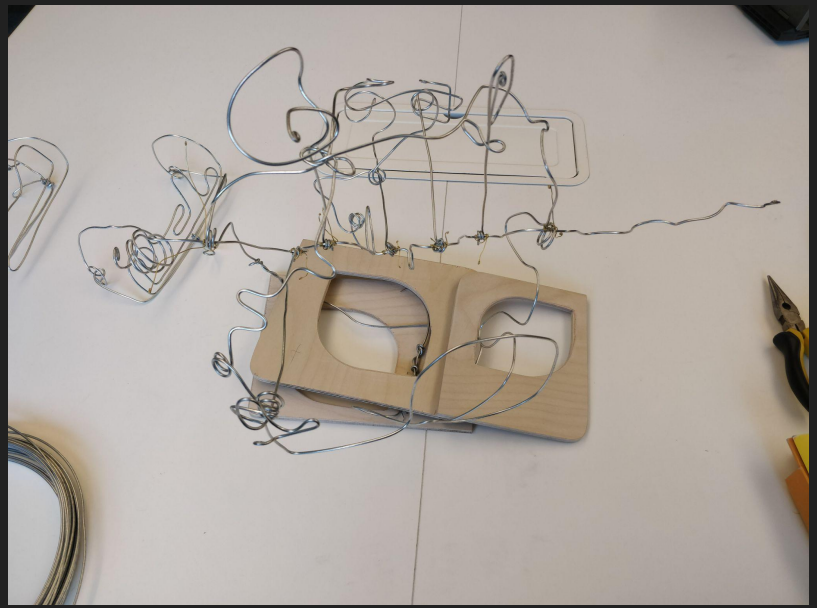
[mutual ground]

The final report

EVERYWHERE – abstract

To socialize and exist as a individual participant within a society requires a lingua franca – a shared means of communication. To socialize, to form connections with people outside of yourself, to put yourself out there, comes with misunderstandings and assumptions and uncomfortability. This piece comes out of a desire to explore the making of a tangible lingua franca, an abstract means of communication that encapsulates all the challenges that come with it. This report covers the process of developing the artifact, the final artifact, observations, and future directions.

EVERYWHERE – abstract

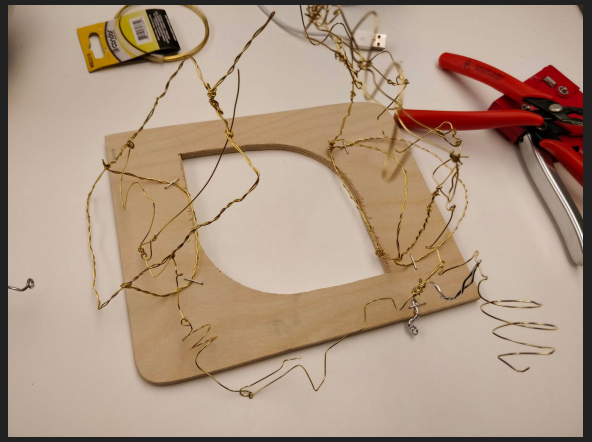


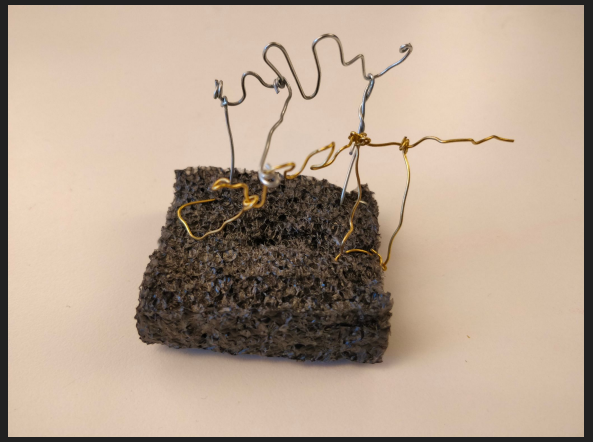
[mutual ground] is an installation piece that intervenes in public and private spaces where it is displayed on an interpersonal-communications level. It can be seen as a specialized electronic device in the home or workplace, or a publicly available resource in an exhibition setting, meant to challenge the user's conception of two-way streams of communication between people via an intimate & transformative physical contact interface.

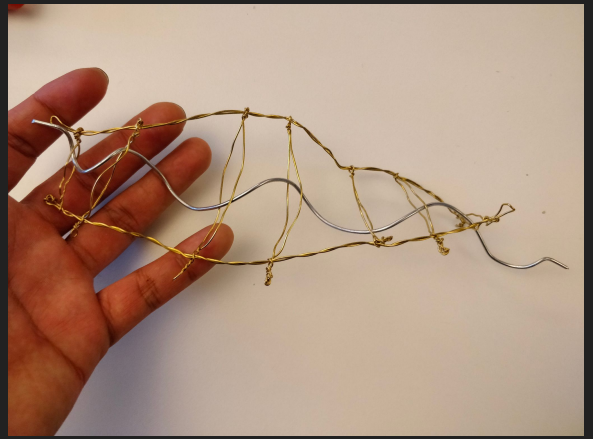
The piece is meant to foster a conversation-type interaction among two people entangled in abstract sensorial communication with each other and the device, which reacts to input from capacitor sensors placed in the users' forearms and on the device itsef. The device transforms sensor input into kinetic sculptural movement that can be experienced by touching the shifting surface of the artifact. Users are afforded a way of transmitting signals to each other through a mutual performance that encourages a breakthrough past the initial discomfort of physical touch needed to activate the device, into a type of understanding and collaboration.

NOWHERE – Prototype Process

Sketching. Building. Wire exploration. Automata attempts on a small scale. Familiarizing with materials and movement. Exploring haptics, sculptures, cages, and cranks. By the time the prototype presentation rolled around, we learned that our current wire was not thick enough to strike the balance between structural integrity and malleability we wanted, and were lacking in actual computing.







Design Narrative & Prototype process

Our intention was to create a conversation type interaction that simulates the process of initial discomfort to discovery of social interaction through an abstracted means of communication. To this end we set out to construct a kinetic sculpture out of wire that would move and respond according to user input.

The choice of material allowed us to narrow down our focus and technique, and encouraged us to develop the material skills to craft wire structures. Our initial attempts were not very structurally sound.

For our second iteration of the device we used a thicker grade of wire, which helped a lot with stability and the integrity of joints and movable parts. The main axis of motion was much more reliable as a functional crank. However the reduced malleability of the wire also brought more challenges, bending it to very specific shapes, in tight arches, or around the base to fix it in place or 'tie' new pieces to the structure required pliers and considerable force.

As more embellishment was added, and also the attachment for the motor, the shape and relative positioning of the moving poles became very important to the functioning of the device. A slight misalignment, a part colliding with another and getting caught, or friction resulting from increased tension at a given section would cause the motor to stop.

We took to working with the motor connected to power for continuous periods of time as we made adjustments to the structure, testing and modifying its shape, re enforcing the motor base and, sometimes cutting wires that had tightened to much, and adding new decorative embellishment.

At some point it became necessary to remove some of the movable poles attached to the main crank axis in order to reduce the friction and simplify the debugging process, we ended up with 3 out of the 5 original 'whiskers'.

We took care to bend back the sharp points resulting from cuts in the wire.

In terms of circuitry we spent some time developing, the capacitive sensors, using a specialized Arduino library and experimenting with detecting value ranges. The results were often unpredictable and we got the best results with a series of 3 100k ohm resistors wired in series and with the computer (acting as power source for the Arduino) was connected to an electrical socket.

The development of the motor code went through two major iterations, at first we thought it would be necessary to use a rotary encoder and a L293D driver and a 10K ohm potentiometer to manipulate the speed. This thread of experimentation did not work out, we could not get the proposed circuit to run.

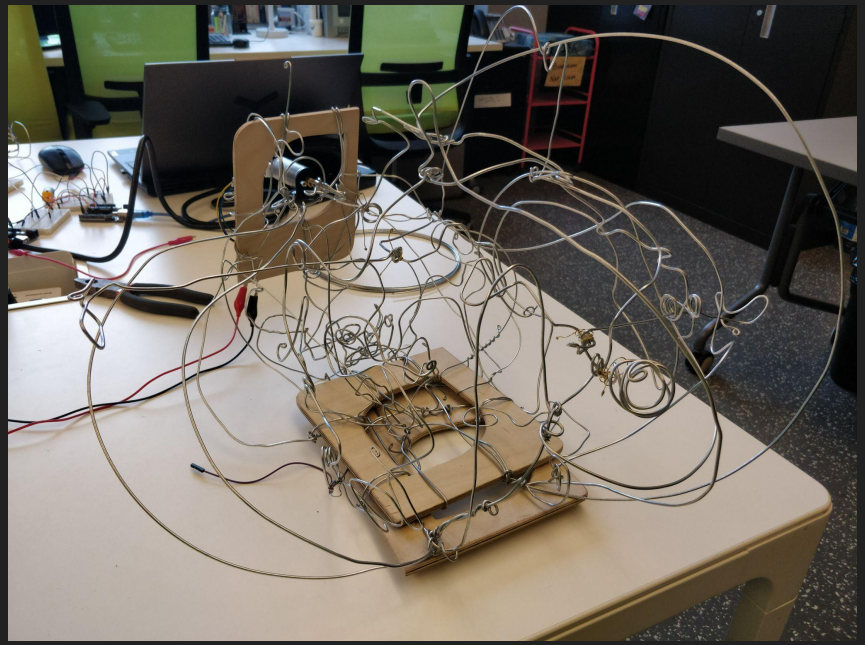
Instead we used a MOSFET transistor to modify the voltage coming in to the motor from a secondary power source through code. The motor needed a minimum of 9 volts of reliable power in order to move, and its lowest speed was still really much faster than we expected, this gave us a very subtly changing range of speeds to work with in the final implementation.

NOW – Final Artifact

Initially, we were planning on relying on shared muscle sensors to power the automata crank but due to their unreliability, switched to capacitive sensors within the automata, as well as shared bracelets to maintain the physical contact.

The final artefact is a dense cloud of wire tangle, the movement produced by the motor too fast and violent, the curves on the bracelets and the spherical shape of the kinetic sculpture range from almost graceful to chaotic. It is perfect in every way to us.

The interaction loop is such that the kinetic sculpture must be touched in order to begin it's functioning, and then turn on-and off at slightly different speed according to configurations of contact made with the components of the piece (the capacitance-enabled bracelets and the kinetic sculpture). This is achieved through code by the definition of states according to capacitance value for each component, these states are then grouped into configurations of states that add an extra layer of unpredictability to the outcome. An initial 'if' statement surrounding these second order state-configuration conditions enables the initial 'touch to activate' feature of the system.



Observations

The unpredictable nature of the state configurations, along with the violent nature of the artifact's movement, made it so that it often felt like touching the device would calm it. It seemed that a common result of touching the device, after the touch-to-activate initial interaction, was that it would turn off. We came to see this as signifying a sort of reassurance, a taming of the beast of social interaction.

