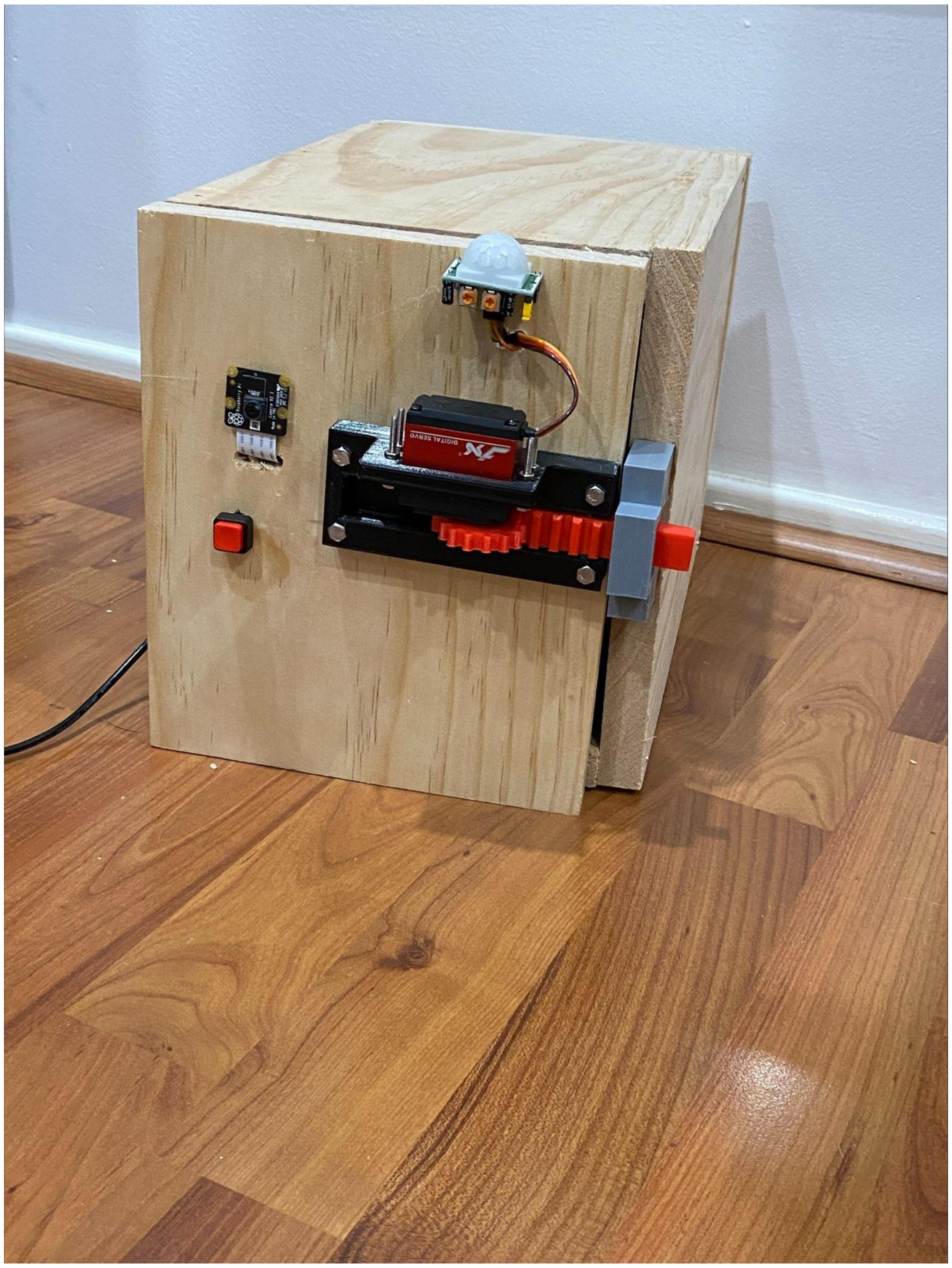


The Jankiest Vault in The World

Jorawar Singh (z5364212) | 25/3/22

You only need to Read the chapters in red

Table of Contents

Link to code

All code will be available on my GitHub repo,

https://github.com/JorawarSingh09/SA_FINAL

Project Outline

For my Something Awesome Project I plan on using 3D printing, 3d Modelling and Arduinos to create several security devices. Some ideas on the table are a fully fledged home security system (inspired by the lack of security In my house, read more here.) I will do this through first designing some gadgets/devices, then modelling them in a 3d modelling software. From there I will print these out and incorporate them with motors/servos/sensors controlled by a Arduino board. The Arduino board will be programmed to operate all connected sensors and motors. I would also like to explore the Internet of Things (IOT) with Arduinos through this project, where I will send events the Arduino picks up wirelessly to my phone, either through email or an app I develop.

UPDATE 25/3/22 ==

The project has been updated to not be based around several small security devices but just one.

I aim to create a vault, which instead of being opened by a key or a password will be accessed through facial recognitions software. For this I plan to use a RaspberryPi and the OpenCV libraries available for use. I will continue to incorporate Arduinos, IOT and 3d printing/modelling in the project.

The security functions of the vault will be:

- Facial Recognition for access to vault.
- A servo operate lock that operates if correct user is identified through facial recognition.
- Motion detection which triggers an email to be sent to the owner, this will cover the IOT learning requirement.

LEARNING GOALS

- Learn how to create active mechanisms through designing and 3D modelling
- Learn how to integrate Arduinos and Raspberry Pi's into one ecosystem. They must be able to communicate with each other and relay any input/outputs going through the system
- Touch on some facial recognition and "Machine Learning"
- Learn how to implement IOT into my ecosystem and update the user through their personal device.

PREVIOUS KNOWLEDGE

3D Modelling

I have very little experience with 3d Modelling, I've made some very simple brackets for doors that are just rectangles with a screw hole. From this I learnt modelling with simple shapes and using real life measurements to design something. I would like to further my experience through my SA project and begin to design moving mechanisms.

3D Printing

While being relatively straight forward (give printer a file and it prints it) 3D printing can be a topic which can be quite complex, from the materials to the fine tuning of every single setting in the printer.

I've had my 3D printer for about 3 months now, I used it pretty frequently, printing anything I could, I even made a RGB lamp for a friend's birthday before (YouTube link). For the past month the printer has been out of commission due to issues calibrating the print bed (its most likely warped.) I've ordered an auto leveler device for it and installed it but I didn't have any luck, I will continue trouble shooting the printer over the course of the project, worse case I will have outsource printing.

Arduinos

Arduinos are physical programmable circuit boards or microcontrollers. Arduinos can read inputs from sensors and send outputs to components such as LEDS, motors, servos etc. in order to achieve a defined use. Arduinos are programmed through a simplified version of C++.

I would say I have a pretty good understanding of Arduinos and their use. The RGB lamp I mentioned above.

This is a picture from before it was all wired up. I also run an after-school program where I teach high school student robotics through Arduinos. Because of my familiarity I've decided not to focus too much on components I'm familiar with but rather try to learn IOT and how to communicate between multiple Arduinos or between an Arduino and a RaspberryPi.

RaspberryPi

RaspberryPi are like Arduinos but are fully fledged computers, while Arduinos are programmed through an IDE on another computer RaspberryPi are just small Linux machines, thus you can connect a keyboard and monitor to one and use it as a mini-PC.

RaspberryPi do have limited performance due to their small size, so you can perform tasks that require a lot of processing power. Just like Arduinos you can connect RaspberryPi's to different sensors and output components to create different devices.

I have no experience with a RaspberryPi, but I believe my history with Linux and Arduinos will prove it relatively easy to learn and the fact that its more capabilities than an Arduino will let me do more. I'm currently looking into facial recognitions with it and there are two options in terms or Facial Recognition libraries, neither are fully compatible with Pi nor are meant to be used on fully fledged Linux systems.

TensorFlow

In fact, TensorFlow is a machine learning framework that helps you build machine learning models. It is more famous for building neural networks, which are one type of algorithms and approaches in machine learning, called deep learning.[1]

OpenCV

OpenCV is a computer vision framework that helps you do all sorts of processing on images and videos. Since its release, it's been a widely used tool for image processing tasks. It enables you to manipulate pixels easily, so that you can build your own image and video processing algorithms if you wish to do so.[1]
From what I've read all sources say that OpenCV will be my best hope for deploying facial recognition of the limited processing power of the RaspberryPi.

THE PLAN

For this project I will be completing my learning through completing the below steps. I find it difficult to follow a course type learning plan and with the number of different systems I will be using I believe it will be more effective to learn what I need as I go.

Step 1: Design locking mechanism and Vault. The body of the vault will most likely be made from wood instead of being 3D printed because it's not something I want to focus on, and it would take forever to print.

Step 2: Fix issues and configure 3D printer

Step 3: Print out components of locking mechanism.

Step 4: Test locking mechanism through just Arduino. I will program the basic functionality using just the Arduino boards, I will incorporate the RPi at a later stage. Due to power delivery issues its most likely I will only drive the servo on the Arduino so any other inputs or outputs will be dealt with through the RPi's input output ports (GPIO). This step will be where I make sure the mechanism works as

[1] https://towardsdatascience.com/which-is-better-for-your-machine-learning-task-opencv-or-tensorflow-ed16403c5799

intended effectively, if started by testing out with the RPi there's a chance the issues I find may be caused by the RPi rather than the mechanism itself. Basically, reducing points of failure.

Step 5: Implement Servo and Arduino Mechanism through the RPi, at a glance I will be using the pyFirmata library which allows for communication between the RPi and Arduinos.

Step 6: Add motion detection, check if there is movement. This sensor will connect to the Rpi.

Step 7: Implement IOT, when there is motion detected, take a picture, send an email.

Step 8: Assemble all components

Demonstration

https://youtube.com/shorts/jNO24AxZOE0?feature=share

ANALYSIS

SECURITY

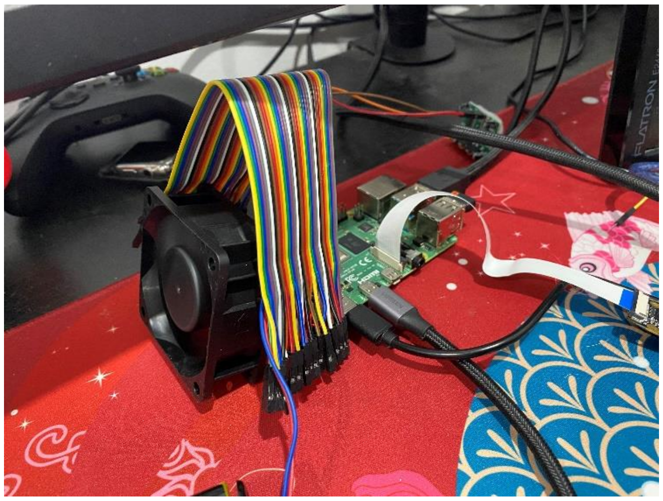
The Vault does have one major flaw when it comes to its level of security. Since it detects face through analyzing 2d images, its unable to tell if it's looking at a face or a flat picture. This allows attackers to use pictures of the owner to enter the vault. Besides that, I believe it functions well. It detects movement quickly and reliably sends out an email. Now that it has been trained properly it doesn't allow strangers who are not authorized to unlock the vault with their face.

Security could be improved by using a camera module that functions like FaceID and scans the 3d structure of your face as well. Increasing performance of the Rpi would also help in solidifying security, as if it's able to process more datasets of faces it would be able to increase its reliability in facial detection. It would also increase the speeds at which it searches for a face as currently due to hardware limitations the feed it gets isn't live.

ISSUES

Disregarding the security issues mentioned above, the most prominent flaw of the system is its ability to power all components. The Rpi must be connected to a wall to DC adaptor or the power it draws carrying out the facial recognition program is too great, especially when paired with that's it is managing an Arduino board as well has keeping track of inputs on its GPIO, it's hard work. When trying to run the Rpi through a portable power source it would crash within seconds of running the software for the vault. In order to overcome this issue, I decided to continue using the wall to DC power adapter.

Heat dissipation is also an issue when it comes to the Rpi. My current solution is just to attach a blower fan to the vault which will keep constantly cooling the boards. A more elegant method I've researched is water-cooling the board, not only is it effective and allows for high CPU clock speeds but it also looks amazing.



I also had a lot of trouble with OpenCV and pyFirmata compatibility with the Rpi or the very least with PI OS which is the Linux flavor running on the Rpi. A lot of modules from the guides were unable to be located and alternatives had to be used.

Reflection

I went into this project inspired by the think like an attacker mindset that was demonstrated to us in week 1 and the course so far. Sebastian taking us through how we would break into an ATM and then patching those weaknesses is what really led to this project becoming what it is.

It started off as I want to lock a vault automatically. I could have used a password, or a command sent from my phone but those could be cracked through various methods, such packet-scanning, or even brute force. That's not to say that facial recognition doesn't have its own faults. I do like to believe that I attempted to patch those faults, however. Even if someone does have a way to beat the facial recognition system, they will be captured by the motion detection system, which will send you an email immediately. This can alert you to act, or even know who made an attempt to attack you and whatever you store in the vault, like Tony Stark Said, "if we can't protect the vault(Earth), you can damn well be sure we'll avenge it".

This project has been surprisingly hard in ways I didn't think it would be. The fact that I was trying to combine so many different things, like 3d printing, Arduino, Raspberry Pi, facial recognition and IOT may have made this project really annoying in terms of compatibility. A lot of my time was spent researching on how to do something and it almost took me a week to get facial recognition up and running properly.

Building the vault and mounting everything on it was something I underestimated, woodwork isn't my forte. I'm not content with the build quality of the vault but it was never really a priority .

Did I fulfill my learning goals?

Learn how to create active mechanisms through designing and 3D modelling

I completed this learning goal; The locking mechanism was something I didn't have problems with. While I didn't follow any real design processes and just winged it, I did gain some useful insight into creating active mechanisms for Arduino projects. I also learnt a lot about setting up my 3D printer in addition to fine tuning it. In the future I would like to create models in fusion360 which is far closer to an industry tool for modelling than blender.

Learn how to integrate Arduinos and Raspberry Pi's into one ecosystem. They must be able to communicate with each other and relay any input/outputs going through the system

This was an easy goal to fulfill through pyFirmata, it let me use the same commands I would use to program Arduinos on the Raspberry Pi machine. There were some limitations though, given further research, it seems that the button bouncing was definitely caused by issues in communicating between the Rpi and the Arduino. In the future I will find a better way to communicate between the two boards so that I can power everything I can off the Arduino, saving unnecessary processing expenditure on the Rpi.

Touch on some facial recognition and "Machine Learning"

The practical experience I got through using OpenCV on the Rpi has furthered my interest in machine learning and neural networks. Being able to perform facial recognition, a task I thought was extremely complex, on a Rpi has me far less intimidated about this branch of computer science. I've been interested in Artificial Intelligence since watching YouTube videos about neural networks in high school and this project has shown me that its within my capabilities. In the future I'll be looking into creating my own implementation of learning-capable neural networks instead of relying on other people's frameworks like I did for facial recognition this project.

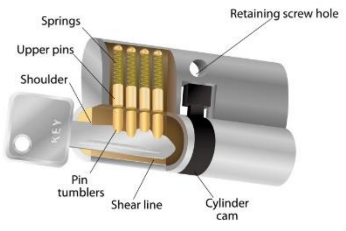
Learn how to implement IOT into my ecosystem and update the user through their personal device.

IOT has always been something that I've been hesitant to explore in my application. Integrating Rpi with Arduino made it easier since the Rpi is just a computer. To do IOT with just Arduinos it would require some sort of internet connectivity, usually through a Wi-Fi module or a board that is pre equipped with WIFI. I plan on revisiting the RGB lamp and aiming to create an app which will control its different RGB modes. I'm not entirely sure of the Arduino nano's ability to connect to Wi-Fi but I'm very motivated to look into it.

Designing the Locking Mechanism

EXAMPLES:

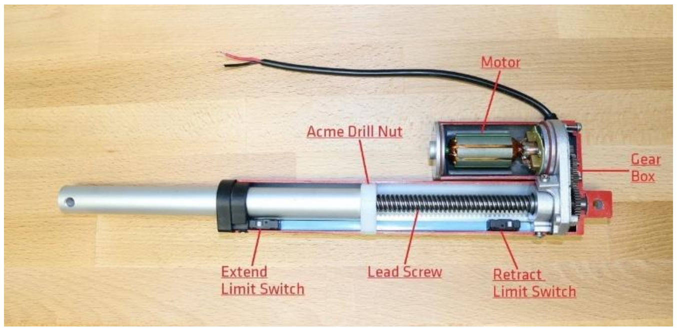
I can't make pin tumbler locks like the one in doors, they're too complicated and intricate to make with my commercial grade 3D printer as the tolerances for a mechanism like that would be way too small. It also would fit in with my auto lock mech using face recog.



Pin-Tumbler Lock

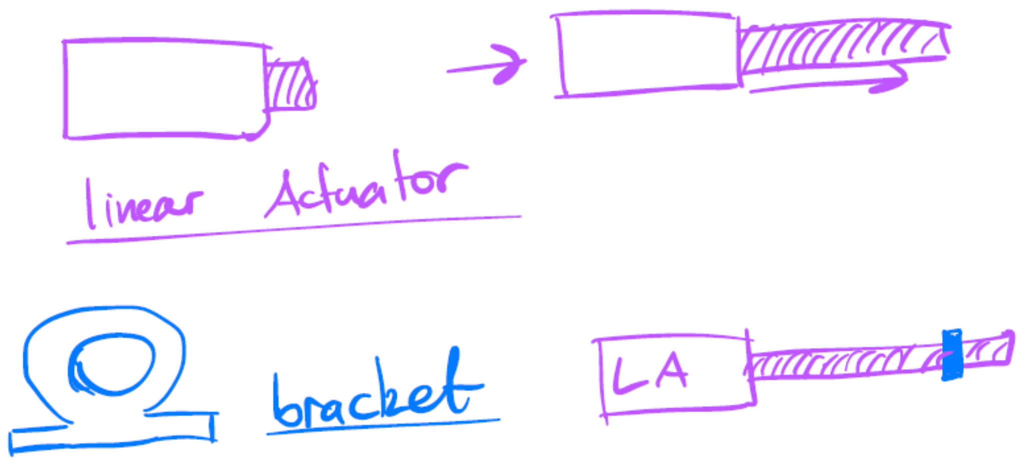
I will most like go for a sliding mechanism since its flexible in terms of how I can implement it. Some preliminary ideas are:

- Worm motor
- Linear actuator
- Servo with a gear attached to it
- Servo witch rotates a latch
- Slider moved with a stepper motor and belt



LINEAR ACTUATOR MECHANISM

Out of these the easiest seems to be the linear actuator, since the motor will extend itself and all I will need to make a bracket for the actuator to go into, simple.

