Amped Up Solar Panels

A Machine Learning Approach to Increase the Electricity Generated by a Solar Panel Using a Dual-Axis Solar Tracker

Table of Contents

Introduction

Solar energy is one of the most widely used sources of alternative energy. Although new advancements have made solar panels more efficient and cost-effective, they have not been used to their full potential, so fossil fuels remain the remain the largest source of electricity. Tilting the panel towards the Sun is a simple way to ensure maximum sunlight reception throughout the day. The purpose of our project is to maximize the amount of electricity generated by solar panels using a smart dual-axis solar tracker. By using machine learning to predict solar output and the angle the Sun, we can calculate and decide the optimal times and angles to tilt thereby increasing the net generation of the solar panel and lowering the energy consumption of robot.

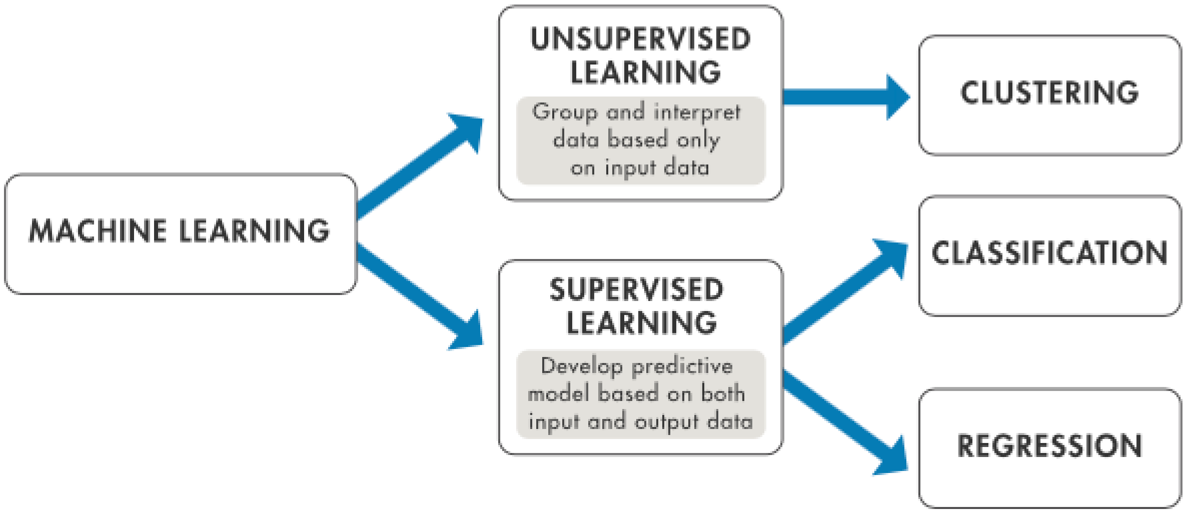
Engineering Goal and Summary

Research Question:
How can we leverage machine learning and mathematical models to optimize the electricity generated by a solar panel using a solar tracker?

Engineering Goal:
The goal for the dual-axis solar tracker is to increase the net generation of electricity by improving the accuracy of the tilt angle and optimizing consumption with machine learning. The expectation is the electricity generated, measured in milliamp hours, by the dual-axis solar tracker will be double the amount generated by a flat solar panel.

Background Research

Machine learning uses two types of techniques - supervised and unsupervised learning. Supervised learning makes a prediction based on evidence of uncertainty whereas unsupervised learning finds hidden patterns


MACHINE LEARNING
UNSUPERVISED LEARNING
Group and interpret data based only on input data
SUPERVISED LEARNING
Develop predictive model based on both input and output data
CLUSTERING
CLASSIFICATION
REGRESSION

and draws information from data sets.

Supervised learning can be further split into classification and regression. Classification techniques predict responses and classify that input data into categories. Regression techniques predict continuous responses like temperature changes. This technique involves data in a range or if the data is numerical. In unsupervised learning, clustering is used to find hidden patterns or groupings in data. It is used for things like gene sequence analysis, market research, and object recognition. Companies that decided where to place cell phone towers can use this to identify the number of clusters of people relying on their towers. Since a phone can only communicate with one tower at a time, clustering is used to find the optimal place with the best reception for their customers.

Neural networks are modeled after the human brain with thousands of nodes that data moves forward through. It is organized into layers where it sends data up the chain. Each of the incoming connections is given a number known as a weight. When the network is active, the node receives a number over each of the connections and multiplies it by the weight. It adds the products together resulting in a single number. If this number is below the threshold value, then it does not pass on the data to the next layer. Otherwise, the node sends the data along all outgoing connections. On a large-scale neural network is given training examples from which they develop a system to learn from. For example, given a set of handwritten numbers, pattern recognition is used to “train” the computer to identify certain shapes to certain number values. An artificial neuron is called a perceptron which takes several binary inputs and outputs a single binary output. To compute the output weights are used- which are real numbers that show the importance of each input to output. For example, if you are trying to decide if you should go to a fireworks festival, you make your decision of these 3 factors. Is the weather good? Do I have transportation? Are my friends accompanying me? However, having transportation to the place might be the most important to you so you would give that the highest weight. The neuron outputs a 0 or 1 which is determined by the sum of the weights of the activated nodes. If it is less than or equal to the threshold, then the computer outputs 0. If it is greater than the threshold, then the computer outputs 1 and sends the data forward.

$$\text{output} = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq \text{ threshold} \\ 1 & \text{if } \sum_j w_j x_j > \text{ threshold} \end{cases}$$

Linear regression is the model used to create a linear relationship between many inputs (x) and one output (y). The basic equation is listed below.
y = B0 + B1*x

The linear regression equation assigns one scale factor, called a coefficient, to each input value. Another coefficient allows the line to move up or down the graph which is known as the bias coefficient. If the coefficient is 0 then it removes the significance of the variable in the regression model. The ordinary least squares procedure can be used to determine the value of the coefficients. Error is the difference from the data point to the line of regression, and OLS procedure deals with the sum of square of the errors. The goal is to find the regression line from the data that minimizes the sum of the squared errors. By treating the data as a matrix, this procedure is able to find the value of the coefficients. Gradient descent is another method which takes random starting values and uses a learning rate as the scale factor to change the coefficient in a direction towards a minimal error. Using this method, it is able to find the minimal sum squared error.

Step wise regression is used to assess the significance of a variable on the linear regression model. This method adds or removes variables to see which variables have the greatest statistical significance on the model. Step wise regression uses a process called backwards elimination. This process assigns a significance level (0.05) to each variable and inputs it into the model. The p value of each coefficient is compared with each other to find the biggest value. Take this value out and repeat this process. At the end you will be left with p values of coefficients that are the closest to the significance level. This indicates the variables that will make the most impact on the machine learning predictor.

Gradient boosting is a machine learning technique for regression and classification problems. It produces a prediction model in the form of ensemble decision trees. It combines weak learners into a single strong learner in a iterative manner. It is efficient in reducing the mean-squares error like OLS.

If a flat panel system is used, the sunlight incident will be along the surface in the morning and evening. The panels would have to be tilted manually to account for seasonal changes. The annual optimal tilt angle is equal to the geographic location for India. Compared to a monthly tilt panel, the loss is 15% of energy when using a flat system. According to studies in Turkey, there was a 64% greater amount of energy generated in the months of June and July using a 2-axis tilt. Measuring improvement depends on the relative amount of direct and infuse radiation. A photoelectric cell uses radiation to generate an output of electrical current. The photoelectric effect is the process where electromagnetic radiation strikes a substance which causes and electron to move from ground state to a higher, excited state. Selenium is one such metal that is very reactive to radiation as it caused it to release electrons. These electrons flow as current, and therefore produce electricity.

A solar panel works by allowing photons, or particles of light, to knock electrons free from atoms, generating a flow of electricity. Solar panels actually comprise many, smaller units called photovoltaic cells. Photovoltaic cells convert sunlight into electricity, these are also called solar cells. Many cells linked together make up a solar panel. Each photovoltaic cell is basically a sandwich made up of 2 slices of silicon, a semiconductor. Photovoltaic cells need to establish a electric field in order to work. Much like a magnetic field, which occurs due to opposite poles, an electric field occurs when opposite charges are separated. To get this field, manufacturers combine silicon with other materials, giving each slice of the sandwich a positive or negative electrical charge. Specifically, they seed phosphorus into the top layer of silicon, which adds extra electrons, with a negative charge, to that layer. The bottom layer gets boron, which results in fewer electrons, or a positive charge. This goes to an electric field between the silicon layers. When a photon of sunlight knocks an electron free, the electric field will push that electron out of the silicon junction.

Solar energy offsets more than 74 million metric tons of carbon dioxide emissions, which is equivalent to taking 15.8 million vehicles of the road and planting 1.9 billion trees. In just United States, there are 1.8 million solar installations and researchers predict that number will go to 2 million in late 2018 and 4 million in 2023. Solar panels are highly effective and it is free to use, but the cost in making and providing the system of collection is much higher. In the Texas, the average cost of a 5kw solar panel system is $3.68 per watt. The levelized cost of solar is 6¢ per kWh compared to 25¢ per kWh as the average utility price. This allows you to get your money back in 9 years and 1 month. The annual power production is 10,405 kWh which is not that significant. 9 years and 1 month seems like a long time so solar panels nowadays may not be very efficient as they should be.

Using past sunrise and sunset data, a sine function can be used to model the number of daylight hours in a location. For this example, the location is in San Diego, California. The equation is: $H(t) = 2.4 \sin(0.017t - 1.377) + 12$.

t is the day of the year (ranges from 1 to 365)

$H(t)$ is the number of hours of daylight at day t

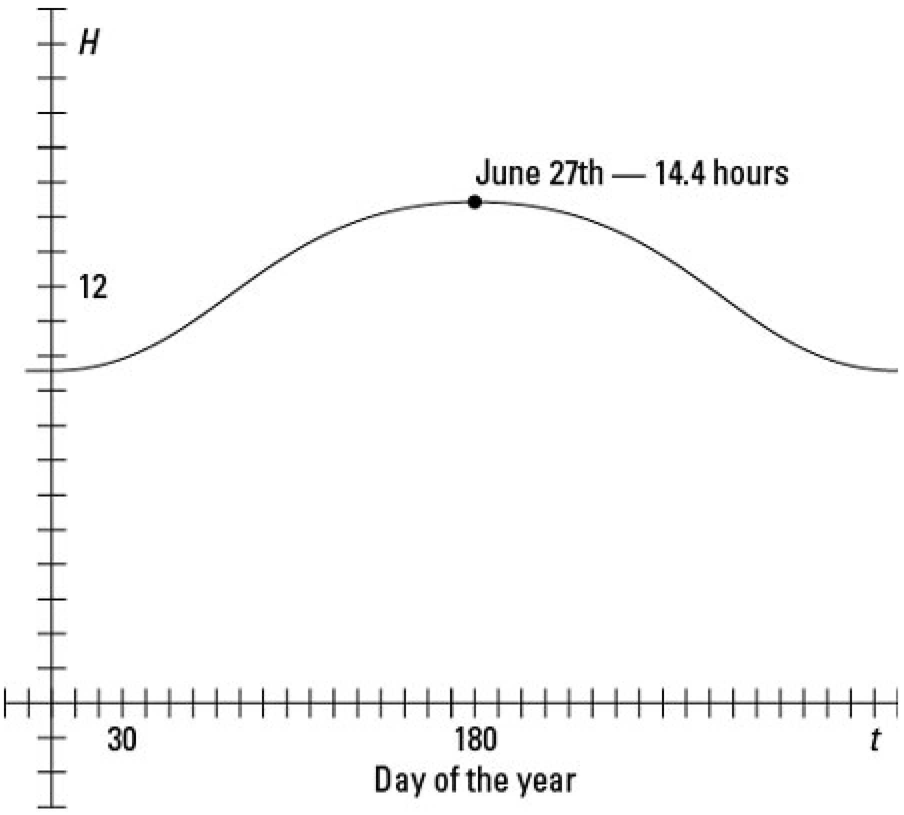
To calculate the minimum and maximum:

Find the derivative: $h'(t) = 0.0408 \cos(0.017t - 1.377)$

$h'(t) = 0$, when $t = 173, 358$

Maximum of $H(t)$: $t = 173$, $H(t) = 14.4$ hours

Minimum $H(t)$: $t = 358$, $H(t) = 9.6$ hours



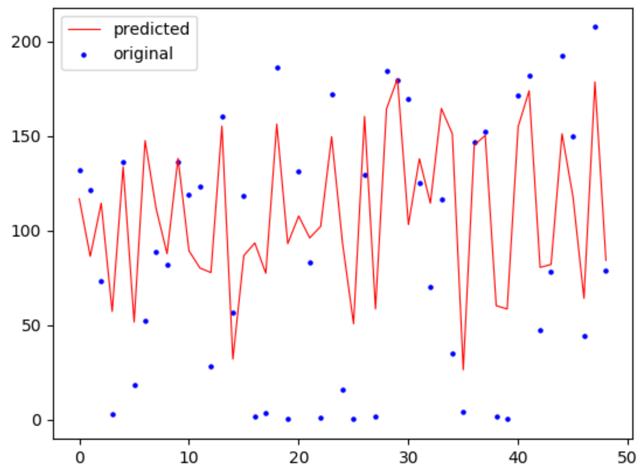
Background Vocab

Kilowatt hour	A measure of electrical energy equivalent to a power consumption of 1,000 watts for 1 hour
Photovoltaic cell	Smaller units in solar panels that convert sunlight into electricity
Watt	Unit of electrical power, equal to 1 joule per second
Silicon	Semiconductor used in many electronics
Electron	Subatomic particle with a negative charge
Direct current (DC)	Electric current flowing in only one direction
Ampere	Unit of electrical current, equal to a flow of one coulomb per second
Solar panel tilt	The degree of the tilt of the solar panel so that it gets the optimum amount of sunlight
Electricity	Form of energy resulting from the existence of charged particles
Solar energy	Power obtained harnessing the energy of the sun’s rays
Solar panel	Panel designed to absorb the sun’s rays as a source of electricity
Green energy	Energy that comes from renewable resources, such as sunlight
Ammeter	Instrument for measuring electrical current in amperes
Circuit	System composed of resistors, transistors, capacitors, conductors, inductors, and diodes through which electrical current can flow
Sunlight	Light from the sun
Sunrise & Sunset	The instant when the upper limb of the Sun’s disk is just touching the horizon

