IoT

Internet of Things

STEM

SCIENCE TECHNOLOGY
ENGINEERING MATHEMATICS

OHIO UNIVERSITY
1804
Voinovich School of Leadership and Public Affairs

AMERICAN ELECTRIC POWER FOUNDATION

PORTSFUTURE
IMAGINING THE OPPORTUNITIES, GATHERING YOUR IDEAS
THE FACILITY AT PIKETON, OHIO

Internet of Things Student STEM Project
Jackson High School

Lesson 3: Arduino Solar Tracker

Lesson 3 – Arduino Solar Tracker

Time to complete Lesson

60-minute class period

Learning objectives

- Students learn about using Solar Power & a DC battery to provide power to the microcontroller & all connected components
- Students learn about servo & stepper motors and why they should use one over the other

21st century technical skills gained through this activity

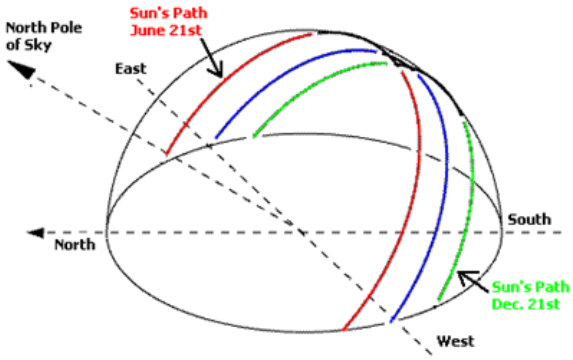
- Electronic Circuit Design
- Computer Programming
- AC & DC Power
- Electronic Motors

Credits!

Code and tutorial based on the great open source lib & example code at:
https://circuitdigest.com/microcontroller-projects/arduino-solar-panel-tracker

Introduction

In this lesson, we are going to make a single axis solar panel tracker using Arduino, in which we will use two LDRs (Light dependent resistor) to sense the light and a servo motor to automatically rotate the solar panel work on a range of principles with the purpose of aligning your panel directly towards the sun. A single axis solar tracker improves solar output by around 25% according to this article onJournal of Power and Energy Engineering.



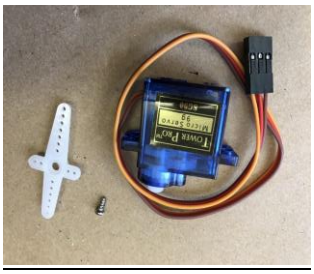
Source: https://file.scirp.org/pdf/JPEE_2014050814004447.pdf

This solar tracker control system is designed to take light measurements from the east and west (left and right) side of the solar panel and determine which way to move the panel to point it directly at the source of the light can provide the supply the maximum power. A servo is used to actuate the panel tracker; these are available in a broad range of sizes and can be scaled according to your panel size. The prototype is very easy to build. Below you will find the complete description of how it works and how the prototype is made.

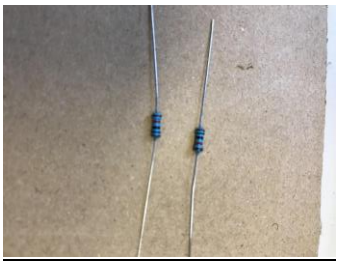
Required Components:

Pencil - cut to <n>cm with a 45° angled face on one end

- Servo



- 10k Ohm Resistors (x2)



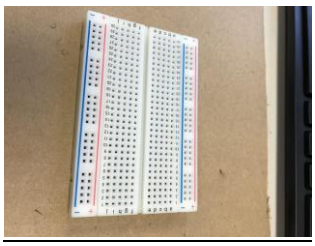
- LED

- Hot Glue
- Wires
- Solar Panel

- Arduino Uno

- LDR (Light dependent resistor) (x2)

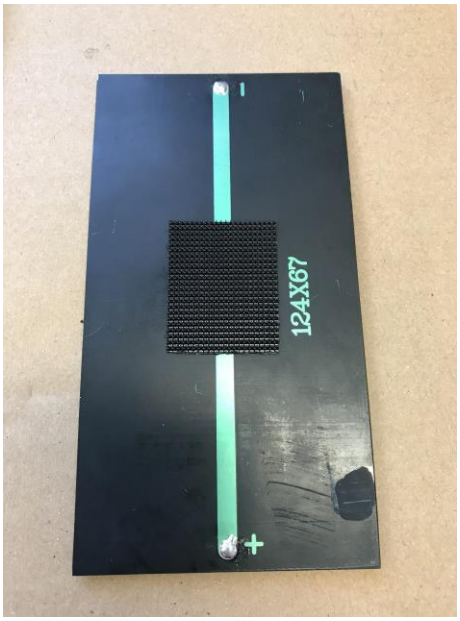
- Breadboard



- Foam Base
- Velcro Strips
  - Rough Side (x3)
  - Soft Side (x2)

