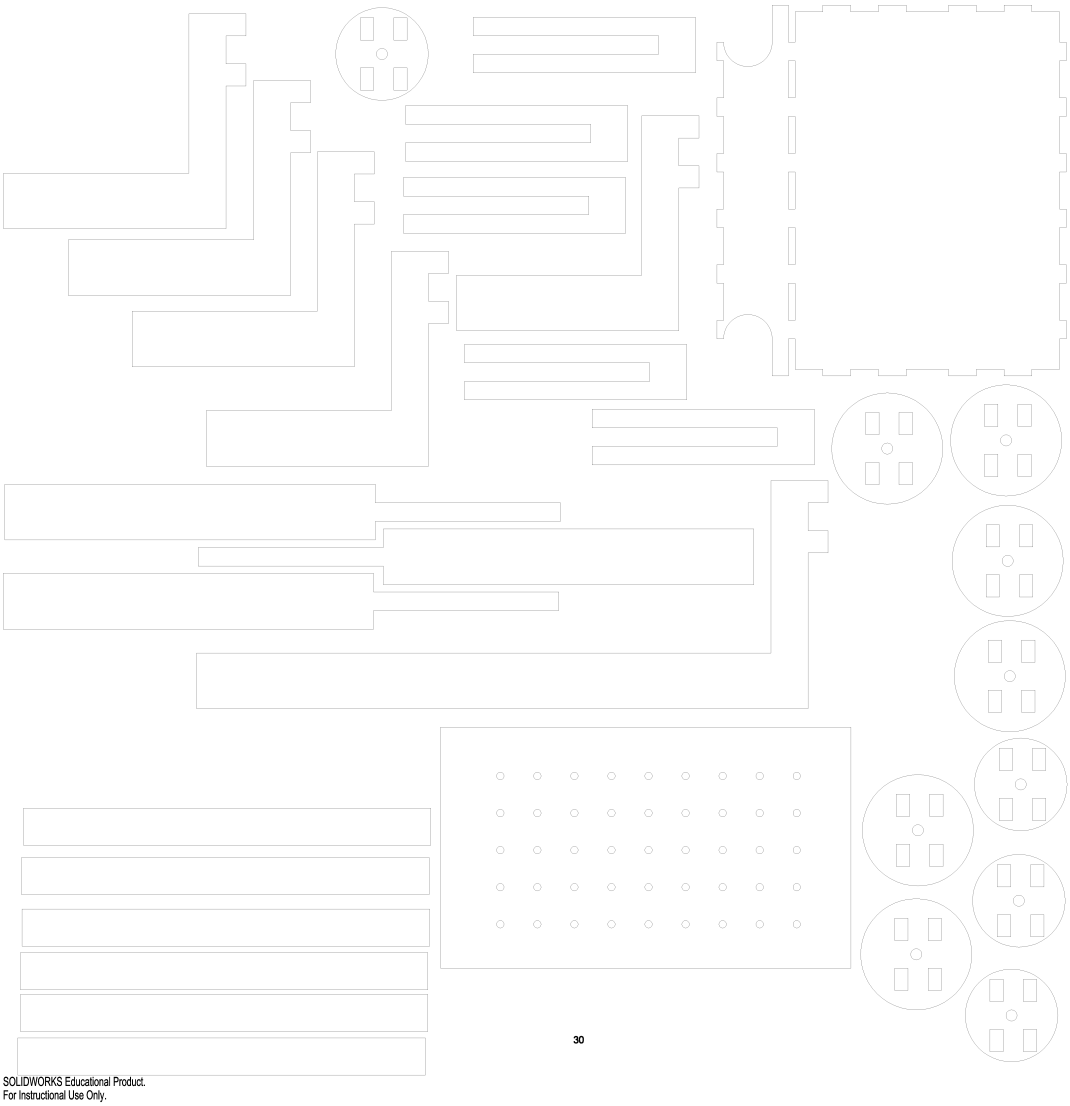


Figure 24: DFX files

## 8.8 Appendix H

### 8.8.1 Final Construction

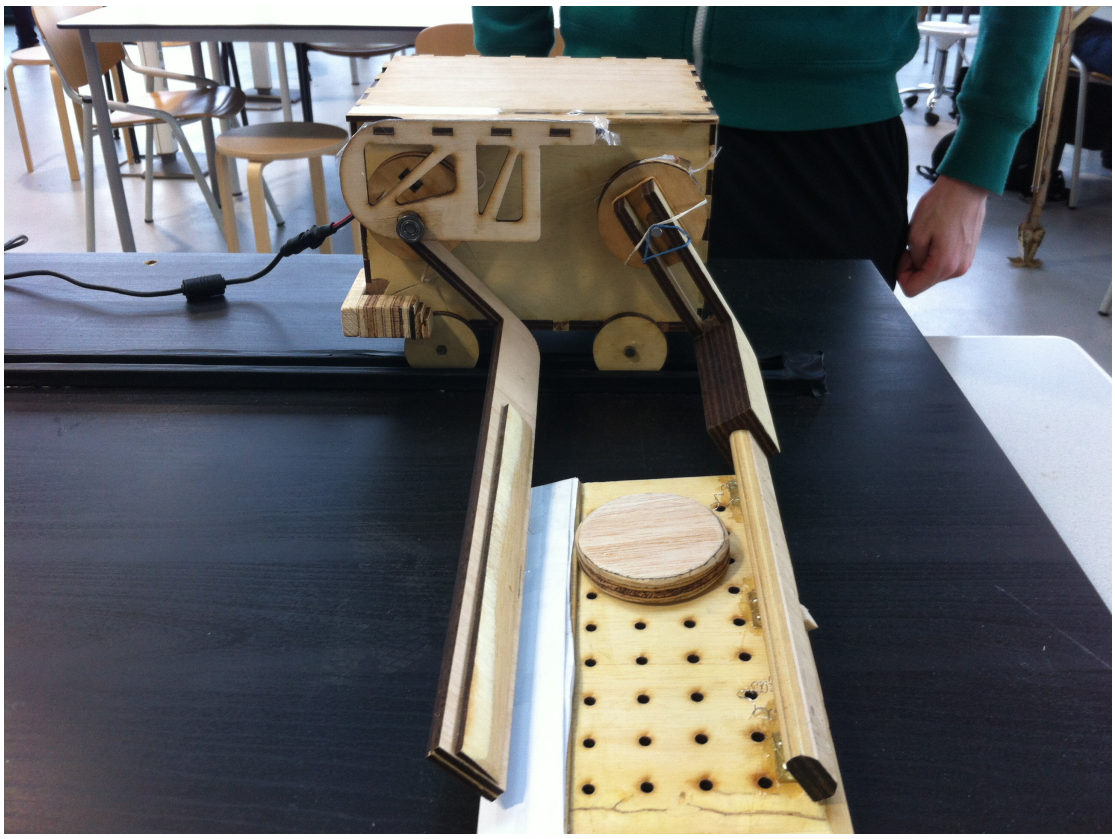

Figure 25: Picture robot

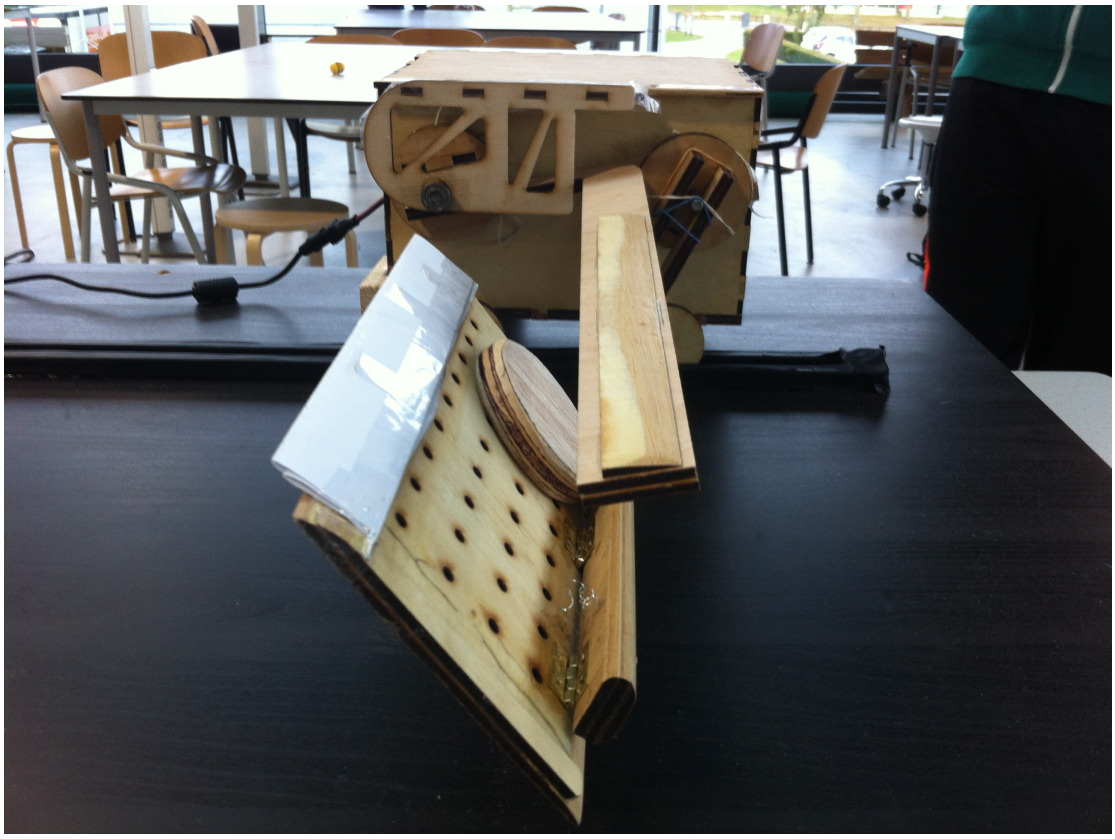

Figure 26: Picture robot

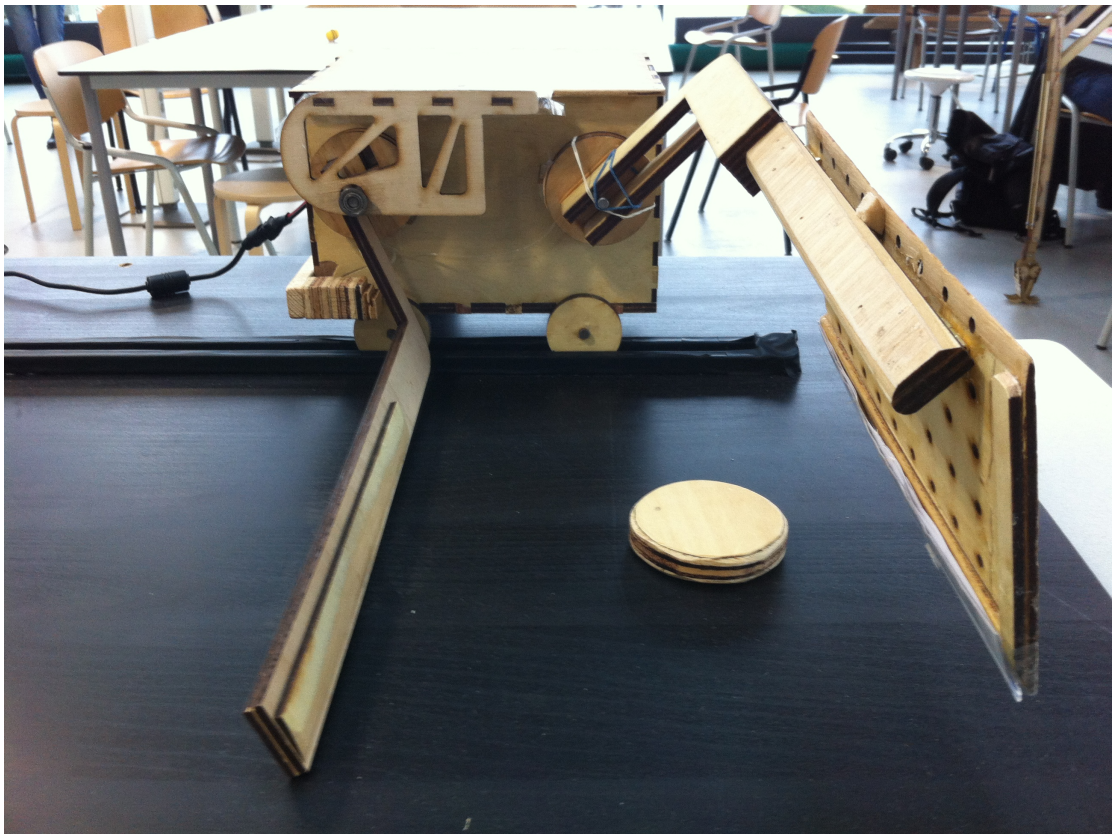

Figure 27: Picture robot

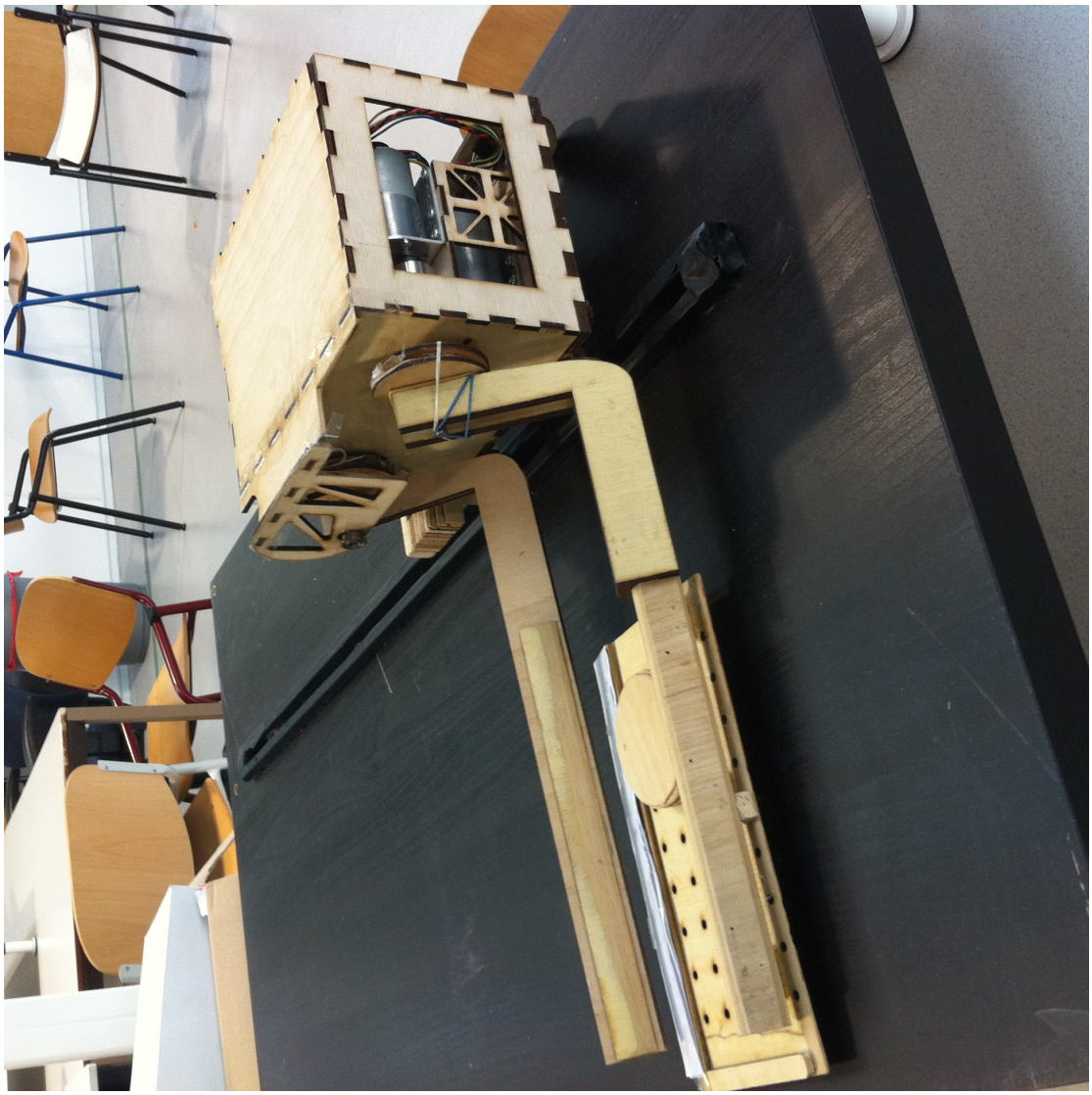

Figure 28: Picture robot

## 8.9 Appendix I

### 8.9.1 Code

```
