Prototype 1 Documentation:

The idea for cart 360: The One and Only! Hold my Hand Bot!

- A machine's search for Meaning

When one thinks of robots they do not necessarily associate such thoughts with empathy. The idea here is to have a reactive mood companion that reacts to heart rate shifts. The point of this project is to connect man and machine further (the budget neuro link which uses heart rate monitoring).

- Mood companion, like a service dog, but digital

The visual design was inspired by a Gameboy color, and also the metro LCDs which display the next station.

As for the reason why I chose this combination, I was sitting in a metro on my way to class, playing on my old Gameboy color (legend of Zelda). I was getting tired, so as I yawned I checked which station was next. Although it said 'Rosemont, with the mixture of fatigue, my squinted eyes, and g-force of the metro made it look like eyes winking at me. So put two and two together and that is how the idea originated.

Sensors that will be used:

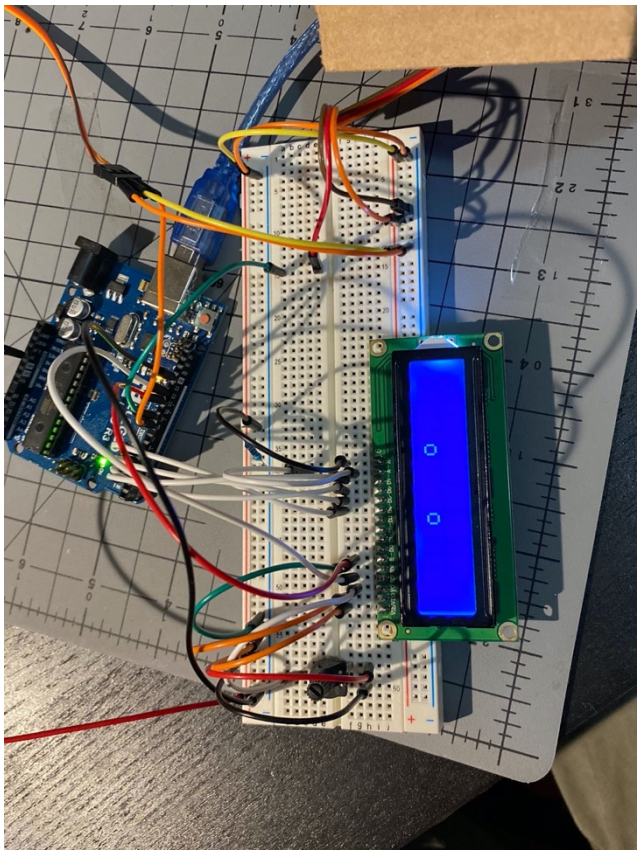
- Heart Rate Sensor: for this component, one has a couple of different options. One can opt to use a wired heart rate sensor that rests on the user or one can opt to get a wireless bracelet and transmitter. (see links below)
  - https://www.adafruit.com/product/1077
  - https://www.adafruit.com/product/1093
- Motion Sensor: This sensor will be located on top of the robot, this sensor will detect whether someone is close or not. Upon detection, the robot will greet the person by waving one of its arms.

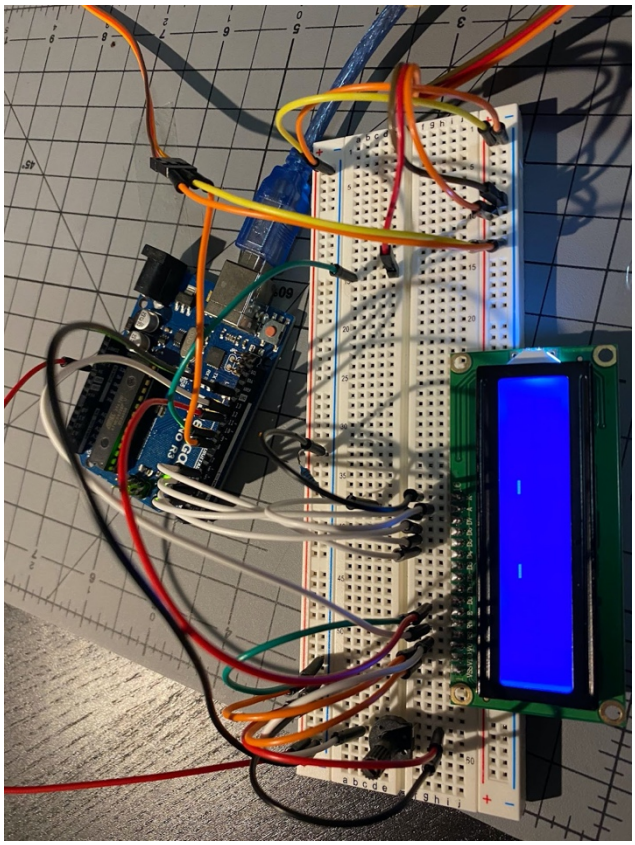
Research Questions:

