JAVA ARDUINO CODE FOR DATA COLLECTION:

```
#include <SPI.h>
#include <SD.h>

const int chipSelect = 53;  // pin connected to CS of the SD
card module

// pins for voltage and current measurements
const int voltagePin1 = A0, currentPin1 = A1;
const int voltagePin2 = A2, currentPin2 = A3;
const int voltagePin3 = A4, currentPin3 = A5;
const int voltagePin4 = A6, currentPin4 = A7;
const int voltagePin5 = A8, currentPin5 = A9;
const int voltagePin6 = A10, currentPin6 = A11;

const String filename = "4-22data.csv";
File dataFile;

void setup() {
    Serial.begin(9600);
    delay(1000);

    // check SD card initialization
    if (!SD.begin(chipSelect)) {
        Serial.println("SD card initialization failed -- check
wiring or SD card.");
        return;
    }
    Serial.println("SD card initialized");

    // check if file exists
    if (!SD.exists(filename)) {
        Serial.println("file not found. creating new file...");
        dataFile = SD.open(filename, FILE_WRITE);
        if (dataFile) {
            dataFile.println("time (min),voltage1 (V),current1
(A),voltage2 (V),current2 (A),voltage3 (V),current3 (A),voltage4
(V),current4 (A),voltage5 (V),current5 (A),voltage6 (V),current6
(A)");
            Serial.println("new file created with headers.");
            dataFile.close();
        } else {
            Serial.println("error creating file -- possibly
permissions or corruption.");
        }
    } else {
```

```
        Serial.println("file exists. appending data...");
    }

    pinMode(LED_BUILTIN, OUTPUT);
}

void loop() {
    // open file once and keep it open while writing data
    dataFile = SD.open(filename, FILE_WRITE);
    bool worked = true;
    if (dataFile) {
        // read voltage from solar panels
        float voltage1 = analogRead(voltagePin1) * (5.0 /
1023.0);
        float voltage2 = analogRead(voltagePin2) * (5.0 /
1023.0);
        float voltage3 = analogRead(voltagePin3) * (5.0 /
1023.0);
        float voltage4 = analogRead(voltagePin4) * (5.0 /
1023.0);
        float voltage5 = analogRead(voltagePin5) * (5.0 /
1023.0);
        float voltage6 = analogRead(voltagePin6) * (5.0 /
1023.0);

        // read shunt voltage and calculate current (V_shunt /
R)
        float current1 = (analogRead(currentPin1) * (5.0 /
1023.0)) / 10.0;
        float current2 = (analogRead(currentPin2) * (5.0 /
1023.0)) / 10.0;
        float current3 = (analogRead(currentPin3) * (5.0 /
1023.0)) / 10.0;
        float current4 = (analogRead(currentPin4) * (5.0 /
1023.0)) / 10.0;
        float current5 = (analogRead(currentPin5) * (5.0 /
1023.0)) / 10.0;
        float current6 = (analogRead(currentPin6) * (5.0 /
1023.0)) / 10.0;

        // write data to SD card
        dataFile.print((millis() / 1000) / 60); // time in
minutes
        dataFile.print(",");
        dataFile.print(voltage1); dataFile.print(",");
        dataFile.print(current1); dataFile.print(",");
        dataFile.print(voltage2); dataFile.print(",");
```

```
        dataFile.print(current2); dataFile.print(",");
        dataFile.print(voltage3); dataFile.print(",");
        dataFile.print(current3); dataFile.print(",");
        dataFile.print(voltage4); dataFile.print(",");
        dataFile.print(current4); dataFile.print(",");
        dataFile.print(voltage5); dataFile.print(",");
        dataFile.print(current5); dataFile.print(",");
        dataFile.print(voltage6); dataFile.print(",");
        dataFile.println(current6);

        dataFile.close();  // close file after writing

        // print to serial monitor
        Serial.print("Time: "); Serial.print((millis() / 1000) /
60);
        Serial.println(" min");

        Serial.print("  voltage1: "); Serial.print(voltage1);
Serial.print(" V, current1: "); Serial.print(current1);
Serial.println(" A");
        Serial.print("  voltage2: "); Serial.print(voltage2);
Serial.print(" V, current2: "); Serial.print(current2);
Serial.println(" A");
        Serial.print("  voltage3: "); Serial.print(voltage3);
Serial.print(" V, current3: "); Serial.print(current3);
Serial.println(" A");
        Serial.print("  voltage4: "); Serial.print(voltage4);
Serial.print(" V, current4: "); Serial.print(current4);
Serial.println(" A");
        Serial.print("  voltage5: "); Serial.print(voltage5);
Serial.print(" V, current5: "); Serial.print(current5);
Serial.println(" A");
        Serial.print("  voltage6: "); Serial.print(voltage6);
Serial.print(" V, current6: "); Serial.print(current6);
Serial.println(" A");

        Serial.println("data appended");
        digitalWrite(LED_BUILTIN, HIGH);
        delay(500);
        digitalWrite(LED_BUILTIN, LOW);
    } else {
        Serial.println("error opening file.");
        worked = false;
    }

    if (worked) {
        int i = 0;
```

```
        while (i < 360) {
            delay(4000);
            digitalWrite(LED_BUILTIN, HIGH);
            delay(900);
            digitalWrite(LED_BUILTIN, LOW);
            delay(100);
            i++;
        }
    } else {
        delay(1800000);
    }
}
```

