METALHEAD ROBOTICS

Niels van der Zijden
Carmen Patrus
5HA

Table of contents

Preface:
We are two students, Niels and Carmen from 5 havo. For our research thesis we've decided to construct a headbanging robot. We both want to gain more experience and knowledge in coding and electrical engineering and decided that this project would be a good challenge for us to learn from. We are both very enthusiastic about software engineering and electrical engineering and are planning on pursuing these fields. This project will be very practical, as we are actually constructing a functioning robot. We are both very excited to see the outcome of our project.

Hello world…

Throughout this paper we will be going through the process of designing and building our very own headbanging robot. Through the use of hobby electronics and four months of labor we are hoping to achieve this goal. We will be thoroughly researching every decision we'll be making along the way. Including the design of the robot and the circuit that will power everything.

The main challenge will be designing and constructing the robot, and see what we can achieve with hobby electronics, compared to expensive professional parts. One of the greatest goals of this robot is to be able to perform a variety of tasks, including, but not limiting to headbanging and showing a "sign of the horns", a hand gesture commonly used by fans of rock and metal genres.

Now, this will not be the first robot ever made through the use of hobby electronics, but it will be our first. This allows us to see what really goes into making one from scratch. To do this, we will have to find out how these servos work, and how we can drive them. We'll have to think of ways to allow the robotic arm, which will make up most of this project, how to rotate, and move in such a way that it can do the things we want it to do, for example the sign of the horns.

The knowledge that we will gain throughout this, isn't limited to just a robot that is able to headbang, but can be applied to a whole array of applications of robotics. For example, in the medical field, complicated machines and robots get used daily, to provide patients with the best care that we can offer. The people that made these machines all had to start somewhere, and that is what this robot is for us. A way for us to learn and understand robotics on a very fundamental level.
The reason we're making this robot instead of an actual useful robot, is finding the motivation. Both of us quite enjoy music, and we wanted to make something that's applicable to our interests. But it's not limited to just our use of this robot. As it's made from open source software and hardware, it can be rebuilt and reprogrammed to do a variety of tasks outside of our scope for this project. Below you can see a quick overview of the goals we set out to achieve

In what way can we construct a headbanging robot?

In what way can we make a realistic moving arm through the use of servos?

- How do we make the arm rotate?
- How do we make the arm extend?

What is required to make the fingers move?

- Which mechanism will we use for the extension, and retention?

How do we make the head bop to preset music?

- Which mechanism will we use?

How do we allow a mobile app to control the robot remotely?

- What do we want to control through the app?
- How will we make this app?

How will we make all the moving parts work together as one?

- Which platform will we use to bind everything together? (e.g. Arduino)

Hypothesis

We are expecting a prolonged research in the beginning to make sure we are using the right base to build upon. This should help us kick it off in the right direction, especially with the parts that we are going to use to construct the robot with.

The majority of this research project will actually be practical, instead of theoretical. The end result of this project will be a physical and controllable robot. We are going to use books about electrical engineering like The Art of Electronics, online tutorials, and different forums dedicated to Electrical Engineering to help us out along the way.

The difference between stepper and servo motors

There are different types of motors that can be used in robotics. The most commonly used motors in robotics are servos and stepper motors. Servos and stepper motors can both deliver effective and reliable power for a successful operating system. Selecting between a servo motor and a stepper motor can be quite challenging as many different factors such as

the torque, the speed, the acceleration, the cost and the drive circuits, play a very important role in selecting the best motor for your operating system. We will be researching the differences between these motors to see which ones are the most effective for our application.

Stepper motors

Stepper motors consist of a stationary stator that carries the windings and a rotor with permanent magnets. A rotor is a moving component of the electromagnetic system as the stator is the stationary part of the machine.
A magnetic flux distribution is generated when current runs through the stator windings. Magnetic flux is a measure of the quantity of magnetic field passing through a surface. The magnetic flux distribution then proceeds to interact with the magnetic field distribution of the rotor to apply a turning force. Stepper motors have very high pole counts, which generally is 50 or more. Each pole is energized by the stepper motor driver in sequence so that the rotor turns in steps or in a series of increments. The motion appears to be continuous as the pole count is very high.

Stepper motors have a good amount of positive attributes. They are generally run open-loop, which means the rotor doesn't give positional feedback to the driver circuit. This eliminates the complexity and the costs of an encoder or resolver. The high pole count allows the stepper to generate very high torque at zero speed.

On the downside, stepper motors unfortunately have speed limitations. Stepper motors generally run best at 1200 revolutions per minute (RPM) or lower. Although stepper motors generate high torque at zero speed, the torque decreases as the speed increases. A motor that generates around 7,2 kg/cm at zero speed, for example, might only deliver 3,6 kg/cm at 500 RPM, and just 0,7 kg/cm at 1000 RPM. These are just theoretical examples, but they do show how efficiency and torque decreases as speed increases.

Another downside is the current draw. Because of the way off the shelf stepper motors are designed, they consume an enormous amount of power. Luckily, the more work a stepper motor has to do, the less current it draws. This also means that if it's barely trying to push any significant load, it actually means it consumes more power. Now knowing this, we realise this would be very counterintuitive, and a real pain to work with.

Besides the speed limitations, steppers also have performance limitations. A stepper could be thought of as a spring-mass system. Friction needs to be broken by the motor to begin turning and move the load, which at a given point, the rotor is not fully controlled. As a result, a command to promote by five steps might result in the stepper motor turning four steps, or even six. This problem can be overcome by using closed-loop stepper motors. These motors have an internal computer that relays the position of the rotor, allowing it to correct for mistakes and be more precise in general. Stepper motors do this through the use of half stepping. This means that multiple magnets in the motor are activated at the same time, allowing the shaft to be influenced by more than the magnets pushing or pulling it. But slightly pushing and pulling with different magnets, you can get a finer control, without having

to add any complex hardware, as it’s all done through the use of software. These closed-loop stepper motors are usually more expensive, as they improve on multiple factors from open-loop stepper motors. If you’re in the market for some new stepper motors, you should weigh both their strengths and weaknesses against each other.

Servo motors

Just like steppers, servos have many implementations. Let's consider the most common design, which consists of a stationary stator with its windings and a rotor with permanent magnets. The current yet again creates a magnetic field distribution that proceeds to interact with the magnetic field distribution of the rotor to develop torque. Servos have a significantly lower pole count than steppers. As a result of this, they must be run closed-loop.

Servos are generally more advanced than steppers. They run at a significantly faster speed than steppers. This makes servo motors capable of being used with gearboxes to deliver higher torque at more useful speeds. Servo motors also deliver more consistent torque across the speed range of the servo motor. They don't have holding torque in itself, unlike stepper motors.

The closed-loop operation activates the drive or controller to command that the load remains at a specific position. The servo motor will make continual adjustments to hold it in that specific position. Consequence of this, servos can deliver in fact holding torque. The zero-speed torque, however, depends on the servo being sized properly to control the load and avert oscillation about the specific location.

Servos generally use rare-earth magnets. Steppers on the other hand, use less expensive and more ordinary magnets more frequently. Rare-earth magnets that servos use kickstarted the development of higher torque in a smaller package. Servos also gain a torque advantage from their size as they are generally smaller in size. Consequence of these factors combined, servo motors have been recorded to deliver torques of up to 340 N/m. But this is the extreme end of things, and should in no way, shape or form be a goal we’d want to achieve.

The combination of the torque and the speed enables servos to deliver better acceleration than steppers. This is at a cost, as they use sprockets and gears to position themselves, compared to steps and half steps, or even quarter steps. These steps, done with permanent and electromagnets, can be more precise, but as stated before, have the issue of skipping or adding a step that the stepper wasn’t ordered to do. The gears of servos allow for much higher reliability, because of the gear box that is usually housed inside the servo motor.

Displayed here, the inside of a typical servo motor. As you can see, the lower part of the motor's housing is dedicated to the circuitry that's used to monitor the position of the rotor, which makes it a closed-loop system. Part of this circuitry is also dedicated to supply power to the DC motor inside the housing, which is used to power the gearbox, and in turn, the rotor.

Servos are usually made for and by hobby electronic engineers. This means that their soft- and hardware are usually open-source. Soft- and hardware that's open-source means that everyone and anyone can access anything that has to do with it. Open-source code means that all the code is free, and easily accessible for anyone that might want to use or change it. This allows for rapid prototyping by people all around the world. Open-source resources are an electrical engineers best friend, as usually there's a ton of information about a certain open-source piece of software or hardware online. If you're ever stuck on something, like a line of code that doesn't work, and you're using, as an example, Arduino, you can go onto their online forums and ask your question there. Other hobbyists visiting that forum can then chime in on what they think might be the issue and maybe even offer a solution. This shared collective knowledge through the internet has allowed for countless of small time engineers to tinker in the safety of their own home, with little to no prior knowledge on the subject.

That is exactly why we chose Arduino. Not only is the platform open-source and has countless resources to allow anyone to build just about anything they want, the platform has been around for a decade and a half now. That might not sound like a lot, but this little microprocessor has had 15 years of RnD (Research and Development) done to it, done by countless people all around the globe.

