The Sun Chaser

"Catch the sun: the Sun Chaser"

Si Un Kim, Laurence Lai, Edwin Chen

Purpose Statement

Sustainability

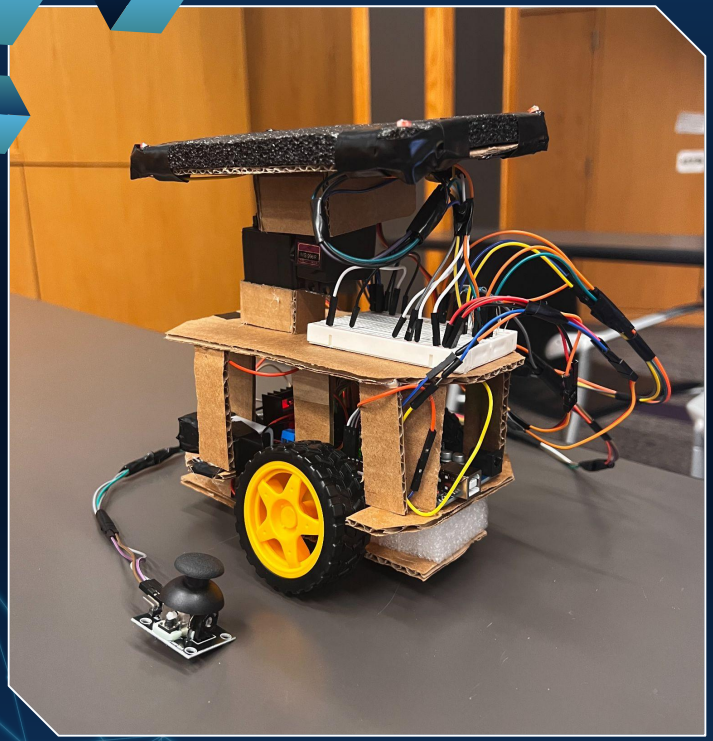
The Sun Chaser helps combat greenhouse gas emission and reduces the collective dependence on fossil fuel by using a solar panel.

Mobility

The Sun Chaser can travel anywhere with just a joystick

Effectiveness

By tilting the solar panel in the direction of the sun, the Sun Chaser captures sunlight with greater efficiency than other solar panels.



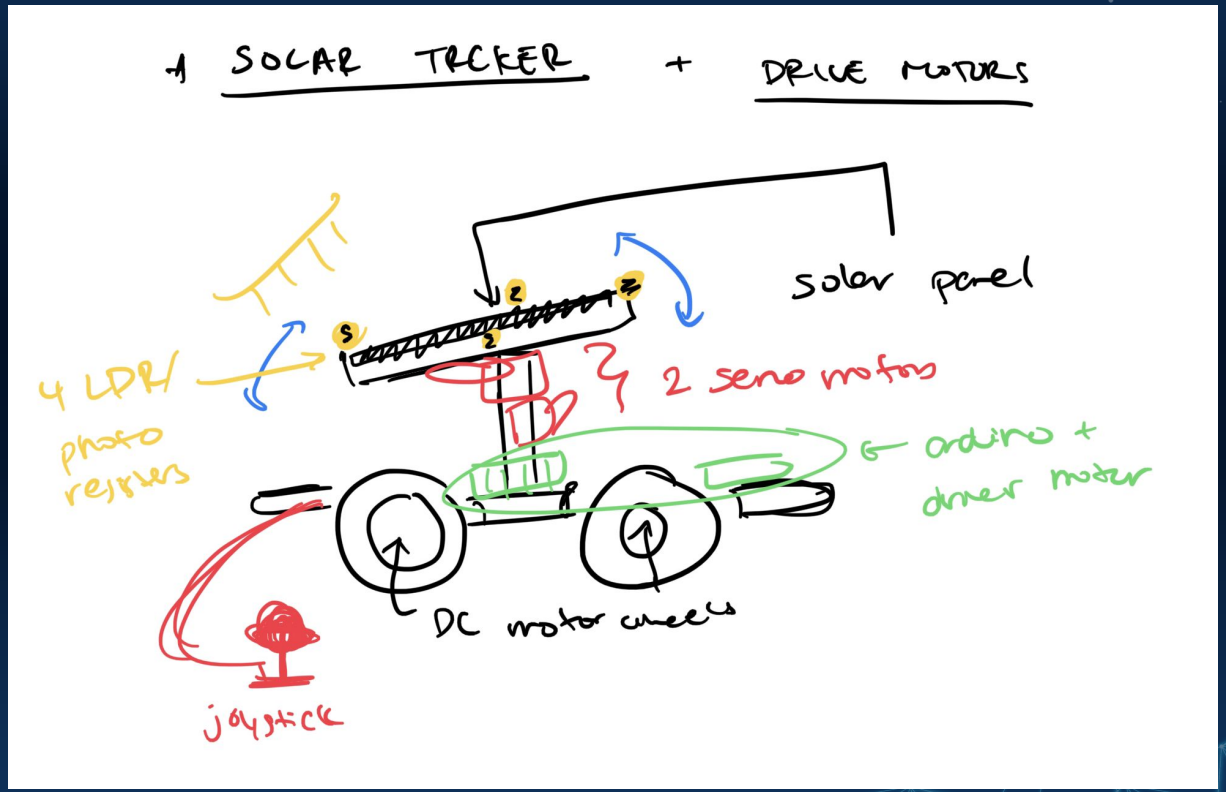
What is Sun Chaser?

