DemoSat Final Report

Community College of Denver

Instructor: Dr. Joel Thompson

Participants: Aleks Kimoni, Angie Torres, Clay Griffin-Derr, Esther You, Jimmy Tangchittsumran, Judit Bergfalk, Luke Wagner, Madison Turner

Team: Space Hawks

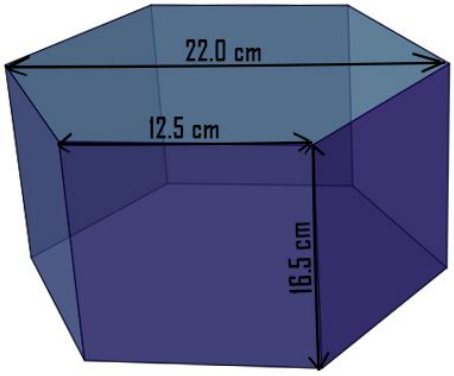
Launch Date: April 7, 2018

Introduction

The overall purpose of this project was to experientially learn and collaborate as a team in a project bigger than ourselves so that we may boldly send a scientific payload to greater heights where few community college students have ever gone before. In order to achieve this, the team met at least once weekly to learn how to put components together through a variety of means, including learning to solder, to code, to properly test the payload, and research as required. There were a variety of objectives set at the beginning of the club. First of which was to design and create a payload which can collect data on the following: temperature, orientation of the payload, pressure, UV light. Secondly, to include a Go-Pro camera to take photos throughout the mission. Thirdly, to include an internal heater to protect the payload from the cold of space. Finally, to have all sensors function throughout the launch, and survive landing impact. The culmination of these efforts resulted in a successful payload being constructed and launched on April 7th, 2018.

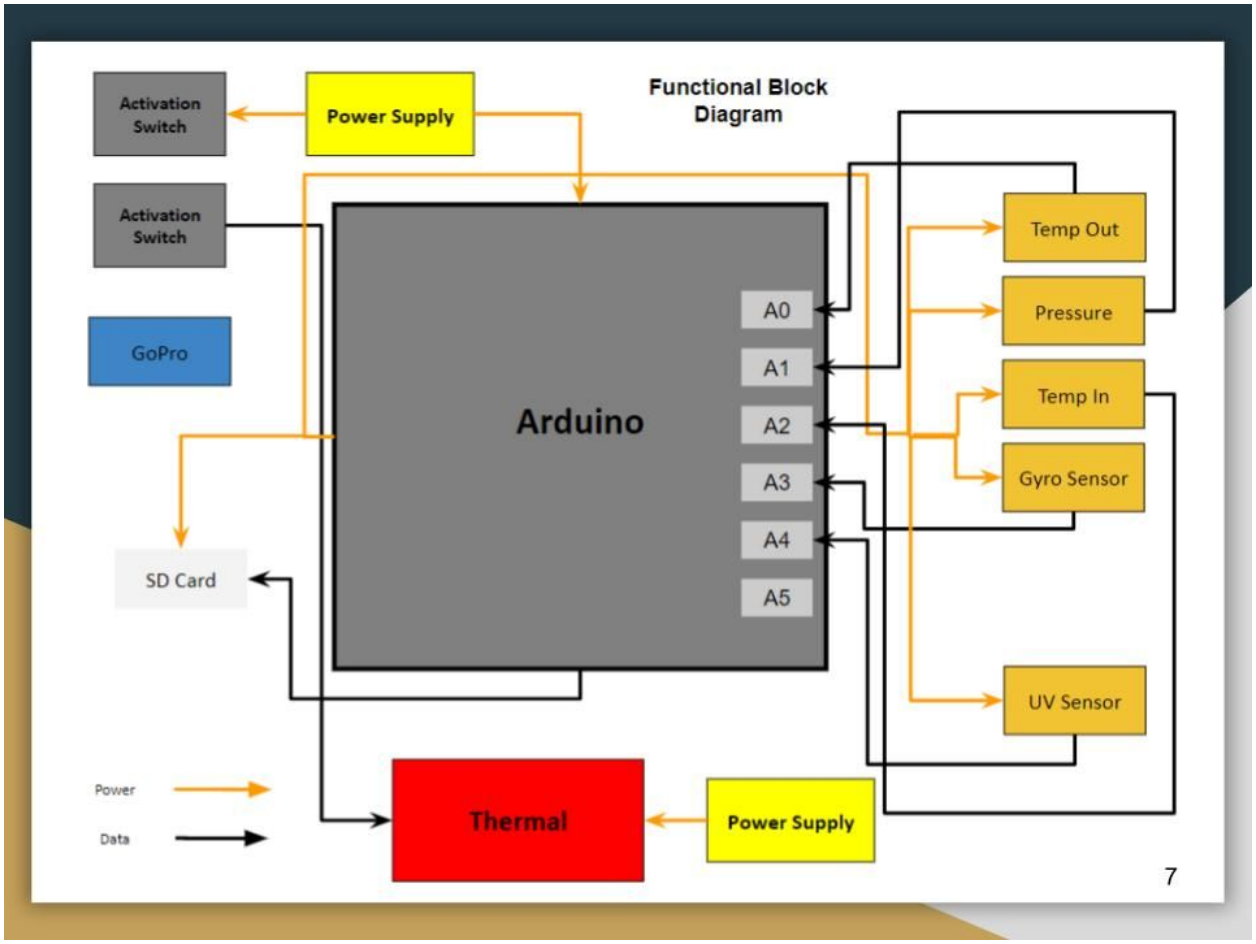
Design:

Structural Overview



Base Edge: 12.5 cm
Height: 16.5 cm
Volume ≈ 6700 cm^3
Final Weight: 790 g

Wiring Diagram:



Arduino Codes:

Temp In/Out/UV Code:

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

int scale = 200; // 3 (±3g) for ADXL337, 200 (±200g) for ADXL377
boolean micro_is_5V = true; // Set to true if using a 5V microcontroller such as the Arduino Uno, false if using a 3.3V microcontroller, this affects the interpretation of the sensor data

//Create variables to store results
float tempin_Read, tempout_Read, uv_Read;

void setup() {
// put your setup code here, to run once:
//put your setup code here, to run once:
  Serial.begin(9600);
  const int chipSelect =8;
pinMode(10, OUTPUT);
```

```
pinMode(chipSelect, OUTPUT);
//card check
if (!SD.begin(chipSelect)){Serial.println("card failed, not present");
//don't do anything more:
return;
}
Serial.println("card initialized");
//write heading
File dataFile = SD.open("DATA.txt", FILE_WRITE); // MAKE SURE THIS NAME MATCHES
THE ONE ON THE CARD!!:
//startheader comma separated
   dataFile.print("Time");
   dataFile.print(",");
   dataFile.print("Temp In"); //Start header in TXT file:
   dataFile.print (","); // creates white space between names, helps create collumns for data to fit
under:
   dataFile.print("Temp Out");
   dataFile.print (",");
   dataFile.println("UV output,");

   //close datafile
   dataFile.close();

}

void loop() {
// put your main code here, to run repeatedly:
// put your main code here, to run repeatedly:
//Define Temp_in;
int sensorValue1 = analogRead(A2);
float volt1 = (sensorValue1/1023.0)*5.0; //Volts
tempin_Read = (volt1 -0.5)/.01; //Celsius
//Define Temp_out;
int sensorValue2 = analogRead(A0);
float volt2 = (sensorValue2/1023.0)*5.0; //Volts
float tempout_Read = (volt2 -0.5)/.01; //Celsius
//Define uv value:
uv_Read = analogRead(A3);

  // Report values via UART
  Serial.print("sensorValue1 = ");
  Serial.print(sensorValue1);
  Serial.print(" volt1 = ");
  Serial.print(volt1);
  Serial.print(" tempin_Read = ");
  Serial.print(tempin_Read);

