ATTITUDE DETERMINATION CONTROL SYSTEM

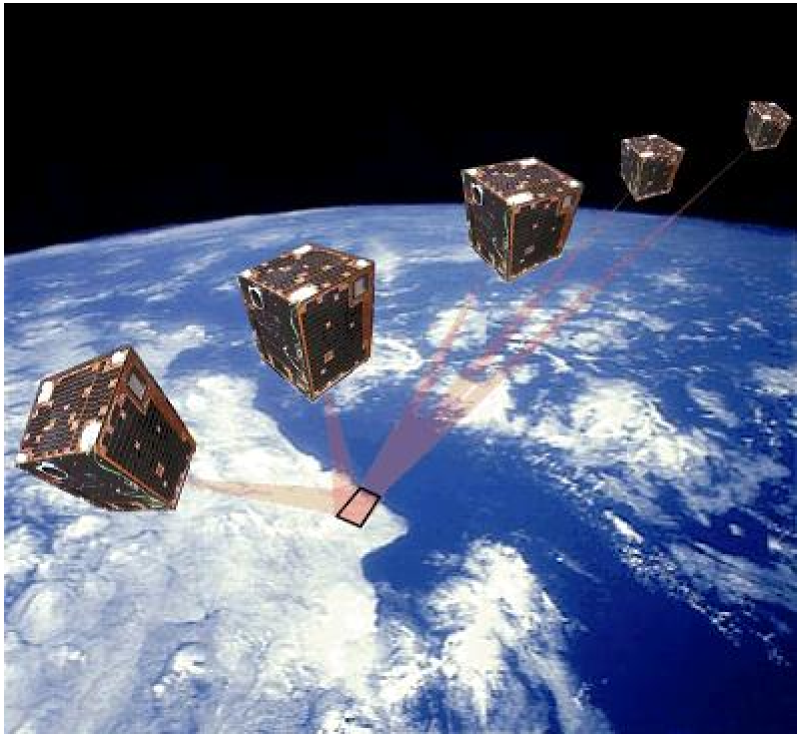
Team Alex: Andrew Hayman, Connor Bayers, Kyle Singer, Ben Wylynko, Matt Wright

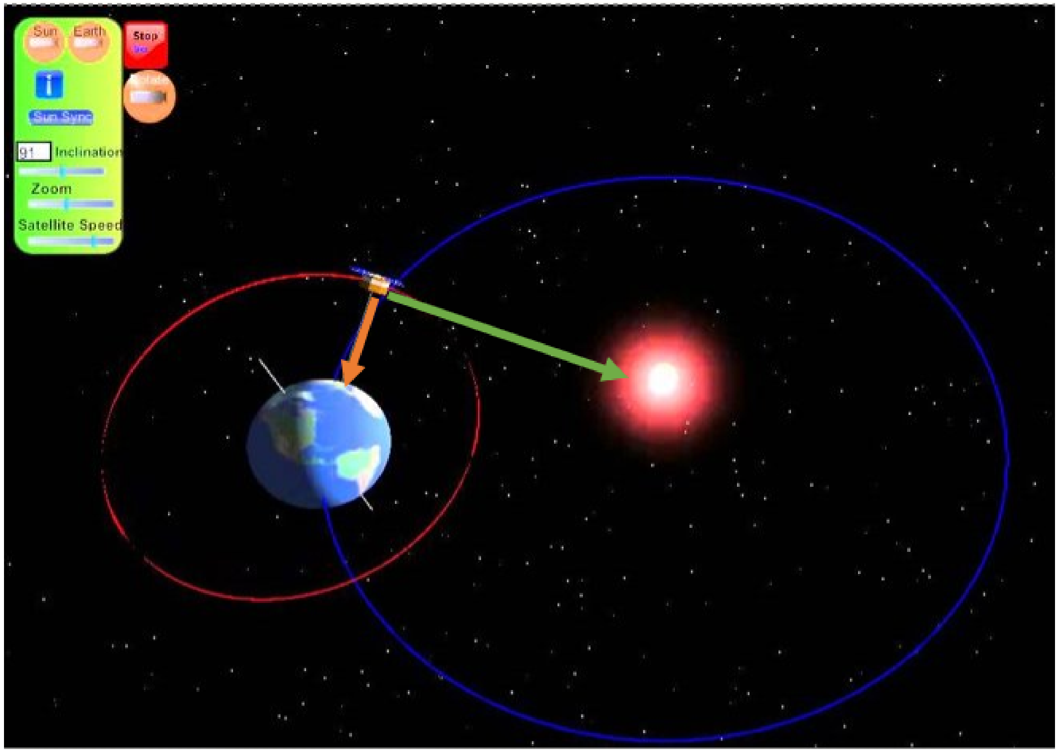
WHAT IS AN ADCS?

WHAT IS AN ADCS?

An Attitude Dertermination and Control System (ADCS) controls the orientation and position of a satellite in space

WHAT IS AN ADCS?





OUR GOAL

OUR GOAL

- Build a low-cost prototype ADCS that can be used as an education tool, which is easily approachable by design teams and EiTs
- Achieve space-grade pointing accuracies: 5 degree pointing accuracy for sun vector

MECHANICAL

- Create an efficient, low-friction, single-axis rotation method to simulate low resistance on a satellite
- House the electrical componenets in a 10 cm cubic structure
- Balance all internal components to level the moment of inertia for the entire body of the ADCS

