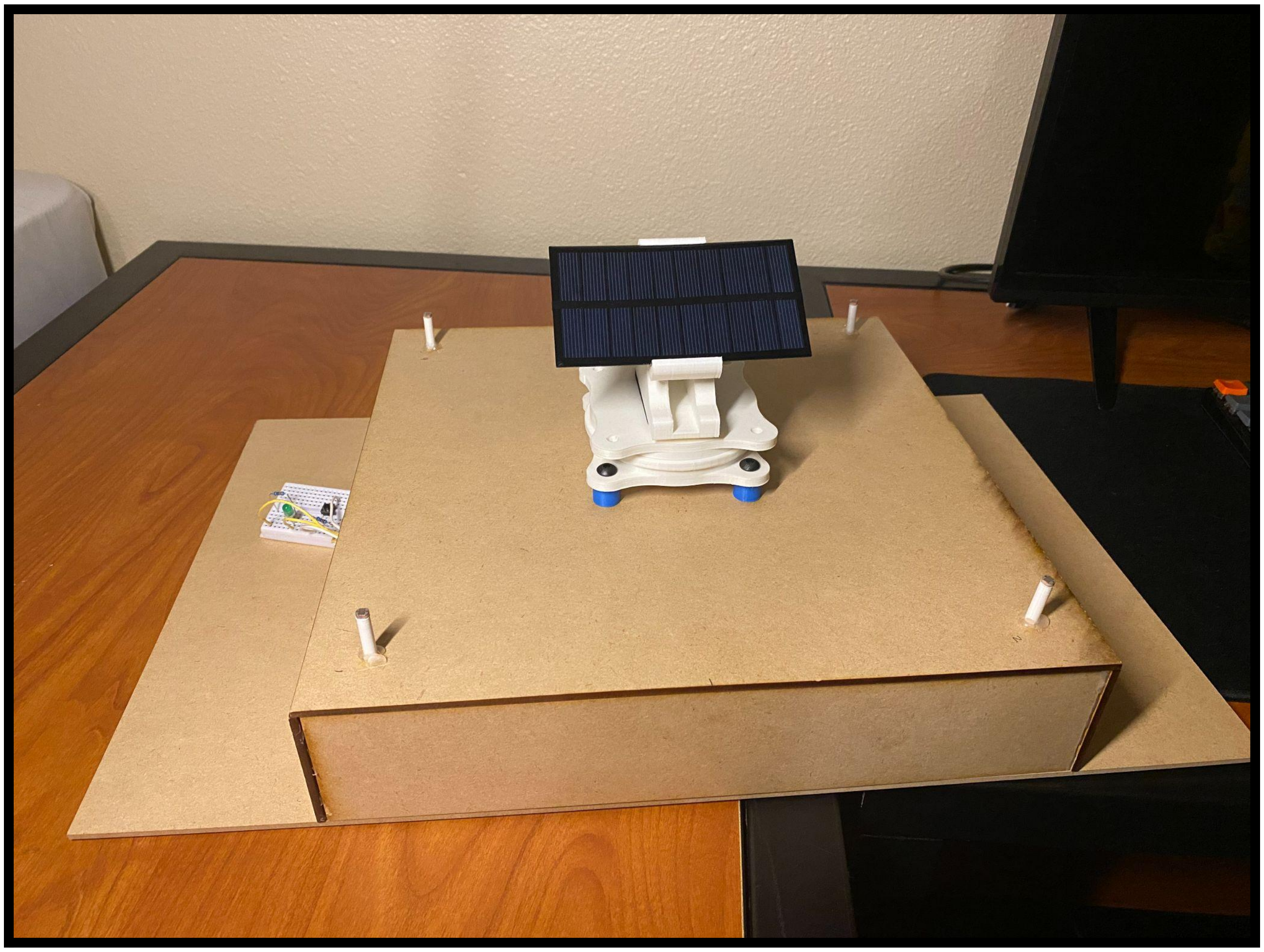
Solar Panel Rotator

Ian Seremet

12/10/2023

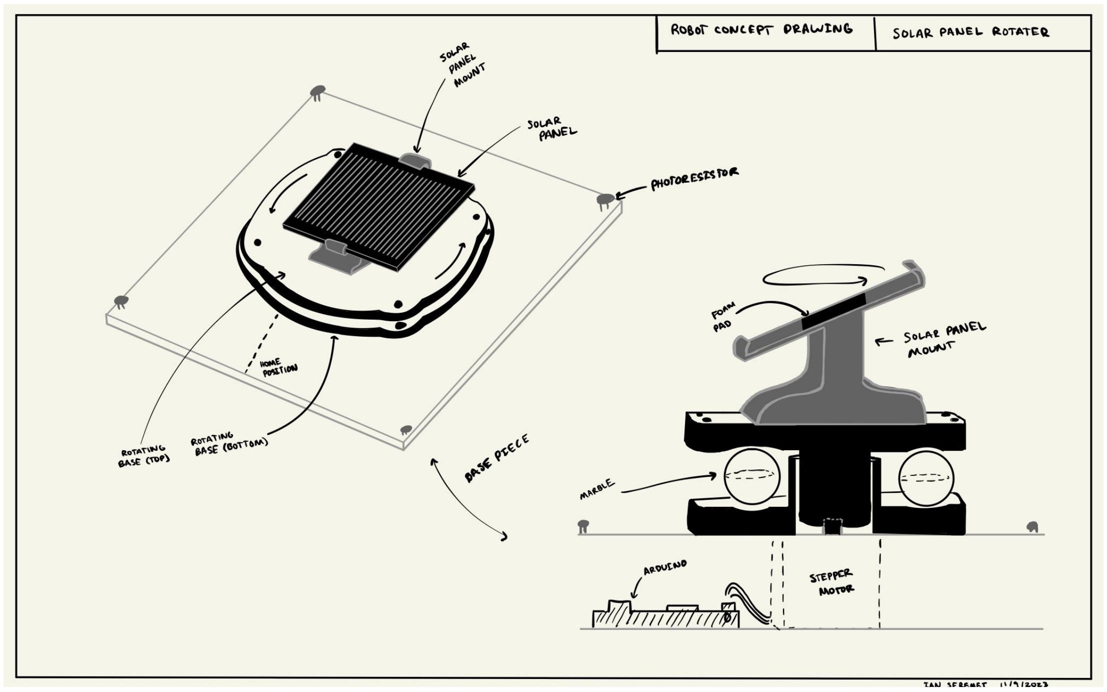
“A robot that rotates a solar panel about a central axis to maximize solar output energy”



“This guy took my invention and just made it way cooler.”
-Charles Fritts (inventor of the solar panel)

Solar Panel Rotator

Ian Seremet
12/10/2023



The robot comprises three main subsystems: a solar panel mount, a bearing piece, and a base. The solar panel mount maintains a fixed 38-degree angle for the solar panel. A bearing piece minimizes friction between the stepper motor and the solar panel mount. The base houses essential components, including the motor, wiring, Arduino, battery, and mounted photoresistors.

The solar panel's design includes a restraining lip that allows sliding motion parallel to its length while preventing perpendicular removal. Foam padding enhances friction to prevent unintended sliding. In the bearing mount, marbles and a marble track reduce friction between two plates through compression fits. An adapter connects the motor to the top bearing plate, which adapts to the solar panel mount, enabling synchronized movement.

Control involves reading photoresistor inputs using voltage dividers, resulting in four circuits. Arduino computes average luminosity by continuously summing these readings. A "center of luminosity" function treats the base as a Cartesian plane, finding a balance point using luminosity as weights. Trigonometry calculates the angle relative to the nearest axis. The quadrant identifies the point's position by examining coordinate values, assisting in determining the total angle by adding 90 degrees for each preceding quadrant.

Robot in CAD
Ian Seremet
11/19/2023

Mechanical System Overview:

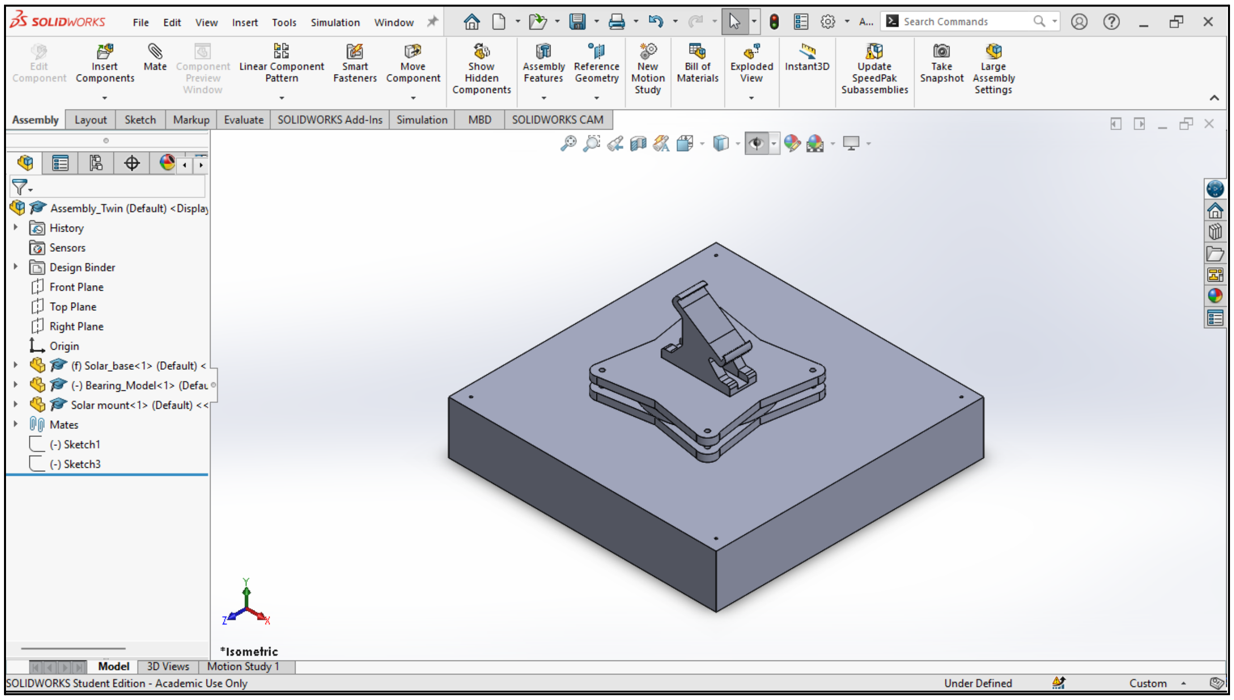


Figure 1: The image above depicts the current robot design. Looking at the tree on the left side of the screen, there are three main parts that make up the robot. Shown in the diagram: solar base, solar mount and the bearing are all shown. The solar bearing sits with one of its faces coincident with the surface of the solar base and will be rigidly fixed with M5 bolts. However, the top of the solar bearing will be free to move by the rotation of a servo motor. The solar mount will be attached to the top plate of the bearing. .

A Deeper Look:

The bottom of the robot includes a cylindrical cutout that adapts with very accurate tolerances to the servo motor. This component is called the servo cap and will connect with the top bearing plate. This will act as an insert into the slot of the bearing as to transfer force from the servo motor to the top piece of the bearing mechanism.

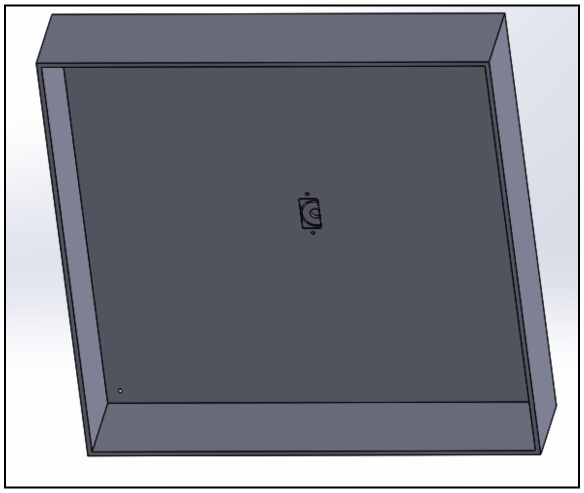


Figure 2: The servo cap sits off-center from the square cutout in the solar base. However, it may be connect to the servo motor to transfer rotational energy to the top plate of the bearing.