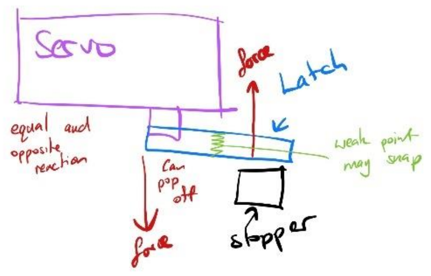


The simplicity of this mechanism means that I won't have to really design anything besides the holding bracket and maybe a mount for the linear actuator itself. This defeats the 3D modeling learning goal I want to achieve so this idea will most likely be scrapped. the power solution for the actuator is also needlessly complicated since it can't run off the Arduino board itself. They're also expensive and space inefficient for this project.

LATCH MECHANISM

The servo with a latch doesn't seem super reliable as the force placed on the servo will be perpendicular which may cause the latch itself to pop off the servo or snap due to the opposing forces on it. The

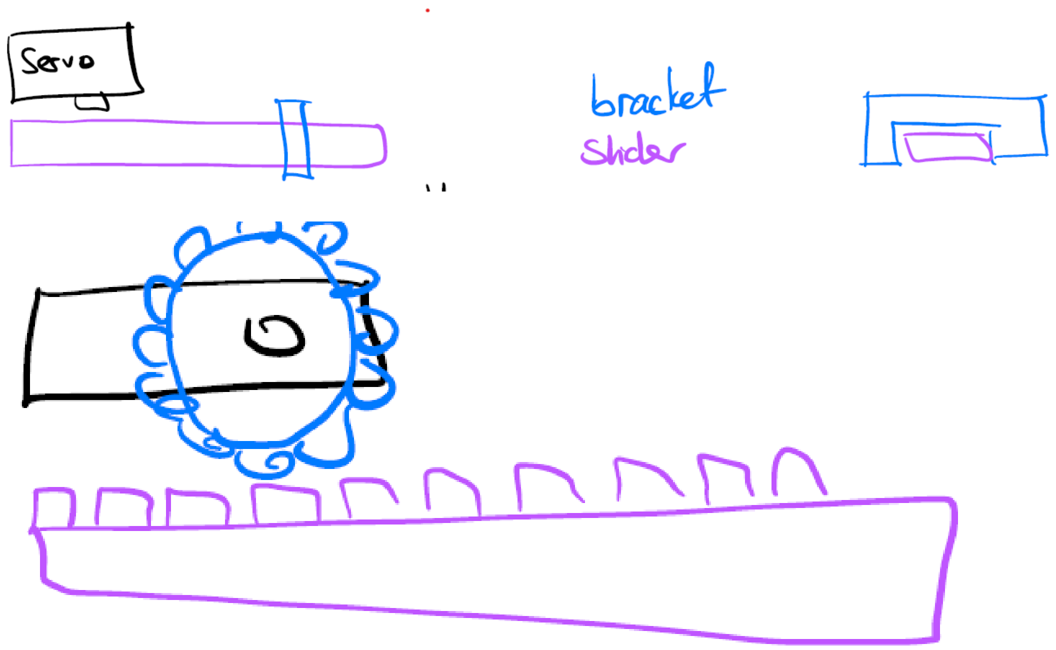
See diagram here



The unreliability of the solution and the fact that the design of the mechanism itself isn't complicated enough to meet the 3D modelling learning goal.

SERVO WITH A GEAR DRIVEN SLIDER

This design will drive a slider into a bracket using a servo. Once the slider is in its intended place it cannot be moved because the servo will keep it locked in its last position, as long as it has enough torque.

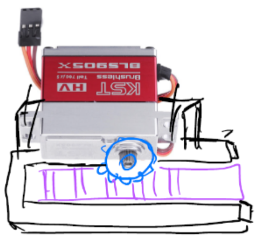


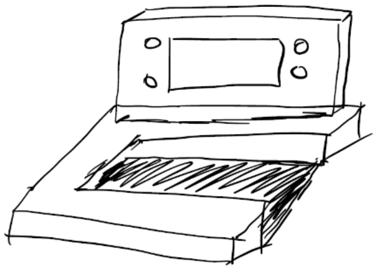
If someone attempts to open the door the lever will push against the bracket and the gear, both which should stop it from popping off. The force against the gear won't cause it to pop off, leaving the attacker to freely move the slider. While being relatively simple I think this is a good design to follow as its effective and allows me to meet my learning goals.

I will need to design a bracket/holder in which the slider will go into. I will also need to mount the servo somewhere, Its best to do these together as it will cut down on print times.

I also found a clip from TASM where he has a automatic door lock: https://vimeo.com/169107345

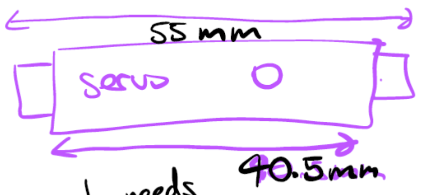
Below is my current solution to the problem:

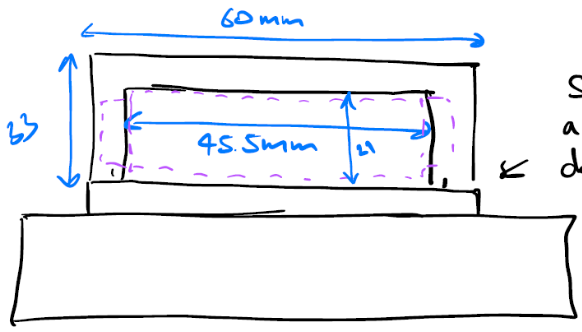




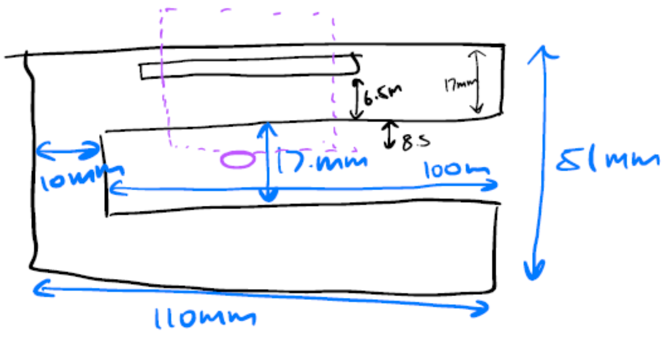
This design holds the motor against the slider and lets the slider move into the bracing bracket. I'm concerned that the mounting for the servo may snap if enough force is applied, however. Will have to investigate finding ways to strengthen it.

Started doing measurements according to servo dimension, really just guessed.
This is my servo: https://www.banggood.com/JX-PDI-6209MG-9KG-High-Precision-Metal-Gear-Digital-Standard-Servo-p-988833.html?rmmds=myorder&cur_warehouse=CN





Servo mount needs a solid base so it doesn't snap under pressure.



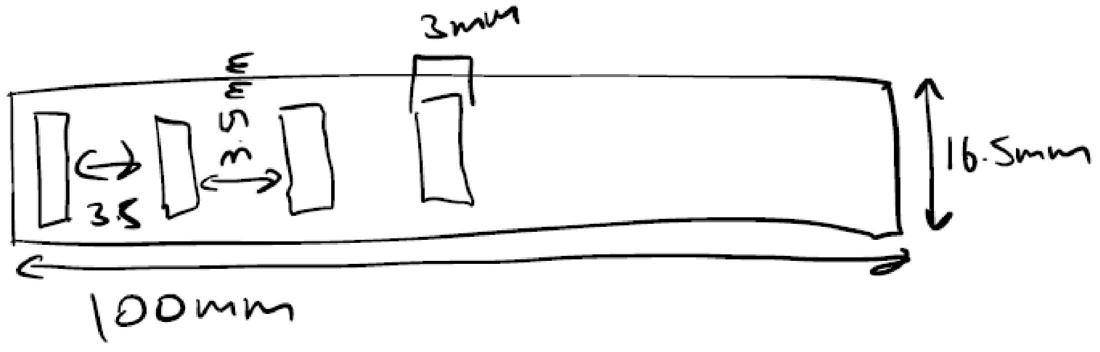
Distan of gear from servo mount must place it over the slider, for maximum support.

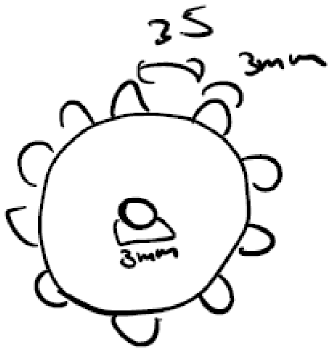
distance (mount + servo gear)

= 15mm

15 - 8.5
= 6.5

Slider



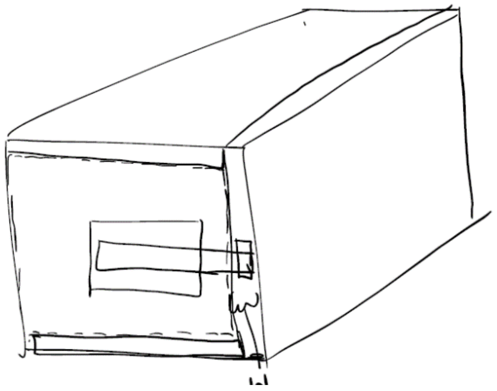


This is pretty much my final design; I will most likely play around with the measurements in the 3D modelling tool since I can bring everything together and kind of test fit it.

Designing the Vault

Like mentioned before I will not really be paying much attention to the design of the vault casing, it will most likely be made from wood and be a big square. I could make it out of metal, but it would make mounting hardware and routing cables through it so much harder. Metal also has the risk of shorting out components which I really want to avoid because not only are they expensive, but they take forever to get to me.

I could print out a vault, but the design is super simple and to get something really strong I would have to use a lot of printing filament (we will talk about this later) and time. The better choice is to go to wood.



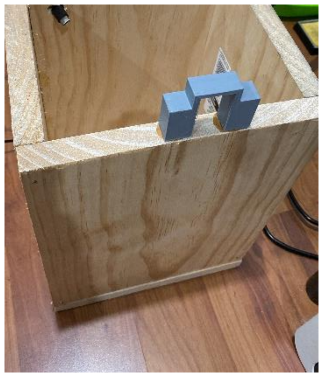
Here is an initial design, as you can see its pretty simple. If you're a professional woodworker on the side don't come for me, I know its janky but that's the theme of this project.

NOTE: coming back to this, janky really was an understatement.

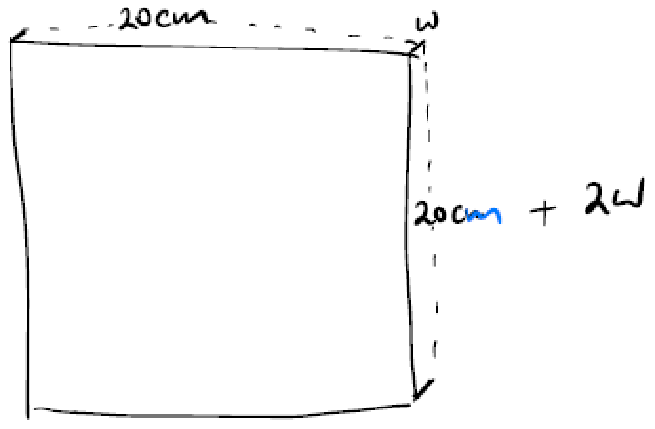
I head down to bunnings and got some wood, used a saw to cut it up and you can see the plans and results on the following pages.

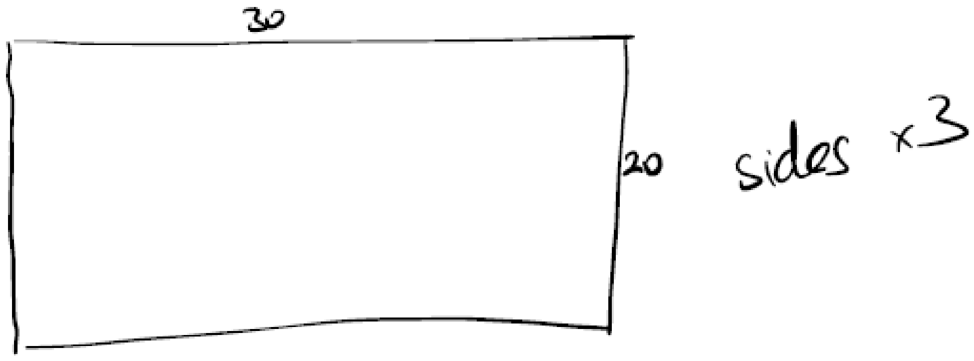
A lot of it is just guessing. When I got to bunnings they didn't have the size of wood I needed so I got the closest piece and worked with that instead.

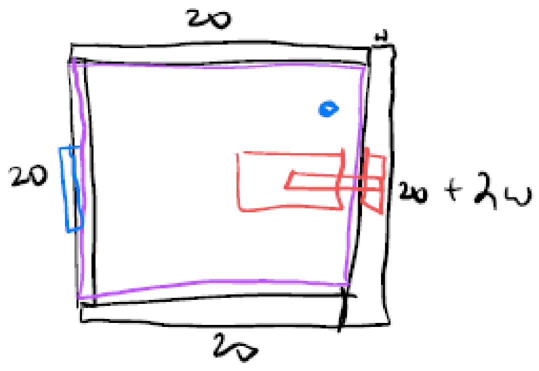
I ended up messing up the length of the side with the locking brace (left picture), but I have an elegant solution (right picture).

