Experimental Techniques in Engineering and Physics

Final Report

Azhar Ahmed & Rohit Biswas

04/29/25

R2D3: Solar Tracker Inspired by R2D2

Abstract

This report presents the full lifecycle of a dual-axis solar tracker designed to maximize photovoltaic efficiency by maintaining optimal orientation to sunlight. Inspired by the form of R2D2, the system integrates an Arduino Uno, servo motors, a solar panel for light sensing, and custom polylactic acid (PLA) 3D-printed housing. Early stages focused on SolidWorks modeling and mechanical layout, followed by iterative refinement to accommodate component fit, movement clearance, and wiring. During assembly, torque limitations and alignment issues prompted structural changes, including the addition of a secondary base and a simplified mounting armature. Electrical setbacks, including unstable USB-C connections and overheating adapters, led to system-level debugging and a hardware shift to a USB-A PC environment. A dynamic analog feedback algorithm was implemented and successfully used to detect sunlight intensity and drive the tracker's dual-axis adjustment. Despite structural imperfections, the final build achieved stable, functional tracking and fulfilled the project's engineering and design objectives.

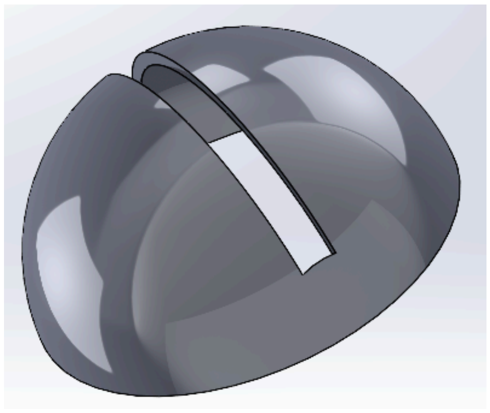
I – Introduction

Solar tracking technologies enhance photovoltaic efficiency by maintaining optimal panel orientation to sunlight. In this project, we aimed to develop a compact, dual-axis solar tracker using an Arduino Uno, servo motors, and a 3D-printed structure. The project was inspired by the iconic R2D2 form, as shown in the figure above, aiming for both functionality and design clarity.

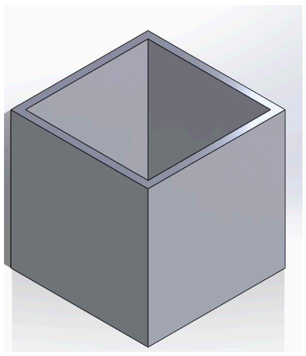
The tracker was designed to adjust both azimuth and elevation using servo motors, with inputs taken from a solar panel acting as a directional light sensor. We used SolidWorks for the mechanical design and PLA, a biodegradable thermoplastic derived from renewable resources, for 3D printing the components. The project proceeded through multiple design phases, starting with sketches, followed by progressive CAD modeling, physical prototyping, and software integration.

II – Work Performed

2.1 Initial Concept & Modeling



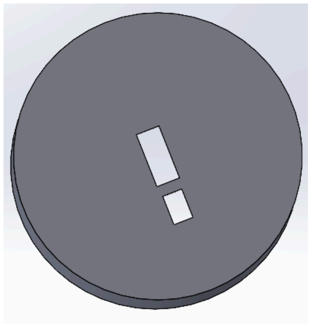
Our initial design was built around a circular base with a rectangular cube housing for electronics and a hemisphere, as shown to the right, to support the solar panel and vertical servo. We initially considered using light-dependent resistors (LDRs) for sensing, but ultimately chose to rely on a solar panel as both the sensing and tracking unit.



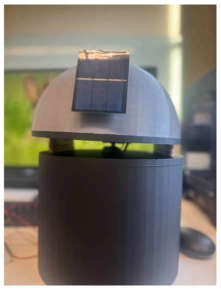
We translated early sketches into SolidWorks, culminating in a structurally complete model. The lower storage cube to the right was designed to be 150 mm x 150 mm x 120 mm, with 16 mm walls and internal spacing to house the Arduino, battery pack, and wiring. The hemisphere included a slit for solar panel placement and cutouts for servo arm clearance.