Future Directions

One possible extension to the design of the kinetic sculpture that came up during development was the addition of handles to allow users to modify and bend the shape of the structure. This came from the experience that we had building it up as the motor was running. It would add another very tangible level of feedback that speaks to the fruits of collaboration.

The circuitry could be streamlined and secured to the sculpture, the wires attaching the board to the bracelets would benefit from being lengthened and made more reliable for human use.

Also a lot of work could be done on the atmosphere and framing of the interaction. Perhaps by adding posters with instructions or designing the space around it to support the themes. Setting this artifact up as an installation rather than as an object would be

Bibliography

- Capacitive Sensing for Dummies | https://www.instructables.com/Capacitive-Sensing-for-Dummies/
- Arduino DC motor speed and direction control with L293D | https://simple-circuit.com/arduino-dc-motor-speed-direction-control-l293d/
- Atau Tanaka - Meta Gesture: https://www.youtube.com/watch?v=bvaws0wMgDc&t=2300s
- Tauba Auerbach & Cameron Mesirow, The Auerglass: http://www.taubaauerbach.com/view.php?id=243
- Theo Jansen, The Strandbeest: https://www.strandbeest.com/

Process videos

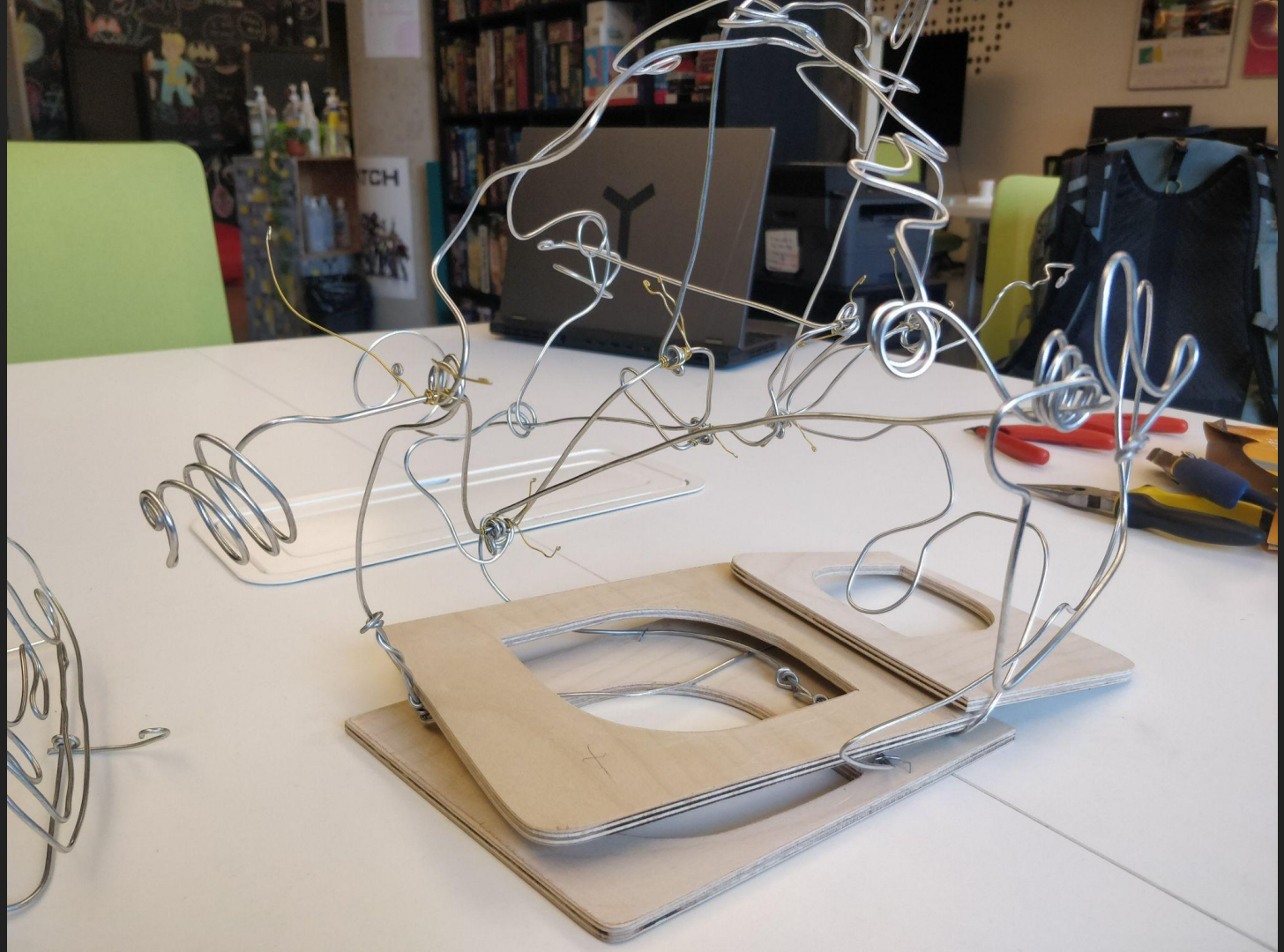
- https://youtu.be/ekgDJwysuzA | [mutual ground] V2 - process vid 1
- https://youtu.be/Q-nD2iNGlek | [mutual ground] V2 - process vid 2
- https://youtube.com/shorts/65cno7eh0Z4?feature=share | [mutual ground] V2 - process vid 3
- https://youtu.be/HoTxoNIX_C0 | [mutual ground] V2 - process vid 4