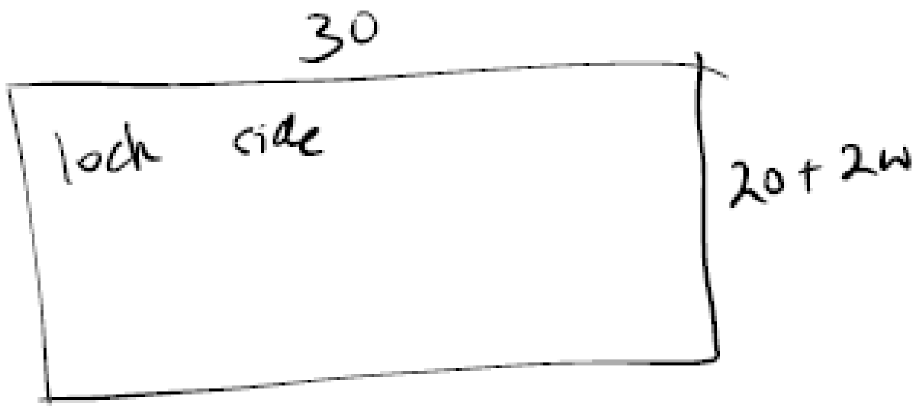


w≥1.5

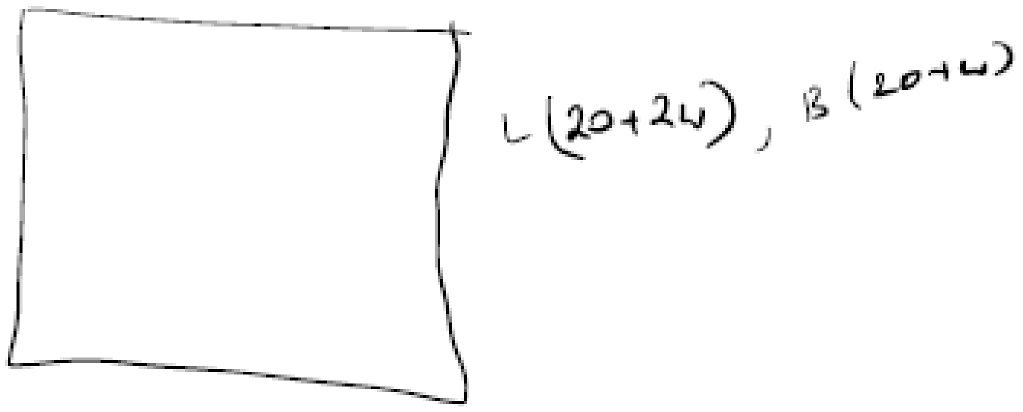




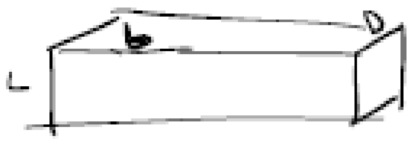




back



Side measurements



3x normal sides

if w = 1.5

30 x 20 cm

locking side

30 x 23 cm

front

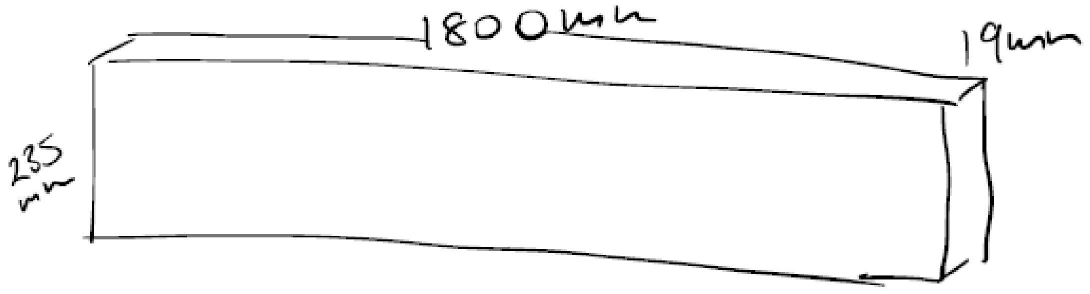
20 x 23 cm

back

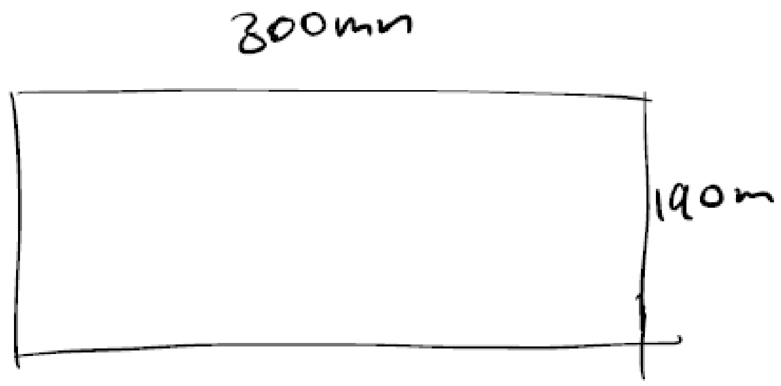
23 x 21.5 cm

Pine

235 x 19 x 1.8m
mm mm

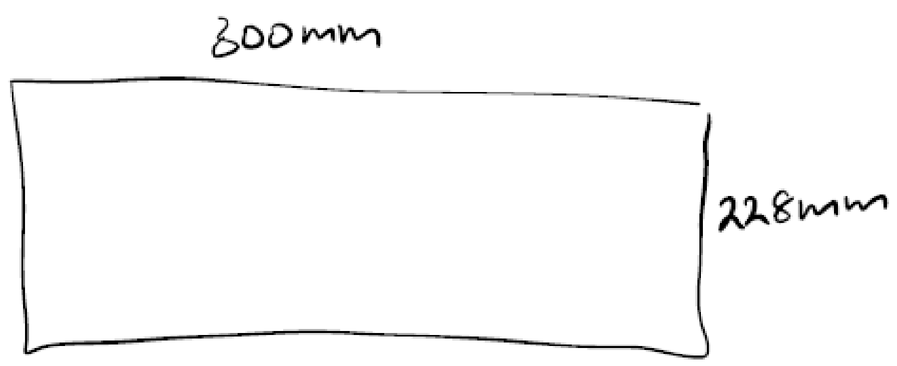


non lock



x3

lock

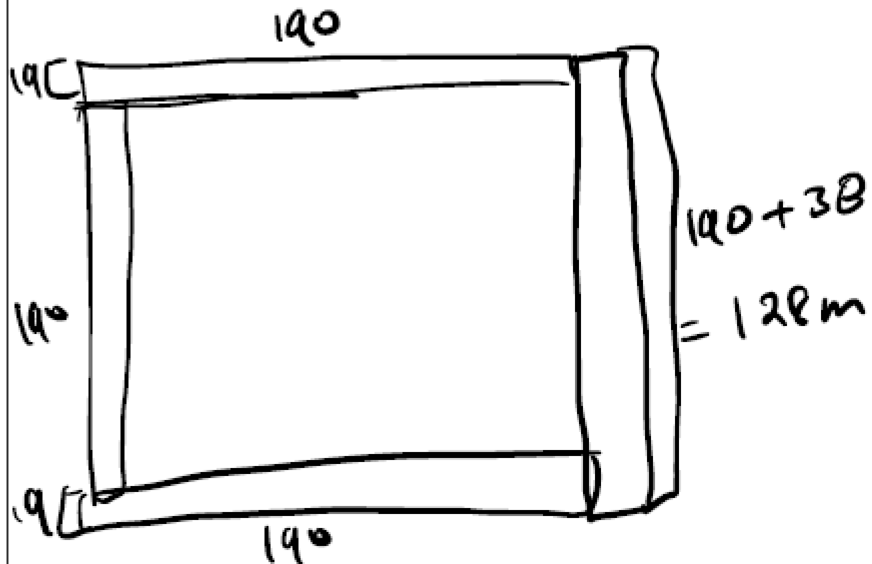


Back =

228m x 209

Front

228x190



Fixing issues with 3D printer

INFORMATION ABOUT MY 3D PRINTER

I have a Creality Ender 3 V2, I bought late December and I've had a ton of fun with it ever since. I've printed desk organizers, tool holders, an RGB Lamp, toys and best of all… ROCKTAPUS.

3D printing has a huge community so you can find a lot of tutorials and models people have made for free.

HOW DOES IT WORK?

A 3D printer extrudes(squeezes) melted plastic - usually around 200C – through a tiny nozzle onto a print bed. The nozzles movements are controlled by instructions generated by a computer. The nozzles lay down melted plastic line by line, layer by layer and as the plastic cools on top of each other it forms a solid object, like building Legos.

WHATS WRONG WITH IT?

Currently (at the beginning of this project) my printer isn't functioning. Most like my print bed is warped (print bed is what the printer squeezes) as my nozzle ran into it and damaged itself and possibly the print bed. I can replace the nozzle and the bed easily, but I do still have to fix the calibration issues.

I tried multiple time to align the bed, but the surface was uneven and had lots of high points and low points along it. My best option was to get an auto-leveler add-on for the printer, this does take a month to get here though.

Leveler is here, the cable they gave me wasn't long enough, so I got some small cables I had lying around and just soldered them together to make an extension because I didn't want to pay $50 for one. Turns out the cable I made has some issues since the leveler wasn't working and flashing red, a quick search online returned that it was an issue with the 5v pin. On inspection of the cable, I saw some exposed wire that may have been causing the issue. I headed down to jaycar and got some long cheap wire to solder into an extension instead

The process

Fig1 is the old cable, as you can see its just lots of smaller cables joined together is a sketchy way. The cable next to it is the replacement I will be using.

Fig 2 shows the new cable, it's a bit long but it works perfectly, and I can level the print bed properly using it.

How the auto-leveler works is it measures 25 points on the bed, tracks their variations and through software adjusts the position of the nozzle accordingly.

Fig 3 shows the generate mesh that came from leveling the printer. I did have to flash new software to my printer to get this to work but it was relatively easy.

The damaged nozzle was also replaced during this process.

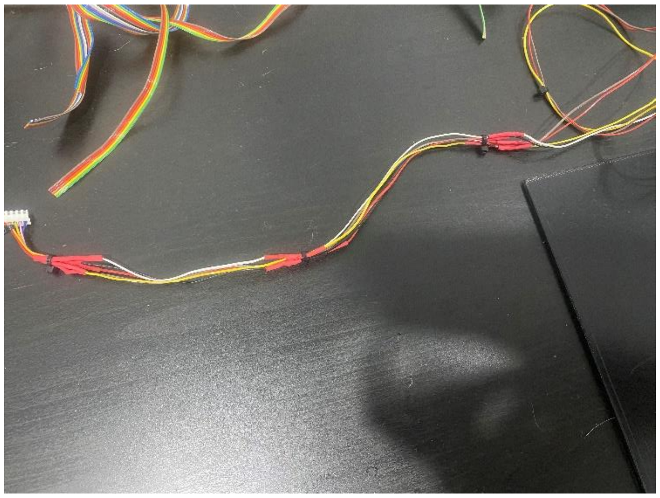


Fig 1 – old cable

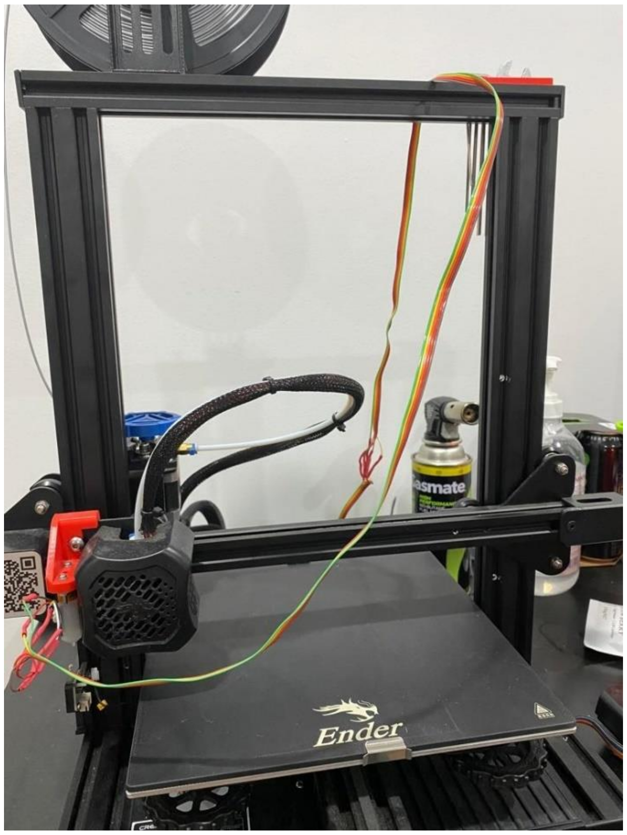
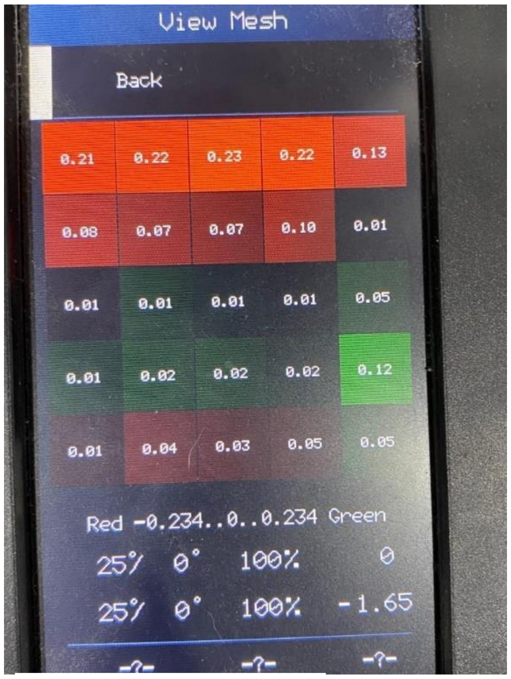


Fig 2 – new cable


View Mesh

Back

0.21	0.22	0.23	0.22	0.13
0.08	0.07	0.07	0.10	0.01
0.01	0.01	0.01	0.01	0.05
0.01	0.02	0.02	0.02	0.12
0.01	0.04	0.03	0.05	0.05

```
Red -0.234..0..0.234 Green
25%  0°  100%     0
25%  0°  100%  -1.65
-?-      -?-      -?-
```

Fig 3 – new cable

CHOOSING MATERIAL

Now that the printer is fully working again the materials used for it need to be determined. There are hundreds of materials available to be used with 3D printers and they all have their uses. The material used to 3D print is often referred to as filament.

Filament	Pros	Cons
ABS	Very sturdy and hard More flexible – easier to work with Suitable for machine parts Increased lifespan Higher melting point	More difficult to print. Heated required Prone to cracking if cooled too quickly. Can clog printer.
PLA	Can be printed on a cold surface, I do have a heated bed though. Environmentally friendly Smoother appearance	Can deform due to heat. Far less sturdy than ABS
PLA Plus	4x better resistance to stress than PLA. ALL the benefits of PLA. Superior layer bonding, meaning the filament sticks to itself very easily.	Can cause clogging in the printer. Higher temperatures needed to print.
Flexible	Prints are flexible and bendy.	Needs special components to print. Not that great when trying to make a sturdy mechanism.
PETG	Higher temperature resistance. Chemical resistance. Resists UV degradation	Sensitive to dampness and humidity, can compromise integrity.

Carbon Fiber Filled	Light weight and high strength.	Expensive. Hard to print. Need special components.
Metal blend	Metal finish.	Not actually strong as metal. Hard to print. Expensive.

From these options its fair to say PLA PLUS is best material for this project, its easy to print with, sturdy and already available to me, saving me money and time.

TUNING THE PRINTER

Now that the material is selected, we need to make sure our settings are correct, a 3D Benchy is used to stress test the printer, just like benchmarking a computer it measures where the printer needs improvement. You can find a guide on how to do this on the 3D benchy website. You can also upload your 3D benchy online to a forum and the community will be more than happy to help with what settings you need to change to get the best out of your 3D printer. Running the print for the first time this is my result (benchy on the left)

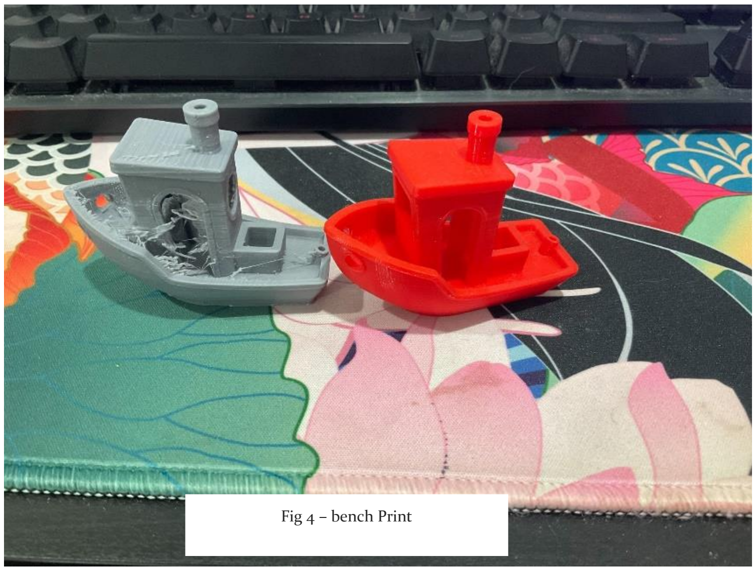
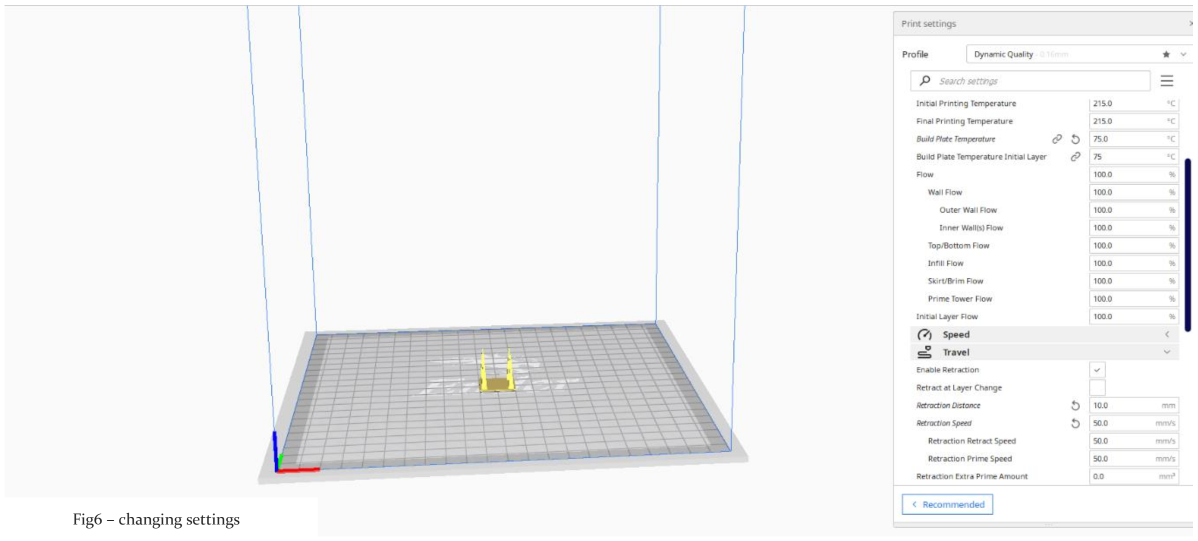


Fig 4 – bench Print

From FIG 4 you can see that the printed model has a lot of strings coming off it compared to the one on the right. This is called stringing and can be caused mainly due to retraction (the speed and distance at which the filament is moved withing the printer lines before being melted) or overheated, which makes the filament all goopy and causes it to string. How can we fix this?

Fortunately, you can change all these setting quite easily in your slicing application. The slicer is what takes a 3D model and converts it to a set of instructions a printer can follow. The slicer I am using is CURA.



Print settings

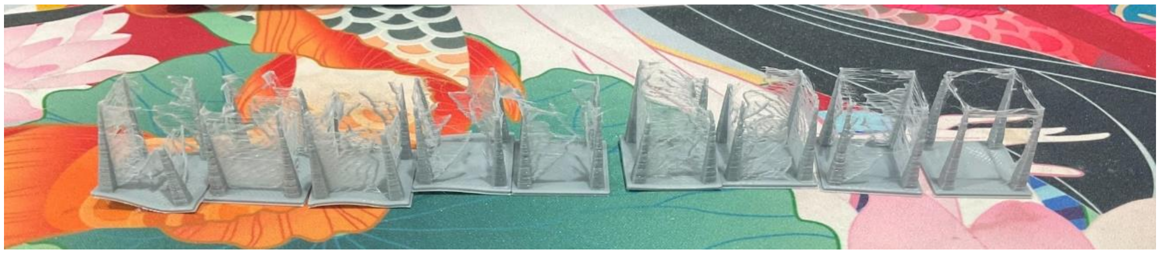
Profile: Dynamic Quality - 0.16mm

Initial Printing Temperature	215.0 °C
Final Printing Temperature	215.0 °C
Build Plate Temperature	75.0 °C
Build Plate Temperature Initial Layer	75 °C
Flow	100.0 %
Wall Flow	100.0 %
Outer Wall Flow	100.0 %
Inner Wall(s) Flow	100.0 %
Top/Bottom Flow	100.0 %
Infill Flow	100.0 %
Skirt/Brim Flow	100.0 %
Prime Tower Flow	100.0 %
Initial Layer Flow	100.0 %
Speed	
Travel	
Enable Retraction	✓
Retract at Layer Change	
Retraction Distance	10.0 mm
Retraction Speed	50.0 mm/s
Retraction Retract Speed	50.0 mm/s
Retraction Prime Speed	50.0 mm/s
Retraction Extra Prime Amount	0.0 mm³

Fig6 – changing settings

Through the menu seen above I was able to dial in my settings to reduce the amount of stringing in my prints. While I did not notice much change in print quality and reduction of stringing when changing retraction settings, I saw a major improvement when reducing the temperature from 220C – 215C, I didn't try to go lower since the PLA plus needs to be printed at 210C at a minimum.

Below are my results



The last print is where I saw the improvements from changing temperature settings.