The Arduino programming language was originally based on Processing, a programming language that artists could use to make beautiful art, through the use of just code. The programming language, and alongside it, hardware, was later changed to allow external components to be added onto the board, instead of it being just a visual processor. Allowing other components to interact with the arduino called for a programming language that was properly equipped to deal with the crosstalk between components. This language ended up being C/C++. These languages are some of the most commonly used languages on the planet right now, and Arduino is partly to thank for that. Because of this, the information on the subject is quite extensive, and there's people in every corner of the internet that might be able to help you if you're ever stuck. Because of the integration and communication with third party items, like a bluetooth device, and the open-source nature of the platform, every single one of the small addon devices has libraries created by people

that go along with them. These libraries can simplify writing code for those small devices. For example, when we add code for a servo motor, we have to use the library specifically made for our servos. This library has functions inside that we can use to easily, and quickly assign an I/O (Input/Output) pin of the arduino to a servo, which we can directly plug into the pin as we specified in the code. Here is an example of how we get the library for our servo, and attach the servo to our specified pinout.

```
#include <Servo.h>

void setup() {
    Servo Myservo.attach(9);

}
```

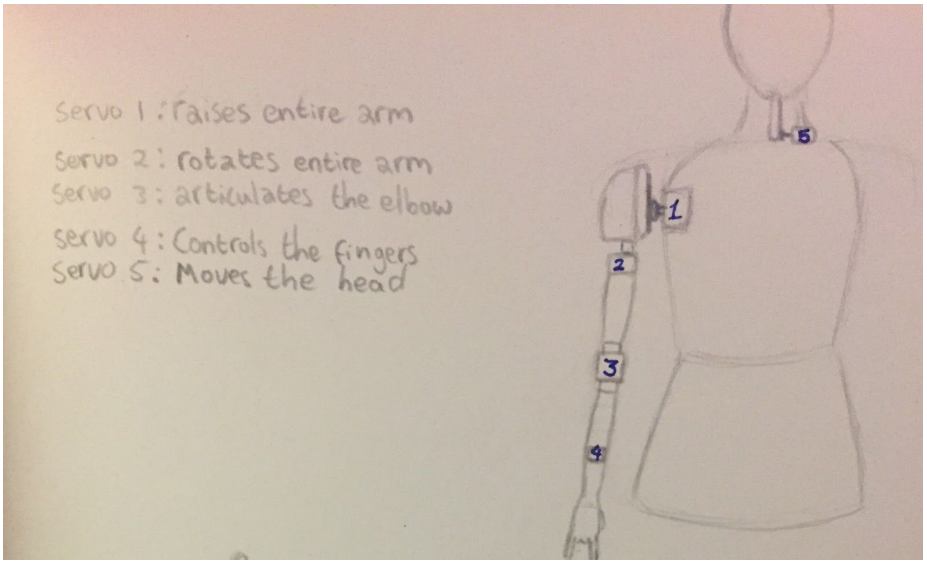
This is the style that Arduino code is written in. Since the code is based on the ever popular C++ and C coding languages like stated before. A written piece of code has to adhere to very specific and important rules to allow a computer, or microprocessor like the Arduino, to read the code, without getting confused. For example, if the # wasn't included in front of the `include <Servo.h>` , the computer reading that section of code wouldn't properly recognize it, and probably mistake it for something else. Because of the statement in between the brackets at the end, it would most likely not even be recognized at all, and the IDE (Integrated Development Environment, the program in which you write your code) would mark it as an error if the IDE was properly made. These IDEs are a great resource of programmers, as they allow them to quickly and effectively identify the issue, and fix it. It could be as easy as forgetting to add a closing parenthesis, or maybe using the wrong name when calling a function you've made.

Overview of the robot

Before constructing the robot, we made a few basic sketches of how we were going to design the moving hand, arm and neck. These were rough sketches and only helped us in further narrowing down on a final design.

We've made the decision of 3D printing most of the parts of the arm. We will be using 3D printed plastic to do so. Parts of the arm will also be made of aluminium plate. It's cheap and light, but hard to manipulate and form without the proper tools, which we don't have access too. That's why we're keeping our use of aluminium low and in places where straight long pieces are needed, as those are the easiest to make.

The plan is to use five servos in total for all the moving parts of the robot. When coming up with the idea we sketched multiple arms with different points of rotation. To simplify our design, we ended up going with three servos for the arm, one for the hand and one for the head. The first servo, servo 1, will be capable of raising the entire arm. We attached the servo to the left shoulder of the mannequin. This servo will be capable of moving the arm up and down. This servo will also have the highest torque of all servo motors that we've used, as it carries all of the arm's weight. The second servo, servo 2, will be capable of rotating the arm. This servo is connected below the shoulder that we've 3D printed. This means that everything below servo 2 will be capable of rotating 270 degrees.



The third servo that we are using, servo 3, will be capable of articulating the elbow. This means that the elbow will make the lower part of the arm move up and down. This servo will be working the most of all servo motors used in this project. The servo has two brackets that work perfectly as a joint. As we are trying to go for a metalhead theme, the lower part of the arm will be moving up and down quite consistent as the music is playing. We will explain this concept further down below.

The fourth servo, servo 4, will be controlling the fingers. We will be 3D printing the hand as well. We will be making the fingers movable and will be attaching springs to the tips of the

ring finger, middle finger and thumb, so that these fingers will be able to be pulled towards the palm of the hand. This will create a ‘sign of the horns’ effect and fits in perfectly with our theme.

The fifth servo we will be using, servo 5, will be capable of making the head bop. We have bought a styrofoam head that we will be attaching to a stick. This stick will be attached to servo 5. This servo will make the head move back and forth as well. We are basically using the same mechanism for most of the arm, as three of the five servo motors will be allowing its object to move back and forth.

Our plan is to make the robot capable of playing music. We will do this by downloading 3 songs and converting them to .wav format audio files and putting them on an SD card. We then will proceed to construct a circuit that will make playing music possible through an amplifier.

This robot will be remote controllable with an app that we are going to build ourselves. We will do this by using the program MIT Appinventor. This means that we will use Bluetooth to control the robot itself. This app will allow us to move the arm and play music by sending a signal to the Arduino UNO board we are using.

The arm

Our main reason for designing this arm is to put it on a metalhead robot, is to wave the sign of the horns with his hand and bop its head to music. For this project we’re using servo motors, as we weighed the pros and cons of both steppers and servos and determined that servos are the correct motors for this project.

There are three distinct parts that we have to figure out in this arm. The first section is the shoulder section. To be able to swing and rotate the arm, and allow the robot to move its sign of the horns back and forth, we’ve got to make a fully articulating shoulder. We are doing this through the use of two servos. One mounted to the body of the robot, which will raise and lower the entire arm, and a second one attached at a 90 degree angle to the first one, which will allow the arm to rotate around its axis. These two servos will be bearing the most stress of any of them, as the rest of the arm is hanging off of them. To make sure that this weak point won’t also be the point of failure, we took precautions and ordered heavy duty servos which can handle multitudes of what we’re throwing at it. To mount these servos onto our robot we’ve employed multiple techniques. Some we bolted onto an aluminium plate, some we screwed into wood, and others we sunk into 3D printed parts. The weak spot would probably be the 3D printed part, but because of the relatively calm things we’re going to do with this arm, it should be fine.

Our initial plan of ensuring that the joints will hold up, we were planning on designing the cad model to be quite rigid, but at the same time lightweight. We wanted to do this by filling in the model internally with honeycomb-like structures. These are proven to be some of the most

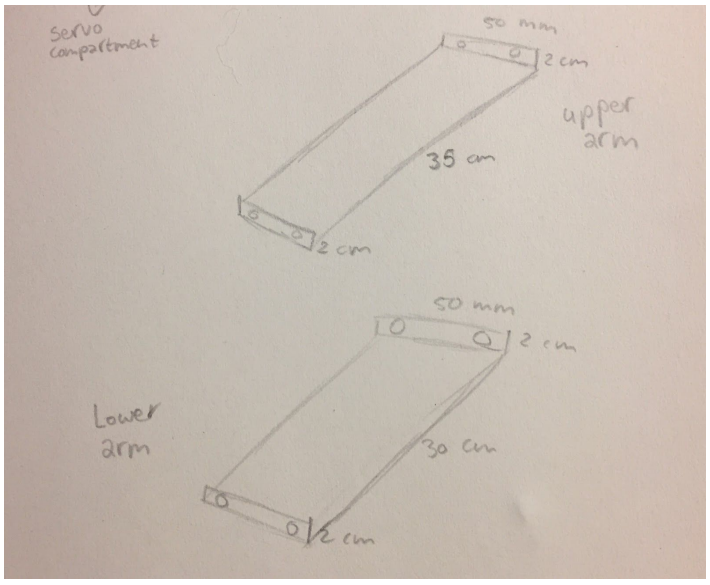
efficient ways of making something strong but lightweight. We were planning to do this by utilizing the properties of different 3D print materials and programs like Cura (A 3D slicing software, used to add the properties to your 3D model and lay out the paths that your printer has to follow). We decided to scratch this plan and use aluminum plates instead for the arm. We did this because aluminum is considered to be a very lightweight but very strong metal. This was convenient for our project and would take less time to construct, unlike 3D printing.

Even though we decided to not 3D print the arm, we were still 3D printing the hand.

There are two main materials that are commonly used for printing in the hobby are PLA and ABS. These are two common thermoplastics that have been around for years. They've been used in things like LEGO, certain car parts that have to be replaceable like bumpers and trim and many other things alike. Because of this rich history of use the strengths and weaknesses of these materials are quite well known.