Sun Chaser is a semi-autonomous robot that aims to fully optimize the power of renewable solar energy. Equipped with a double axis solar tracker and two robust DC motor wheels, Sun Chaser will bring solar panels to the next level.

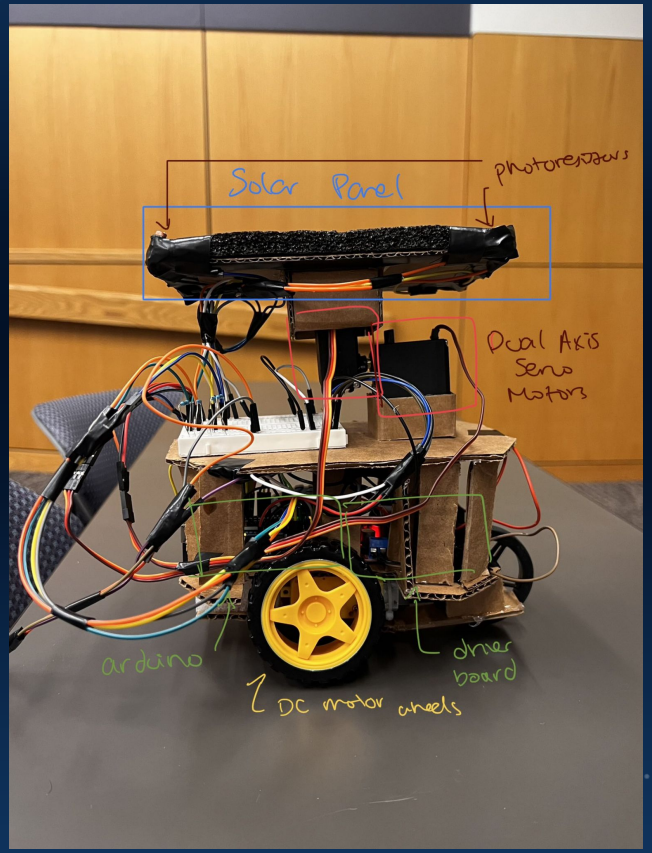
