NORTHERN ARIZONA UNIVERSITY

**Department of Mechanical Engineering**
ME486C - 002

To: Dr. Sarah Oman

From: CubeSat Team - Jacob Belin, Richard Campos, Jackie Fonseca, Zack Retzlaff

Date: September 04, 2020

Re: Hardware Review 01

Since the beginning of the semester, we contacted and updated our client on our summer's progress and discussed our plans for this semester, meetings, academic schedule, etc. Internally, the CubeSat's members are responsible for segments of the design and are reporting updates/requesting support bi-weekly. The following document discusses the results of this progress.

## Concept Review

The team started working over the summer on finalizing the CAD model for our project. This is shown below in Figure 1. The updated CAD model highlights the feedback given to the team by General Atomics by taking out two of the vertical counterweight rails. The team opted to go with this design because GA didn't think it was necessary to have four counterweight rails on each corner. Two counterweight rails will do the same job in displacing the center of gravity.

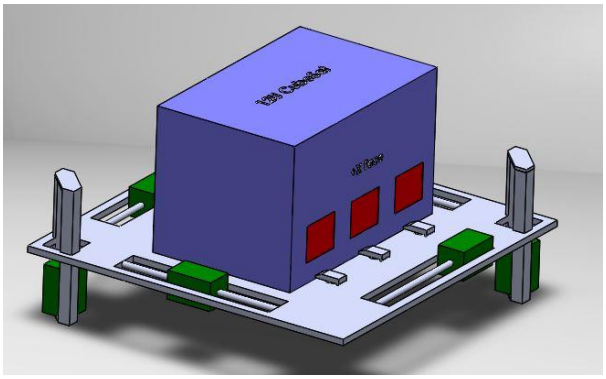

**Figure 1. CubeSat Mounting Fixture Final Design**

## Test Stand and Replica's for Prototyping

Figure 2 shows the CAD model of the CubeSat replica based on the dimensions given to the team by General Atomics. This CubeSat was then modeled in Figure 3 to build a working replica out of plywood. Figure 3 was dimensioned by using the thickness of the plywood as part of the x-direction, y-direction and z-direction measurements. This allowed for ease of manufacturing where 6 sheets of wood were cut to the dimensions in the model and were connected by wood screws.

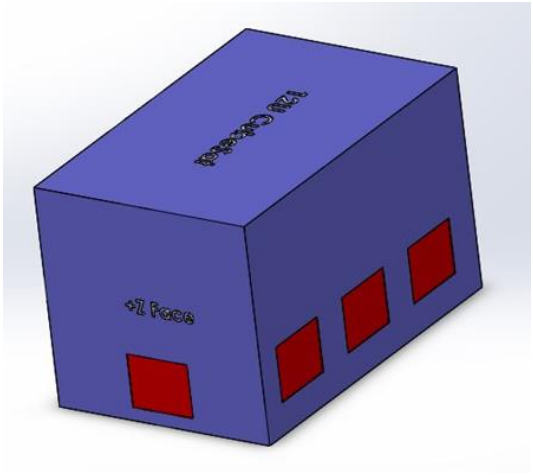

**Figure 2. CubeSat Theoretical CAD Model**

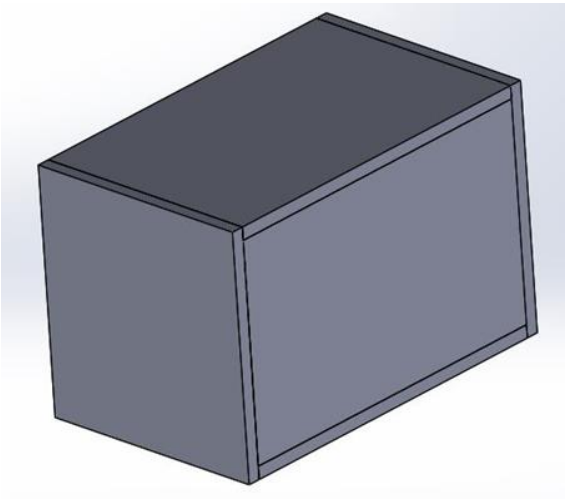

**Figure 3. CAD Model of CubeSat Prototype**

Figure 4 shows the replica CubeSat the team created out of plywood. This was created so the team could perform numerous tests to ensure the fixture would work appropriately. The team will first use this replica for testing the force sensor code with the testing table shown in Figure 5. This table is just a rough model to test the Arduino codes before actually building the final prototype. Figure 6 shows the CubeSat on top of the testing table. The team will insert different payloads in numerous locations in the CubeSat and plans to test the code to ensure it can calculate the center of gravity effectively using force sensors.

