Materials & Methods

Hardware Approach

The overarching goal of gaining control over each component in the DVD OPU remains the same, but the means of doing so will now be explored using hardware. The idea is to use existing off-the-shelf components (i.e. H-bridges) and wire them directly to the components. What needs to be done from the hardware perspective is to develop the scanning stage. The rest will be software scripts to control the actual motion of going to each point and recording a height value.

Begin by selecting a DVD drive, as shown below. This can be done by unscrewing a PC computer and locating the drive, or these drives can be purchased online. Unscrew the top so the inner components are accessible.

Figure 5 - Pictures of DVD Drive

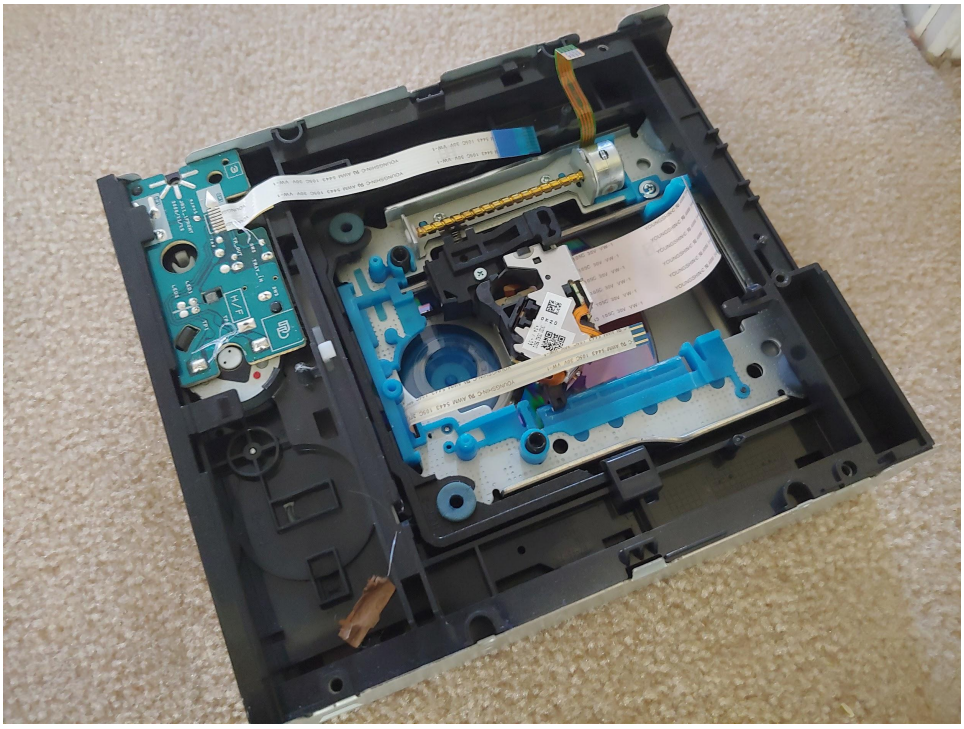


Figure 6 - Inside of DVD Drive with Casing Removed

Now having removed the top, observe the brushed DC motor in the top left of the image and the stepper motor connected to the linear stage in the center of the OPU. These are the main components that will be utilized for developing the motion stage.

