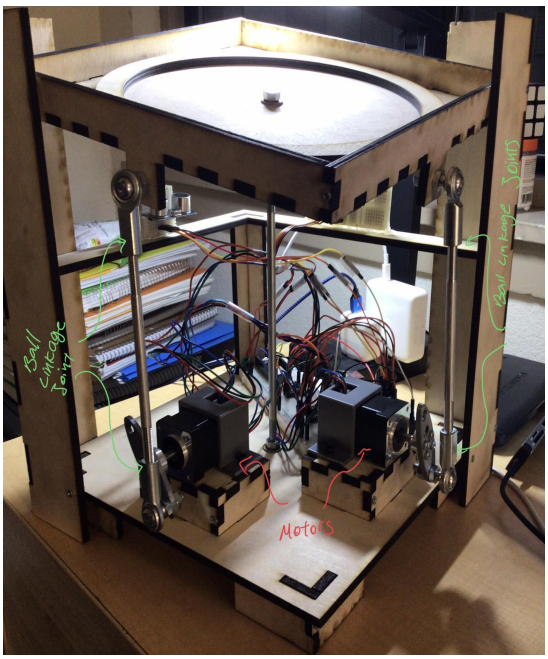
ME 102B Final Report
Circoloco

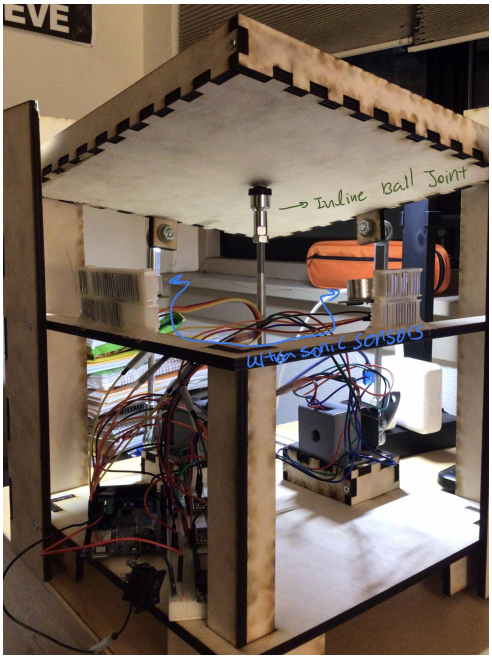
Joel Parks and Sina Shakeraneh

Opportunity: This device is a passion project and was pursued due to it's combination of both technical and creative challenges. It's purpose was to explore the full spectrum of content discussed in 102B as much as possible in a single project and two person team. Real world applications are not readily apparent. However, such applications could include developing this project into kits to be utilized as in classroom labs. Instructors would be able to tailor the project to areas of specific focus within their class. This project could also lay the groundwork for developing more sophisticated projects such as high tech marble runs.

High Level Strategy: The original strategy was to have a ball placed on a table at a random position be able to move fully autonomously (programmed) in a desired pattern. We had hoped to be able to do various patterns and the patterns to be made purely based on table movement. However, due to budget and time constraints we were unable to acquire a resistive touchscreen and had to make some adjustments. We had to limit the patterns down to one and partly obtain the pattern through physical hardware constraints. In the end the ball was able to autonomously move in a circular pattern on the table. This is done with a pair of servo motors and ultrasonic sensors. The motors are connected to the table through ball joint linkages providing table tilt as motors rotate. L brackets constraint theta rotation. The ultrasonic sensors had issues with noise and we had to adjust their position in the final iteration. To move the table precisely with a high amount of torque, two Nema 14 Stepper Motor 40Ncm 35x35x52mm 1.5A were the actuators selected. Accuracy is important since we didn't have a sensor to find the location of the ball at each time step therefore, the movement had to be precise in order to produce a nice circular path for the ball. The motors produced a maximum speed of 600 rpm, which was enough as we originally wanted to run the motors at 300 rpms and we ended up using about 100 rpms with the final iteration. Higher than 100 rpms and the ball would be pushed off track.