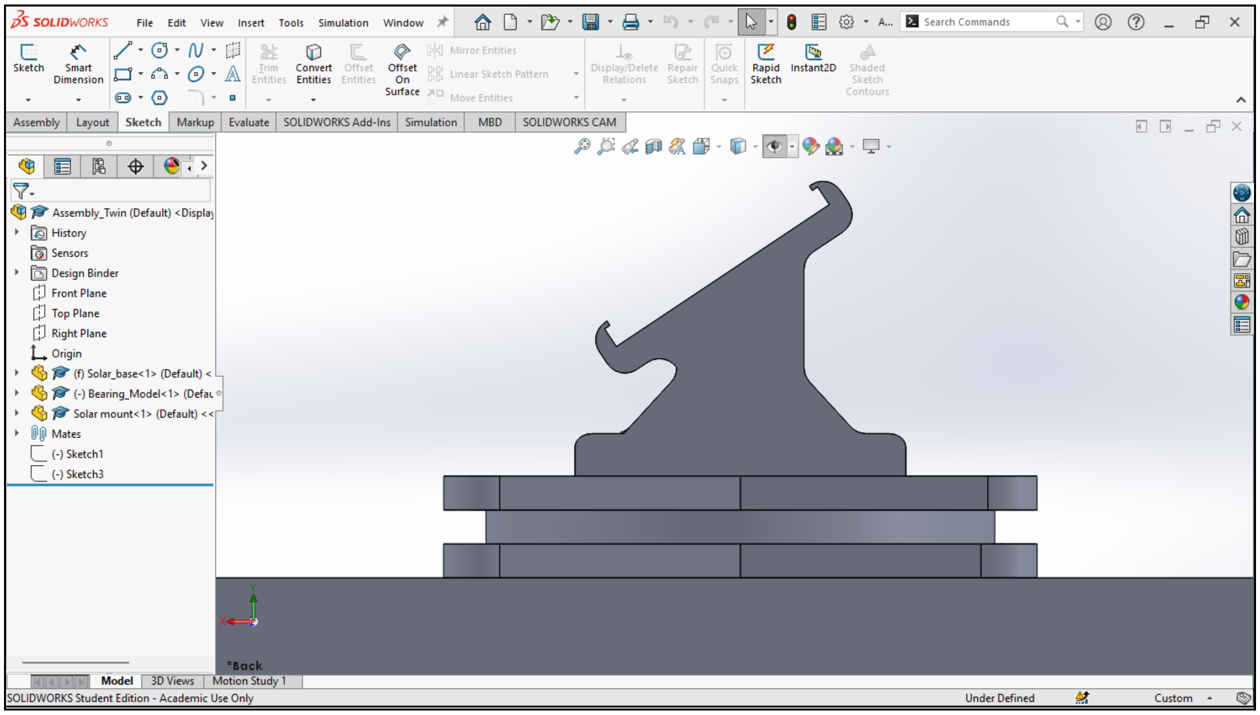


Figure 3: The solar mount sits on top of the top plate of the solar bearing. The bearing is separated into a top and bottom plate, by an intermediate bearing ring. Rotation is permitted by marbles on the inside.

In order for the servo cap to transfer its rotational energy, there must be a considerable amount of friction. After viewing this design in CAD, it has influenced my design choice of using a TPU filament in order to offer high friction with the PLA surface of the solar bearing.

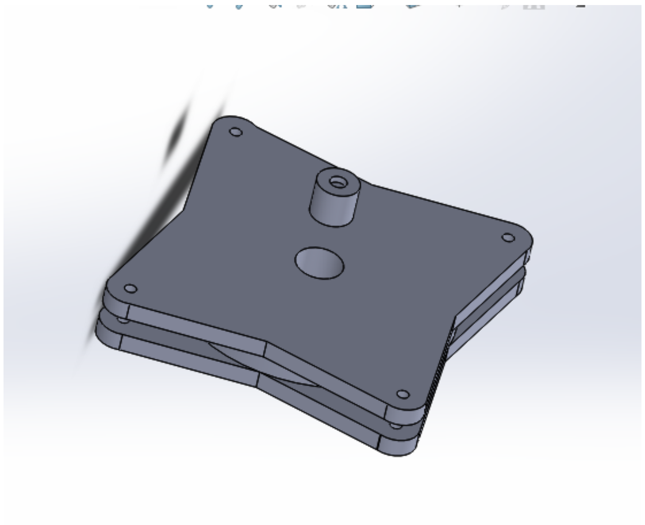


Figure 3: An exploded view of the bearing with the servo cap on the outside. The servo cap will adapt with the servo and couple the servo to the top plate of the bearing. The servo cap will be made from TPU.

Manufacturing decisions (DFM):