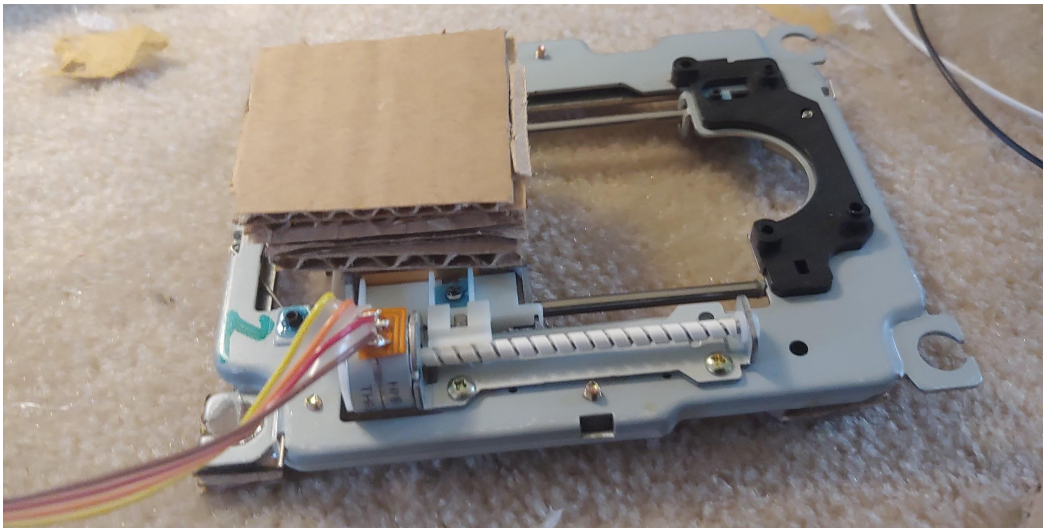


Figure 7 - Stepper Motor

Continue dismantling the DVD drive until the components are freely accessible as shown in the images above. For the brushed DC motor, solder jumper wires to the exposed leads and connect this to a breadboard with 12V. For this project, the power supply of an old PC was used to supply 12V, but other sources such as a 12V battery can be used.