We built the arm mainly out of aluminum strips. Aluminum is a strong but bendy metal. It's lighter weight than most metals and is also quite soft. For this reason, we used aluminum. Before cutting aluminum up, we used the size of an actual arm as a reference. We did this so that the proportions would be correct, as our mannequin is about the same size as an average woman. We then went to the technical college velsen to use their equipment, as we don't have the right tools to bend aluminum or drill holes into our aluminum strips.

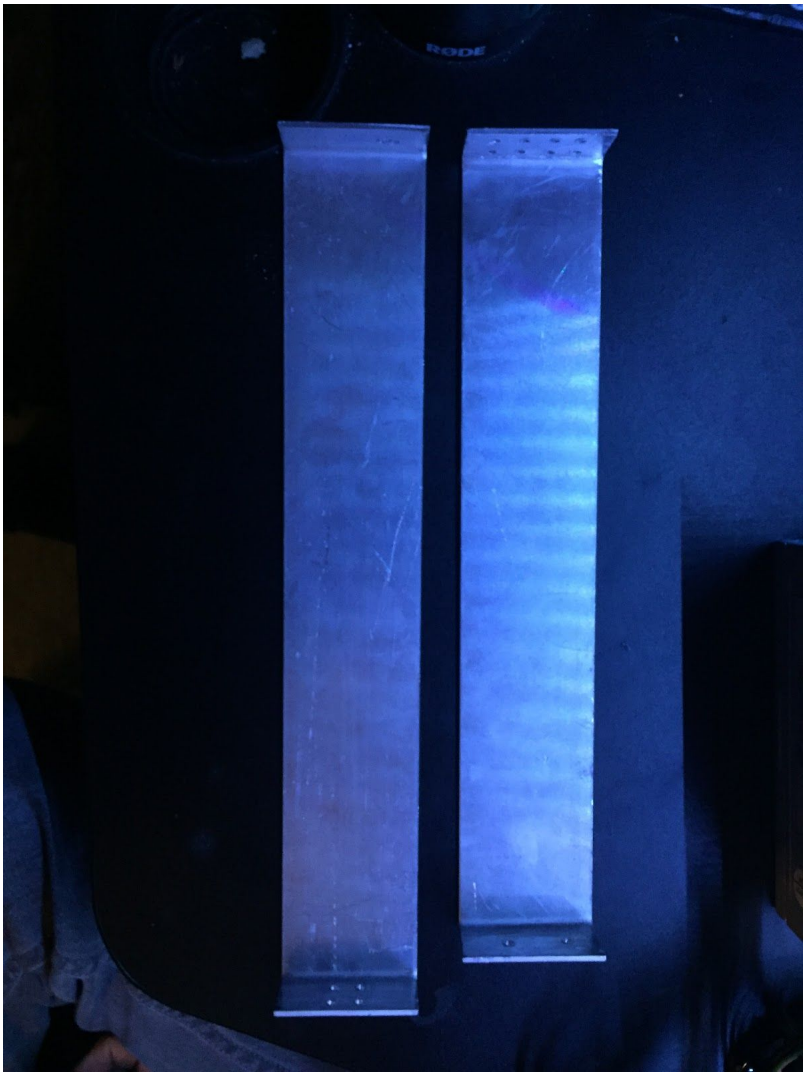
The measurements for the upper arm are 50 mm wide and 35 cm long. We added 4 cm to the length of the strip so that we could bend 2 cm of both ends in. So the aluminum strip for the upper arm was about 39 cm long.



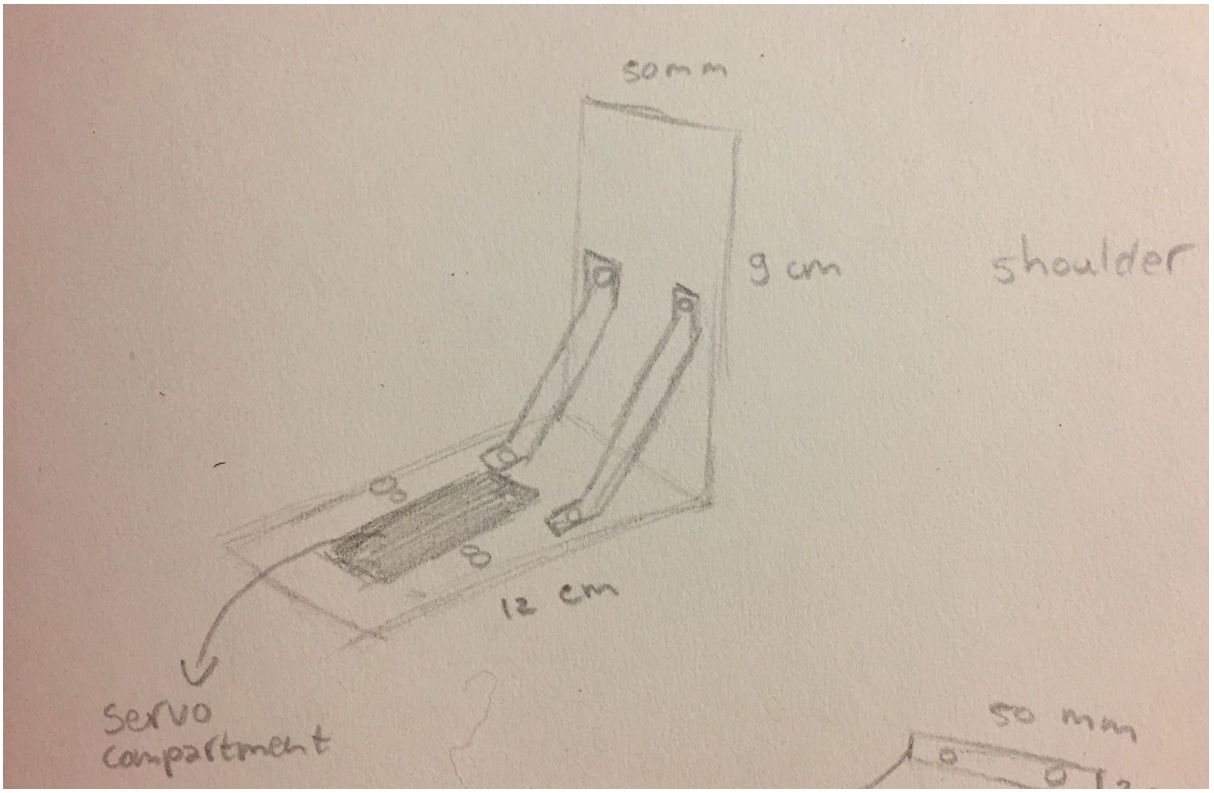
The measurements for the lower arm are 50 mm wide and 30 cm long. We added 4 cm to the length of the strip so that we could bend 2 cm of both ends in. So the aluminum strip for the lower arm was about 34 cm long. We bent the ends into a 90 degree angle so that we could drill holes into the bent surface. This makes attaching servos and other parts of the arm significantly easier.
The aluminum strip was initially 1 metre long and a 100 mm wide. We cut the strip in half in length, measured out both 34 cm and 39 cm, and cut both pieces out.

We used the excess of the aluminum strip to make a shoulder to attach the arm and servos to. This strip of aluminum was about 25 cm long. We bent this strip into a 90 degree angle and sketched out the exact measurements of a 35kg servo in the center of the surface of our aluminum strip. This is servo 2. We then proceeded to drill a lot of holes along the lines of

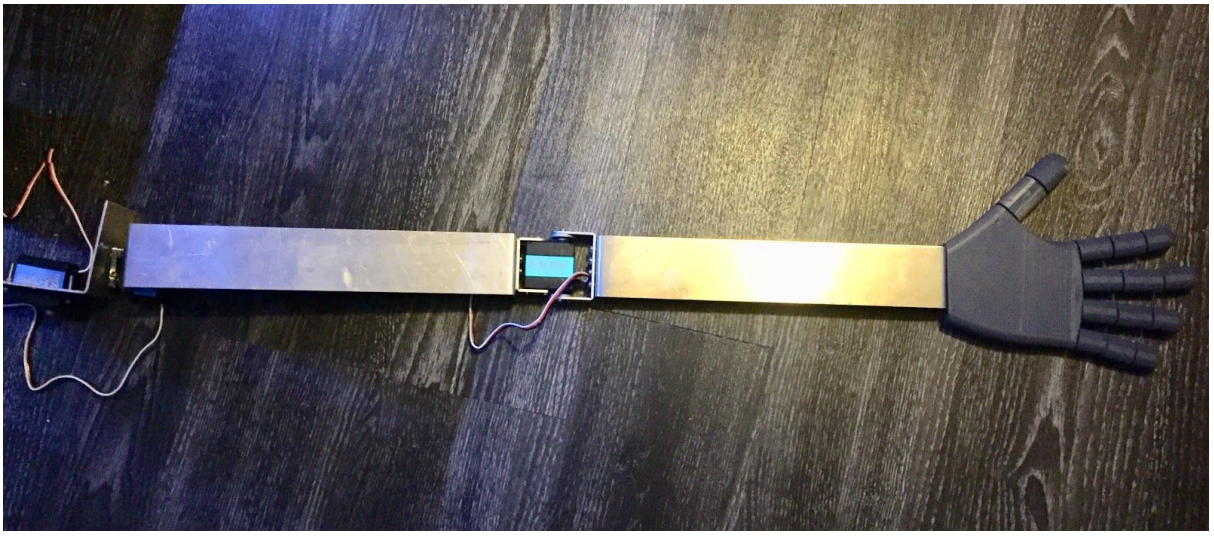
our sketch. This helped us create a rectangle the servo would fit in. This servo has two



brackets, so that we can attach the servo to the strip. After drilling in the holes, we filed the harsh edges, and our servo fit perfectly in our rectangular shape. We came to the discovery that aluminum might be a too bendy material to use for the shoulder. This could result in a big problem, as the shoulder carries most of the arms weight. For this reason, we wanted to make the shoulder stronger. We did this by cutting out two small strips of steel, and attaching them diagonally to the inside of our 90 degree angle. We did this by bending both sides of these two strips inward, and drilling them into our aluminum shoulder. This created a stronger shape for our shoulder, which was more reliable, as the entire weight of the arm hangs from this point.