PYTHON CODE FOR DATA PROCESSING:

```
# import numpy, pandas, and matplotlib under standard aliases
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.stats import linregress
In [68]:
# read .csv of raw data into a dataframe
plant_df = pd.read_csv("./4-22DATA.csv")
In [69]:
# read .csv of raw height data into a dataframe
height_df = pd.read_csv("./plantdataclean.csv")
In [ ]:

DATA CLEANING:
In [70]:
# pull the first 20 rows of the data from the df
plant_df.head(20)
Out[70]:
```

	time (min)	voltage1 (V)	current1 (A)	voltage2 (V)	current2 (A)	voltage3 (V)	current3 (A)	voltage4 (V)	current4 (A)	voltage5 (V)	current5 (A)	voltage6 (V)	current6 (A)
0	0	4.47	0.29	4.46	0.27	3.88	0.24	2.76	0.21	2.35	0.21	3.54	0.26
1	0	4.42	0.29	4.46	0.27	3.84	0.24	2.75	0.21	2.33	0.21	3.51	0.26
2	30	2.30	0.21	2.31	0.21	2.16	0.20	3.28	0.25	2.46	0.22	3.62	0.25
3	60	3.05	0.24	2.21	0.21	2.07	0.19	2.80	0.23	3.88	0.27	2.17	0.20
4	90	2.56	0.22	2.84	0.23	2.59	0.21	2.20	0.20	3.06	0.23	2.19	0.21
5	120	2.07	0.19	2.14	0.20	2.02	0.19	2.02	0.19	2.23	0.20	2.10	0.20
6	150	1.97	0.19	2.03	0.20	1.92	0.19	1.97	0.19	2.14	0.20	1.89	0.19

	time (min)	voltage1 (V)	current1 (A)	voltage2 (V)	current2 (A)	voltage3 (V)	current3 (A)	voltage4 (V)	current4 (A)	voltage5 (V)	current5 (A)	voltage6 (V)	current6 (A)
7	180	1.92	0.19	1.98	0.19	1.92	0.19	1.89	0.19	2.05	0.19	1.96	0.19
8	210	1.91	0.19	2.00	0.19	1.92	0.19	1.91	0.19	2.07	0.19	1.91	0.19
9	240	1.88	0.19	1.97	0.19	1.90	0.18	1.91	0.19	2.03	0.19	1.90	0.19
10	270	1.93	0.19	2.00	0.19	1.94	0.19	1.93	0.19	2.06	0.19	1.90	0.19
11	300	1.86	0.18	1.94	0.19	1.86	0.18	1.89	0.18	2.00	0.19	1.97	0.19
12	330	1.86	0.18	1.94	0.19	1.87	0.18	1.89	0.18	2.00	0.19	1.86	0.18
13	360	1.86	0.18	1.94	0.19	1.83	0.18	1.88	0.18	2.00	0.19	1.87	0.18
14	390	1.86	0.18	1.92	0.19	1.73	0.17	1.84	0.18	1.99	0.19	1.91	0.19
15	420	1.86	0.18	1.91	0.18	1.67	0.17	1.84	0.18	1.99	0.19	1.95	0.19
16	450	1.86	0.18	1.90	0.19	1.77	0.18	1.88	0.18	1.97	0.19	1.87	0.18
17	480	1.85	0.18	1.93	0.19	1.75	0.18	1.87	0.18	1.99	0.19	1.86	0.18
18	510	1.85	0.18	1.90	0.19	1.70	0.17	1.84	0.18	1.97	0.19	1.92	0.19
19	540	1.84	0.18	1.91	0.19	1.77	0.18	1.86	0.18	1.96	0.19	1.84	0.18

