**After-Action Report**

***Robogator***

Augusto Mota Pinheiro, 11/2020 – 12/2020

T.O.C.

[Preface 2](#_Toc64901785)

[Intro 2](#_Toc64901786)

[Accomplishments 3](#_Toc64901787)

[Plans 3](#_Toc64901788)

[Testing 5](#_Toc64901789)

[Design 6](#_Toc64901790)

[Printing 9](#_Toc64901791)

[Assembly 12](#_Toc64901792)

[Software 14](#_Toc64901793)

[Failures & Teachings 17](#_Toc64901794)

[Plans 17](#_Toc64901795)

[Testing 17](#_Toc64901796)

[Design 17](#_Toc64901797)

[Printing 18](#_Toc64901798)

[Assembly 18](#_Toc64901799)

[Software 18](#_Toc64901800)

[Conclusion 19](#_Toc64901801)

# Preface

Welcome to the After-Action Report of *Robogator*. This report will go into detail about what my plans were for the project, what I managed to accomplish and what I didn’t and what I learned from the whole experience. I’ll finish off with a short conclusion to wrap up everything.

# Intro

Everything started when I, with the help of John, came up with this project to better utilize my capabilities and my interests in the *Interactive Media Workshop* class. It was around the end of October, 2020 and I had around a month until the end of the semester to come up with the concept, plan it and create it. John also proposed that I extend the project into the next semester as the theme of *Art of Physical Computing* would fit well.

With that in mind, I got to work and after rejecting a few initial ideas, I landed on making a robot whose gait would be similar to an alligator’s.

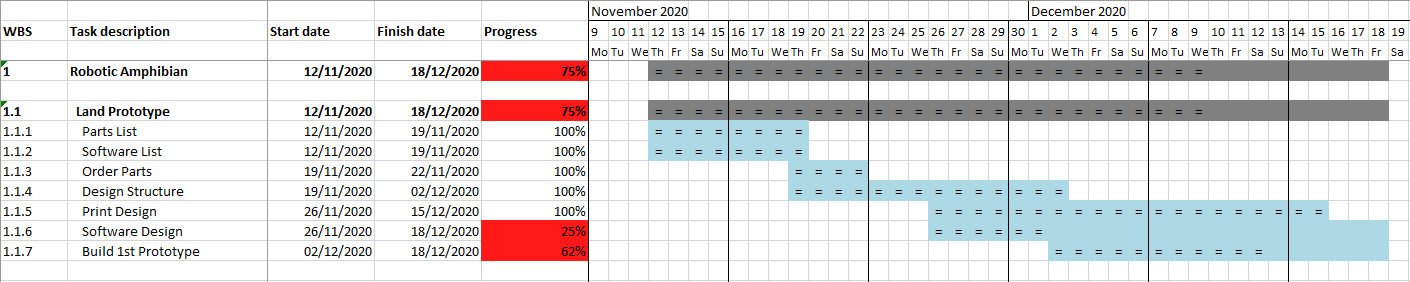
# Accomplishments

## Plans

I had a solid idea on how to proceed with everything in my head, however I realize now that I should’ve written it down, not only to have a record of it, but also so that John could look at it.

That being said, my initial plans were to make a fully articulated robot, where each leg would be divided into 2 independent segments (each controlled by a 9g servo), the shoulder would also have a degree of freedom (being controlled by a standard servo) and the body could twist a little in the middle (making it easier to turn) by having a standard servo connecting the two body segments.

I created a Gantt chart to keep track of my progress, as well as to have a way of keeping all of the necessary tasks in order:



Having that in place, I created a parts list with their cost to have an idea of the finances of it all:



I chose to put standard servos (the bigger ones) on the shoulder for more power as they were rotating the entire leg at any one time, I chose the AA batteries as my power source to have a quick and easy power supply that I could have on hand easily and the Arduino seemed like the proper choice to power everything, since I was going to code everything by hand (so no Artificial Intelligence involved).

Next, I did some research of what software would work well with my chosen components (principally with the Arduino) and came up with a short list:



Also, throughout the project I kept adding sources and adding personal notes too!

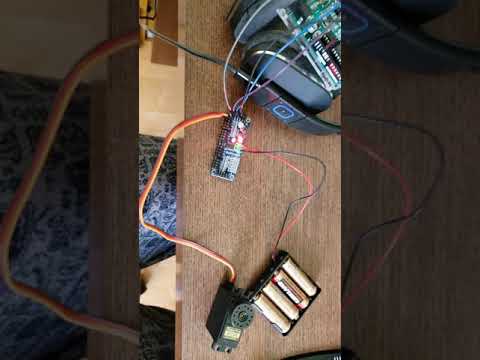


As we can see, there were a few failures and problems that occurred here and there, which will be discussed at the end of this report.

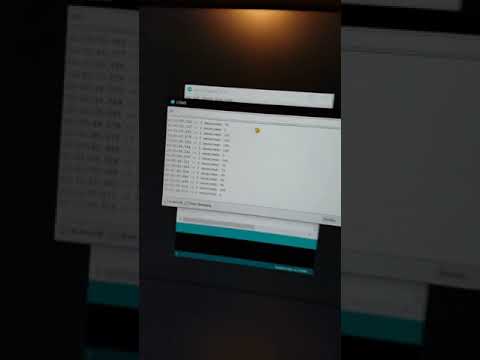
## Testing

When the parts I ordered arrived, I proceeded with immediately testing the servos, the servo PWM driver and the Arduino. I got these later in the process, so in the second video, a leg will already be printed and assembled:

1. PCA9685 PWM Servo Driver Test

[](https://www.youtube.com/embed/9Fn8Kn1znDc?feature=oembed)

2. Shoulder Servo Test

[](https://www.youtube.com/embed/bq7EOkSEyMk?feature=oembed)