We then drilled the brackets of servo 3 into one end of both the upper arm and the lower arm. All we did from here is drill our hand into the other end of the lower arm. We smoothed out the burs from each hole that we drilled to ensure a proper fit with the nuts and bolts. Servo 4 is connected to the back of our aluminum shoulder-piece and the shoulder of the mannequin. There is no specific speed we want to acquire, because of our conditions with the music circuit. This will be described later in our thesis.We have made the decision to not make the arm move to a certain bpm, or the specific bpm of a song, as this might be too work to achieve with our short timespan. Another reason for this is, we are unsure if the material of the arm can handle high speeds and it’s unclear how much our servos can handle.



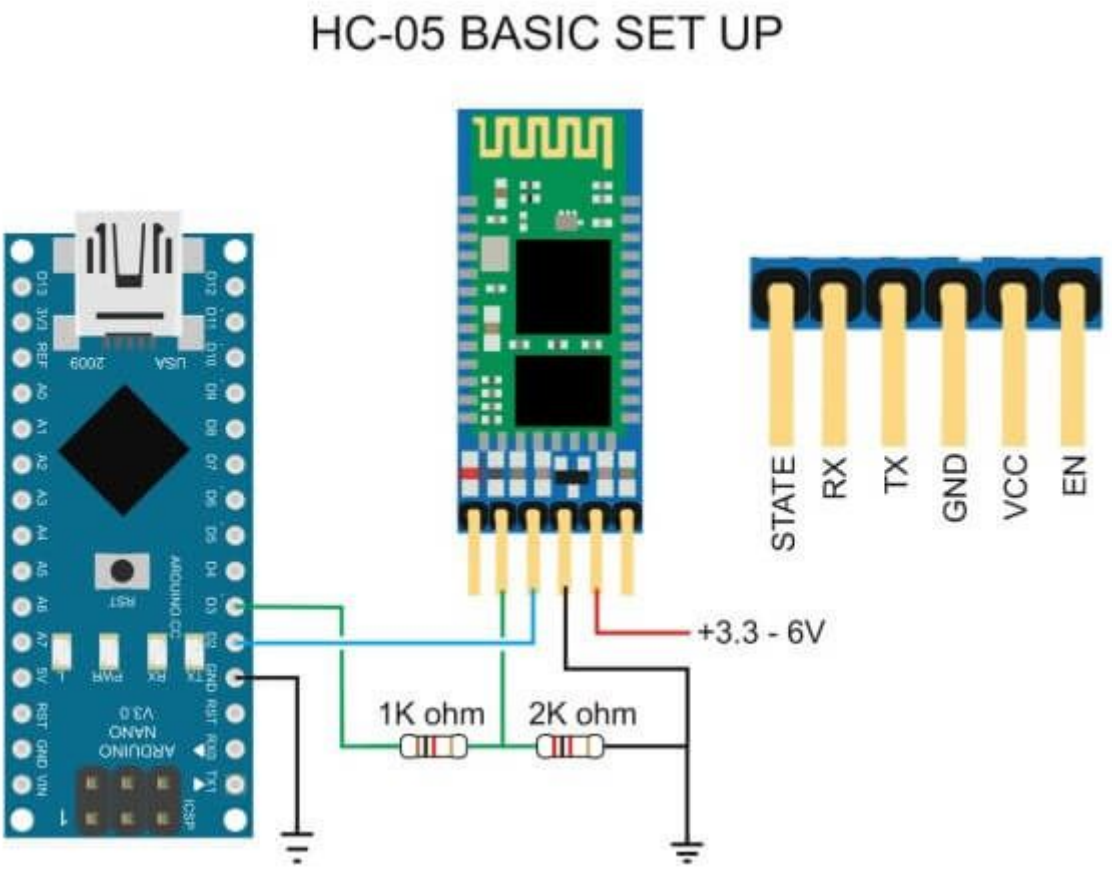
The mobile app

Because of the complicated nature of this project, and what we want to accomplish, we have run into the problem that controlling all these intricate parts is harder than we initially predicted. The original idea before we started this project was to add buttons to move the arm back and forth, up and down, and side to side. But these buttons would have to be expanded to also include the head, an adjustable knob so we could sync up the rate at which the arm and the head moved to different music with different BPM (Beats Per Minute). This array of buttons would soon be too difficult to easily operate, which is what we didn't want. They would also take up valuable pins that we needed for other things.

To overcome the hurdle of properly controlling the robot, we started looking at different solutions to this problem. Luckily, because of our own experience with hobbyist electronics, we were able to source from that experience, and landed on using a quite elegant way of controlling the robot. What we decided on is making the robot bluetooth enabled, and making our own app through which we can adjust any perimeter or variable which we want. This addition of a digital connection also meant that we could connect the music and robot directly to each other. Through the app, we can select any song that is loaded onto our SD card, and play it through our robot. The robot will then automatically recognize which song is being played and show the pre programmed pattern of movement associated with that particular song.

The app itself is made using the MIT Appinventor software. This lightweight easy to use developer tool allows us to make a custom app on the fly, and even allow us to make changes on the spot. This software was originally created by Google, but was handed off to a team at MIT to further support and develop it. This software is sadly not open-source, as they want to protect their intellectual property. Luckily, this doesn't mean we can't use it, it just means we can't edit the software ourselves.

But in this software, you can fully set up and customize your own app. Because of the more developer nature of this program, you can also code the backend of your app, allowing us to add the much needed integration of the bluetooth connection to our robot. To connect the phone with the app on it to our robot, we are using an HC-05 Bluetooth board that hooks into the Arduino through the use of some more external components. The module has a few pins that do different things depending on how you hook it up.



The HC-05 bluetooth module, shown in the configuration we are using.

As you can see, there are six i/o pins on the board, but for our use case, we will only be using the middle four as displayed above. The outer right pin is dedicated to switching the module into a slave or master configuration. This means multiple things, all of which are quite useful, depending on your use case, but not for ours. A slave module means that it only listens to what other devices order it to do, like sending a signal. A master can send out those signals, and command other modules around.

The other outer pin is used as a feedback pin, and allows you to check if the module is working correctly. But this can also be done by looking at the build in status LED on the board, right next to the other status LEDs. These LEDs, marked Tx and Rx are acronyms for Transmit and Receive. If either of these LEDs is lit up, it means it's busy with either transmitting data, or receiving data, depending of course on which LED is active. These can in turn also be used to confirm if your board is working correctly or not, without having to look at the data stream that is coming in at the arduino. A very clever piece of hardware that can be bought for less than €5. That price might be normal for us now, but it didn't used to be. These modules have hundreds of components, and ICs (Integrated Circuit) that themselves have hundreds, if not thousands of tiny transistors. And all this for just €5. That's why hobby electronics can be a fun and not too expensive hobby per se. Unless you start building a robot, that is.

The hand

To make this an actual metal robot, the one thing it needs to be able to do, is rock out and show people the sign of the horns. This is a hand gesture common to fans of metal and rock genres, popularized by metal legend Ronnie James Dio during his first tour with Black Sabbath back in 1979. Ever since it's been ingrained into the culture and multiple subcultures. But originally the sign was used during Wicca rituals to incorporate the devil into their religious rituals. Dio himself originally got the sign from the grandmother that used it to fend off the "Evil Eye", a curse that could be put on someone by simply glancing at them menacingly.

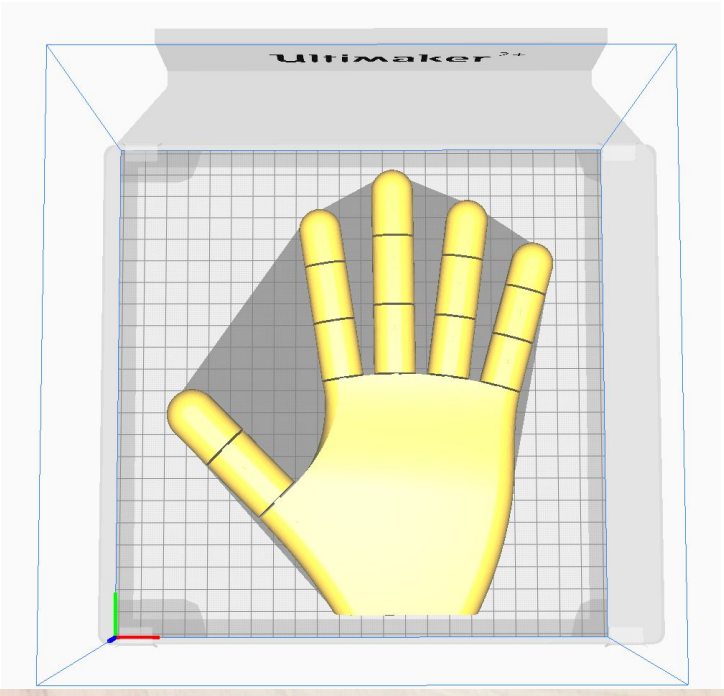
Because of this prevalent gesture among the fans, naturally we had to incorporate it into our robot, to really have it radiate pure metal spirit.

