The University of Texas at El Paso

Mechanical Engineering

Mechatronics Lab

Amauri Marin

MECH 3103 CRN: 13445

Fall 2025

12/5/2025

Final Project

Catching Sunlight

Alondra Maciel - Ana Lucia Valenzuela - Ignacio Duarte

Introduction

In today's world, technology influences every aspect of our lives. We normally use it to solve important problems, but sometimes we can use it for purposes that don't offer much to society. Unlike past generations, who didn't consider the long-term effects of their actions on the planet, we now have a clearer understanding of how our choices impact the environment. Because of this increased awareness, it is essential to find sustainable solutions that reduce the harm we inflict on Earth.

One of the most important areas that requires improvement is the way we produce and consume energy. Currently, a large portion comes from fossil fuels. Although they have powered our daily lives for decades, they have also become a major contributor to environmental damage. Burning fossil fuels releases greenhouse gases that pollute the air and cause climate change. These emissions trigger extreme weather, disrupt ecosystems, and harm human health. In addition, obtaining fossil fuels requires mining, drilling, and other extraction methods that destroy habitats. These harmful effects show how much we rely on nonrenewable energy sources that are extremely dangerous for the planet's future [1].

As these issues became more recognized, the need for alternatives grew, leading to advances in renewable energy. One of the most promising renewable sources is solar energy, which uses sunlight, a resource that doesn't require mining or extraction [8] . Solar panels have the capacity for converting sunlight into usable power [9], creating a more environmentally friendly option. However, traditional solar panels have a limitation: they are installed at a fixed angle. Because the sun moves throughout the day and year, fixed panels only absorb sunlight efficiently during limited hours. This means they don't operate at their full potential. To improve this, tracking systems were developed to follow the sun's movement, increasing efficiency. However, these systems are far more expensive, making them less accessible [5].

When we began this project, we focused on two main questions: how the system would detect the direction of sunlight and how we could convert that information into movement. After brainstorming, we chose photoresistors because they are inexpensive, reliable, and easy to integrate. We placed four photoresistors evenly at the top of our design so each could collect light data from a different direction.

Next, we needed smooth control of movement. Rather than using simple if–else logic, which can cause sudden jumps, we implemented a PID (Proportional-Integral-Derivative) controller. The PID controller reacts in real time, balancing error correction and prediction to achieve stable motion.

For the physical structure, we chose a sunflower-inspired design—symbolic and functional. Four photoresistors are mounted on the "flower," wires run through the "stem," and all hardware is inside the base to keep everything organized and visually clean. Our design includes two degrees of motion: rotation of the entire stem and tilting of the top section. This allows the system to track the sun on two axes.

Through this project, we aimed to show that creating a solar tracking system does not need to be expensive or overly complex. With just a few accessible components—photoresistors, an actuator, and a PID controller—it's possible to build a functional model inspired by advanced solar tracking systems. Our design demonstrates that practical innovation can also be affordable.

Methodology

The methodology followed for this project consisted of the full design cycle, including sensor placement, mechanical design iterations, electrical integration, and software development using a PID controller. The goal was to create a low–cost solar tracking system capable of responding to changes in light intensity through a closed-loop control system, similar to the systems described in [10].

Sensor Subsystem

To detect the direction of incoming sunlight, four LDRs were mounted at the top of the flower head in a cross-shaped layout. One LDR was positioned at the top, one at the bottom, one on the left, and one on the right, equally spaced across the circular head surface. This layout ensured that each sensor measured light under identical conditions and allowed the system to determine direction by comparing the intensity between regions.

The LDRs were installed flush across the head, which reduced geometric distortion and allowed the averages between sensors to be calculated accurately. The logic used by the Arduino computes two primary differences: the top and bottom average for tilt control, and the left and right average for horizontal rotation. This method is commonly used in solar tracking applications due to its simplicity and reliability [6]. The sensor operation and flow of information through the system follow the logic diagram shown in Figure 1.

Mechanical System Design and Prototyping

The mechanical system consisted of multiple custom 3D printed parts, including the stem, flower head, and top and bottom gear assemblies. These components are shown in Figures 4, 5, 6, and 7. Early in the design, a slip-ring system was tested to allow continuous rotation; however, the slip ring available did not contain enough electrical channels to support all four LDRs and the servo connections simultaneously. Because of this limitation, the design was adjusted so that the base would only rotate 180 degrees rather than continuously. This ensured that wires would not twist or disconnect during operation.

Throughout prototyping, several failures occurred that required redesign. The mounting brackets on the head broke repeatedly due to the thin wall thickness. To improve durability, the brackets were made thicker, holes were enlarged, and chamfers were added to reduce stress concentrations. These changes improved print strength, similar to reinforcement recommendations found in [3].

Pivot points for the tilt mechanism also failed during assembly due to the combined weight of the head, sensors, and wiring. The pivot geometry was modified by increasing thickness and adding fillets to better distribute load. Despite these improvements, the internal space inside the head and stem became a significant challenge. The wires used for the LDRs and servos were much thicker than expected, and once routed through the stem, the tilt mechanism no longer had sufficient clearance to move. This made the head unable to tilt properly during final assembly.

Another issue encountered was weight imbalance. The base, shown in Figure 3, was too light compared to the weight of the head and stem. This caused the lower servo to struggle to rotate the structure, resulting in slow movement and limited torque. Increasing the base weight or using a higher–torque servo would have improved performance.