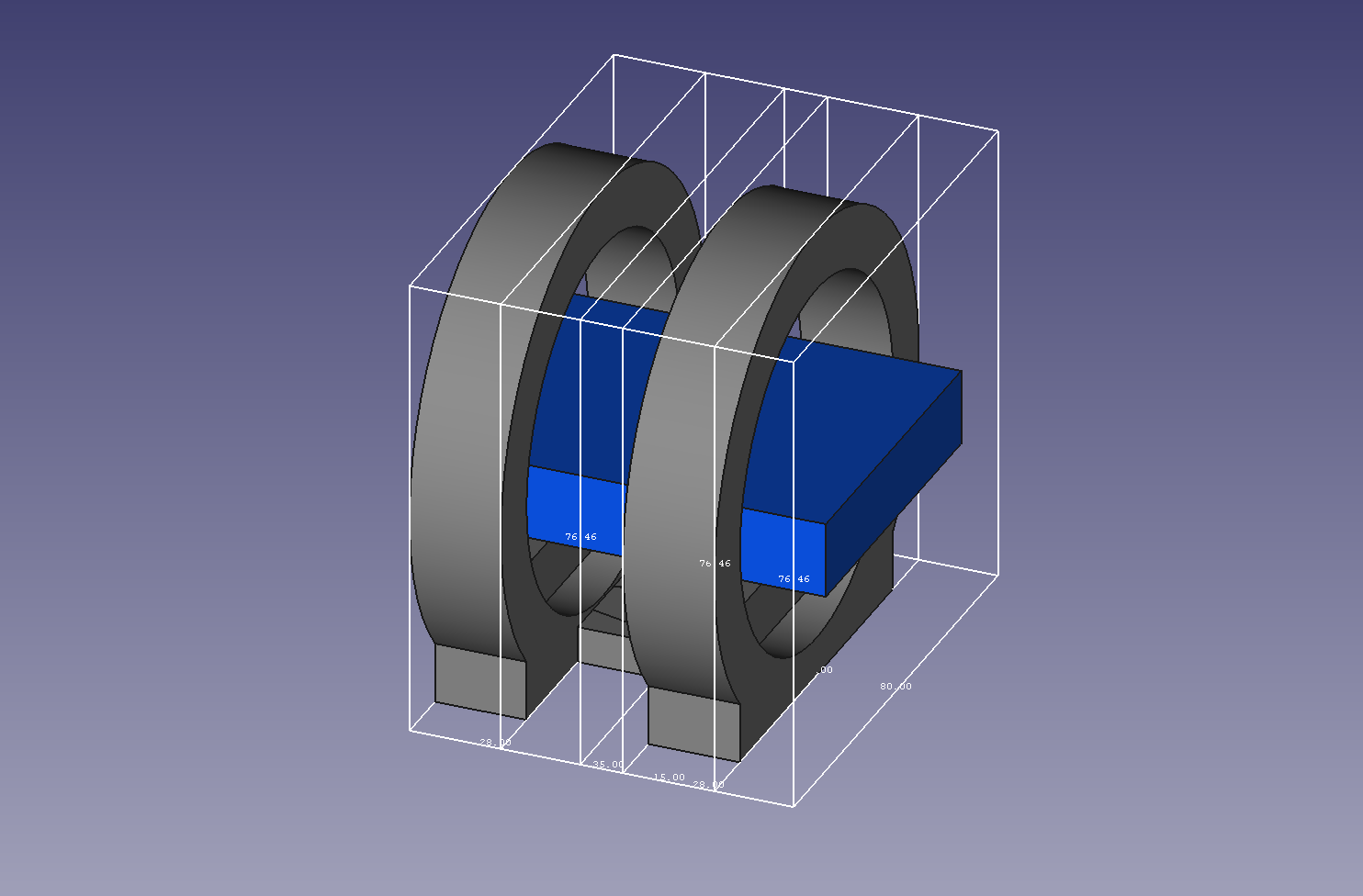
## Design

A note before I continue: I’m a software engineer at heart, so I’m very used (some could even say dependent) to the *undo* function in whatever I am working on; physical objects (like 3D printed ones) don’t have that function. It seems obvious now, but I truly only realized that pretty late in the project.