Now, to make the sign of the horns, you're going to need a hand first. Our hand is made from PLA plastic, a more rigid but strong plastic. We 3D printed it using an Ultimaker 2 printer, which was quite a nice experience as it's a quite high end model. It made the process quite easy and almost enjoyable. Once we had the hand printed out, we had to find out a way of moving the fingers into the actual position to display the sign of the horns. The sign consists of the pinky and index finger extended, while the ring and middle finger are held down by the thumb. To do this with a plastic hand, we need to be able to hold down and extend certain fingers. Our idea to do this is by making use of springs and ropes and another servo to rig something together. The initial plan was to use springs to hold down the ring and middle finger and thumb, and to fix the index and pinky finger in place. This means that the hand is constantly displaying the sign of the horns. If we wanted to make the hand show a normal flat hand, we have to run some strings to the three fingers that are being held down by the springs. We then wanted to run those strings up the back of the hand and arm to a servo we've attached there, and attach the wires to the servo. When we control this servo to pull on the strings, we effectively pull the fingers into the upright position.

After 3D printing the hand, we had a minor inconvenience. We forgot to put the right measurements in the program we were using to

3D print. This resulted in us waiting 8 hours for a miniature hand that most definitely did not fit the size of our mannequin.

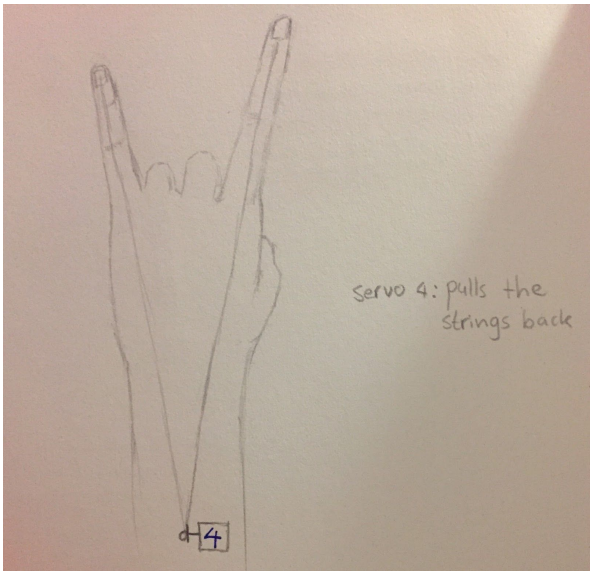
We then made a second attempt at 3D printing the hand, and this time made sure to actually put in the right measurements. After waiting 24 hours, we successfully had a lifesize 3D printed hand.
The hand was printed all at once. We did not have to put any parts together.



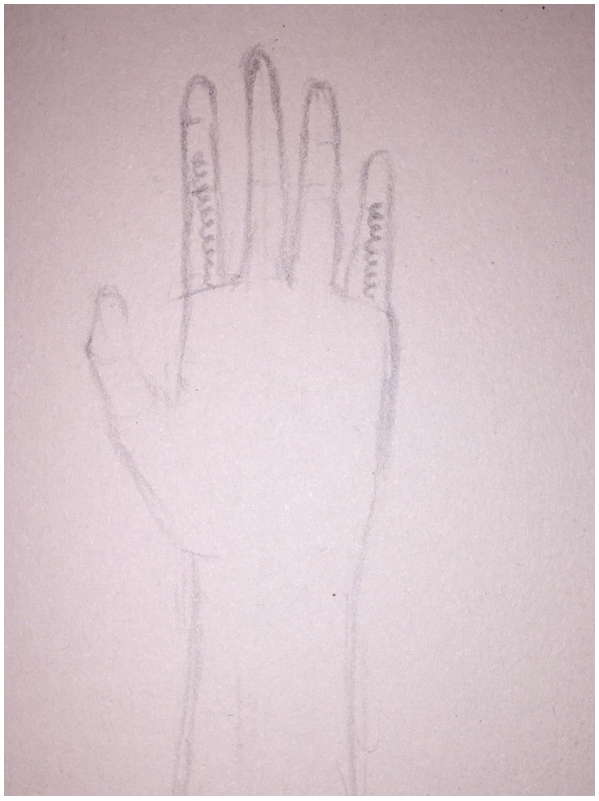
Model of the hand before printing

The hand after printing

After taking a close look at the hand, we came to the conclusion that it might be a better idea if we used the ropes and springs to pull the index finger and pinky up. This would mean that the servo would be on top of the hand instead of on the inside of the arm. This is more convenient considering the nature of the 3D printed hand. The fingers tend to automatically fall down to the palm of the hand. Pulling the two fingers up would make this project more convenient for us. Because of our change of plan, we made a new sketch:



We glued the strings to the fingertips and used a nail as a canal for the string to run through. We also glued two springs to the bottom of the index finger and pinky, as seen on our sketch down below:



We attached a small stick to the servo and connected the strings to the stick. Each other end of these two strings are attached to the pinky finger and the index finger. As a result, the fingers are pulled up when the servo makes the stick move up. We attached the hand to the arm by drilling it into the aluminum lower arm. This can be seen on the picture of our arm.
Through this mechanism, we can move the index and pinky finger on command to ready the hand to show the sign of the horns. A key component to the metal part of the robot, and the goal of this project.

Playing music

Adding music or sound to your project will make it significantly more interesting and more attractive to people. This obviously depends on your project, but for our music inspired research, we've concluded that adding music to our robot would make it remarkably more engrossing and would contribute to the metalhead theme we are going for. For this part of our project, we used an SD card, a normal speaker and Arduino Uno to add sound to our robot.

The hardware required to construct this system is reasonably easy to work with. We used an Arduino Uno board, an SD card reader module, an SD card, breadboard and some connecting wires.

We discovered that for playing sound from an SD card using arduino, the audio files need to be in .wav format because Arduino boards can only specifically play .wav audio files. A possible method to convert an mp3 audio file to .wav is to use an online audio converter. A great example for a website like this is: https://audio.online-convert.com/convert-to-wav . We then proceeded to use these settings, as we found that they worked the best and gave us higher audio quality:

Bit Resolution	8 Bit
Sampling Rate	16000 Hz
Audio Channel	Mono
PCM format	PCM unsigned 8-bit

After converting the files, we finally formatted our SD card and saved the .wav file into it. We made sure we formatted it before adding the file. After putting on the audio files, we named the audio files 1, 2 and 3. We memorized every audio with its name and did not change the name afterwards. This could interfere with the coding. This is where it still all went flawlessly.

After saving our audio files on our SD card, we interfaced the SD card reader module with our Arduino. The SD card communicates with the Arduino using the SPI communication protocol. As a consequence the Module is interfaced with the SPI pins of the Arduino, as shown in our circuit above. Serial Peripheral Interface (SPI) is an interface bus that is commonly used to convey data between microcontrollers compact peripherals such as

sensors, SD cards and shift registers. The interface uses separate data, clock and select line. The select line is used to choose the device you choose to talk to.

Arduino	**SD card module**
+5V	Vcc
Gnd	Gnd
Pin 12	MISO (Master In Slave out)
Pin 11	MOSI (Master Out Slave In)
Pin 13	SCK (Synchronous Clock)
Pin 4	CS (Chip Select)

The Arduino will be capable of playing audio from the SD card and play it from pin number 9. The audio coming from pin 9 unfortunately is inaudible. For this reason, we used an amplifier to amplify the sound. Arduino digital and analog pins only output just 20 milliamperes at 5 volts, which is insufficient for amplifying the sound, hence we need to use an amplifier. Line-Out is an unamplified audio signal, which is in the millivolt range. This is what we're getting from the audio output pin. To amplify this quite weak audio signal, we indeed need to amplify it. During the RnD process we used a fancy amplifier and speaker setup to eliminate some variables, but during live presentations of the robot we decided to use a regular guitar amplifier, as both of us have access to one. It might not give the cleanest sound, but for the heavy rock and metal genres, this might almost be a desired trait.

We then proceeded to program our Arduino. We did this by inserting the card our SD card module and used a library to make our project work. The library simplifies reading and playing audio files from an SD card through an arduino, which is immensely useful to us. We added the zip to our Arduino IDE by selecting 'sketch' and going to 'include library' and adding .ZIP library. We then selected the ZIP file that we had downloaded. We then put the program into our Arduino Program, clicked on Upload and were ready to play our music files.