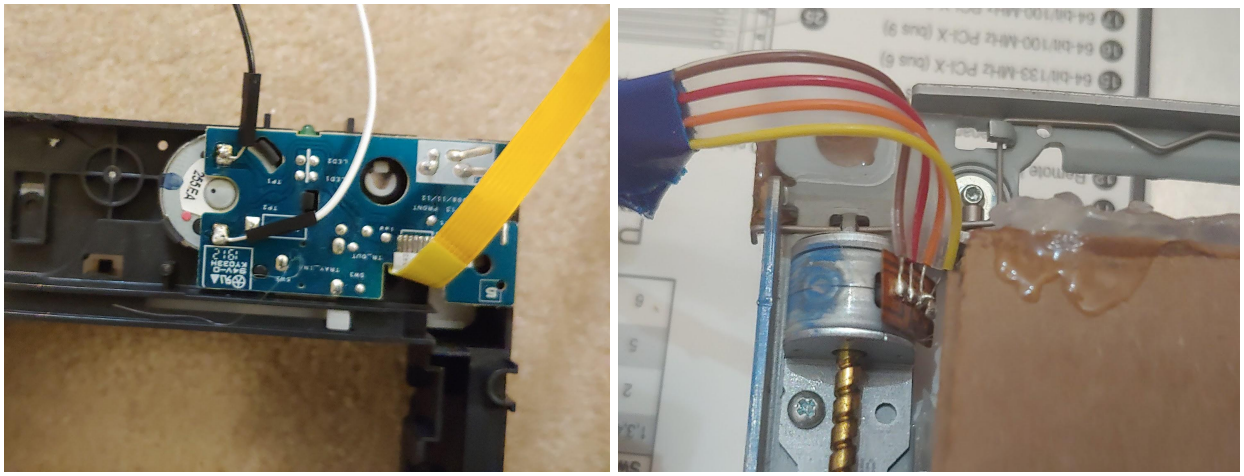


Figure 8 - Soldered leads to DC, Stepper Motors

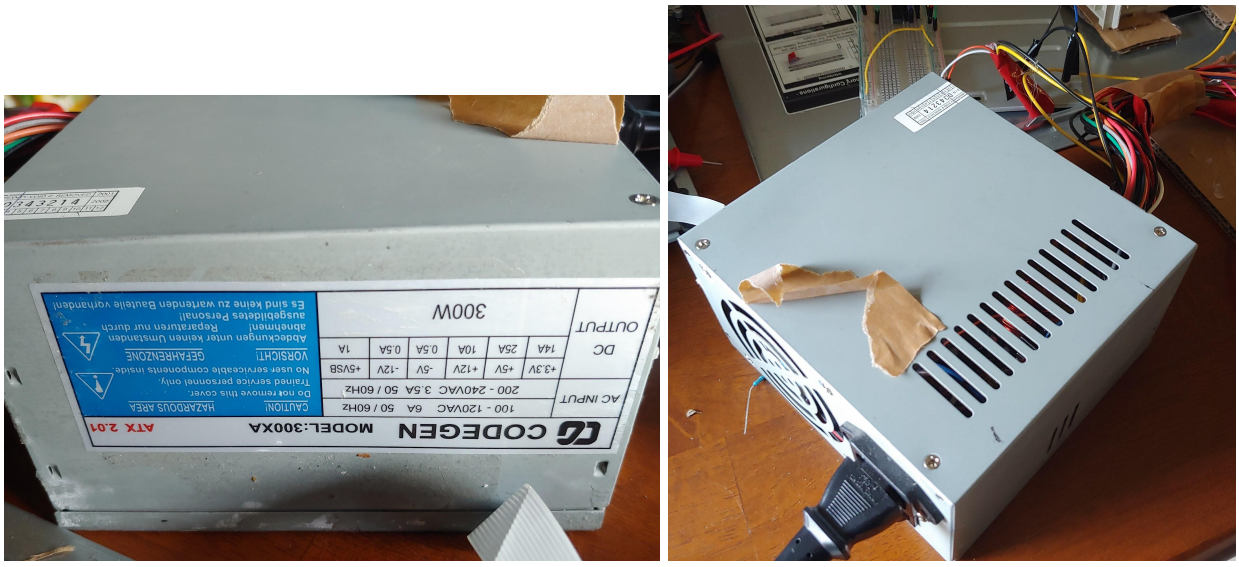


Figure 9 - Power Supply Used

Use hot-glue and cardboard to add a cardboard platform onto one of the gears on the DC motor. This will be the rotating stage. The image below shows where the platform would be using straws and a battery. These objects were placed there temporarily as a test to see if the motor could support running a stage.

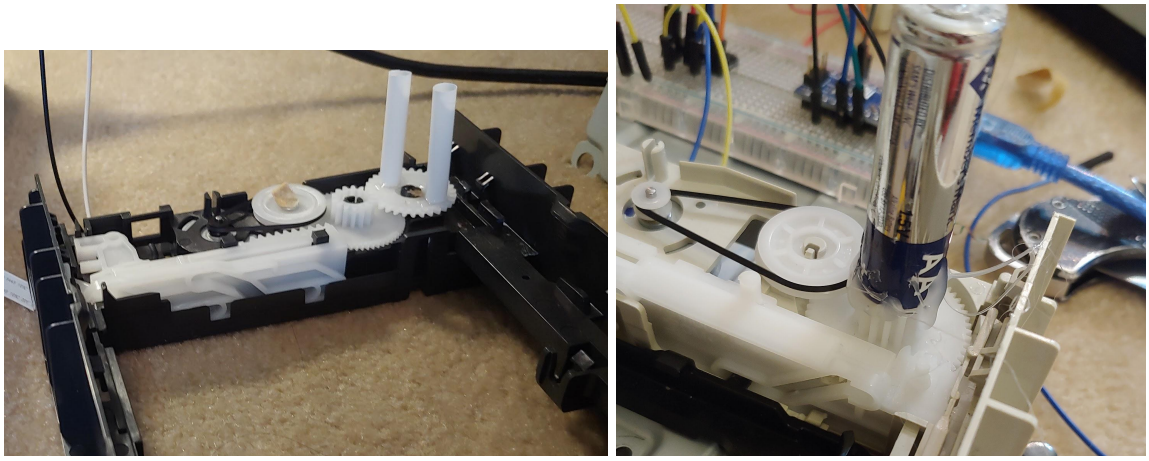
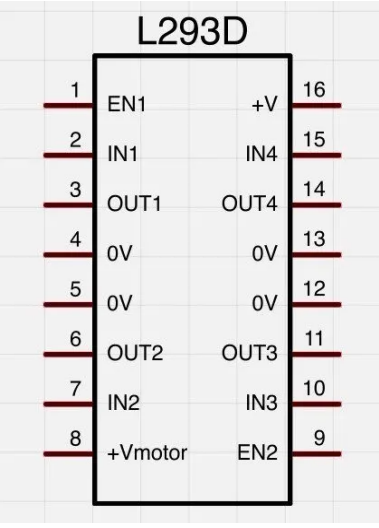


Figure 10 - Brushed DC Motor and Gear Chain

To control these motors, the MP6500 and L293D H-bridges were used, with the former controlling the stepper motor and the latter controlling the DC motor. Pinout configurations of the chips used are shown below, as well as pictures of the wiring on the breadboard used:



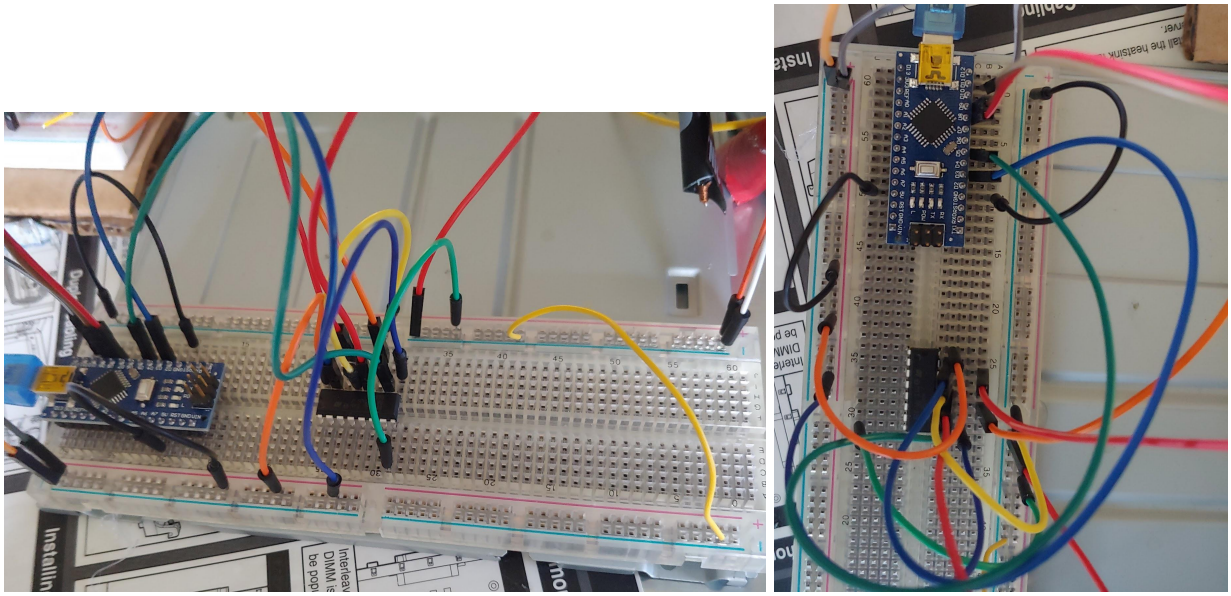


Figure 11 - Pinout diagrams of L293D & Implementation

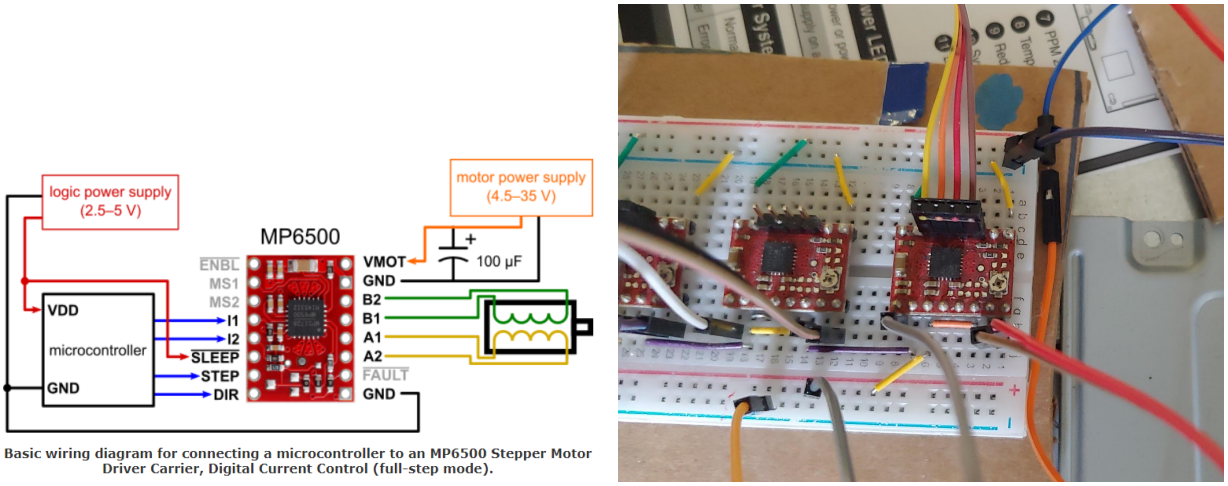


Figure 12 - Pinout diagrams of MP6500 & Implementation

Altogether, the wiring is shown below. Now proceed to developing the actual motion stage by gluing a cardboard stage to one of the gears of the DC motor. This will control the "theta" aspect of the polar coordinates. It should end up looking as such:

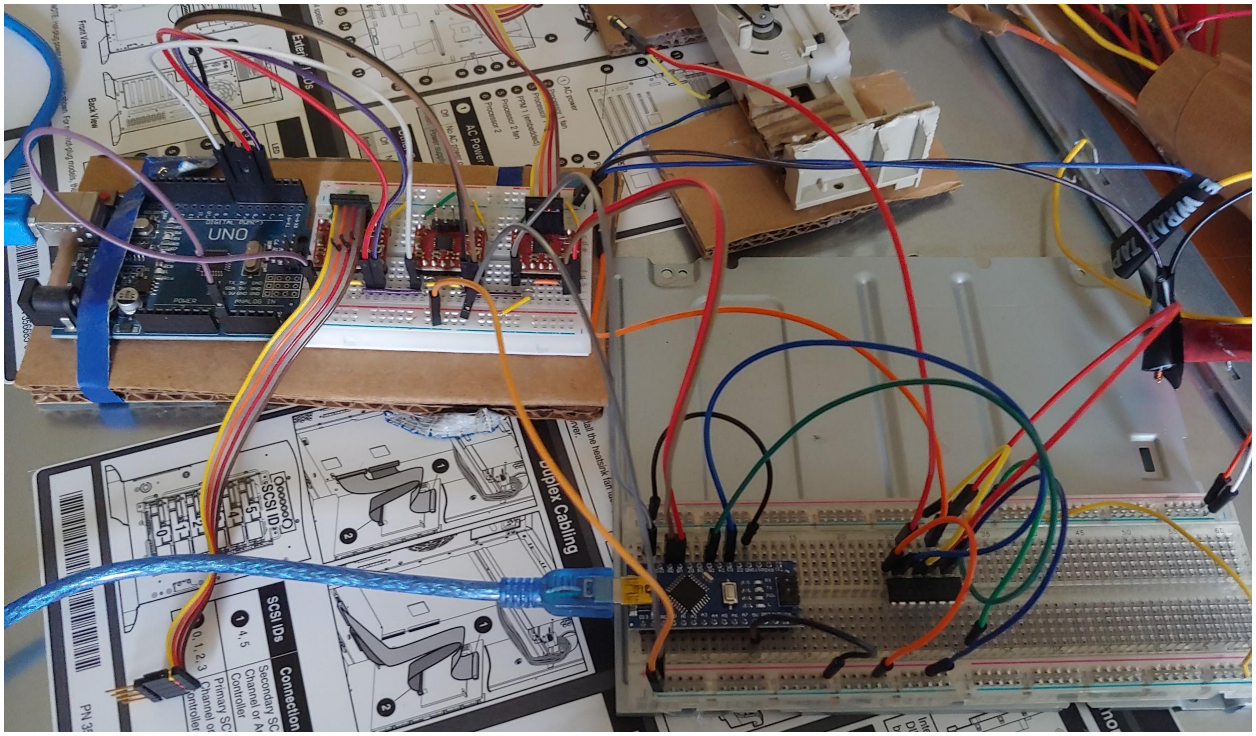


Figure 13 - Overall Wiring Setup

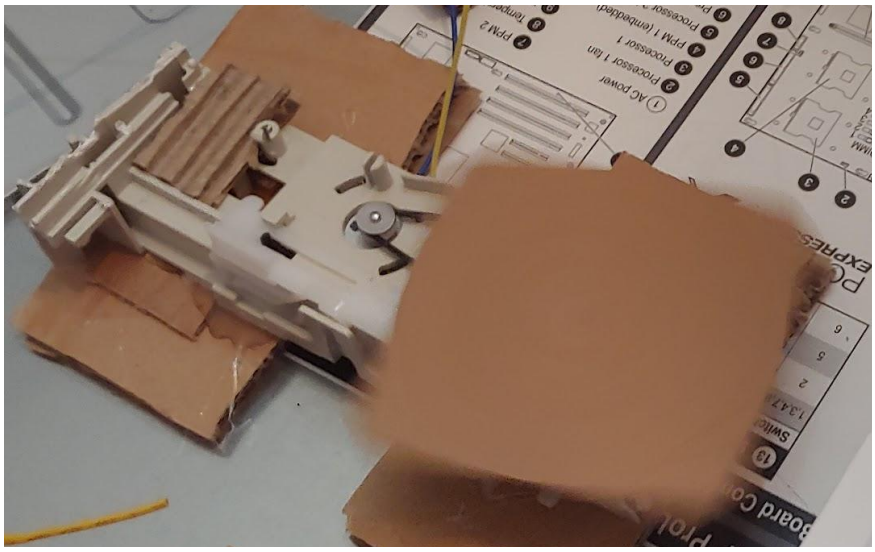


Figure 14 - Cardboard Rotation Stage on DC Motor Gears

Now begin developing the rest of the motion stage involving the stepper motors. Use hot-glue and cardboard to develop the type of configuration shown below. The