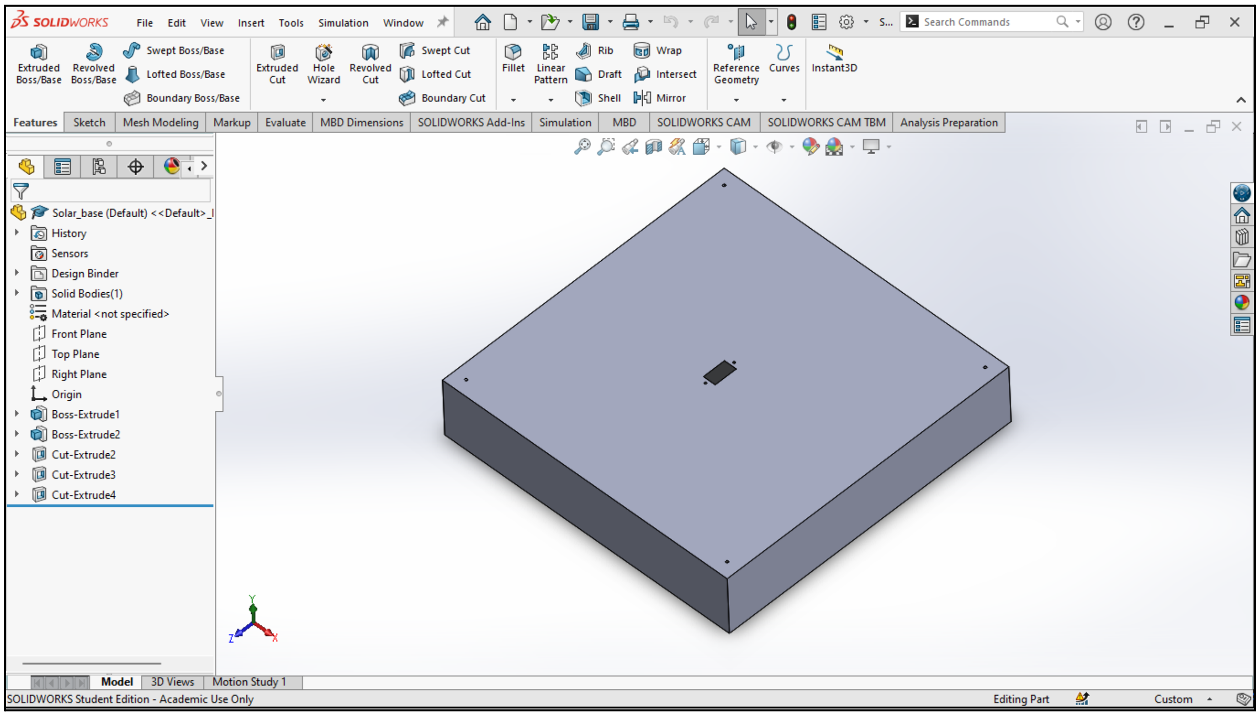


Figure 4: A screenshot of the solar base by itself. It has a square cutout and two holes for mounting the servo motor. In the corners there are 4 holes for mounting the photoresistors.

The solar base will be completely manufactured out of MDF. This will help to ensure accurate placement of cutouts for the servo motor and the photoresistor slots. The solar base can be reduced into .DXF files for laser printing. This is optimal for centering the servo motor, and making the spacing between the photoresistors fair

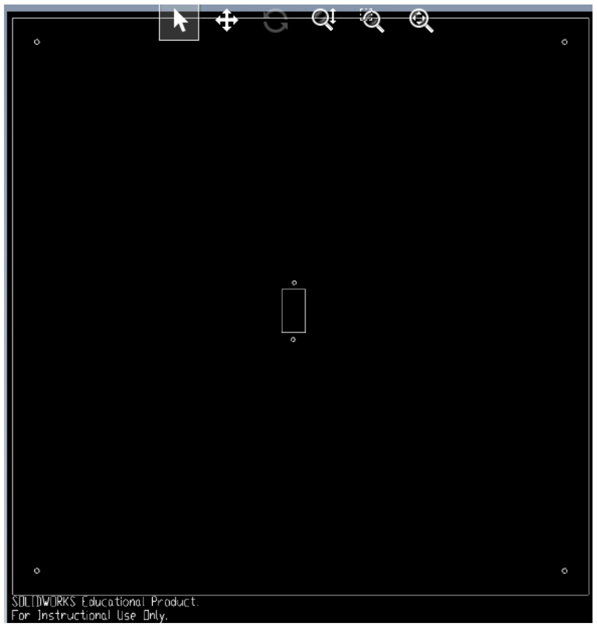


Figure 5: CAD drawing file for the top of the solar base. The white regions depict the cutouts.

The bearing could be easily printed if it was broken into 3 smaller pieces. A bottom plate bearing ring and a top plate would be able to be 3D printed easily as there were no overhanging structures. This design choice would help to ensure a high quality part.

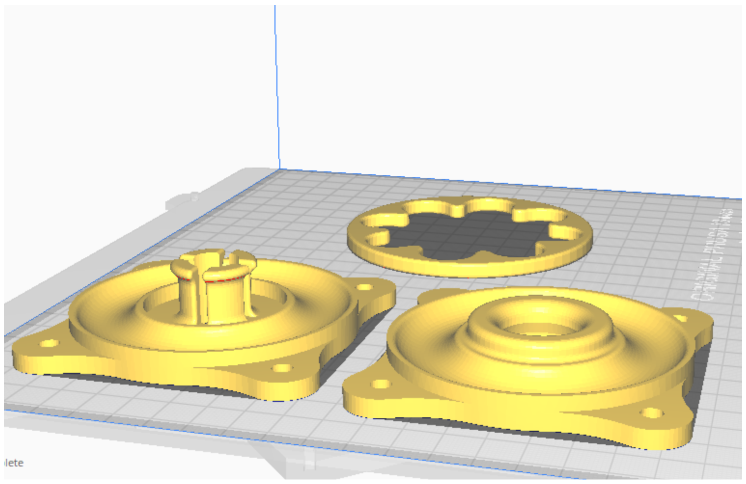
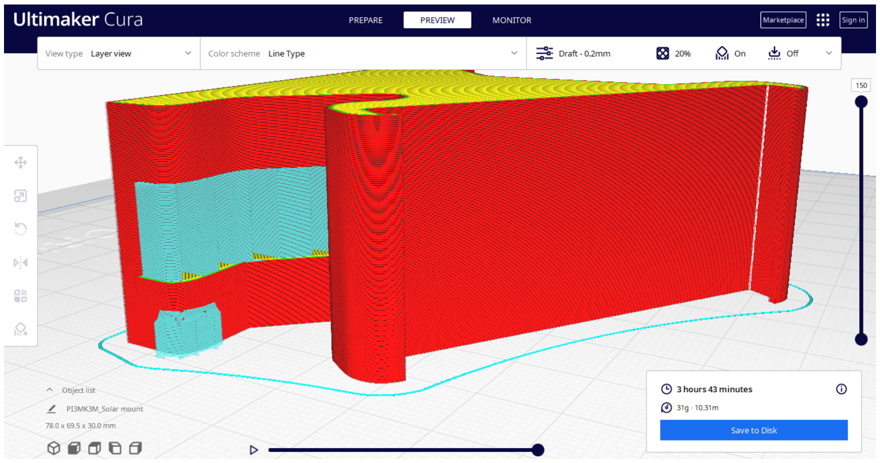


Figure 6: The 3 components for the middle bearing. The U-shape allows for the marbles to travel on a track. The servo cap will adapt to the component on the left of the image.