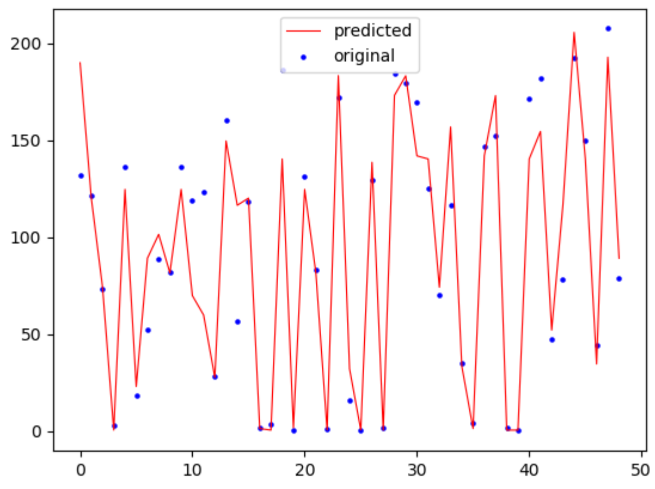
Azimuth	Angular distance from the north or south point of the horizon to the point at which a vertical circle passing through the object intersects the horizon
Zenith	The point on the celestial sphere directly overhead the observer. It has an altitude of +90° in the horizontal coordinate system
Cartesian Coordinates	Solar elevation is plotted on x-axis and the azimuth is plotted on the y-axis
Polar Coordinates	Point in space as an angle of rotation around the origin and a radius from the origin
Solar Altitude	The angle of the sun relative to the Earth's horizon
Latitude	Angular distance north or south from the equator of a point on the earth's surface
Longitude	Angular distance of a place east or west of the prime meridian
Solar Noon	The moment in a day when the sun reaches its highest position above the horizon on that day
Ordinary Least-Squares	OLS minimizes the residual sum of squares between the observed targets in the dataset, and the targets predicted by the linear approximation.
Gradient Boosting	GB builds an additive model in a forward stage-wise fashion; it allows for the optimization of arbitrary differentiable loss functions.