  Serial.print(" sensorValue2 = ");
```

```
  Serial.print(sensorValue2);
  Serial.print(" volt2 = ");
  Serial.print(volt2);
  Serial.print(" tempout_Read = ");
  Serial.println(tempout_Read);

//The following opens file named in "" and prepares to
//using READ so I don't overwrite headers
File dataFile = SD.open("2Tprobes.txt", FILE_WRITE);
//setup time stamp
//millis returns the number of ms since arduino was reset
const int timeDelay = 1000; // Define how long we want the delay to be here. I removed the
timestamp because we can use the # of linebreaks to detirmine timing:
const int timeStamp = millis();
  dataFile.print(timeStamp);
  dataFile.print(",");
  dataFile.print(tempin_Read);
  dataFile.print (",");
  dataFile.print(tempout_Read);
  dataFile.print (",");
  dataFile.print(uv_Read);
  dataFile.println(",");


  dataFile.close();

delay(timeDelay);
}
```

Pressure and XYZ Code:

```
#include <Wire.h>
#include <SparkFun_MS5803_I2C.h>
#include <SD.h>
#include <SPI.h>

MS5803 sensor(ADDRESS_HIGH);

int scale = 200; // 3 (±3g) for ADXL337, 200 (±200g) for ADXL377
boolean micro_is_5V = true; // Set to true if using a 5V microcontroller such as the Arduino Uno,
false if using a 3.3V microcontroller, this affects the interpretation of the sensor data

//Create variables to store results
float temperature_c, temperature_f;
double pressure_abs, pressure_relative, altitude_delta, pressure_baseline;

// Create Variable to store altitude in (m) for calculations;
double base_altitude = 1655.0; // Altitude of SparkFun's HQ in Boulder, CO. in (m)
```

```
void setup() {
  // put your setup code here, to run once:
        Serial.begin(9600);
        //Retrieve calibration constants for conversion math.
        sensor.reset();
        sensor.begin();
        const int chipSelect = 8;
pinMode(10, OUTPUT);
pinMode(chipSelect, OUTPUT);
//card check
if (!SD.begin(chipSelect)){Serial.println("card failed, not present");
//don't do anything more:
return;
}
Serial.println("card initialized");
//write heading
File dataFile = SD.open("XYZPRESSURE.txt", FILE_WRITE);
//startheader comma separated
  dataFile.print("time");
  dataFile.print(",");
  dataFile.print("RawX,RawY,RawZ,SX,SY,SZ");
  dataFile.print(",");
  dataFile.println("P1,T1,");
  //close datafile
  dataFile.close();

        pressure_baseline = sensor.getPressure(ADC_4096);

}

void loop() {

  int rawX = analogRead(A0);
  int rawY = analogRead(A1);
  int rawZ = analogRead(A2);

        // Scale accelerometer ADC readings into common units
  // Scale map depends on if using a 5V or 3.3V microcontroller
  float scaledX, scaledY, scaledZ; // Scaled values for each axis
  if (micro_is_5V) // Microcontroller runs off 5V
  {
        scaledX = mapf(rawX, 0, 675, -scale, scale); // 3.3/5 * 1023 =~ 675
        scaledY = mapf(rawY, 0, 675, -scale, scale);
        scaledZ = mapf(rawZ, 0, 675, -scale, scale);
  }

        //PRESSURE SENSOR CODE
```

```
  // Read temperature from the sensor in deg C. This operation takes about
  temperature_c = sensor.getTemperature(CELSIUS, ADC_512);

  // Read temperature from the sensor in deg F. Converting
  // to Fahrenheit is not internal to the sensor.
  // Additional math is done to convert a Celsius reading.
  temperature_f = sensor.getTemperature(FAHRENHEIT, ADC_512);

  // Read pressure from the sensor in mbar.
  pressure_abs = sensor.getPressure(ADC_4096);



  // Let's do something interesting with our data.