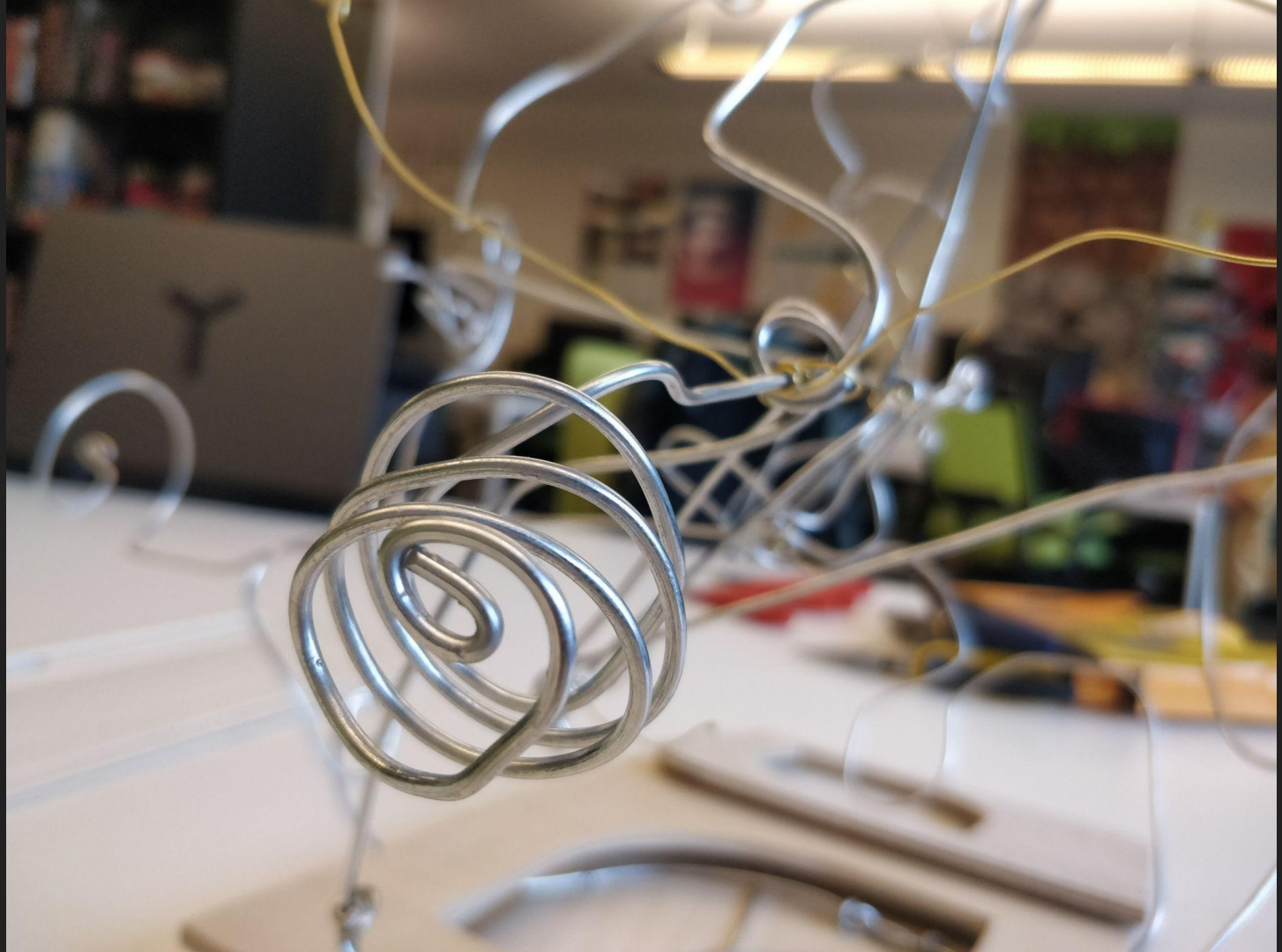
Interaction videos

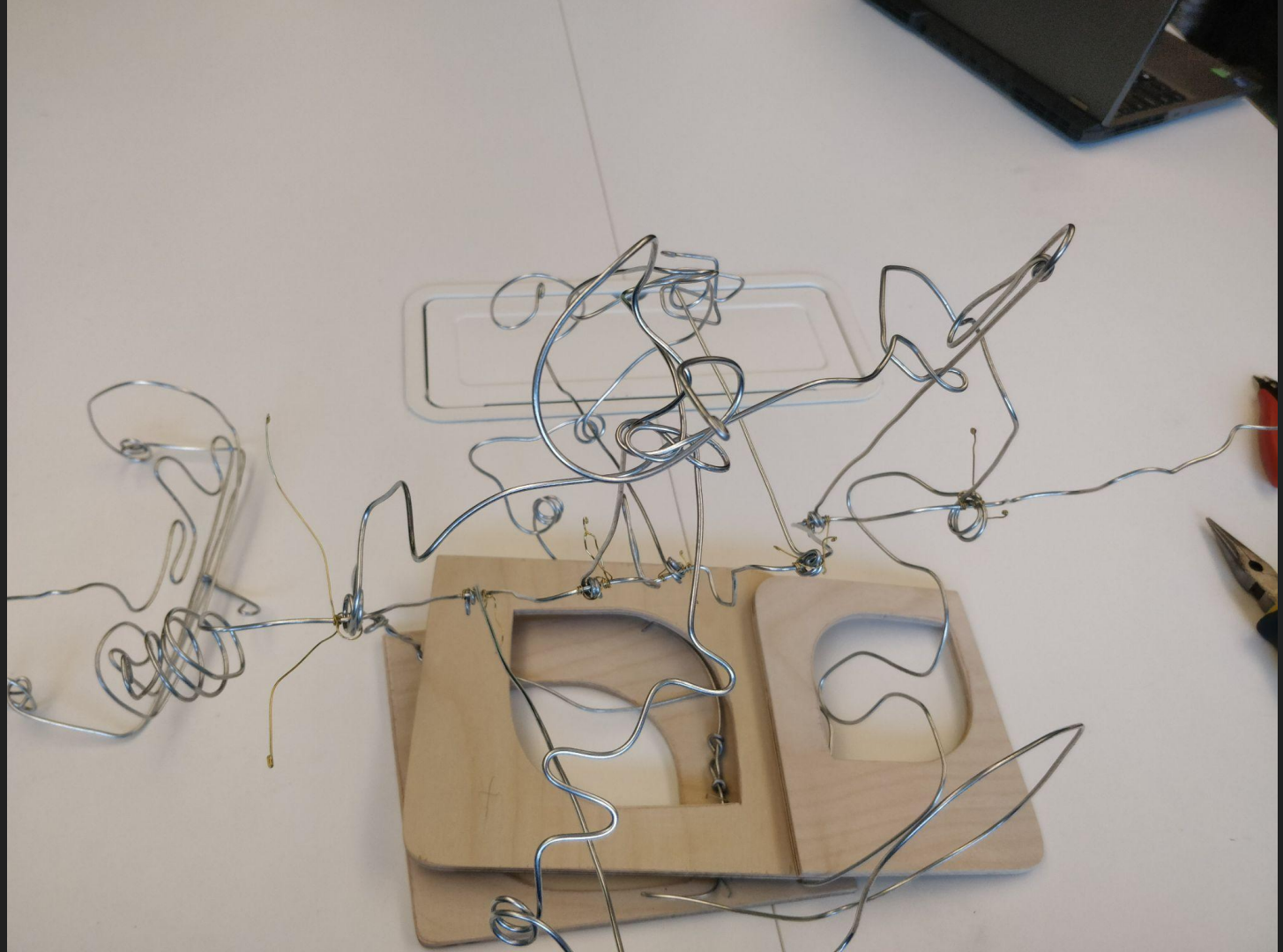
- https://youtu.be/klMMoPvIq_E | [mutual ground] V2 - interact vid 1

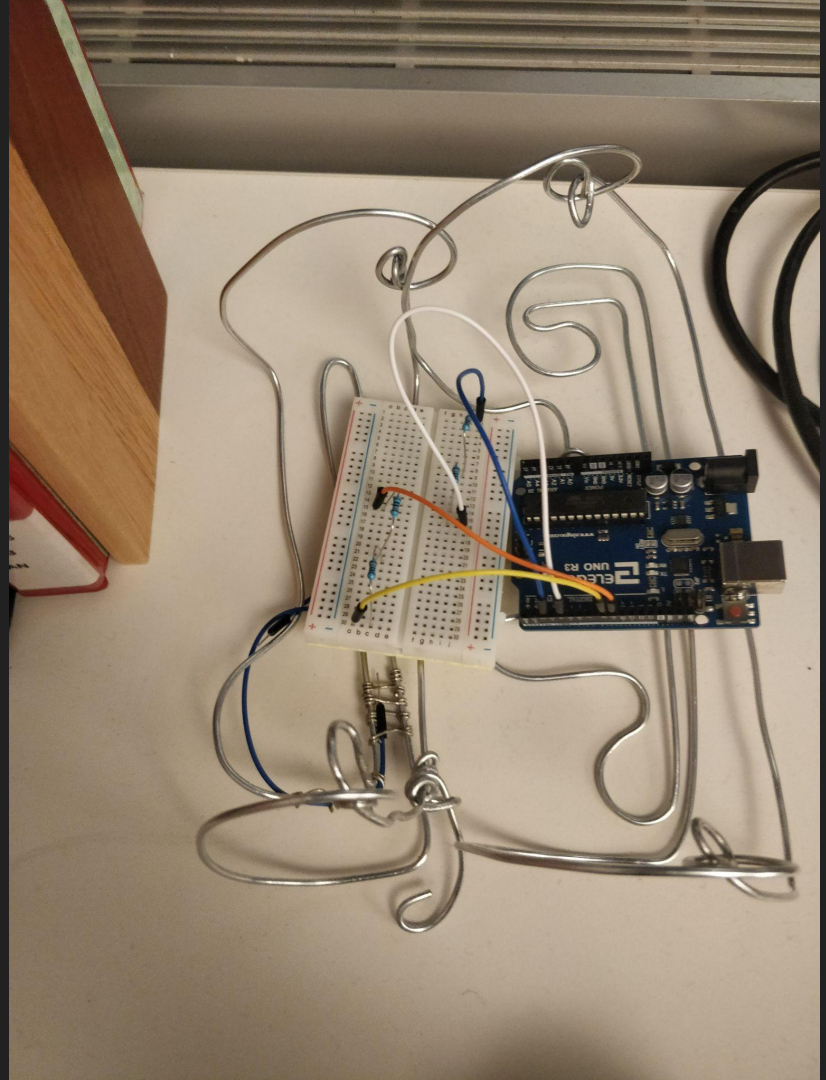
- https://youtu.be/Qnx_4A49nms | [mutual ground] V2 - interact vid 2

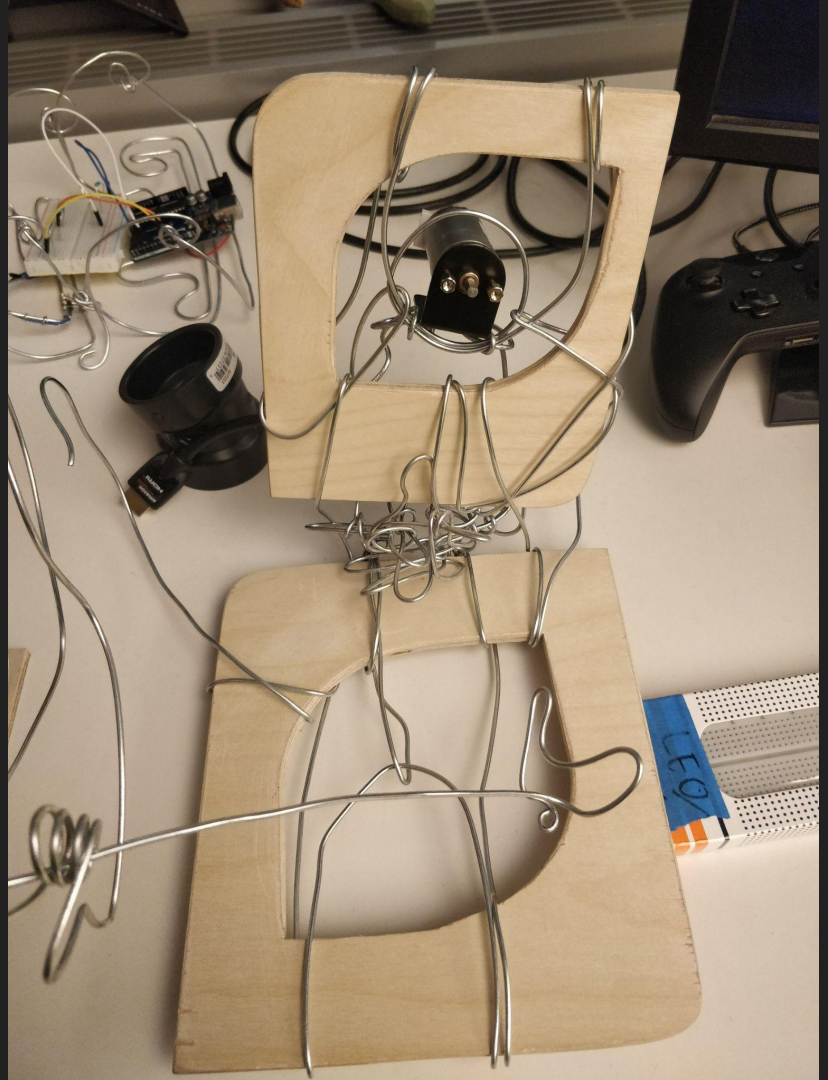
- https://youtu.be/16GgdXAtgSI | [mutual ground] V2 - interact vid 3