Fully assembled photos with labelings:





Function-critical decisions (project elements): The initial strategy of this project was to be able to control the trajectory of a ball that is placed on the top table using two motors. However, since the project was coded in an open loop format there was no feedback on the location of the ball. This resulted in not being able to find the optimal speed for the motors to make the ball rotate in a circular path. In order to constrain the ball to this motion, a circular ring was placed on the table. Since the motors did not give feedback on their positions, we had to use ultrasonic sensors in order to determine the flatness of the table. For the ball to be able to rotate in a circular path, the motors had to be in an offset of 90° from one another. To achieve this criteria, initially the table had to be flattened, using the ultrasonic sensors, then one of the motors would rotate 90° before both motors were running at the same time. The initial location of the ultrasonic sensors were at the bottom plate, however, the acquired data were noisy. Therefore, we had decided to move the sensors closed to the top plate for better value readings. As far as the motors, we chose two Nema 14 stepper motors in order to produce a torque that can withstand the weight of the ball and the plate while being able to run at a reasonable speed. We chose the table to move up and down at the most 1", therefore the hubs that we chose were 2" in diameter. Furthermore, the mass that each motor had to overcome was calculated using the equations below. We had estimated that each motor will be responsible for only 10% of the mass of the table since the pivot point will take care of the rest.

Mass of the table:

$$0.1 * (10 * 10 * 0.5\,in^3) * (\frac{1}{12^3} ft^3/in^3) * (12lb/ft^3) = 0.0833lb = 0.038kg$$

Mass of the ball: $0.250kg$

Mass of the linkages and rods: $0.230kg$

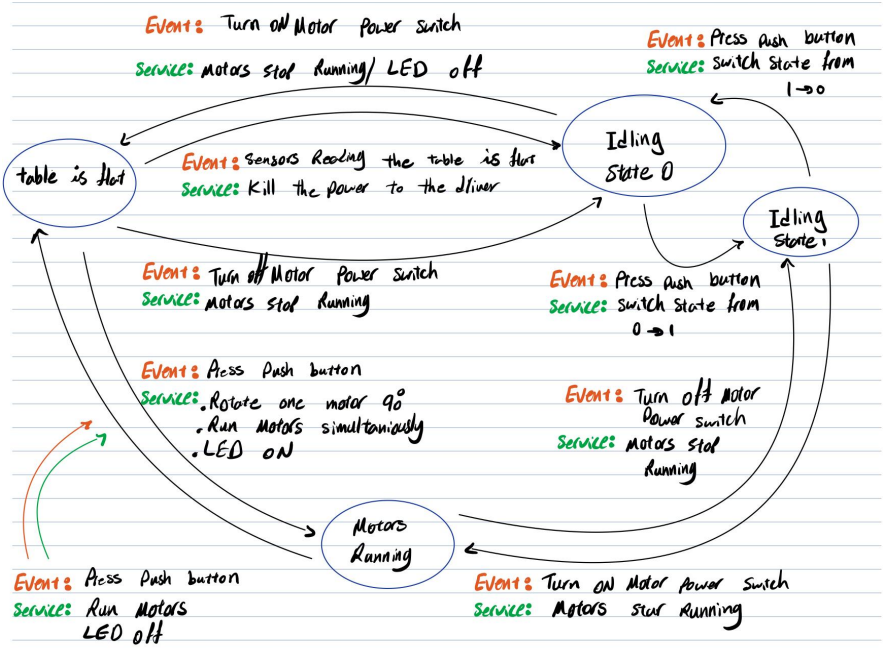
Total Mass = $0.250kg + 0.038kg + 0.230kg = 0.518kg$

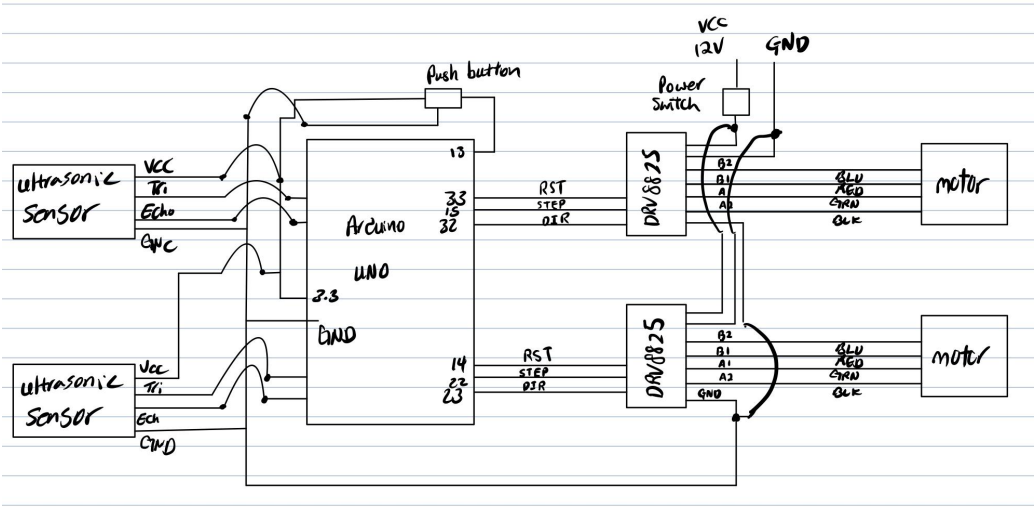
Required torque to move the table was obtained using the following calculations:

$$\tau = F * r = (0.518kg * 9.81m/s^2) * (2.54cm) = 12.9072N.cm$$

The ball joint linkages from McMaster were selected to provide the necessary movement of the system. Ball joint linkages are commonly used for pivot movement purposes and therefore, were an obvious choice for our project. In addition, we went with McMaster as this piece of equipment was critical to our project and needed quality ball joints.

Circuit and State Diagrams:





Future reflection: Overall, this project was a fun way to implement the various concepts learned throughout the semester. It can be used as a basis to create more developed and applicable devices. Creating a division of labor plan where each team member focused on a particular area was critical for our team. It was difficult finding certain types of hardware for the project so it pays to start even earlier than the project deadlines.

Appendix

Bill of Materials:

ITEM	QUANITY	COST (EA)	USE	LINK
Plywood - 1/4" x 12" x 24" (variable)	4	$1.67	Fabrication of hardware structure	ore.jacobshall.org/products/plywood-1-4
Ball Joint Linkage 1/4"-28 Internal Thread, Right-Hand Shank, Right-Hand Ball Stud	4	$8	Joint/Linkage	https://www.mcmaster.com/60645K421/
Connecting Rod Aluminum, 6" Overall Length, 1/4"-28 Thread	2	$7.37	Connector	https://www.mcmaster.com/6516K177/
Zinc-Plated Carbon Steel, 1/4"-28 Internal Thread, 9" Long	1	$12.02	Connector	https://www.mcmaster.com/6516K7
Medium-Strength Steel Hex Nut Grade 5, Zinc-Plated, 1/4"-28 Thread Size	1 (pack)	$6.00	Fastening	https://www.mcmaster.com/95462A505/
DRV8825 Stepper Motor Driver Carrier, High Current	2	$11.95	Motor Driver	https://www.pololu.com/product/2133
Nema 14 Bipolar 1.8deg 40Ncm (56.7oz.in) 1.5A 4.2V 35x35x52mm 4 Wires	2	$18.53	Motor	pper-motor/nema-14-bipolar-1-8deg-40r
Pololu Universal Aluminum Mounting Hub for 5mm Shaft, M3 Holes (2-Pack)	1	$7.49	Mounting hub for motor	ps://www.pololu.com/product/1998/sp
Grey PLA 3D printer material (sent to friend to print)	NA	NA	Joint and housing for motor	NA
Internally Threaded Inline Ball Joint Linkages	1	$6.35	Acting as the pivot point	https://www.mcmaster.com/8412K44/
Ball	1	$9.95	Ball for the manouver	me-steel-ball-bearings/products/1-1-2-in
HC-SR04 Ultrasonic Sonar Distance Sensor (microkit)	2	NA	Height of table measurements	s://microkit.berkeley.edu/ultrasonic-sen
4 x 3/4 wood screws	15	$0.12	fastening	Ace Hardware
1/4 washer	5	$0.16	fastening	Ace Hardware
8-32 Hex Nut	4	$0.16	fastening	Ace Hardware
8-32x2 screw	4	$0.55	fastening	Ace Hardware
Breadboard (microkit)	3	NA	Electrical connection	NA

3D CAD Model:

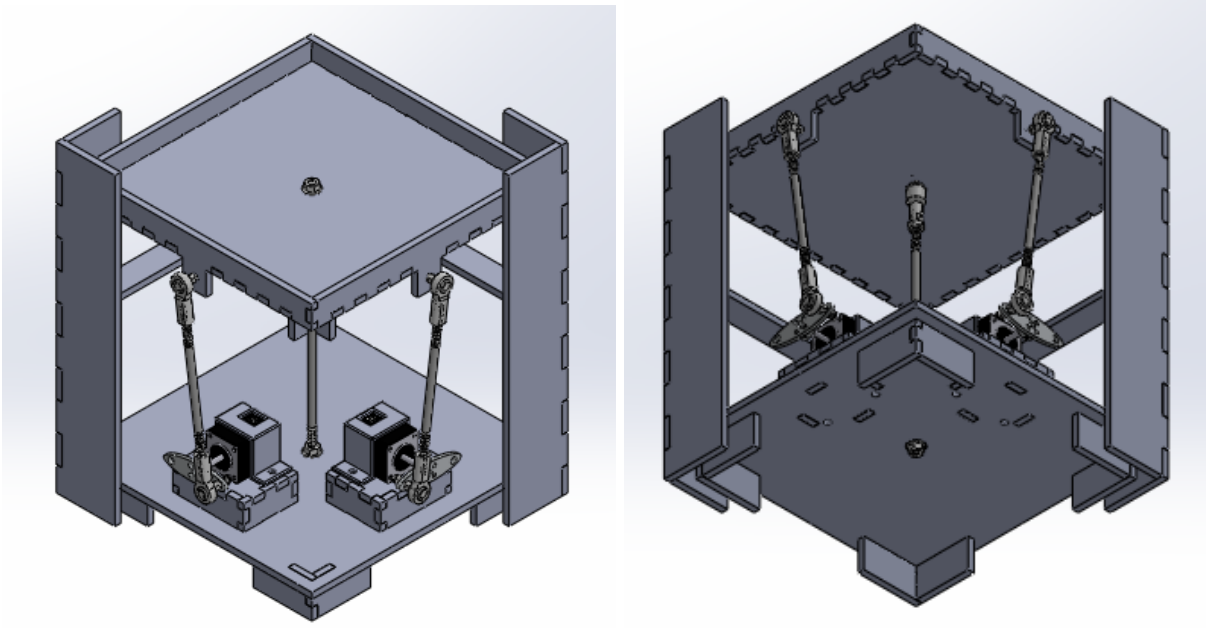


Image : Isometric view of CAD model from two different perspectives

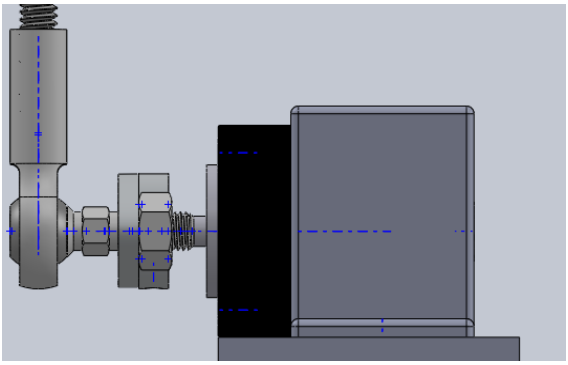


Image: close up view of motor, motor housing, motor hub/joint, and ball joint linkage

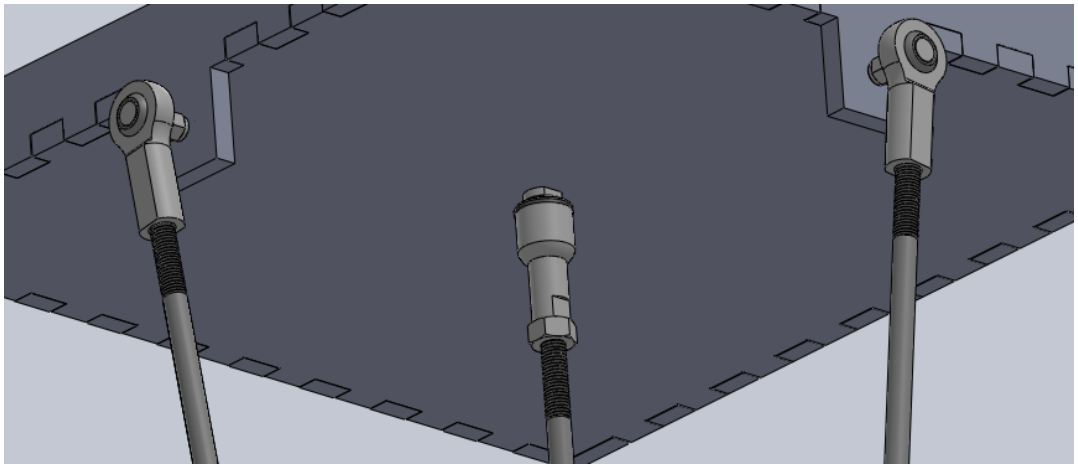


Image: close up of shaft/ball joint linkage connector to top table portion of the system