  // Convert abs pressure with the help of altitude into relative pressure
  // This is used in Weather stations.
  pressure_relative = sealevel(pressure_abs, base_altitude);

  // Taking our baseline pressure at the beginning we can find an approximate
  // change in altitude based on the differences in pressure.
  altitude_delta = altitude(pressure_abs , pressure_baseline);

  // Print out raw X,Y,Z accelerometer readings
  Serial.print("X: "); Serial.println(rawX);
  Serial.print("Y: "); Serial.println(rawY);
  Serial.print("Z: "); Serial.println(rawZ);

  // Print out scaled X,Y,Z accelerometer readings
  Serial.print("X: "); Serial.print(scaledX-104); Serial.println(" g");
  Serial.print("Y: "); Serial.print(scaledY-102); Serial.println(" g");
  Serial.print("Z: "); Serial.print(scaledZ-100); Serial.println(" g");
  Serial.println();

  // Report values via UART
  Serial.print("Temperature C = ");
  Serial.println(temperature_c);

  Serial.print(" Raw Voltage ");
  Serial.println(ADC_4096);

  Serial.print("Pressure abs (mbar)= ");
  Serial.println(pressure_abs);

//The following opens file named in "" and prepares to
//using READ so I don't overwrite headers
File dataFile = SD.open("XYZPRESSURE.txt", FILE_WRITE);
//setup time stamp
```

```
//millis returns the number of ms since arduino was reset
const int TimeStamp = millis();
  dataFile.print(TimeStamp);
  dataFile.print(",");
  dataFile.print(rawX);
  dataFile.print(",");
  dataFile.print(rawY);
  dataFile.print(",");
  dataFile.print(rawZ);
  dataFile.print(",");
  dataFile.print(scaledX-104);
  dataFile.print(",");
  dataFile.print(scaledY-102);
  dataFile.print(",");
  dataFile.print(scaledZ-100);
  dataFile.print (",");
  dataFile.print(pressure_abs);
  dataFile.print(",");
  dataFile.print(temperature_c);
  dataFile.println(",");
  //close datafile
  dataFile.close();

  delay(1000);
  }
float mapf(float x, float in_min, float in_max, float out_min, float out_max)
{
  return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;
}

// Thanks to Mike Grusin for letting me borrow the functions below from
// the BMP180 example code.

 double sealevel(double P, double A)
// Given a pressure P (mbar) taken at a specific altitude (meters),
// return the equivalent pressure (mbar) at sea level.
// This produces pressure readings that can be used for weather measurements.
{
    return(P/pow(1-(A/44330.0),5.255));
}
// Given a pressure measurement P (mbar) and the pressure at a baseline P0 (mbar),
// return altitude (meters) above baseline.