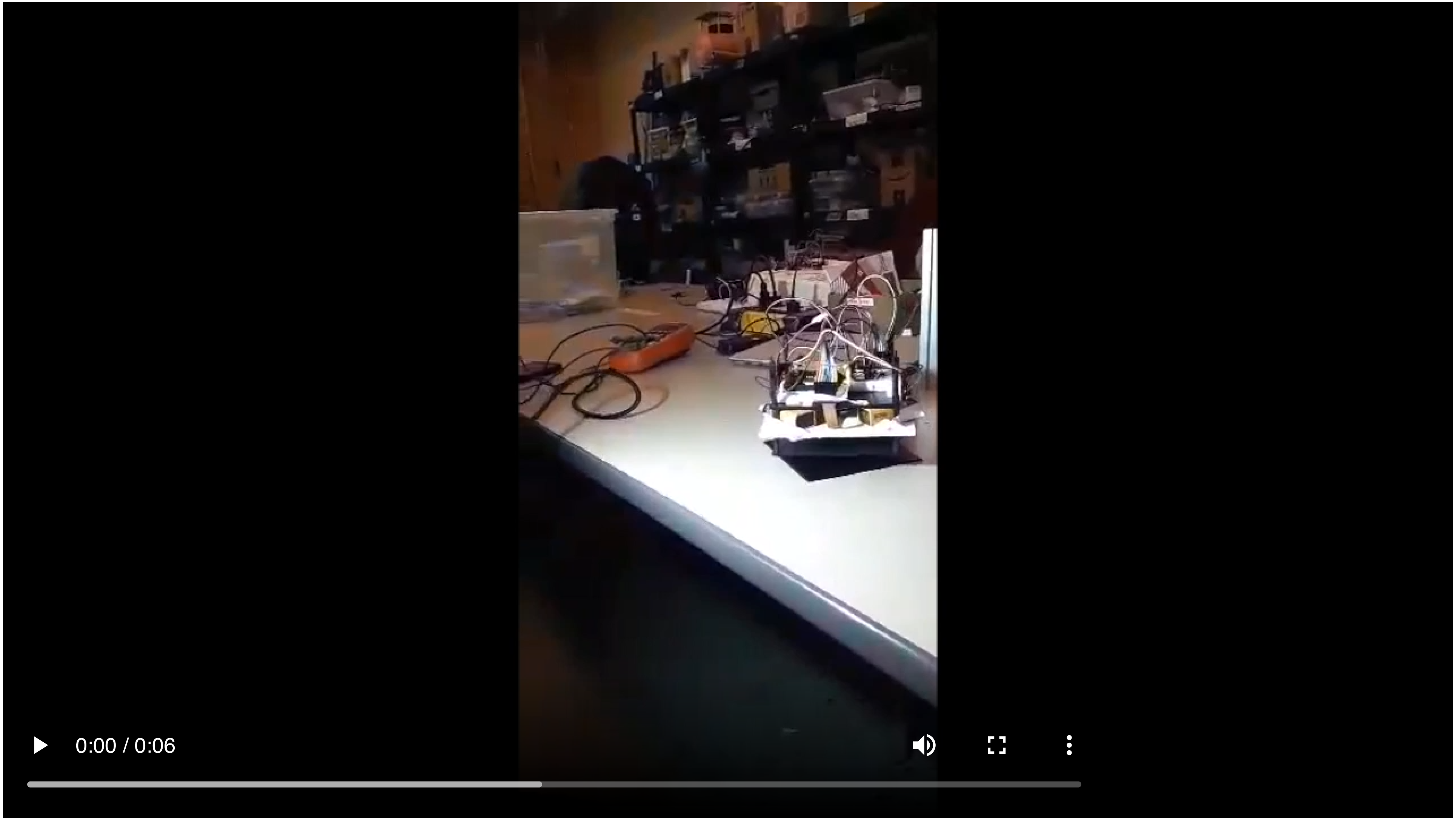
ELECTRICAL

- Determine the sun angle for resolving the satellite's attitude
- Use the sun angle (relative to the satellite) to orient the system towards the sun
  - Critical for solar power charging

SOFTWARE

- Implement magnetoquers such that they can de-tumble the ADCS
- Control the system to consistently maintain perpindicular orientation towards a light source