Electrical Subsystem

The electrical circuit, shown in the TinkerCAD diagram in Figure 10 and the physical breadboard layout in Figure 11, used the Arduino Uno as the main controller. All four LDRs were connected to analog input pins, each paired with a resistor to form a voltage divider. The servos used PWM pins to control pan and tilt motion. The wiring was kept consistent to reduce electrical noise and ensure stable readings. The circuit testing performed outside of the mechanical assembly showed that all components functioned correctly.

Control Algorithm and PID Tuning

To control the movement of the system, a PID controller was implemented, following the concepts described in [2, 4, 7]. The PID algorithm used the differences between sensor averages to compute the error in both the horizontal and vertical directions. A dead-band was applied to filter out small fluctuations in light readings.

The tuning of the PID gains was done manually through trial and error. First, the proportional gain was increased until the system oscillated, then derivative gain was added to stabilize the motion. The integral gain was adjusted last to remove steady-state error. This tuning was performed before the mechanical assembly was finalized, using wheels attached to the servos to simulate motion. During isolated testing, the PID responded smoothly and accurately when light conditions changed.

However, once the system was fully assembled, the mechanical limitations significantly affected PID performance. The added weight, friction, cable resistance, and restricted movement caused the tuned values to behave differently, highlighting the importance of tuning

the system under final load conditions. This behavior is consistent with what is documented in closed-loop system theory [10].

Final Assembly and System Integration

After assembling all subsystems, final testing revealed several limitations. The wires inside the head prevented the tilt mechanism from moving freely, and the lower servo lacked sufficient torque to rotate the structure smoothly. Even though each part of the system worked individually, combining them introduced new constraints that reduced overall functionality.

The assembly renders in Figures 8 and 9 show the internal arrangement of the components and the structure's mechanical layout. While the complete model did not operate exactly as intended, the coding logic, PID implementation, and individual hardware components performed successfully during testing. The primary issues that prevented full operation were mechanical restrictions, wiring congestion, and insufficient torque, all of which would be addressed in future iterations.

Flowchart

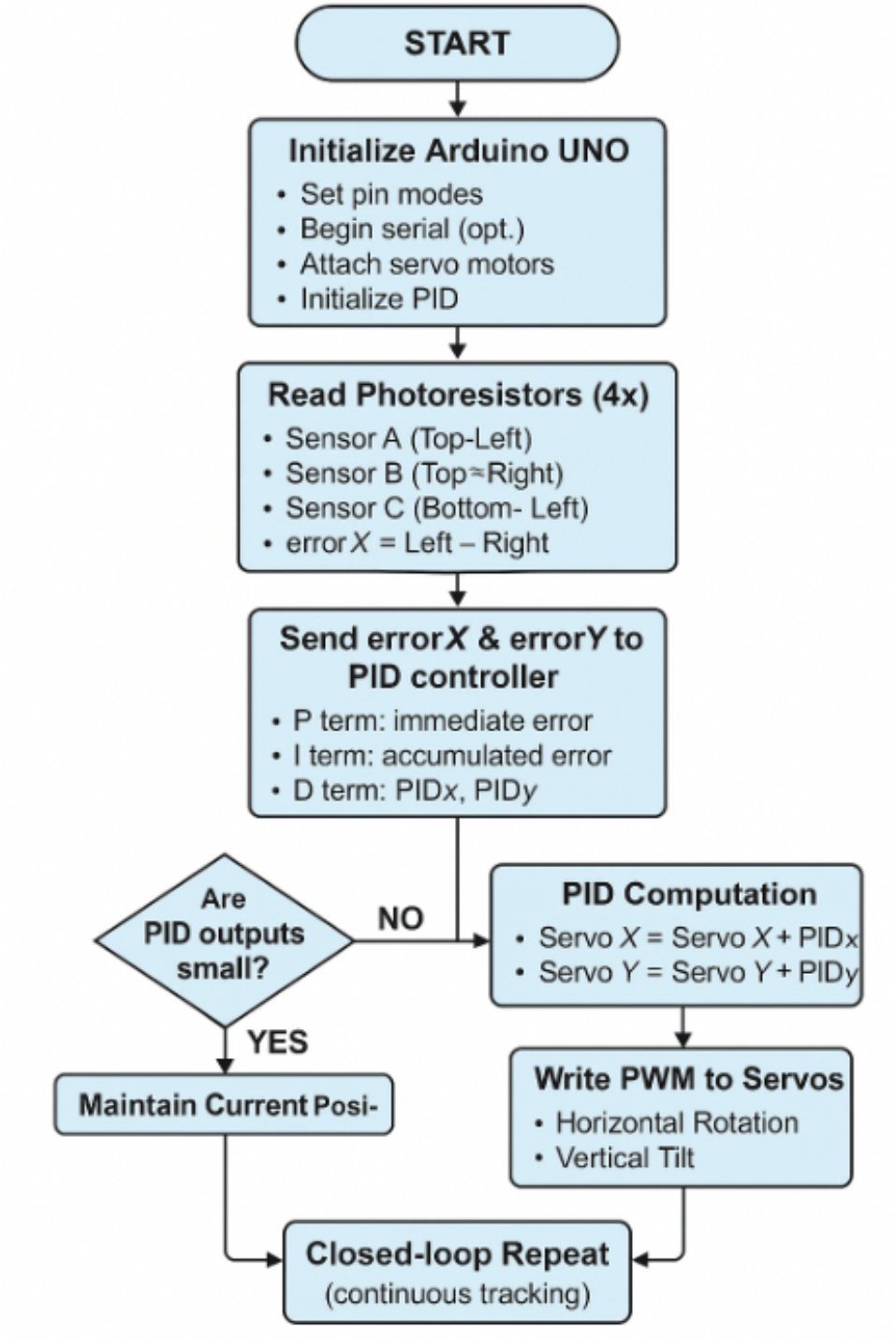
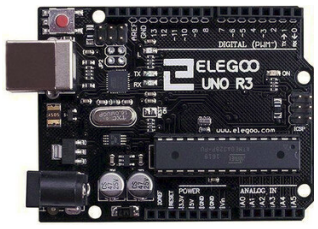