Question 1: As humans, we are capable of developing a materialistic attachment to certain items, such as valuables, or memorabilia. Which can now prompt the question of: can a machine simulate a connection with a human (aka flip the subject on its head)?

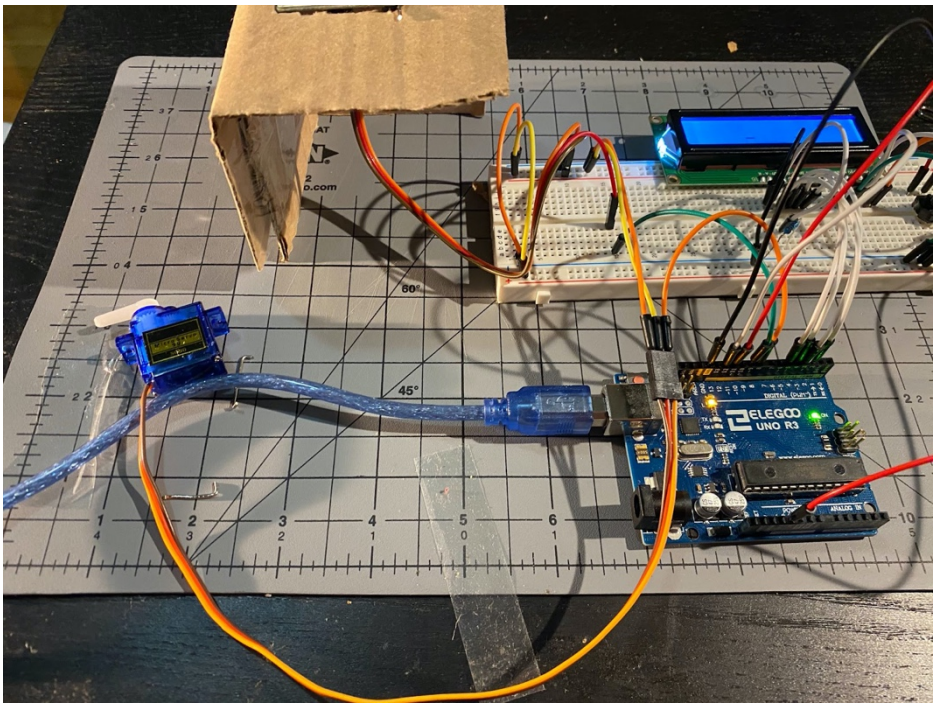
Question 2: Can someone feel comfort and/or some form of reassurance from a machine?

Question 3: Is this actually going to work?