CAD of 3D printed components:

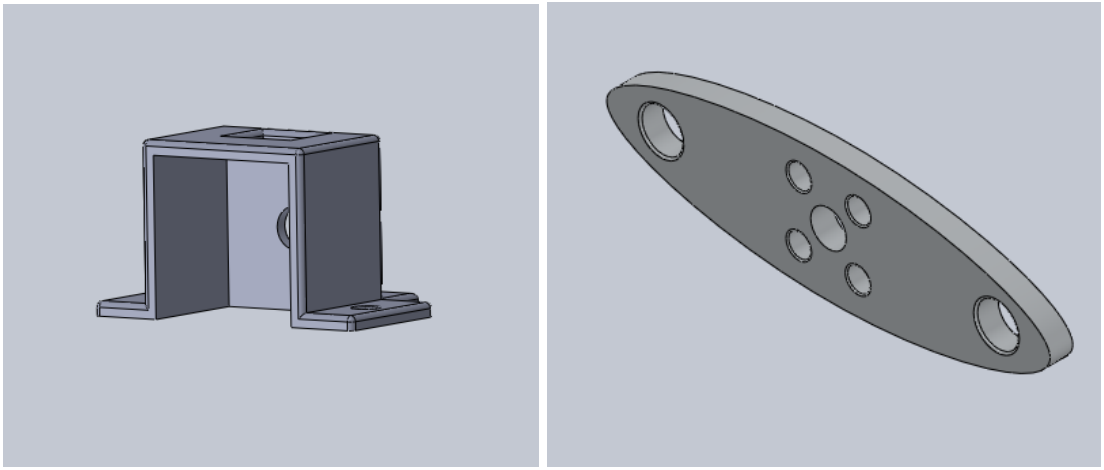
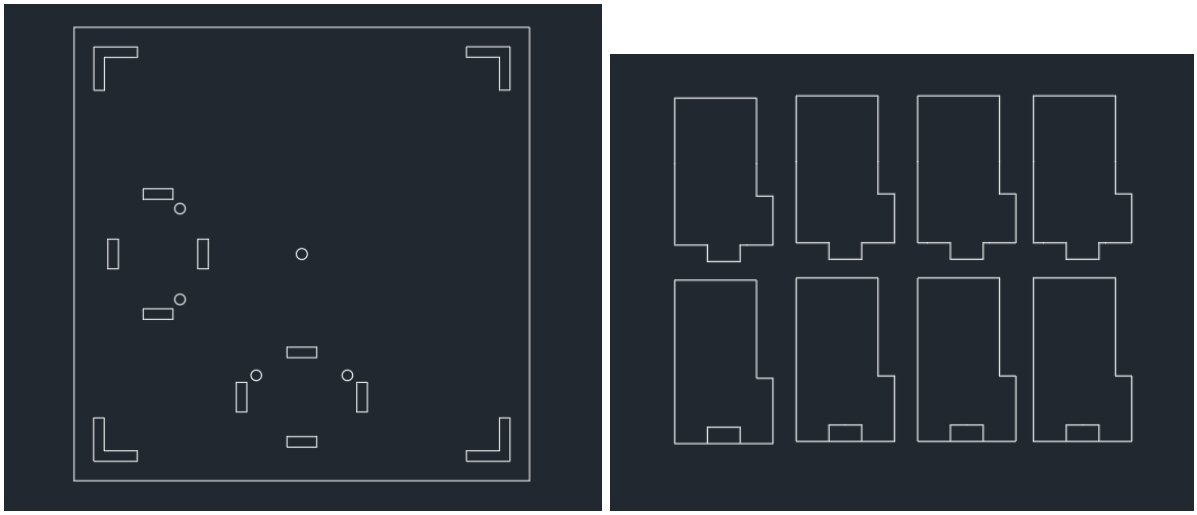
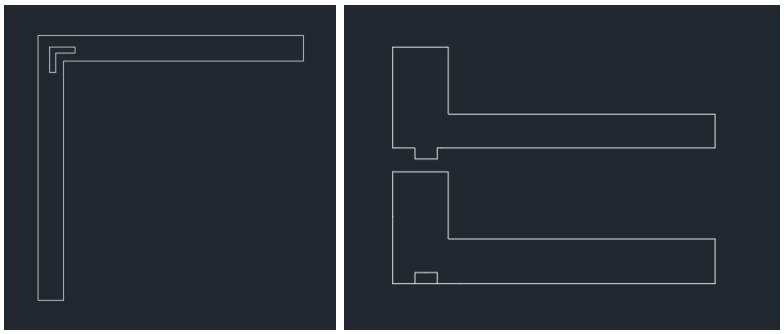