Short Demo Video

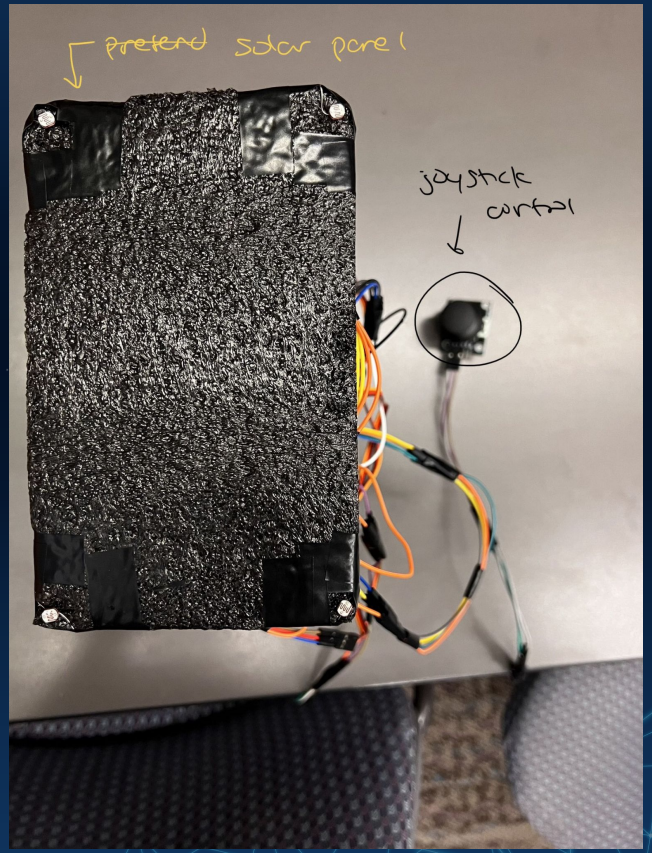
Design

Basic Prototype Drawing



Final Design

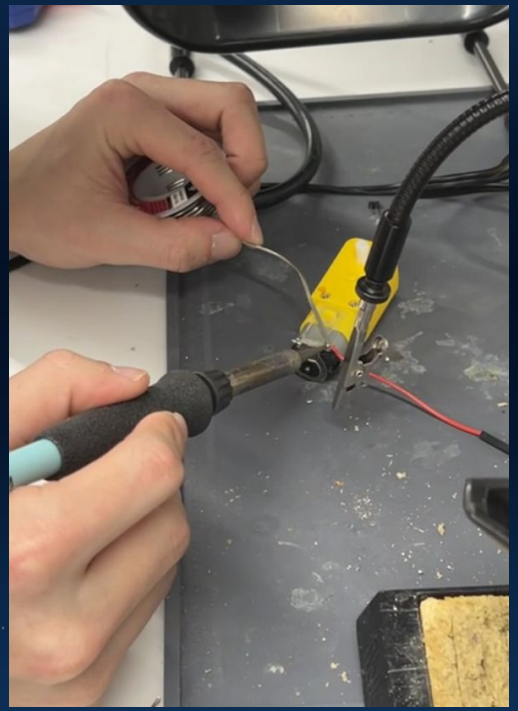


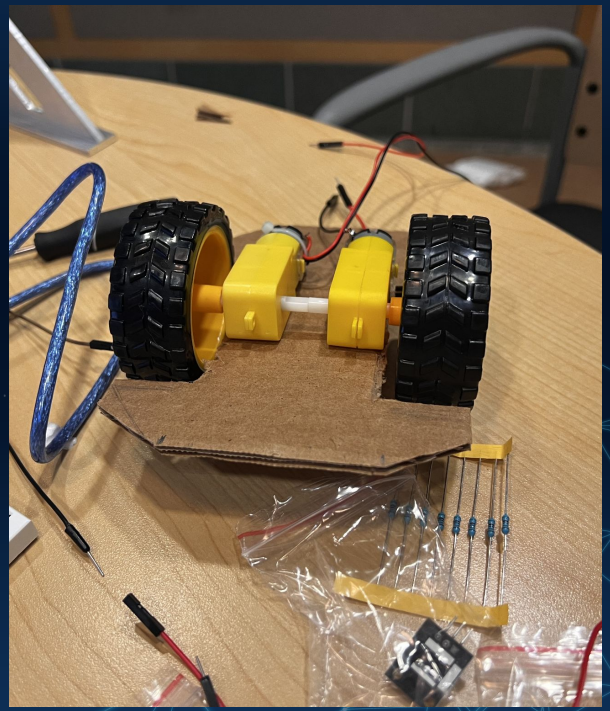


02

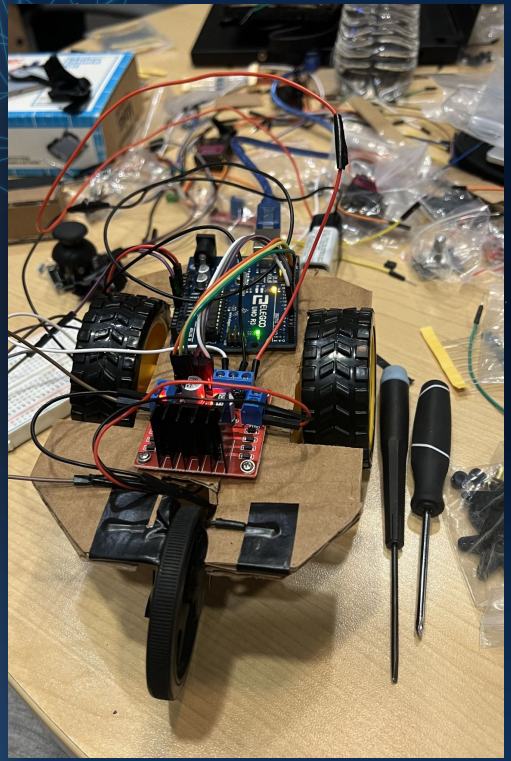