OUR PROGRESS

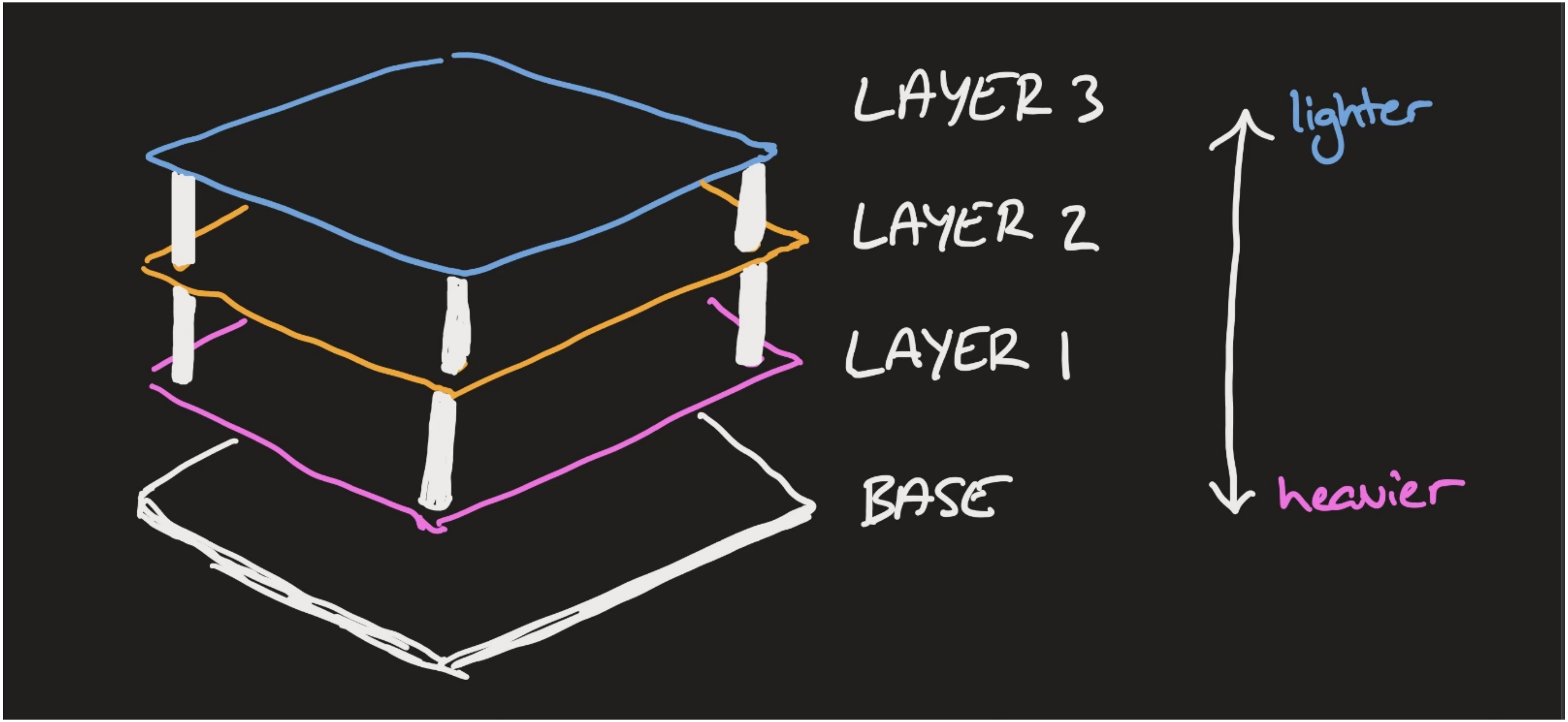

0:00 / 0:06

MECHANICAL

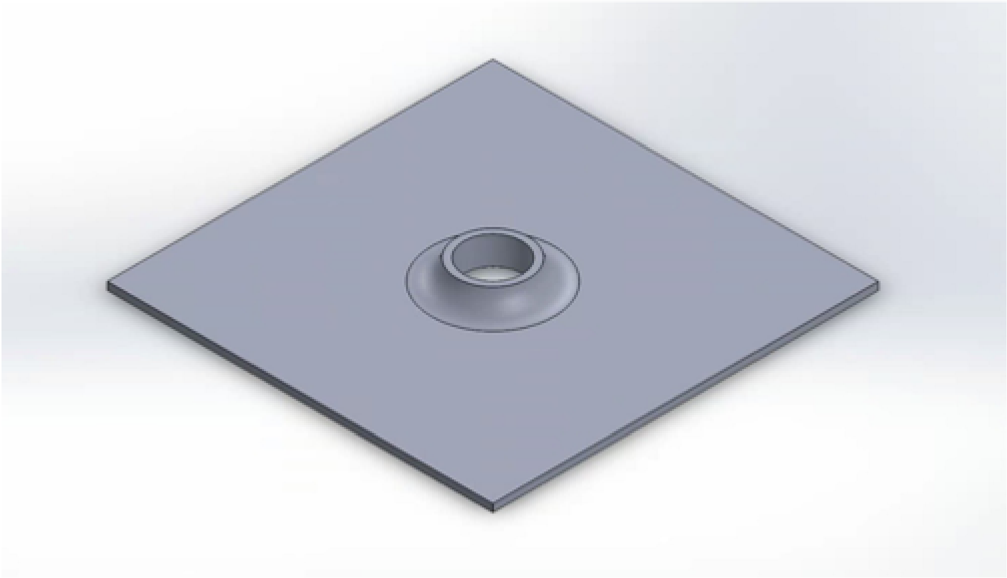
PROGRESS

- Multiple plates have been 3D printed with M3 standoffs separating each one
- Full structure is set between two sets of paired ring bearings
- Each bearing rests in a circular impression and are connected by plastic posts maximizing freedom of rotation