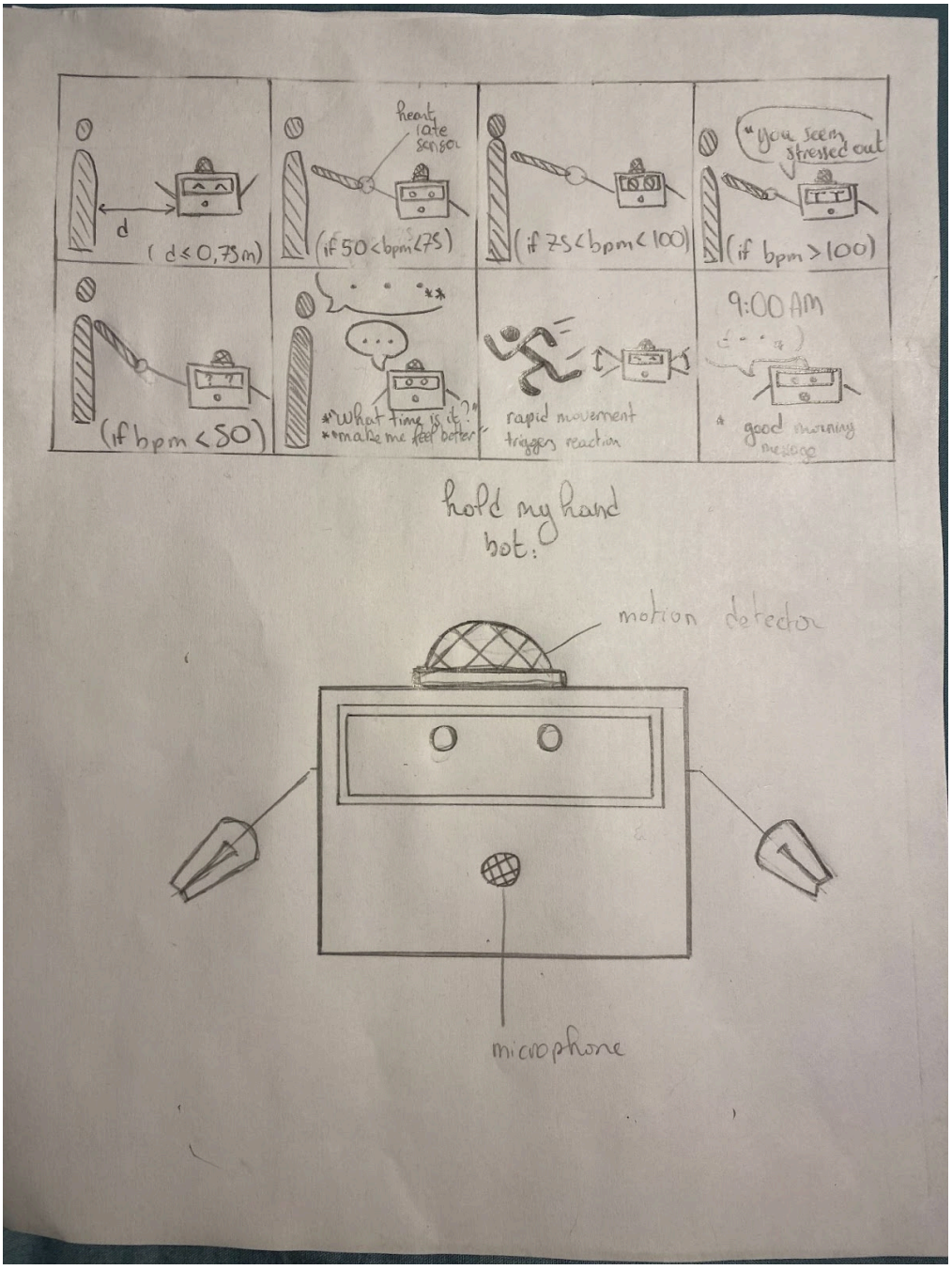




Breadboard setup 1 for: “Hold My Hand Bot”



Original Sketches For Hold My Hand Bot



Prototype 2 and 3 Process + Documentation of Workflow

Prologue:

Why is this a prologue one may ask? Well the answer to that is in the last line of this section.