2.2 Fabrication Adjustments and Secondary Base Design

Upon receiving our 3D prints, several unexpected design issues arose. The horizontal motor struggled to rotate the full upper assembly, including the hemisphere. To reduce the torque load, we introduced a secondary cylindrical base mounted on top of the horizontal servo motor, which allowed only the hemisphere to rotate. A 1-inch passthrough hole was drilled in this new base to maintain wiring continuity from the top.



However, this modification led to two alignment issues: the vertical motor within the hemisphere was not precisely centered, and the horizontal motor in the upper lid to the right was slightly misaligned. This created a noticeable tilt in the final structure. Due to limited time and printing capacity at the university's Lite Center, we chose to proceed without reprinting the corrected parts. To adjust for the misalignment, we added popsicle stick stacks to keep it stable, as shown to the right.



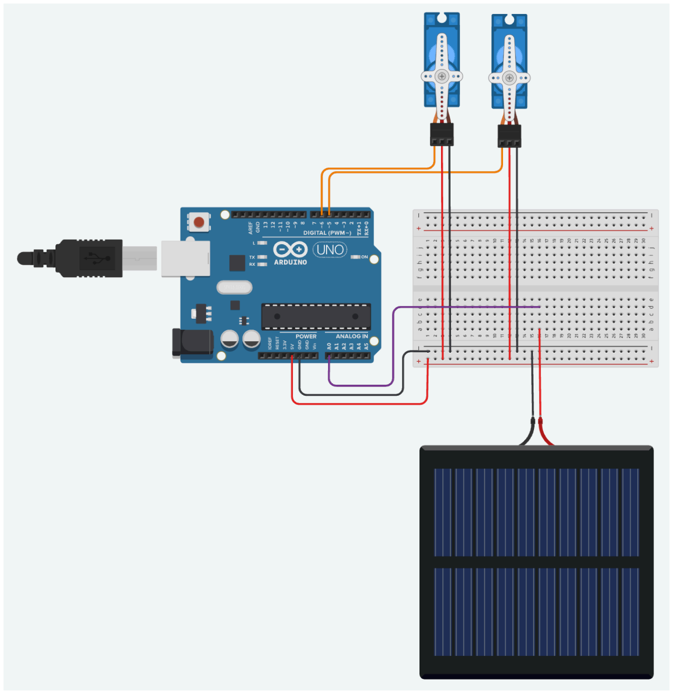
To connect the solar panel, we attached a standard servo horn to a popsicle stick armature as shown to the right. The arm was glued to a milled wooden connector. This formed a basic pivot arm to support vertical panel movement.

2.3 Electrical Integration

Electrical integration included the Arduino Uno, two servo motors, jumper wires, and the solar panel. A key issue arose with code uploading: when using MacBooks with USB-C to USB-A adapters as shown to the right, the Arduino failed to connect. Further investigation revealed the adapter was

overheating due to back current from the Arduino. Switching to a PC with native USB-A to USB-mini B connectivity resolved the issue.