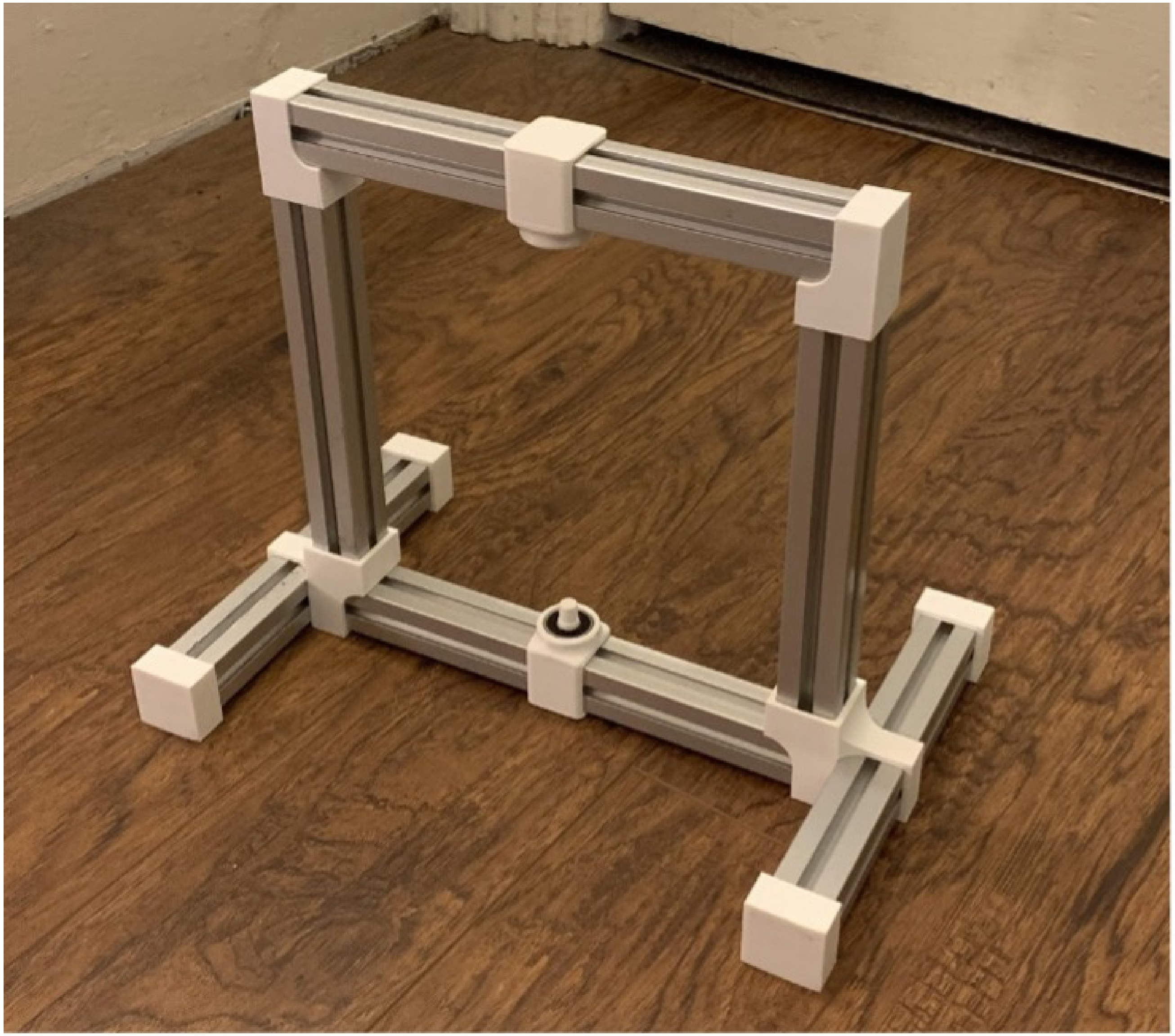
PROGRESS



PROGRESS



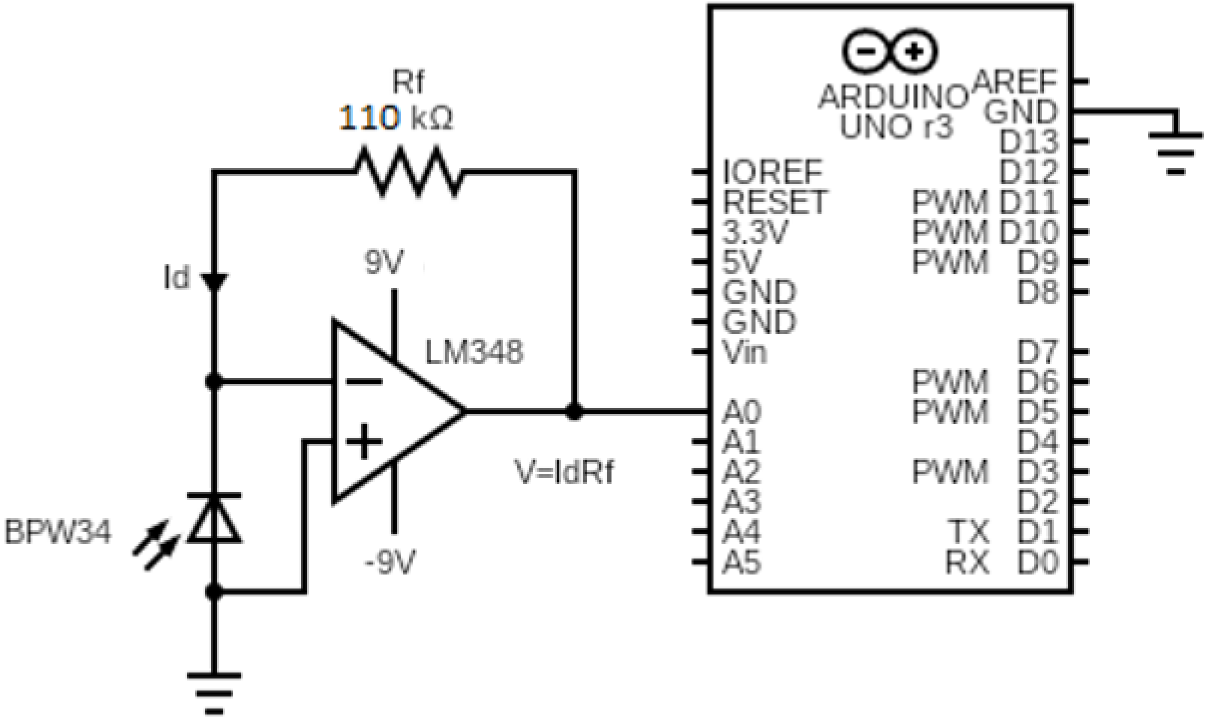
PROGRESS



ELECTRICAL

PROGRESS

- BPW34 photodiode with transimpedance amplifier

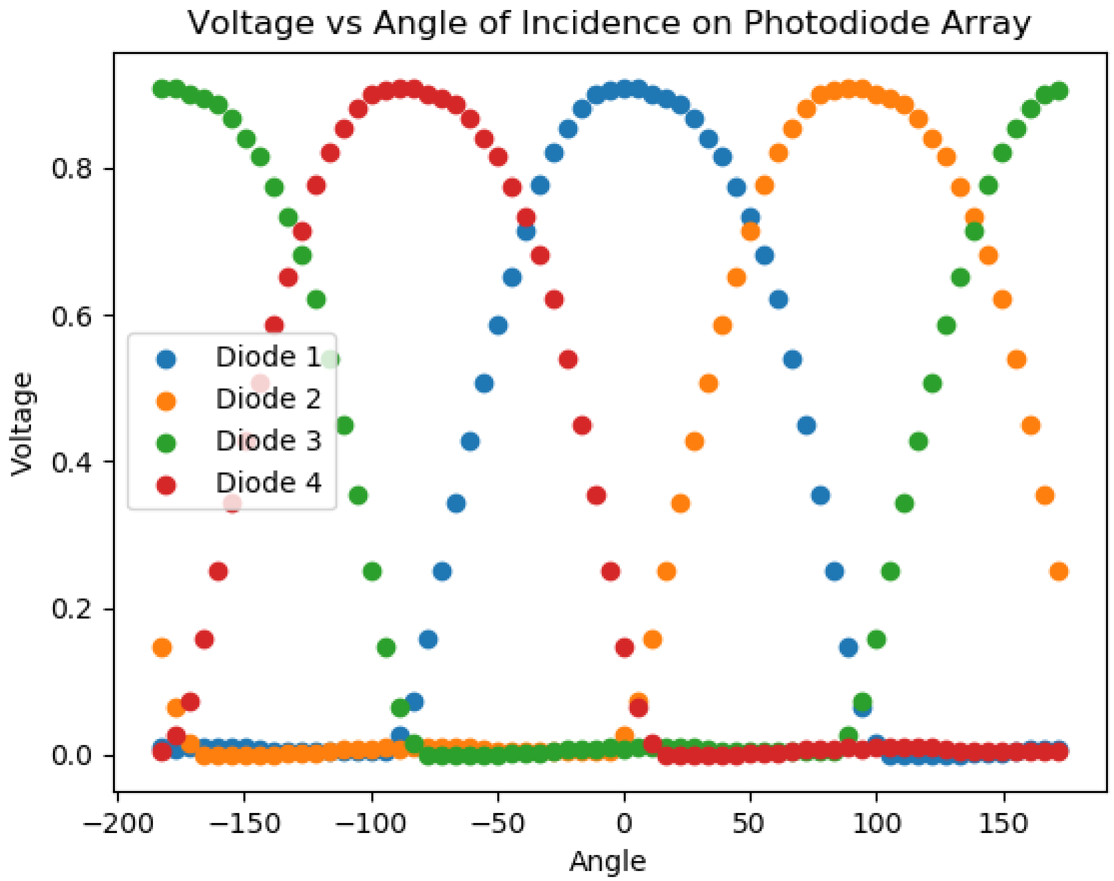


- Circuit was built four times for four photodiodes

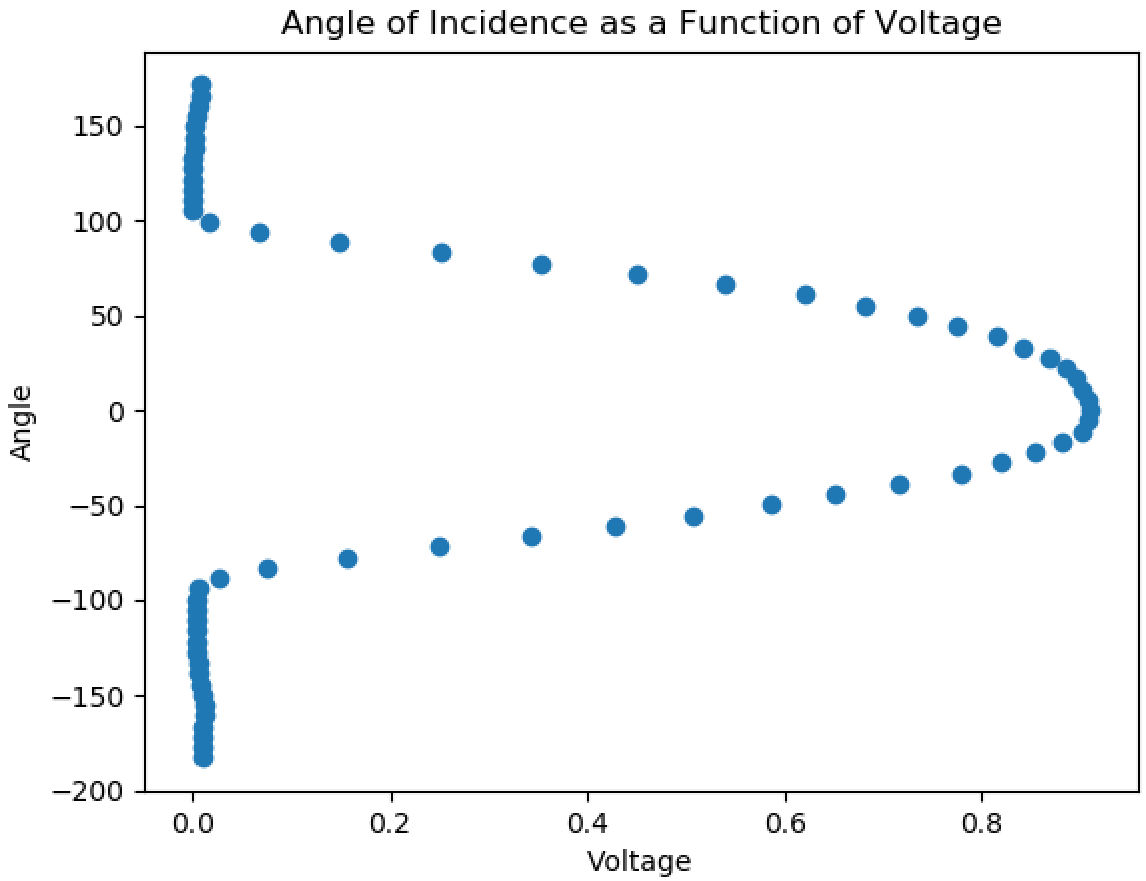
PROGRESS

- Using a 360° voltage sweep, each photodiode's regression model is calibrated
- Combining computed angles allows for a 360° angle of incidence calculation