In [71]:

```
# remove first row and reset index
plant_df = plant_df.iloc[1:].reset_index(drop=True)
```

In [72]:

```
# reset time column to remove irregularities from resets
plant_df.iloc[:, 0] = [i * 30 for i in range(len(plant_df))]
In [73]:
# pull the first 20 rows of the data from the df
plant_df.head(20)
Out[73]:
```

	time (min)	voltage1 (V)	current1 (A)	voltage2 (V)	current2 (A)	voltage3 (V)	current3 (A)	voltage4 (V)	current4 (A)	voltage5 (V)	current5 (A)	voltage6 (V)	current6 (A)
0	0	4.42	0.29	4.46	0.27	3.84	0.24	2.75	0.21	2.33	0.21	3.51	0.26
1	30	2.30	0.21	2.31	0.21	2.16	0.20	3.28	0.25	2.46	0.22	3.62	0.25
2	60	3.05	0.24	2.21	0.21	2.07	0.19	2.80	0.23	3.88	0.27	2.17	0.20
3	90	2.56	0.22	2.84	0.23	2.59	0.21	2.20	0.20	3.06	0.23	2.19	0.21
4	120	2.07	0.19	2.14	0.20	2.02	0.19	2.02	0.19	2.23	0.20	2.10	0.20
5	150	1.97	0.19	2.03	0.20	1.92	0.19	1.97	0.19	2.14	0.20	1.89	0.19
6	180	1.92	0.19	1.98	0.19	1.92	0.19	1.89	0.19	2.05	0.19	1.96	0.19
7	210	1.91	0.19	2.00	0.19	1.92	0.19	1.91	0.19	2.07	0.19	1.91	0.19
8	240	1.88	0.19	1.97	0.19	1.90	0.18	1.91	0.19	2.03	0.19	1.90	0.19
9	270	1.93	0.19	2.00	0.19	1.94	0.19	1.93	0.19	2.06	0.19	1.90	0.19
10	300	1.86	0.18	1.94	0.19	1.86	0.18	1.89	0.18	2.00	0.19	1.97	0.19
11	330	1.86	0.18	1.94	0.19	1.87	0.18	1.89	0.18	2.00	0.19	1.86	0.18

	time (min)	voltage1 (V)	current1 (A)	voltage2 (V)	current2 (A)	voltage3 (V)	current3 (A)	voltage4 (V)	current4 (A)	voltage5 (V)	current5 (A)	voltage6 (V)	current6 (A)
12	360	1.86	0.18	1.94	0.19	1.83	0.18	1.88	0.18	2.00	0.19	1.87	0.18
13	390	1.86	0.18	1.92	0.19	1.73	0.17	1.84	0.18	1.99	0.19	1.91	0.19
14	420	1.86	0.18	1.91	0.18	1.67	0.17	1.84	0.18	1.99	0.19	1.95	0.19
15	450	1.86	0.18	1.90	0.19	1.77	0.18	1.88	0.18	1.97	0.19	1.87	0.18
16	480	1.85	0.18	1.93	0.19	1.75	0.18	1.87	0.18	1.99	0.19	1.86	0.18
17	510	1.85	0.18	1.90	0.19	1.70	0.17	1.84	0.18	1.97	0.19	1.92	0.19
18	540	1.84	0.18	1.91	0.19	1.77	0.18	1.86	0.18	1.96	0.19	1.84	0.18
19	570	1.84	0.18	1.89	0.18	1.67	0.17	1.82	0.18	1.94	0.19	1.83	0.18

In [74]:

```
# print a tuple of the shape of the df
plant_df.shape
```

Out[74]:

```
(482, 13)
```

In [75]:

```
# cut off the last two rows of the data to limit data to exactly
10 days (480 data points)
plant_df = plant_df.iloc[:-2].reset_index(drop=True)
```