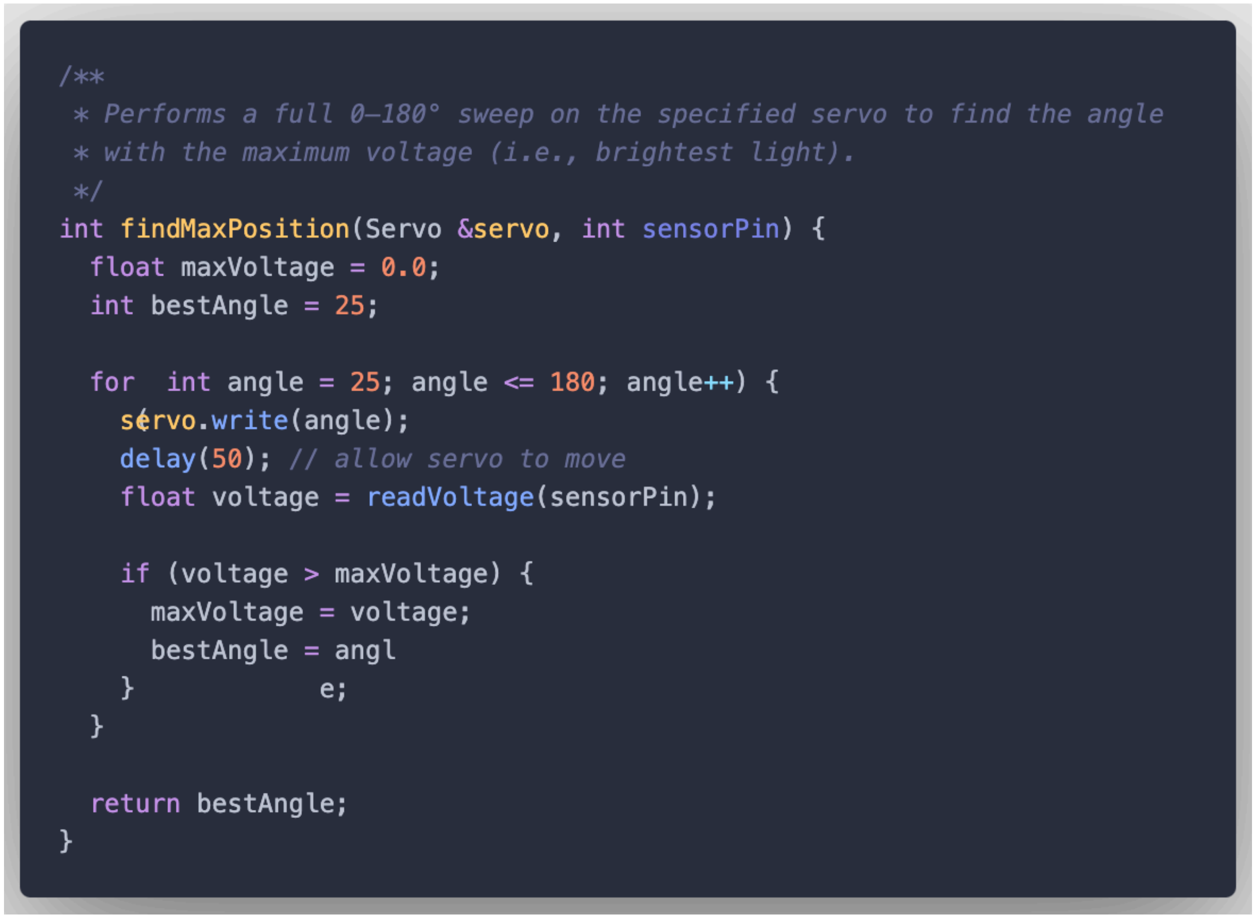
The wiring was otherwise stable, and all power and signal lines were routed through pass-through holes in the housing. The circuit diagram below is what we used to configure our model:



All elements of the circuit are wired in series, with the exception of the power terminal from the panel being connected to the analog pin (A0) on the Arduino to read the voltage. The analog signals will vary depending on the detection of the light, in particular, the light source intensity; if the intensity is greater, the Arduino would read a lower voltage compared to the 5V steady state reading, signaling light incident on the panel.

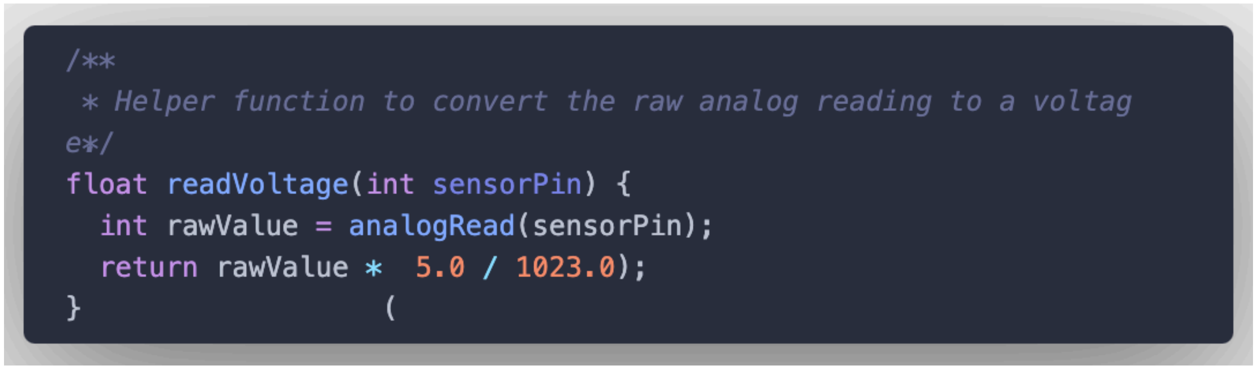
2.4 Software Development and Revisions

Our code below reads data from the solar panel to detect the direction of strongest light and adjusts the servo motors accordingly. Horizontal motion is managed by the motor housed in the lower cube, while vertical tilt is managed by the motor within the upper hemisphere. The control uses PWM signals for smooth transitions and includes logical safeguards to avoid over-rotation.