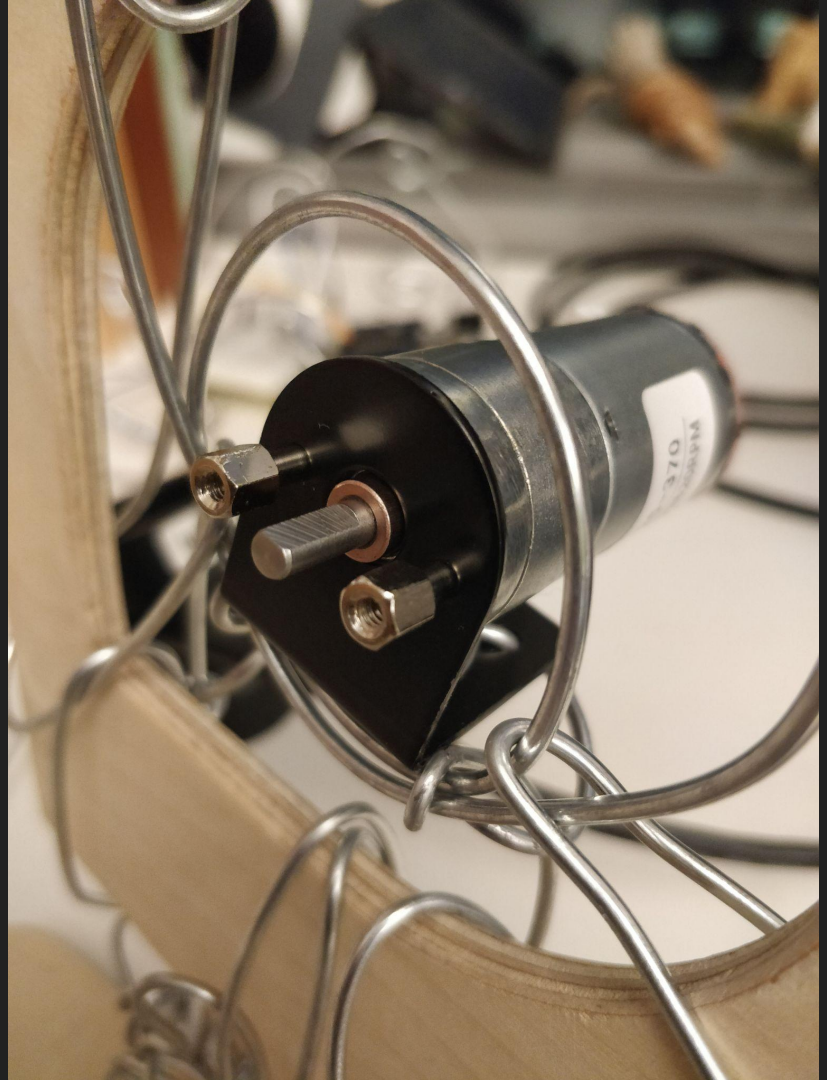


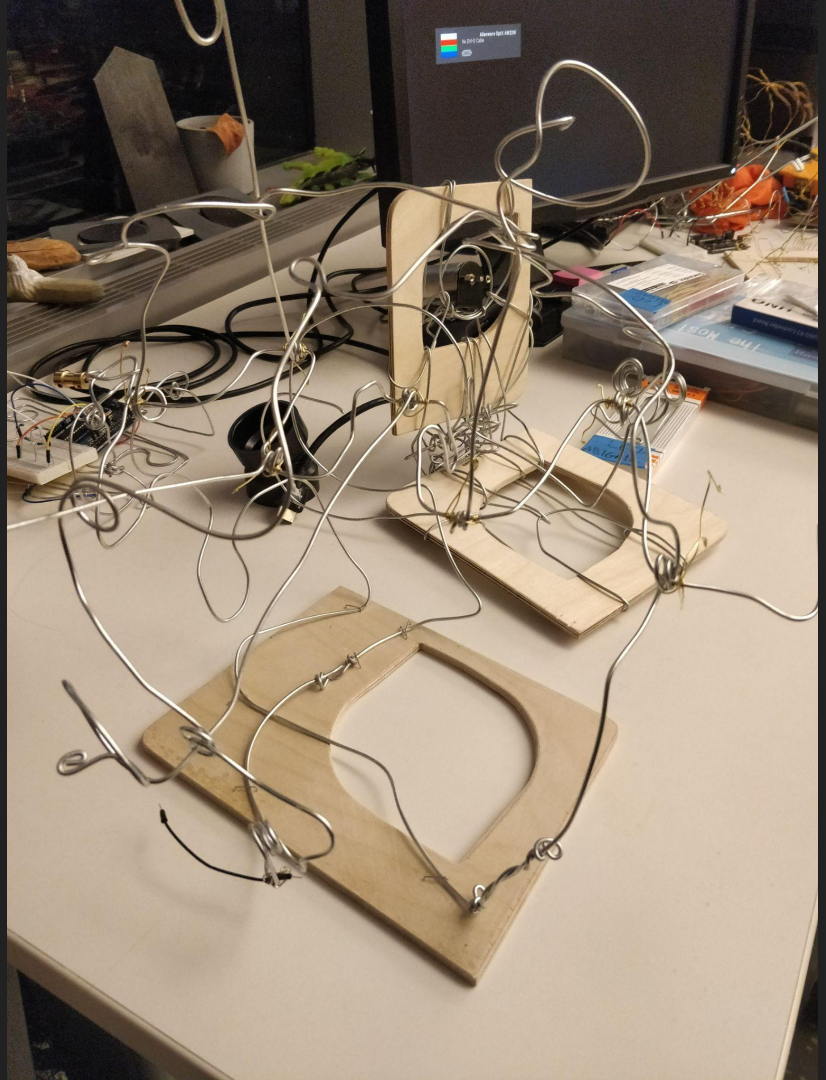


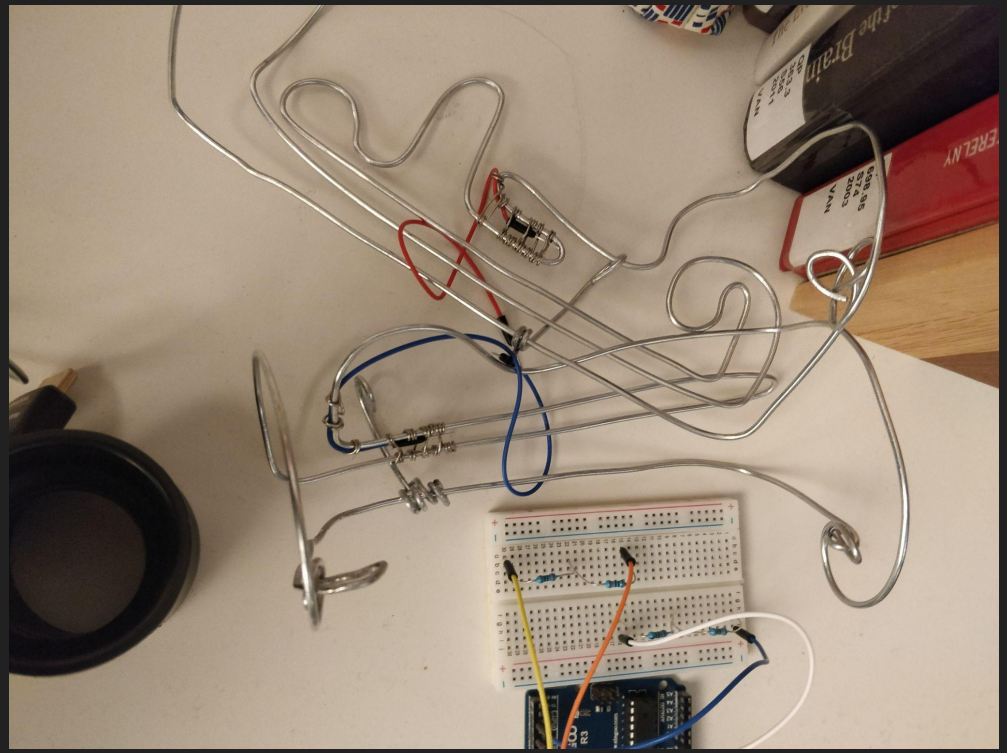






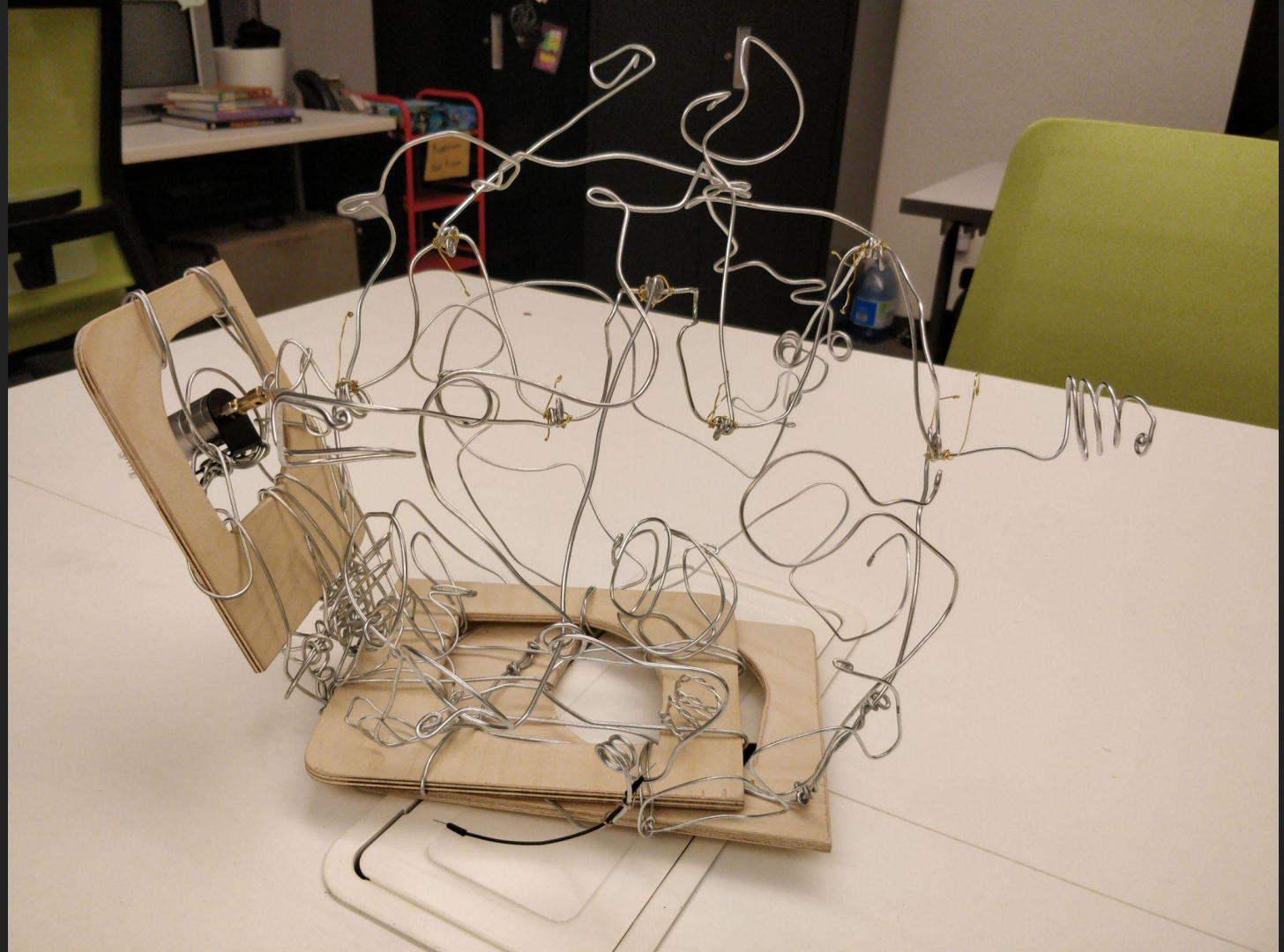