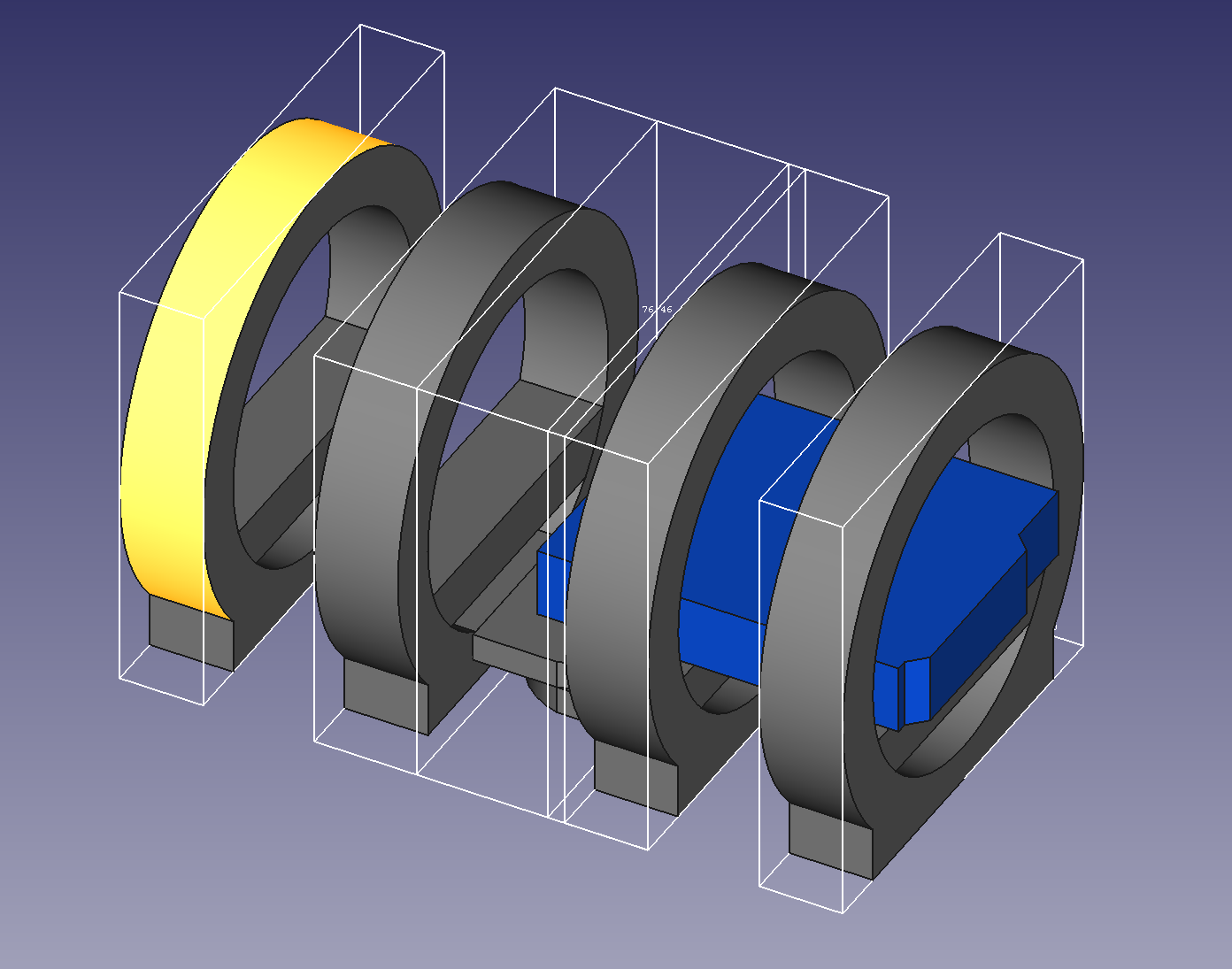
Now, I had no experience in any CAD or technical modelling software before this project, so a big part of my time was dedicated to learn everything I deemed necessary about FreeCAD. I chose FreeCAD for one big reason: it’s free! Looking back now, I couldn’t really do any better for free, but I had completely overlooked the fact that I could request a student license for one of the Autodesk software (like Inventor) which would’ve been easier to use.

Regardless, I got to work and came up with what I thought was a pretty nice design. The following pictures will be a quick timeline of the design milestones I reached; note that the blue part is the Arduino, the red are the body segments, the yellow are leg segments and green are servo motors:

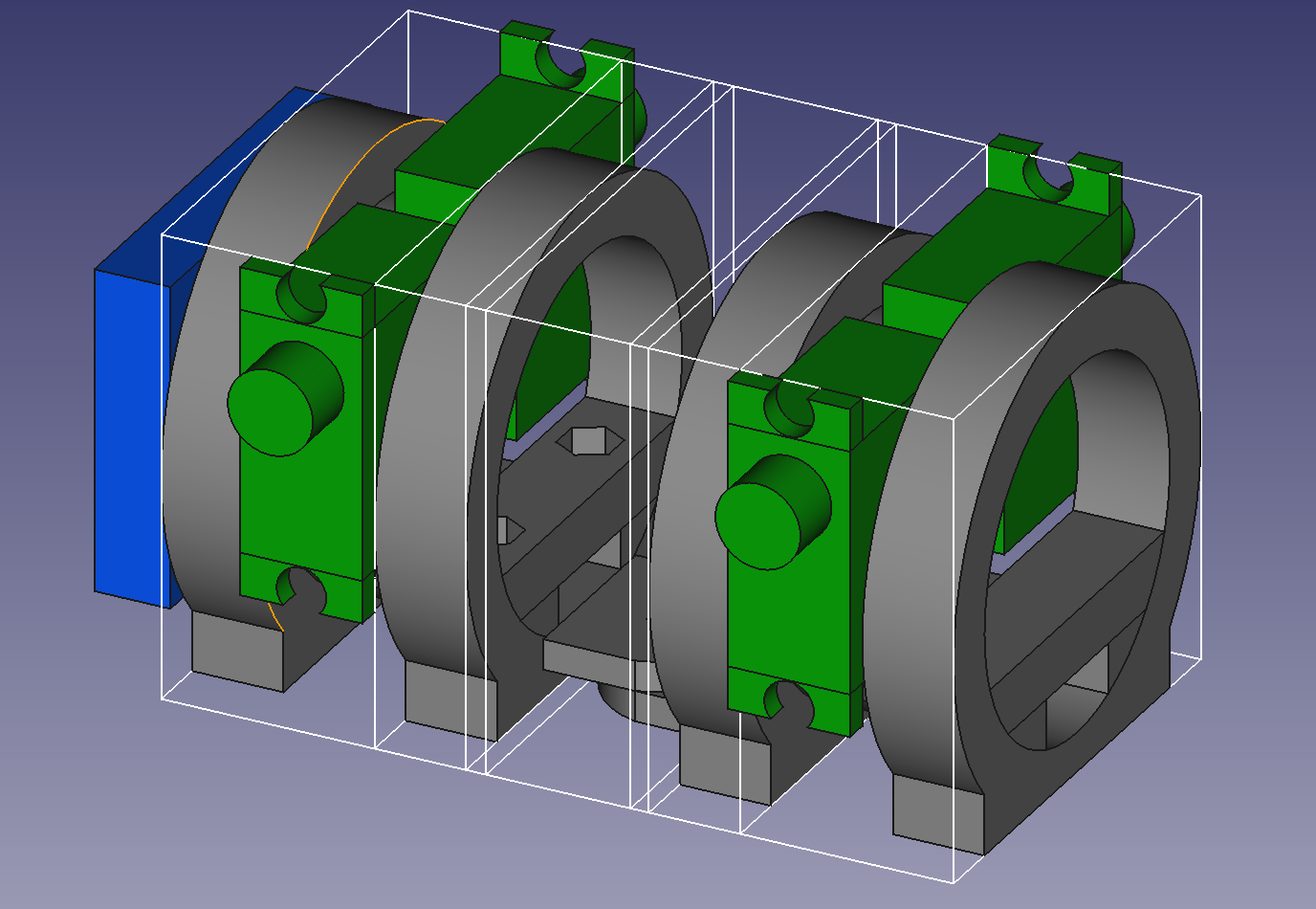
3. Initial Arduino Mount Design



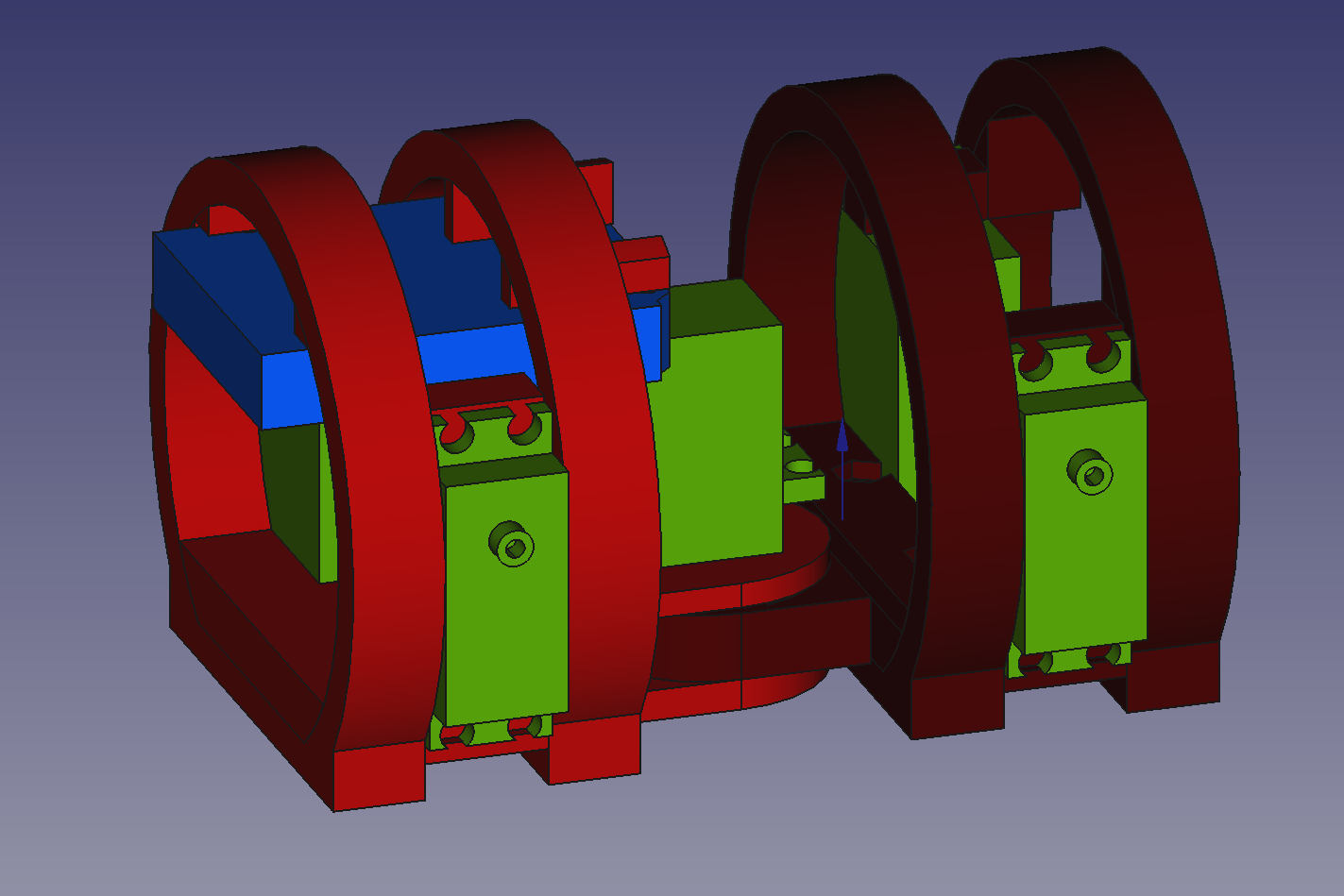
4. Initial Skeleton Design



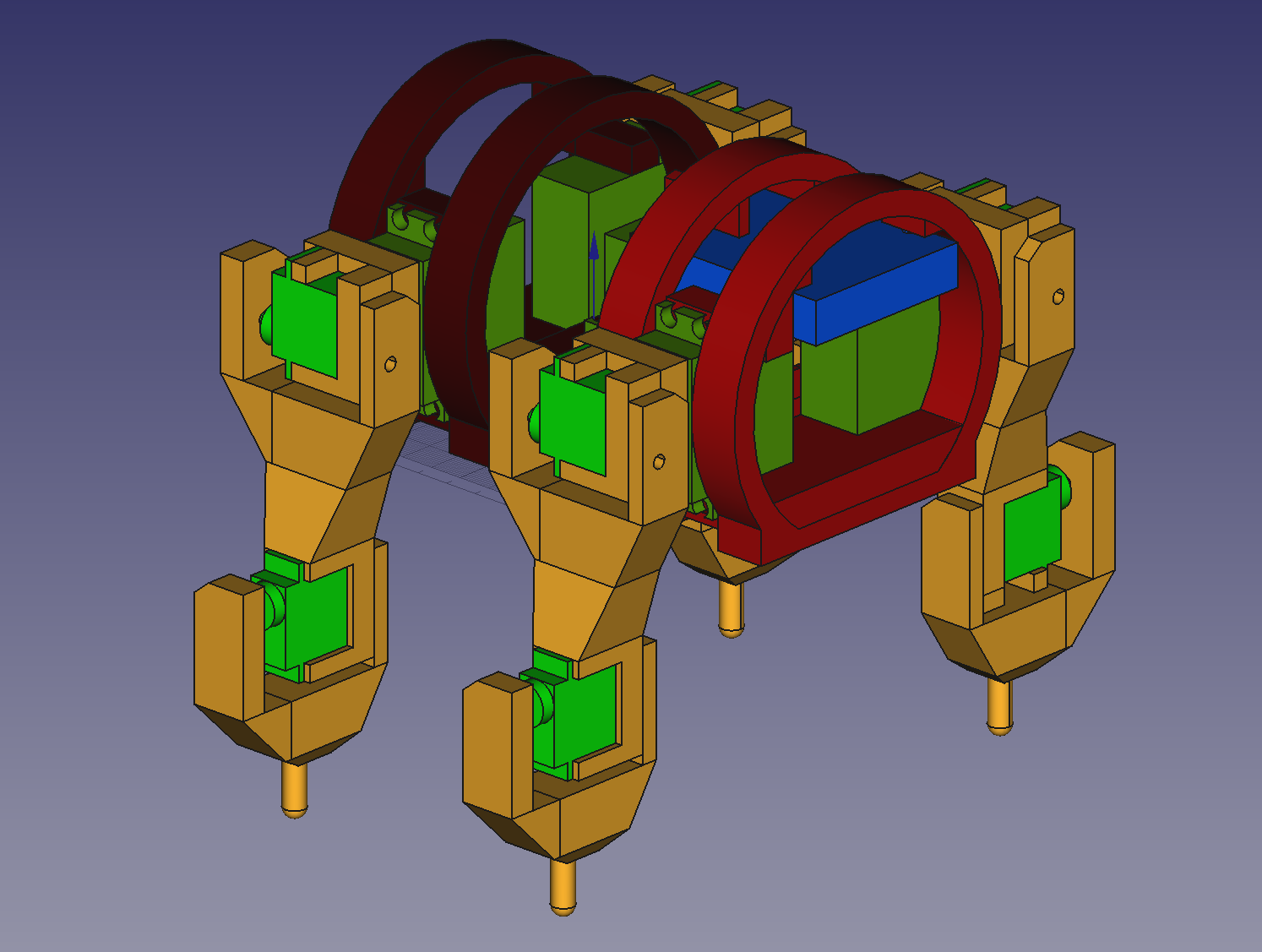
5. Skeleton with Motors

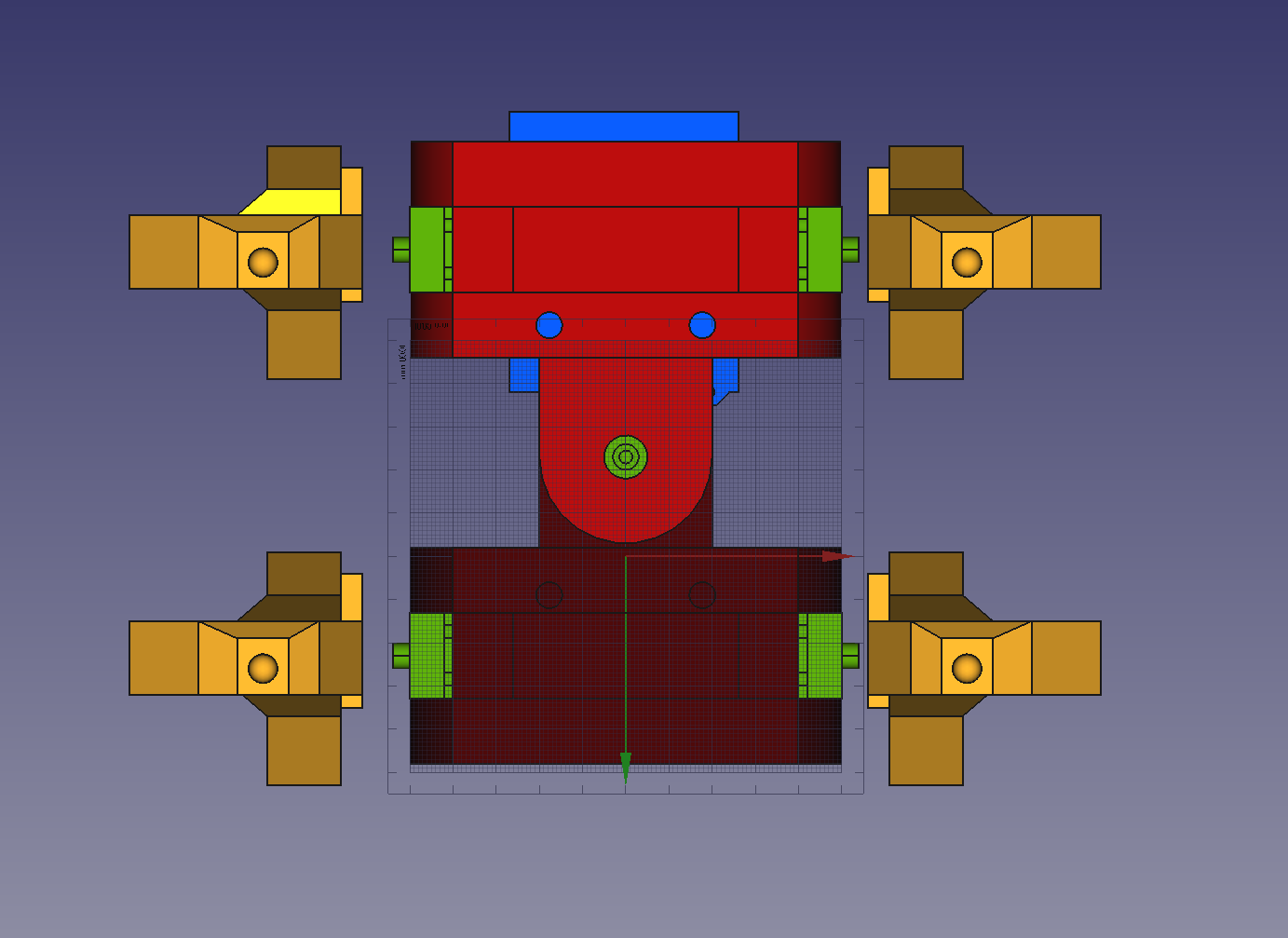


6. Final Skeleton Design with Motors

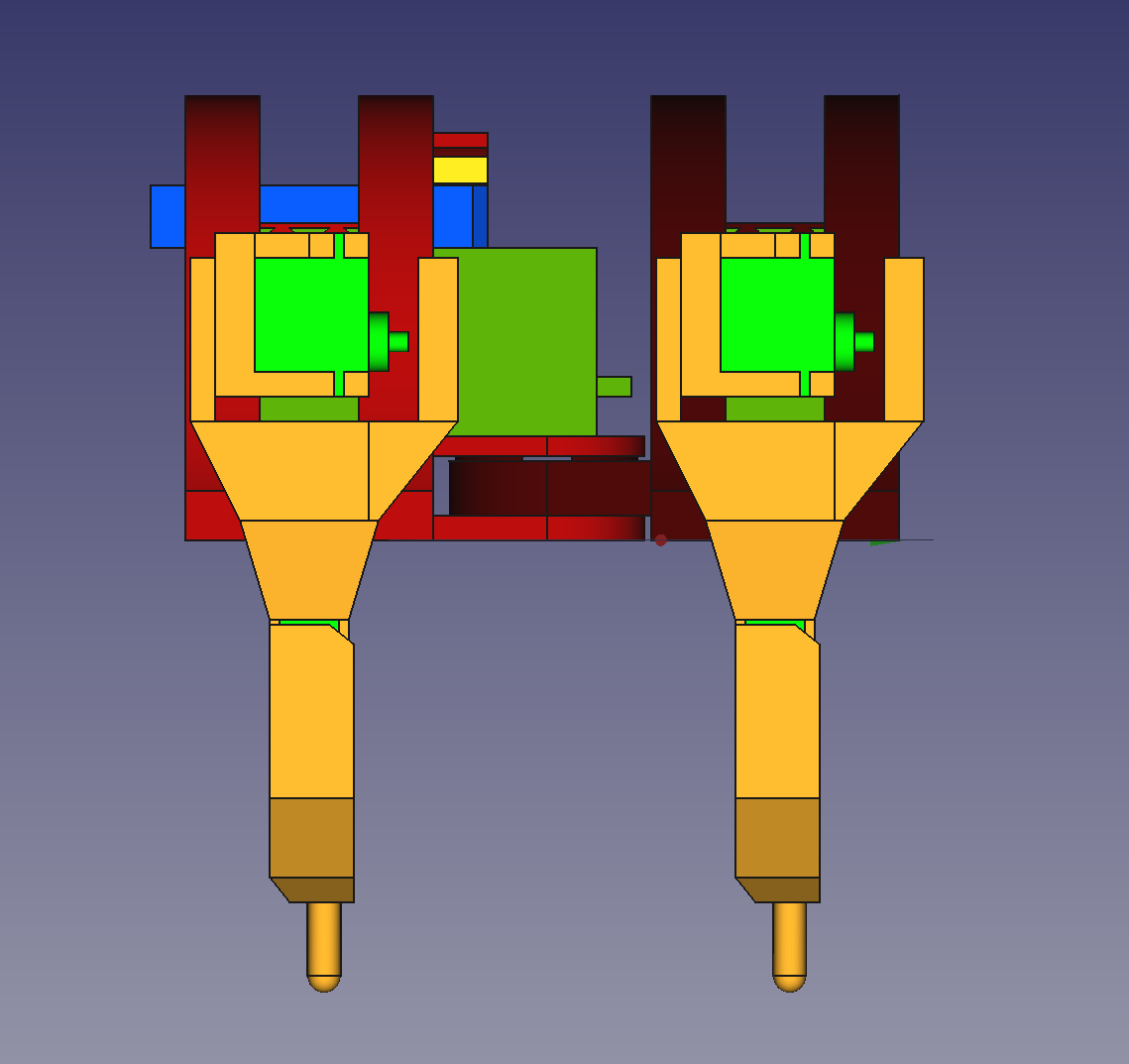


7. Final Land Prototype Design

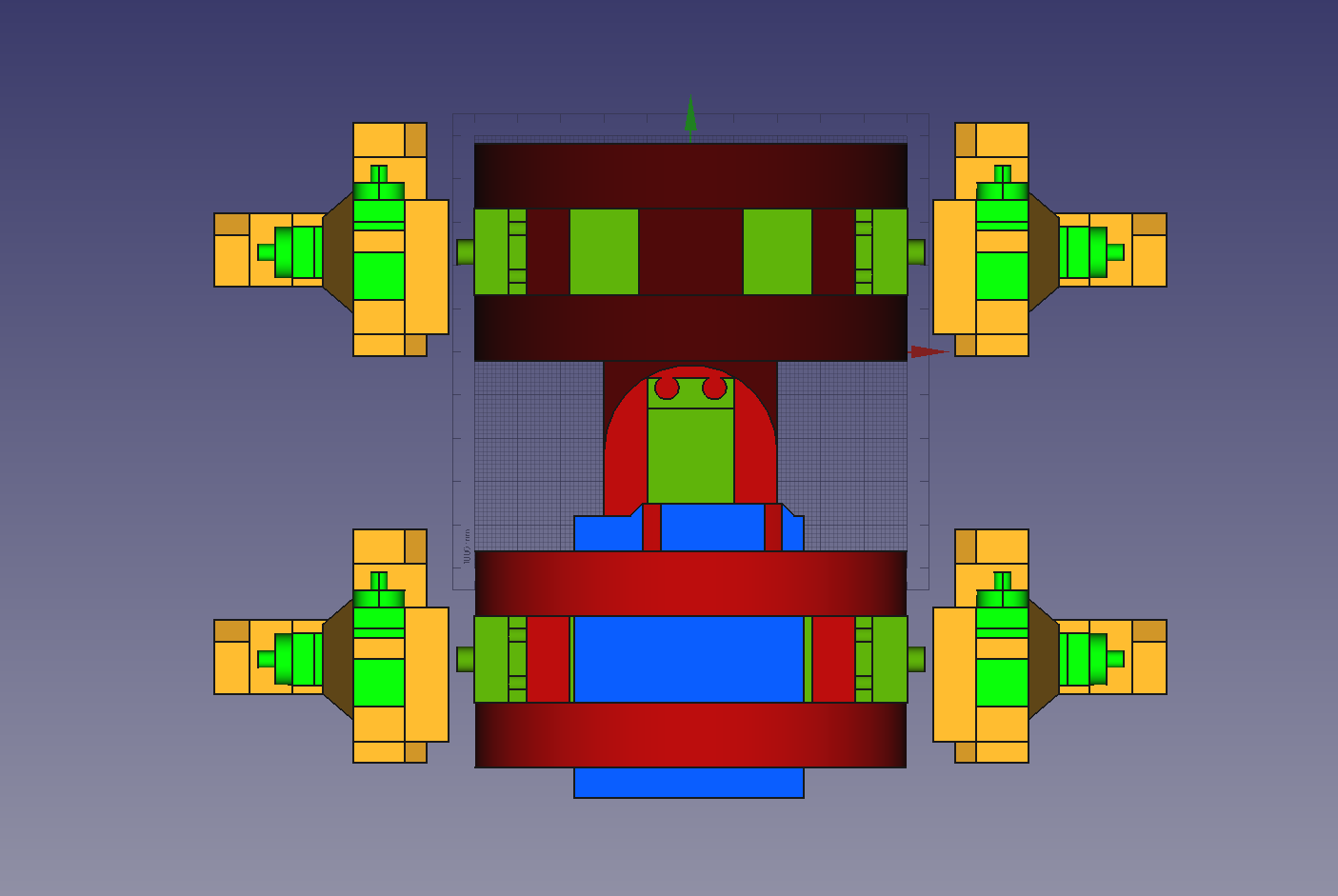