Fig7 – benchmark prints

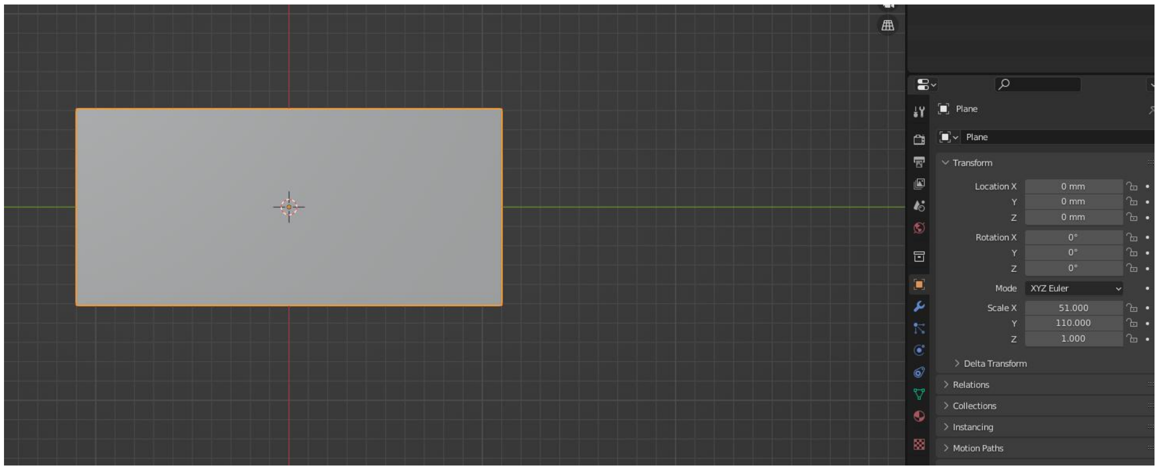
Modelling components

For modelling for this project, I will be using a software called blender. I have had experience with this software before in high school where I used it for animation. You can easily model objects in blender and then animate them using keyframing.

I plan to create a test fit of the locking mechanism on Blender and then animating it demonstrate how it would work

SERVO/SLIDER MOUNT

The servo and slider mount will be modelled based on the drawings found in “designing the locking mechanism” section.

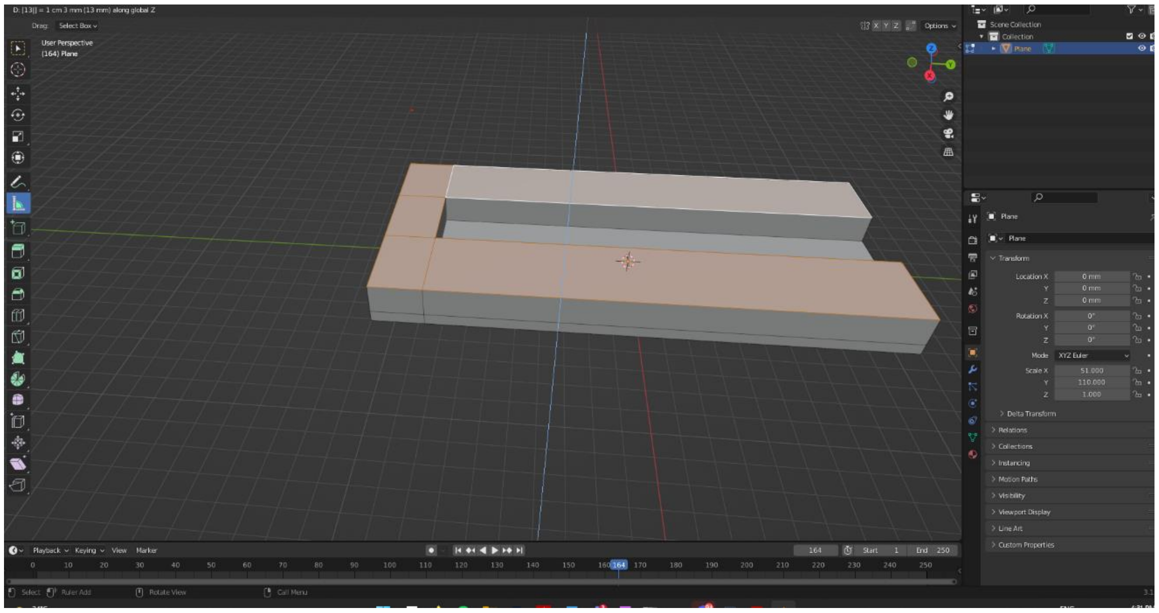


Plane

Transform

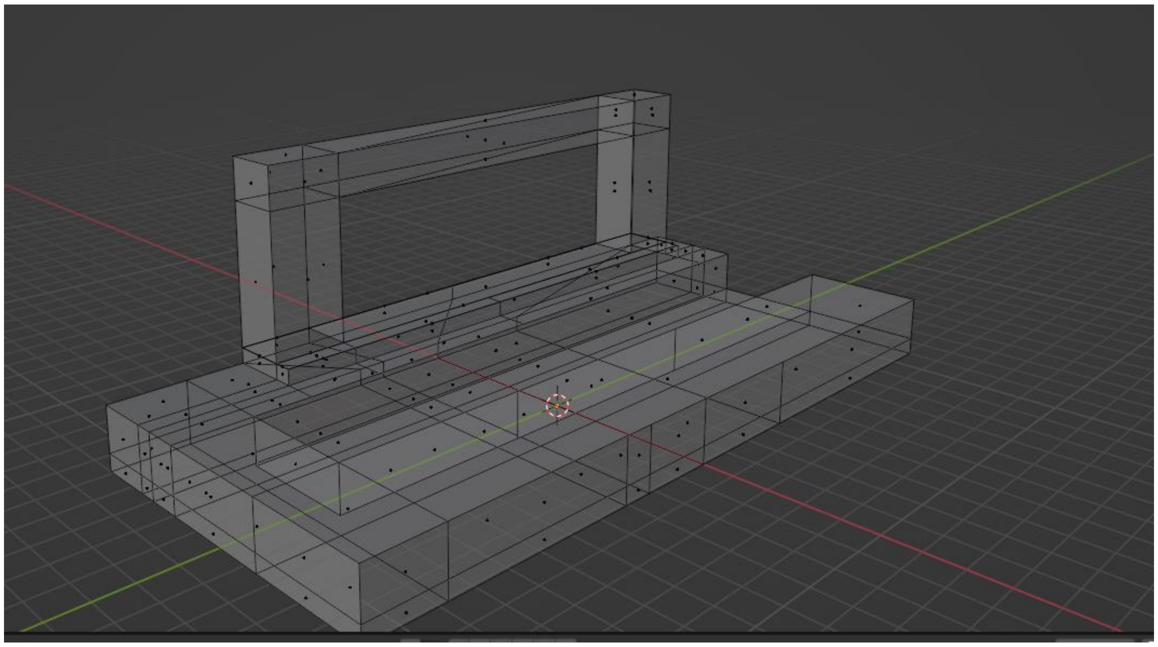
Location X 0 mm
Y 0 mm
Z 0 mm
Rotation X 0°
Y 0°
Z 0°
Mode XYZ Euler
Scale X 51.000
Y 110.000
Z 1.000

I started with a flat plane based on our measurements

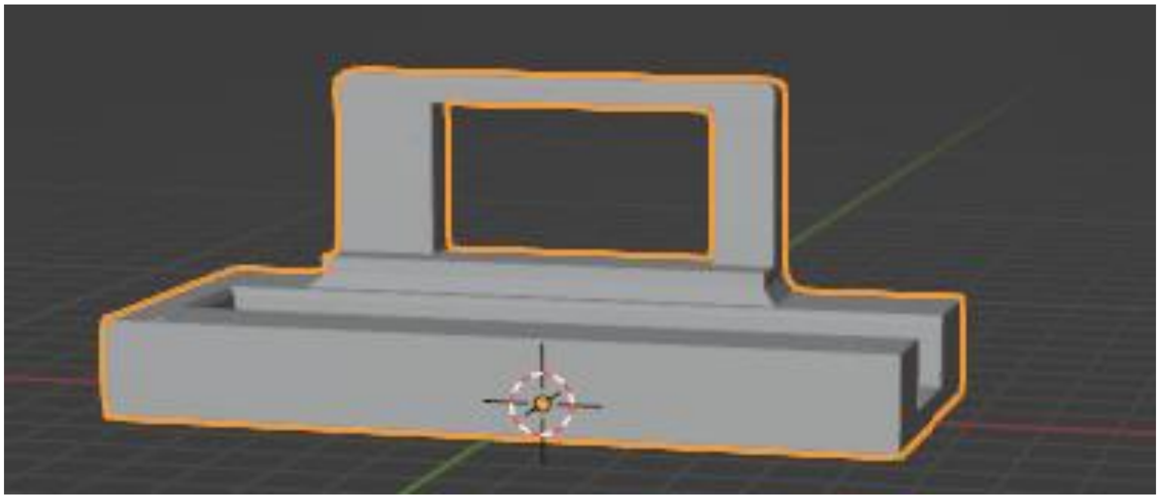


Location X 0 mm
Y 0 mm
Z 0 mm
Rotation X 0°
Y 0°
Z 0°
Mode XYZ Euler
Scale X 51.000
Y 110.000
Z 1.000

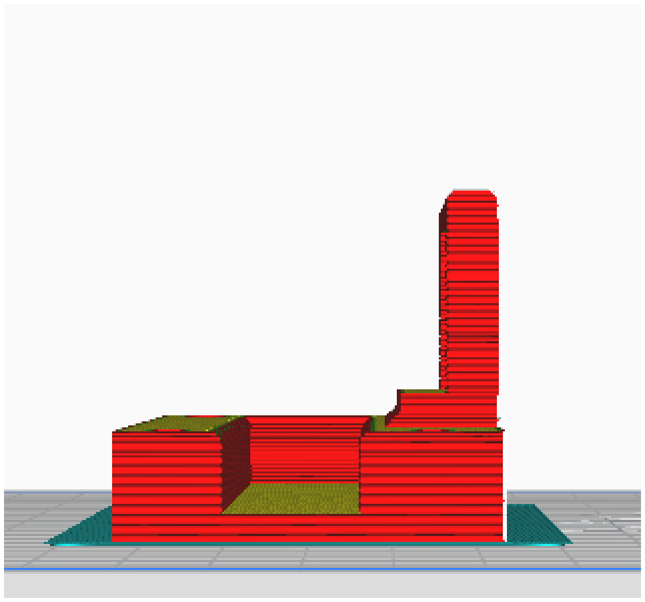
I then divided the top of the rectangle into sections, creating a channel for the slider to go into, I extruded all face up 5mm and then extruded all faces except for the channel for the slider. This resulted in the above object, and as you can see its already coming along. Now we need to model the mounting point for the servo, I can't just extrude two small sections up however since we need the mounting point to be sturdy.



I extruded a section of the top first and then extruded two pillars on either side, I extruded another 13mm off those pillars and then joined the faces to bridge the gap.



This was the final product, to strengthen the base of the servo mount I extruded and scaled the top of the extrusion down. This gave me a gradual curve, meaning there was more volume in the base, this it would be sturdier.

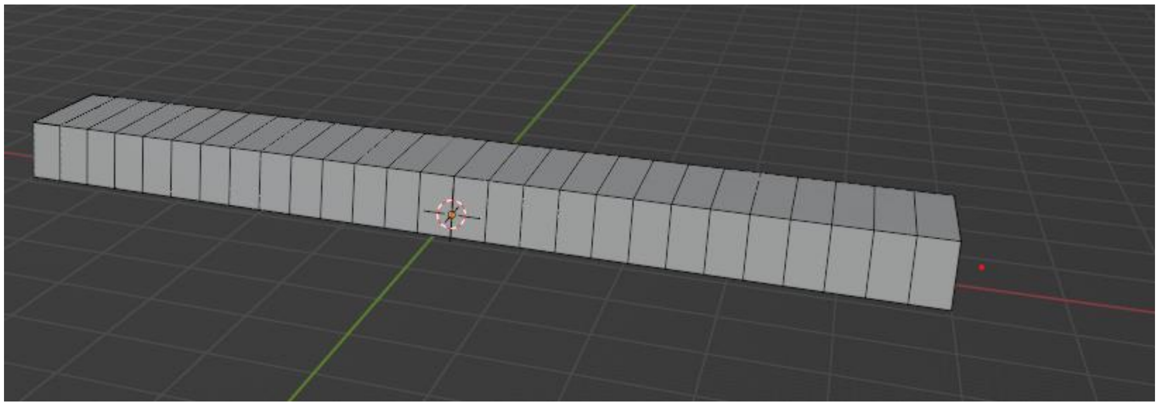


Once imported into CURA, I got errors saying that I need to print supports as the printer will not be able to bridge the gap at the top of the servo mount, this is an easy fix as I just select the add supports option.

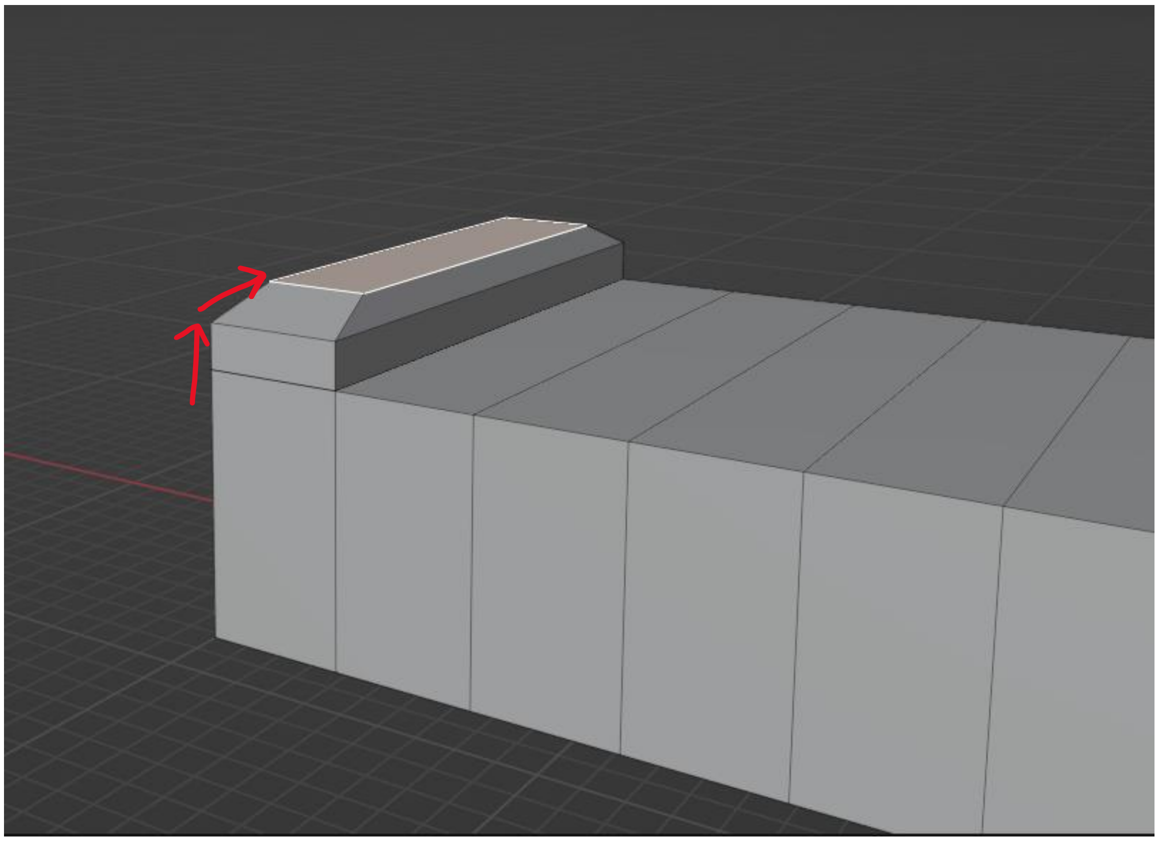
I also need to turn the infill up to 100 percent to make the mount as sturdy as possible. Infill determines how hollow the object is going to be, the higher the infill amount the denser, the lower the more gaps inside the walls. A higher infill amount takes much more time and materials to print.

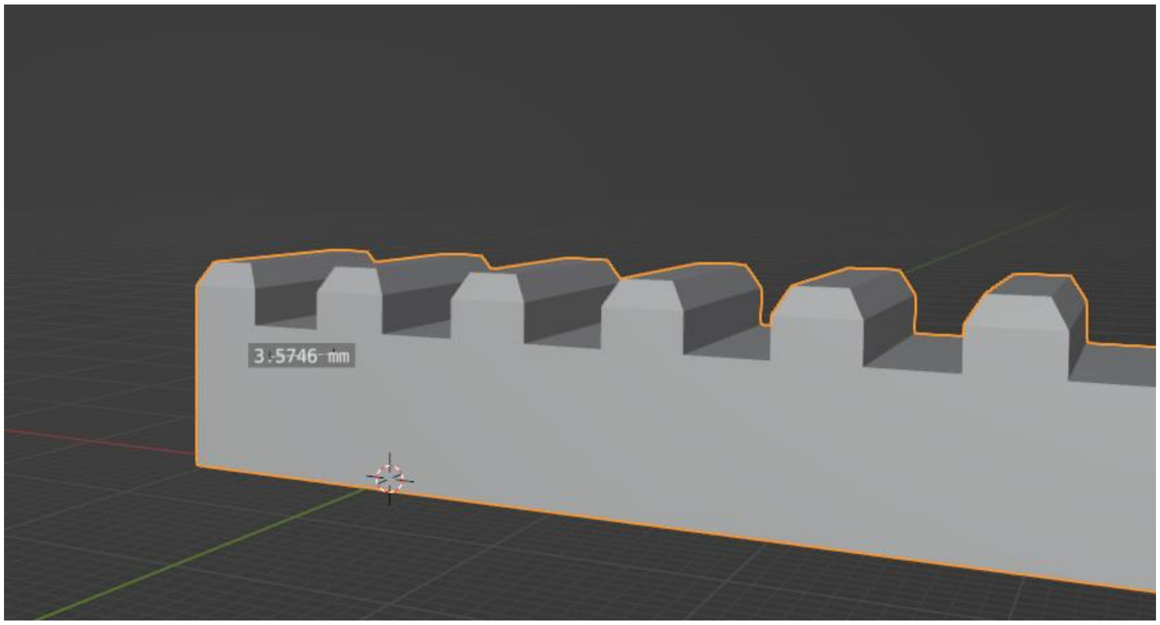
SLIDER

The slider is going to be a simple rectangle with some teeth for the gear to catch on to, we need to make the teeth gradually scale down so they can fit properly with the teeth on the gear. We can achieve this through extruding, and then extruding again and scaling the end face down a little bit.