Machine Learning Model Accuracy Graphs

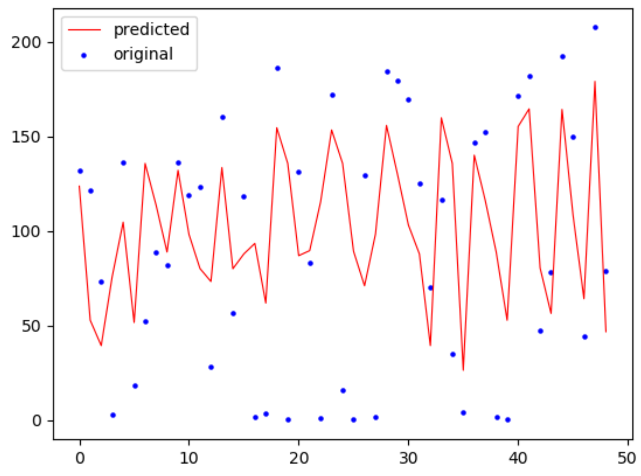
K Nearest Neighbors



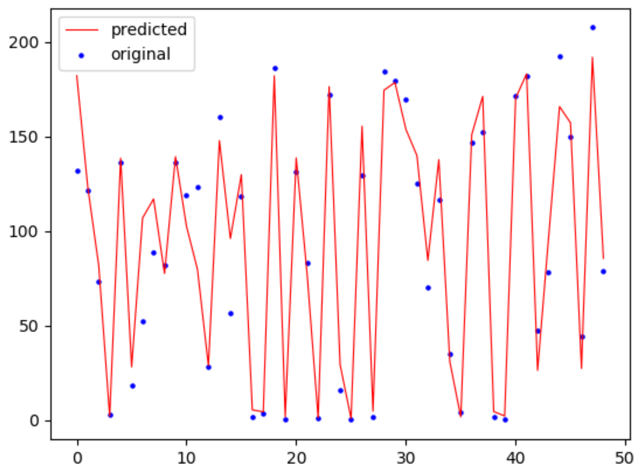
Decision Trees



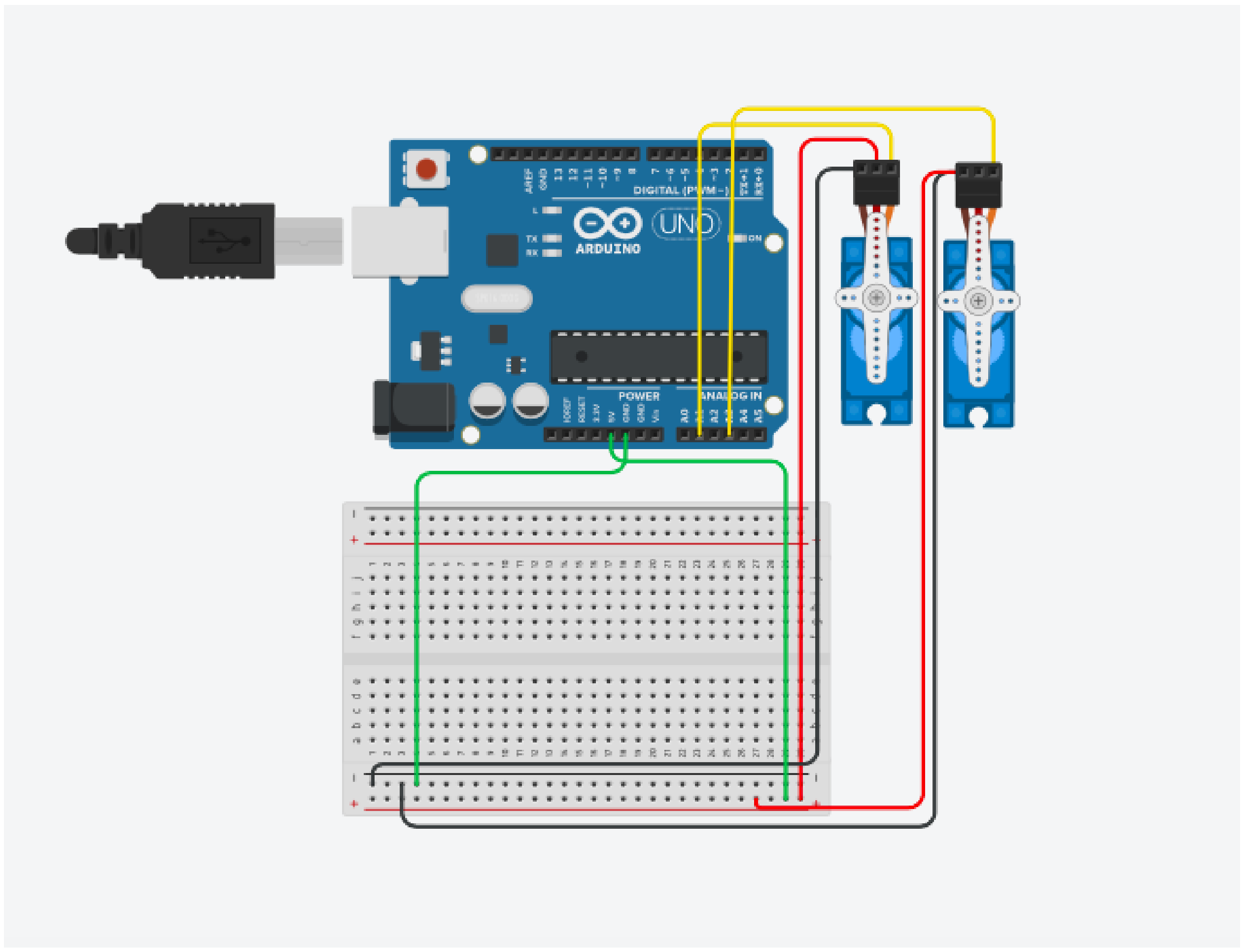
Ordinary Least Squares



Gradient Boosting Regression

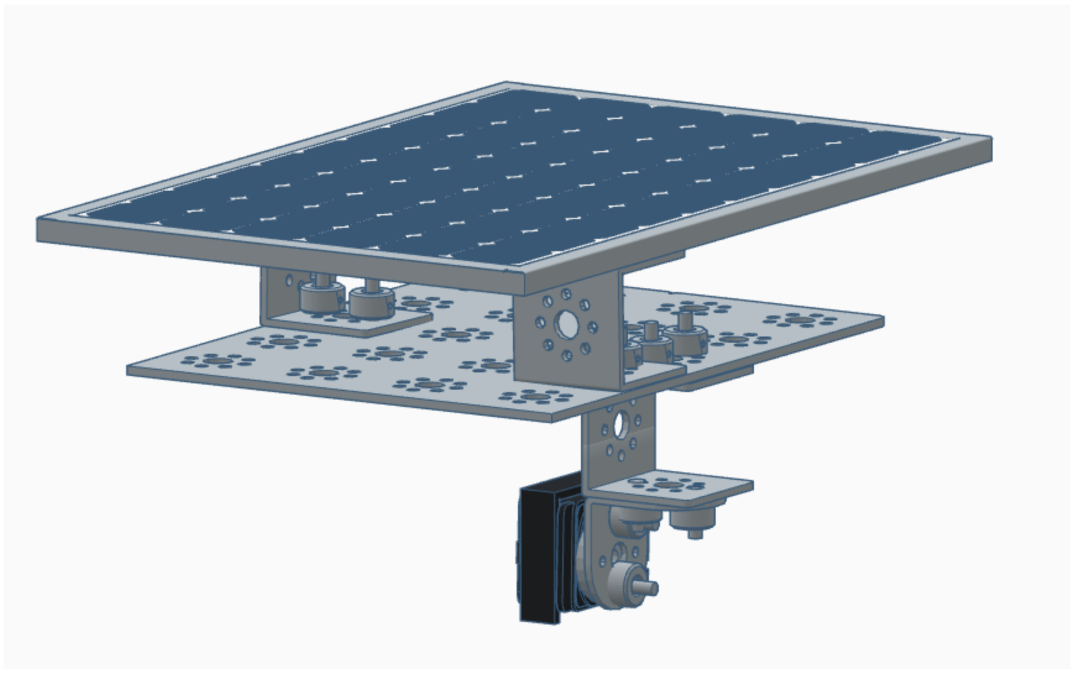


Circuit Diagram

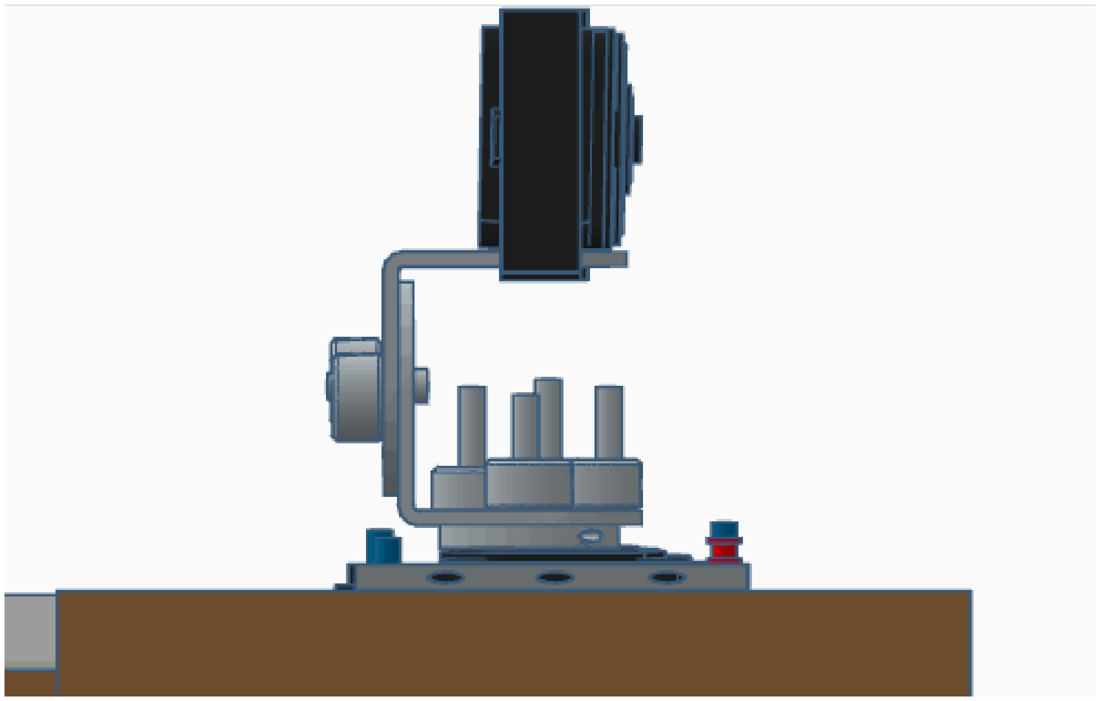


Screenshot taken by Experimenter B on 11/24/20

Dual-Axis Solar Panel Construction Method



Screenshot taken by Experimenter B on 11/9/20



Screenshot taken by Experimenter B on 11/9/20

Innovation

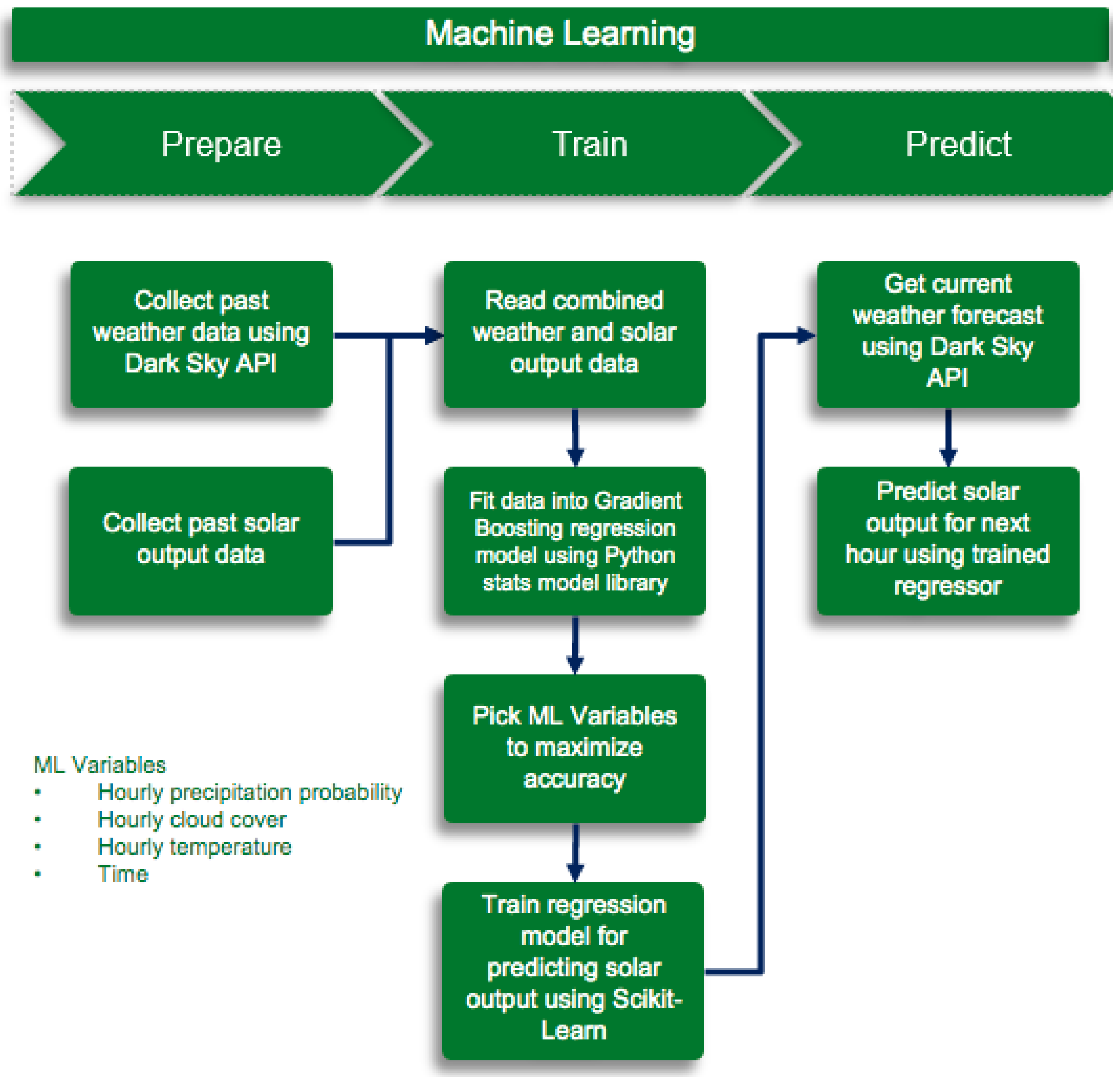
Iteration 1

We initially used OLS regression equation to predict solar output since it was the most straightforward. We tried algorithms in 4 different categories: Linear Regression, Nearest Neighbors, Decision Trees, and Ensemble Methods. We realized that the accuracy of the Gradient Boosting Decision Tree, an ensemble method, produced the highest accuracy result, so we decided to switch our regression algorithm. OLS had a 52% accuracy while gradient boosting decision trees had a 92% accuracy rate.

Iteration 2

The implementation of a gyro sensor gave the robot access to a wider range of capabilities to be positioned on any type of terrain. This sensor allowed us to calculate the angle of the surface to take into account during the calculation of our tilt. For example, a robot placed on a slanted roof will have to compensate for the surface angle in comparison to a robot resting on level ground.

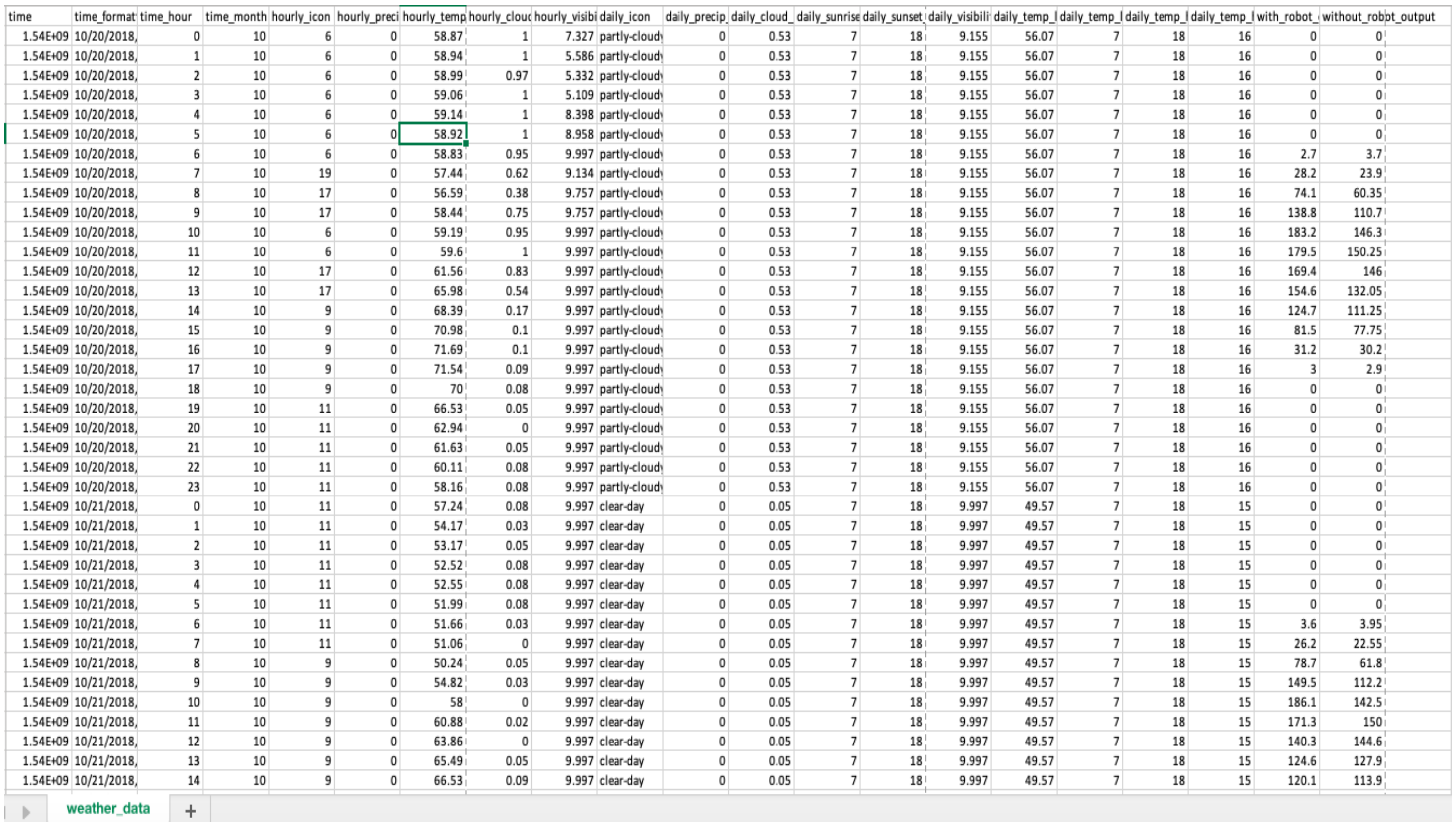
Machine Learning Program Flowchart



Machine Learning
Prepare
Train
Predict
Collect past weather data using Dark Sky API
Collect past solar output data
Read combined weather and solar output data
Fit data into Gradient Boosting regression model using Python stats model library
Pick ML Variables to maximize accuracy
Train regression model for predicting solar output using Scikit-Learn
Get current weather forecast using Dark Sky API
Predict solar output for next hour using trained regressor
ML Variables
• Hourly precipitation probability
• Hourly cloud cover
• Hourly temperature
• Time