Hardware Building Process

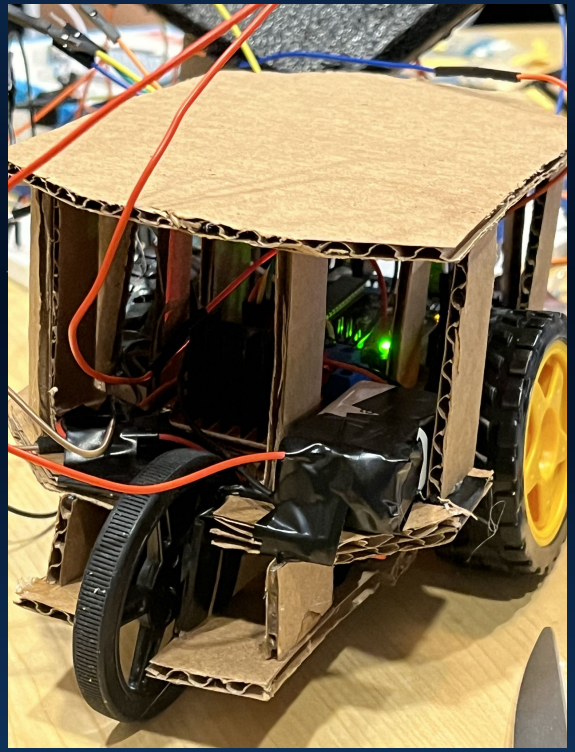
Building Process

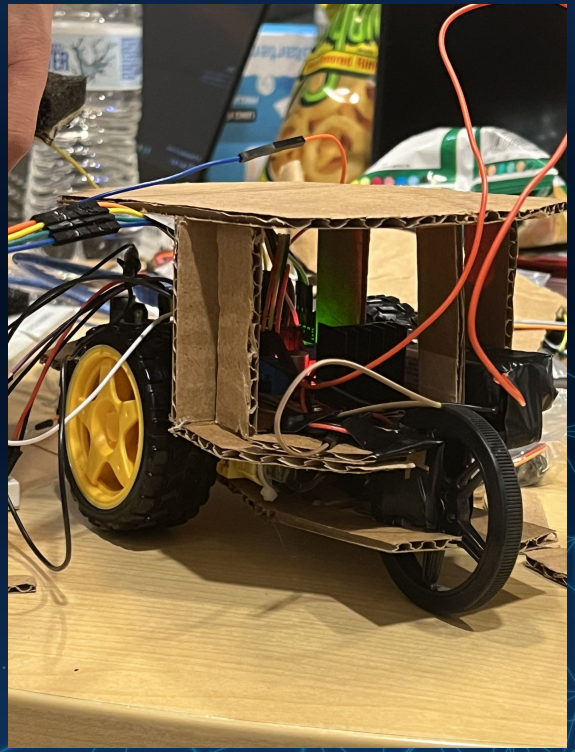




Building Process







03

Arduino Code

General Overview

Split between two functions for both functions of Sun Chaser:

- motorControl() for joystick control of the DC motor wheels
- solarTrack() for solar panel adjustment based off of photoresistors

motorControl() utilizes mapping from the joystick analog values to the dc motors drive values

solarTrack() calculates the average amount of light on each side with photoresistors and utilizes servos to adjust so that each side has a similar amount of light

Definitions

```
#include <Servo.h>

// motor/joystick constants
int enableAPin = 3;
int in1Pin = 4;
int in2Pin = 5;
int enableBPin = 8;
int in4Pin = 9;
int in3Pin = 10;

// solar tracker constants
Servo horizontal; // horizontal servo
int servoh = 90;
int servohLimitHigh = 40;
int servohLimitLow = 150;

Servo vertical; // vertical servo
int servov = 90;
int servovLimitHigh = 60;
int servovLimitLow = 120;

// photoresistor pin connections
int topleft = A2; // yellow
int topright = A1; // orange
int botleft = A3; // green
int botright = A4; // blue
```

- Describes the pins for the motors and joystick
- initiates 2 servo objects and boundaries for dual axis control
- Defines the photoresistors analog pins by location

setup() and loop()

```
void setup() {
  pinMode(enableAPin, OUTPUT);
  pinMode(in1Pin, OUTPUT);
  pinMode(in2Pin, OUTPUT);

  horizontal.attach(12);
  vertical.attach(13);

  Serial.begin(9600);
}
```

- Assigns pins to outputs or inputs
- Attaches servo pins
- Begins serial monitor

```
void loop() {
  motorControl();

  // intiate servos
  horizontal.write(90);
  vertical.write(90);

  solarTrack();

  // debugging photoresistors
  // int tlVal = analogRead(topleft);
  // int trVal = analogRead(topright);
  // int blVal = analogRead(botleft);
  // int brVal = analogRead(botright);

  // Serial.println(tlVal);
  // Serial.println(trVal);
  // Serial.println(blVal);
  // Serial.println(brVal);
  // Serial.println(" ");
}
```

- motorControl() and solarTrack() functions - next slides
- Serial print debugging for LDRs/photoresistors

motorControl()

```
void motorControl() {
  int motorPWMSpeed = 0;
  int joystickValue = analogRead(A0);   //Joystick gives values ranging from 0 to 1023.

  if (joystickValue >= 530)           //This will move motor in forward direction
  {
    motorPWMSpeed = map(joystickValue, 530, 1023, 0, 255);
    digitalWrite(in1Pin, HIGH);
    digitalWrite(in2Pin, LOW);
    analogWrite(enableAPin, motorPWMSpeed);

    digitalWrite(in4Pin, HIGH);
    digitalWrite(in3Pin, LOW);
    analogWrite(enableBPin, motorPWMSpeed);
  }
  else if (joystickValue <= 490)      //This will move motor in reverse direction
  {
    motorPWMSpeed = map(joystickValue, 490, 0, 0, 255);
    digitalWrite(in1Pin, LOW);
    digitalWrite(in2Pin, HIGH);
    analogWrite(enableAPin, motorPWMSpeed);

    digitalWrite(in4Pin, LOW);
    digitalWrite(in3Pin, HIGH);
    analogWrite(enableBPin, motorPWMSpeed);
  }
  else                                //Stop the motor
  {
    digitalWrite(in1Pin, LOW);
    digitalWrite(in2Pin, LOW);
    digitalWrite(in4Pin, LOW);
    digitalWrite(in3Pin, LOW);
  }
}
```