#include "mbed.h"
#include "HIDScope.h"
#include "BiQuad.h"
#include "MODSERIAL.h"

#include "FastPWM.h"
#include "QEI.h"
float pwmprocent1 = 15;         // introduce duty cycle variable translational
motor
float pwmprocent2 = 15;         // introduce duty cycle for rotational motor (SLOW
DOWN TO PREVENT OVERSHOOT?)
int modesw;                     // introduce mode to switch from translational to
rotational control
FastPWM motorpwm1(D5);          // define PWM Pin
FastPWM motorpwm2(D6);          // define motor 2 pwm pin
float totalpwm1 = pwmprocent1/100;
float totalpwm2 = pwmprocent2/100;
int upperlimit = 600;           // spatula shouldn't move closer to top for safety
int lowerlimit = 2000;          // and shouldn't scoop an entire turn
int timercount = 0;             // initialize modesw condition
DigitalOut m1direction(D4);     // direction translational motor
DigitalOut m2direction(D7);     // direction

int count;
QEI Encoder(D12,D13,NC,32);

//Define objects
AnalogIn    emg1_in( A0 ); /* read out the signal */
AnalogIn    emg2_in( A1 );
AnalogIn    emg3_in( A2 );
DigitalIn   max_reader1( SW2 );
DigitalIn   max_reader3( SW3 );

Ticker      main_timer;
Ticker      max_read1;
Ticker      max_read3;
HIDScope    scope( 5 );
DigitalOut  red(LED_RED);
DigitalOut  blue(LED_BLUE);
DigitalOut  green(LED_GREEN);
MODSERIAL   pc(USBTX, USBRX);


// EMG variables
  //Right Biceps
  double emg1;
  double emg1highfilter;
  double emg1notchfilter;
  double emg1abs;
  double emg1lowfilter;
  double emg1peak;
  double maxpart1;
  // Left Biceps
  double emg2;
  double emg2highfilter;
  double emg2notchfilter;
  double emg2abs;
  double emg2lowfilter;
  double emg2peak;
```