Figure 1: Flowchart

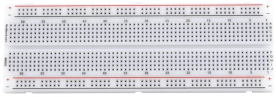
Bill of Materials

The main electronic components used in the project are listed below and illustrated in Figure 2.

- Elegoo Arduino Uno R3 (Fig. 2a)
- Breadboard (Fig. 2b)
- Breadboard jumper cables (Fig. 2c)
- Resistors (Fig. 2d)
- 2x SG-90 servo motors (Fig. 2f)
- 4x photoresistors (Fig. 2e)
- Arduino IDE (not pictured)

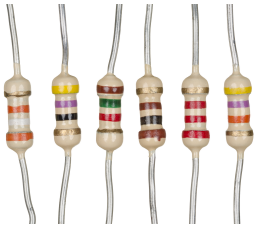


(a) Arduino Uno

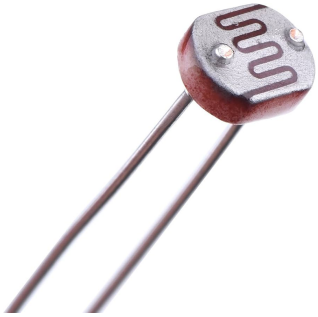


(b) Breadboard

(c) Jumper cables



(d) Resistors



(e) Photoresistor

(f) SG-90 servo motor

Figure 2: Electronic components used in the project.