Screenshot taken by Experimenter A on 11/24/20, made using draw.io

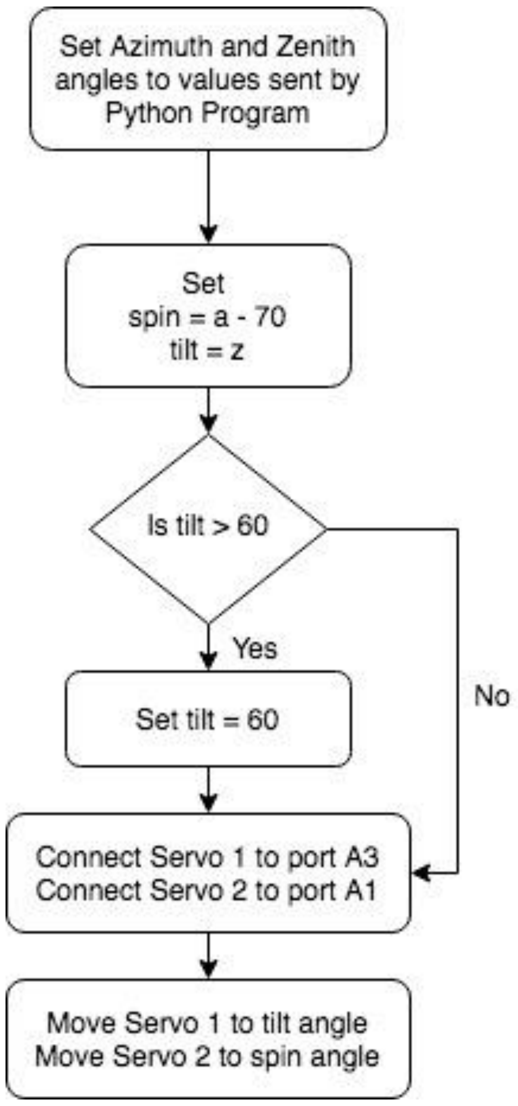
Sample CSV



time	time_format	time_hour	time_month	hourly_icon	hourly_preci	hourly_temp	hourly_clouc	hourly_visibi	daily_icon	daily_precip_	daily_cloud_	daily_sunrise	daily_sunset_	daily_visibili	daily_temp_l	daily_temp_l	daily_temp_l	daily_temp_l	with_robot_	without_robot_output
1.54E+09	10/20/2018,	0	10	6	0	58.87	1	7.327	partly-cloudy	0	0.53	7	18	9.155	56.07	7	18	16	0	0
1.54E+09	10/20/2018,	1	10	6	0	58.94	1	5.586	partly-cloudy	0	0.53	7	18	9.155	56.07	7	18	16	0	0
1.54E+09	10/20/2018,	2	10	6	0	58.99	0.97	5.332	partly-cloudy	0	0.53	7	18	9.155	56.07	7	18	16	0	0
1.54E+09	10/20/2018,	3	10	6	0	59.06	1	5.109	partly-cloudy	0	0.53	7	18	9.155	56.07	7	18	16	0	0
1.54E+09	10/20/2018,	4	10	6	0	59.14	1	8.398	partly-cloudy	0	0.53	7	18	9.155	56.07	7	18	16	0	0
1.54E+09	10/20/2018,	5	10	6	0	58.92	1	8.958	partly-cloudy	0	0.53	7	18	9.155	56.07	7	18	16	0	0
1.54E+09	10/20/2018,	6	10	6	0	58.83	0.95	9.997	partly-cloudy	0	0.53	7	18	9.155	56.07	7	18	16	2.7	3.7
1.54E+09	10/20/2018,	7	10	19	0	57.44	0.62	9.134	partly-cloudy	0	0.53	7	18	9.155	56.07	7	18	16	28.2	23.9
1.54E+09	10/20/2018,	8	10	17	0	56.59	0.38	9.757	partly-cloudy	0	0.53	7	18	9.155	56.07	7	18	16	74.1	60.35
1.54E+09	10/20/2018,	9	10	17	0	58.44	0.75	9.757	partly-cloudy	0	0.53	7	18	9.155	56.07	7	18	16	138.8	110.7
1.54E+09	10/20/2018,	10	10	6	0	59.19	0.95	9.997	partly-cloudy	0	0.53	7	18	9.155	56.07	7	18	16	183.2	146.3
1.54E+09	10/20/2018,	11	10	6	0	59.6	1	9.997	partly-cloudy	0	0.53	7	18	9.155	56.07	7	18	16	179.5	150.25
1.54E+09	10/20/2018,	12	10	17	0	61.56	0.83	9.997	partly-cloudy	0	0.53	7	18	9.155	56.07	7	18	16	169.4	146
1.54E+09	10/20/2018,	13	10	17	0	65.98	0.54	9.997	partly-cloudy	0	0.53	7	18	9.155	56.07	7	18	16	154.6	132.05
1.54E+09	10/20/2018,	14	10	9	0	68.39	0.17	9.997	partly-cloudy	0	0.53	7	18	9.155	56.07	7	18	16	124.7	111.25
1.54E+09	10/20/2018,	15	10	9	0	70.98	0.1	9.997	partly-cloudy	0	0.53	7	18	9.155	56.07	7	18	16	81.5	77.75
1.54E+09	10/20/2018,	16	10	9	0	71.69	0.1	9.997	partly-cloudy	0	0.53	7	18	9.155	56.07	7	18	16	31.2	30.2
1.54E+09	10/20/2018,	17	10	9	0	71.54	0.09	9.997	partly-cloudy	0	0.53	7	18	9.155	56.07	7	18	16	3	2.9
1.54E+09	10/20/2018,	18	10	9	0	70	0.08	9.997	partly-cloudy	0	0.53	7	18	9.155	56.07	7	18	16	0	0
1.54E+09	10/20/2018,	19	10	11	0	66.53	0.05	9.997	partly-cloudy	0	0.53	7	18	9.155	56.07	7	18	16	0	0
1.54E+09	10/20/2018,	20	10	11	0	62.94	0	9.997	partly-cloudy	0	0.53	7	18	9.155	56.07	7	18	16	0	0
1.54E+09	10/20/2018,	21	10	11	0	61.63	0.05	9.997	partly-cloudy	0	0.53	7	18	9.155	56.07	7	18	16	0	0
1.54E+09	10/20/2018,	22	10	11	0	60.11	0.08	9.997	partly-cloudy	0	0.53	7	18	9.155	56.07	7	18	16	0	0
1.54E+09	10/20/2018,	23	10	11	0	58.16	0.08	9.997	partly-cloudy	0	0.53	7	18	9.155	56.07	7	18	16	0	0
1.54E+09	10/21/2018,	0	10	11	0	57.24	0.08	9.997	clear-day	0	0.05	7	18	9.997	49.57	7	18	15	0	0
1.54E+09	10/21/2018,	1	10	11	0	54.17	0.03	9.997	clear-day	0	0.05	7	18	9.997	49.57	7	18	15	0	0
1.54E+09	10/21/2018,	2	10	11	0	53.17	0.05	9.997	clear-day	0	0.05	7	18	9.997	49.57	7	18	15	0	0
1.54E+09	10/21/2018,	3	10	11	0	52.52	0.08	9.997	clear-day	0	0.05	7	18	9.997	49.57	7	18	15	0	0
1.54E+09	10/21/2018,	4	10	11	0	52.55	0.08	9.997	clear-day	0	0.05	7	18	9.997	49.57	7	18	15	0	0
1.54E+09	10/21/2018,	5	10	11	0	51.99	0.08	9.997	clear-day	0	0.05	7	18	9.997	49.57	7	18	15	0	0
1.54E+09	10/21/2018,	6	10	11	0	51.66	0.03	9.997	clear-day	0	0.05	7	18	9.997	49.57	7	18	15	3.6	3.95
1.54E+09	10/21/2018,	7	10	11	0	51.06	0	9.997	clear-day	0	0.05	7	18	9.997	49.57	7	18	15	26.2	22.55
1.54E+09	10/21/2018,	8	10	9	0	50.24	0.05	9.997	clear-day	0	0.05	7	18	9.997	49.57	7	18	15	78.7	61.8
1.54E+09	10/21/2018,	9	10	9	0	54.82	0.03	9.997	clear-day	0	0.05	7	18	9.997	49.57	7	18	15	149.5	112.2
1.54E+09	10/21/2018,	10	10	9	0	58	0	9.997	clear-day	0	0.05	7	18	9.997	49.57	7	18	15	186.1	142.5
1.54E+09	10/21/2018,	11	10	9	0	60.88	0.02	9.997	clear-day	0	0.05	7	18	9.997	49.57	7	18	15	171.3	150
1.54E+09	10/21/2018,	12	10	9	0	63.86	0	9.997	clear-day	0	0.05	7	18	9.997	49.57	7	18	15	140.3	144.6
1.54E+09	10/21/2018,	13	10	9	0	65.49	0.05	9.997	clear-day	0	0.05	7	18	9.997	49.57	7	18	15	124.6	127.9
1.54E+09	10/21/2018,	14	10	9	0	66.53	0.09	9.997	clear-day	0	0.05	7	18	9.997	49.57	7	18	15	120.1	113.9

Screenshot taken by Experimenter A on 11/19/20

Arduino Program Flowchart


Set Azimuth and Zenith angles to values sent by Python Program
Set
spin = a - 70
tilt = z
Is tilt > 60
Yes
No
Set tilt = 60
Connect Servo 1 to port A3
Connect Servo 2 to port A1
Move Servo 1 to tilt angle
Move Servo 2 to spin angle

Screenshot taken by Experimenter A on 11/24/20, made using draw.io

Read Me

```
Using Machine Learning to predict solar output generated by solar tracker
(Powered by Dark Sky)

Training the model based on GradientBoosting Regression explained in:
https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.Gradien
tBoostingRegressor.html

Summary

The project is split into three parts
preparer.py: Collects weather data from Dark Sky's API. Selects only
specific columns and formats to fit the csv. Collects solar output data
from past year's project results. Combines the two.
trainer.py: Fits the data into Regression model.
predictor.py : Gets current weather forecast and predicts solar output
weather.py : Python module for reusable methods and variables
CompareTrainers.py: Compare the accuracy scores and plot the predicted
values for different model
angle_calculator.py: Calculates the azimuth and zenith angle of the Sun
```

Machine Learning Code - Preparer

```
import csv
from datetime import datetime as datetime_datetime, timedelta, date
import datetime
import weather

#Create csv file for past weather data with the headers
def create_csv():
    csvfile = open("weather_data.csv",'w')
    output = csv.writer(csvfile)
    solar_output_headers = ['with_robot_output', 'without_robot_output']
    # solar_output_headers = ['with_robot_output', 'without_robot_output']
    output.writerow(weather.common_headers + weather.hourly_headers
                    + weather.daily_headers + solar_output_headers)
    return output

def add_one_row(daily_data, row):
    millis = row['time']
    date_time = datetime_datetime.fromtimestamp(millis)
    time_formatted = date_time.strftime("%m/%d/%Y, %H:%M:%S")
    common_row_data = [row['time'], time_formatted, weather.millis_hour(millis), weather.millis_month(millis)]
    hourly_row_data = [len(row.get('icon','')), row.get('precipProbability',''),
                       row['temperature'], row.get('cloudCover',''), row.get('visibility','')]
    daily_row_data = [daily_data['icon'], daily_data['precipProbability'], daily_data['cloudCover'],
    weather.millis_hour(daily_data['sunriseTime']), weather.millis_hour(daily_data['sunsetTime']),
daily_data['visibility'],
    daily_data['temperatureMin'], weather.millis_hour(daily_data['temperatureMinTime']),
                              weather.millis_hour(daily_data['temperatureMax']),
                              weather.millis_hour(daily_data['temperatureMaxTime'])]
    solar_output_row = solar_output.get(str(millis),{})
    solar_output_row_data = [solar_output_row.get('with_robot_output',0),
                             solar_output_row.get('without_robot_output',0)]

    row_data = common_row_data + hourly_row_data + daily_row_data + solar_output_row_data
    weather_output.writerow(row_data)

#Add a row with weather and solar output data for each date
def create_csv_rows(all_dates):
    for single_date in all_dates:
        parsed = weather.forecast_response(single_date)
        hourly_data = parsed['hourly']['data']
        daily_data = parsed['daily']['data'][0]
        # Format each row of json response to fit the columns in the csv
        for row in hourly_data:
            add_one_row(daily_data, row)

#Create csv with past solar output
def prepare_past_solar_output():
    solar_dict = {}
    input_file = csv.DictReader(open("solar_output_2020.csv"))
    for row in input_file:
        solar_dict[row['time']] =  {'with_robot_output': row['with_robot_output'],
                                    'without_robot_output': row['without_robot_output']}
    return solar_dict
```

```
def all_dates():
    all_dates =  [datetime.datetime(2018, 10, 20), datetime.datetime(2018, 10, 21), datetime.datetime(2018, 10,
27)]
    all_dates += [datetime.datetime(2018, 10, 28), datetime.datetime(2018, 10, 29), datetime.datetime(2018, 11,
3)]
    all_dates += [datetime.datetime(2018, 11, 4), datetime.datetime(2018, 11, 10), datetime.datetime(2018, 11,
11)]
    all_dates += [datetime.datetime(2018, 11, 12), datetime.datetime(2018, 11, 19), datetime.datetime(2018, 11,
20)]
    all_dates += [datetime.datetime(2018, 11, 22), datetime.datetime(2018, 11, 23), datetime.datetime(2018, 11,
24)]
    all_dates +=  [datetime.datetime(2019, 11, 9), datetime.datetime(2019, 11, 10), datetime.datetime(2019, 11,
16)]
    all_dates += [datetime.datetime(2019, 11, 17), datetime.datetime(2019, 11, 18), datetime.datetime(2019, 11,
23)]
    all_dates += [datetime.datetime(2019, 11, 24), datetime.datetime(2019, 11, 25), datetime.datetime(2019, 11,
30)]
    all_dates += [datetime.datetime(2019, 12, 1)]
    return all_dates

weather_output = create_csv()
solar_output =  prepare_past_solar_output()
create_csv_rows(all_dates())
```

Machine Learning Code – Trainer

```
import pandas as pd
import numpy as np

from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error
from sklearn.model_selection import train_test_split
import statsmodels.api as sm
import matplotlib.pyplot as plt

import weather

default_predictors = ['hourly_precip_probability', 'hourly_icon',
'time', 'time_hour', 'hourly_cloud_cover','hourly_temperature']

def train_model(regressor, column_name, predictors = default_predictors):
    x = weather_data_df()[predictors] #setting the col names
    y = weather_data_df()[column_name] #setting the col names
    regressor.fit(x, y)
    return regressor

def test_model(regressor, column_name, predictors = default_predictors):
    x = weather_data_df()[predictors] #setting the col names
    y = weather_data_df()[column_name] #setting the col names

    xtrain, xtest, ytrain, ytest=train_test_split(x, y, random_state=12,
             test_size=0.2)
    regressor.fit(xtrain, ytrain)
    ypred = regressor.predict(xtest)
    mse = mean_squared_error(ytest,ypred)
    print(regressor.__class__.__name__)
    print("MSE: %.2f" % mse)
    print(regressor.score(xtest, ytest))
    x_ax = range(len(ytest))
    plt.scatter(x_ax, ytest, s=5, color="blue", label="original")
    plt.plot(x_ax, ypred, lw=0.8, color="red", label="predicted")
    plt.legend()
    plt.show()

def weather_data_df():
    weather_data_df = pd.read_csv('weather_data.csv')
    weather_data_df.dropna(inplace=True)
    weather_data_df = weather_data_df[~weather_data_df['with_robot_output'].isin([0])]
    return weather_data_df
```