// Same functionality as Arduino's standard map function, except using floats
double altitude(double P, double P0)
{
  return(44330.0*(1-pow(P/P0,1/5.255)));
```

Testing:

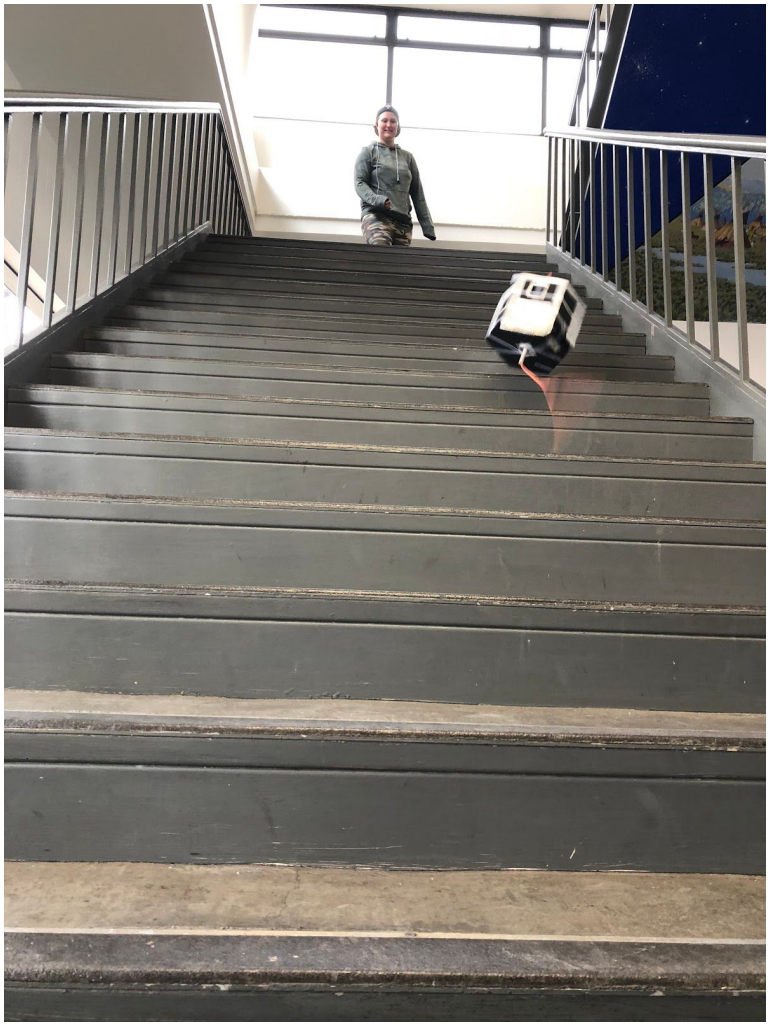
Cold-Test

Photo 1. Cold Test

Stair Pitch:

Drop case with an equal mass of equipment secured inside down a flight of concrete stairs to simulate being dragged across the ground.

Photo 2. Stair Pitch



Drop Test:

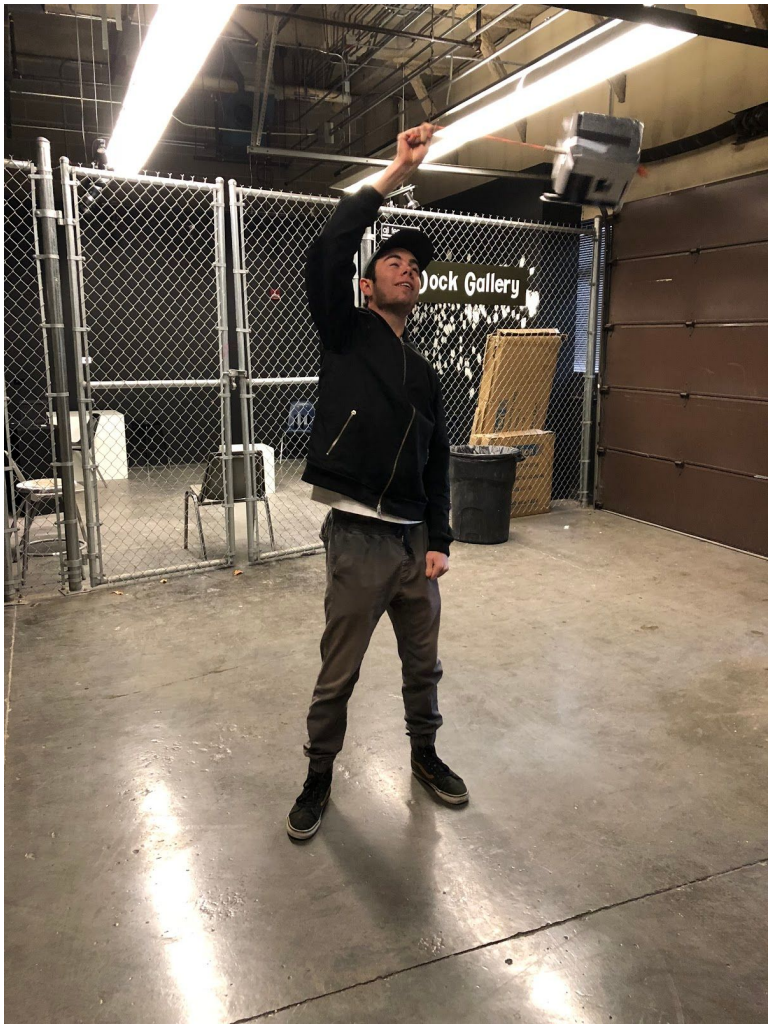
Drop case from 20 feet onto a hard surface to simulate worst-case fall.

Photo 3. Drop Test

Whip Test:

Spin payload on a string, like it will be mounted during the actual launch. Spin the payload harshly, simulating strong winds during payload ascent or descent.

Photo 4. Whip Test



Test Results:

The box performed well during tests and was deemed structurally secure. All sensors are coded and giving out efficient data. Batteries are charged and ready.

Flight Day Comments:

There were a few complications observed after recovering the payload.

- One of the memory cards fell out of the MicroSD shield and was later found in the payload.
- The other memory card did not record any data. Further investigations suggested the MicroSD shield being defective.
- The Go-Pro tipped out of its case, although recorded most of the flight up but not coming down. It also only recorded in 12-minute segments.