3D Printed Assembly

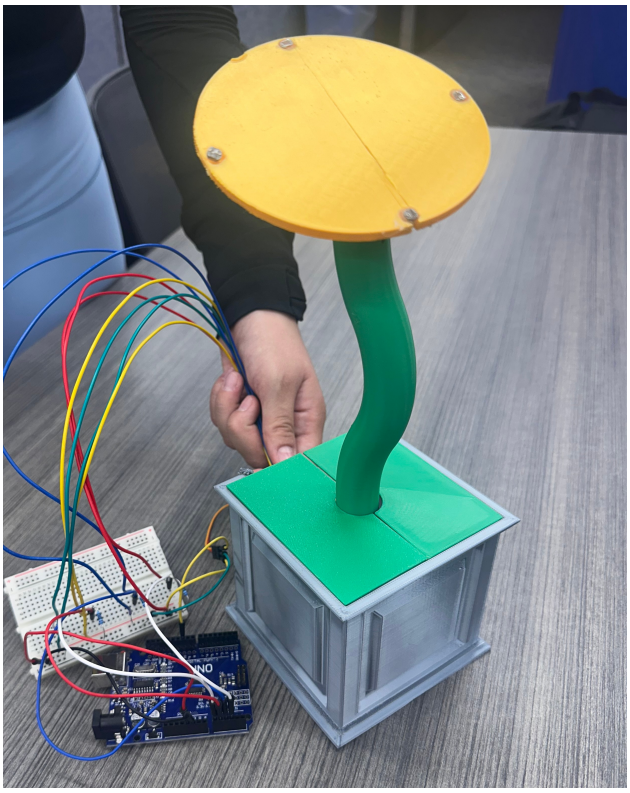


Figure 3: 3D Printed Assembly

Mechanical Parts

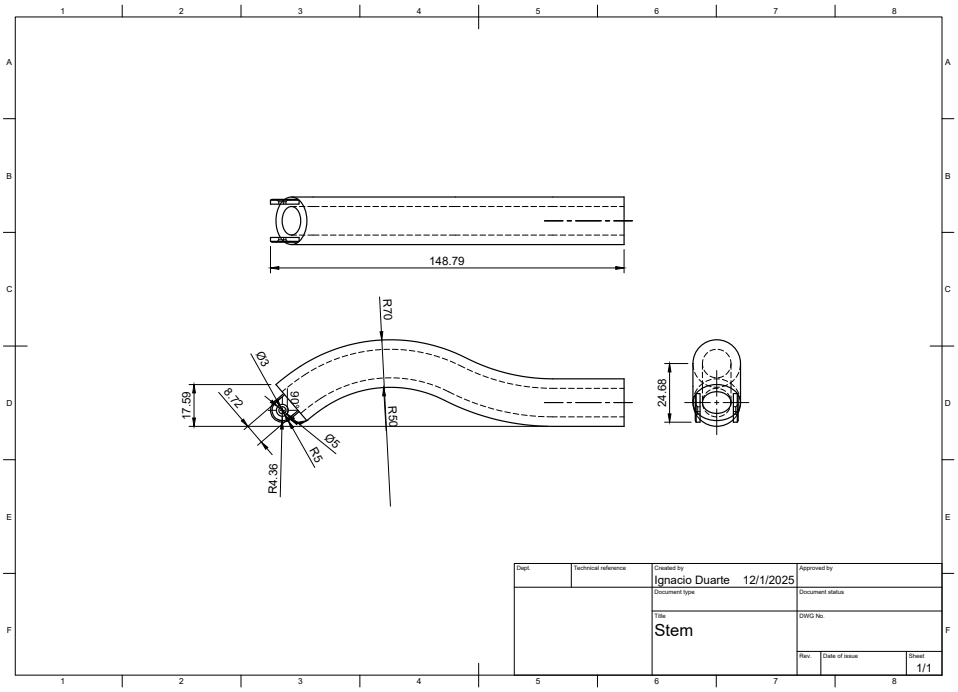


Figure 4: Stem Drawing

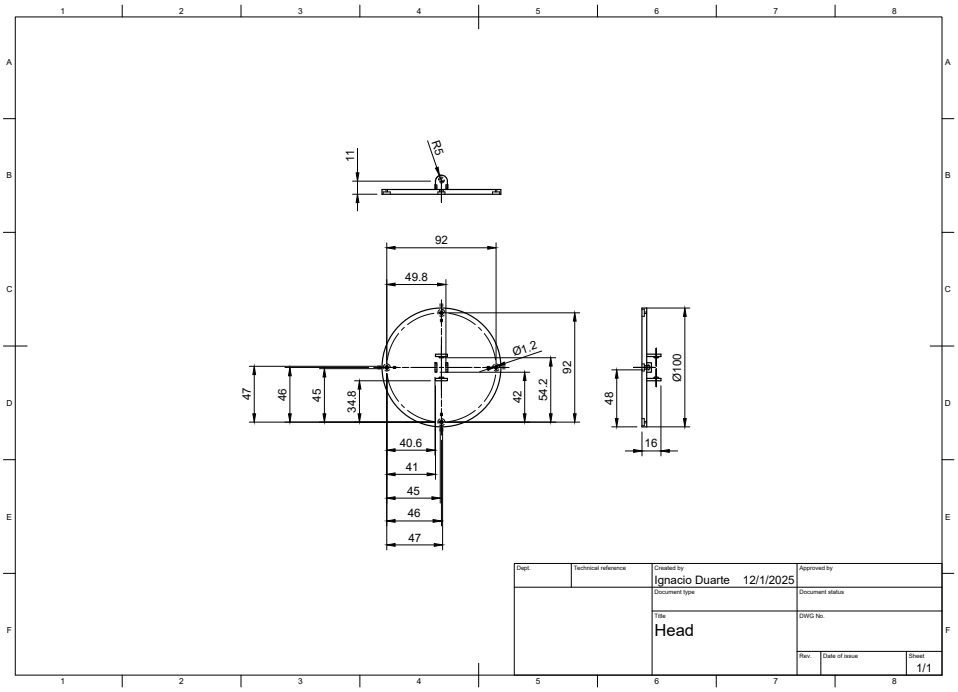


Figure 5: Flower Head Drawing

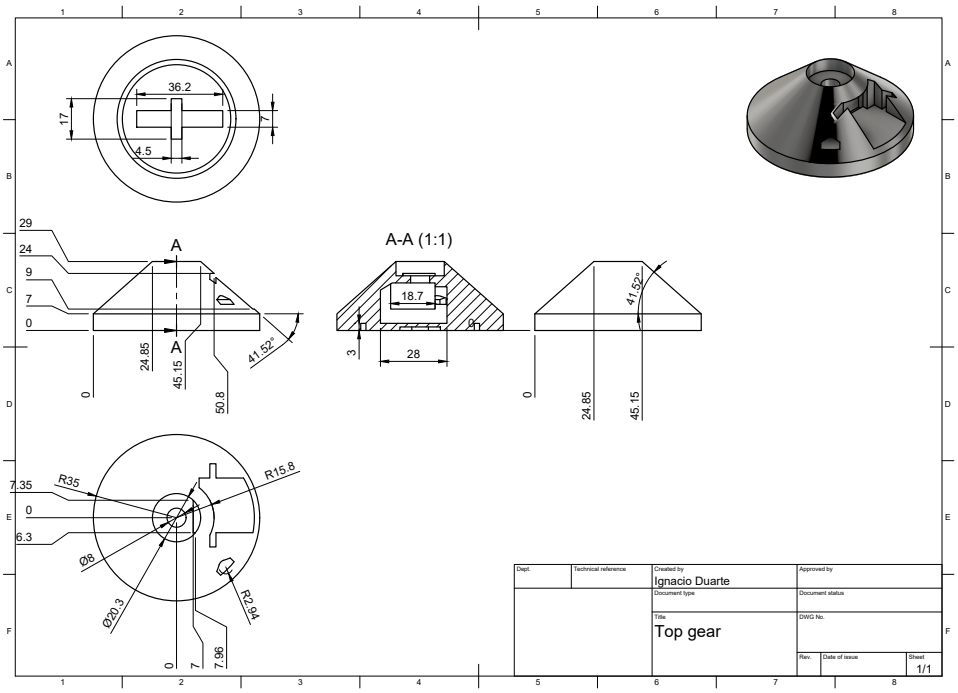