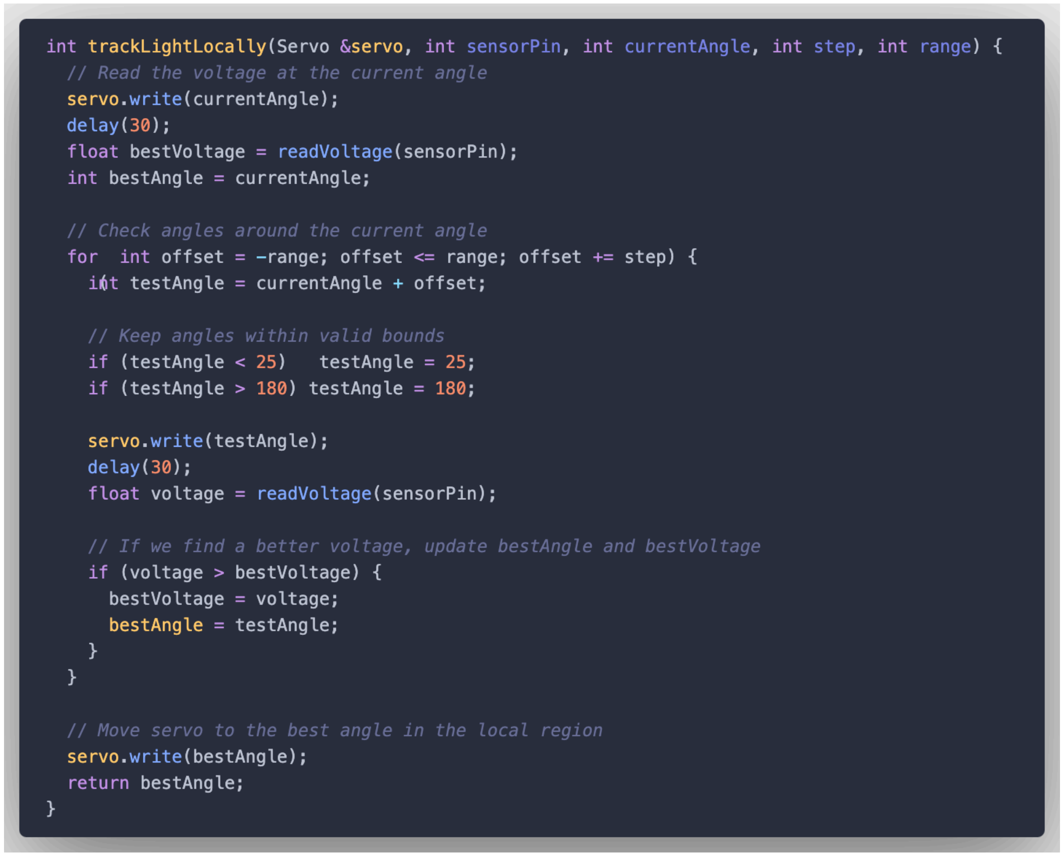
```
/**
 * Performs a full 0–180° sweep on the specified servo to find the angle
 * with the maximum voltage (i.e., brightest light).
 */
int findMaxPosition(Servo &servo, int sensorPin) {
  float maxVoltage = 0.0;
  int bestAngle = 25;

  for  int angle = 25; angle <= 180; angle++) {
    servo.write(angle);
    delay(50); // allow servo to move
    float voltage = readVoltage(sensorPin);

    if (voltage > maxVoltage) {
      maxVoltage = voltage;
      bestAngle = angl
    }            e;
  }

  return bestAngle;
}
```

The readVoltage() function in the code above is the main component, as it is defined below. This helper function reads in the analog voltage from the A0 pin on the Arduino and translates it to intelligible values by multiplying by 5.0 / 1023.0.