horizontally-placed stepper motor will control the “radius” in the motion stage, while the 2 stepper motors on each side will control the “z”.

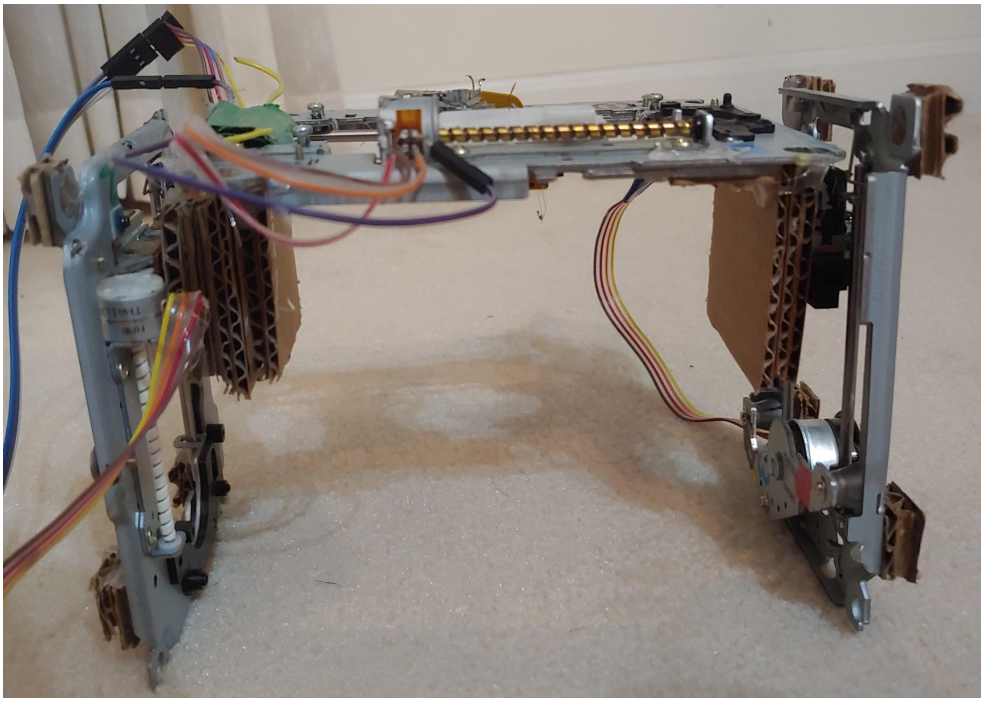


Figure 15 - R & Z Control

Lastly, glue this stage above the rotating platform and the hardware components are completed.