**Figure 4. CubeSat Replica**

**Figure 5. CubeSat Testing Table**

**Figure 6. CubeSat Testing Table with CubeSat Replica**

## CG and Code Program Review

The Arduino prototype code has undergone extensive improvements from what was seen in the Self-learning Assignment. The current prototype is still using the Arduino Esplora board with an integrated accelerometer and RGB LED (Figure 7). The initial code consisted of an accelerometer angle reading being converted to a 180 degree range about the x and y axes, and an if-statement that has an on board RGB LED to switch from green to red when the tilt exceeds 30 degrees. In that state, the code could only detect the 30 degree tilt in the x and y directions, which is a draw back considering that one of the customer requirements was for it to be able to work in any direction.

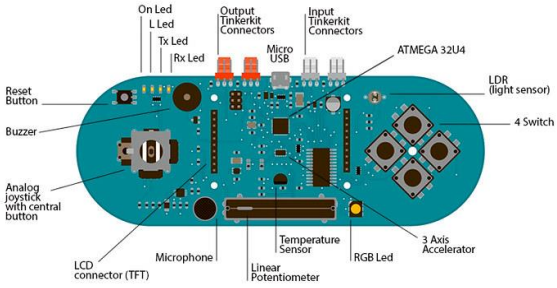


**Figure 7. Arduino Esplora board currently used for prototyping**

To correct this, some vector calculus has been hard-coded into the program. Specifically, the steps required to find the angle between two planes have been coded. It was decided that this was the route to go when team members realized that that was essentially what was being looked for, with one plane being level to the ground, and the other being the plane at which the fixture is at. This process required the calculation of the normal vector of the fixture plane, which was derived and then hard-coded (Eq 1).

$$\vec{\boldsymbol{n}} = (-z_1 * y)\hat{\imath} + (x * z_2)\hat{\jmath} + (x * y)\hat{k} \quad (1)$$

All of the intermediate variables for the components of that normal vector are also being calculated in the program and can be seen in the serial monitor in the Arduino IDE. While significant progress has been made in the program, the current issues with it is that when the angle between the normal vector and the z-axis, it only outputs a value of either 0 or 180 degrees. It is suspected that the issue with this has to do with how the unit x and y vectors in the program have been mapped. Other than that issue, all the intermediate calculations appear to be accurate which gives confidence to the team that the code is moving in the right direction. The current code can be seen in Appendix A.

It is our intent to use the Arduino program to take inputs from the fixture and output configuration for the fixture to adjust the CG accordingly. Among the things to consider for that black body concept is the mass of the counterweights and essentially the approach to solving for CG adjustments. A static analysis of the CubeSat and fixture assembly gave us a free-body diagram. Members added assumptions of the system to calculate a maximum mass for the counterweights. The CG specs of a 12U CubeSat are available, but the team decided this approach designs to a variable payload so that our customer can utilize the fixture for other devices beyond a CubeSat.