Machine Learning Code - Predictor

```
import pandas as pd
import numpy as np

import time
from datetime import datetime
import json
import requests

import serial
import syslog

import weather
import trainer
from sklearn.ensemble import GradientBoostingRegressor

import angle_calculator

#Get current weather forecast and format it using the same columns used for past data
def current_weather_forecast():
    r = requests.get(weather.forecast_weather_url())
    json_response = r.text
    parsed = json.loads(json_response)
    currently = parsed['currently']
    daily = parsed['daily']['data'][0]
    return weather.current_forecast(parsed, currently, daily)

#Predict the solar output using the trained model
def predict_output(column_name):
    gbr = GradientBoostingRegressor(n_estimators=600,
    max_depth=5,
    learning_rate=0.01,
    min_samples_split=3)
    regressor = trainer.train_model(gbr, column_name)
    forecast = current_weather_forecast()
    forecast_df = pd.DataFrame([forecast],
                    columns = weather.common_headers + weather.hourly_headers + weather.daily_headers)
    return regressor.predict(forecast_df[trainer.default_predictors])

# Send a signal to arduino
def write_to_arduino():
    port = '/dev/cu.usbmodem14101'
    ard = serial.Serial(port,9600,timeout=5)
    # Serial write section
    ard.flush()
    time.sleep(10)
    time_tp = datetime.today().timetuple()
    angles = angle_calculator.calc(time_tp.tm_yday, time_tp.tm_hour, time_tp.tm_min)
    anglestr = "a:" + str(angles['azimuth']) + "z:" + str(angles['zenith'])
    ard.write(anglestr.encode())
    print ("Python value sent: ")
    print(anglestr)
    time.sleep(10)
    ard.close()

with_robot_prediction = int(predict_output('with_robot_output'))
without_robot_prediction = int(predict_output('without_robot_output'))
print("with_robot_prediction: " + str(with_robot_prediction))
print("without_robot_prediction: " + str(without_robot_prediction))
# If the model predicts that solar output with robot is greater than consumption
# send a signal to arduino
if ((with_robot_prediction - without_robot_prediction) > 10):
    print("y")
    # write_to_arduino()
else:
    print('n')
```

Machine Learning Code – Weather

```python
import requests
import json
import csv
from datetime import datetime as datetime_datetime, timedelta, date
import time
import datetime

common_headers = ['time','time_formatted', 'time_hour', 'time_month']
hourly_headers = ['hourly_icon', 'hourly_precip_probability', 'hourly_temperature',
                  'hourly_cloud_cover', 'hourly_visibility']
daily_headers =  ['daily_icon', 'daily_precip_probability', 'daily_cloud_cover',
                  'daily_sunrise_hour', 'daily_sunset_hour',
                  'daily_visibility','daily_temp_low', 'daily_temp_low_hour',
                  'daily_temp_high', 'daily_temp_high_hour' ]

def base_url():
    secret_key = '1b6fd0e44eae21e7b9598c8875382483'
    latitude = '33.0198'
    longitude = '-96.6989'
    base_url = 'https://api.darksky.net/forecast/'
    return base_url + secret_key + '/' + latitude + ',' + longitude

def timemachine_weather_url(time):
    exclude_list = 'flags, currently'
    return base_url() + ','  + str(time) + '?exclude=' + exclude_list

def forecast_weather_url():
    return base_url()

def millis_hour(millis):
    date_time = datetime_datetime.fromtimestamp(millis)
    return int(date_time.time().hour)

def millis_month(millis):
    date_time = datetime_datetime.fromtimestamp(millis)
    return int(date_time.date().month)

def forecast_response(one_date):
     # Call the api to get weather data, parse the json response
    timems = int(time.mktime(one_date.timetuple()))
    r = requests.get(timemachine_weather_url(timems))
    json_response = r.text
    parsed = json.loads(json_response)
    return parsed

#Format current forecast
def current_forecast(parsed, currently, daily):
    millis = currently['time']
    date_time = datetime_datetime.fromtimestamp(millis)
    time_formatted = date_time.strftime("%m/%d/%Y, %H:%M:%S")
    common_row_data = [currently['time'], time_formatted, millis_hour(millis), millis_month(millis)]
    hourly_row_data = [len(currently.get('icon','')), currently.get('precipProbability',''),
                       currently['temperature'], currently.get('cloudCover',''),
                       currently.get('visibility','')]
    daily_row_data = [daily['icon'], daily['precipProbability'], daily['cloudCover'],
                      millis_hour(daily['sunriseTime']), millis_hour(daily['sunsetTime']),
                      daily['visibility'], daily['temperatureMin'],
                      millis_hour(daily['temperatureMinTime']), daily['temperatureMax'],
                      millis_hour(daily['temperatureMaxTime'])]
    row_data = common_row_data + hourly_row_data + daily_row_data
    return row_data

print(forecast_weather_url())
print(forecast_response(date.today()))
```

Machine Learning Code - CompareTrainers

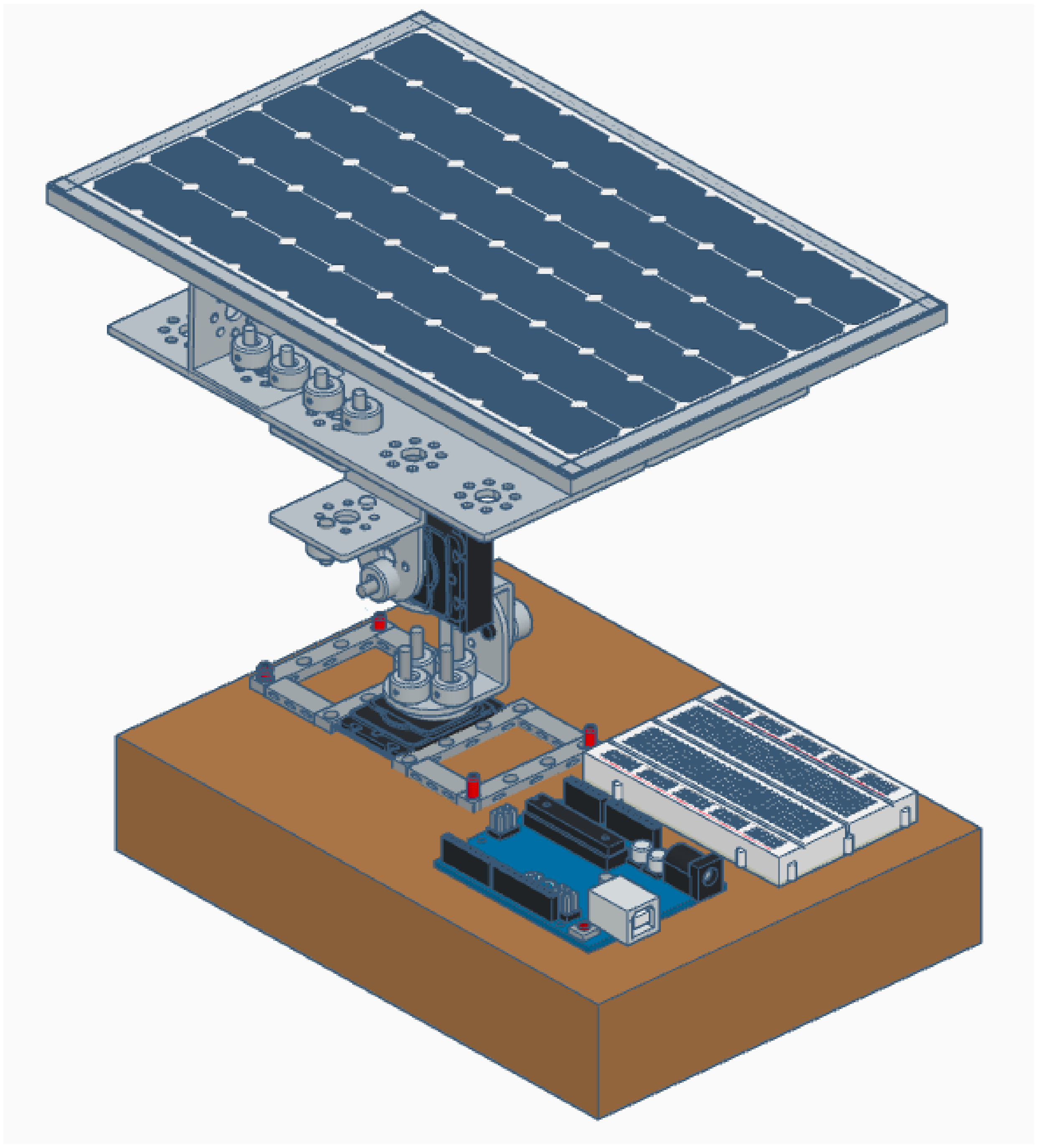
```
from sklearn import linear_model
from sklearn import tree
from sklearn import ensemble
from sklearn import neighbors
from sklearn import naive_bayes

import trainer

def test_linear_models(column_name):
    br = linear_model.BayesianRidge()
    trainer.test_model(br, column_name)
    l = linear_model.Lasso()
    trainer.test_model(l, column_name)
    ols = linear_model.LinearRegression()
    trainer.test_model(ols, column_name)
    en = linear_model.ElasticNet()
    trainer.test_model(en, column_name)

def test_decision_tree_models(column_name):
    dt = tree.DecisionTreeRegressor()
    trainer.test_model(dt, column_name)


def test_ensemble_models(column_name):
    rfr = ensemble.RandomForestRegressor()
    trainer.test_model(rfr, column_name)
    abr = ensemble.AdaBoostRegressor()
    trainer.test_model(abr, column_name)
    gbr = ensemble.GradientBoostingRegressor()
    trainer.test_model(gbr, column_name)

def test_nearest_neighbor_models(column_name):
    knn = neighbors.KNeighborsRegressor()
    trainer.test_model(knn, column_name)

test_linear_models('with_robot_output')
test_decision_tree_models('with_robot_output')
test_ensemble_models('with_robot_output')
test_nearest_neighbor_models('with_robot_output')
```

Mathematical Algorithm Code – Angle_Calculator

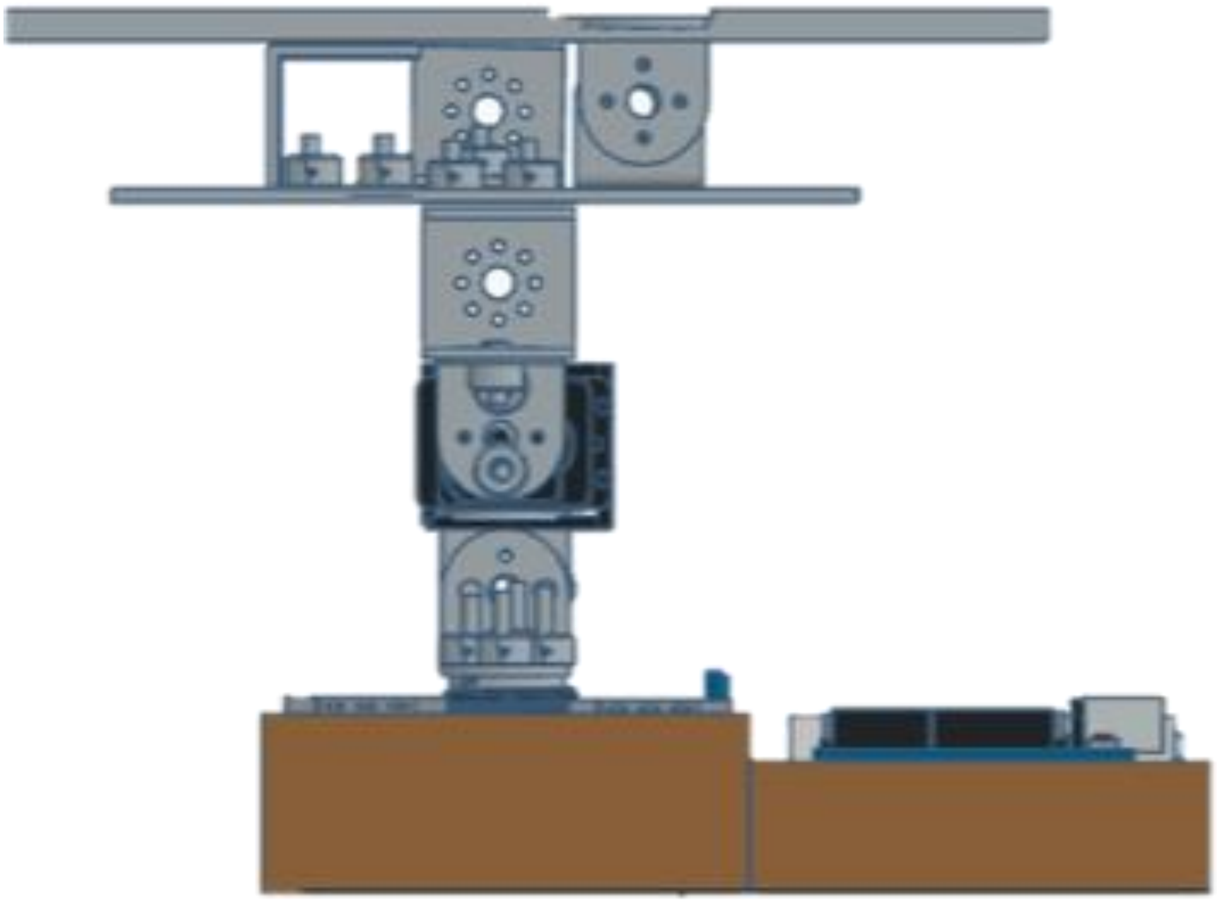
```
import csv
import math

a = 0
z = 0

def cosine (var):
    return math.cos(math.radians(var))

def sine (var):
    return math.sin(math.radians(var))

def arc (var):
    return math.acos(var) * (180/math.pi)

def calc (year, hour, mn, lat=32, lng=-96, tz=-6):
    print("y,h,m: " + str(year) + "," + str(hour) + "," + str(mn))
    y = (2 * math.pi/365) * year - 1 + (hour - 12)/24
    eq = 229.18 * (0.000075 + 0.001868 * math.cos(y) - 0.032077 * math.sin(y) - 0.014615 * math.cos(2 * y) -
0.040849 * math.sin(2 * y))
    decl = 0.006918 - (0.399912 * math.cos(y)) + (0.070257 * math.sin(y)) - (0.006758 * math.cos(2 * y)) +
(0.000907 * math.sin(2 * y)) - (0.002697 * math.cos(3 * y)) + (0.00148 * math.sin(3 * y))
    time_offset = eq + (4 * lng) - (60 * tz)
    tst = (hour * 60) + mn + time_offset
    ha = (tst/4) - 180
    coszenith = (sine(lat) * math.sin(decl)) + (cosine(lat) * math.cos(decl) * cosine(ha))
    z = arc(coszenith)
    a = arc(-(sine(lat) * coszenith - math.sin(decl))/(cosine(lat) * sine(z)))
    if (ha > 0):
        a = 360 - a
    print(a)
    print(z)
    return {"azimuth":round(a,2), "zenith":round(z,2)}
```