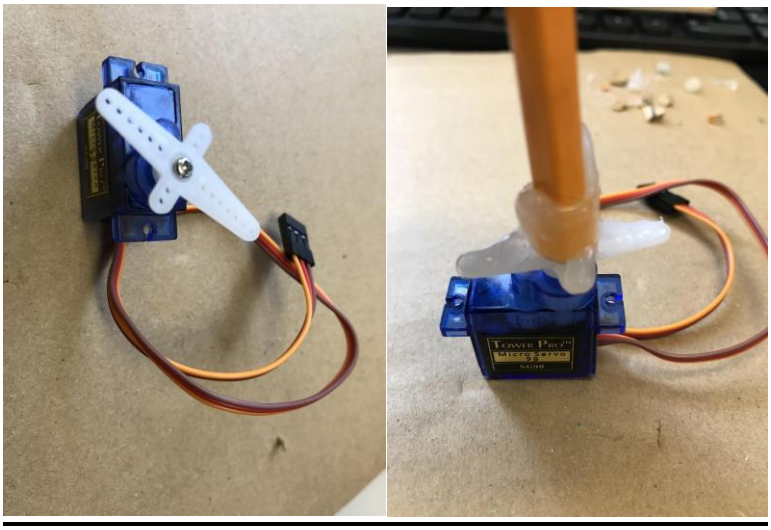
How to Make the Prototype

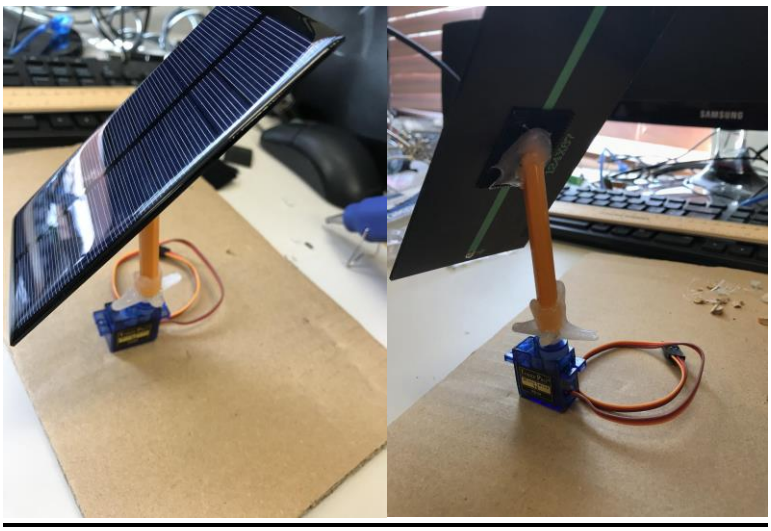
1. Attach one of the rough velcro strips to the center of the back side of the solar panel.



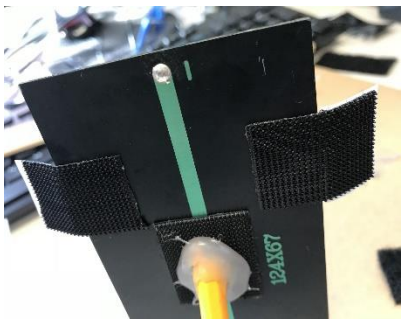
2. Use hot glue to attach the pencil's angled face to the velcro on the solar panel.