- PoC sun tracking capabilites implement with coarse sun sensors
- Initial testing shows that angle estimation has significant noise around the normal

PROGRESS

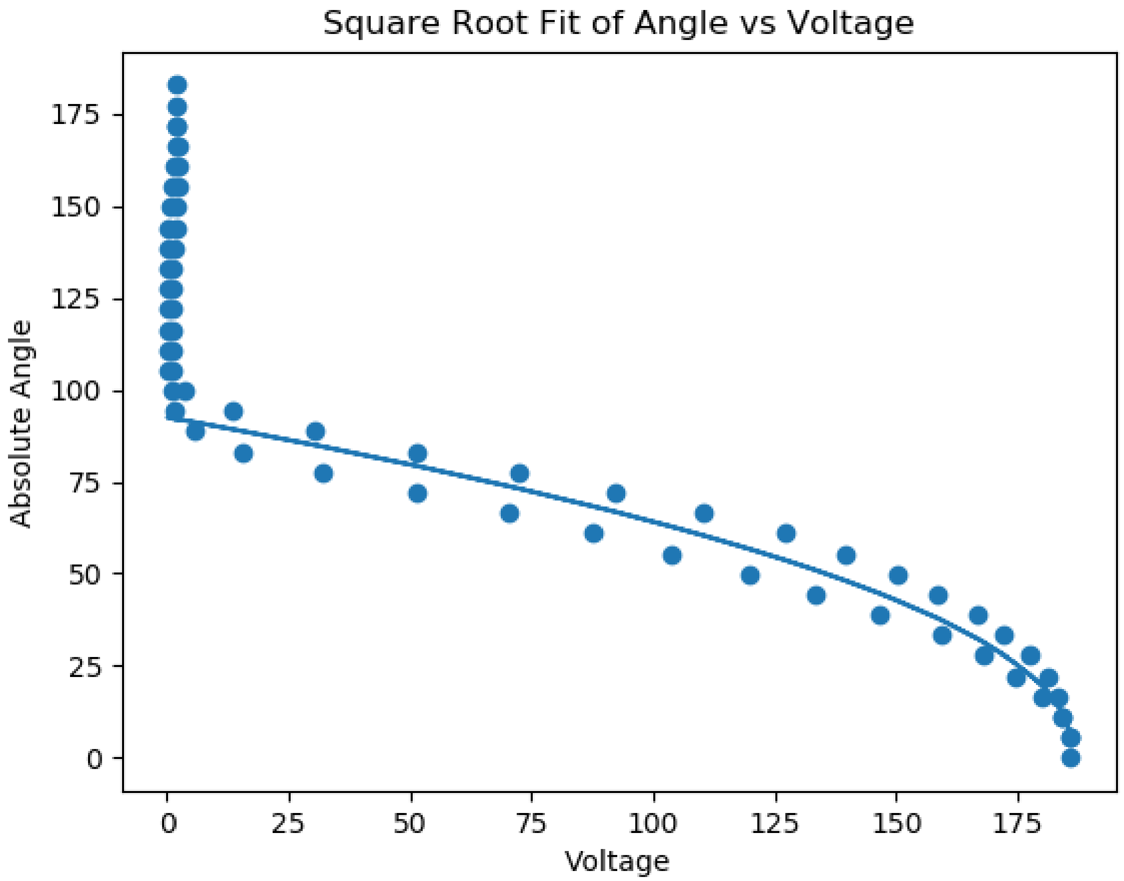

Voltage vs Angle of Incidence on Photodiode Array
Voltage
Angle
0.0
0.2
0.4
0.6
0.8
−200
−150
−100
−50
0
50
100
150
Diode 1
Diode 2
Diode 3
Diode 4

PROGRESS



Angle of Incidence as a Function of Voltage
Angle
Voltage
150
100
50
0
−50
−100
−150
−200
0.0
0.2
0.4
0.6
0.8