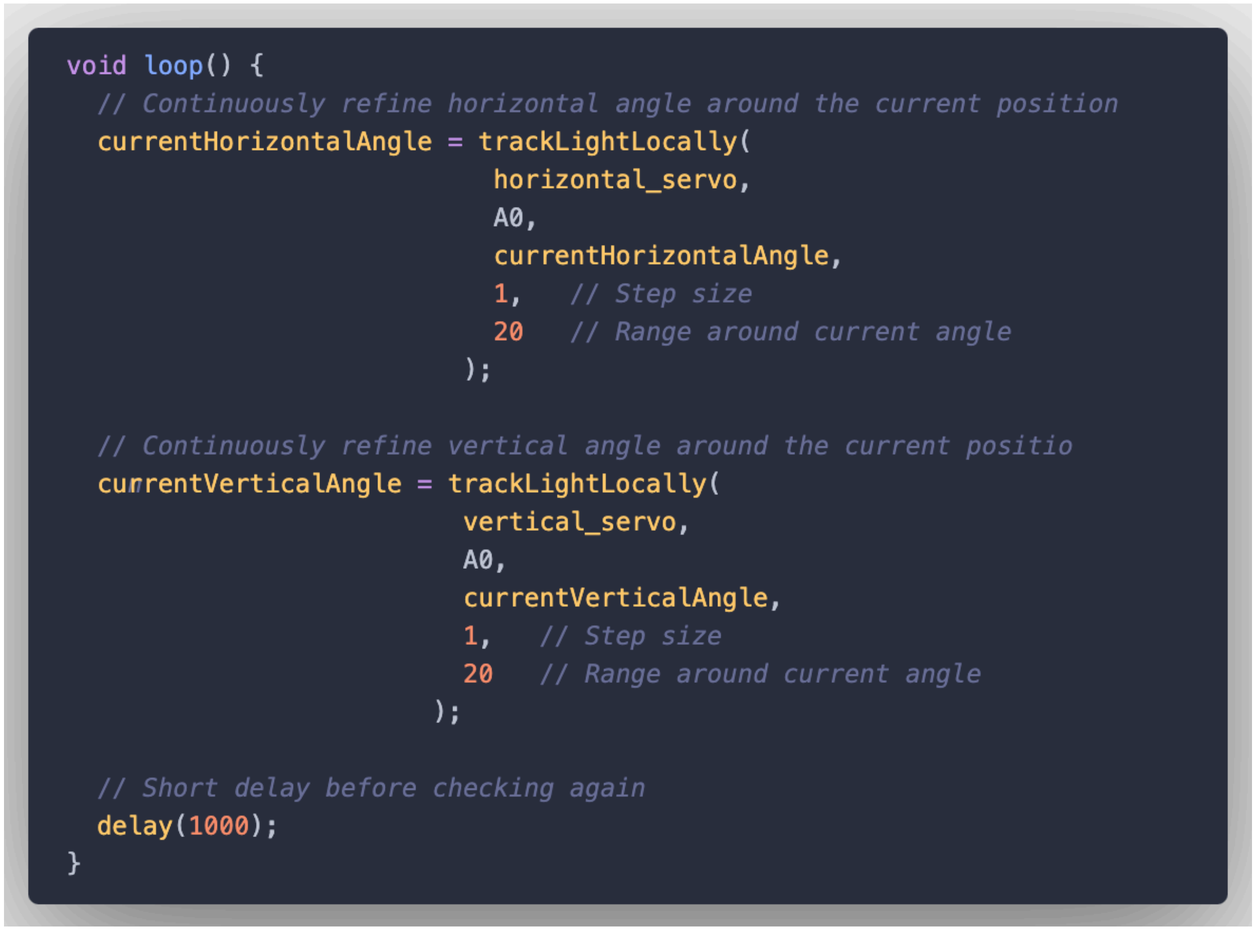
```
/**
 * Helper function to convert the raw analog reading to a voltag
e*/
float readVoltage(int sensorPin) {
  int rawValue = analogRead(sensorPin);
  return rawValue *  5.0 / 1023.0);
}                  (
```