Arduino Code

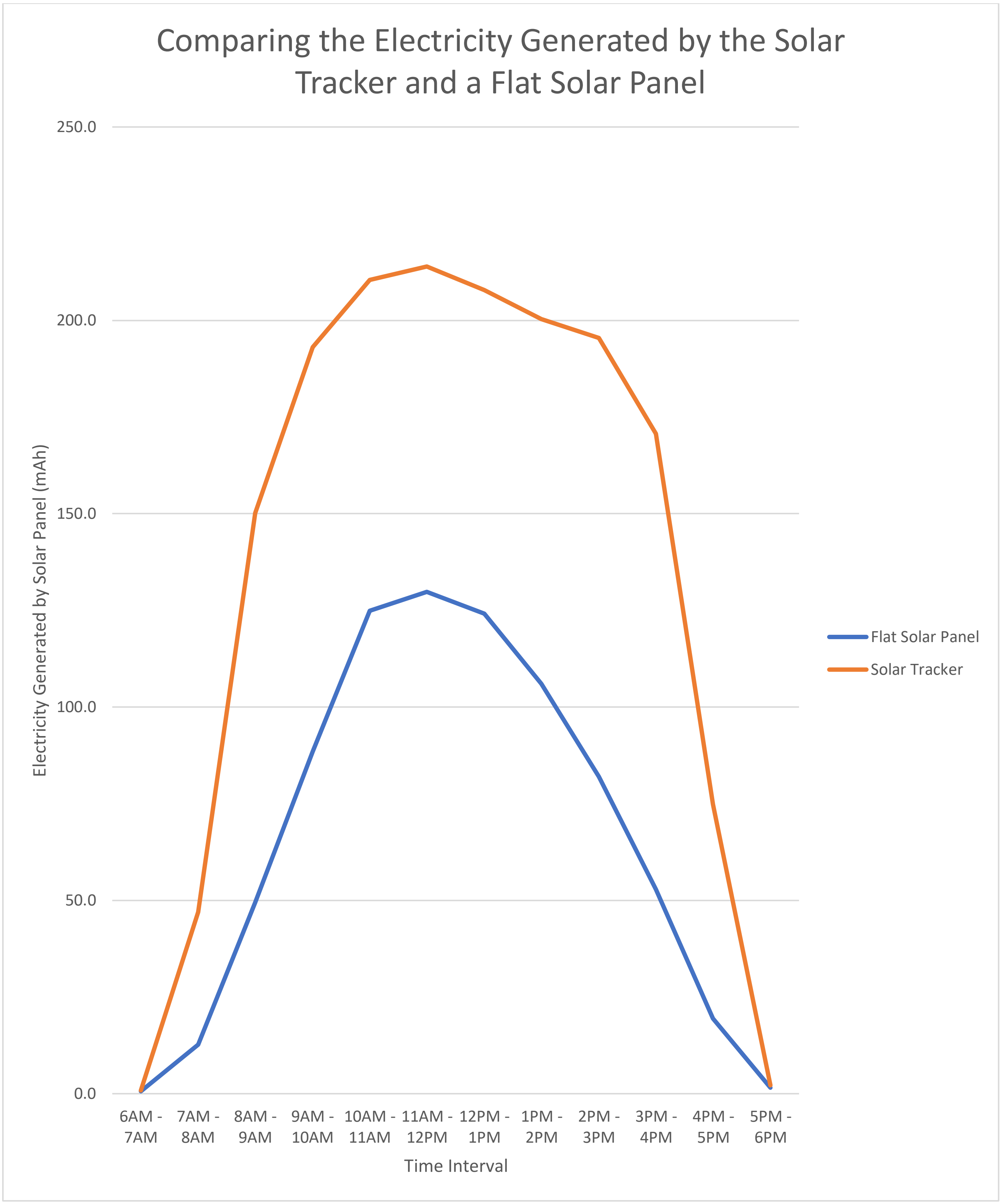
```
#include <Servo.h>

Servo spinny;
Servo tilty;
boolean newData = false;
String receivedStr;
int azimuth,zenith;

void spinAndTilt() {
  double spin = azimuth - 75;
  double tilt = zenith;
  if (tilt > 60){
    tilt = 60;
  }
  spinny.attach(A1);
  tilty.attach(A3);
  delay(500);
  spinny.write(spin);
  delay(500);
  tilty.write(tilt);
  delay(60000);
  Serial.println(a);
  Serial.println(z);
}
void readSerial() {
  if (Serial.available() > 0) {
    receivedStr = Serial.readString();
    newData = true;
  }
}

void readAngle() {
  if (newData == true) {
    Serial.println("Angles:" +receivedStr);
    int a = receivedStr.indexOf('a');
    int z = receivedStr.indexOf('z');
    azimuth = receivedStr.substring(0,a);
    zenith = receivedStr.substring(a+1,z);
    newData = false;
  }
}

void setup() {
  Serial.begin(9600);
  delay(5000);
}

void loop() {
  delay(5000);
  readSerial();
  if (newData == true) {
     readAngle();
     spinAndTilt();
    }
    newData = false;
}
```



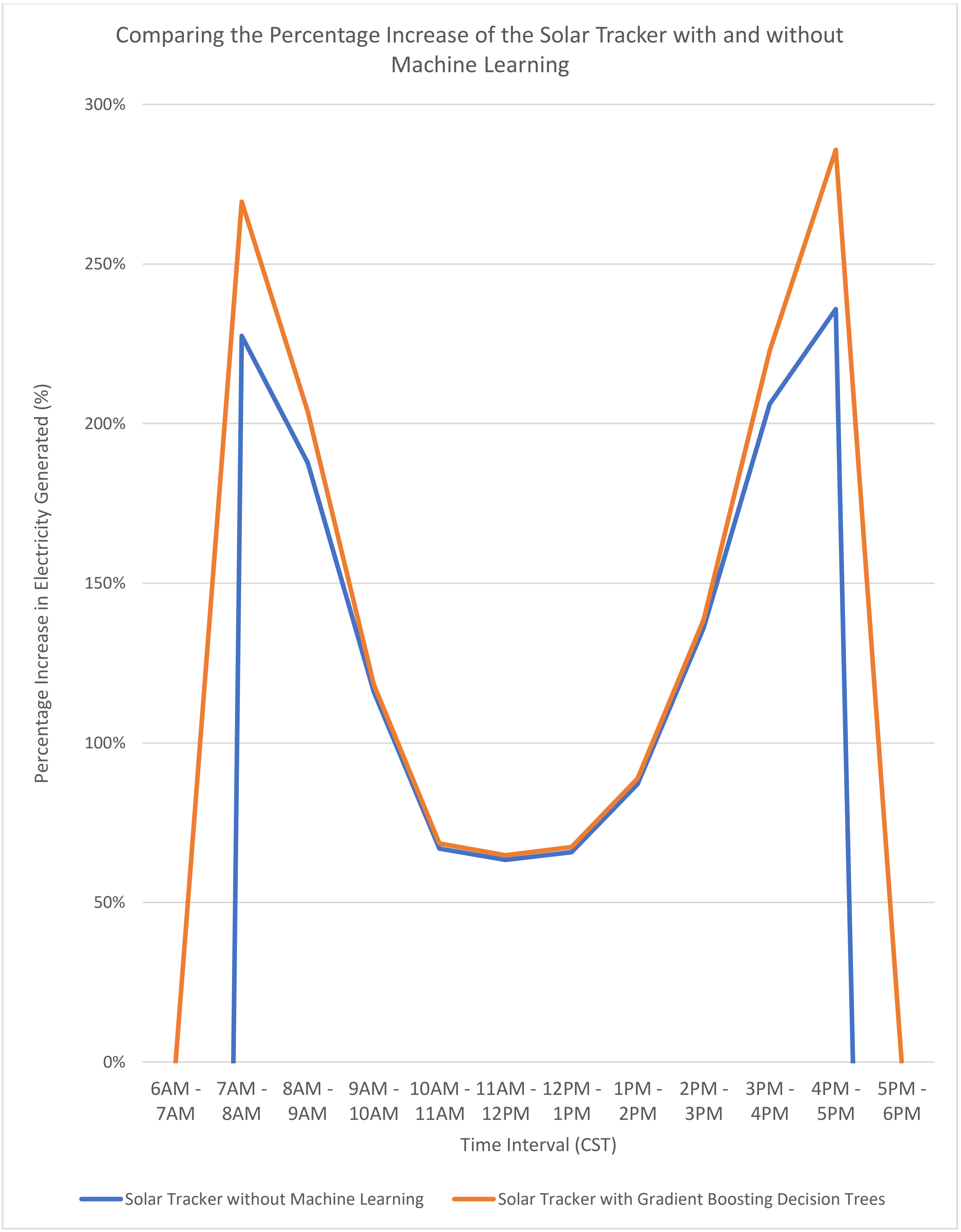
Screenshot taken by Experimenter B on 11/9/20, made using Tinkercad



Screenshot taken by Experimenter B on 11/9/20, made using Tinkercad



Graph created by Experimenter A on 12/1/20 using MS Excel



Graph created by Experimenter A on 12/1/20 using MS Excel

Data Table #1 - Solar Tracker without Machine Learning

Increasing the Electricity Generated by an Intelligent Dual-Axis Solar Tracker													
	Net Electricity Generated (mAh) - Energy Consumption of Robot is Subtracted												Percentage Increase in Electricity Generated Using the Solar Tracker without Solar Panels
	Flat Solar Panel						Solar Tracker without Machine Learning						
Time Interval (CST)	11/7/20	11/9/20	11/12/20	11/16/20	11/17/20	Average	11/7/20	11/9/20	11/12/20	11/16/20	11/17/20	Average	
6AM - 7AM	0.6	0.8	0.6	0.6	0.5	0.6	-8.9	-8.8	-9.4	-9.3	-9.4	-9.2	-1577%
7AM - 8AM	13.7	13.9	13.4	11.2	11.3	12.7	46.0	45.2	44.3	34.2	38.3	41.6	228%
8AM - 9AM	49.2	50.3	50.3	48.4	49	49.4	142.3	143.0	144.3	139.7	142.0	142.3	188%
9AM - 10AM	92.3	94.5	93.4	80.4	82	88.5	193.6	193.9	192.3	190.3	185.5	191.1	116%
10AM - 11AM	127.6	126.2	128.2	120.3	122.2	124.9	213.5	211.2	208.2	198.1	211.4	208.5	67%
11AM - 12PM	130.4	132.3	130.2	128.6	127.4	129.8	213.9	213.2	214.3	204.6	213.6	211.9	63%
12PM - 1PM	126	124.5	128.1	120.1	122.2	124.2	208.4	205.4	207.0	200.1	208.3	205.8	66%
1PM - 2PM	105	105.3	109.3	103.4	107.3	106.1	201.1	199.2	200.3	193.3	197.8	198.3	87%
2PM - 3PM	83.4	84.5	85.2	80.2	76.4	81.9	197.5	193.6	193.7	188.2	194.3	193.5	136%
3PM - 4PM	56.2	55.2	53.1	49.8	50	52.9	156.9	162.3	168.3	160.2	161.5	161.8	206%
4PM - 5PM	19.3	20.6	20	18.2	18.9	19.4	70.2	68.2	63.3	60.5	63.6	65.2	236%
5PM - 6PM	1.7	1.8	1.7	1.3	1.2	1.5	-8.7	-8.5	-9.0	-9.0	-8.6	-8.8	-669%