3. Assemble the servo and hot glue the other end of the pencil to the top.

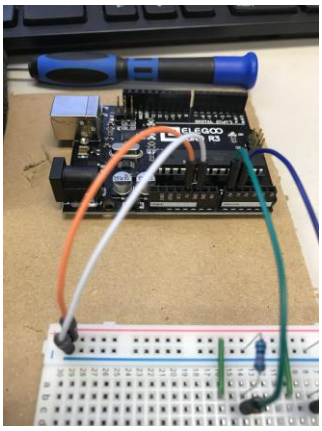




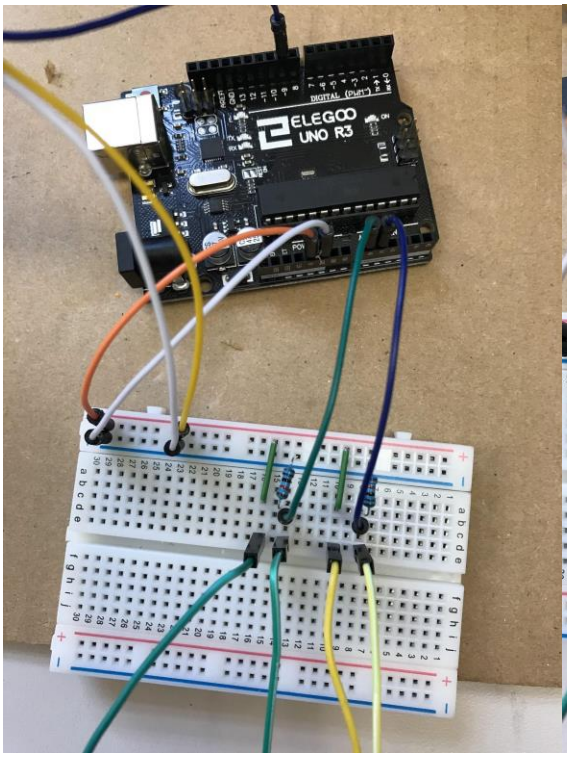
4. Use the other two rough velcro strips to create flaps on either side of the solar panel by attaching them with half of each strip off the edge of the panel. (In the images, the exposed adhesive backs of the velcro strips are covered with paper, but that is not necessary.)

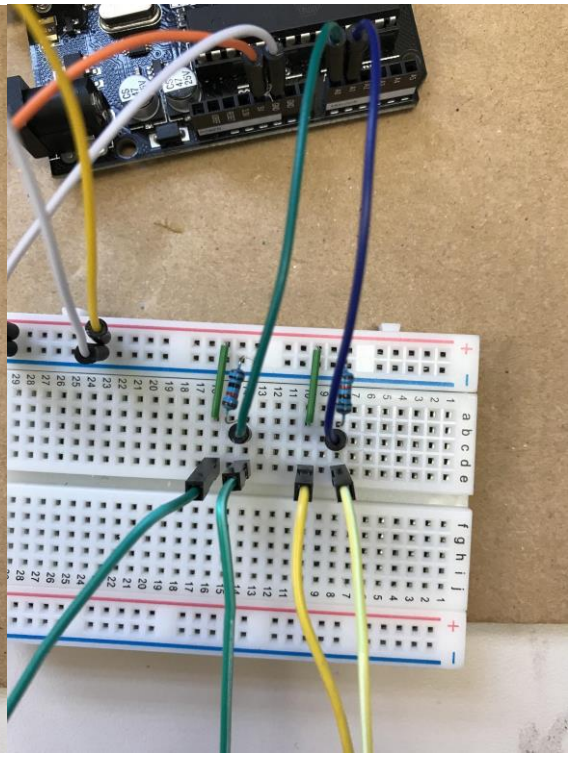


5. Wire the arduino's 5v and GND pins to the breadboard's positive and negative strips (the orange and white wires shown below).
6. Add wires to A0 and A1 analog pins on the arduino (green and blue wires).