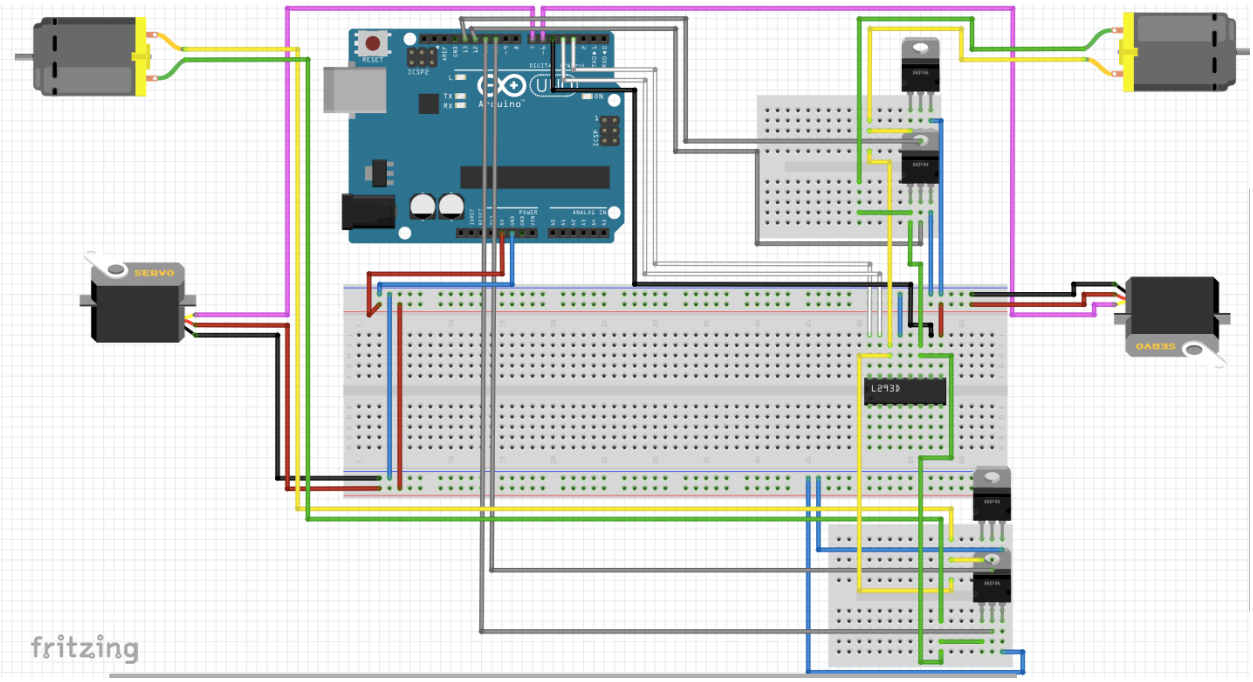
The idea for Pulse-Bot (or “Hold My Hand Bot” as I initially named it), originally involved a robotic arm with a pulse sensor attached to the hand. The robot would, upon detecting an element within its vicinity with the ultrasonic sensor, extend its arm prompting the user/individual to shake its hand. Upon contact with the said hand, the pulse of the user would be determined, and the robot would react according to the value read. However, after many complications trying to order pulse sensors from china, and them not arriving in time or costing 50+ CAD in shipping to arrive at a ‘reasonable time, I scrapped this idea.

However, I still was quite fond of the idea of having a robot that gets your pulse and wanted to consider other possibilities. So I started looking at the price of low-cost medical equipment in hopes to find ANYTHING, which would allow me to get one’s current pulse to the Arduino. All I wanted was to get a pulse value in the form of an integer on the micro-controller. This would be the one key/distinguishing element which would make PulseBot, well… a PULSE-bot. The journey would not be an easy one though. Let me structure this tale in the form of a play script, I feel like that would somewhat accurately depict how comically, unnecessarily tedious and difficult this process was.