Figure 6: Top Gear Drawing

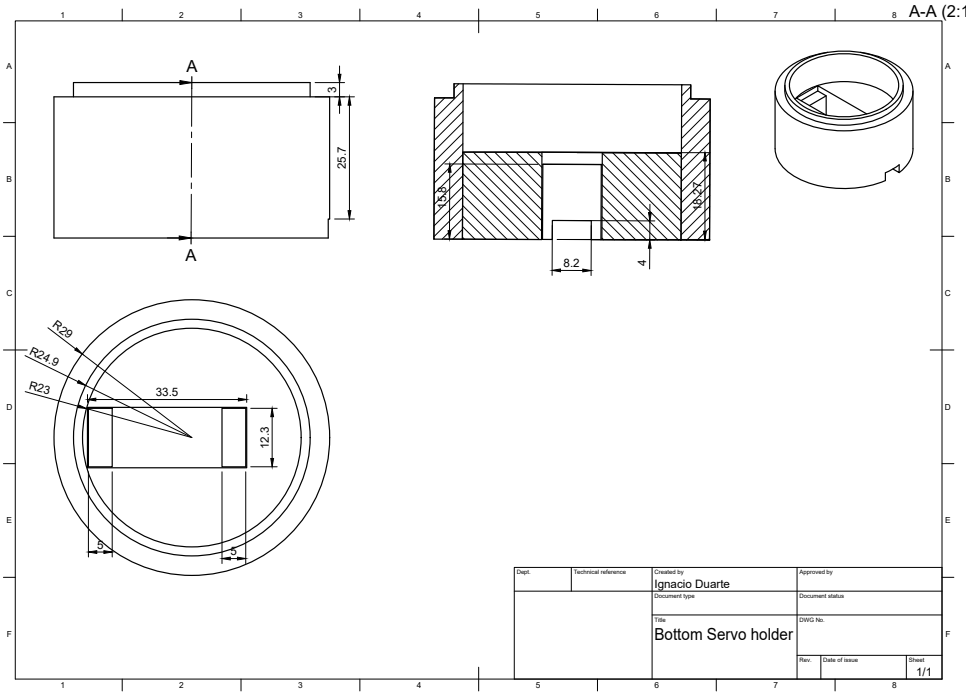


Figure 7: Bottom Gear Drawing

Assembly Renders

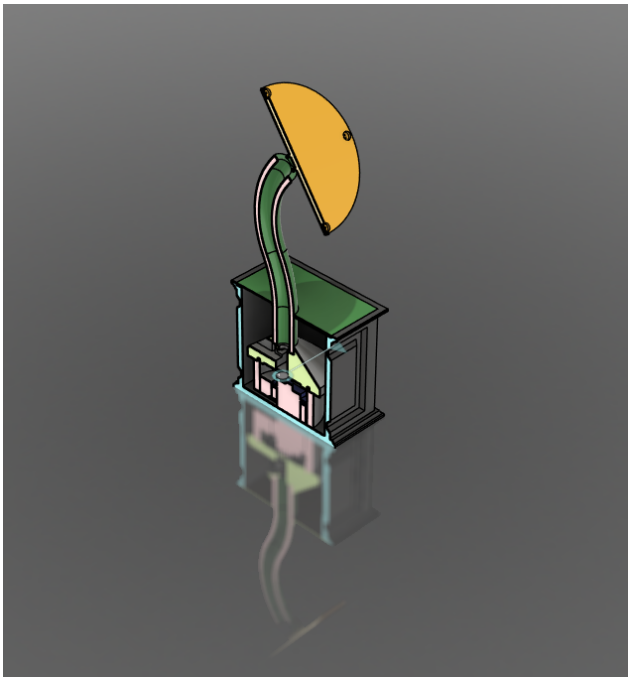


Figure 8: Section View

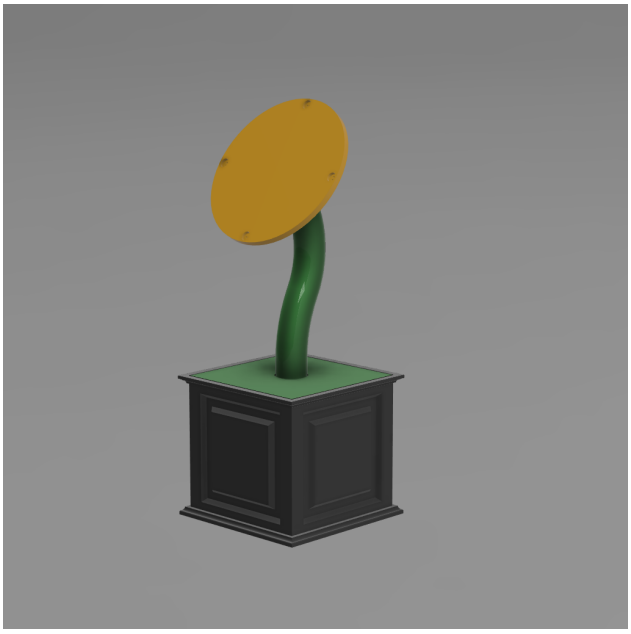


Figure 9: Assembly Render (Exterior View)