Maps joystick north direction to forward direction on motor

Maps joystick south direction to reverse direction on motor

Does nothing on idle state

solarTrack()

```
void solarTrack() {
  int lt = analogRead(topleft); // top left
  int rt = analogRead(topright); // top right
  int ld = analogRead(botleft); // down left
  int rd = analogRead(botright); // down right
  int dtime = 50; int tol = 90; // dtime=diffirence time, tol=toleransi
  int avt = (lt + rt) / 2; // average value top
  int avd = (ld + rd) / 2; // average value down
  int avl = (lt + ld) / 2; // average value left
  int avr = (rt + rd) / 2; // average value right
  int dvert = avt - avd; // check the diffirence of up and down
  int dhoriz = avl - avr;// check the diffirence og left and rigt

  if (-1*tol > dvert || dvert > tol) {
    if (avt > avd) {
      servov = ++servov;
    if (servov > servovLimitHigh)
      {servov = servovLimitHigh;}
    }
  else if (avt < avd)
    {servov= --servov;
    if (servov < servovLimitLow)
      { servov = servovLimitLow;}
    }
    vertical.write(servov);
  }
```

Defines local variables for solarTrack - LDR analog values, average value on each side, difference between sides

Vertical servo adjusts for light differences on vertical sides (top and bottom) within a tolerance

solarTrack() - continued

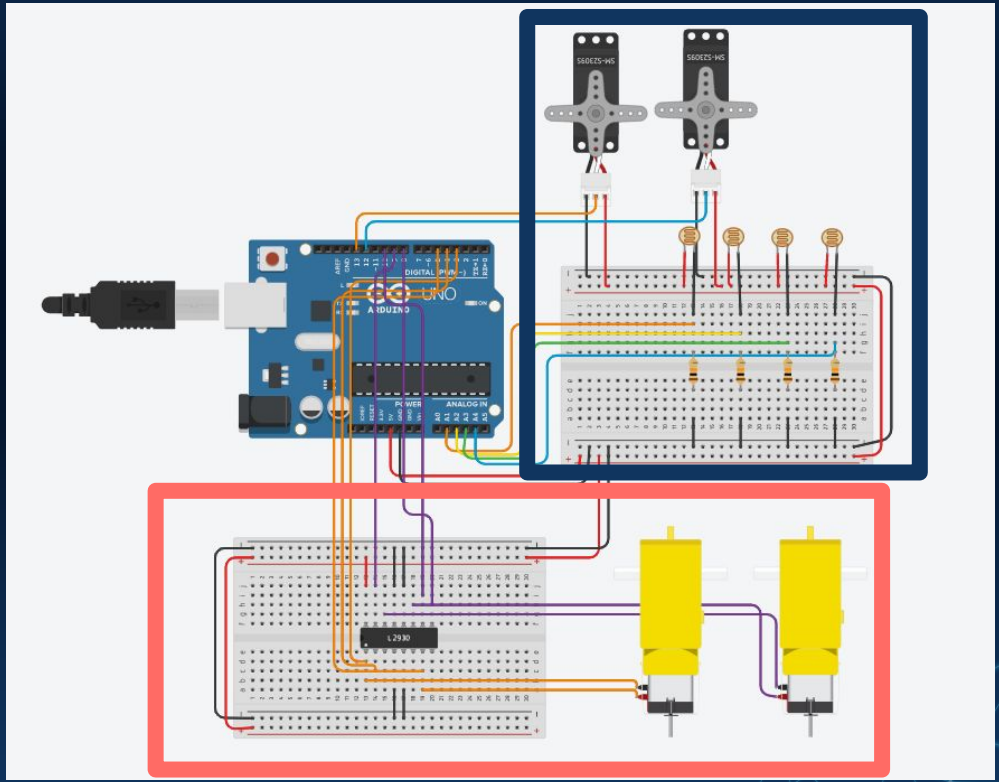
```
if (-1*tol > dhoriz || dhoriz > tol) // check if th
{
  if (avl > avr) {
    servoh = ++servoh;
    delay(20);
    if (servoh < servohLimitLow) {
      servoh = servohLimitLow;
    }
  }
  else if (avl < avr) {
    servoh = --servoh;
    delay(20);
    if (servoh > servohLimitHigh) {
      servoh = servohLimitHigh;
    }
  }
  else if (avl = avr) {
    delay(5000);
    }
  horizontal.write(servoh);
  }
delay(dtime);
}
```

Horizontal servo adjusts for light differences on horizontal sides (left and right) within a tolerance

04

Circuit Diagram

Circuit Diagram



05

Final Product!

Final Product!