Figure 29: Code Part 1

```
  double max1;
  double maxpart2;
  // Left Lower Arm OR Triceps
  double emg3;
  double emg3highfilter;
  double emg3notchfilter;
  double emg3abs;
  double emg3lowfilter;
  double emg3peak;
  double max3;
  double maxpart3;

// BiQuad Filter Settings
// Right Biceps
BiQuad filterhigh1(9.704e-01,-1.9408,9.704e-01,-1.9389,9.427e-01); /* High-pass
Filter at 10 Hz */
BiQuad filternotch1(9.495e-01,-1.8062,9.495e-01,-1.8062,8.992e-01); /* NotchFilter
at 50 Hz */
BiQuad filterpeak1(1.0033,-1.984,9.852e-01,-1.9838,9.8855e-01); /* 4dB Gain peak at
11 Hz */
BiQuad filterlow1(3.4869e-04,6.974e-04,3.4869e-04,-1.9616,9.630e-01); /* Low-pass
Filter at 6 Hz */
// Left Biceps
BiQuad filterhigh2(9.704e-01,-1.9408,9.704e-01,-1.9389,9.427e-01);
BiQuad filternotch2(9.495e-01,-1.8062,9.495e-01,-1.8062,8.992e-01);
BiQuad filterpeak2(1.0033,-1.984,9.852e-01,-1.9838,9.8855e-01);
BiQuad filterlow2(3.4869e-04,6.974e-04,3.4869e-04,-1.9616,9.630e-01);
// Left Lower Arm OR Triceps
BiQuad filterhigh3(9.704e-01,-1.9408,9.704e-01,-1.9389,9.427e-01);
BiQuad filternotch3(9.495e-01,-1.8062,9.495e-01,-1.8062,8.992e-01);
BiQuad filterpeak3(1.0033,-1.984,9.852e-01,-1.9838,9.8855e-01);
BiQuad filterlow3(3.4869e-04,6.974e-04,3.4869e-04,-1.9616,9.630e-01);
//

// stopfunction
 void superstop()
{
    if (modesw==0)
    {
         motorpwm1.write(0);
         for (int a = 0; a < pwmprocent1; a = a + 1)
         {
             if (m1direction = 0)
             {
                 m1direction = 1;
             }
             else
             {
                 m1direction = 0;
             }
         }
     }
     else
     {
         motorpwm2.write(0);
         for (int a = 0; a < pwmprocent2; a = a+1)
         {
             if (m2direction = 0)
             {
```

Figure 30: Code Part 2

```
                m2direction = 1;
            }
            else
            {
                m2direction = 0;
            }
        }
    }
}


// Finding max values for correct motor switch if the button is pressed
void get_max1(){
    if (max_reader1==0){
        green = !green;
        red = 1;
        blue = 1;

        for(int n=0;n<2000;n++){  /* measure 2000 samples and filter it */
          emg1 = emg1_in.read();      /* read out emg */
          emg1highfilter = filterhigh1.step(emg1); /* high pass filtered */
          emg1notchfilter = filternotch1.step(emg1highfilter); /* notch filtered */
          emg1abs = fabs(emg1notchfilter); /* take the absolute value */
          emg1peak = filterpeak1.step(emg1abs); /* 4dB gain peak */
          emg1lowfilter = filterlow1.step(emg1peak); /* low pass filtered */

          if (max1<emg1lowfilter){
              max1 = emg1lowfilter; /* set the max value at the highest measured
value */
          }
          wait(0.001f); /* measure at 1000Hz */
        }
        green = 1;
    }
    maxpart1 = 0.25*max1; /* set cut off voltage at 25% of max for right biceps */
    maxpart2 = 0.25*max1; /* set cut off votage at 15% of max for left biceps */
}

void get_max3(){
    if (max_reader3==0){
        green = 1;
        blue = 1;
        red = !red;
        for(int n=0;n<2000;n++){

          emg3 = emg3_in.read();
          emg3highfilter = filterhigh3.step(emg3);
          emg3notchfilter = filternotch3.step(emg3highfilter);
          emg3abs = fabs(emg3notchfilter);
          emg3peak = filterpeak3.step(emg3abs);
          emg3lowfilter = filterlow3.step(emg3peak);

          if (max3<emg3lowfilter){
              max3 = emg3lowfilter; /* set the max value at the highest measured
value */
          }
          wait(0.001f);
        }
        red = 1;
```