To add on to the above code, the code below is implemented after finding a maximum value. To model a real-life scenario, we created a local tracking implementation that models solar tracking. Since the sun's azimuthal angle does not change very much throughout the day, this local tracking feature would, in principle, allow us to easily follow the sun without doing a full scan every time. By analyzing a hotspot zone, we receive data about, we are able to do more incremental tracking.



```
int trackLightLocally(Servo &servo, int sensorPin, int currentAngle, int step, int range) {
  // Read the voltage at the current angle
  servo.write(currentAngle);
  delay(30);
  float bestVoltage = readVoltage(sensorPin);
  int bestAngle = currentAngle;

  // Check angles around the current angle
  for  int offset = -range; offset <= range; offset += step) {
    int testAngle = currentAngle + offset;

    // Keep angles within valid bounds
    if (testAngle < 25)   testAngle = 25;
    if (testAngle > 180) testAngle = 180;

    servo.write(testAngle);
    delay(30);
    float voltage = readVoltage(sensorPin);

    // If we find a better voltage, update bestAngle and bestVoltage
    if (voltage > bestVoltage) {
      bestVoltage = voltage;
      bestAngle = testAngle;
    }
  }

  // Move servo to the best angle in the local region
  servo.write(bestAngle);
  return bestAngle;
}
```

The code below puts the new dynamic tracking method to use, as we loop through the horizontal and vertical zones, we want to get our hotspot values to then dynamically track light. In other words, instead of scanning a full 180 degrees on each update, the system now targets movement within a smaller range based on recent light positions. This reduces memory usage and increases responsiveness, mimicking realistic solar tracking behavior more closely.



```
void loop() {
  // Continuously refine horizontal angle around the current position
  currentHorizontalAngle = trackLightLocally(
                             horizontal_servo,
                             A0,
                             currentHorizontalAngle,
                             1,   // Step size
                             20   // Range around current angle
                           );

  // Continuously refine vertical angle around the current positio
  currentVerticalAngle = trackLightLocally(
                           vertical_servo,
                           A0,
                           currentVerticalAngle,
                           1,   // Step size
                           20   // Range around current angle
                         );

  // Short delay before checking again
  delay(1000);
}
```

In addition to this unique strategy, we also made some finer adjustments to our code that are of importance based on our mechanical design:

- Adjusted the initial zero point to 25° on the vertical servo to avoid physical collisions with the base.

- Increased delays between motor movements to stabilize the structure before each reading.

III – Results

The completed tracker successfully demonstrated dual-axis functionality. Despite structural misalignments, the system:

- Could rotate horizontally and vertically
- Detected brighter light sources and reoriented accordingly
- Showed stable operation with the scanning algorithm

Limitations include:

- Tilted vertical axis due to off-center mounting
- Uneven weight distribution from the secondary base

Still, the system operated within expectations for a proof-of-concept prototype.

IV – Conclusions and Future Considerations

This project highlighted the interdependence of mechanical design, electrical stability, and coding strategy. From SolidWorks modeling to real-world assembly, every phase required adaptation and refinement.

Critical lessons include:

- Align mechanical parts precisely to reduce system strain
- Account for motor torque limitations in early design
- Test all USB connectivity options before relying on adapters
- Build software that tolerates physical imperfections

Future improvements could include:

- Using geared or brushless servos for stronger rotation

- Integrating a PCB for neater and more reliable wiring
- Redesigning the base for better weight distribution

The final build, while imperfect, fulfilled the initial goal of building a functioning dual-axis tracker and provided a robust learning experience in engineering problem-solving, prototyping, and integration.

References

Arduino® UNO R3 product reference manual. Retrieved from Arduino Documentation

Banerjee, R. (2015). Solar tracking system. International Journal of Scientific and Research Publications, 5(3), 1–7. Retrieved from https://www.ijsrp.org/research-paper-0315/ijsrp-p3923.pdf

Deekshith K., Aravind D., Nagaraju H., Reddy B. (2015). Solar tracking system. International Journal of Scientific & Engineering Research, 6(9), 994–995. Retrieved from https://www.ijser.org/researchpaper/SOLAR-TRACKING-SYSTEM.pdf

Zhang, Y., Li, X., & Wang, J. (2024). Advancements in renewable energy technologies: A comprehensive review. Renewable Energy Reports, 10(2), 123–145. https://doi.org/10.1016/j.rer.2024.01.016