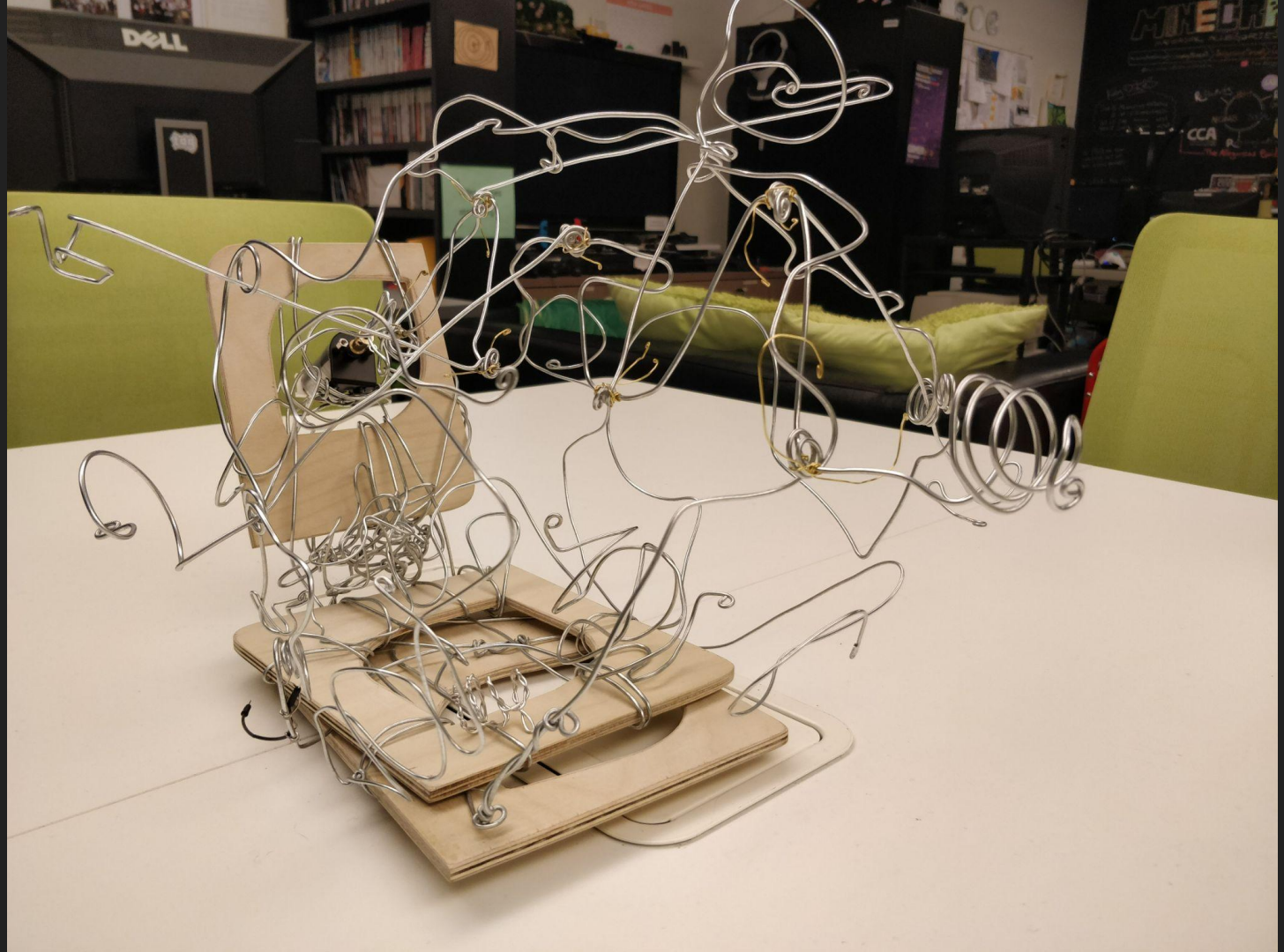




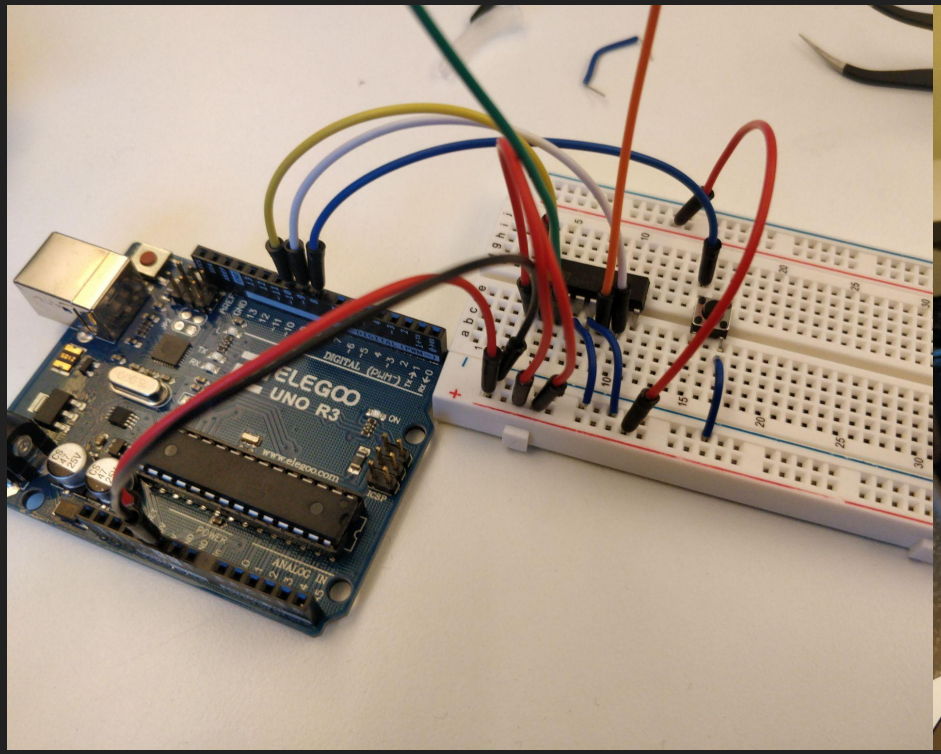

DC12V6-20RPM

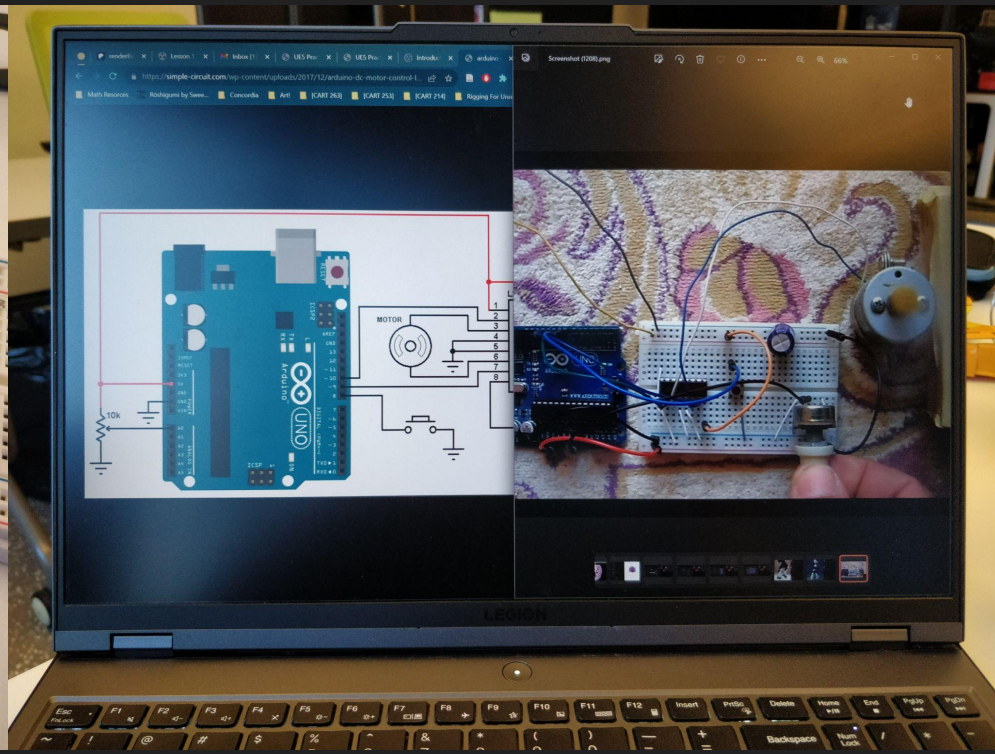
Bot Replicator 2
Networks of the Brain
A HOSTILE WORLD