Figure 31: Code Part 3

```
    }
    maxpart3 = 0.35*max3; /* set cut off voltage at 25% of max for forearm */
}

// Filtering & Scope
void filter() {
    timercount--; //// timercount = timercount - 1;

    // Right Biceps
    emg1 = emg1_in.read();
    emg1highfilter = filterhigh1.step(emg1);
    emg1notchfilter = filternotch1.step(emg1highfilter);
    emg1abs = fabs(emg1notchfilter);
    emg1peak = filterpeak1.step(emg1abs);
    emg1lowfilter = filterlow1.step(emg1peak); /* Final Right Biceps values to be
sent */
    // Left Biceps
    emg2 = emg2_in.read();
    emg2highfilter = filterhigh2.step(emg2);
    emg2notchfilter = filternotch2.step(emg2highfilter);
    emg2abs = fabs(emg2notchfilter);
    emg2peak = filterpeak2.step(emg2abs);
    emg2lowfilter = filterlow2.step(emg2peak); /* Final Left Biceps values to be
sent */
    // Left Lower Arm OR Triceps
    emg3 = emg3_in.read();
    emg3highfilter = filterhigh3.step(emg3);
    emg3notchfilter = filternotch3.step(emg3highfilter);
    emg3abs = fabs(emg3notchfilter);
    emg3peak = filterpeak3.step(emg3abs);
    emg3lowfilter = filterlow3.step(emg3peak); /* Final Lower Arm values to be sent
*/


    /* Compare measurement to the calibrated value to decide actions */
    if (maxpart1<emg1lowfilter && timercount < 500){ /* See if right biceps is
contracting */
        red = 0;
        blue = 1;
        green = 1;
        if (timercount==0)
        {
            timercount = 1000;

            switch (modesw)         // next few lines switch modesw from 0 to 1 and
vice versa for switching motor
            {
                case 0:
                {
                    modesw = 1;

                    break;
                }
                case 1:
                {
                    modesw = 0;
                    break;
                }
            }
```

Figure 32: Code Part 4

```
        }

    }
    else if (maxpart2<emg2lowfilter){ /* See if left biceps is contracting */
        red = 1;
        blue = 0;
        green = 1;

        switch (modesw)
        {
            case 0:
            {
                m1direction = 1;                    // direction of motor 1
                motorpwm1.write(totalpwm1);         // speed of motor 1 in that
direction
                break;
            }
            case 1:
            {
                count = Encoder.getPulses();        // count encoder pulses
                m2direction = 1;                    // direction motor 2
                if (count<lowerlimit)               // safety distance
                {
                    motorpwm2.write(totalpwm2);     // motor 2 speed in that
direction
                }
                break;
            }
        }
    }

    else if (maxpart3<emg3lowfilter && timercount<500){ /* See if lower arm is
contracting */
        red = 1;
        blue = 1;
        green = 0;

        switch (modesw)
        {
            case 0:
            {
                m1direction = 0;
                motorpwm1.write(totalpwm1);
                break;
            }
            case 1:
            {
                if (count>upperlimit)           // safety distance
                {
                    m1direction = 0;
                    motorpwm2.write(totalpwm1);
                }
            break;
            }
        }
    }

    else {
        red = 1; /* Shut down all led colors if no movement is registered */
```

Figure 33: Code Part 5

```
        blue = 1;
        green = 1;
        superstop();                    // stop motor quick
    }
    /* Set the sampled emg values in channel 0 (the first channel) and 1 (the
second channel) in the 'HIDScope' instance named 'scope' */
    scope.set(0, emg1lowfilter ); /* plot Right biceps voltage */
    scope.set(1, emg2lowfilter ); /* Plot Left biceps voltage */
    scope.set(2, maxpart1 ); /* Show the line above which the motor should run for
right biceps */
    scope.set(3, emg3lowfilter ); /* Plot Lower Arm voltage */
    scope.set(4, maxpart3 ); /* Plot the line above which the motor should run for
lower arm */

    scope.send(); /* send everything to the HID scope */
}

int main(){

    main_timer.attach(&filter, 0.001); /* set frequency for the filters at 1000Hz
*/
    max_read1.attach(&get_max1, 0.2); /* set the frequency of the calibration loop
at 5Hz */
    max_read3.attach(&get_max3, 0.2);
    while(1) {}
}
```

Figure 34: Code Part 6