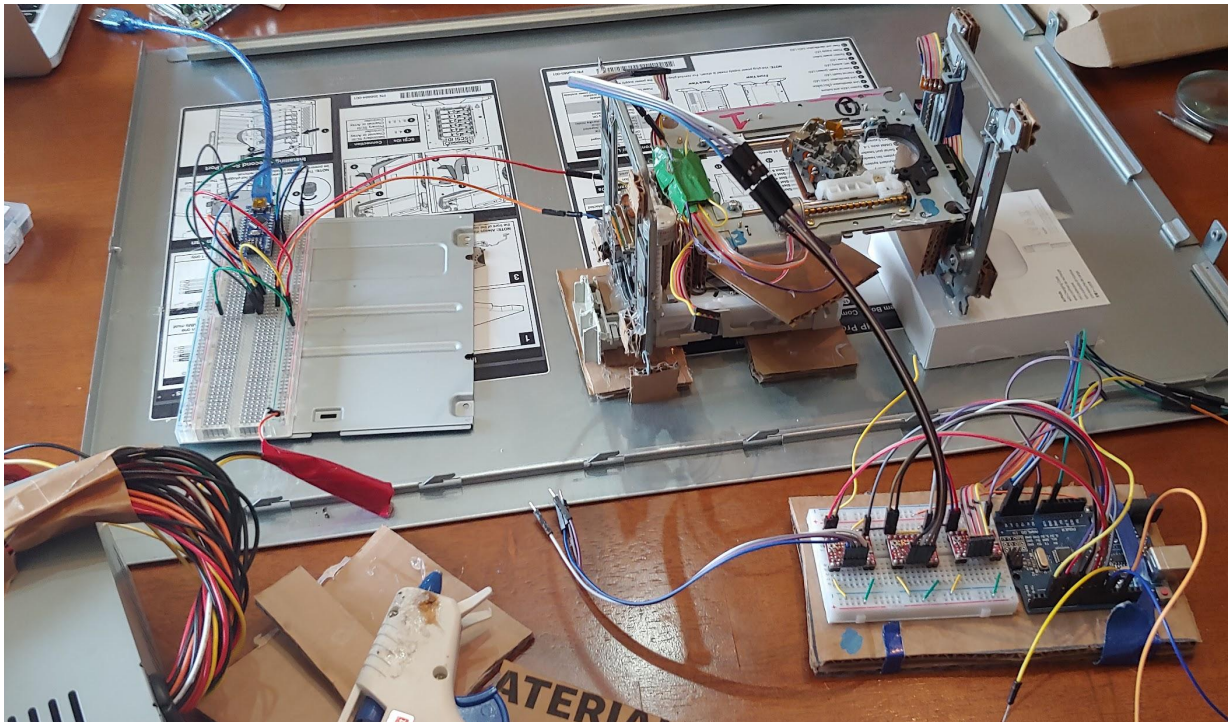


Figure 16 - Final Product

Software Approach

For the software, a program needs to be written that can control the movement of each parameter in the polar coordinates system: r, θ, and z. As mentioned earlier, θ will be controlled using the DC motor, while r & z will be controlled using the stepper motors. An Arduino Nano microcontroller was used for wiring the stepper / DC motors, so the program will be written in C++ and can be found on Github (Program.ino). Attached below are some screenshots of the program.

```
//Written with the help of Imaad Syed and Dr. Lafe Spietz

#include <Stepper.h>

int delayus = 150;
int dirPin3 = 8;
int stepPin3 = 9;
int enPin3 = 10;

void setup() {
  // put your setup code here, to run once:
    Serial.begin(9600);
    pinMode(3, OUTPUT);
    pinMode(5, OUTPUT);


    pinMode(dirPin3,OUTPUT);
    pinMode(stepPin3,OUTPUT);
    pinMode(enPin3,OUTPUT);


    digitalWrite(dirPin3,LOW);
    digitalWrite(stepPin3,LOW);
    digitalWrite(enPin3,HIGH);
}

void loop() {
  // put your main code here, to run repeatedly:

  digitalWrite(5, LOW);
  analogWrite(3, 175);
  //delay(100);

  moveUp(400);
  delay(400);
  moveDown(400);
  delay(400);

}

void moveDown(int nSteps){
     digitalWrite(dirPin3,LOW);
    digitalWrite(enPin3,LOW);

    for(int index = 0;index < nSteps;index++){
      digitalWrite(stepPin3,HIGH);
      delayMicroseconds(delayus);
      digitalWrite(stepPin3,LOW);
      delayMicroseconds(delayus);
    }
    digitalWrite(enPin3,HIGH);
}

void moveUp(int nSteps){
    digitalWrite(dirPin3,HIGH);
    digitalWrite(enPin3,LOW);

    for(int index = 0;index < nSteps;index++){
      digitalWrite(stepPin3,HIGH);
      delayMicroseconds(delayus);
      digitalWrite(stepPin3,LOW);
      delayMicroseconds(delayus);
    }
    digitalWrite(enPin3,HIGH);
}
```

Figure 17 - Program Screenshots