PROGRESS

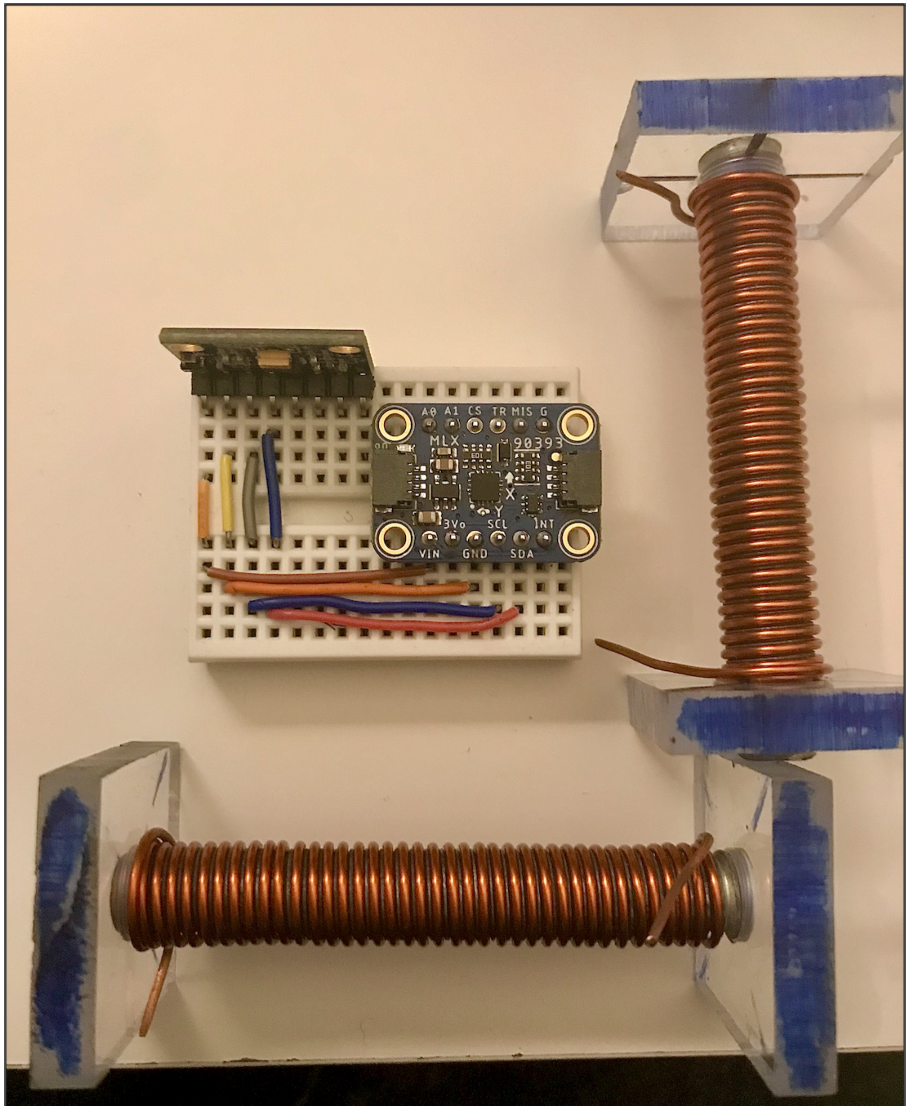

Square Root Fit of Angle vs Voltage
Absolute Angle
Voltage
0
25
50
75
100
125
150
175

COMPUTER

MAGNETORQUERS

- Used for de-tumbling the satellite
- Current (A) values outputted from Arduino when system rotates
- Magnetorquers have been created and wrapped in copper wire

MAGNETORQUERS



MAGNETORQUERS

```
#include <Wire.h>
#include "Adafruit_MLX90393.h"
#include <MPU6050.h>
#define MLX90393_CS 10

Adafruit_MLX90393 sensor = Adafruit_MLX90393();
MPU6050 mpu;

float currentThroughCoils[3]; // 3 column array (3D vector ->
float k = 46040; // going to need a relative magnitude value
float kCurrent = 1; //current used to calculate k -> 1 A
float magneticField[3]; // earth's magnetic field -> 3 coumn
bool movement; // how are we moving -> detumbling satellite v
float AREA_OF_COILS = PI*pow(0.0045,2);
int NUMBER_OF_TURNS = 61;
```

MAGNETORQUERS

```
#include <Wire.h>
#include "Adafruit_MLX90393.h"
#include <MPU6050.h>
#define MLX90393_CS 10

Adafruit_MLX90393 sensor = Adafruit_MLX90393();
MPU6050 mpu;

float currentThroughCoils[3]; // 3 column array (3D vector ->
float k = 46040; // going to need a relative magnitude value
float kCurrent = 1; //current used to calculate k -> 1 A
float magneticField[3]; // earth's magnetic field -> 3 coumn
bool movement; // how are we moving -> detumbling satellite v
float AREA_OF_COILS = PI*pow(0.0045,2);
int NUMBER_OF_TURNS = 61;
```

MAGNETORQUERS

```
  sensor.setResolution(MLX90393_Z, MLX90393_RES_16);

  // Set oversampling
  sensor.setOversampling(MLX90393_OSR_2);

  // Set digital filtering
  sensor.setFilter(MLX90393_FILTER_6);
}

void currentCalculator(Vector gyro, float mX, float mY, float
  float constant = k*NUMBER_OF_TURNS*AREA_OF_COILS;

  mX = mX*pow(10,-6);
  mY = mY*pow(10,-6);
  mZ = mZ*pow(10,-6);
  float angularVelocity = gyro.XAxis*0.0174533; //rad/s
```