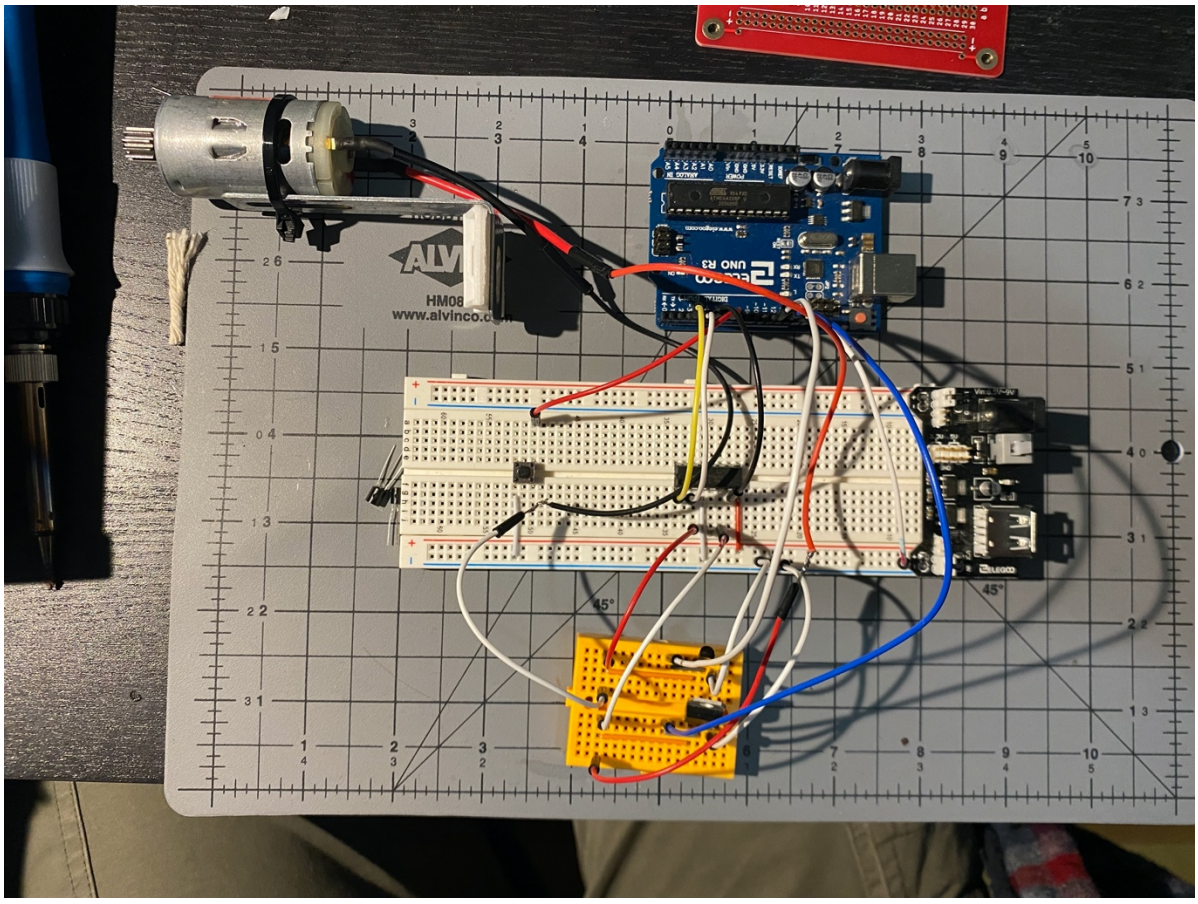
Act 1: The Robotic Arms Pickle

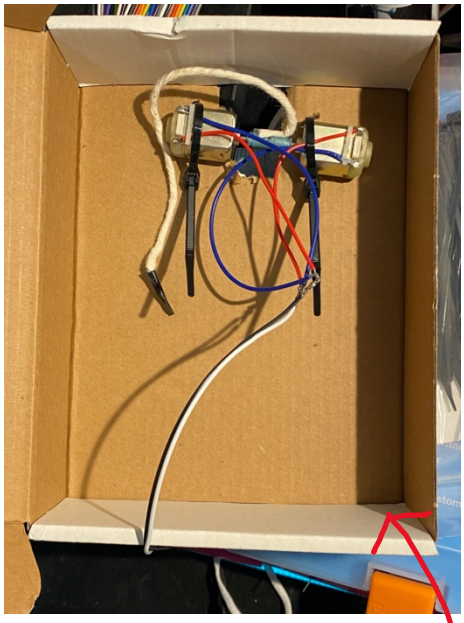
For a couple of weeks, I couldn’t find anything and started conceptualizing how I would build the robotic arms. I sketched out a couple of ideas that involved having one small low-power lightweight motor for the elbow joint, and one bigger 6V ~11000 rpm motor for the shoulder joint. I started building the arms using thin metal corner joints, which didn’t weigh too much. My main concern while building these arms was that the weight would be too much for the motors to lift.

Robotic Arms Initial Sketch, Fritzing:



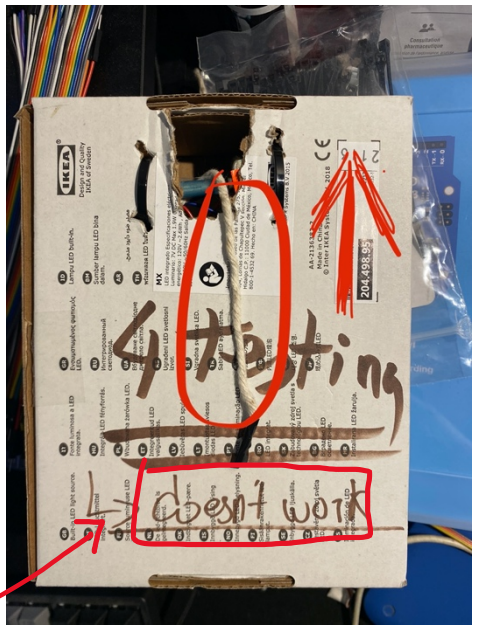
Breadboard Test Hookup





Long behold my concerns turned out to be a reality as not even the biggest motors I bought from ‘Addison Electronics’ would be enough to lift the bicep joint, the forearm joint, and the small motor controlling the forearm joint. So I attached a rubber band with slight tension to the elbow joint and the top of the chassis, thinking the upwards tension from the rubber band would be enough to help the motor lift up the arm. I was very wrong once again.