Now, for the technical side of playing music through an arduino.

```
#include <SD.h>                      // need to include the SD library
//#define SD_ChipSelectPin 53  //example uses hardware SS pin 53 on Mega2560
#define SD_ChipSelectPin 10  //using digital pin 4 on arduino nano 328, can use other pins
#include <TMRpcm.h>           //  also need to include this library...
#include <SPI.h>

TMRpcm tmrpcm;   // create an object for use in this sketch


void setup(){

  tmrpcm.speakerPin = 9; //5,6,11 or 46 on Mega, 9 on Uno, Nano, etc

  Serial.begin(9600);
  if (!SD.begin(SD_ChipSelectPin)) {  // see if the card is present and can be initialized:
    Serial.println("SD fail");
    return;   // don't do anything more if not
  }
}



void loop(){

  if(Serial.available()){
    Serial.println("bread achieved");
    if(Serial.read() == 'p'){ //send the letter p over the serial monitor to start playback
      tmrpcm.play("music");
    }
  }

}
```

We used this code to see if our circuit is capable of playing music. We did this by connecting an LED light to a breadboard. The LED lighting up indicates that a signal is sent and music should start playing. We then proceeded to connect our circuit to an amplifier. We did this by soldering a 3.5 mm jack to the two cables that are connected to our arduino. We then connected our circuit to our amplifier.

The unfortunate part of this plan is that it did not work as it should have. When we turned on the amplifier we wanted to use, we got a very distorted sound. This was simple artifacting and that was when we realised that the arduino is simply not well enough equipped to handle high quality music conversion.

The head

We've made the decision of making a bobbing head to fit in with the metalhead theme. For the neck we have also made the decision to use servo motors. We've thought through the different types of materials that were most convenient for the head to ensure a smaller likelihood of failure. Our plan is to use a head that's lightweight and easily attachable to the servo. For this reason, we've chosen a head made out of styrofoam, as styrofoam is lightweight. Before making this decision, we wanted to use a mannequin hairdressing training head. We realized soon after that this might be too heavy for practical use, considering the servo's torque. We concluded that it's better to have an excessive amount of torque to be safe.

To attach the head to servo motor number 5, we've chosen to use a stick to put the styrofoam head on. While purchasing the styrofoam head, we discovered that there was a hole on the bottom of the neck of the head. This was very convenient, considering the method we used to attach the head to the servo. We've attached this stick to the servo by drilling it into the servo-horn. This allows the head to move back and forth, creating the illusion that the robot is headbanging. The servo itself is attached to the neck of our mannequin. We did this by glueing the servo onto the neck of the mannequin with epoxy.

Dressing up the robot

Our robot is constructed to resemble a metalhead. Beside the sign of the horns, we need different attire for the resemblance of a metalhead. We dressed up the robot with a classic Metallica t-shirt. Metallica is one of the most successful heavy metal and thrash metal bands of all time. The band was formed in 1980 in Los Angeles, California, with band members James Hetfield and Lars Ulrich. Since then, the band has sold over a hundred million records. A lot of people consider Metallica to be the band who reinvented metal. Metallica's first album, Kill Em' All, introduced exciting, spiky acerbic thrash metal, but it was their second record Ride The Lightning that changed the game immensely. Metal elitists may ascribe the creation of thrash metal to the efforts of bands such as Motörhead, Venom and a set of other obscure bands that added some extra fire and spice to the heavy metal footprint, but Metallica's tribute album truly is the starting point of a new sub-genre that continues to exert an expansive influence today. As Metallica has had a lot of influence on our music taste and lives, we've chosen their band-tee. Beside our t-shirt choice, we made the decision to put a wig on the styrofoam head of the robot. The reason for this is because it gives a more metal head vibe and look, which makes the headbanging look more traditional, vintage and interesting.

We both came up with the idea of building a headbanging robot on a call on Discord, at around 4 AM in 4 havo. We had this idea of building a robot that was capable of playing music and headbanging. We were both very excited about this idea and made the decision to build a robot for our practical assignment for physics class. We ended up constructing the robot for our research thesis because we had more freedom in doing research and had more opportunities regarding the 90 hour time limit of our thesis.

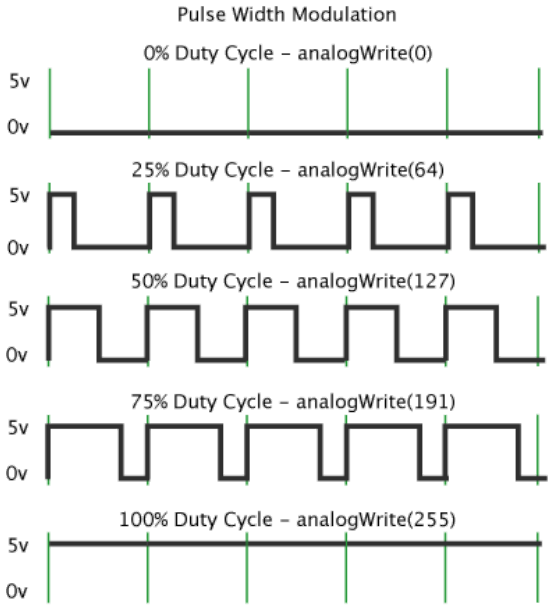
The circuit

After making the decision of using servo motors instead of stepper motors, we've settled on using Arduino Uno to control the servos used in our project. The Arduino Uno is the best board to get started with coding and electronics. Arduino Uno is a microcontroller board. The board contains 14 digital input/output pins, six of which can be used as PWM outputs. PWM means Pulse Width Modulation. It's a fancy term used to describe turning a signal on and off multiple times per second. This might remind you of AC power, as that's an oscillating signal with a frequency of 50 to 60 Hertz, depending on where you are in the world, as some countries supply their appliances with 50 or 60, or even both, like is the case in Japan. It also has a USB connection, used to program the chip on board from a pc, or just deliver power to the board. It also has an ICSP header, a different way to program the chip, a powerjack and a reset button. It carries everything that is required to support the microcontroller; simply connect it to a computer with a USB cable, and upload your code. It is possible to power it with an AC-to-DC adapter of battery as well. It does only support DC voltages of 5 to 20 volts, so you will definitely need to step down the power that comes from your wall and rectify the oscillating frequency through the use of a normal power brick.

Now, to power a servo you need a power supply with the recommended voltage and amperage for your specific servo. The most common servo on the hobbyist market right now is the 9g SG90 servo. It's a small little thing with not a lot of power, but has a great value for starting electronics hobbyists. As almost all servos are controlled the same way, through a PWM signal from a microcontroller, you can gain the experience you might want before stepping up to bigger, and thus more expensive servos. This might save you some headaches, and has definitely helped us.

An easy layout of the three pin header that comes with almost all standard servos.
The red cable is for your recommended voltage, which is in this case 5 volts, a ground pin, and a PWM pin.

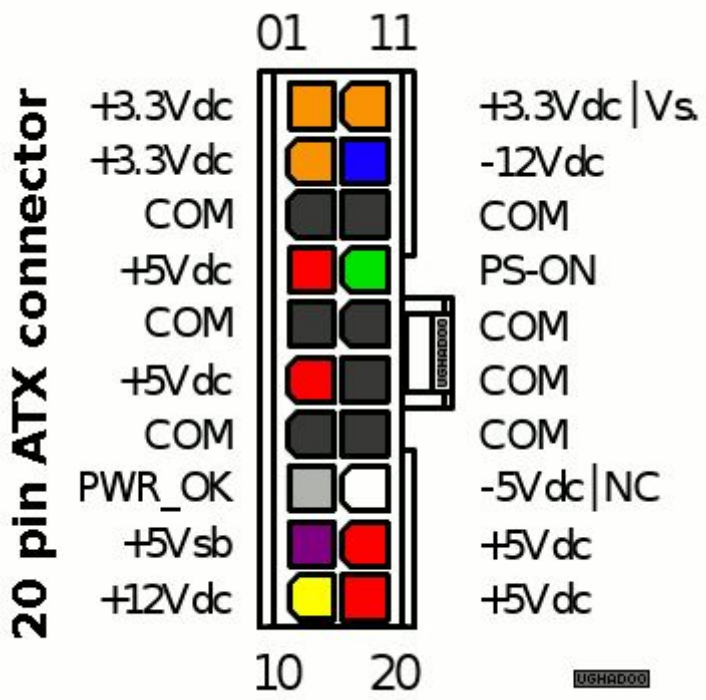
To control your servo, you first need to have it hooked up to power supply first, then you can send a PWM signal through the designated PWM pin to order the horn of the servo to go to a certain position in its scale, which is usually from 0 to 255, its maximum.