MAGNETORQUERS

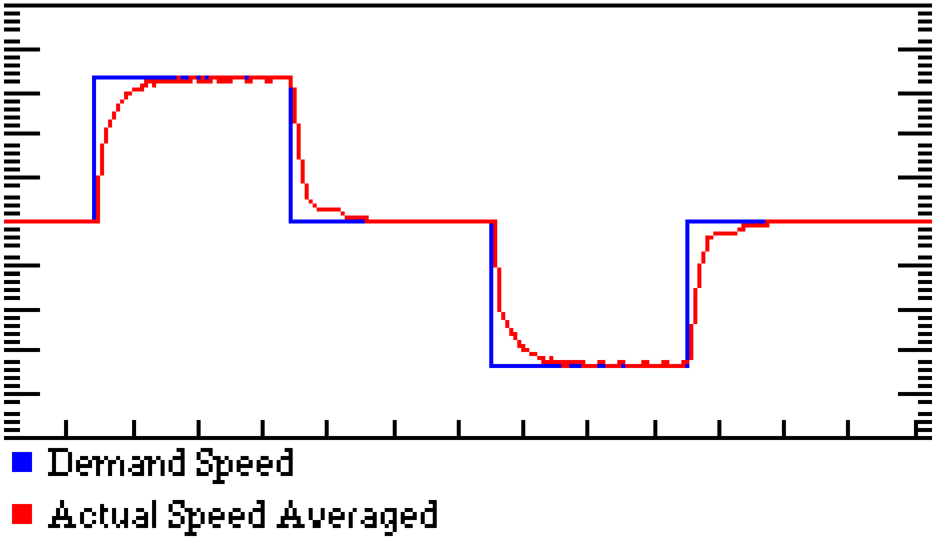
```
  // Set oversampling
  sensor.setOversampling(MLX90393_OSR_2);

  // Set digital filtering
  sensor.setFilter(MLX90393_FILTER_6);
}

void currentCalculator(Vector gyro, float mX, float mY, float
  float constant = k*NUMBER_OF_TURNS*AREA_OF_COILS;

  mX = mX*pow(10,-6);
  mY = mY*pow(10,-6);
  mZ = mZ*pow(10,-6);
  float angularVelocity = gyro.XAxis*0.0174533; //rad/s

```

MAGNETORQUERS

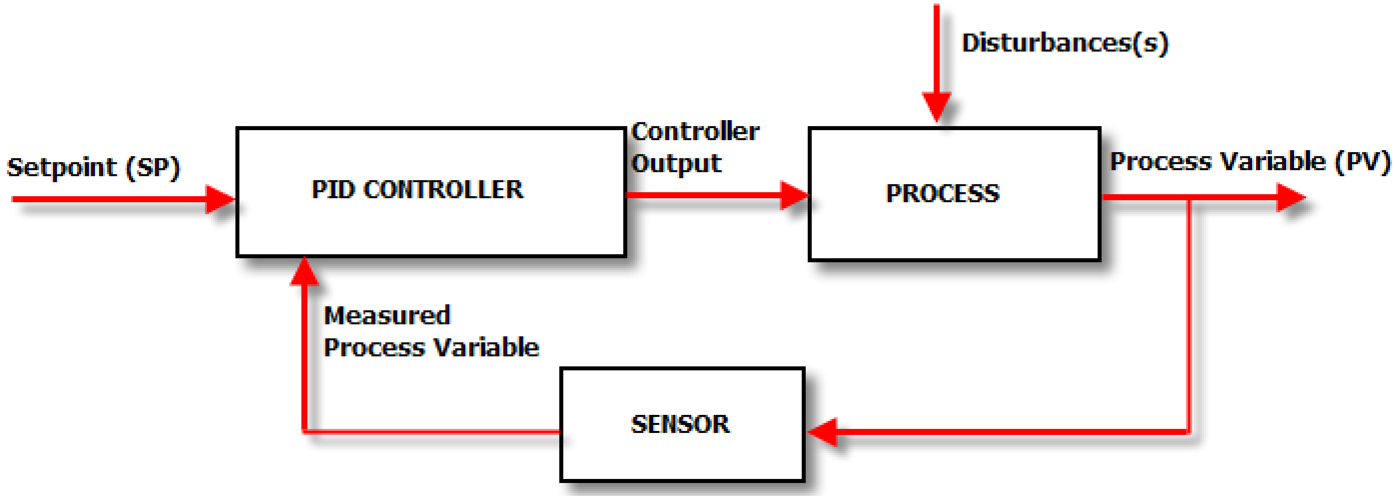
```
void currentCalculator(Vector gyro, float mX, float mY, float
  float constant = k*NUMBER_OF_TURNS*AREA_OF_COILS;

  mX = mX*pow(10,-6);
  mY = mY*pow(10,-6);
  mZ = mZ*pow(10,-6);
  float angularVelocity = gyro.XAxis*0.0174533; //rad/s

  currentThroughCoils[0] = constant*mY*angularVelocity;
  currentThroughCoils[1] = constant*mX*angularVelocity;


  if (currentThroughCoils[0]+currentThroughCoils[1] > kCurren
  float magnitude = sqrt(pow(currentThroughCoils[0],2) + pow(
  currentThroughCoils[0] = currentThroughCoils[0]/magnitude*k
  currentThroughCoils[1] = currentThroughCoils[1]/magnitude*k
```

MOTORS

- Conservation of angular momentum calculations to determine required RPM
- Commands are given to the motor
- Motor's speed rises to meet the request



PID CONTROLLER



Set point: Photodiode voltage threshold
Sensor input: current photodiode reading
Output: PWM duty cycle for motor speed

SAFETY

- Soldering and de-soldering using ventilation and protective glasses
- Not over discharging LiPo battery

NEXT STEPS

MECHANICAL & POWER

ITERATIVE (WEEKS 10-12):

- Some aspects of the design may cause mechanical drag with limited time to fix the problem
- 3D printed pieces need to be managed and well-documented for future use
- Update power system to use 3S LiPo battery with capacitor and fuses

CONTROL SYSTEM

WEEK 10

- Develop testing plan to validate angle prediction and sun tracking accuracy
- Research and identify methods to power op-amp from single 12V source
- Implementing DACs with magnetorquer system

CONTROL SYSTEM

WEEK 11

- Design calibration protocol for sun sensors for when satellite is spinning in space
- Sampling photodiodes at specific frequencies based on angular velocity
- Testing time for ADCS to slow down with magnetorquer system running
  - Magnets needed to increase the magnetic field surrounding the ADCS