This led me to create a sort of **pulley** system with 2 motors facing each other, spinning in the same direction. I attached one end of a small kitchen rope to my pulley and attached the other end to the middle joint of the arm. I thought, “if one motor can’t do the trick, why not 3!”. Three motors lifting one joint seemed like the perfect solution.



Pulley System Test:

Act 1 Conclusion:

The result of my experimentation subsequently led me to think my life was a sitcom because no, the third time was NOT the charm. I was left with a pulley system that worked, but not well enough to execute the task I built it for, a robotic arm that moved 2 cm upwards at best, and worst of all the disappointment of finding out none of my ideas would actually work. So I took a break for a little bit in order to air out my mind and give myself time to think of something.

Act 2: The Apple Rabbit Hole

A week after starting my break I had an epiphany. This eureka moment happened right after a workout when I checked my Apple Watch’s health app. There was my pulse, right there! right on the small screen on my wrist. The answer was in front of me the whole time and wasn’t seeing it because I was failing to take a step back and look outside the box. I was so focused on having the Arduino get my pulse through some form of physical interaction, that I didn’t even CONSIDER the notion of wireless.

So there began my research, I began learning about how to create an apple watch application and the process of getting it on the watch itself. But the more I researched and learned, the more the task seemed impossible and far out of my field of proficiency. But I didn't stop there, on my wrist I had the key to getting over all the roadblocks I encountered along the way.

Not wanting to get tunnel vision as I had before, I took a step back and booted up Twitch, a streaming platform where content creators live-stream anything from app-dev to horror games. One would think I would find some form of answer in the app-dev section, but as fate would have it, I had no luck there either. All the streamers I asked talked directly gave me the same answers I already had: "You could probably make an apple watch app and directly deploy it to your personal hardware, but that would require some form of tweaking with your apple watch and probably void your warranty".

I didn't want to risk breaking this very expensive piece of tech, so I dismissed this and kept looking. Soon enough I realized I was out of luck yet again, and joined a friend's stream. Little did I know lady luck had not yet abandoned me.

He was playing a horror game called 'Outlast 2' and had a **pulse monitor** on the bottom left-hand side of his stream!

For context, some streamers have a pulse monitor on their streams when they are partaking in an activity/game which either involves stress, fear, or even adrenaline. In other words, it's a way to show twitch chat how scared/agitated the streamer ACTUALLY is.

This obviously immediately caught my eye, however, I didn't want to get my hopes up just yet, as I had failed countless times beforehand. I joined the discord channel he was in and asked him how he managed to get his heart rate on the stream. He told me about this app on his APPLE WATCH! called 'Health Data Server'. Hearing this, I was obviously both thrilled and relieved, I finally had some kind of direction. I immediately paid the 5$ needed and downloaded the app on my watch. It was the first big step in the right direction.

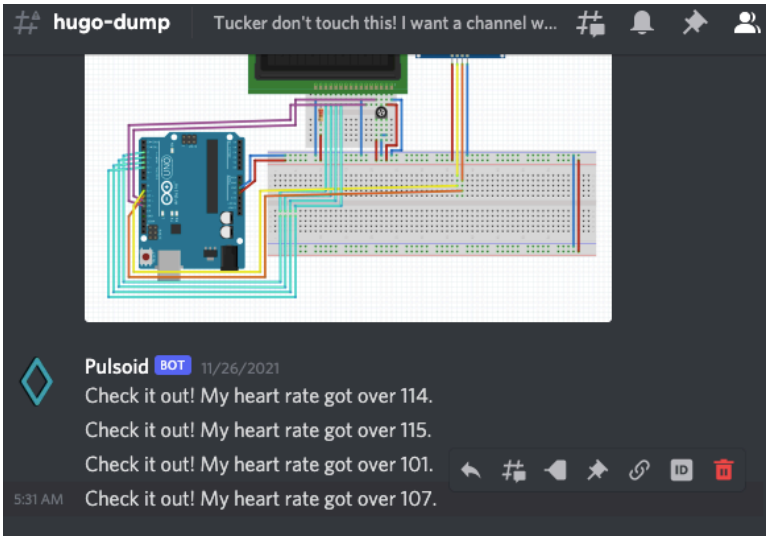
Although as irony and misfortune would have it, I would soon be disappointed to find out I spent 5$ for nothing. The app is not open source therefore has no API documentation. You couldn't even copy and paste the text on the app's interface. Not wanting to waste my time chasing a pig with wings, I swallowed my pride, deleted the app, and moved on. I was disappointed for sure but not demotivated

Act 2 Conclusion:

I then looked for alternatives. This is when I discovered the answer to everything: 'Stromno', the alternative to 'Health Data Server' for streamers. It has the same functionalities as 'Health Data Server', but it's not open-source. However, it can connect to 'Pulsoid', an API which is to a certain extent. I immediately created a Pulsoid account and linked it to my Stromno application. I then looked through the integrations Pulsoid offered and found it could connect to both Discord AND IFTTT. What this meant is I had options.