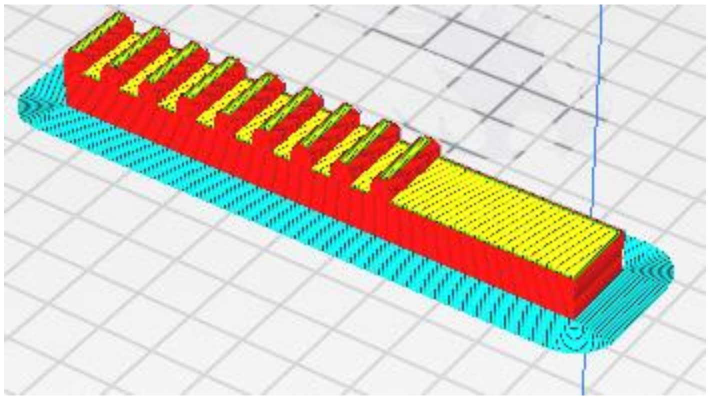


After extruding for the second time, I scale the face down and it gives a gradient on the tooth that will fit in with the teeth on the gears, from here on its just rinse and repeat until Ive made enough teeth on the slide. I won’t need to cover it all up in teeth since the servo wont travel along the length of the entire slide.





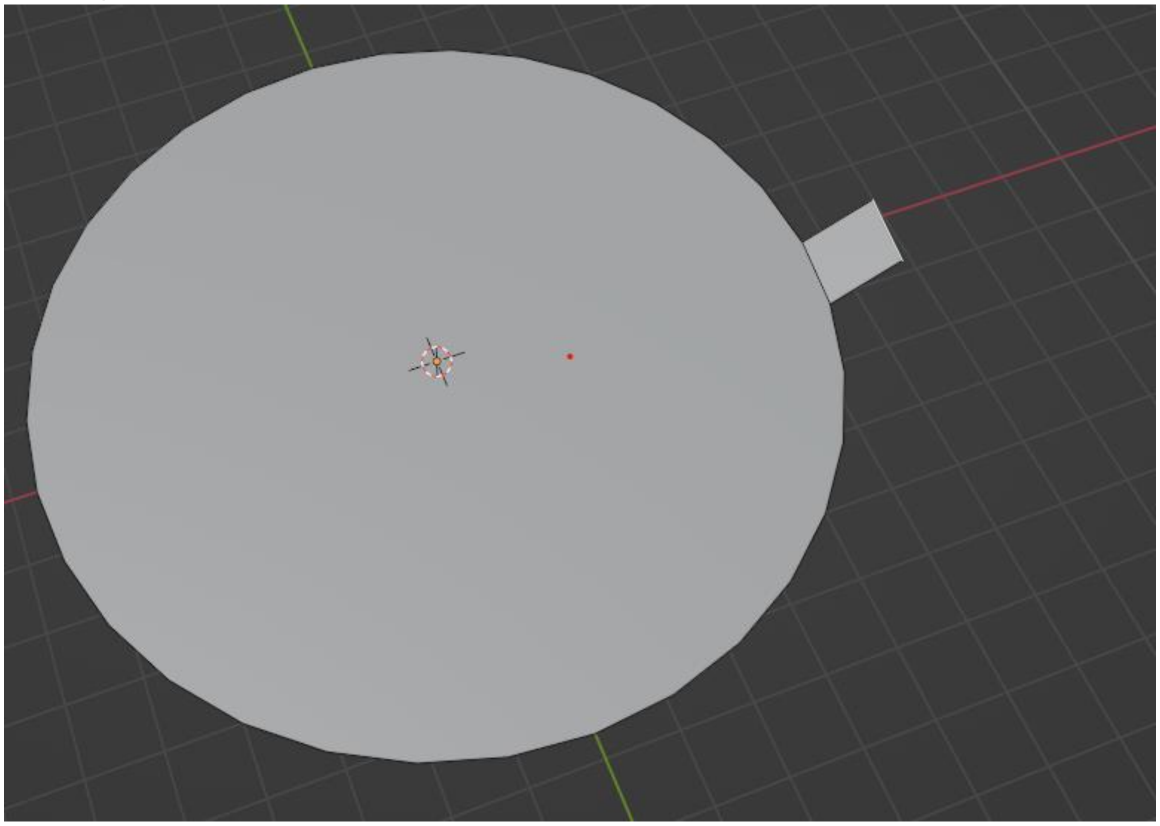
Final product. Ready for printing.

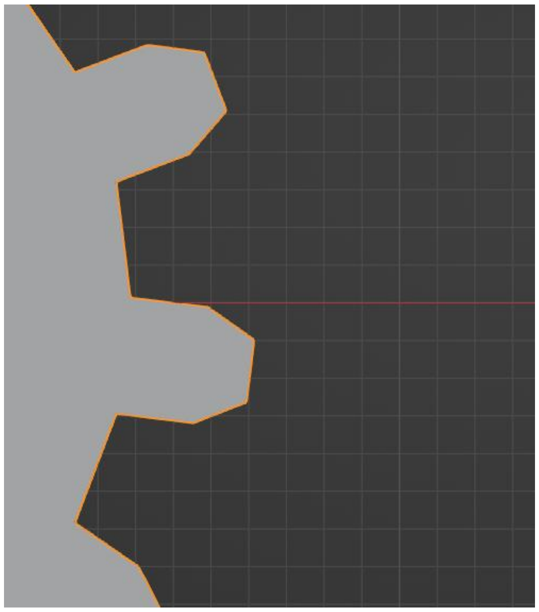


The blue outline around the slider is called a 'raft', this is printed on the first layer of the print so help keep the base layer stuck firmly on the print bed.

GEAR

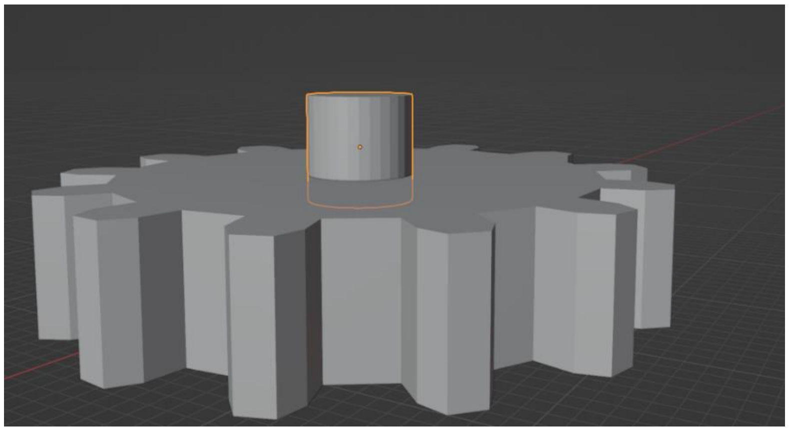
The gear can be made using a circle as a base. In blender a circle mesh (2d plane object) doesn't have a smooth circumference, its just a polygon with so many small sides that they almost give the illusion of a smooth curve. See below.

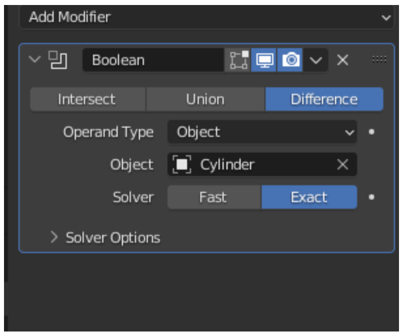




I extruded one of the edges from the "circle". This is how we will create the teeth for out gear, extruding each edge, then scaling it down to give it a gradient that will match the slider.

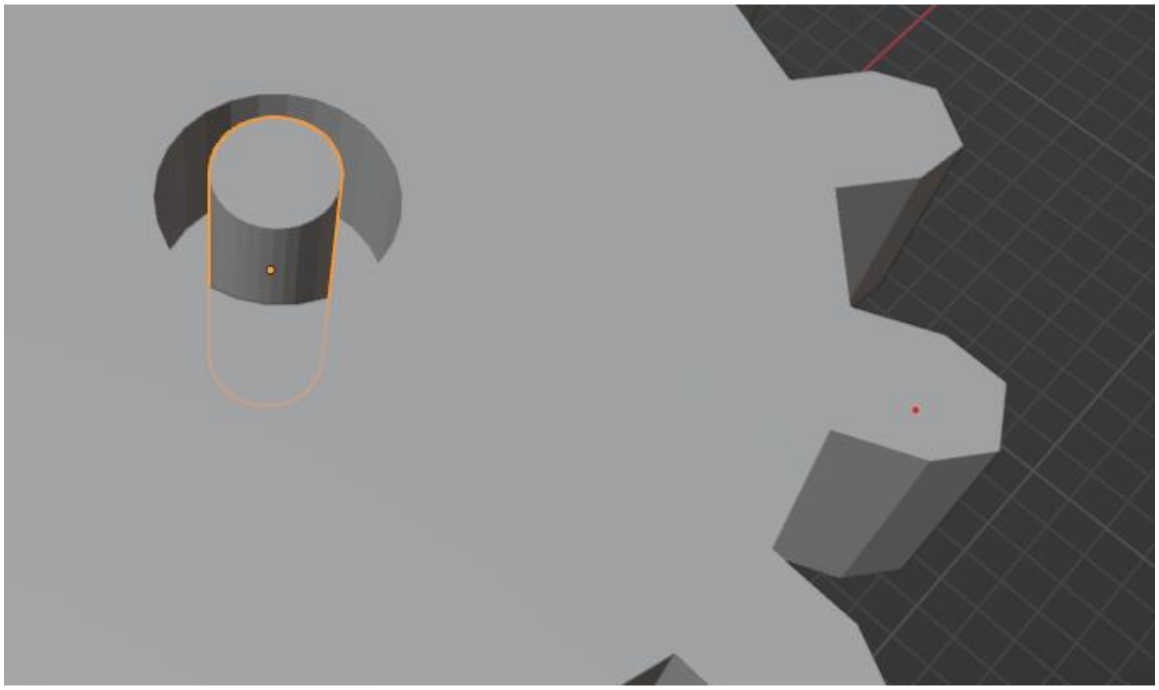
I now need to make a hold for the drive shaft of the servo to go in. I can do this through object modifiers.



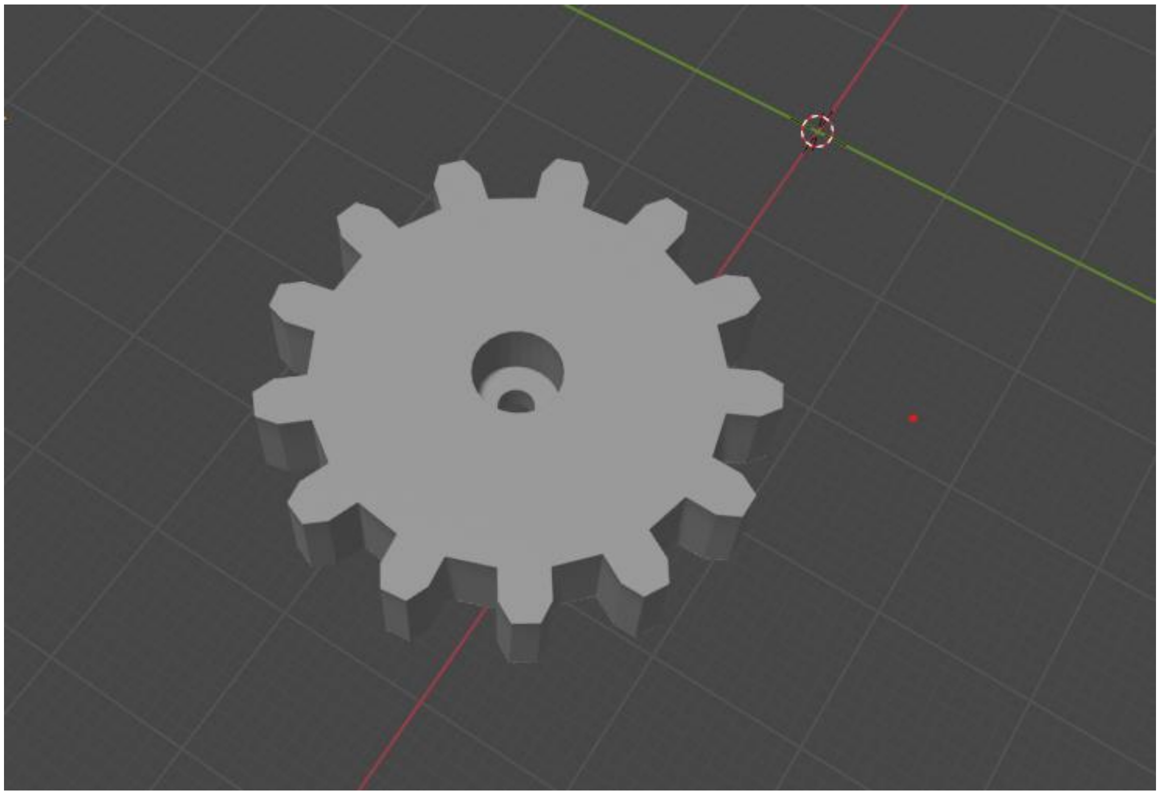


Operand Type: Object
Object: Cylinder
Solver: Fast [✓ Exact]

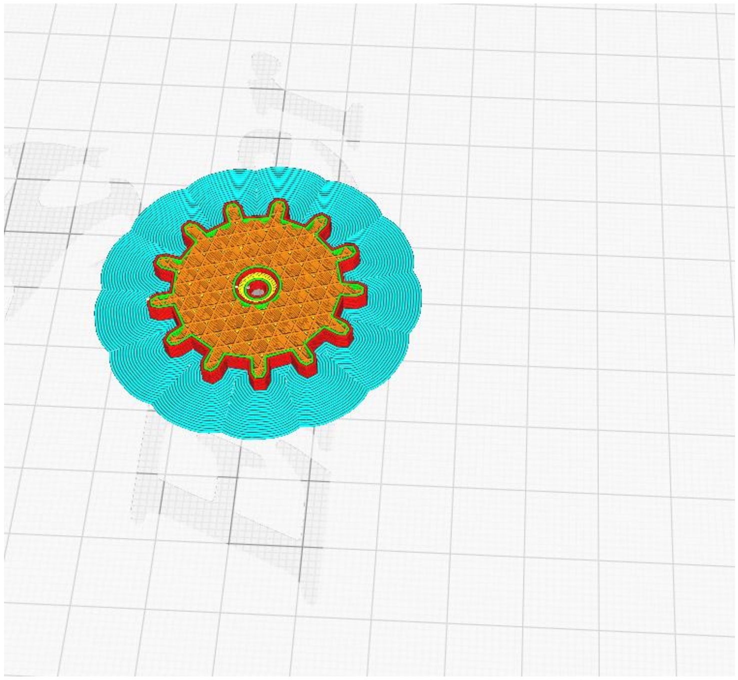
What the Boolean modifier allows me to do is cut a hole in the gear the size of that cylinder. I don't want the hole to go all the way through for the shaft so only 1/3 of the gear is being cut through.



Now that the larger hole for the drive shaft has been made, I will create one for the screw that connect to the drive shaft using the same method.



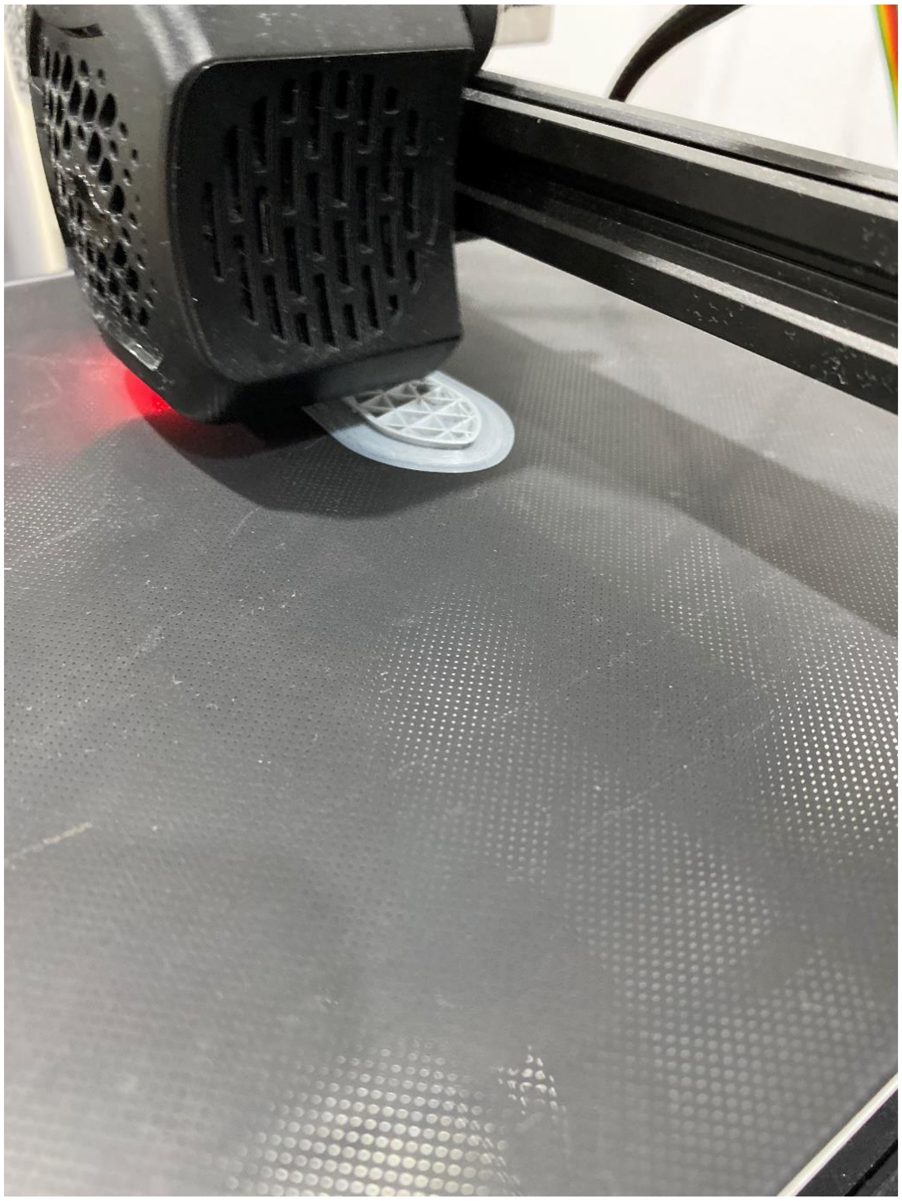
Ready to be printed.