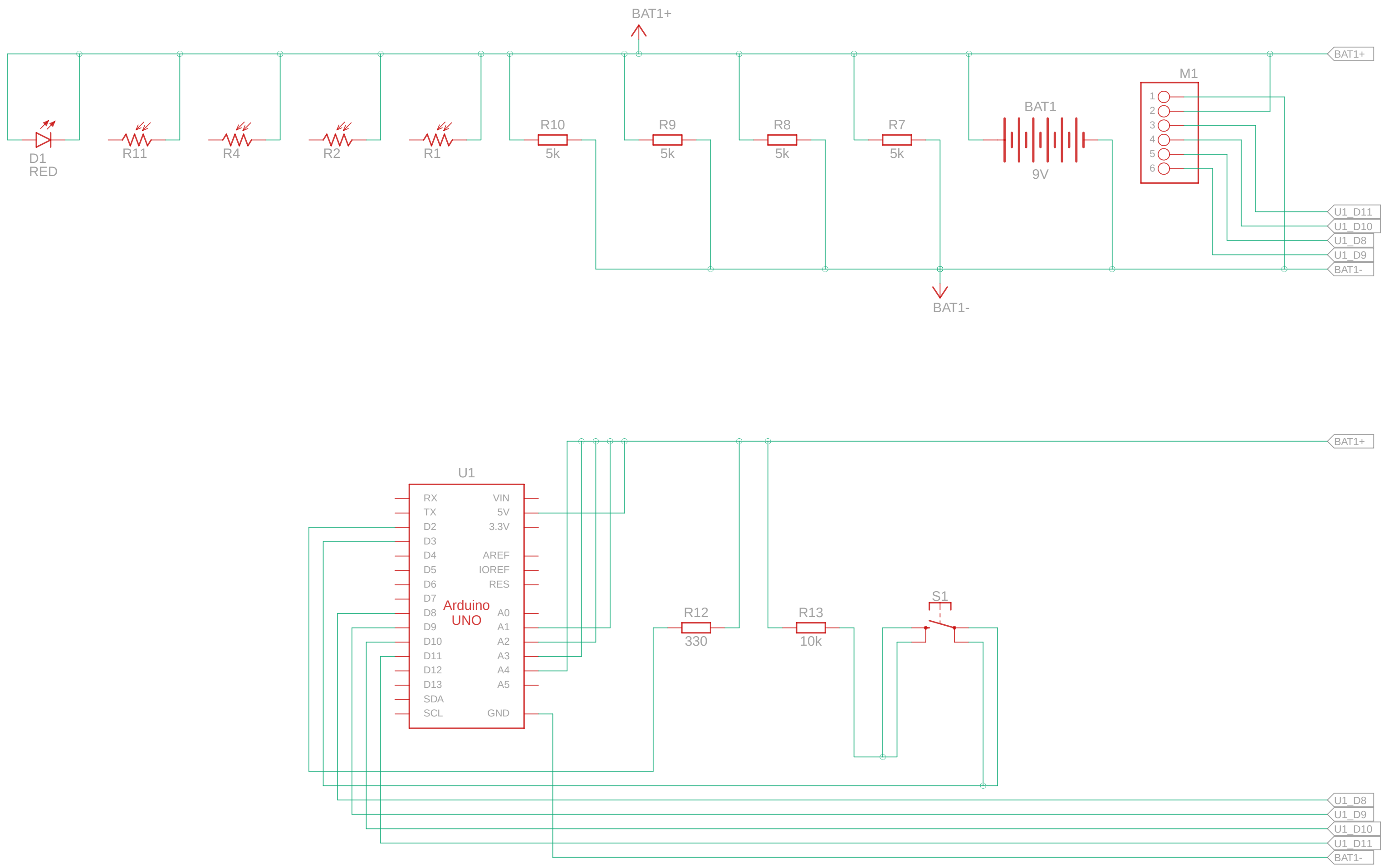
The 3D printer part is built layer by layer from the build plate, so I knew the orientation for the solar mount would be best printed on its side. Then I took into consideration support that would have to exist in the empty region if the mount was oriented on its side. However, I knew when sliced into g-code the supports would be easily removable as they would be fringed and connected. One tug with pliers would ensure the supports were fully removed.



View type Layer view
Color scheme Line Type
Draft - 0.2mm
20%
On
Off
PI3MK3M_Solar mount
78.0 x 69.5 x 30.0 mm
3 hours 43 minutes
31g · 10.31m

Figure 7: Shows the support pattern and orientation for the solar mount.