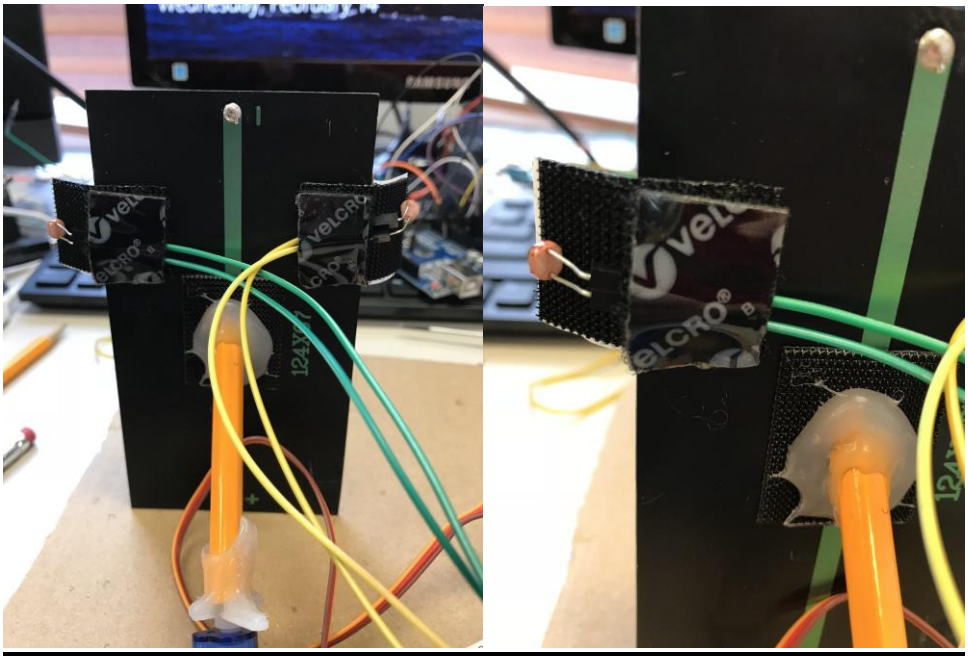
7. Next, add the wires for the left and right photosensors (the pairs of yellow and green wires in the images below). There will be two wires for each sensor. Connect one to the 5v strip and the other will be on the same row as the analog pin's wire with a 10k Ohm resistor connecting it to the GND strip.



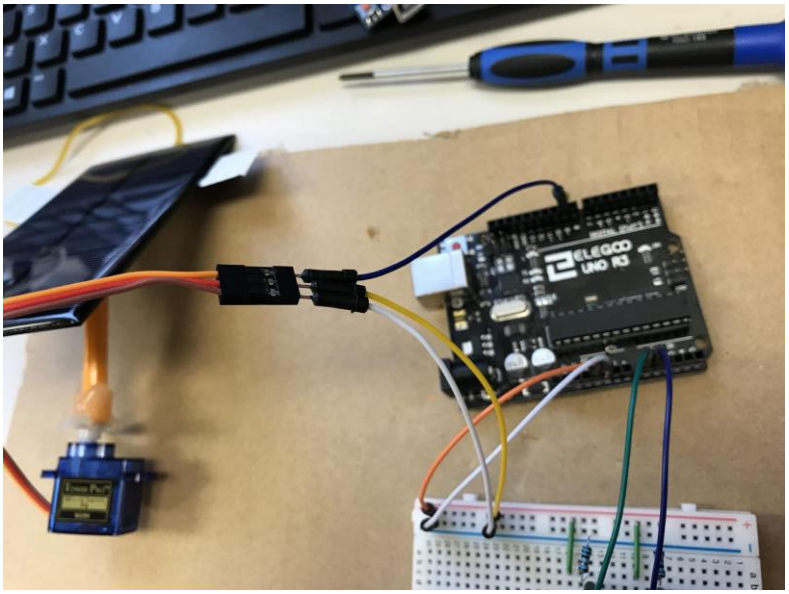


8. Attach a photosensor to each of the pairs of wires added to the breadboard. (The sensor may have trouble staying in the female ends of the wires, so it may help to use some tape here.)

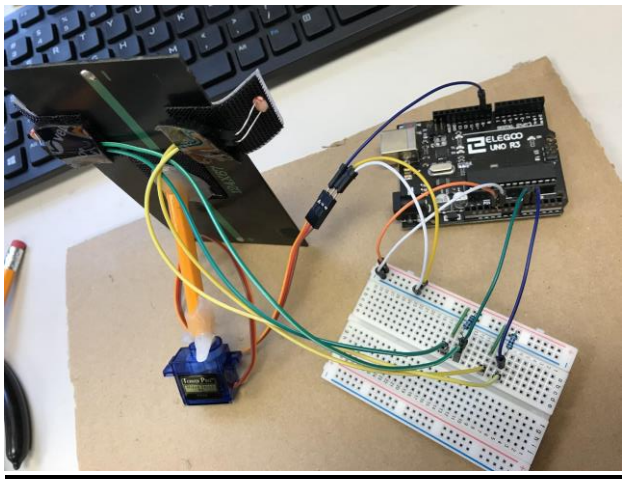
9. Use the two soft velcro strips to fix the sensors behind the flaps on the solar panel.