I don't have here is an example of the inside of a print, this is when infill is at 50%, you can see there are supports inside the walls of the print but there are also gaps between them, setting the infill at 100 would cause the entire model to be a solid, no gaps.

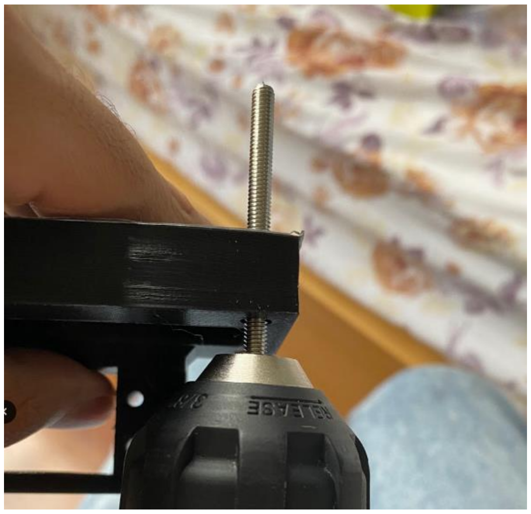
PRINTING

I lost the timelapses from my gopro of the printing, I do however have a picture of a benchy being printed to further demonstrate infill.



Assembling Locking Mechanism

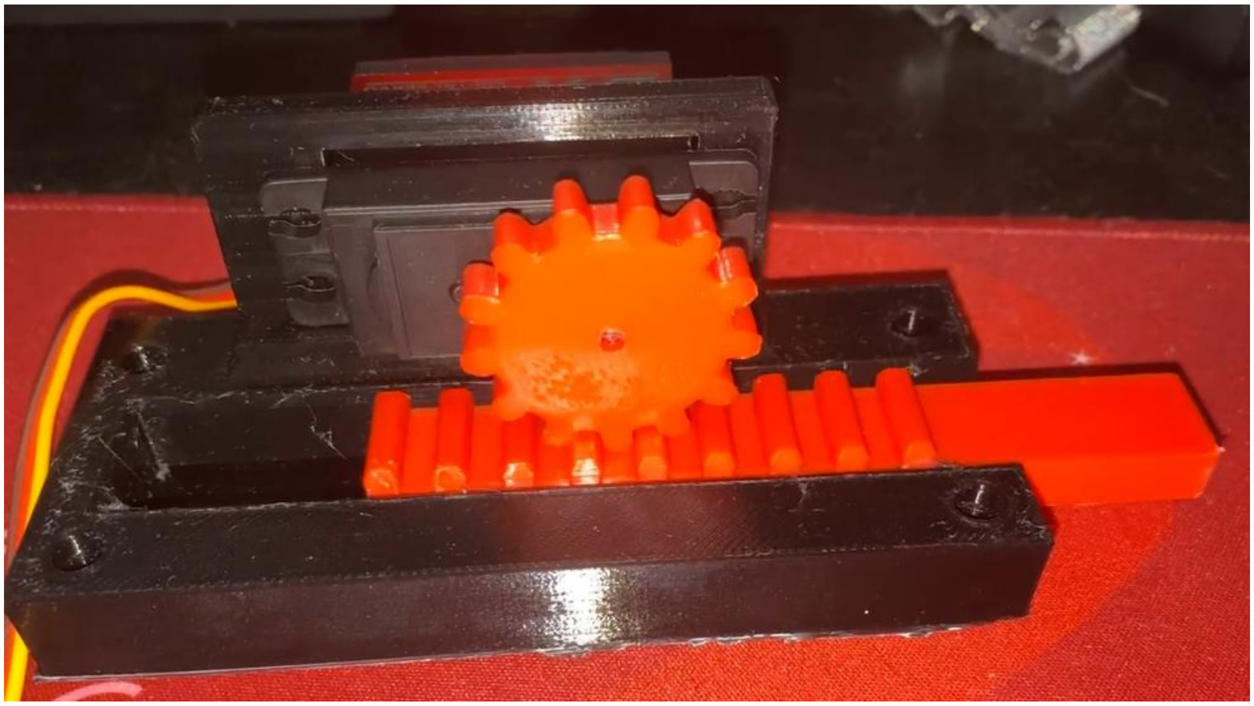
The Mount for the slider and servo needs to have holes drilled in it first, this wasn't done during the modeling phase since it most like would not have been accurate, especially with the principles of "measure once, cut twice" this project has been adhering to.



I first drilled the holes with a normal drill bit and the drove m5 and m3 bolts into them to thread the holes, this was janky but very effective.

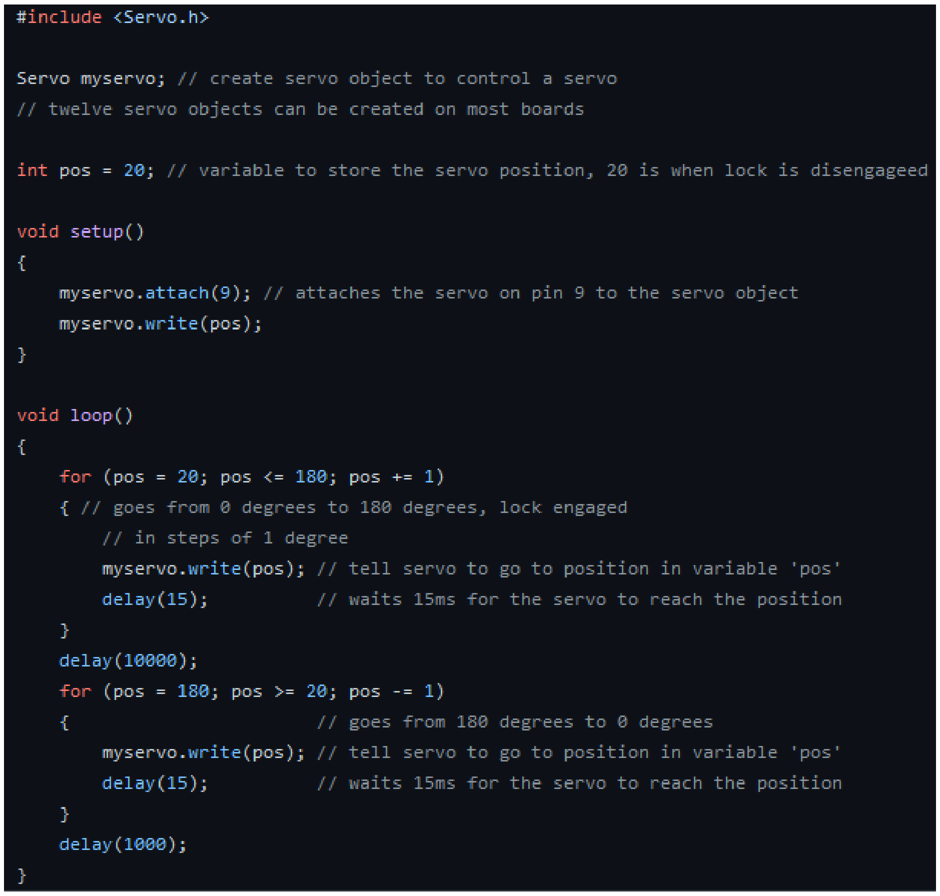
M5 bolts were for mounting the base onto the wooden vault and the M3 bolts were to mount the servo.

Once this was done, all the components were put together and the servo mounted .



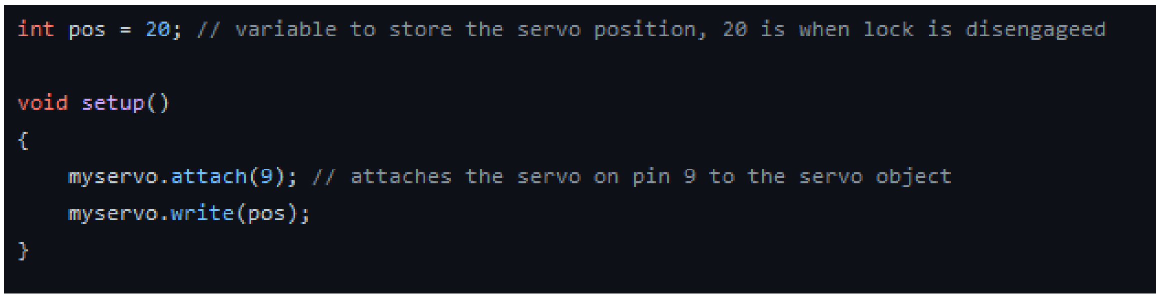
Everything fit well enough, I had to the cavity for the servo for it to fit properly but otherwise there weren't really any issues. The teeth on the gear fit well enough with the slider, which was the greatest worry.

Driving the Servo with an Arduino



```
#include <Servo.h>

Servo myservo; // create servo object to control a servo
// twelve servo objects can be created on most boards

int pos = 20; // variable to store the servo position, 20 is when lock is disengageed

void setup()
{
    myservo.attach(9); // attaches the servo on pin 9 to the servo object
    myservo.write(pos);
}

void loop()
{
    for (pos = 20; pos <= 180; pos += 1)
    { // goes from 0 degrees to 180 degrees, lock engaged
        // in steps of 1 degree
        myservo.write(pos); // tell servo to go to position in variable 'pos'
        delay(15);          // waits 15ms for the servo to reach the position
    }
    delay(10000);
    for (pos = 180; pos >= 20; pos -= 1)
    {                       // goes from 180 degrees to 0 degrees
        myservo.write(pos); // tell servo to go to position in variable 'pos'
        delay(15);          // waits 15ms for the servo to reach the position
    }
    delay(1000);
}
```

CODE EXPLANATION

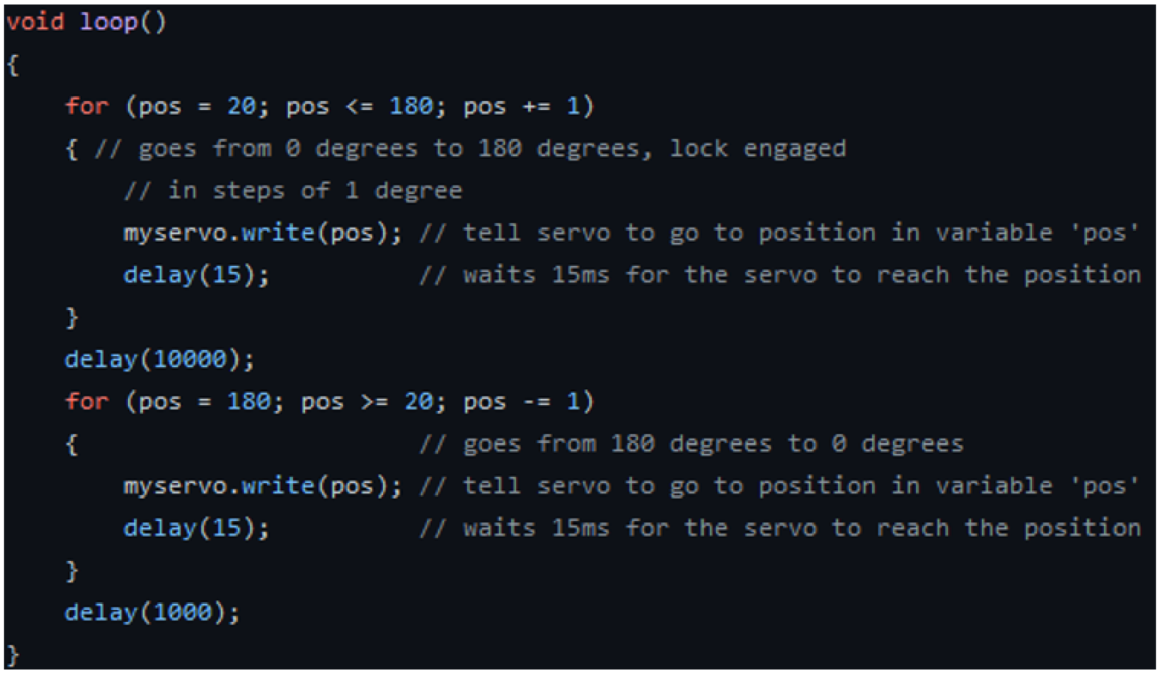


```
int pos = 20; // variable to store the servo position, 20 is when lock is disengageed

void setup()
{
    myservo.attach(9); // attaches the servo on pin 9 to the servo object
    myservo.write(pos);
}
```

Start the code with the servo positioned at 20 degrees, also the position where the lock is open.

Void setup{} is run once at the start of the program and this is where you set the initial state of your program and tell the Arduino which component is connected to what port.

myservo,attach(9) Tells the Arduino that the servo is connected to port 9 on the board
myservo.write(pos) Sends the command to move the servo to 20 degrees.



```
void loop()
{
    for (pos = 20; pos <= 180; pos += 1)
    { // goes from 0 degrees to 180 degrees, lock engaged
        // in steps of 1 degree
        myservo.write(pos); // tell servo to go to position in variable 'pos'
        delay(15);          // waits 15ms for the servo to reach the position
    }
    delay(10000);
    for (pos = 180; pos >= 20; pos -= 1)
    {                       // goes from 180 degrees to 0 degrees
        myservo.write(pos); // tell servo to go to position in variable 'pos'
        delay(15);          // waits 15ms for the servo to reach the position
    }
    delay(1000);
}
```

Void loop{} runs infinitely in the Arduino, unless told otherwise.

The first for loop slowly takes the servo from 20 degrees to 180 degrees, which is closed

The program then uses delay, which is equivalent to sleep, for 10000 milliseconds (10 seconds). The program then slowly returns back to 20 degrees which is the open position.

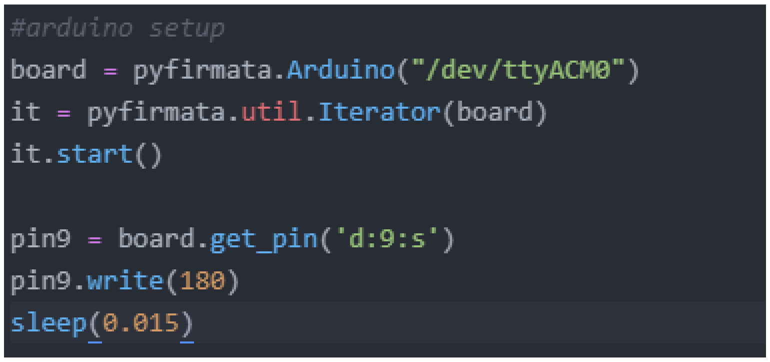
Here is the video of it operating.

Integrating Arduino with Raspberry Pi

The pyFirmata library will be used for the RPi to send instructions to the Arduino.

In Linux you must first install the module using “pip install pyFirmata”
This will install the pyFirmata library and allow you to import it into your files

Import as “import pyFirmata”



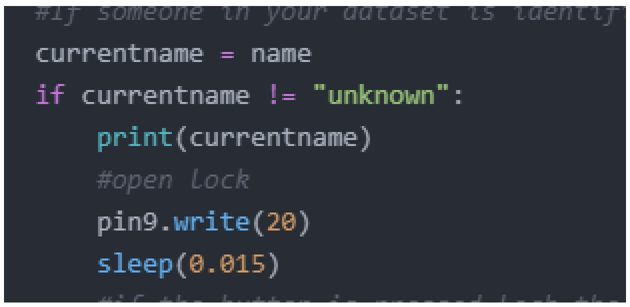
```
#arduino setup
board = pyfirmata.Arduino("/dev/ttyACM0")
it = pyfirmata.util.Iterator(board)
it.start()

pin9 = board.get_pin('d:9:s')
pin9.write(180)
sleep(0.015)
```

You then need to setup your Arduino board. Im connecting the Arduino to the Rpi through USB so the board can be found at “/dev/ttyACM0”.

You then need to create the board object that can be found at path “board”, using
it = pyfirmata.util.Iterator(board)
and then initialize it using
it.start()

Just like we did on the Arduino, we need to tell the Rpi where the servo can be accessed from.
pin9 = board.get_pin('d:9:s')
Tells us that we can find the servo at pin9 on the Arduino “board”.

Just like the write command on the Arduino we use
pin9.write(180) to set the servo to a closed position at 180 degrees.
we then need to send a sleep command for 15ms, enough time to get the servo moving.



```
#if someone in your dataset is identif
currentname = name
if currentname != "unknown":
    print(currentname)
    #open lock
    pin9.write(20)
    sleep(0.015)
```

This is the code for when the Rpi finds a match on facial id. We will get to this in the facial id section.

RPI AND FACIAL RECOGNITION

I was able to get starter code and instructions on how to use it and OpenCV from the Core Electronics webpage, here is the link.
I did not develop the facial recognition software myself as it would have been far to complicated and that alone would have taken too much time. Instead, I used the implementation created by Tim from Core and integrated my own uses into it.

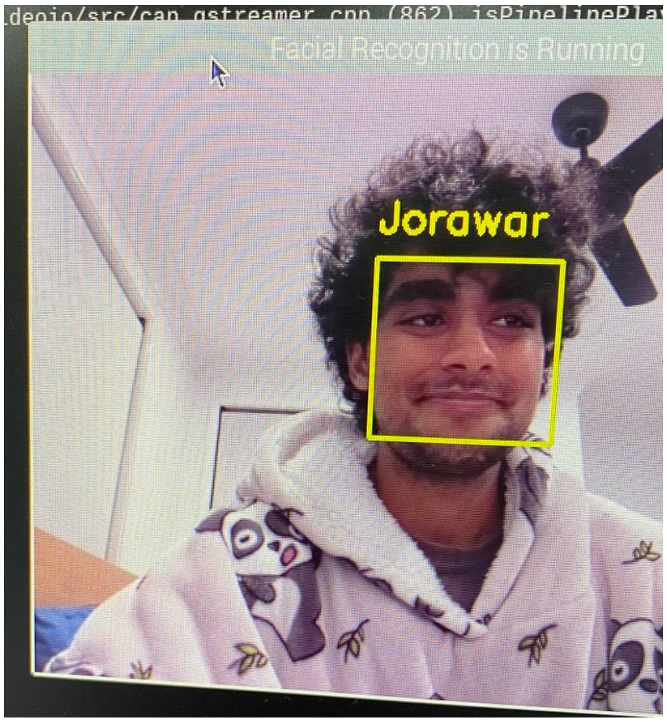
Initially I had a lot of issues installing the modules and dependencies required for this method and it took hours of scouring through forum posts to find alternative modules that would work. IMUTILS was a module that gave me a lot of grief, it would keep failing after each reboot even when ran in python virtual environments. Eventually I found another way to install inutile which stopped a lot of my problems.