**Bottom**



**Right**



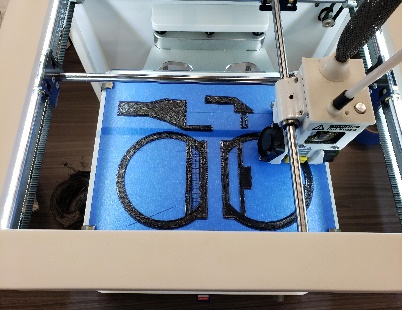
**Top**

## Printing

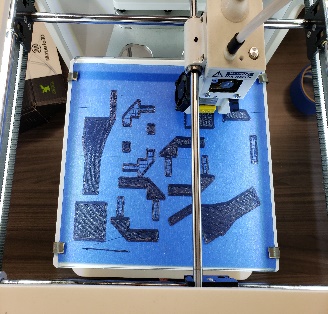
After designing all the 3D models, it was time to print everything part and assemble it! Following my mom’s suggestion, I asked my old high school (Collège Beaubois) if I could use their 3D printers for my project, as my printer’s bed was pretty small and wouldn’t be able to fit the bigger pieces. I loaded all of my *.stl* files on Onedrive and got to Beaubois, where I used *Tinkerine Suite* as the slicer for the *Tinkerine* printers they have. On my first print job, I sent my 2 biggest pieces (part of the skeleton) and one of them failed:



On the second job, I had 5 pieces, where 3 failed (not show below):