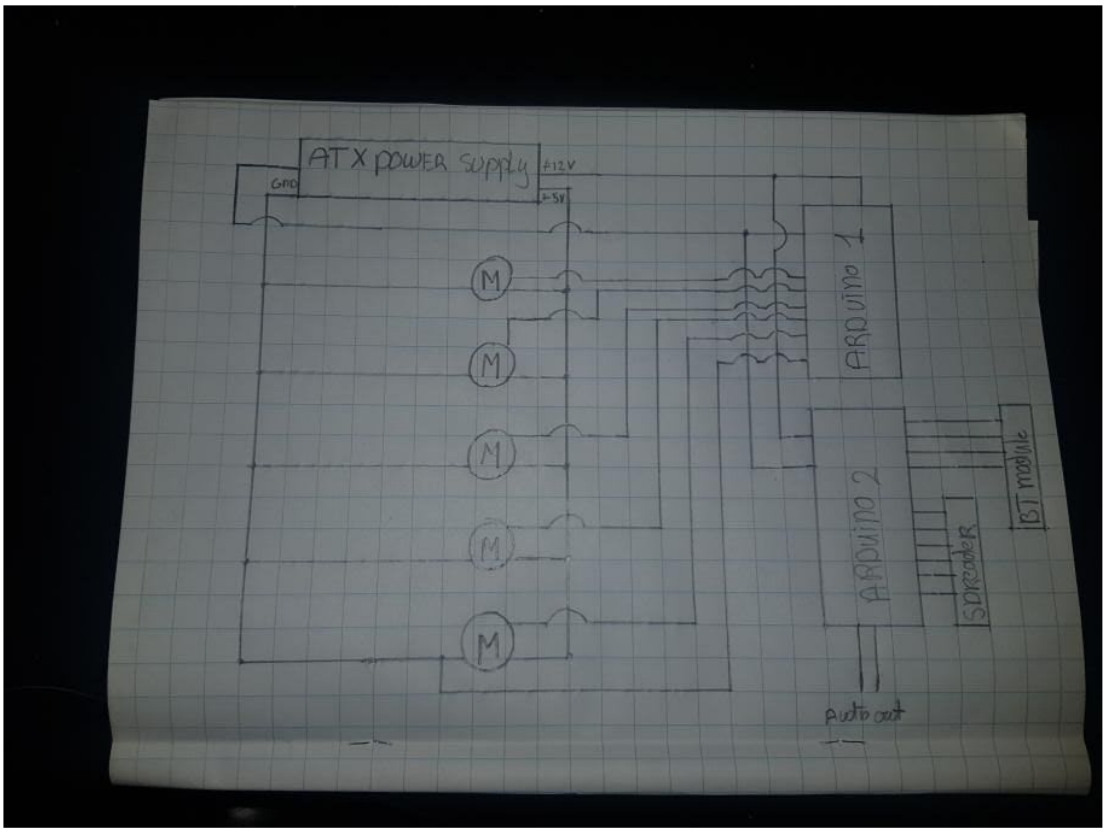
Now, as you can see in the picture on the left, a PWM signal can look a little intimidating, but if you're using the right tools, it doesn't have to be. Because of the arduino, and other people's open source libraries for servos, we can easily create this PWM signal with just a line or two of code. All further processing is done automatically by the arduino, so we don't have to. If you look at the different duty cycles, you may notice that percentage scales linearly with the written analog value. Because 0 is well, 0, and 255 is it's maximum, we can deduce that if we write a 0 to the servo, the horn will turn to the 0 position. And if we write a 255 to the servo, it will turn to its 255 position. These 0 and 255 positions coincide with their respective duty cycles. Position 0 is a 0% duty cycle, and the max position, 255, coincides with a 100% duty cycle. This is how we can control our servo fairly accurately. But not as accurate or precise as a stepper, which, like we have said before, has half steps or quarter steps in between its 0 and 255 step values.

Now that we can control our servo through the use of an arduino and an external power supply, we can start figuring out what we can actually do with it. Most servos have a stall torque, which indicates the maximum torque it can deliver, as the motor is stalled i.e. not moving. A stalled motor or even servo can be quite dangerous and harmful to the motor itself. That's why when building something with a motorized part, like a robot, you always want to figure out the maximum stall torque of your motor. Because that is the maximum weight you can put on the servo, so you can figure out the limitations of whatever you're building. Because we are adding these servos to control an arm and a head, we are dealing with levers. This means that a weight, or a force, is much harder to hold the farther away from the rotation point it gets. Because this is an arm, we will have quite a considerable length in between the servos in the shoulder and the elbow, and the hand at the end of the arm. Because we didn't want to make a robot that couldn't even list its own arm, we used much stronger servos than you might typically find in a hobbyists shop. Our servos can deliver an astonishing 29 kgcm torque at its stall, at the 5 volts we are powering them with. This will be more than enough,as the arm will weigh no more than a kilo or two. The lever of our arm is also not that long compared to the strength of our servos, so we will not have any problems on that front either. But to be sure, let's take the theoretical length of our arm, around 50 cm. And a weight of 2 kilos. All we need to lift that arm is 2 kgcm at the base of the servo, and just slightly more as the length of the arm increases. The servos we're using are definitely strong enough to lift these in comparison miniscule weights.

The reason that we are feeding 5 volts into the servos instead of a little higher voltage, which could give us a kilogram or two of more torque, is because we are trying to build this robot not to be the best there is, but to the best of our abilities. And this includes sourcing parts from friends and family. This includes the clothes mannequin that we built the robot on, the power supply that we scrounged from an old pc and had to modify to actually work, and an array of smaller parts used in the construction and dressing of the robot, like old t-shirts and the lower chassis of a swivel chair, as written about before. Because these components aren't perfectly made to our standards and use case, they don't quite overlap with what we need, like with the power supply. It only has 5, 3,3, and 12 volt



Because we are using a scavenged ATX PSU (Power Supply Unit), we did run into some issues regarding the actual power delivery method. Because it's an old computer PSU, it had a lot of connectors that were quite useless for us. They were custom to fit into certain computer parts, like for example, the 20-pin connector fit into the 20-pin motherboard power input. We don't really have a need for a 20-pin connector, but we do have a need for the wires that go into it. Mainly, the yellow wires with a potential of 12 volts, and the corresponding ground wires. For our power supply, these small yellow 12 volt power cables have a rated power of a whopping 300 watts. This is more than enough to power our servos, as the max rated power consumption we will see from our high power servos will be 38 watts. This is calculated by taking the supply voltage (5 volts), and multiplying that with the stall current (1.9 Amperes) which gives us the max power consumption for one servo, which is 9.5 watts. Then multiply that with the amount of high strength servos we have, which is 4, and you get 38 watts total. Since our maximum suppliably power is 300 watts, and we are only using 38 watts for the main servos, which are the bulk of our power draw, we have plenty of excess power left to go around to smaller components like the arduino board, and its components, like the HC-05 bluetooth board. These require significantly less power than the power hungry beasts that the servos are. The current used by them is in the milliampere range, instead of the normal ampere range. Even after adding the rest of the components, we have plenty of spare power to go around.



Circuit diagram of the entirety of the robot

One other big plus of hobbyist electronics is that a lot of compatible components are all made to support the same voltages. These hobbyist components can't be too dangerous to their usually inexperienced or careless user base, so they are usually made to support voltages below 48 volts DC. And many components don't even need such high voltages, as we can see with the arduino, which can be powered with a voltage between 5 and 20 volts, but by manufacturer recommendation it's better to power it with a voltage between 7 and 12 volt. Because of this, we are using the 12 volt rail on the power supply. And even if a certain component doesn't use the 5 volts supplied by the arduino or power supply, we still have two options to make it work. Either we modify the power supply some more to use its other voltages on its power rail ranging from 3,3 to 12 volt. And if these voltages don't coincide with the needed voltage, we can even add in a buck/boost converter to step up, or down, the supplied voltage to match the required voltage.

Currently our robot is sadly bound by a wire, as we are using a normal ATX power supply from a computer, which needs regular 240v AC power to function. If we were to make a second robot with more time and money on our hands, it would probably be powered by a large battery bank, as we can then bring the robot to locations that might not have wall power.

Results

This research thesis has required a lot of time and effort in order to make our project work. This came with trial and error and a lot of hours consisting of coding in agony. In the end, it was worth it, as we've built a fully functioning head banging robot. We've learnt new things and gained a lot of experience in many fields that we were not as familiar with at the start. Even with some failures and change of plans, we've constructed an actual functioning robot.

After our project, we've come up with the following results:
A realistically moving arm is quite hard to make with the use of servos. Especially considering that servos are not actual joints, and the human body is extremely difficult to artificially recreate with robotics. However, we've made the arm capable of rotating and extending. This is what was required for our project, and these were goals we set for ourselves. We accomplished these goals by bending strips of aluminum. These worked as a base for the arm. We drilled servos into them and then proceeded to power the servos by using a power supply unit. After doing so, we used arduino, and a Bluetooth module to control the arm through an app we built ourselves.
We made the fingers move by glueing springs to the bottom of the fingers. We then glued strings over the top of the fingers. We attached a nail to the top of each finger, which worked as a guide for the strings. We attached the strings to a stick, which we drilled into a servo. We placed this servo on the top of the hand, which created a smaller distance between the strings and the fingers. This resulted in the fingers going up when the servo would move the stick away from the hand. Making the index finger and pinky finger move up was convenient, considering the fingers naturally fell down towards the palm of the hand. The servo we used for this had a lower torque, as the fingers were quite lightweight.

When trying to make the head bop to predetermined music, we came across a few difficulties. One of them being severely distorted audio. We converted music files into .wav files and saved them onto an sd card after formatting it. We then built the circuit required to make this work and soldered a 3.5mm jack to the cable connected to the arduino. Unfortunately, it did not work. This could have been because of the circuitry, the programming, the wiring,the Bluetooth module, the SD card reader or because of the soldering. For this reason, we scratched this plan and decided to only make our robot headbang. This on the other hand, was not as complicated to construct, considering the mechanism is less complicated. We put a styrofoam head on a stick and attached the stick to the servo. By doing this, the servo can rotate back and forth as the head moves with it. This created an uncanny headbanging effect, which we ended up trying to cover by using a wig. We powered this servo with the same power supply.