In [76]:

```
# and confirm it worked
plant_df
```

Out[76]:

	time(min)	voltage1 (V)	current1 (A)	voltage2 (V)	current2 (A)	voltage3 (V)	current3 (A)	voltage4 (V)	current4 (A)	voltage5 (V)	current5 (A)	voltage6 (V)	current6 (A)
0	0	4.42	0.29	4.46	0.27	3.84	0.24	2.75	0.21	2.33	0.21	3.51	0.26
1	30	2.30	0.21	2.31	0.21	2.16	0.20	3.28	0.25	2.46	0.22	3.62	0.25
2	60	3.05	0.24	2.21	0.21	2.07	0.19	2.80	0.23	3.88	0.27	2.17	0.20
3	90	2.56	0.22	2.84	0.23	2.59	0.21	2.20	0.20	3.06	0.23	2.19	0.21
4	120	2.07	0.19	2.14	0.20	2.02	0.19	2.02	0.19	2.23	0.20	2.10	0.20
...	...	...	...	...	...	...	...	...	...	...	...	...	...
475	14250	1.97	0.19	1.95	0.19	1.90	0.18	1.87	0.18	1.98	0.19	1.87	0.18
476	14280	3.71	0.28	3.05	0.23	2.66	0.21	2.20	0.20	3.53	0.24	3.01	0.23
477	14310	1.89	0.19	1.91	0.19	1.87	0.18	1.82	0.18	1.93	0.19	1.88	0.18

	time (min)	voltage1 (V)	current1 (A)	voltage2 (V)	current2 (A)	voltage3 (V)	current3 (A)	voltage4 (V)	current4 (A)	voltage5 (V)	current5 (A)	voltage6 (V)	current6 (A)
478	1434.40	1.95	0.19	1.93	0.19	1.88	0.18	1.84	0.18	1.95	0.19	1.89	0.18
479	1437.70	1.92	0.18	1.89	0.18	1.86	0.18	1.83	0.18	1.91	0.19	1.86	0.18

480 rows × 13 columns

In [77]:

```
# rename columns for angles
plant_df.rename(columns={
    'voltage1 (V)': 'voltage_control (V)',
    'current1 (A)': 'current_control (A)',
    'voltage2 (V)': 'voltage_60deg (V)',
    'current2 (A)': 'current_60deg (A)',
    'voltage3 (V)': 'voltage_80deg (V)',
    'current3 (A)': 'current_80deg (A)',
    'voltage4 (V)': 'voltage_90deg (V)',
    'current4 (A)': 'current_90deg (A)',
    'voltage5 (V)': 'voltage_20deg (V)',
    'current5 (A)': 'current_20deg (A)',
    'voltage6 (V)': 'voltage_0deg (V)',
    'current6 (A)': 'current_0deg (A)',
}, inplace=True)
```

In [79]:

```
# create a 'time (hr)' column
plant_df['time (hr)'] = plant_df['time (min)'] / 60

# move the 'time (hr)' column to after 'time (min)'
cols = list(plant_df.columns)
cols.remove('time (hr)')
cols.insert(cols.index('time (min)') + 1, 'time (hr)')
```

```
# reorder the columns
plant_df = plant_df[cols]

# display to check it worked
plant_df.head()
Out[79]:
```

	time (min)	time (hr)	voltage_control (V)	current_control (A)	voltage_60deg (V)	current_60deg (A)	voltage_80deg (V)	current_80deg (A)	voltage_90deg (V)	current_90deg (A)	voltage_20deg (V)	current_20deg (A)	voltage_0deg (V)	current_0deg (A)
0	0	0.0	4.42	0.29	4.46	0.27	3.84	0.24	2.75	0.21	2.33	0.21	3.51	0.26
1	30	0.5	2.30	0.21	2.31	0.21	2.16	0.20	3.28	0.25	2.46	0.22	3.62	0.25
2	60	1.0	3.05	0.24	2.21	0.21	2.07	0.19	2.80	0.23	3.88	0.27	2.17	0.20
3	90	1.5	2.56	0.22	2.84	0.23	2.59	0.21	2.20	0.20	3.06	0.23	2.19	0.21
4	120	2.0	2.07	0.19	2.14	0.20	2.02	0.19	2.02	0.19	2.23	0.20	2.10	0.20

```
In [80]:


In [81]:
# replace NaN values with 0 in rows 3 and 4
height_df.iloc[3:5] = height_df.iloc[3:5].fillna(0)

In [82]:
# identify the average height (cm) columns
avg_cm_cols = [col for col in height_df.columns if
col.endswith("average height (cm)")]
```

```
# create a copy of the DataFrame to modify
df_updated = height_df.copy()

# loop through each average height (cm) column
for col in avg_cm_cols:
    # calculate change in height
    height_series = height_df[col]
    valid_heights = height_series.replace(0, np.nan).ffill()
    delta = valid_heights.diff().fillna(0)

    # create new column name for the delta
    delta_col = col.replace("average height (cm)", "Δ height
(cm)").strip()

    # find the insertion point (after the current column)
    col_index = df_updated.columns.get_loc(col)

    # insert the delta column right after the current average
height (cm) column
    df_updated.insert(col_index + 1, delta_col, delta)

df_height_updated = df_updated

In [ ]:

DATA VISUALIZATION:
In [83]:
plant_df.describe()
Out[83]:
```