Part #	Description	Quantity	Source	Subtotal Cost
1	DAOKI Mini Tactile Push Button Switch	1	ME2011 kit	NC
2	22 AWG Copper Solid-Core Wire (25')	1	ME2011 kit	NC
3	Green LED	1	ME2011 kit	NC
4	PLA Marble Bearing Risers	4	Amazon	0.23
5	HiLetgo Photoresistor (3k-8k Ohm Range)	4	Amazon	0.30
6	PLA Marble Bearing Base Piece	1	Amazon, Thingiverse	0.56
7	PLA Marble Bearing Top Piece	1	Amazon, Thingiverse	0.32
8	PLA Marble Bearing Ring	1	Amazon, Thingiverse	0.53
9	Party Marbles	8	Hobby Lobby	1.49
10	Solderless Prototyping Breadboard (Mega)	1	ME2011 kit	NC
11	Solderless Prototyping Breadboard (Mini)	1	ME2011 kit	NC
12	ULN2003 Stepper Motor Driver	1	ME2011 kit	NC
13	FellDen Micro Solar Panels (200mA, 5V)	1	Amazon	1.05
14	M5 Bolt BLKOXIDE Hex PHS (50mm length)	4	Amazon	4.19
15	M5 Brass Threader Nut (10 mm length)	4	Amazon	3.30
16	Double Sided Adhesive Pads 3M	2	Amazon	1.20
17	Medium Density Fiberboard (12" X 24" x 1/8")	2	HSEC Makerspace	22.54
18	Resistor (5k Ohm)	4	ME2011 kit	NC
19	Resistor (330 Ohm)	1	ME2011 kit	NC
20	Resistor (10k Ohm)	1	ME2011 kit	NC
21	Jumper Cables	6	ME2011 kit	NC
22	5V Stepper Motor 28BYJ-48	1	ME2011 kit	NC
23	2mm Bolt SS	2	Amazon	1.26
24	2mm Nut SS	2	Amazon	0.83
25	All-Temp Gorilla Glue Hot Glue Stick	1	HSEC Makerspace	0.10
26	TPU Stepper Motor Coupler	1	ME2011 kit	0.5
27	Arduino Uno R3	1	ME2011 kit	NC
28	Arduino 9V Battery Adapter	1	ME2011 kit	NC
29	PIXCELL 9V E522 Battery	1	Walmart	NC
30	All-Weather Gorilla Duct Tape (2")	1	HSEC Makerspace	0.20
31	PLA Photovoltaic Risers (Protective)	4	Amazon, Thingiverse	0.22
32	PLA Solar Panel Mount	1	Amazon	0.95
	DEMO RELATED ITEMS BELOW			
33	Flexible Lamp (Black Color) Variable Light Source	1	Amazon	16.99
			Total Cost ($):	56.53