Since it was the first time where I printed in such big quantities, I imagined that this was normal, so I did the same thing for the next one, where I had 8 pieces:

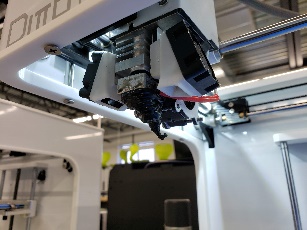
 

However, the print job after that (11 pieces) was complete chaos and that’s when the printing process started to go downhill:

8. Failed Prints



9. Jumbled Extruder

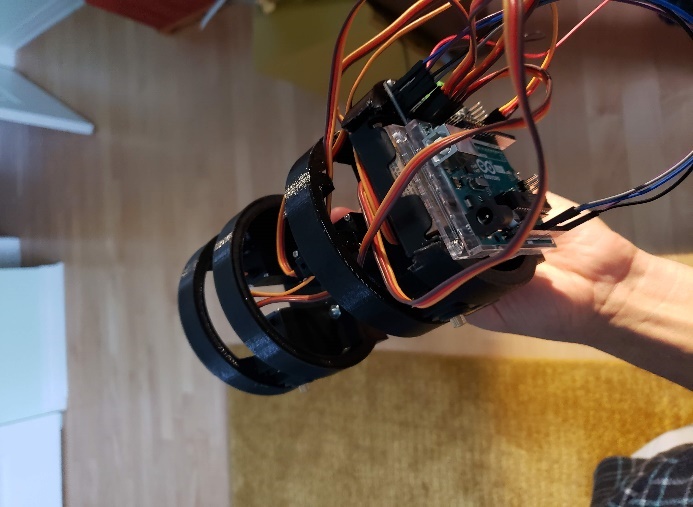
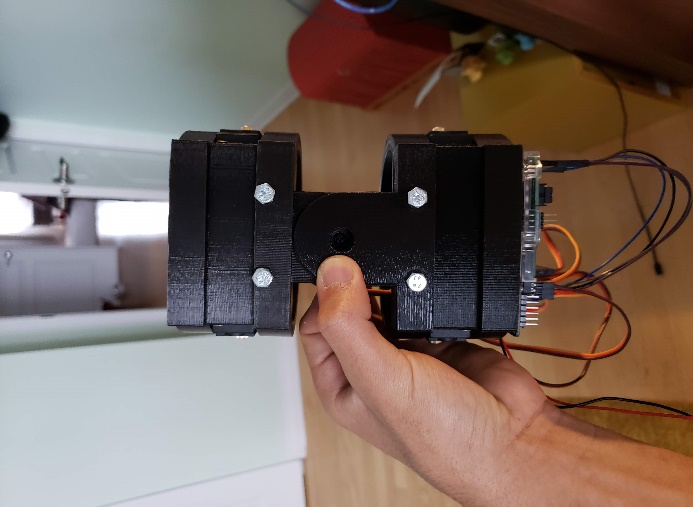