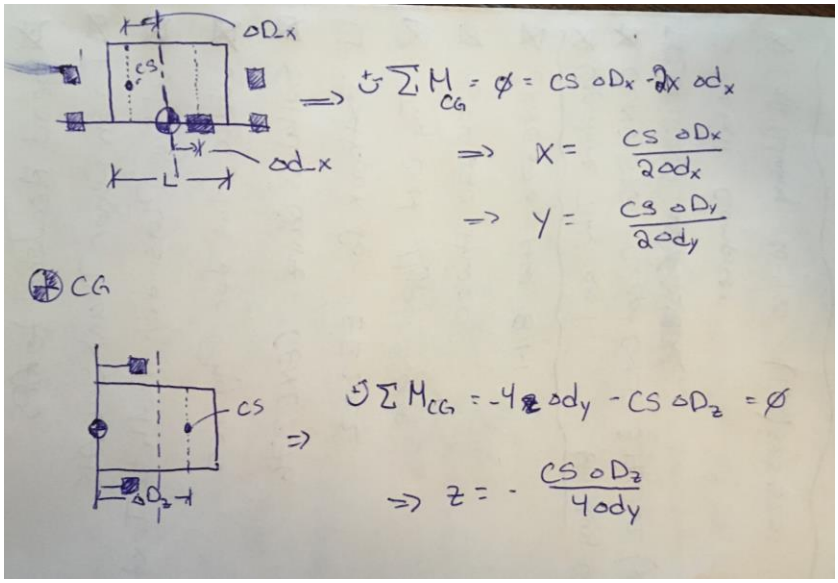


**Figure 8. FBD sketch and sample equations**

The diagrams and approach to this problem was confirmed by a faculty member and also gave recommendation of another approach that the team will review soon. The calculations above need to be translated into a program and the variables need to be assigned the signals from the fixture. This was and is going to be the primary objective of this analysis for the counterweights. If we are able to solve and optimize these equations while simultaneously obtaining more of the material we purchased, our hope is to add this to the test stand prototype from Figure 6.

## Future Plans

The team's plans on to continuously work on the separate projects until purchased items come in. The team will continue to edit and update the 3D Solidworks model of any changes that are made, as well as making edits to fully dimension the overall project. As the team continues and changes are agreed upon this will be reflected within the 3D model. The team will also continue to fully develop a workable code. As mentioned, the code will take in inputs from the fixture and pressure sensors and will output the distance that the counterweights need to be moved in order to adjust the center of gravity to the correct location wanted. Then, as parts start to come in the team will continue the prototyping and testing processes.

As the purchased items get delivered the team will continue the manufacturing of the prototype. The team will first begin with setting up each sub-system. One sub-system is the Pneumatic System. This system is what powers the power screws to spin, therefore powering the counterweights to move. This system will be put together and tested separately to assure there is enough air flow to move the counterweights. The team will then test the Pneumatic System to find an optimal air flow wanted, as well as minimum air flow properties required to power the system. Another sub-system is the entire counterweight system. This system includes the power screw rails, and the counterweight themselves. This is the system that will be adjusting the center of gravity and accomplishing the designs required task. For this system the team will test to assure that the lengths of the power screws are an adequate length. The team will also test the overall ability for this sub system to work. After testing the individual sub systems, the team will combine the components and test the whole fixture.

## Appendix A: Arduino Program

```
#include <Esplora.h>

void setup() {

  Serial.begin(9600);

}

void loop() {

  int x_axis = Esplora.readAccelerometer(X_AXIS);
  int y_axis = Esplora.readAccelerometer(Y_AXIS);
  int z_axis = Esplora.readAccelerometer(Z_AXIS);

      x_axis = map(x_axis, 160, -160, 90, -90);
      y_axis = map(y_axis, 180, -130, 90, -90);

     int x = 100 * cos( 3.14 * x_axis/180);
     int z1 = 100 * sin( 3.14 * x_axis/180);

     int y = 100 * cos( 3.14 * y_axis/180);
     int z2 = 100 * sin( 3.14 * x_axis/180);

      int i = -z1 * y;
      int j = x * z2;
      int k = x * y;

      int mag_n = sqrt(i^2 + j^2 + k^2) ;
      int mag_level = 9900;

      int n_dot_level = 9900 * k;

      int theta = 180/3.14 * acos(n_dot_level/(mag_n*mag_level));


    if (x_axis > 30 || y_axis > 30 || x_axis < -30 || y_axis < -30)
```

```
    else

    Esplora.writeRGB(0,255,0);


  Serial.print("x theta: ");
  Serial.print(x_axis);

  Serial.print("\tx dist: ");
  Serial.print(x);

  Serial.print("\tz1 dist: ");
  Serial.print(z1);

  Serial.print("\ty theta: ");
  Serial.print(y_axis);

  Serial.print("\t y dist: ");
  Serial.print(y);

  Serial.print("\t z2 dist: ");
  Serial.print(z2);

  Serial.print("\t i: ");
  Serial.print(i);

  Serial.print("\t j: ");
  Serial.print(j);

  Serial.print("\t k: ");
  Serial.print(k);

  Serial.print("\t theta: ");
  Serial.println(theta);

  delay(100);

}
```