- The sensor that measured the outside temperature ended up inside the payload due to not securing it right.

- The connection to one of the batteries came unloose.

Photo 5. Interior of the Payload

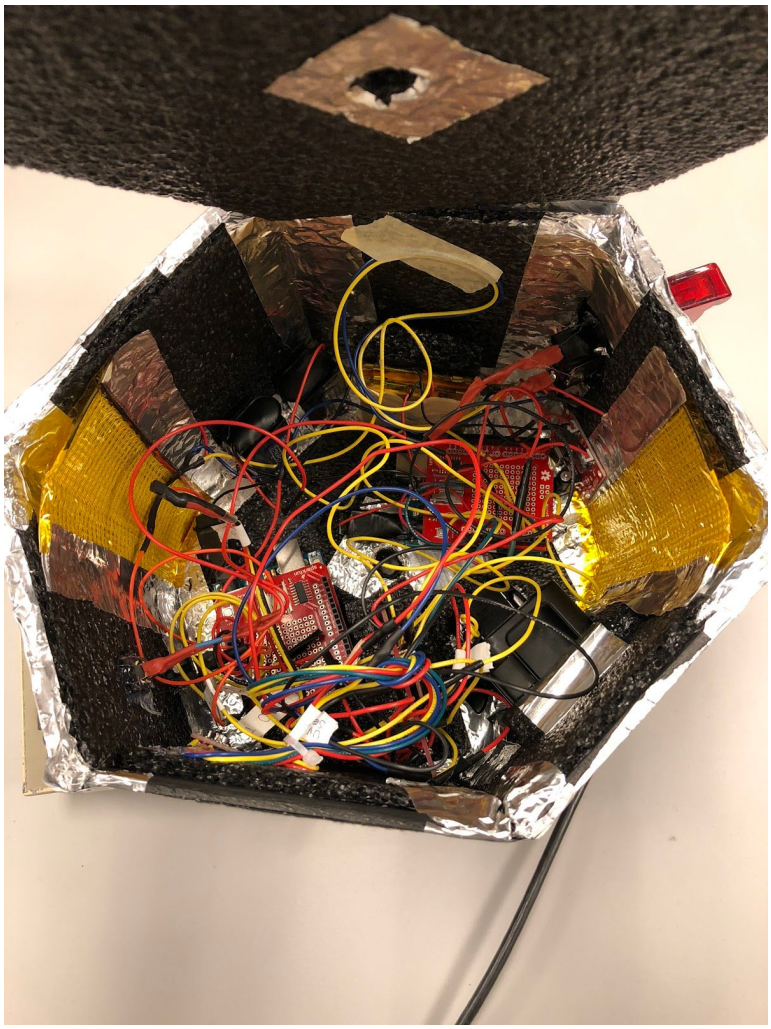


Photo 6. Weather Balloon Ascending

Photo 7. GPS Tracking

Photo 8. Screenshot of the Go-Pro Footage

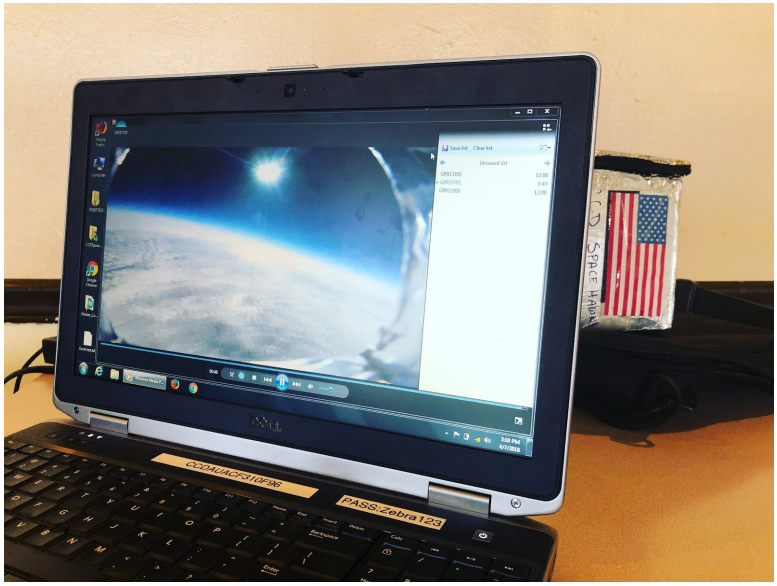


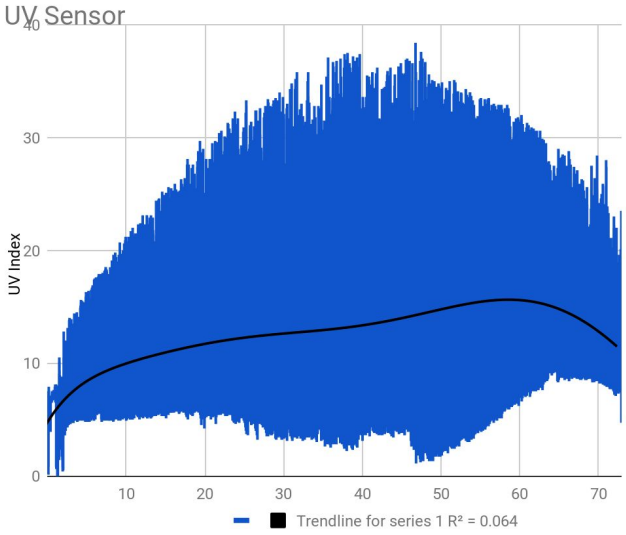
Photo 9. Space Hawks Team After Recovering Payload

Recommendations:

- Secure all components, including SD card attached to Arduino, battery tops, temperature probe outside of the payload.
- When coding, ensure everyone uses the same computer.
- Upload everything to Google Drive.
- Test everything and test thoroughly. Set a hard deadline at least three weeks prior to launch date.
- Be prepared to chase payload for hours.
- Be prepared for the cold weather when we get there to launch the balloon.

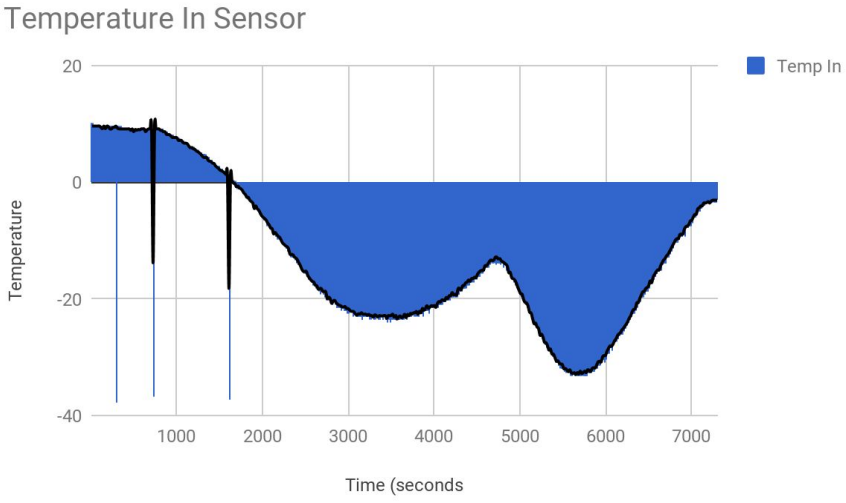
Data Analysis:

UV Sensor Data:



- The UV index is used to judge the danger from ultraviolet radiation on any given day. A 6 on the scale is a high-risk day. A rating of 10 or more is extremely dangerous. The highest ever recorded on the surface of the earth was a rating of 43, detected in the high Andes of South America. In this data we can see our reading peak in the high 30s.