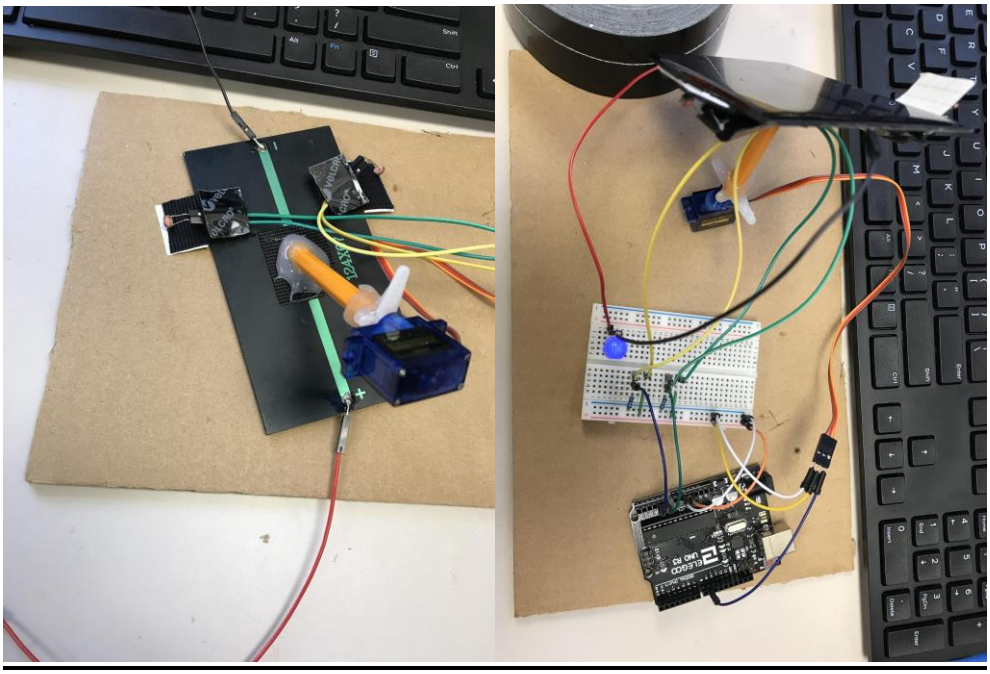
10. Wire the servo to the arduino and breadboard
    a. The dark purple servo wire connects to the GND strip on the breadboard.
    b. The center red wire on the servo connects to the 5v strip on the breadboard.
    c. The orange servo wire connects to the arduino's digital pin 9.



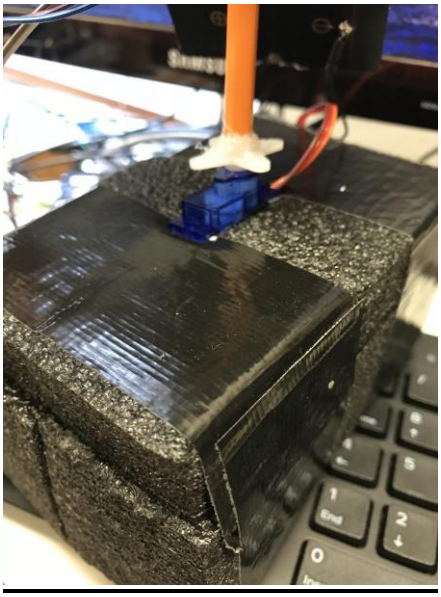
11. Take a look at what we have so far



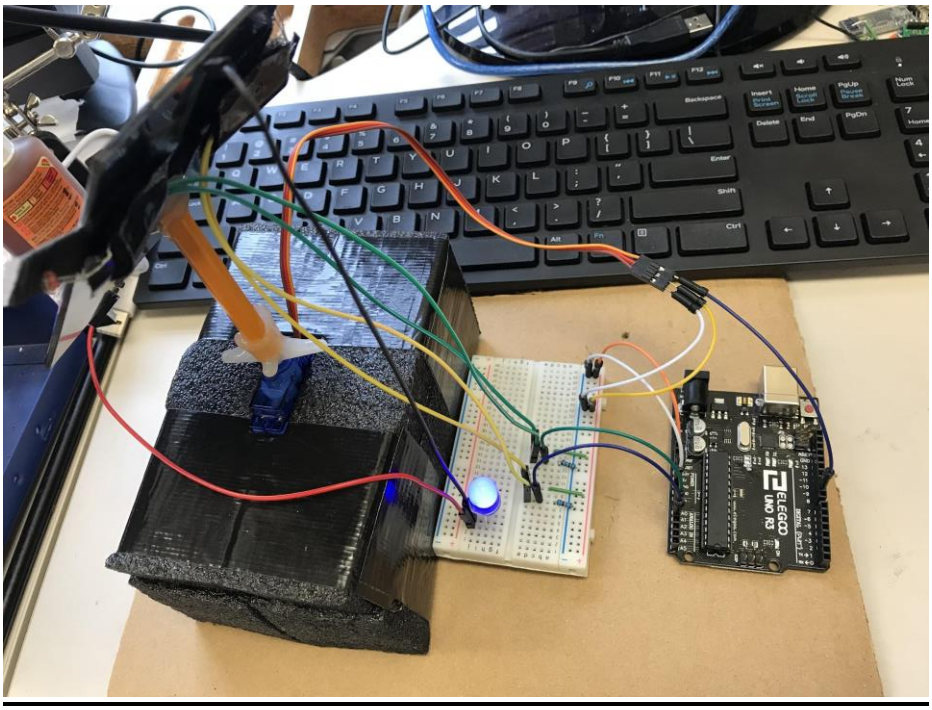
12. Wire the positive and negative solar panel outputs to the LED. The long pin on the LED gets wired to the positive end and the short pin on the LED gets wired to the negative end.