Image: CAD models of motor joint and housing for 3D printing

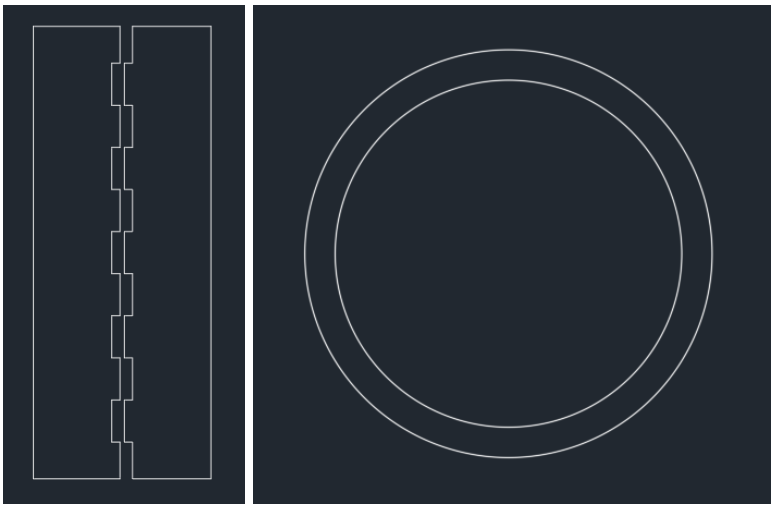
AutoCAD of Laser cut components:



AutoCAD drawing of bottom stand and it's legs



AutoCAD drawing of Ultrasonic stand and legs



AutoCAD drawing of L bracket and circular constraint

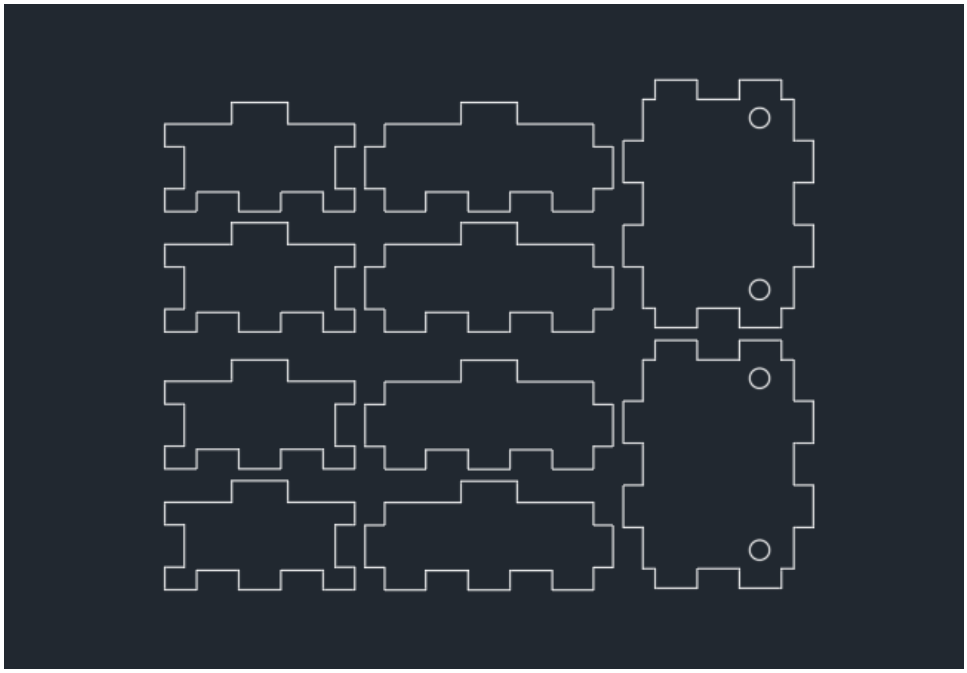
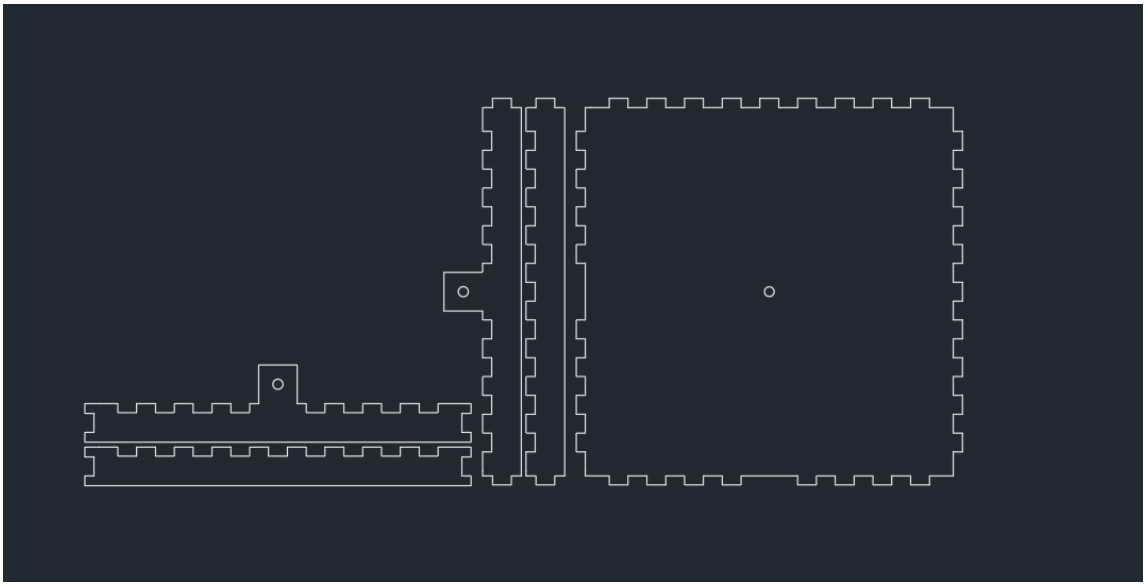