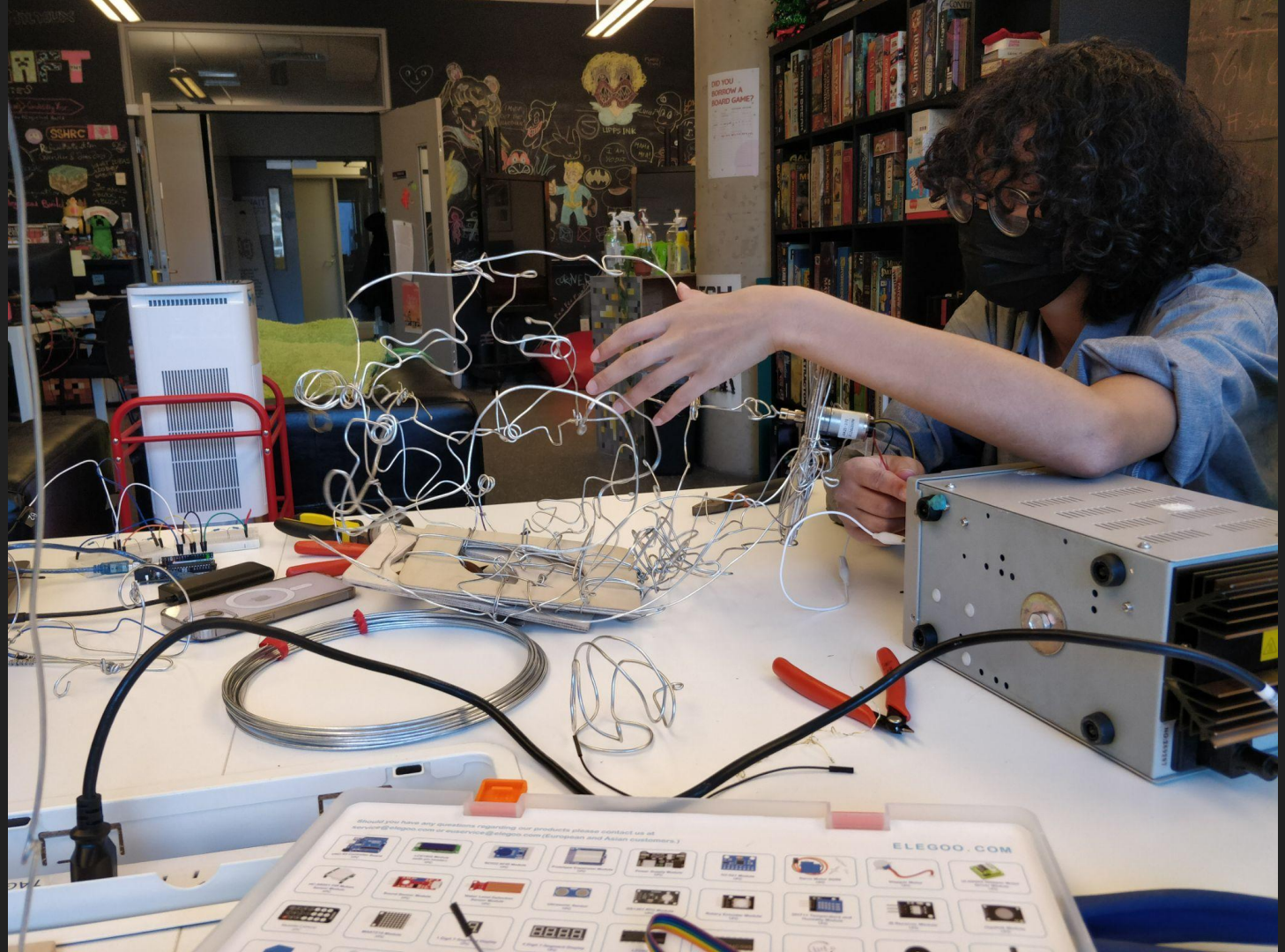


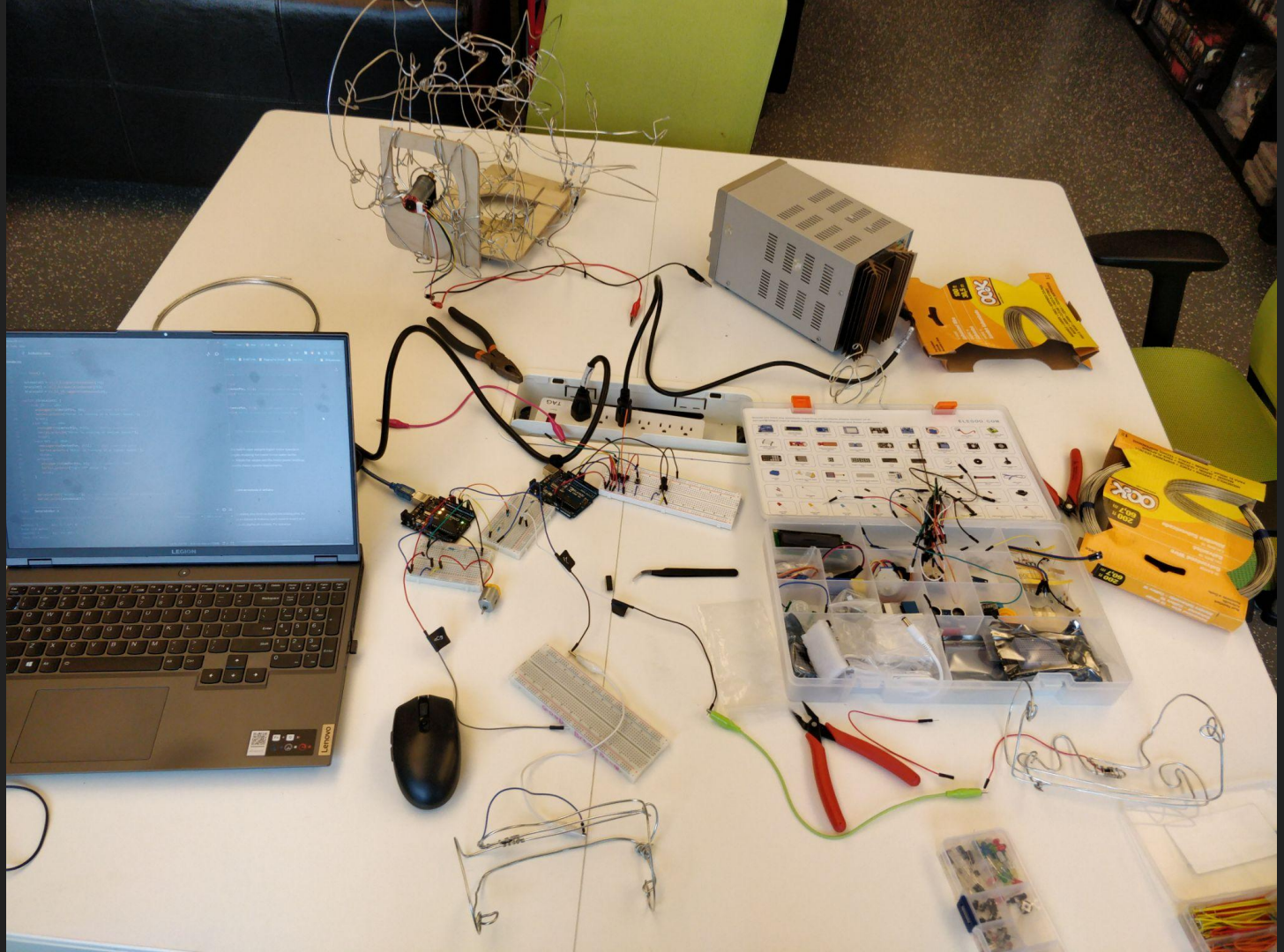

DELL

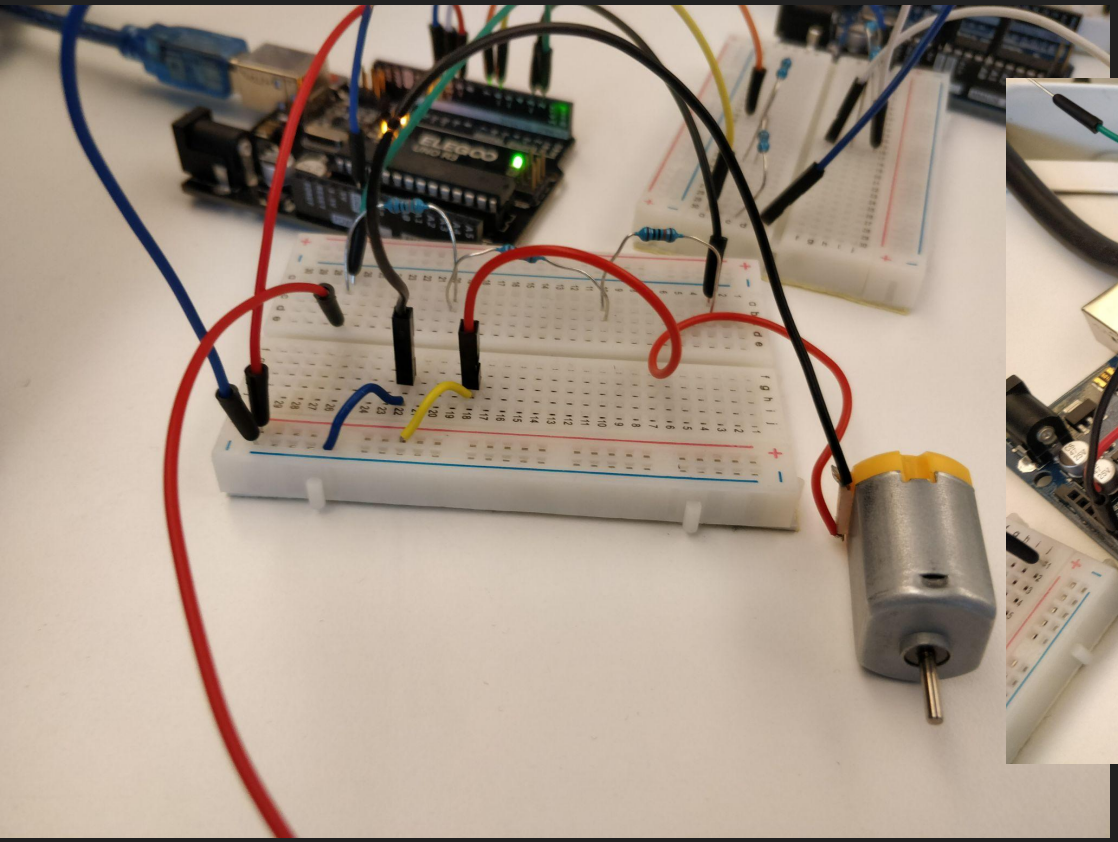

JGA25-370
DC12V620RPM

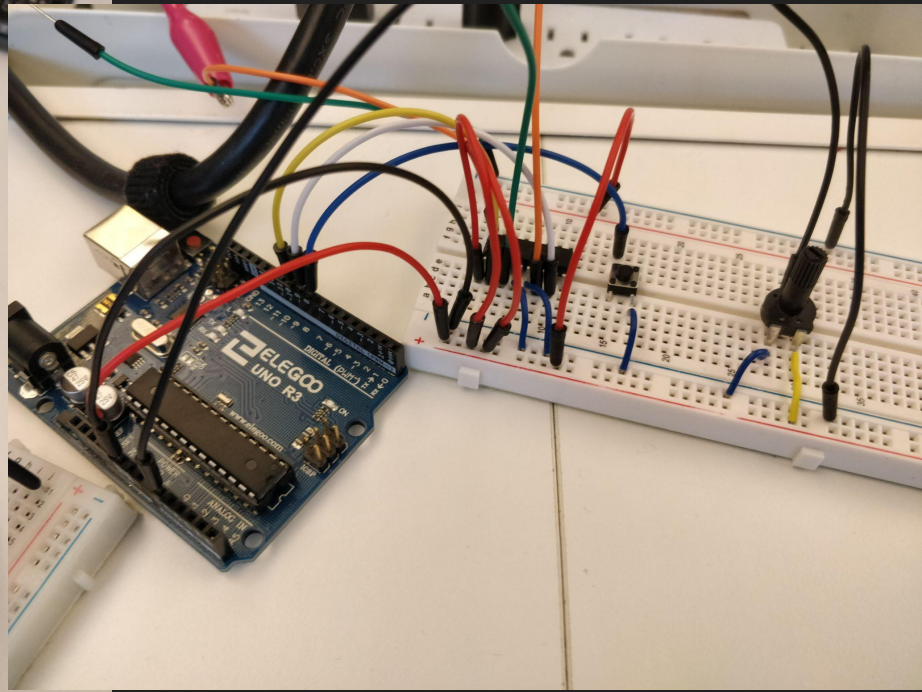

ELEGOO
UNO R3
DIGITAL (PWM~)
ANALOG IN
POWER

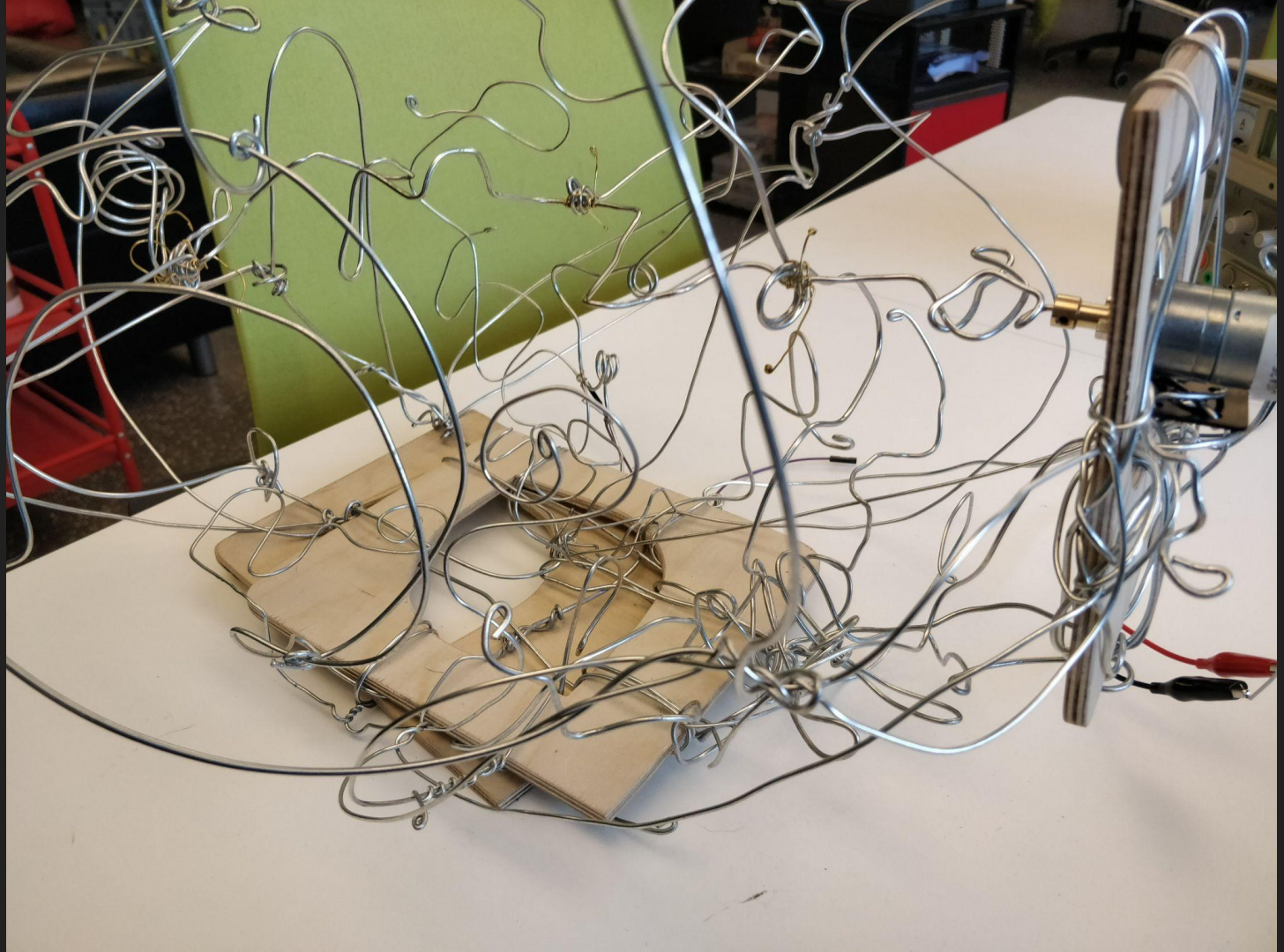

MOTOR
10k
LEGION

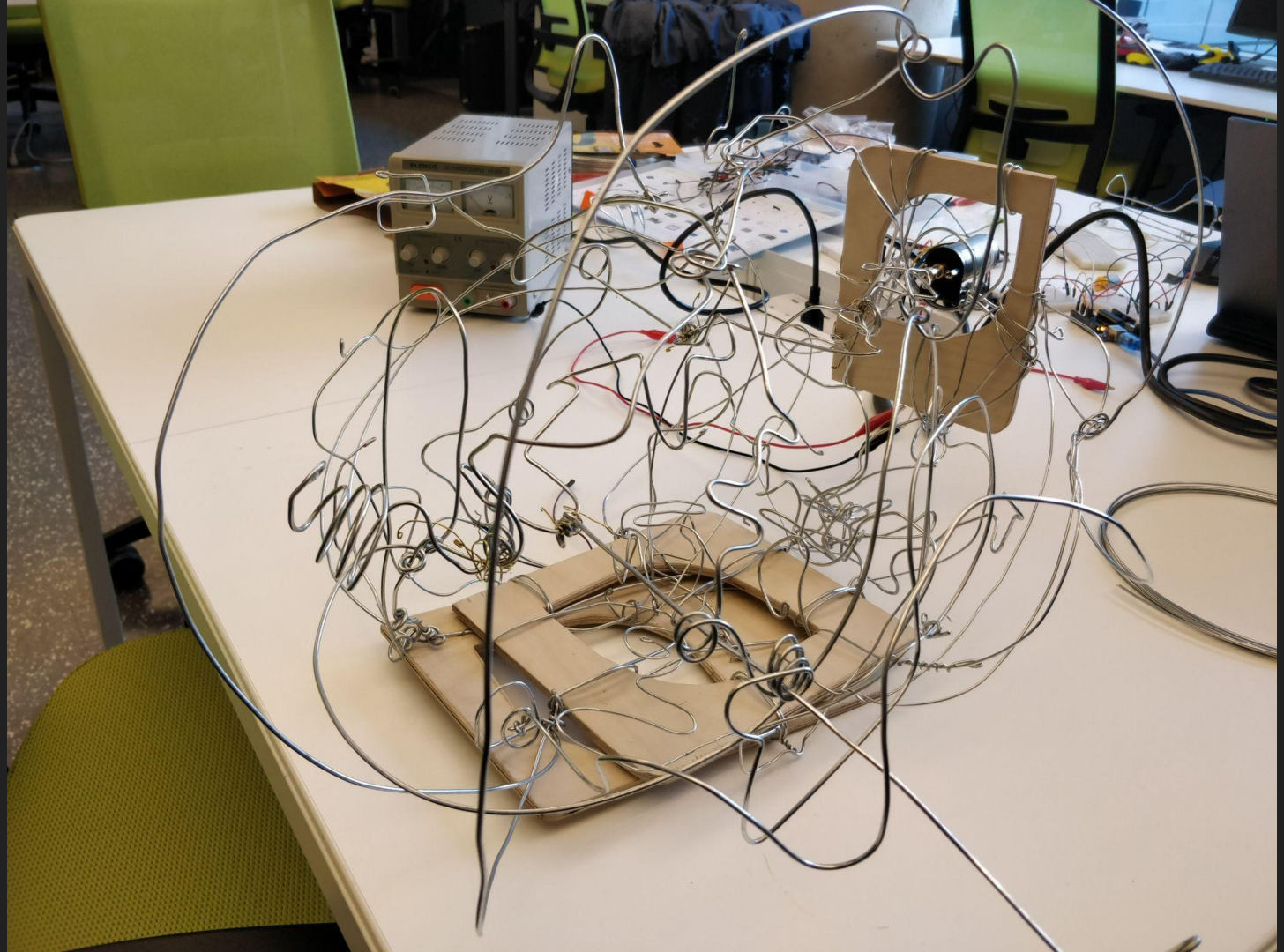

DID YOU BORROW A BOARD GAME?
SSHRC
LIPS INK
ELEGOO.COM

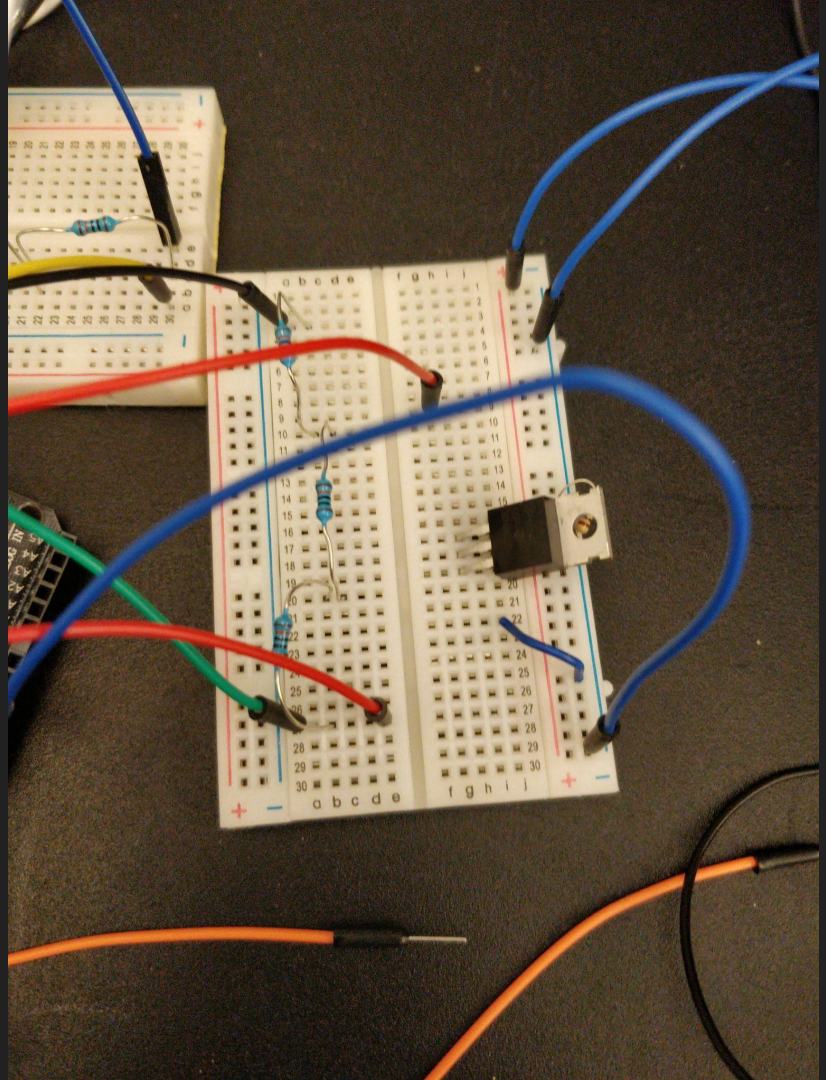

LEGION
Lenovo

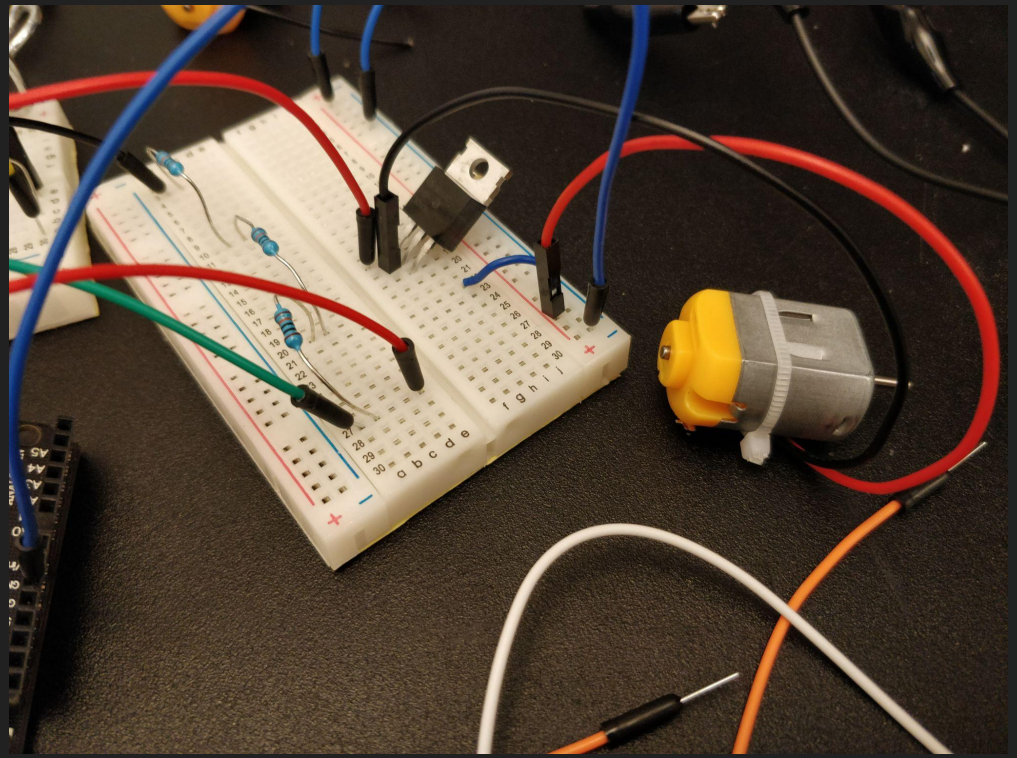

ELEGOO

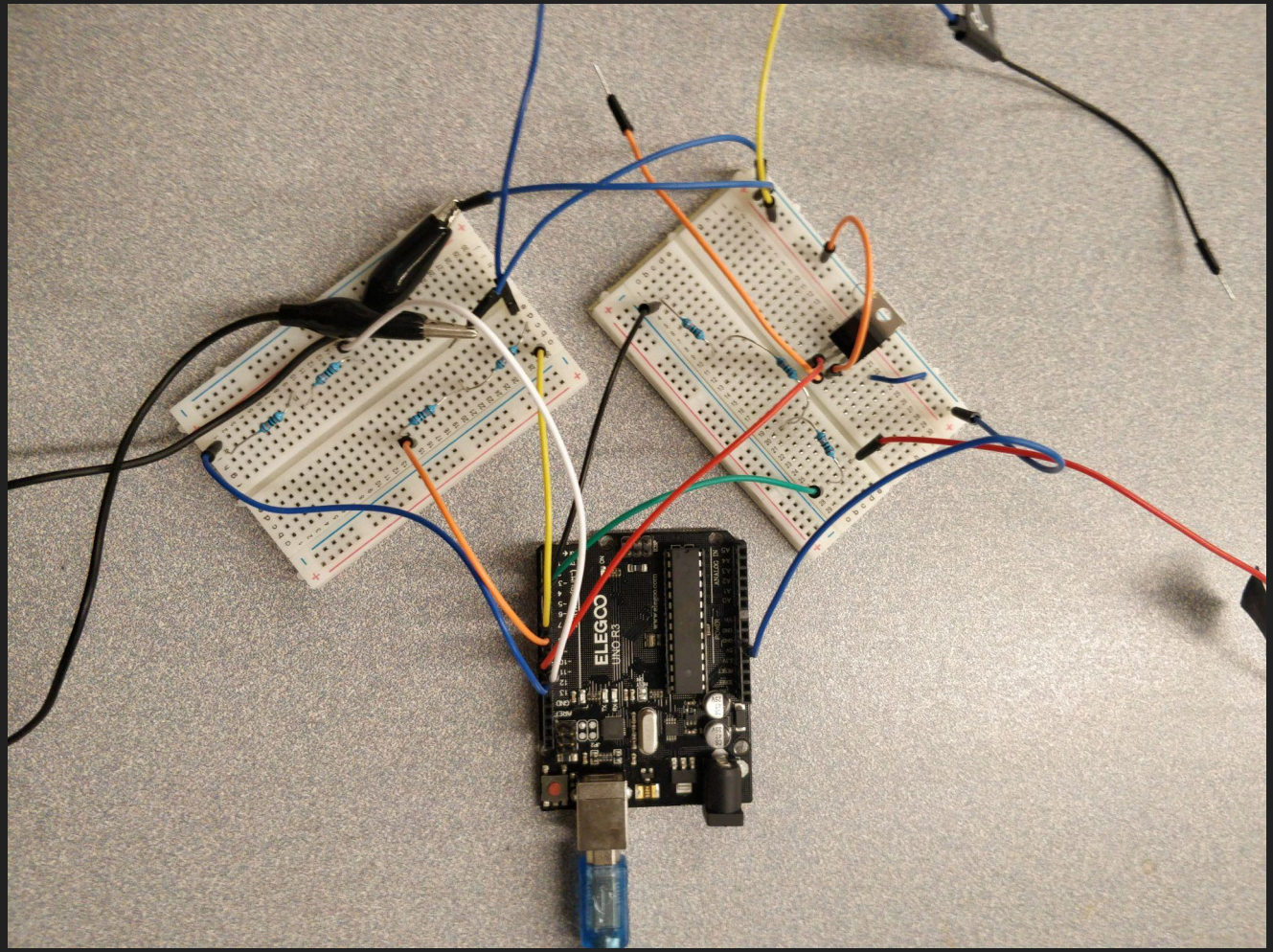

ELEGOO
UNO R3











Circuit

- MOSFET transistor for motor voltage control via external power source.
- 2-3 100k ohm resistors wired in series for each capacitor sensor.

automata_megaStates | Arduino IDE 2.2.1

File Edit Sketch Tools Help

Select Board

automata_megaStates.ino

```ino
#include <CapacitiveSensor.h>

long automataCS;
long bracelet1, bracelet2;
int motorPin = 9;
int braceletStates[3] = {0}; // index position (0 or 1) refers to either bracelet1 or bracelet2

CapacitiveSensor cs_3_2 = CapacitiveSensor(3, 2);  // 10 megohm resistor between pins 4 & 2, pin 2 is sensor pin, add wire, foil
CapacitiveSensor cs_7_6 = CapacitiveSensor(7, 6);
CapacitiveSensor cs_11_10 = CapacitiveSensor(11, 10);

enum {
  A,
  B,
  C,
  D
} state;


void setup() {
  pinMode(motorPin, OUTPUT);
  Serial.begin(9600);
  Serial.println("start");
  cs_3_2.set_CS_Autocal_Millis(0xFFFFFFFF);  // turn off autocalibrate on channel 1 - just as an example Serial.begin(9600);
  cs_7_6.set_CS_Autocal_Millis(0xFFFFFFFF);
  cs_11_10.set_CS_AutocaL_Millis(0xFFFFFFFF);

}

void getBracelet(int b, int idb) {

switch (b) {
    case 30 ... 100:
      if(!idb) braceletStates[0] = A;
      if(idb) braceletStates[1] = A;
      if(idb = 2) braceletStates[2] = A;
      // Serial.println("AA");
      break;

    case 101 ... 200:
      if(!idb) braceletStates[0] = A;
      if(idb) braceletStates[1] = A;
      if(idb = 2) braceletStates[2] = B;
      // Serial.println("BB");
      break;

    case 201 ... 300:
      if(!idb) braceletStates[0] = A;
      if(idb) braceletStates[1] = B;
      if(idb = 2) braceletStates[2] = A;
      // Serial.println("BB");
      break;

    case 301 ... 400:
      if(!idb) braceletStates[0] = B;
      if(idb) braceletStates[1] = A;
      if(idb = 2) braceletStates[2] = A;