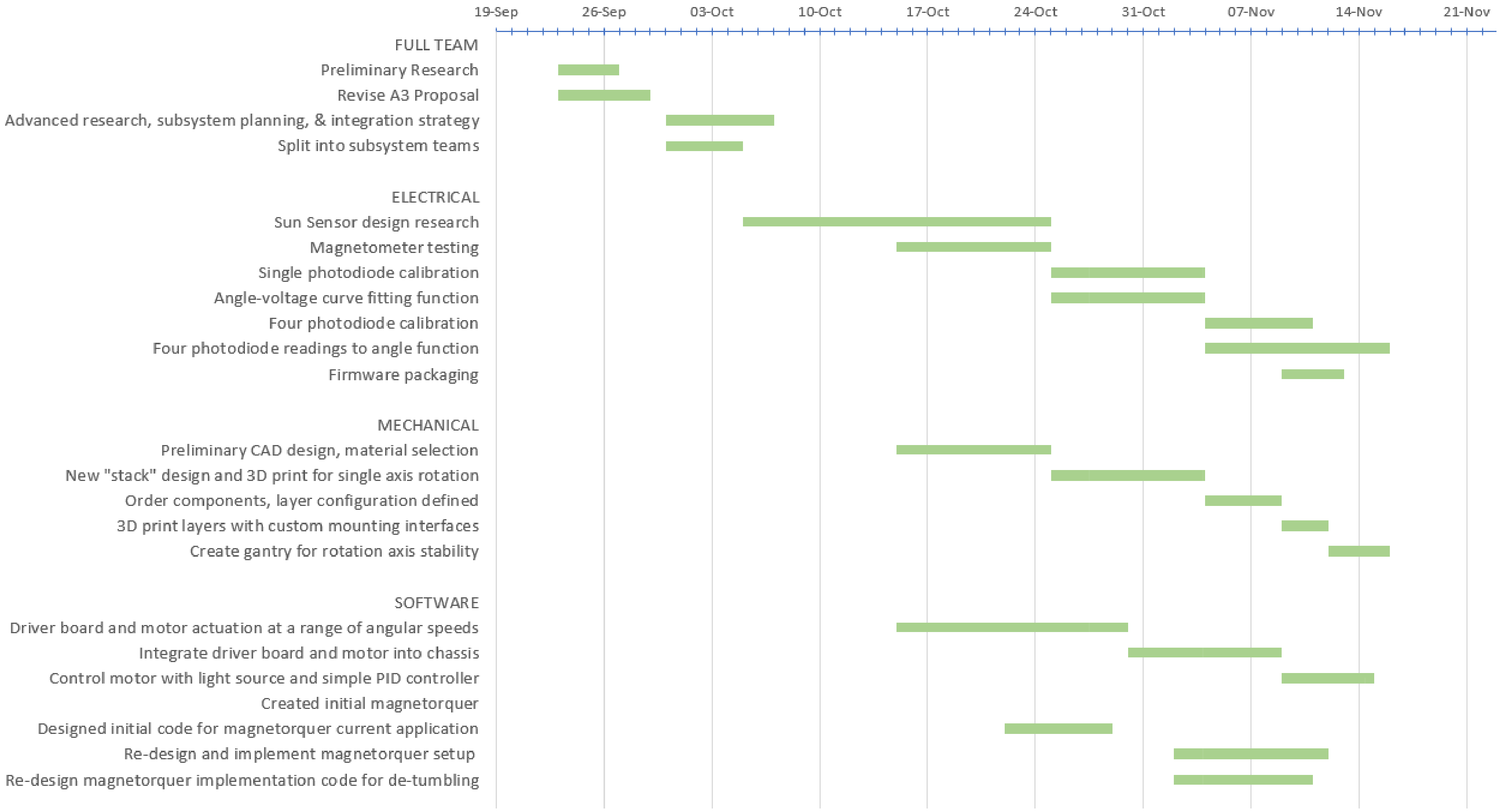
CONTROL SYSTEM

ITERATIVE (WEEKS 10-12):

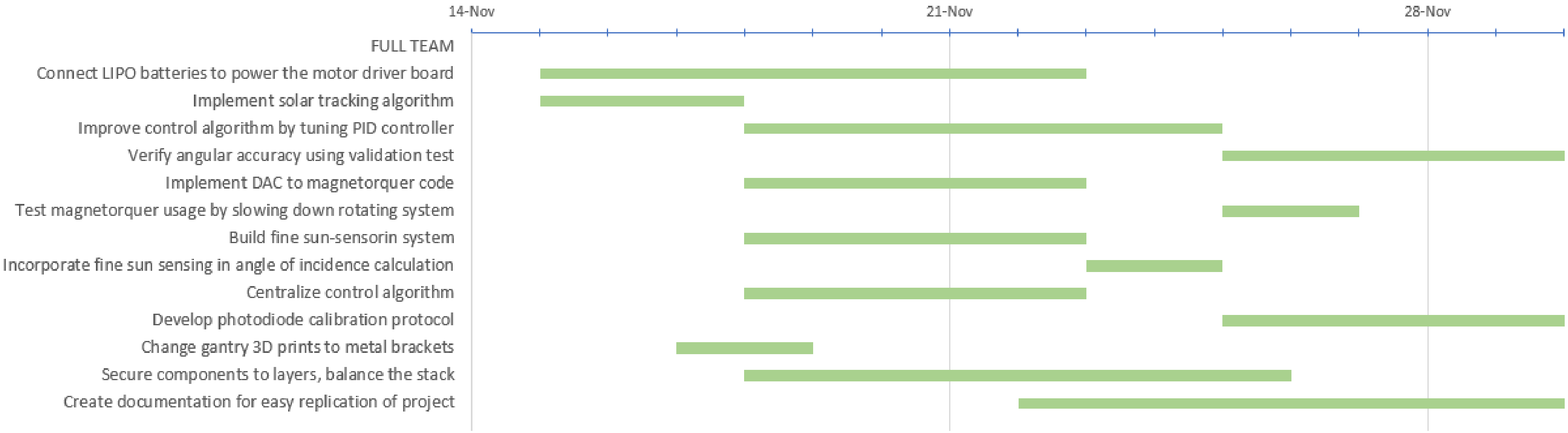
- Design a fine tune sun sensing system
- Adding a three dimensional light vector estimation
- Integrating motors with the four photodiode system
- Tuning control algorithm for further accuracy

SUMMARY

COMPLETED


19-Sep
26-Sep
03-Oct
10-Oct
17-Oct
24-Oct
31-Oct
07-Nov
14-Nov
21-Nov
FULL TEAM
Preliminary Research
Revise A3 Proposal
Advanced research, subsystem planning, & integration strategy
Split into subsystem teams
ELECTRICAL
Sun Sensor design research
Magnetometer testing
Single photodiode calibration
Angle-voltage curve fitting function
Four photodiode calibration
Four photodiode readings to angle function
Firmware packaging
MECHANICAL
Preliminary CAD design, material selection
New "stack" design and 3D print for single axis rotation
Order components, layer configuration defined
3D print layers with custom mounting interfaces
Create gantry for rotation axis stability
SOFTWARE
Driver board and motor actuation at a range of angular speeds
Integrate driver board and motor into chassis
Control motor with light source and simple PID controller
Created initial magnetorquer
Designed initial code for magnetorquer current application
Re-design and implement magnetorquer setup
Re-design magnetorquer implementation code for de-tumbling

TO BE COMPLETED




QUESTIONS?