Image: AutoCAD drawing of Motor housings



AutoCAD drawing of top stand

Summary:

- Bottom stand, legs for stand, motor stands, circular constraint, L shape brackets, and top stand was laser cut using Jacobs Hall's Plywood - 1/4" x 12" x 24" and their laser cutters
- Motor housing and joints were 3D printed

Implemented Code:

```
#include <AccelStepper.h>
#include <MultiStepper.h>
#include <Arduino.h>

#define motorInterfaceType 1
int counter = 0;
int counter1 = 0;



MultiStepper steppers;
//motor 1
int x_slp = 4;
int x_step = 3;
int x_dir = 2;

//motor2

int y_slp = 7;
int y_step = 6;
int y_dir = 5;


////////////////////////////////////////////////////////////////////////////
//Ultrasonic sensors
int TRI1_PIN = 8;
int ECH1_PIN = 9;
int TRI2_PIN = 10;
int ECH2_PIN = 11;

long duration1;
long duration2;

float distance1 = 0;
float distance2 = 0;

bool stateX = 1;
```

```
bool stateY = 1;

int runOnce = 0;


//////////////////////////////////////////////////////////////////////////////
// Create a new instance of the AccelStepper class:
AccelStepper x = AccelStepper(motorInterfaceType, x_step, x_dir);
AccelStepper y = AccelStepper(motorInterfaceType, y_step, y_dir);


//////////////////////////////////////////////////////////////////////////////
//LED
const int ledPin = 13;       // the pin that the LED is attached to


//////////////////////////////////////////////////////////////////////////////
//Button scheme
#define  buttonPin  21    // the pin that the pushbutton is attached to
int buttonPushCounter = 0;   // counter for the number of button presses
volatile bool buttonIsPressed = false;
byte state = 0;
int buttonState = 0;         // current state of the button
int lastButtonState = 0;     // previous state of the button


int prevButtonState = 1;
long debounceDelay = 180;
long prevDebounceTime = 0;
long currentTime;


int leveled = 1;

void isr() {  // the function to be called when interrupt is triggered
  debounce();
  delay(200);
}



void debounce()
{
  currentTime = millis();
  if ((currentTime - prevDebounceTime) > debounceDelay)
  {
    Serial.println("Change state");
    buttonIsPressed = !buttonIsPressed;
    prevDebounceTime = millis();
    prevButtonState = buttonIsPressed;
  }
  //  else
  //  {
  //    buttonIsPressed = prevButtonState;
  //  }
}




void setup() {
  delay(2000);    //delay so that motors dont start immediately
  Serial.begin(9600);
  //Button
  pinMode(buttonPin, INPUT);  // configures the specified pin to behave either as an input or an output
  attachInterrupt(digitalPinToInterrupt(buttonPin), isr, RISING);


  //LED
  pinMode(ledPin, OUTPUT);
  digitalWrite(ledPin, LOW);

  //Ultrasonic Sensor readings
  pinMode(TRI1_PIN, OUTPUT);
```

```
  pinMode(TRI2_PIN, OUTPUT);
  pinMode(ECH1_PIN, INPUT);
  pinMode(ECH2_PIN, INPUT);


  // Configure each stepper
  //  x.setMaxSpeed(700);
  //  y.setMaxSpeed(700);
  x.setMaxSpeed(400);