	time (min)	time (hr)	voltage_control (V)	current_control (A)	voltage_60deg (V)	current_60deg (A)	voltage_80deg (V)	current_80deg (A)	voltage_90deg (V)	current_90deg (A)	voltage_20deg (V)	current_20deg (A)	voltage_0deg (V)	current_0deg (A)
count	480.000000	480.000000	480.0000	480.0000	480.000000	480.000000	480.000000	480.000000	480.000000	480.000000	480.000000	480.000000	480.000000	480.000000

	time (min)	time (hr)	voltage_control (V)	current_control (A)	voltage_60deg (V)	current_60deg (A)	voltage_80deg (V)	current_80deg (A)	voltage_90deg (V)	current_90deg (A)	voltage_20deg (V)	current_20deg (A)	voltage_0deg (V)	current_0deg (A)
mean	7185.000000	119.750000	1.366333	0.125042	1.349021	0.121396	1.279979	0.116896	1.292229	0.122083	1.405188	0.124271	1.263417	0.117500
std	4161.249812	69.354164	1.136691	0.100029	1.142862	0.096638	1.069744	0.093179	1.052053	0.097224	1.190480	0.098098	1.047514	0.093156
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	3592.500000	59.875000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	7185.000000	119.750000	1.885000	0.180000	1.900000	0.190000	1.850000	0.180000	1.860000	0.180000	1.935000	0.190000	1.840000	0.180000
75%	10777.500000	179.625000	2.130000	0.200000	2.090000	0.200000	2.020000	0.190000	2.070000	0.200000	2.170000	0.200000	1.970000	0.190000

	time (min)	time (hr)	voltage_control (V)	current_control (A)	voltage_60deg (V)	current_60deg (A)	voltage_80deg (V)	current_80deg (A)	voltage_90deg (V)	current_90deg (A)	voltage_20deg (V)	current_20deg (A)	voltage_0deg (V)	current_0deg (A)
max	1437.000000	239.500000	4.420000	0.330000	4.590000	0.300000	4.030000	0.280000	3.550000	0.270000	4.570000	0.300000	3.840000	0.270000