Circuit Diagram

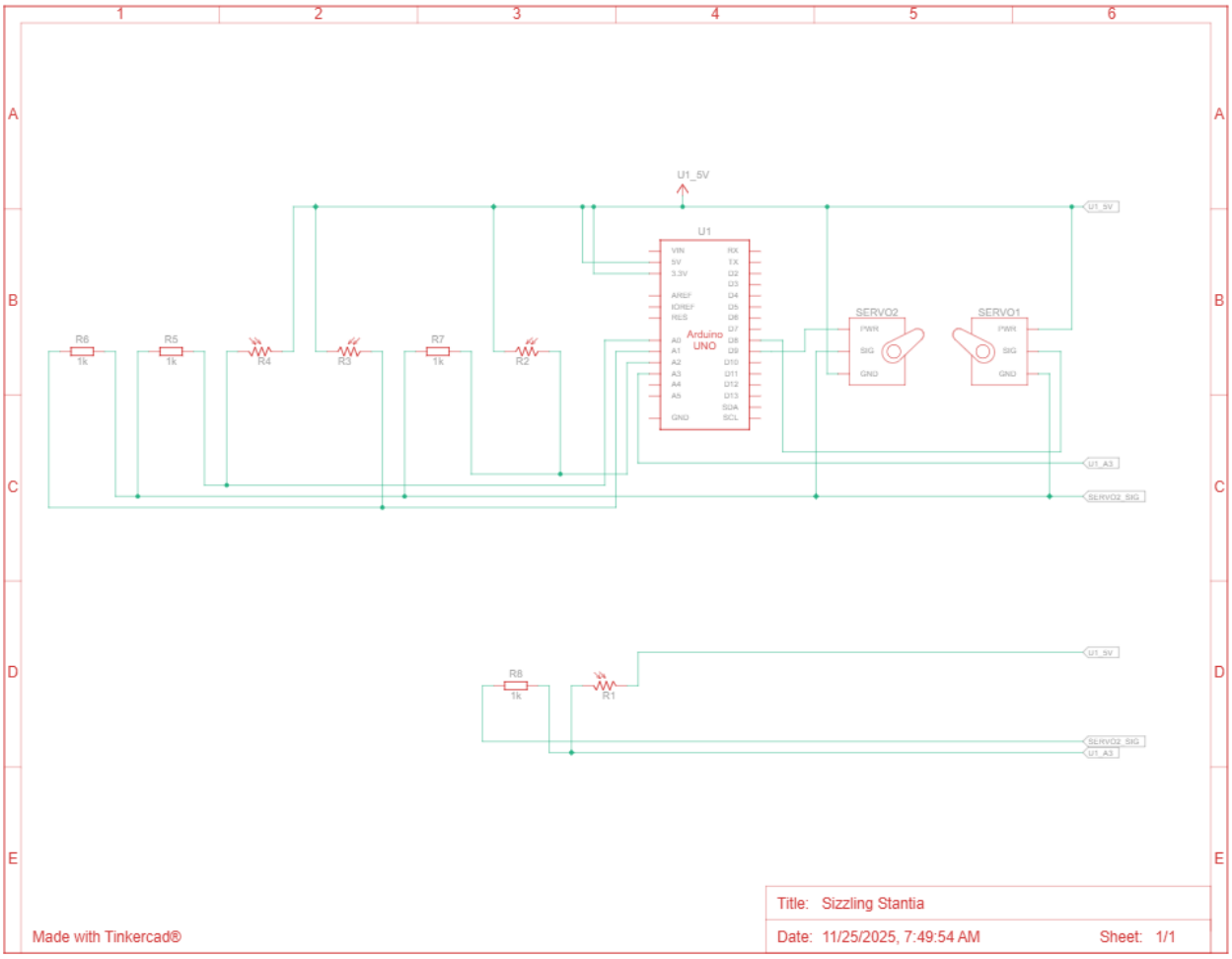


Figure 10: Circuit diagram created in TinkerCAD.