Data Table created by Experimenter A on 12/1/20 using MS Excel

Data Table #2 - Solar Tracker with Machine Learning

Increasing the Electricity Generated by an Intelligent Dual-Axis Solar Tracker													
	Net Electricity Generated (mAh) - Energy Consumption of Robot is Subtracted												Percentage Increase in Electricity Generated Using the Solar Tracker with Gradient Boosting Decision Trees
	Flat Solar Panel						Solar Tracker Using Gradient Boosting Regression						
Time Interval (CST)	11/7/20	11/9/20	11/12/20	11/16/20	11/17/20	Average	11/7/20	11/9/20	11/12/20	11/16/20	11/17/20	Average	
6AM - 7AM	0.6	0.8	0.6	0.6	0.5	0.6	0.6	0.8	0.6	0.6	0.5	0.6	0%
7AM - 8AM	13.7	13.9	13.4	11.2	11.3	12.7	48.4	49.4	50.2	42.4	44.3	46.9	270%
8AM - 9AM	49.2	50.3	50.3	48.4	49	49.4	153.3	149.9	150.2	148.7	148.6	150.1	204%
9AM - 10AM	92.3	94.5	93.4	80.4	82	88.5	195.6	195.9	194.3	192.3	187.5	193.1	118%
10AM - 11AM	127.6	126.2	128.2	120.3	122.2	124.9	215.5	213.2	210.2	200.1	213.4	210.5	69%
11AM - 12PM	130.4	132.3	130.2	128.6	127.4	129.8	215.9	215.2	216.3	206.6	215.6	213.9	65%
12PM - 1PM	126	124.5	128.1	120.1	122.2	124.2	210.4	207.4	209	202.1	210.3	207.8	67%
1PM - 2PM	105	105.3	109.3	103.4	107.3	106.1	203.1	201.2	202.3	195.3	199.8	200.3	89%
2PM - 3PM	83.4	84.5	85.2	80.2	76.4	81.9	199.5	195.6	195.7	190.2	196.3	195.5	139%
3PM - 4PM	56.2	55.2	53.1	49.8	50	52.9	168.9	170.3	174.3	170.2	169.5	170.6	223%
4PM - 5PM	19.3	20.6	20	18.2	18.9	19.4	80.4	77.8	65.3	75.3	75.4	74.8	286%
5PM - 6PM	1.7	1.8	1.7	1.3	1.2	1.5	1.7	1.8	1.7	1.3	1.2	1.5	0%