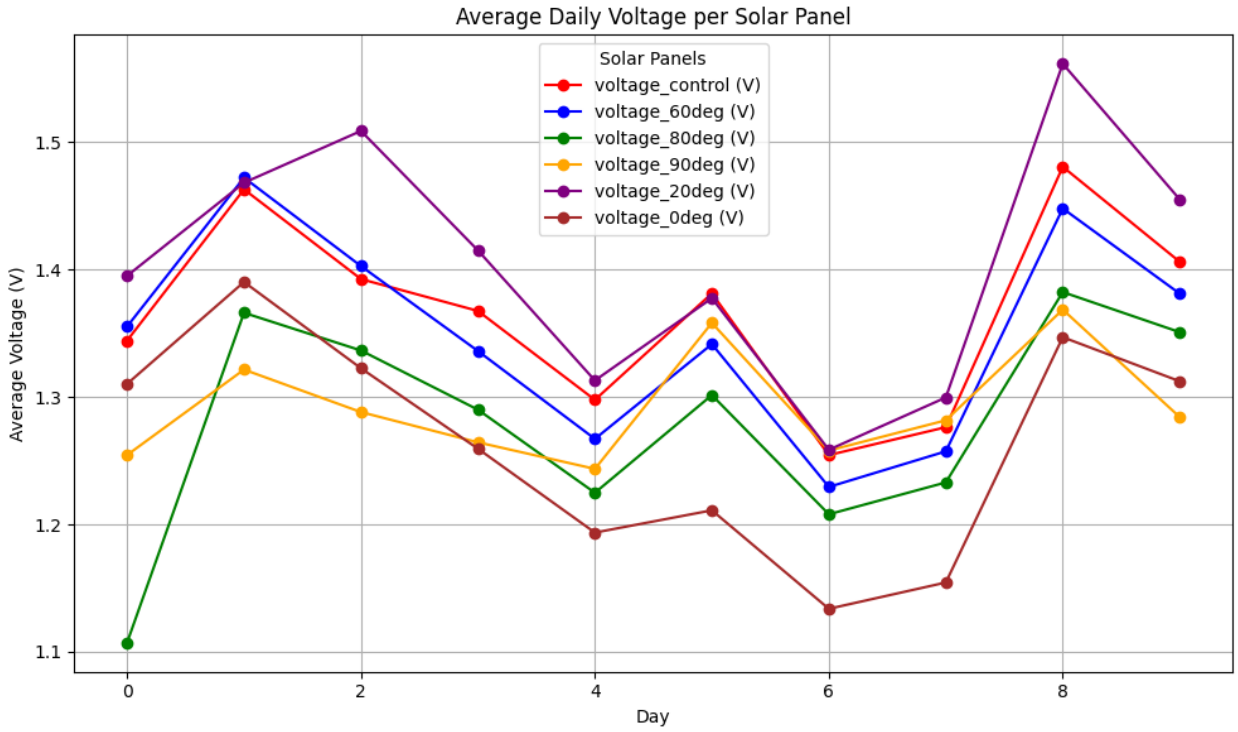
In [84]:

```
# voltage columns
voltage_columns = ['voltage_control (V)', 'voltage_60deg (V)',
'voltage_80deg (V)',
                   'voltage_90deg (V)', 'voltage_20deg (V)',
'voltage_0deg (V)']

# define custom colors
colors = ['red', 'blue', 'green', 'orange', 'purple', 'brown']

plt.figure(figsize=(10, 6))

# plot each voltage with a different color
for col, color in zip(voltage_columns, colors):
    plt.plot(plant_df['time (min)'], plant_df[col], 'o',
label=col, color=color)

# labels and legend
plt.xlabel('Time (min)')
plt.ylabel('Voltage (V)')
plt.title('Voltage Readings Over Time')
plt.legend(title='Voltage Channels')
plt.grid(True)
plt.tight_layout()
plt.show()
```

In [85]:

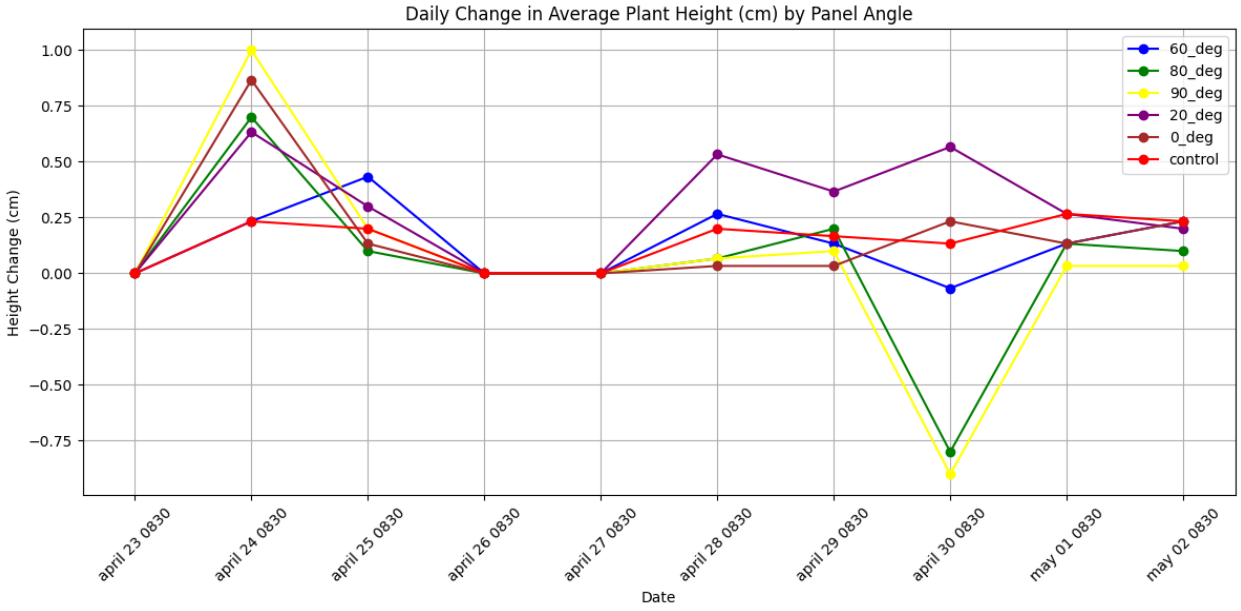
```
# add a 'day' column (assuming time starts at 0 and increments
by 30 minutes)
plant_df['day'] = plant_df['time (min)'] // (24 * 60)  # 24
hours * 60 minutes

# voltage columns
voltage_columns = ['voltage_control (V)', 'voltage_60deg (V)',
'voltage_80deg (V)',
                   'voltage_90deg (V)', 'voltage_20deg (V)',
'voltage_0deg (V)']

# group by day and calculate mean voltage per panel
daily_avg_voltage =
plant_df.groupby('day')[voltage_columns].mean()

# plot
plt.figure(figsize=(10, 6))
colors = ['red', 'blue', 'green', 'orange', 'purple', 'brown']

for col, color in zip(voltage_columns, colors):
    plt.plot(daily_avg_voltage.index, daily_avg_voltage[col],
marker='o', label=col, color=color)

plt.xlabel('Day')
plt.ylabel('Average Voltage (V)')
plt.title('Average Daily Voltage per Solar Panel')
```

```
plt.legend(title='Solar Panels')
plt.grid(True)
plt.tight_layout()
plt.show()

plant_df['day'] = plant_df['time (min)'] // (24 * 60)  # 24
hours * 60 minutes
```




```
In [86]:
# define x-axis labels
dates = df_height_updated['date']

# correct column names and color mapping
change_columns = {
    '60_deg': ('60_deg Δ height (cm)', 'blue'),
    '80_deg': ('80_deg Δ height (cm)', 'green'),
    '90_deg': ('90_deg Δ height (cm)', 'yellow'),
    '20_deg': ('20_deg Δ height (cm)', 'purple'),
    '0_deg':  ('0_deg Δ height (cm)', 'brown'),
    'control': ('control Δ height (cm)', 'red')
}

# create the plot
plt.figure(figsize=(12, 6))

for label, (col, color) in change_columns.items():
    plt.plot(dates, df_height_updated[col], marker='o',
color=color, label=label)
```