13. Attach the servo to the foam base.



Assembly Complete



The Code

Headers and global variables

```
/*********************************************
                  LIBRARIES - NONE
*********************************************/
#include <Servo.h>

/*********************************************
                  INSTANCES
*********************************************/
// Assigns servo as an object of the 'Servo' class
Servo myservo;

/*********************************************
               DEFINED VARIABLES
*********************************************/
int pos   = 90;    // sets initial position of Servo #1
int LDR_0 = A0;  // LDR (photoresistor) connected to Analog 0
int LDR_1 = A1;  // LDR (photoresistor) connected to Analog 1

// Set 'tolerance' based on how different your LDR readouts are from each other.
// i.e. LDR_0 reads '700' & LDR_1 reads '750' under the same conditions,
// you will need to set a tolerance greater than 50 to adjust for this.
int tolerance = 100;
```

Setup function to initialize serial communication and set the pins

```
/*********************************************
             Void Setup() Function
*********************************************/
void setup() {
 Serial.begin(9600);      // Begin Serial @ 9600 baud
 myservo.attach(9);  // attaches the servo on pin 9 to the servo object


 pinMode(LDR_0, INPUT);   // Sets Analog 0 Pin as INPUT
 pinMode(LDR_1, INPUT);   // Sets Analog 1 Pin as INPUT

 delay(2000); // Intial setup delay
}
```

The loop function runs repeatedly as long as the arduino has power.
First, it reads the input values from the photosensors.

```
/*********************************************
            Void Loop() Function
*********************************************/
void loop() {
  // put your main code here, to run repeatedly:
  int val0 = analogRead(LDR_0);   // Reads the analog input from LDR_0 and stores it in 'val0'
  int val1 = analogRead(LDR_1);   // Reads the analog input from LDR_1 and stores in in 'val1'
```

Next, we print the values through the serial connection.

```
    // Serial Readout of Each Sensor to Compare (@9600 baud)
    Serial.print("[LDR SENSOR A0]: ");
    Serial.print(val0);
    Serial.print("    -    ");
    Serial.print("[LDR SENSOR A1]: ");
    Serial.print(val1);
    Serial.print("    -   ");
```

The program then checks to see if the difference between the values read from the sensors is within the tolerance we set earlier. If it is not, the pos variable is updated.

```
    if((abs(val0 - val1) <= tolerance) || (abs(val1 - val0) <= tolerance)) {
      // This will output if we are within our defined tolerance for error of the LDRs
      Serial.println("*WITHIN TOLERANCE LIMIT*");
      //delay(100);
    }
    else{
      if(val0 > val1){
        Serial.println("*LDR_0 Value > LDR_1 Value*");
        pos = --pos;
      }
      if(val0 < val1){
        Serial.println("*LDR_0 Value < LDR_1 Value*");
        pos = ++pos;
      }
    }
```

Here, the program makes sure that the pos variable is within valid bounds for the servo and resets the value if it is not. Then we move the servo to the new position.

```
90      if(pos > 170) { pos = 170; } // reset to 180 if it goes higher
91      if(pos < 0) { pos = 0; }     // reset to 0 if it goes lower
92      myservo.write(pos);          // write the position to servo
93
94
95      //delay(100); // Brief Delay
96    }
```