      // Serial.println("BB");
      break;

    case 401 ... 6000:
      if(!idb) braceletStates[0] = B;
      if(idb) braceletStates[1] = A;
      if(idb = 2) braceletStates[2] = B;
      //  Serial.println("AB");
      break;

    default:
      if(!idb) braceletStates[0] = B;
      if(idb) braceletStates[1] = B;
      if(idb = 2) braceletStates[2] = B;
      // Serial.println("BA");
      break;
  }

}

void runBState() {

if (automataCS > 30) {
  if (braceletStates[0] == A && braceletStates[1] == A && braceletStates[2] == A) {
      analogWrite(motorPin, 240);  // Low power to motor
        // Serial.println("V1.");

    } else if (braceletStates[0] == A && braceletStates[1] == A && braceletStates[2] == B) {
      analogWrite(motorPin, 245);  // Medium power to motor
        // Serial.println("V2");

    } else if (braceletStates[0] == A && braceletStates[1] == B && braceletStates[2] == A) {
      analogWrite(motorPin, 250);  // Medium power to motor
        // Serial.println("V3");

    } else if (braceletStates[0] == B && braceletStates[1] == A && braceletStates[2] == A) {
      analogWrite(motorPin, 252);  // High power to motor
        // Serial.println("V4");

    } else if (braceletStates[0] == B && braceletStates[1] == A && braceletStates[2] == B) {
      analogWrite(motorPin, 255);  // High power to motor
        // Serial.println("MAX SPEED.");

    } else if (braceletStates[0] == B && braceletStates[1] == B && braceletStates[2] == B) {
      analogWrite(motorPin, 0);  // No power to motor
        // Serial.println("motor  off.");
    }
  }

}

void loop() {

  automataCS = cs_3_2.capacitiveSensor(30);
  bracelet1 = cs_7_6.capacitiveSensor(30);
  bracelet2 = cs_11_10.capacitiveSensor(30);

  getBracelet(bracelet1, 0); // set the state of bacelet1 acording to value thresholds
  getBracelet(bracelet2, 1);// set the state of bacelet2 acording to value thresholds
  getBracelet(automataCS, 2);
  // Serial.println(braceletStates[0]);

    runBState(); // power motor at a set intensity for each state configuration



  Serial.print("autoCS: ");  // tab character for debug window spacing
  Serial.print(automataCS);  // print sensor output 1

 Serial.print("\t");
 Serial.print("b1: ");
 Serial.print(bracelet1);


  Serial.print("\t");
  Serial.print("b2: ");
  Serial.println(bracelet2);

  delay(10);  // arbitrary delay to limit data to serial port