  y.setMaxSpeed(400);
  x.setAcceleration(100);
  y.setAcceleration(100);


  // Then give them to MultiStepper to manage
  steppers.addStepper(x);
  steppers.addStepper(y);


  pinMode(x_slp, OUTPUT);
  pinMode(y_slp, OUTPUT);
  digitalWrite(y_slp, HIGH);
  digitalWrite(x_slp, HIGH);


  x.setCurrentPosition(0);
  y.setCurrentPosition(0);

}



////////////////////////////////////////////////////////////////////////////
void loop() {
  Serial.println(buttonIsPressed);
```

```
  switch (buttonIsPressed)
  {
    case 1: //run the motors
      digitalWrite(x_slp, HIGH);
      digitalWrite(y_slp, HIGH);
      digitalWrite(ledPin, HIGH);
      startMotorResponse();
      break;
    case 0: // level the motors
      digitalWrite(ledPin, LOW);
      Leveling();
      break;
  }
}



//Event Service Responses
void Leveling() {
  stateX = 1;
  stateY = 1;
  if (runOnce == 0)
  {
    while ((stateX) | (stateY))
    {
      digitalWrite(TRI1_PIN, LOW);
      delayMicroseconds(2);
      digitalWrite(TRI1_PIN, HIGH);
      delayMicroseconds(10);
      digitalWrite(TRI1_PIN, LOW);
      duration1 = pulseIn(ECH1_PIN, HIGH);
      distance1 = duration1 * 0.034 / 2;
      Serial.print("Distance 1 : ");
      Serial.println(distance1);
```

```
      delay(500);

      if ((distance1 >= 5.80) | (distance1 <= 5.60) & stateX == 1)
      {
        x.setCurrentPosition(0);
        x.moveTo(5);
        x.runToPosition();
      }
      else if ((distance1 <= 5.80) & (distance1 >= 5.60))
      {
        stateX = 0;
        Serial.println("Distance 1 Leveled");
      }
      digitalWrite(TRI2_PIN, LOW);
      delayMicroseconds(2);
      digitalWrite(TRI2_PIN, HIGH);
      delayMicroseconds(10);
      digitalWrite(TRI2_PIN, LOW);
      duration2 = pulseIn(ECH2_PIN, HIGH);
      distance2 = duration2 * 0.034 / 2;
      Serial.print("Distance 2 : ");
      Serial.println(distance2);
      delay(500);

      if ((distance2 >= 6.10) | (distance2 <= 5.90) & stateY == 1)
      {
        y.setCurrentPosition(0);
        y.moveTo(5);
        y.runToPosition();

      }
      else if ((distance2 <= 6.10) & (distance2 >= 5.90))
      {
        stateY = 0;
        Serial.println("Distance 2 Leveled");
      }
```

```
       Serial.print("Button State: ");
       Serial.println(buttonIsPressed);
     }
     runOnce = 1;
     leveled = 1;
   }
   else
   {
     digitalWrite(x_slp, LOW);
     digitalWrite(y_slp, LOW);
   }
}


void startMotorResponse() {
  // This is to run the motor
  //     90 degree offset
  if (leveled)
  {
    x.moveTo(200);
    x.runToPosition();
  }

  long positions[2]; // Array of desired stepper positions
  counter =  300;
  counter1 = 300;
  x.setCurrentPosition(0);
  y.setCurrentPosition(0);


=

  positions[0] = counter;
  positions[1] = counter1;
  steppers.moveTo(positions);
  steppers.runSpeedToPosition(); // Blocks until all are in position
  runOnce = 0;

  leveled = 0;

}
```