At least one print job at that point and beyond fully failed consistently and instead of finishing everything in 4 four days, it took 2 weeks which kind of threw my planned schedule away. In total, I used 570g of plastic, while only 412g actually ended up in useful pieces.

## Assembly

For the assembly of the robot, I had planned two main ways of doing so: using M5 bolts for the body and general screws, those that came with the servos and some short wood screws that I bought at Home Depot. There’ll be a dedicated problems section later on, but a problem that I identified immediately when starting this step was that the Arduino would not fit in the supports I had printed for it, so I improvised and put it at the “head” of the robot along with the battery and the servo driver. Other than that, the rest went pretty well and the initial assembly with all of the body segments looked like this:

10. Skeleton Assembled with Control Boards

The next step was to screw on the shoulder joints. Those were a pain to assemble, because I hadn’t made any holes in my pieces, thinking that the plastic would be easy to screw into:

11. Shoulder Joints Screwed On



The hardest part was the next one, as I had to figure out a way of assembling the leg segments with one another (with no screw holes or anything). Since I didn’t want to re-print everything with tapped holes, I used brute force to get everything somewhat working together. After many frustrating nights, I got everything in one piece and wired up:



To my big disappointment, the contraption had no balance because there was no rigidity in the legs, so the legs would open and the entire structure would fall down.

## Software

We’re finally to the part that I am comfortable with: software! Unfortunately, due to my incapability to get a balanced robot, I couldn’t get to the part where I was ready to really program it. I did however have this piece of test code:

Adafruit\_PWMServoDriver pwm = Adafruit\_PWMServoDriver();

byte incomingByte;

void setup() {

*// put your setup code here, to run once:*

Serial.begin(9600);

pwm.begin();

pwm.setOscillatorFrequency(27000000);

pwm.setPWMFreq(50);

setServoToDeg(0, 45);

setServoToDeg(1, 45);

setServoToDeg(2, 45);

setServoToDeg(3, 45);

setServoToDeg(4, 0);

*//setServoToDeg(1, 0);*

*//setServoToDeg(2, 0);*

*//uint8\_t pulseLength = map(0, 0, 180, 150, 550);*

*//Serial.println(pulseLength);*

*//pwm.setPWM(1, 0, map(90, 0, 180, 150, 550));*

*//byte pinNb = 1;*

*/\*for (uint8\_t i = 0; i < 180; i++)*

*{*

*pwm.setPWM(pinNb, 0, map(i, 0, 180, 150, 550));*

*Serial.println(map(i, 0, 180, 150, 550));*

*delay(100);*

*}\*/*

}

void loop() {

*// put your main code here, to run repeatedly:*

if (Serial.available() > 0) {

*// read the incoming byte:*

incomingByte = Serial.readStringUntil('\n').toInt();

*// print what you received:*

setServoToDeg(incomingByte, 90);

Serial.print("I received: ");

Serial.println(incomingByte);

}

}

void setServoToDeg(byte *numb*, byte *degrees*)

{

pwm.setPWM(numb, 0, map(degrees, 0, 180, 150, 600));

*//Serial.println(map(degrees, 0, 180, 150, 600));*

}

# Failures & Teachings

For this section, I’ll proceed with a more bullet point summary of everything, as I think it will be clearer and quicker to get my point across.

## Plans

* A month was way too little to get a project of this size up and running, despite that I am surprised at how far I got considering I started from basically nothing.
* If any type of processing is required (as the plan was to have some sort of Inverse Kinematic algorithm) a more powerful processor and board is needed.
* Being such an ambitious project, I didn’t spend enough time on the planning and designing phase to get rid of all possible problems and quirks.

## Testing

* More testing should’ve been done and at a bigger frequency, to make sure my designs were sound.

## Design

* Lack of pre-tapped holes was a big problem when assembling later on.
* Straight legs (and other supporting pieces) are not a good idea, an angle is preferred so that in case it’s not completely balanced, the robot won’t fall over.
* Creating channels for cable management is a good idea!
* Having such dense pieces used only more plastic and didn’t really contribute to the structure of the bot (rather it fragilized it with the added weight).
* Having such small (and sometimes inexistant) tolerances was a big mistake and required various tools to correct them after the fact.
* Small leg endpoints didn’t help (and probably caused) the stability of the robot.
* Speaking of legs, each segment shouldn’t have been connected to the other through the servos only: better support would have improved stability and guaranteed balance.
* As previously said, more time should have been spent on the designing phase.

## Printing

* I did scale the objects to account for some printing warping and shrinking, but playing with the global scale of every piece was a bad idea as nothing lined up anymore.
* I also suspect the *Tinkerine Suite* software to have produced less than great G-code, explaining some of the failed prints.
* I learned this one and fixed it during the process, but even if the build plate allows it, you should limit the number of prints you do at a time to minimize failures.

## Assembly

* Screws are a good way of assembling 3D prints, but they need pre-tapped holes or need to use nuts to hold them in place.

## Software

* Inverse Kinematics (IK) are cool and I will probably make use of them again in some shape or form.

# Conclusion

To conclude, I learned a lot and gained an entire skillset from working on this project, skills that will probably pay off handsomely in my future. Knowing what I know, for my next project and specially if it’s for school, I’ll start from an existing foundation (like already made 3D models) and go from there.