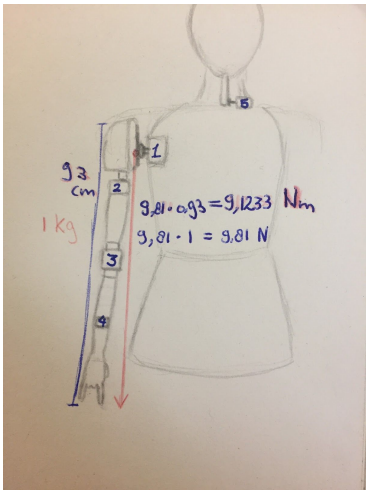
We also resulted in a mobile app that allowed us to move the arm. We initially wanted to control the audio-playing circuit through the app as well, which did actually work. After scratching that plan, the only thing that was left to control was the servos of the shoulder, the

elbow and the hand. We constructed this mobile app by using MIT app inventor, which was simple to use.

We made all the parts move together as one, by keeping things relatively simple. This came with scratching plans, changing plans and trial and error. We used a power supply to power the servos and used arduino to build circuits to use for our Bluetooth module. We needed this for the app we built, to make the robot remote controlled. We did end up using arduino to bind things together, which came with some more coding to do, but it ended up working for our project, which was the most important part.

Our results for the force of the arm:
F=m*g
F=1*9,81=9,81 N

Our results for the torque:
M=F*r
M=9,81*0,93=9,1233 Nm



Conclusion

This project came out much different than we initially expected. We had to jump through and hurdles all the time, and had to work around the clock to meet deadlines. We were even warned beforehand that taking on this much work for our thesis was more than we could handle, and it ended up being much more than we had bargained for. We had good days, but there were definitely bad days too, for example, pulling all-nighters because we finally had a day in our schedule that lined up so we could work alongside each other. It was definitely tough, but in the end, it was all worth it. We learned a great deal of new techniques and skills that we can later put to good use in our studies. Even when we thought we knew what we were doing, we were thrown another curveball, which presented a new challenge we had to overcome. But we worked through it, and we're glad that we did.

In conclusion, a headbanging robot can be constructed by using different skills, techniques and knowledge about electrical engineering, circuitry and coding. Constructing a robot can be done through the use of servo motors, for rotation. A way to power these motors is by the use of a power supply unit. The base of the arm can be made out of different materials. One of these materials is aluminum. When building an aluminum base, multiple tools are needed to bend the metal and drill holes into the metal. In order to make the robot remote controlled, a Bluetooth module should be used. For this to work, using an arduino board could be a good method to construct a well functioning circuit. A way to make the fingers move is by 3D printing a hand and attaching strings and springs to the fingers. By using a servo with a lower torque, these fingers can be pulled up and let down.
The whole robot can work together as one by making use of an arduino uno board. This was required to wire the entirety of our robot.
Overall, this project was a success, and we've learnt what it takes to construct a headbanging robot.

Special Thanks to:
Massimo Banzi(creator of arduino) and his team, the technical college velsen, mister Cijsouw for his help, support and feedback.

Sources

https://www.3dhubs.com/knowledge-base/pla-vs-abs-whats-difference/#what-are-abs-and-pla
https://learn.adafruit.com/all-about-stepper-motors/
https://forum.arduino.cc/index.php?topic=522508.0
https://www.linearmotiontips.com/how-does-closed-loop-stepper-control-work/
https://processing.org/
https://www.arduino.cc/en/reference/servo
https://www.machinedesign.com/motion-control/what-s-difference-between-servo-and-stepper-motors
https://circuitdigest.com/microcontroller-projects/arduino-audio-music-player
https://www.robotshop.com/community/tutorials/show/arduino-5-minute-tutorials-lesson-5-servo-motors

https://www.youtube.com/watch?v=nL34zDTPkcs
https://appinventor.pevest.com/2015/01/23/part-1-basic-bluetooth-communications-using-app-inventor/
https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi/all
https://github.com/TMRh20/TMRpcm
https://howtomechatronics.com/tutorials/arduino/arduino-and-hc-05-bluetooth-module-tutorial/
https://components101.com/wireless/hc-05-bluetooth-module
https://www.c-sharpcorner.com/article/audio-play-using-sd-card-module-and-arduino/
https://www.arduino.cc/en/reference/SD
https://gyazo.com/22d9e31312f513b18909cfb89263e851
https://i.gyazo.com/6704ed42881dd7b254818e0a291976cb.jpg
https://gyazo.com/0c60bf8e7fc383f633c48397a51d3932
https://a.pololu-files.com/picture/0J222.450.jpg?5d3c70fbaa99db0ed6512154e034b40f
http://www.hizook.com/files/users/3/Meka_Robotics_A2_Compliant_Manipulator.jpg
http://inmoov.fr/youvideo/
https://www.youtube.com/watch?v=AGHwW5TSNCY
https://easyeda.com/
https://forum.sparkfun.com/viewtopic.php?t=35352
https://forum.pololu.com/t/connecting-servos-in-series/8446
https://forum.arduino.cc/index.php?topic=573503.0
https://ultimaker.com/software/ultimaker-cura
https://www.tinkercad.com/
http://everycircuit.com/app/
https://www.instructables.com/id/Convert-A-Computer-Power-supply-to-a-Bench-Top-Lab/

And a thanks to the stores that helped us pick out the right parts:
Baco Army Goods
Stals Ijmuiden
Aliexpress
Alles Ijmuiden

Logbook

Logbook Carmen:

Discussing robot - 2 hours - 10-9-2019
sketching - 1 hour 10-9-2019
Discussing with mister Cijsouw - 2 hours 13-9-2019
sketching - 2 hour- 13-9-2019
Scavenging for mannequin - 4 hours 24-9-2019
Research - 2 hours 27-9-2019
Ordering parts - 1 hour 2-10-2019
Writing - 2 hours 3-10-2019
Installing programs and setting them up - 1 hour 3-10-2019
Research - 2 hour - 4-10-2019
Writing - 4 hours - 4-10-2019
Building circuit & research- 6 hours - 7-10-2019
Coding - 4 hours - 8-10-2019
Writing - 5 hours - 12-10-2019
Testing servos & coding - 3 hours -13-11-2019
Working with metals - 6 hours- 14-11-2019
Writing - 2 hours - 20-10-2019
Putting arm together - 2 hours 20-11-2019
Writing - 2 hours - 20-10-2019
3D printing attempt 1 - 2 hours - 11-12-2019
Writing - 1 hour 11-12-2019
3D printing attempt 2 - 1 hour 12-12-2019
Reviewing progress with mister Cijsouw - 30 minutes 12-12-2019
Writing - 3 hours 12-12-2019
Sketching - 1 hour 13-12-2019
Constructing hand - 3 hours 27-12-2019
Coding - 1,5 hour 27-12-2019
Testing - 1 hour 27-12-2019
Research - 2 hour 29-12-2019
Testing - 1 hour 29-12-2019
Coding - 3 hours 29-12-2019
Circuitry - 3 hours 30-12-2019
Constructing neck - 2 hours 02-01-2020
Writing - 4 hours - 02-01-2020
Coding - 3 hours - 03-01-2020
Writing - 3 hours - 03-01-2020
Photoshopping - 30 minutes 05-01-2020
Writing - 4 hours 07-01-2020
Coding - 3 hours 08-01-2020
Coding - 2 hour 09-01-2020
Writing - 6 hours 09-01-2020
Finishing touches research thesis - 2 hours 10-01-2020

Logbook Niels:
Discussing robot - 2 hours 10-9-2019
sketching - 1 hour 10-9-2019
Discussing with mister Cijsouw - 2 hours 13-9-2019
sketching - 1 hour 13-9-2019
Scavenging for mannequin - 4 hours 24-9-2019
Research - 2 hours 27-9-2019
Ordering parts - 1 hour 2-10-2019
Writing - 2 hours 3-10-2019
Installing programs and setting them up - 1 hour 3-10-2019
Research - 1 hour 4-10-2019
Writing - 4 hours 4-10-2019
Building circuit & research- 6 hours 7-10-2019
Coding - 4 hours 8-10-2019
Writing - 5 hours 12-10-2019
Writing - 2 hours 12-10-2019
Constructing power supply - 3 hours 17-10-2019
Testing servos & coding - 3 hours 13-11-2019
Writing - 4 hours 13-11-2019
Working with metals - 6 hours 14-11-2019
Putting arm together - 2 hours 20-11-2019
Writing - 4 hours - 21-11-2019
3D printing attempt 1 - 2 hours 11-12-2019
Writing - 3 hours 12-12-2019
3D printing attempt 2 - 1 hour 12-12-2019
Sketching - 1 hour 13-12-2019
Writing - 3 hours 13-12-2019
Getting supplies - 1,5 hours 27-12-2019
Constructing hand - 3 hours 27-12-2019
Testing circuitry- 2 hour 27-12-2019
Circuitry - 3 hours 30-12-2019
Coding - 2 hours 30-12-2019
Constructing neck - 2 hours 02-01-2020
Coding - 2 hours 03-01-2020
Writing - 4 hours 03-01-2020
Writing - 4 hours 05-01-2020
Coding - 2 hours 07-01-2020
Testing - 2 hours 08-01-2020
Coding - 3 hours 08-01-2020
Writing - 6 hours 09-01-2020