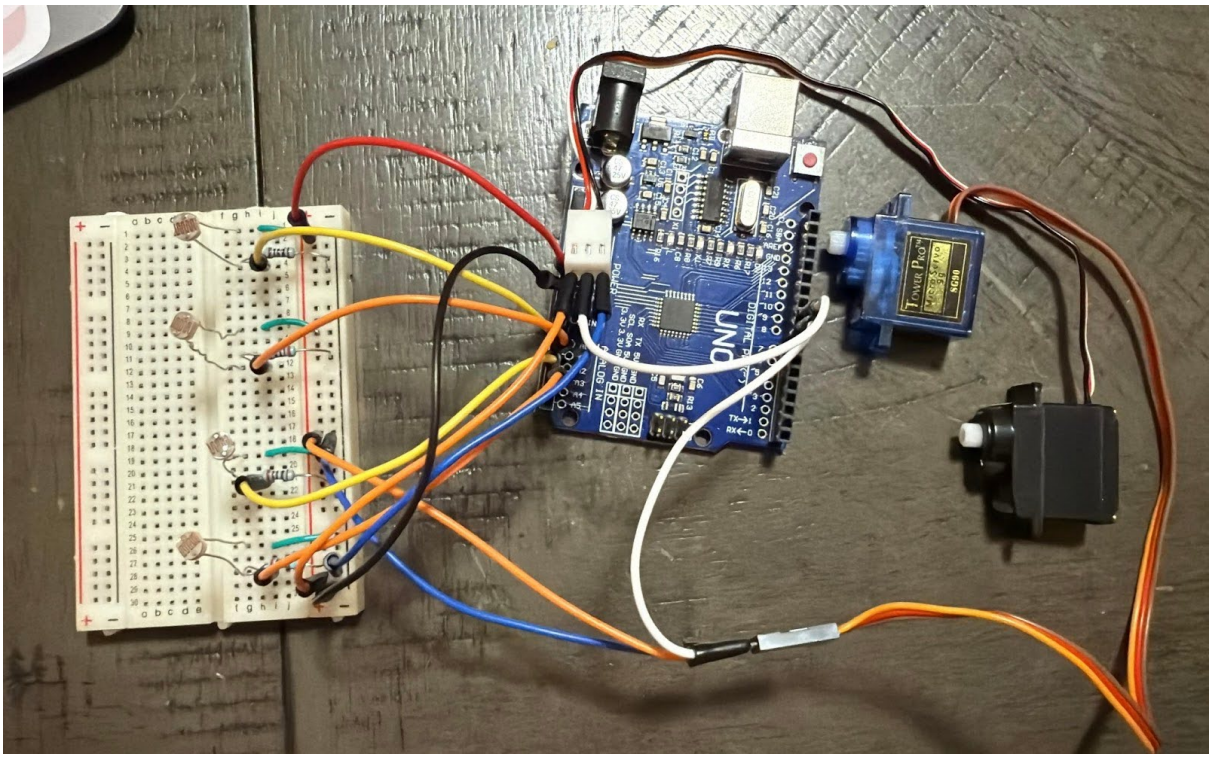


Figure 11: Physical circuit on breadboard.

Code

```
#include <Servo.h>

// Assign the input pins
const int LDR_TL = A0; // Top Left
const int LDR_TR = A1; // Top Right
const int LDR_BL = A2; // Bottom Left
const int LDR_BR = A3; // Bottom Right

// Two digital pins that output PWM signals, also assigning the pin with
// the hardware and code; one is in charge of spinning and the other of tilt.
const int SERVO_PAN_PIN = 9; // horizontal rotation
const int SERVO_TILT_PIN = 10; // vertical tilt

Servo servoPan;
Servo servoTilt;

// Both motors start at 90 degrees (initial position before tracking)
```

```
18 float panAngle = 90; // center
19 float tiltAngle = 90; // center
20
21 // PID gains: proportional, integral, derivative components
22 float Kp_pan = 0.02;
23 float Ki_pan = 0.0005;
24 float Kd_pan = 0.1;
25
26 float Kp_tilt = 0.02;
27 float Ki_tilt = 0.0005;
28 float Kd_tilt = 0.1;
29
30 // PID state variables (tell PID which direction and how much to move)
31 float errorPan = 0, lastErrorPan = 0, integralPan = 0;
32 float errorTilt = 0, lastErrorTilt = 0, integralTilt = 0;
33
34 // Time difference between readings
35 unsigned long lastTime = 0;
36
37 // Dead band: ignore very small errors from sensor noise
38 const int DEAD_BAND = 20;
39
40 void setup() {
41   // Start serial communication
42   Serial.begin(9600);
43
44   // Attach servo motors to their pins
45   servoPan.attach(SERVO_PAN_PIN);
46   servoTilt.attach(SERVO_TILT_PIN);
47
48   // Move servos to the starting positions
49   servoPan.write((int)panAngle);
50   servoTilt.write((int)tiltAngle);
51
52   // Record current time
53   lastTime = millis();
54 }
55
```

```
56 void loop() {
57   // 1) Read light sensors
58   int tl = analogRead(LDR_TL);
59   int tr = analogRead(LDR_TR);
60   int bl = analogRead(LDR_BL);
61   int br = analogRead(LDR_BR);
62
63   // 2) Calculate light intensity averages
64   float topAvg = (tl + tr) / 2.0;
65   float bottomAvg = (bl + br) / 2.0;
66   float leftAvg = (tl + bl) / 2.0;
67   float rightAvg = (tr + br) / 2.0;
68
69   // 3) Compute errors (how we should rotate)
70   errorTilt = topAvg - bottomAvg;
71   errorPan = leftAvg - rightAvg;
72
73   // 4) Ignore small errors (dead band)
74   if (abs(errorTilt) < DEAD_BAND) errorTilt = 0;
75   if (abs(errorPan) < DEAD_BAND) errorPan = 0;
76
77   // 5) Time step for integral and derivative terms
78   unsigned long now = millis();
79   float dt = (now - lastTime) / 1000.0; // seconds
80   if (dt <= 0) dt = 0.001; // safety
81   lastTime = now;
82
83   // 6) PID calculation for pan (horizontal)
84   integralPan += errorPan * dt;
85   float derivativePan = (errorPan - lastErrorPan) / dt;
86   float outputPan = Kp_pan * errorPan
87                   + Ki_pan * integralPan
88                   + Kd_pan * derivativePan;
89   lastErrorPan = errorPan;
90
91   // 7) PID calculation for tilt (vertical)
92   integralTilt += errorTilt * dt;
93   float derivativeTilt = (errorTilt - lastErrorTilt) / dt;
```

```
  float outputTilt = Kp_tilt * errorTilt
                   + Ki_tilt * integralTilt
                   + Kd_tilt * derivativeTilt;
  lastErrorTilt = errorTilt;

  // 8) Incremental changes to servo angles
  panAngle += outputPan;
  tiltAngle += outputTilt;

  // 9) Add limits to the servo angles
  panAngle = constrain(panAngle, 0, 180);
  tiltAngle = constrain(tiltAngle, 0, 180);

  // 10) Command servos to new positions
  servoPan.write((int)panAngle);
  servoTilt.write((int)tiltAngle);

  // 11) Show real-time sensor readings and angles
  Serial.print("TL:"); Serial.print(tl);
  Serial.print("␣TR:"); Serial.print(tr);
  Serial.print("␣BL:"); Serial.print(bl);
  Serial.print("␣BR:"); Serial.print(br);
  Serial.print("␣|␣panAngle:"); Serial.print(panAngle);
  Serial.print("␣tiltAngle:"); Serial.println(tiltAngle);

  // 12) Small delay so the system doesn't update too fast
  delay(20);
}
```