Made with Tinkercad®

Title: Brave Esboo

Date: 12/10/2023, 7:17:12 PM

Sheet: 1/1

```
// libraries and classes
#include <math.h>     // library for math functions,
such as atan
#include <Stepper.h>  // class for the stepper motor
functions

// Constant variables, doesn't use memory with define
#define DISTANCE 100.0
#define PHOTORESISTOR_PIN1 A1 //photoresistor number 1
analog input
#define PHOTORESISTOR_PIN2 A2 //photoresistor number 2
analog input
#define PHOTORESISTOR_PIN3 A3 //photoresistor number 3
analog input
#define PHOTORESISTOR_PIN4 A4 //photoresistor number 4
analog input
#define WATCHLED  2           // LED is pin 2
#define STARTBUTTON 3         // Start button is pin 3

const int stepsPerRevolution = 2038; // Number of steps
per revolution for 28BYJ-48 motor
const int rPerMinute = 7;

Stepper myStepper(stepsPerRevolution, 8, 10, 9, 11);
//Instance of stepper class, connects to digital pins

void setup() {
  pinMode(WATCHLED, OUTPUT);    // sets the LED pin as
output
  pinMode(STARTBUTTON,INPUT);   // sets the button pin
as input
  myStepper.setSpeed(rPerMinute);
  Serial.begin(9600);
  Serial.println("Setup complete");
}

//Gets the average of 20 luminosity values for a
photoresistor
  int getAverageLuminosity(uint8_t pin){
    int sum = 0; //Creates a sum variable to represent
all of the luminosity readings added together
      for (int i = 0; i < 19; i++){ //Reads and adds the
last 20 values to get the average value
        sum += analogRead(pin);
      }
    return sum/20;
  }

// For reference of later functions
//       y 3         |          2
//       |           |
//       |           |
//       |           |(0,0)
//       |-----------*---------- HOME (0 degrees)
//       |           |
//       |           |
//       |           |
//       |_4 _ _ _ _ |_ _ _ _ _ 1_x
//
//treat the platform as a cartesian plane and determine
which quadrant it lies in

//Gets the  "center of light" or balance point of the
intensity values
  double getCoordinate(int l1, int l2, int l3, int l4){
    double sum = l1 + l2 + l3 + l4;
    return ((l1 * DISTANCE + l2 * DISTANCE) - (l3 *
DISTANCE + l4 * DISTANCE))/sum;
```

```
  }

//Gets the quadrant that the average x and y value are
in
  int getQuadrant(double x, double y){ //returns the
quadrant number
    if (x>0 && y>0){ // if (+, +)
      return 1;
    } else if (x<0 && y>0){ // if  (-, +)
      return 2;
    } else if (x<0 && y<0){ // if (-, -)
      return 3;
    } else if (x>0 && y<0){ // if (+, -)
      return 4;
    } else {
      return 0;
    }
  }

//Finds the angle with respect to the x-axis of the
imaginary cartesian plane
  int getTheta(int quad, double x, double y) {
    double theta = 0.0;
    if (quad == 1 || quad == 3){
      theta = atan(abs(y)/abs(x));// * (180/PI);
    } else if (quad == 2 || quad == 4) {
      theta = atan(abs(x)/abs(y));// * (180/PI);
    }
    double t = (180.0/PI) * theta;
    int finalTheta = ceil(t);
    return finalTheta;
  }

//Finds the angle that the servo needs to rotate with
respect to home.
  int getServoAngle(int quad, int theta){
    return (quad > 0 ? quad - 1 : 0) * 90 + theta;
  }

void loop() {
  // Flash watchdog LED while waiting for button press
  while (digitalRead(STARTBUTTON) == LOW){
    digitalWrite(WATCHLED,HIGH); // LED on
    delay(250);
    digitalWrite(WATCHLED,LOW);  // LED off
    delay(750);
  }
  delay(2000); // Pause for effect
   char msg[128];

  //ROBOT-SPECIFIC CODE STARTS
  //Executes function to read intensity values 20 times
and find the average
  int luminosity1 =
getAverageLuminosity(PHOTORESISTOR_PIN1);
  int luminosity2 =
getAverageLuminosity(PHOTORESISTOR_PIN2);
  int luminosity3 =
getAverageLuminosity(PHOTORESISTOR_PIN3);
  int luminosity4 =
getAverageLuminosity(PHOTORESISTOR_PIN4);
  Serial.println("avgs found"); //debug script:
checkpoint 1
  sprintf(msg, "lum1 = %d, lum2 = %d, lum3 = %d, lum4 =
%d", luminosity1, luminosity2, luminosity3,
luminosity4);
  Serial.println(msg);
  //Gets the average x and y values w.r.t (0,0) from the
average luminosity values
  double xValue = getCoordinate(luminosity1,
luminosity2, luminosity3, luminosity4);
  double yValue = getCoordinate(luminosity2,
luminosity3, luminosity1, luminosity4);

  char buff[6]; //Debug x-value, prints later in msg
  dtostrf(xValue, 4, 2, buff);

  char buff2[6]; //Debug y-value, prints later in msg
  dtostrf(yValue, 4, 2,buff2);

  sprintf(msg, "X = %s, Y = %s", buff, buff2);
  Serial.println(msg);

  //Executes and gets theta
  int finalAngle = 0; //finalAngle represents the
complete number of degrees to rotate w.r.t home position
  int theta = 0; //theta represents the angle to rotate
from the nearest axis
  int quadrant = 0; //quadrant treats surface as a
cartesian plane and notifies which quadrant the point is
in

  if (xValue == 0){
    if (yValue > 0){
      finalAngle = 90;
    } else if(yValue < 0){
      finalAngle = 270;
    } else if (yValue == 0){
      finalAngle = 0;
    }
  } else if (yValue == 0) {
      if (xValue > 0){
        finalAngle = 0;
      } else if (xValue < 0){
        finalAngle = 180;
      } else if (xValue == 0){
        finalAngle = 0;
      }
    } else if (xValue != 0 && yValue != 0){
     quadrant = getQuadrant(xValue, yValue);
     theta = getTheta(quadrant, xValue, yValue);
     finalAngle = getServoAngle(quadrant, theta);
  }

  sprintf(msg, "quadrant = %d, theta = %d, finalAngle =
%d", quadrant, theta, finalAngle);
  Serial.println(msg); //debugs angle, quadrant and
final angle values

  //Finds the steps that the stepper needs to rotate
from the home position
  int steps = ((double)stepsPerRevolution / 360.0) *
(double) finalAngle;
  int stepsccw = steps * -1;
  sprintf(msg, "steps = %d, stepsccw = %d", steps,
stepsccw);
  Serial.println(msg);
  //Rotate the motor by x steps
  myStepper.step(stepsccw);
  delay(10000); // Wait a minute before the next action
  myStepper.step(steps); //returns to the original
position...assuming the power doesn't go out
}
```