Data Table created by Experimenter A on 12/1/20 using MS Excel

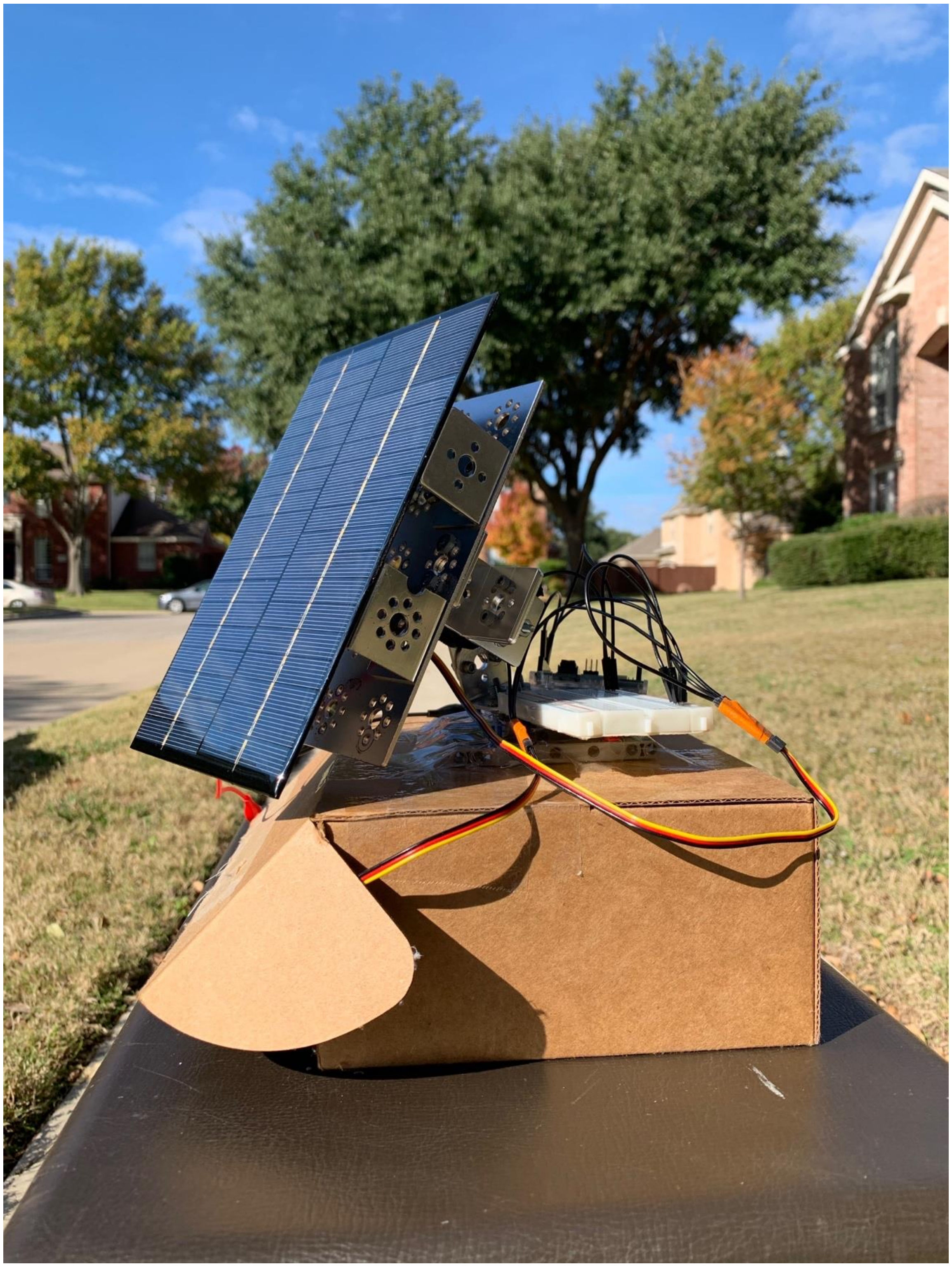


Photo Taken by Experimenter A on 11/10/20

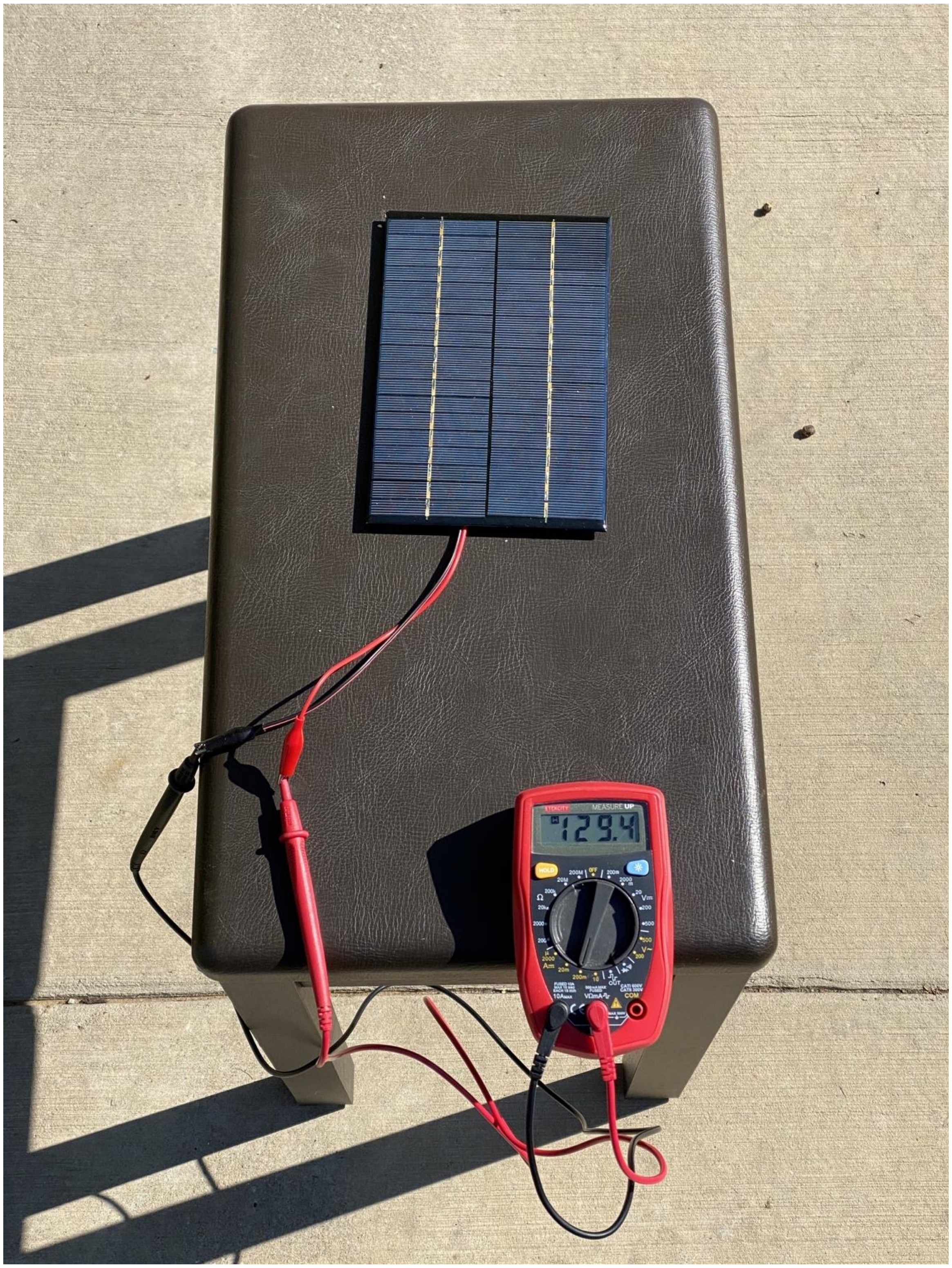


Photo Taken by Experimenter A on 11/9/20

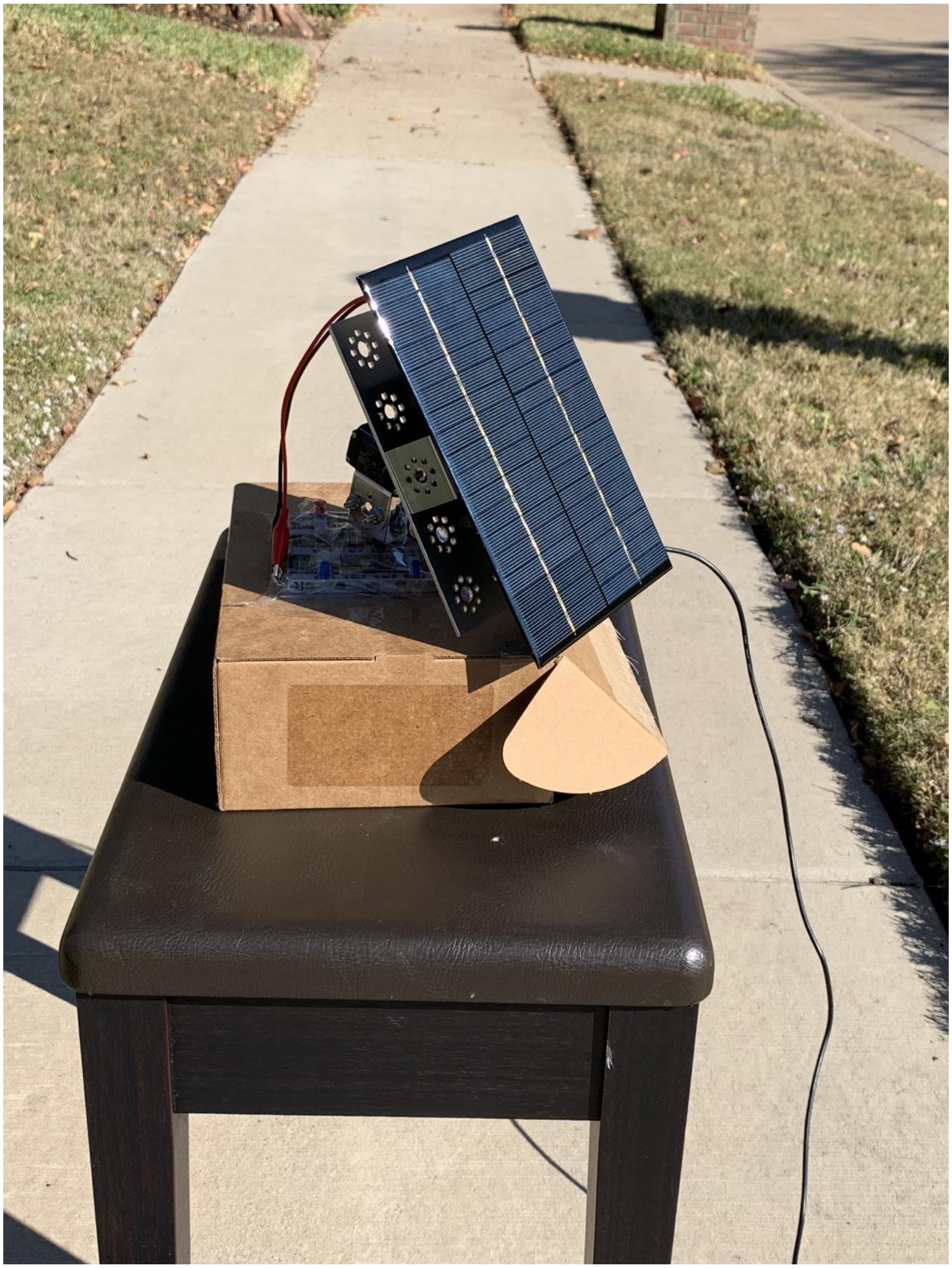


Photo Taken by Experimenter B on 11/25/20

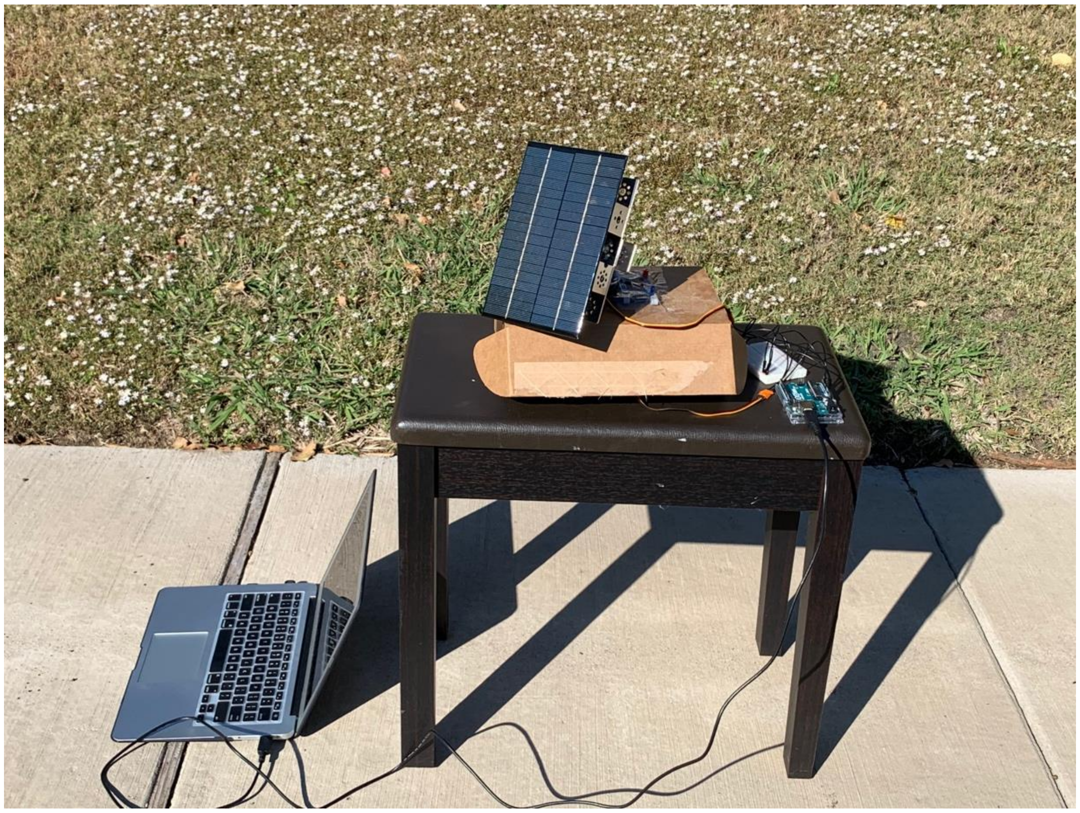


Photo Taken by Experimenter A on 11/30/20

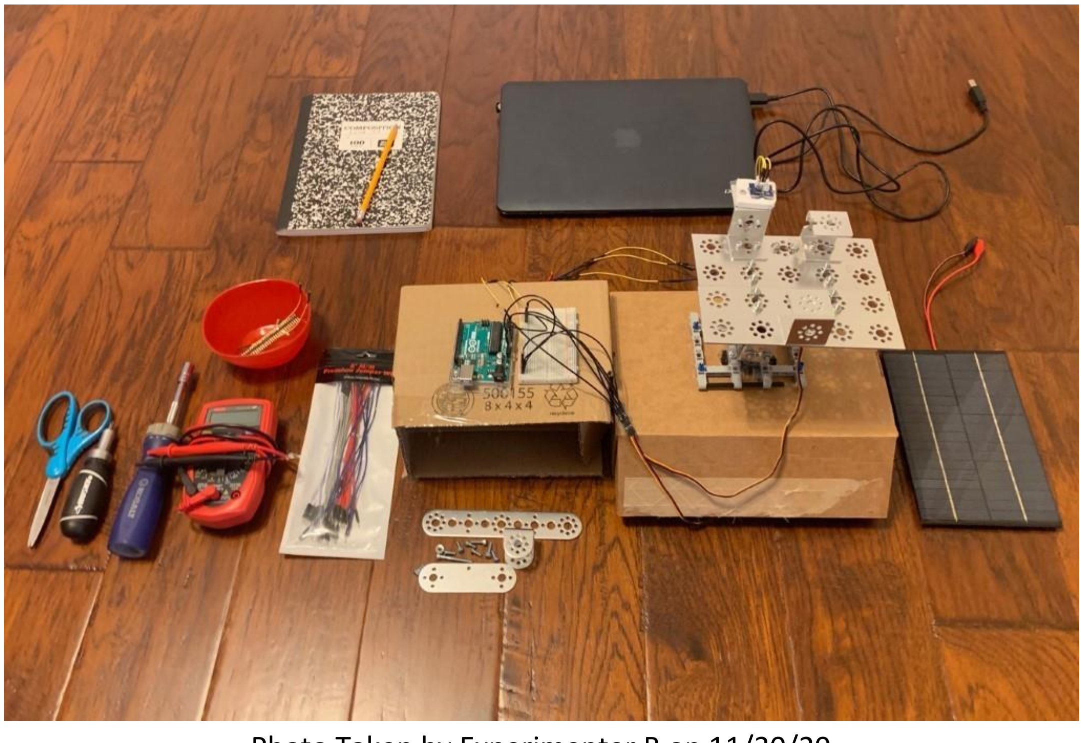


Photo Taken by Experimenter B on 11/30/20

Error Analysis

The machine learning algorithm uses data from the past two years during the same two-month period. This is not significant enough to make an accurate prediction. The data was mostly tested and collected on sunnier days which make it less reliable to predict the output for days with high cloud cover or rain. To fix this, we can include data tested throughout the year.

Data Analysis

The stationary panel, on average, produced over 120 mAh for a duration of 3 consecutive hours with a peak value of 129.8 mAh. In comparison the robot, on average, produced over 120 mAh (net generation) for 8 consecutive hours with its highest value at 213.9 mAh. The use of a dual-axis solar tracker ensured that there is maximum sunlight reception for a sustained period of time in contrast to one peak value for a short period of time without the robot. The robot without machine learning produced an average increase of 102% from the flat panel. The addition of machine learning produced a 119% increase in net generation of electricity. The greatest increase in electricity generation was 7:00 – 8:00 AM and 4:00 – 5:00 PM when the sun shines brightly but is not directly above the solar panel. The solar tracker graphs are shaped like a table top, and the flat solar panel graphs are shaped like a bell curve.

Conclusion

The stated engineering goal was confirmed. The electricity generated was 119% greater in the robot surpassing the goal of a 100% increase. The robot was able to maximize energy production and sustain it for a longer duration, thus enabling the dual-axis robot to generate more electricity than the solar panel would have on its own.

Application & Future Research

Using a machine learning and mathematical approach with a dual-axis solar tracker, solar panels can be used to their maximum potential and be a greener, cost effective way to power the world.
Future research could include the incorporation of energy generated from solar farms. The machine learning algorithm could be improved to include data throughout the year in multiple locations. Additionally, ideal geographical locations for solar power generation can be further explored to find maximum sunlight reception.

Annotated Bibliographies

General Solar Position Calculations. (n.d). Retrieved from https://www.esrl.noaa.gov/gmd/grad/solcalc/solareqns.PDF.

> This source explains and provides the equations necessary for calculating the hour angle based on the latitude of the location and the sun declination. Using this, the source also provides the equation to find the UTC time of sunrise, solar noon, and sunset. This source can be used to calculate when the time of sunrise and sunset in Plano; also it can be used to calculate the time when the sun will be directly "up" which is called solar noon. The derived hour angle will help the tilting of the solar tracker to the correct angle. This source is reliable because its provides a general mathematical approach to calculate information crucial for the solar tracker. The inputs for the functions are latitude, longitude, and declination and can be set to reflect the specific time of year and location of Plano.

How to Maximize Solar Panel Efficiency. (n.d.). Retrieved from https://www.wholesalesolar.com/solar-information/solar-panel-efficiency.

> This source provides exact numerical information on the optimum position that a solar panel should tilt to for maximum efficiency for certain latitudes. It also provides information about magnetic declination which can affect the positions at which the panel will be facing its true direction. This source can be used to help determine the factors that can be used when determining the best position for the solar panel. This source has a small biased towards the company that makes solar kits and products.

Kolstad, J. (2016). Solar Energy. In B. W. Lerner, K. L. Lerner, & T. Riggs (Eds.), In Context Series. Energy: In Context (Vol. 2, pp. 798-803). Farmington Hills, MI: Gale. Retrieved from https://link.gale.com/apps/doc/CX3627100203/SCIC?u=j043910&sid=SCIC&xid=715767cb.

> This source outlines the process for capturing solar radiation and the conversion into electrical energy by using thermal radiation and solar cells. It also highlights the historical background of where and how solar energy started during the classical era. It concludes with the applications and issues of solar energy for everyday use. This source can be used to identify the issues that make solar energy less efficient. Identifying major issues with solar energy, can help orient the project to meet those needs. This source is credible and reliable as it states information from both negative and positive impacts.

Maehlum, M.A. How Much Do Solar Panel Cost. Retrieved from https://energyinformative.org/solar-panels-cost/

> This source explains the cost of an average solar panel in a residential setting. It lays out the amount for all the fees involved in addition to providing information about the electricity usage of an average single family home. The information in this source can be used to determine the cost of existing PV systems, and it can be used to compare to the amount after the

implementation of a dual-axis solar tracker. The information in this source has minimal bias because it is an educational organization for the purpose of educating the reader.

McQuistan, A. (2017, December 6). Using Machine Learning to Predict the Weather: Part 1. Retrieved from https://stackabuse.com/using-machine-learning-to-predict-the-weather-part-1/.

This source provides a detailed procedure and advice to predict the weather using machine learning. The first part is data collection and processing from Weather Underground. The next step is modeling the data with linear regression using Sci-kit learn Python libraries. The final step is the modeling of neural networks. This source can be used as a guide for the steps required to make a weather predictor for the application of a solar tracker. This source has a bias towards Python because the website specializes in Python tips. This source is credible because it has a full detailed procedure of the steps taken to predict the weather.

McQuistan, A. (2017, December 6). Using Machine Learning to Predict the Weather: Part 2. Retrieved from https://stackabuse.com/using-machine-learning-to-predict-the-weather-part-2/.

This article provides a detailed description about how the Scikit Learn Python library works. It highlights a process called stepwise regression which uses backwards elimination. This process assigns a significance level (0.05) to each variable and inputs it into the model. This indicates the variables that will make the most impact on the machine learning predictor. This source can be used to help with the identification of the key weather variables for our machine learning model.The information in this source has a minimal bias because it is based off of the experience of one user.

Schroeder, D. V. (n.d.). The Sun and the Seasons. Retrieved from https://physics.weber.edu/schroeder/ua/sunandseasons.html

This source provides information about the position of earth relative to the sun. It talks about the position of the sun at different latitudes on earth along with information about peak sun exposure. This information can be used to determine the optimum angle of tilt for certain times during the year that would potentially generate more electricity. It can be used to predict patterns of the sun for which data can be incorporated into our program. This source is meant for educational purposes, so it eliminates most bias except for a factor of unreliability by the author.

Use the Sine to Show the Number of Daylight Hours in a Location. Retrieved from https://www.dummies.com/education/math/trigonometry/use-the-sine-to-show-the-number-of-daylight-hours-in-a-location/.

This source derives the equation for finding the number of hours of daylight in a single day given the time of year. This equation is specific for San Diego, California because the latitude affects the equation for calculating the number of daylight hours. The graph is a sine function with an amplitude of 2.4 which means that the number of daylight hours extends to 2.4 hours above and below 12 hours (the average number of daylight hours). This information will be used as a guide to calculate the sine function for daylight in Plano, Texas to determine the time frame the solar tracker should be active. The bias in this article is that this method is focused on San Diego and not any city in general. This source is reliable because it provides a solid mathematical approach to calculating the number of daylight hours in a location.

What Is Machine Learning?: How It Works, Techniques & Applications. (n.d.). Retrieved from https://www.mathworks.com/discovery/machine-learning.html.

The source above mentioned information about the types of machine learning and examples of uses. It explains the methodology behind the process of how it takes data and analyzes it. This source can be used to understand the process of how the computer understands our data and the ways in which it can be applied to a real world problem. It also provides a tool, MATLAB, which can be used to generate graphs and diagrams for data. This source provides helpful resources that are biased due to potential sponsorship.

Zwass, V. (2019, September 27). Neural network. Retrieved from https://www.britannica.com/technology/neural-network.

This source mentions the basics of a neural network and the process through which information flows. Additionally, it outlines the applications of this technology in the real world. This information will be used as a basis of how to analyze data and use it to potentially reduce the electricity being generated. This source is reliable and provides background information about how deep learning works.

(n.d.). Retrieved from https://store.arduino.cc/usa/grove-light-sensor-v1-2.

This source includes information about a key component for detecting the light intensity - a light sensor. This product is compatible with arduino and produces an output of an analog value (the brighter the light, the larger the value). This can be used in our project as reliable device to measure the intensity of light, so it can determine which angle had the highest value. Because it is from a company selling this product, the information in this source is biased towards them.