Final Act: Middleware Meltdown

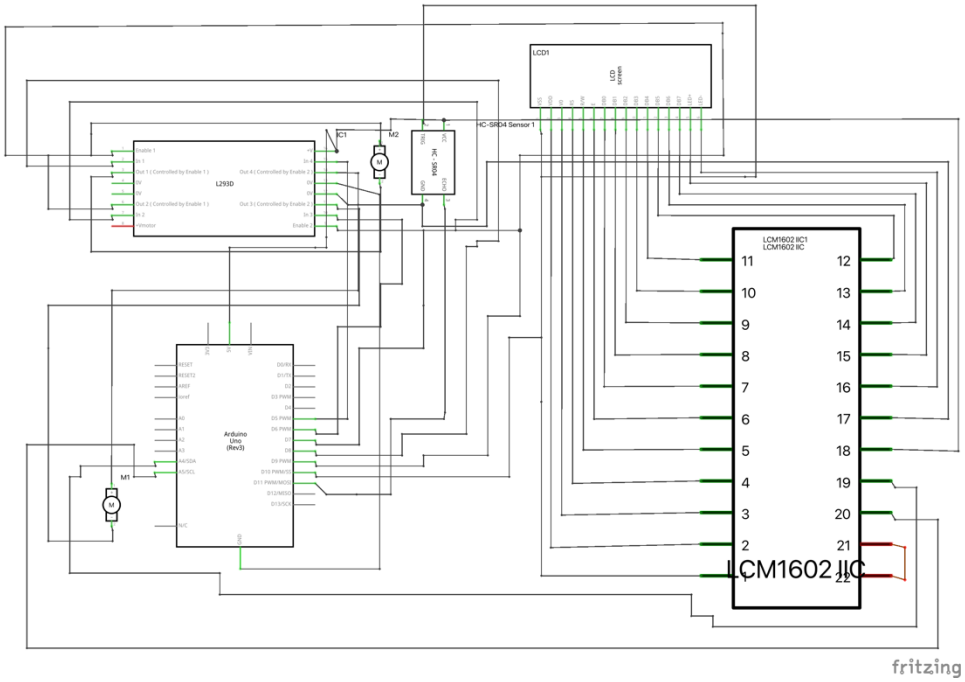
I started by testing out the Discord integration, which worked surprisingly well. The API was able to very quickly post new messages to a text channel at a rapid frequency. Not to mention you could customize how often the Pulsoid bot would post messages through the website, and set threshold values for it to post only certain readings. However, fetching Discord message logs would involve me creating my own bot, which I had never done before (plus I was running out of time, so I had to stick to what I already knew now).