- Tons of rapid variation in UV index reading, likely since the spinning box often prevented the sensor from receiving the sun's direct radiation.
- The general trend is initially upwards, reaching an average of 15 on the UV index. At the point when the balloon popped, there's a quick decrease, since the balloon's fall was much faster than its ascent.

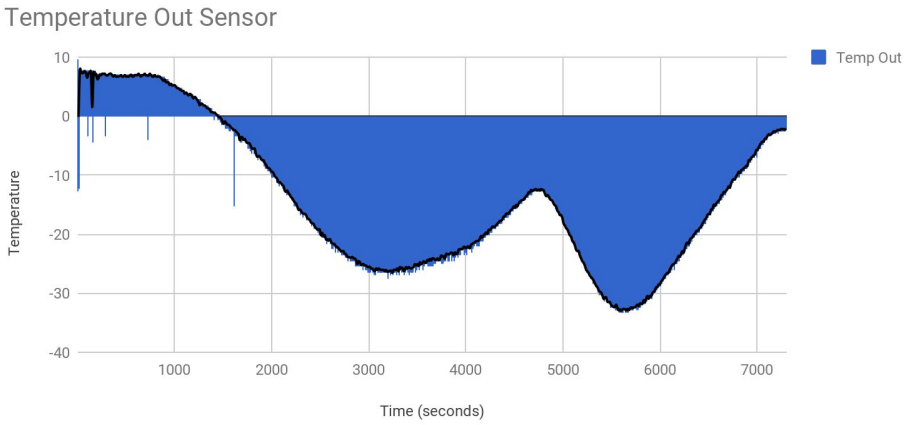
Temperature Inside Box Data:



- Readings are in degrees Celsius.

- This sensor was deep inside the box, along with the heater pads. However, we can still see how cold it got in the box!
- The temperature resembles a quadratic equation, with distinct parabolas.
- The two sudden and great decreases in temperature could have happened when the balloon and attached payloads broke the cover of the clouds and received more direct sunlight. The resulting temperature increase would have then decreased again as the balloon rose higher.
- The rise starting at 4,000 seconds in, and ending at about 4,750 seconds in, could be attributable to an air current, or in some variation in heat caused by the payload's internal heating.

  Temperature Outside Box Data:



- This sensor's data looks much like the inside temperature sensor's data. Since the outside sensor shifted and fell inside the box early in the launch, this is exactly as we predicted it would look.

  No Data was recovered for Pressure and XYZ, possibly due to defective/broken Arduino Shield. ;_;

Final Comments:

In conclusion, the team's collaborated efforts resulted in the successful construction, testing, launch, and retrieval of the payload, as well as a positive learning experience all around for all members. There were many difficulties in this project that needed to be overcome, including time management, planning, and balancing academic coursework in addition to club activities. However, the team's solid communication and mutual respect for each other allowed for a cohesive joint endeavour that reduced many of the barriers to project success. Future club members should be aware that the DEMOSAT program is a challenging, but exceptionally worthwhile opportunity. Being able to cooperate as a team in a group setting, working on a significant project with peers, and learning technical skills are invaluable experiences that will continue to pay off for many years to come. As the team prepares for a future career in the

various area of their chosen professions, the realization is that the sky's not the limit and that mankind’s potential is boundless.