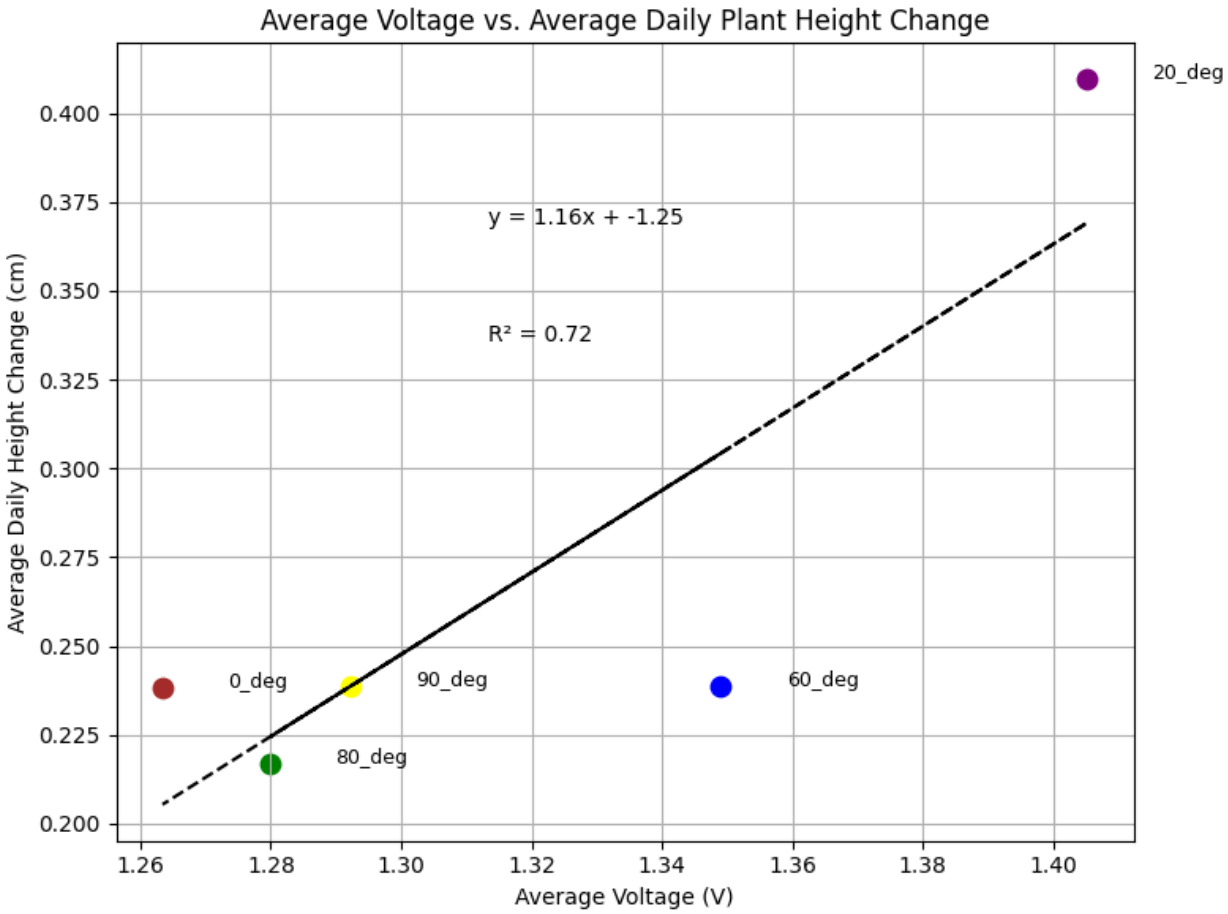
```
# plot formatting
plt.title("Daily Change in Average Plant Height (cm) by Panel
Angle")
plt.xlabel("Date")
plt.ylabel("Height Change (cm)")
plt.xticks(rotation=45)
plt.grid(True)
plt.legend()
plt.tight_layout()

plt.show()
```



In [87]:

```
# color mapping
color_map = {
    '60_deg': 'blue',
    '80_deg': 'green',
    '90_deg': 'yellow',
    '20_deg': 'purple',
    '0_deg':  'brown',
}

# prepare data for plotting
voltages = np.array([avg_voltage[angle] for angle in color_map])
growth = np.array([avg_height_change[angle] for angle in
color_map])
labels = list(color_map.keys())
colors = [color_map[label] for label in labels]

# linear regression
```

```
slope, intercept, r_value, p_value, std_err =
linregress(voltages, growth)
best_fit_line = slope * voltages + intercept

# plotting
plt.figure(figsize=(8, 6))

# plot each point with corresponding color
for i, label in enumerate(labels):
    plt.scatter(voltages[i], growth[i], color=colors[i], s=80)
    plt.text(voltages[i] + 0.01, growth[i], label, fontsize=9)

# plot regression line
plt.plot(voltages, best_fit_line, color='black', linestyle='--',
label='Best Fit Line')

# equation and R²
eq_text = f'y = {slope:.2f}x + {intercept:.2f}'
r2_text = f'R² = {r_value**2:.2f}'
plt.text(min(voltages) + 0.05, max(growth) * 0.9, eq_text,
fontsize=10)
plt.text(min(voltages) + 0.05, max(growth) * 0.82, r2_text,
fontsize=10)

# final formatting
plt.title("Average Voltage vs. Average Daily Plant Height
Change")
plt.xlabel("Average Voltage (V)")
plt.ylabel("Average Daily Height Change (cm)")
plt.grid(True)
plt.tight_layout()
plt.show()
```


Average Voltage vs. Average Daily Plant Height Change
y = 1.16x + -1.25
R² = 0.72
0_deg
90_deg
60_deg
20_deg
80_deg
Average Daily Height Change (cm)
Average Voltage (V)
0.400
0.375
0.350
0.325
0.300
0.275
0.250
0.225
0.200
1.26
1.28
1.30
1.32
1.34
1.36
1.38
1.40