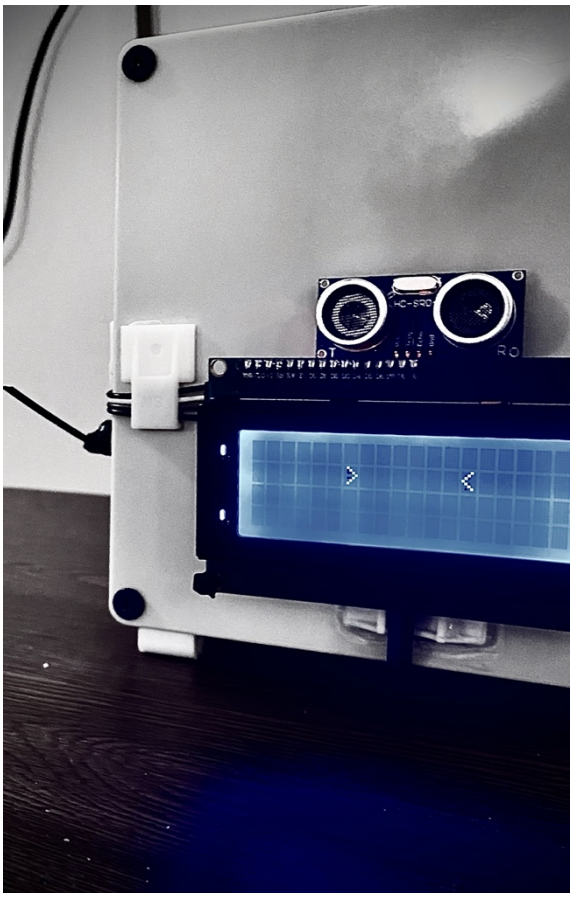


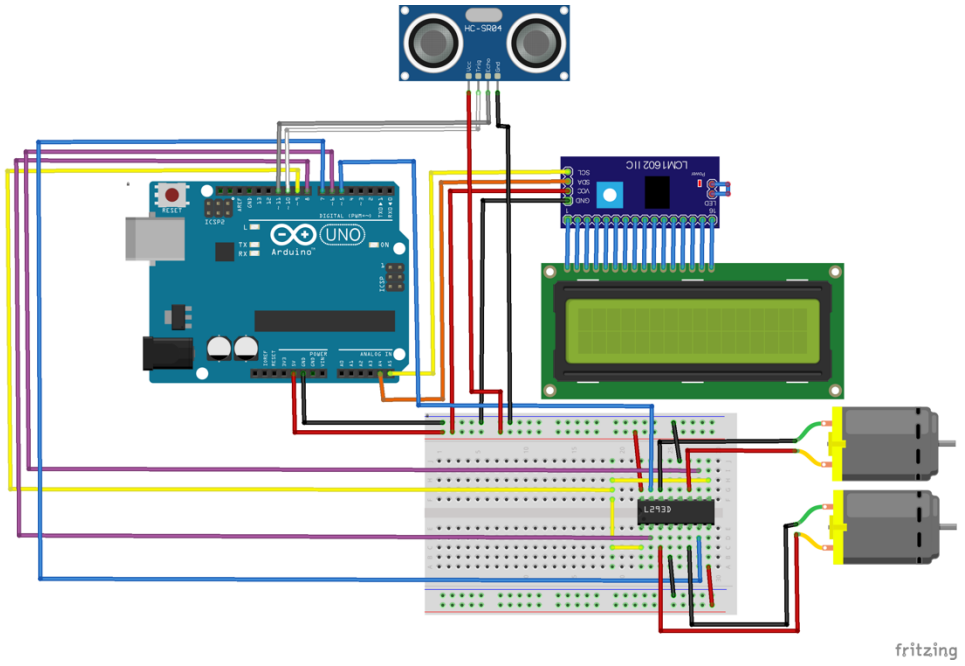
I then turned my attention towards IFTTT, also known as: 'If This Then That'. Luck was on my side because apparently, the one-hundredth time is the charm. IFTTT can fetch Pulsoid webhooks, take the data inside of them, and send that data as a POST request to a website (*Although I must add that at first I thought of sending the incoming webhooks to GitHub Gists, but yet again, scrapped this idea too as it only posted NEW Gists instead of altering already existing ones*). But I was finally face to face with something I was familiar with, Node.js.

I quickly created a very small node.js application using express. It's sole purpose was to take POST requests, Stringify the json from the post request, and alter said string so it's only the numeric value of the post request body (turn: *"{'89', ' '}"* into => *"89"*). Now we are getting somewhere. I must add that I have not yet talked about

This string was then written to a text file called 'heartRate.txt' located in my node project folder. Which begs the question: how do I get Arduino to read a text file? Well the simple answer is you can't, Arduino is separate from your computer and cannot access it's files. However! Arduino can read incoming serial communication coming through the USB port it's connected to. So I wrote a small python script that read heartRate.txt, encoded it to bytes, and shot that data to the Arduino through the USB. After more debugging then I originally anticipated, I finally had pulse values right where I wanted them. No need for fancy robotic arms anymore, no need to order anything else from China, none of that! I finally found my treasure and my journey was coming to an end. Here is sketch and schematic for the final product along with some hard earned pictures of my darling creation working:







Final Act Conclusion:

Although this journey was tedious, highly stressful, and quite frankly painful at times. This taught me a valuable lesson, sometimes the answer isn't to overcome the roadblocks which block your path. Rather than climbing over, sometimes it's easier to take the long way and walk around it.