Another issue I found is that the "facial recognition" package is not compatible with python3, so I must make sure the Rpi runs python2.7 when it auto starts the program.

TRAINING THE FACIAL RECOGNITION SOFTWARE

As per instruction on the core electronics page I first trained the software to recognize my face. This was done take about 50 image of my face and tried to vary the angles for the pictures. The program then processes those images and creates a map of my face. A major flaw in the system is that it analyses 2d images, so unlike apples FaceID it can be fooled by a simple picture of the user. Maybe this can be trained out of it??

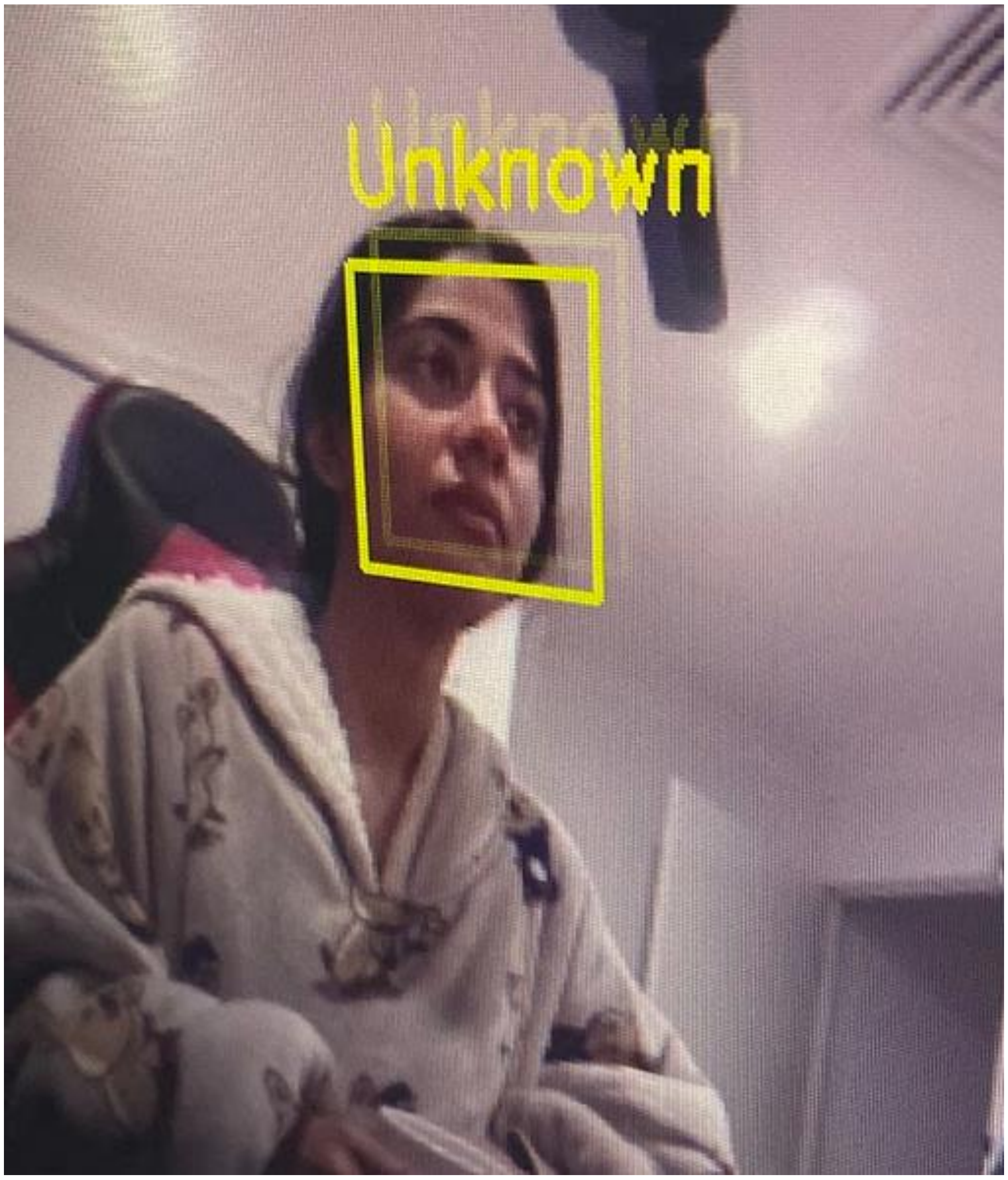
After training it for my face I ran the software and behold!! It knows my face.


.deoio/src/cap gstreamer cpp (862) isPipelinePla
Facial Recognition is Running
Jorawar

This was pretty rewarding after spending all day trying to get the software to work and jumping through so many hoops to stich this program together.

I asked my dad to come give it a try, and it recognized him as me. This was pretty concerning, and some further research revealed that I will have to train the software against faces that aren't mine in order for effectively differentiate faces that may be similar to mine.

On a sidenote, when I asked my sister to sit down Infront of it, the software didn't recognize her so the issue may have been that my dad just looks like me.



I found a dataset of faces from here, marked them as unknown in the file and processed them with the OpenCV software, after this it seemed like it only recognized me as "Jorawar" which is the intended behavior.

EXPLANATION OF CODE

Facial Recognition

Since there is a lot of code for the facial recognition software, and it has already been commented I will not go through it line by line. I will however provide a summary to demonstrate that I adequately understand how the software functions.

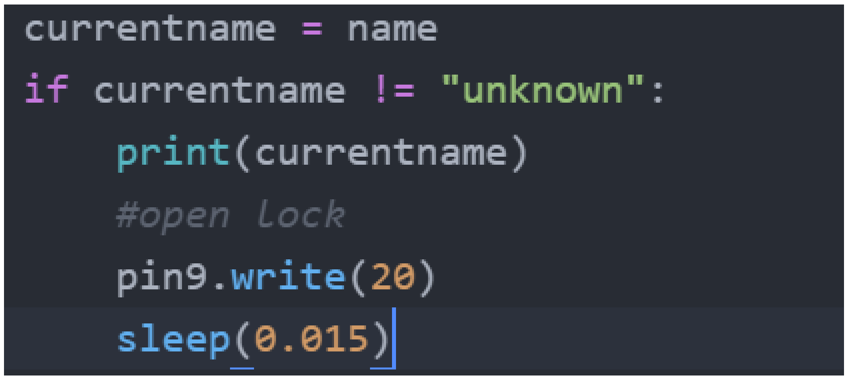
The software performs a search frame by frame (expensive on processing power). It runs a compare function “matches = face_recognition.compare_faces(data["encodings"], encoding)” which goes through the known faces which it has previously been trained to recognize.

If the compare function returns a match, it then needs to determine which face has been matched a greater number of times. This is in the case where there are multiple people in frame, and it needs to determine which face to take as the predominant one. This decision for the dominant user is since they have been in the frame longer, does not mean they are they actual user.

After getting the name of the face the system considers dominant, it checks if it is unknown. This is where our lock opens.

Moving a Servo when a Face is recognized

If the dominant face is in the list of authorized faces, we want the lock to open.



```
currentname = name
if currentname != "unknown":
    print(currentname)
    #open lock
    pin9.write(20)
    sleep(0.015)
```

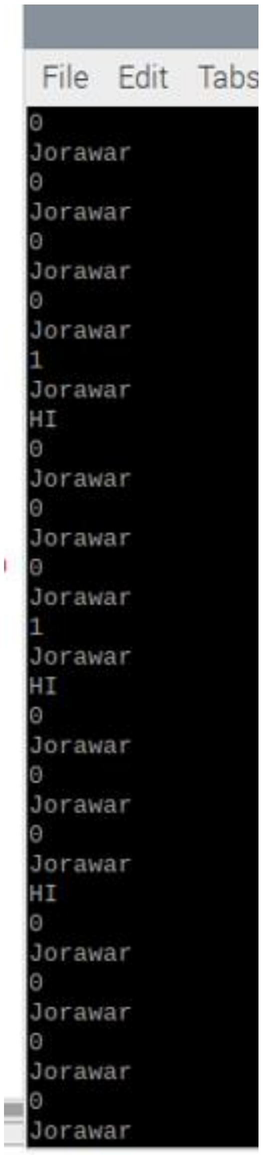
This is how we achieve that, if the name of the face is identified and in the list of recognized faces, we set the lock to open position.
this is pretty simple, and we’ve done this before in the initialization step as well.
Here is an example of the code working like above, this code also had a step where if there was no face detected the lock would close automatically. This was removed because when the door to the vault opens the lock would automatically close before the door was shut again.
Now we want to investigate how we can close the lock when the user of the vault is done with.

BUTTON INPUTS

Initially there was a button sending inputs to the Arduino. The signal was then sent from the Arduino to the Rpi. Both the servo and button were connected to the Arduino now.

Button Bouncing

A major issue encountered when button input was taken by the Arduino was BUTTON BOUNCE. This is when there is electrical interference in the circuit driving the button which causes presses not to be registered or false signals being received by the Arduino. This cause problems as it would close the lock when it wasn't meant to be closed.

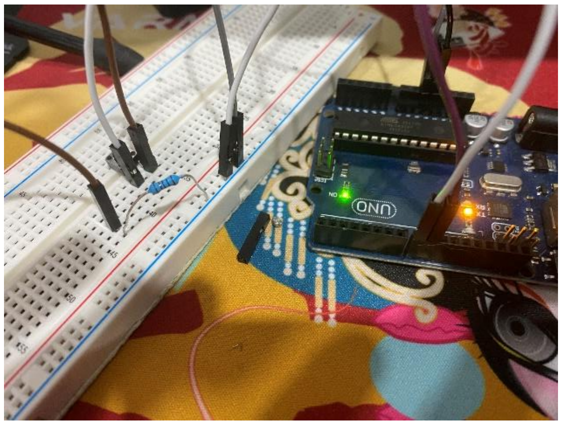


```
0
Jorawar
0
Jorawar
0
Jorawar
0
Jorawar
1
Jorawar
HI
0
Jorawar
0
Jorawar
0
Jorawar
1
Jorawar
HI
0
Jorawar
0
Jorawar
0
Jorawar
HI
0
Jorawar
0
Jorawar
0
Jorawar
0
Jorawar
```

This is some returns I got in terminal when there is button bouncing happening in the system. This code returned a 1 when input was HIGH and 0 when input was LOW. The button was not triggered at any point, the changed were caused by electrical interference, most likely from the servo.

There's also the chance that pyFirmata may not handle inputs from Arduino that well.

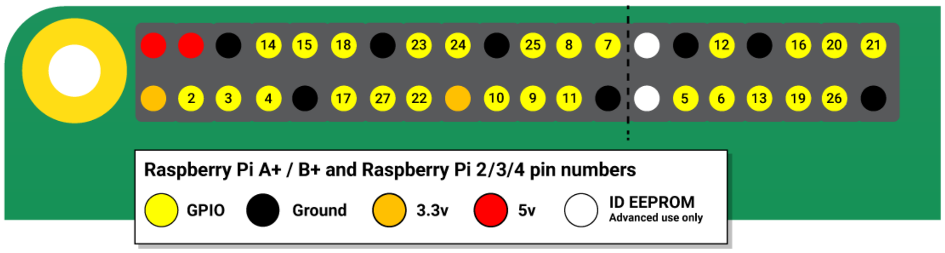
I tried solving this through adding a 220 and 10k resistors to the circuit but they merely reduce the number of “bounces” in the circuit and did not entire solve the issue.



This cause major reliability issues so the input was integrated with the Rpi which was able to do it far more reliably.

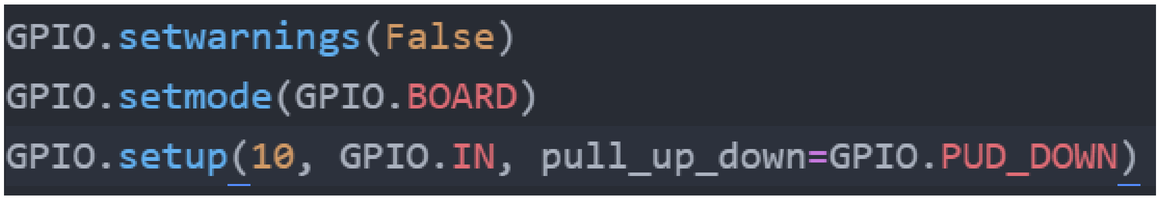
Button Inputs on Rpi

The input and output ports of the raspberry Pi are known as the GPIO ports and each pin has a different function. Unlike the Arduinos, the Rpi doesn't have its pins labeled and you have to lookup a diagram online to find out which ports you can use.



I connected the button on GPIO port 10 and coded it accordingly;

Initialize:

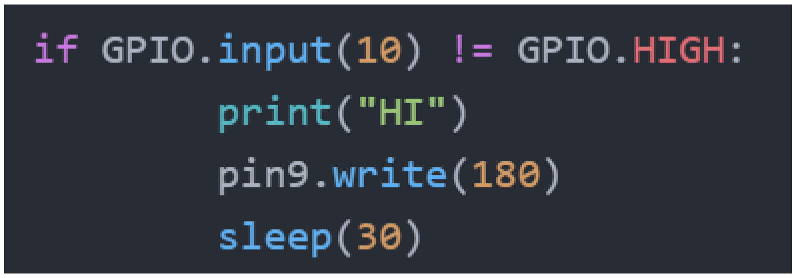


```
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
GPIO.setup(10, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
```

This registers pin 10 as in INPUT on the Rpi.

I checked if the button was pressed for two conditions.

At the beginning of the loop, before any faces were recognized.



```
 if GPIO.input(10) != GPIO.HIGH:
         print("HI")
         pin9.write(180)
         sleep(30)
```

When the button is pressed the circuit is broken and the Rpi registers it as LOW/zero. So, if we check for when the circuit is low on the GPIO port 10. I should have used GPIO.input(10) == GPIO.LOW:

but I was testing the if statements when I was having issues with button bouncing and forgot to change them over.

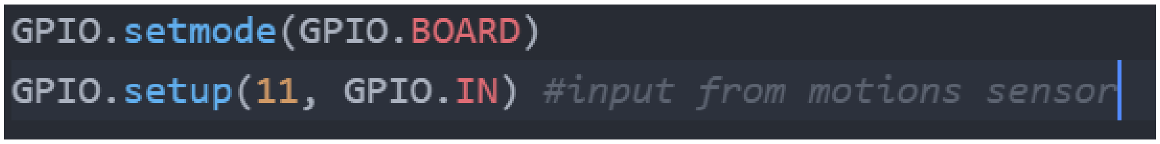
Once the Rpi detects that the button has been pressed we will close the lock and send the program to sleep for 30 seconds. This means that for 30 seconds no one will be able to unlock the safe, this time can be increased to ensure extra safety but for demonstration purposes the time has been kept to 30 seconds.

We also check if the button has been pressed after a face has been detected and Its done exactly as above.

MOTION DETECTION WITH RASPBERRY PI

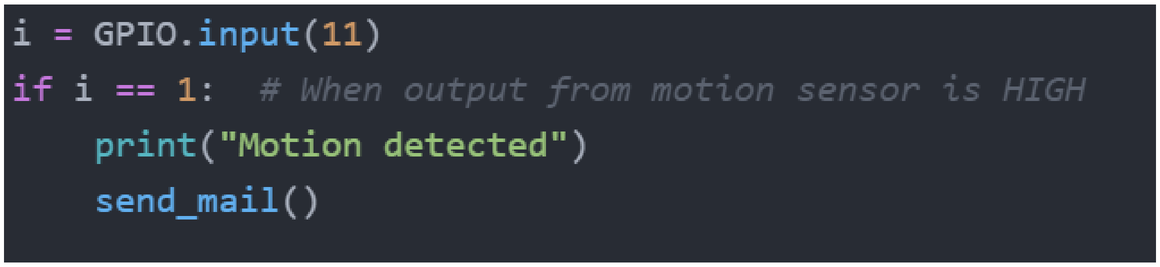
Motion detection will be used to determine if there is any movement around the Rpi. I was originally going to use an ultrasonic sensor to detect motion, but I ripped out the motion detector module from another Arduino project and user that instead.