}
```

automata_megaStates | Arduino IDE 2.2.1

File Edit Sketch Tools Help

Select Board

automata_megaStates.ino


```ino
    default:
      if(!idb) braceletStates[0] = B;
      if(idb) braceletStates[1] = B;
      if(idb = 2) braceletStates[2] = B;
      // Serial.println("BA");
      break;
  }

}

void runBState() {

if (automataCS > 30) {
  if (braceletStates[0] == A && braceletStates[1] == A && braceletStates[2] == A) {
      analogWrite(motorPin, 240);  // Low power to motor
        // Serial.println("V1.");

    } else if (braceletStates[0] == A && braceletStates[1] == A && braceletStates[2] == B) {
      analogWrite(motorPin, 245);  // Medium power to motor
        // Serial.println("V2");

    } else if (braceletStates[0] == A && braceletStates[1] == B && braceletStates[2] == A) {
      analogWrite(motorPin, 250);  // Medium power to motor
        // Serial.println("V3");

    } else if (braceletStates[0] == B && braceletStates[1] == A && braceletStates[2] == A) {
      analogWrite(motorPin, 252);  // High power to motor
        // Serial.println("V4");

    } else if (braceletStates[0] == B && braceletStates[1] == A && braceletStates[2] == B) {
      analogWrite(motorPin, 255);  // High power to motor
        // Serial.println("MAX SPEED.");

    } else if (braceletStates[0] == B && braceletStates[1] == B && braceletStates[2] == B) {
      analogWrite(motorPin, 0);  // No power to motor
        // Serial.println("motor  off.");
    }
  }

}

void loop() {

  automataCS = cs_3_2.capacitiveSensor(30);
  bracelet1 = cs_7_6.capacitiveSensor(30);
  bracelet2 = cs_11_10.capacitiveSensor(30);

  getBracelet(bracelet1, 0); // set the state of bacelet1 acording to value thresholds
  getBracelet(bracelet2, 1);// set the state of bacelet2 acording to value thresholds
  getBracelet(automataCS, 2);
  // Serial.println(braceletStates[0]);

    runBState(); // power motor at a set intensity for each state configuration



  Serial.print("autoCS: ");  // tab character for debug window spacing
  Serial.print(automataCS);  // print sensor output 1

 Serial.print("\t");
 Serial.print("b1: ");
 Serial.print(bracelet1);


  Serial.print("\t");
  Serial.print("b2: ");
  Serial.println(bracelet2);

  delay(10);  // arbitrary delay to limit data to serial port

}
```


Ln 15, Col 5 No board selected