Listing 1: Solar tracker PID control code

Conclusion

Throughout the development of our project, the process of building the system required constant trial and error, we spent a good amount of time making sure the program responded correctly to changes in light and that the PID controller adjusted smoothly. While the code worked as intended, the hardware presented challenges that didn't allow the system to work properly. The greatest challenges that we faced were time limitations and issues related to 3D printing and assembly. During the first stages, we designed several versions of the hardware and created a smaller prototype to be able to focus on the coding portion. Once it was time to print the final structure, we included all the needed features to install the wiring, sensors, and motors. However, we encountered printing errors, ran out of filament, and did not have enough time to reprint or adjust the parts. Because of these setbacks, we were unable to confirm whether all the wiring and electrical components would fit properly inside the painted structure.

When we attempted to assemble the base, we realized that the components were too cramped, and we had understated the thickness and lack of flexibility of the cables. As a result, we removed one of the base walls just to make the pieces fit, but this caused the base to be too light. With the weight of the long stem and the heavy flower head at the top, it became difficult to rotate. Another issue was that we could not test the installation of the second motor inside the flower's vine, which meant we couldn't enable the tilting motion for the top of the structure.

Even though the hardware or printed parts had problems, the circuit worked perfectly when tested before installation. We attached wheels to the motors to simulate movement and confirmed that the PID controller responded as expected. When light hit any photoresistor, the system processed the signal and the PID responded by rotating the mechanism toward the light's direction. Once it reached the correct position, the controller implemented small, fine adjustments to reduce the error between the intended location and the actual output. The result proved that the coding logic was successful.

Looking back, we realized that if more weight was added to the base, the rotation would've been smoother and more stable. We also noticed that the stem had been printed too narrow for the cables we used, this caused the cables to be extremely crowded and ended in a restricted movement. Another improvement we considered, but didn't have time to test, was adding a slip ring. This extra component would've allowed a smoother continuous rotation without twisting the wires and could've solved many mechanical issues that we encountered. A slower servo speed could be the solution to the destabilization seen during the rotation.

Additionally, we overlooked certain structural factors. We assumed that silicone glue would be strong enough to hold the servo horns to the plastic parts, but the structure was much heavier than expected, and the glue did not bond securely. The holes for the horns were also not deep enough, causing them to slip and making the structure fall. We also failed to consider printing tolerances, it is known that it's extremely important to consider the variations that can occur when printing with plastic. Because we didn't consider these, several pieces did not fit together as intended.

Despite these challenges, the project demonstrated that it is possible to create a simple at home solar tracking system using super basic components from an Arduino kit. If more time was available, we know exactly which areas we would redesign: making the stem wider, adding more weight to the base, adjusting the motor speed, properly securing the servo horns, and utilizing a slip ring. Although the final physical model didn't work as we thought, we noticed that the code PID worked correctly. This project taught us the importance of considering tolerances, weight distribution and other factors that determine if the system will work properly including the hardware area.

References

[1] Environmental Protection Agency. (n.d.). *EPA*. Retrieved from https://www.epa.gov/energy/learn-about-energy-and-its-impact-environment

[2] GeeksforGeeks. (2023, October 25). *Proportional Integral Derivative Controller in control system*. Retrieved from https://www.geeksforgeeks.org/electronics-engineering/proportional-integral-derivative-controller-in-control-system/

[3] EnergySage. (n.d.). *Is a solar tracking system worth it?*. Retrieved from https://www.energysage.com/business-solutions/solar-trackers-everything-need-know/

[4] Kohanbash, D. (2019, December 24). *PID control (with code), verification, and scheduling*. Retrieved from https://www.robotsforroboticists.com/pid-control/

[5] Mishra, S. (2025, July 29). *Solar tracking system: Its working, types, Pros, and cons*. Retrieved from https://www.solarsquare.in/blog/solar-tracker/

[6] Electronics Notes. (n.d.). *Light dependent resistor LDR: Photoresistor*. Retrieved from https://www.electronics-notes.com/articles/electronic_components/resistors/light-dependent-resistor-ldr.php

[7] UATeam. (n.d.). *Python PID controller example: A complete guide*. Retrieved from https://medium.com/@aleksej.gudkov/python-pid-controller-example-a-complete-guide-5f35589eec86

[8] U.S. Department of Energy. (n.d.). *Solar Energy Basics*. Retrieved from https://www.energy.gov/eere/solar/solar-energy-basics

[9] Institute for Environmental Research and Education. (2025a, June 7). *How does the way we make power affect our environment?*. Retrieved from https://iere.org/how-does-the-way-we-make-power-affect-our-environment/

[10] Teja, R. (2024, September 17). *Closed loop system: How it works & examples*. Retrieved from https://www.electronicshub.org/closed-loop-system/