This sensor detects with there is any movement around it and initializing it on Rpi GPIO is very similar to initializing any other Input.



```
GPIO.setmode(GPIO.BOARD)
GPIO.setup(11, GPIO.IN) #input from motions sensor
```

The motion detection module returns (1) everytime it detects movement around it, using this property we create a if statement checking for motion each frame,



```
i = GPIO.input(11)
if i == 1:  # When output from motion sensor is HIGH
    print("Motion detected")
    send_mail()
```

If motion is detected, we send an email using IOT

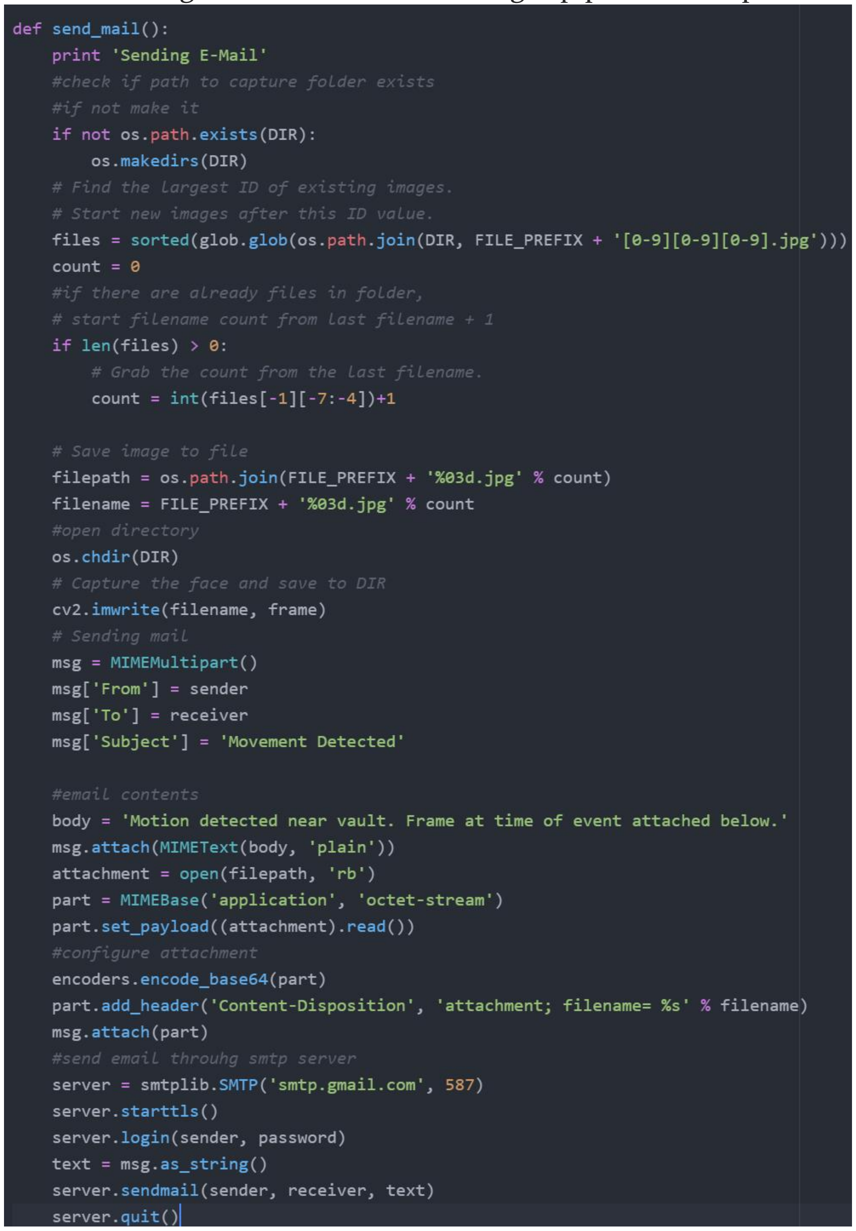
Demo video

IOT: SENDING AN EMAIL

To use IOT and implement it in my system I used the SMTP (simple mail transfer protocol). This provides functions to send and receive emails through a TCP/IP network.

In order to use this, I needed a Gmail account with security turned off and use from unknown apps enabled. This would leave a email heavily compromised so I created a new email which was not linked to any of my accounts at all. I also must provide my email and password for the account I wish the email to be sent from so anyone who has access to this document, or my GitHub would be able to easily access my email account.

After installing the SMTP module through “pip install smtp”



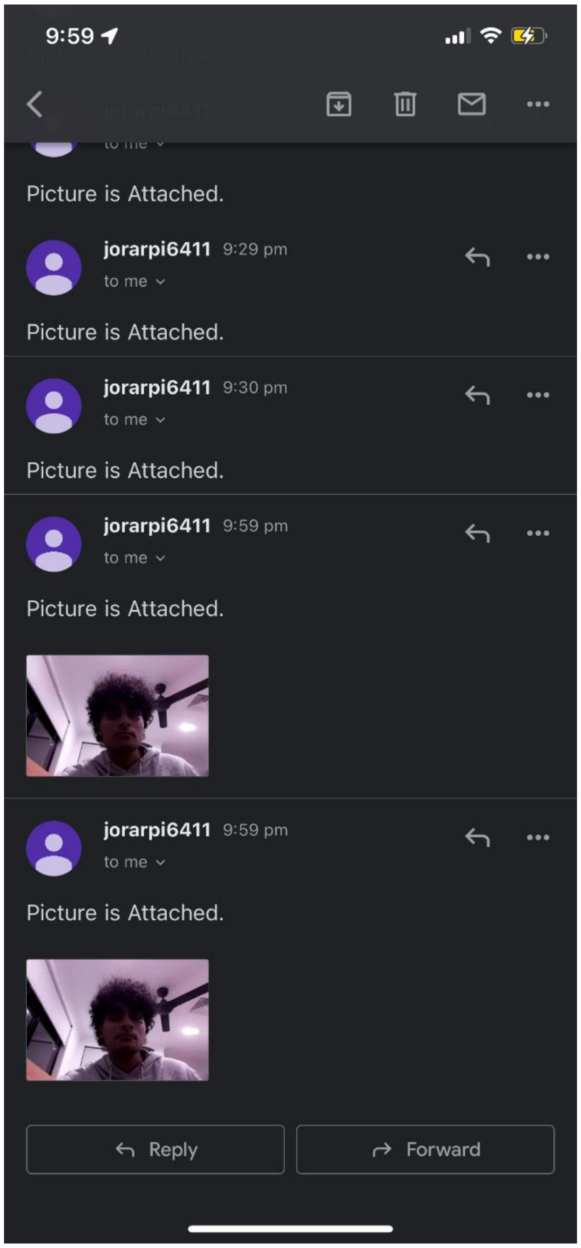
```python
def send_mail():
    print 'Sending E-Mail'
    #check if path to capture folder exists
    #if not make it
    if not os.path.exists(DIR):
        os.makedirs(DIR)
    # Find the largest ID of existing images.
    # Start new images after this ID value.
    files = sorted(glob.glob(os.path.join(DIR, FILE_PREFIX + '[0-9][0-9][0-9].jpg')))
    count = 0
    #if there are already files in folder,
    # start filename count from last filename + 1
    if len(files) > 0:
        # Grab the count from the last filename.
        count = int(files[-1][-7:-4])+1

    # Save image to file
    filepath = os.path.join(FILE_PREFIX + '%03d.jpg' % count)
    filename = FILE_PREFIX + '%03d.jpg' % count
    #open directory
    os.chdir(DIR)
    # Capture the face and save to DIR
    cv2.imwrite(filename, frame)
    # Sending mail
    msg = MIMEMultipart()
    msg['From'] = sender
    msg['To'] = receiver
    msg['Subject'] = 'Movement Detected'

    #email contents
    body = 'Motion detected near vault. Frame at time of event attached below.'
    msg.attach(MIMEText(body, 'plain'))
    attachment = open(filepath, 'rb')
    part = MIMEBase('application', 'octet-stream')
    part.set_payload((attachment).read())
    #configure attachment
    encoders.encode_base64(part)
    part.add_header('Content-Disposition', 'attachment; filename= %s' % filename)
    msg.attach(part)
    #send email throuhg smtp server
    server = smtplib.SMTP('smtp.gmail.com', 587)
    server.starttls()
    server.login(sender, password)
    text = msg.as_string()
    server.sendmail(sender, receiver, text)
    server.quit()
```

I created a function to send a message, to keep my main loop as neat as possible.

The function first checks if there already exists a directory where we are saving our captured images when motion is detected. If there isn't one it creates it. If there is it gets the count of images and adds one for the image currently being captured.

We then want to save the image to the capture directory in order to keep everything logged and in the right place.

example of it working here



Picture is Attached.

jorarpi6411 9:29 pm
to me

Picture is Attached.

jorarpi6411 9:30 pm
to me

Picture is Attached.

jorarpi6411 9:59 pm
to me

Picture is Attached.

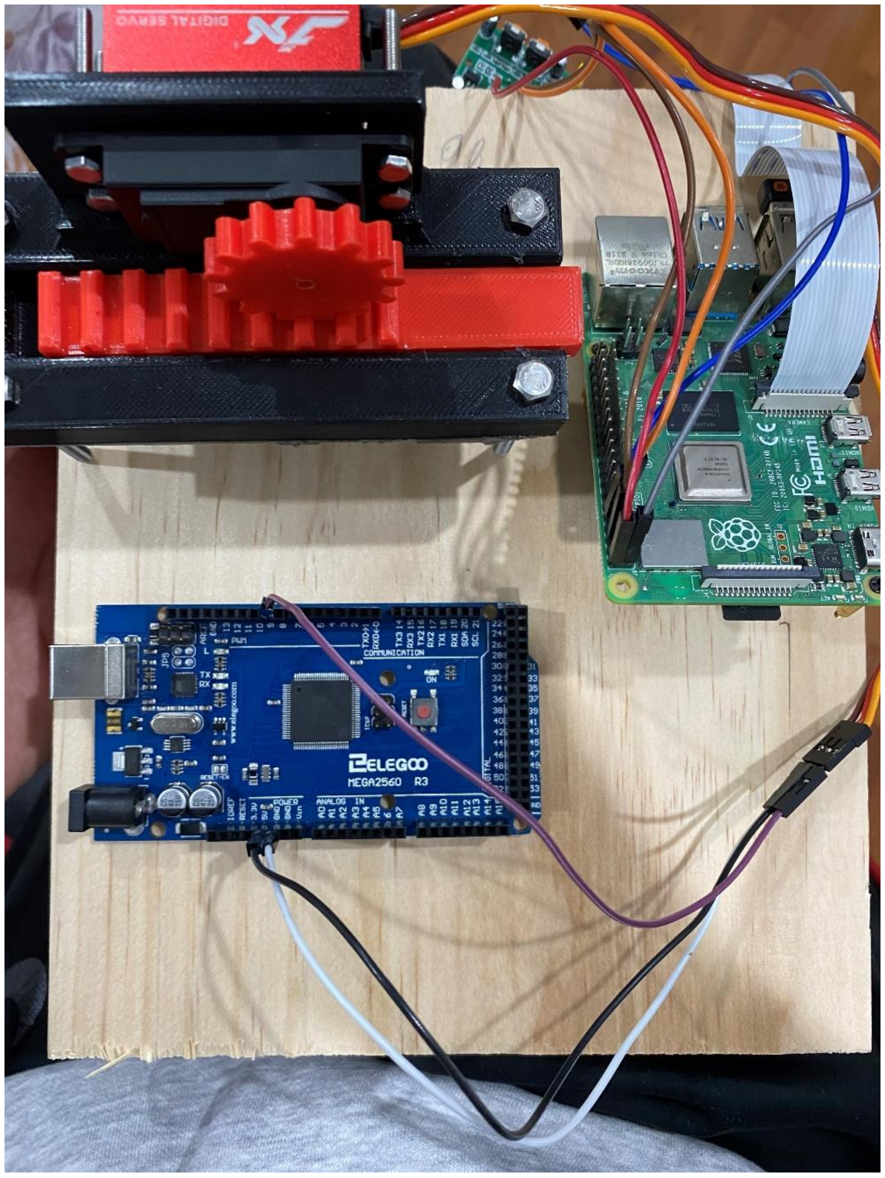
jorarpi6411 9:59 pm
to me

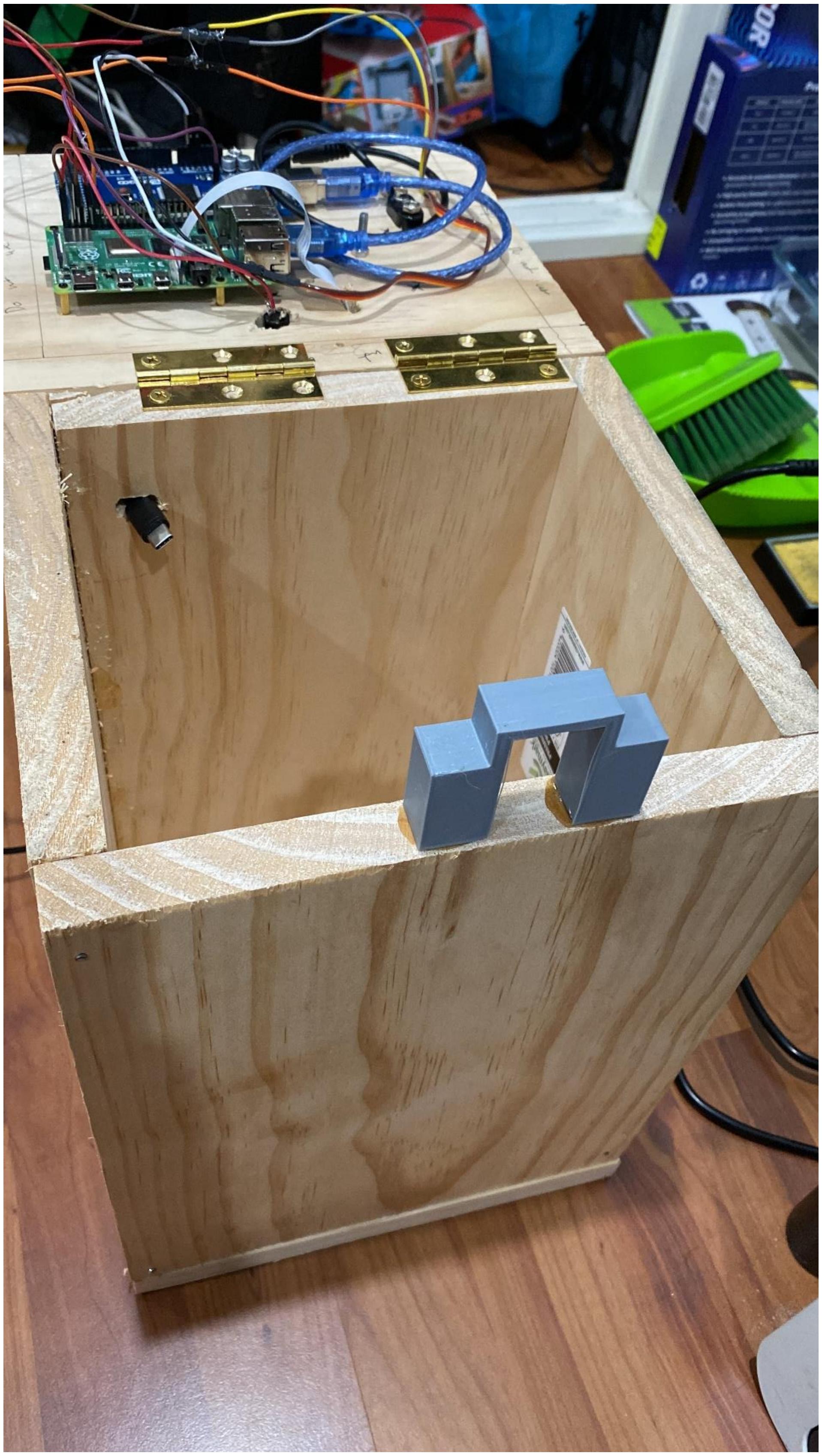
Picture is Attached.

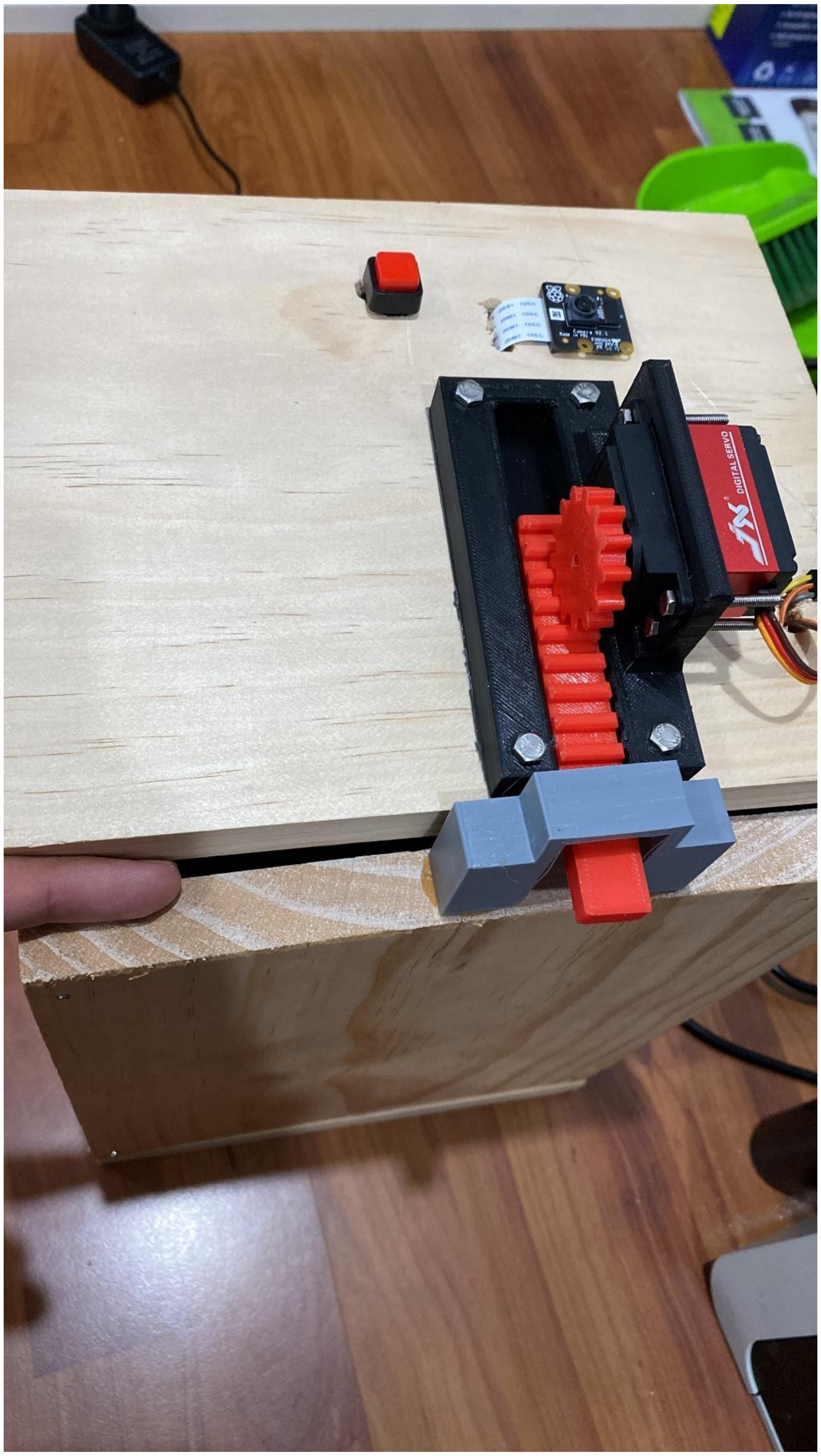
Final Assembly

I'm missing a lot of mounting hardware; I still have standoffs for the pi camera being shipped to me. Hot glue is amazing and fixed pretty much all my problems.

I didn't have a decent way to create holes in the wood, so I just put my drill to work and made a bunch of small holes that joined to make a big one.






DIGITAL SERVO