Colorado Space Grant Consortium

DEMOSAT DESIGN DOCUMENT

Peak Power

Project by:
Isaak Sanders, Megan Robards, Daniel March, Karina Marcias Leyva

April 11, 2023
Revision 1

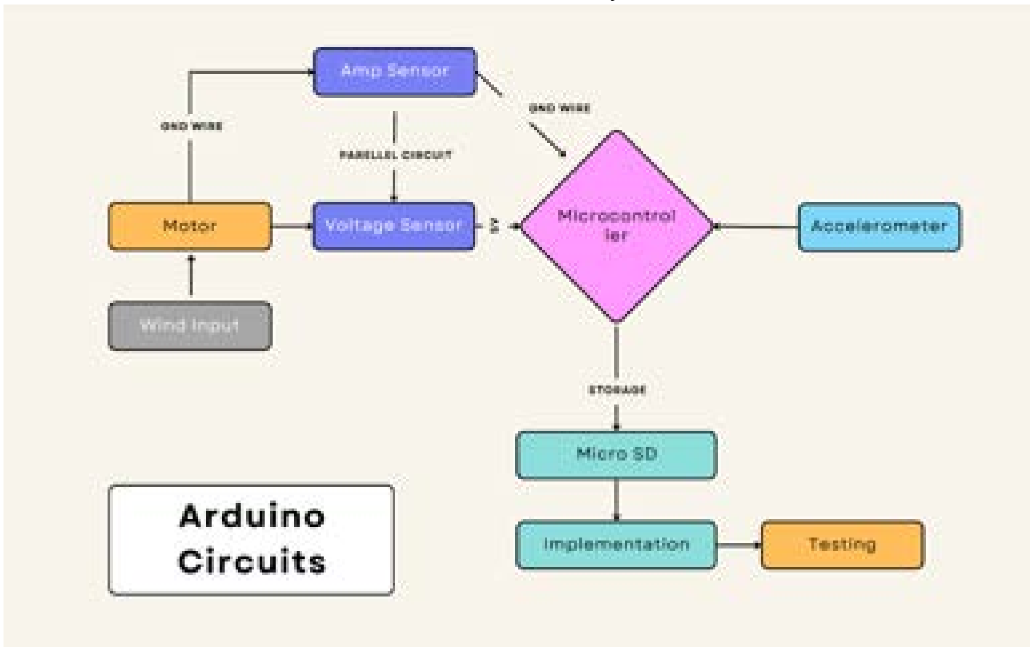
Table of Contents

1.0 Mission Overview

The goal of this mission was to test various propellers' efficiency when powering a payload at different altitudes. This will demonstrate the effectiveness of propellers for powering aircraft, and spacecraft. It will also show what types of propellers work best to power such vehicles. This mission will show how and if wind turbines can be used on aerial vehicles as a source of renewable energy.

2.0 Design

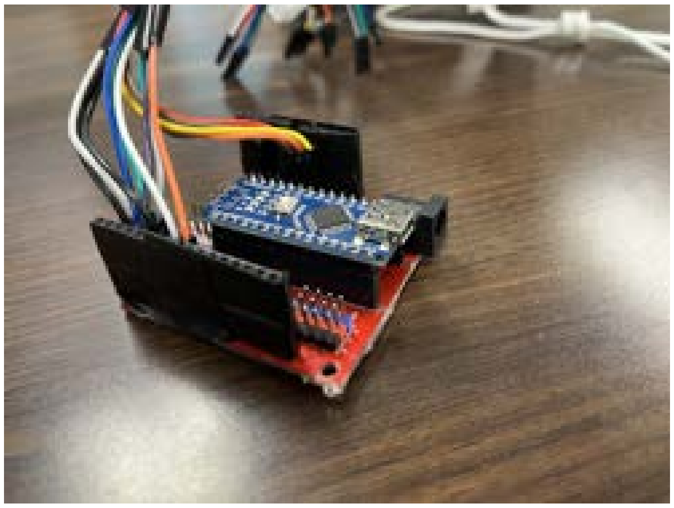
In order to test the various propeller's efficiency in conducting power, team-member Megan Robards was tasked with building an arduino microcontroller that used a current sensor and voltage sensor to test for wattage efficiency. The data collected will then move onto a micro SD card for later analysis.



Functional Block Diagram Logic

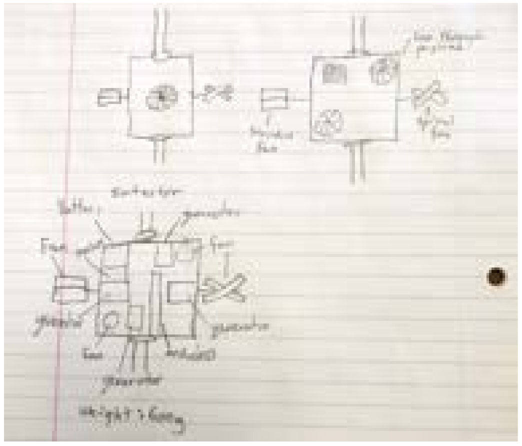
Parts (Rev B)

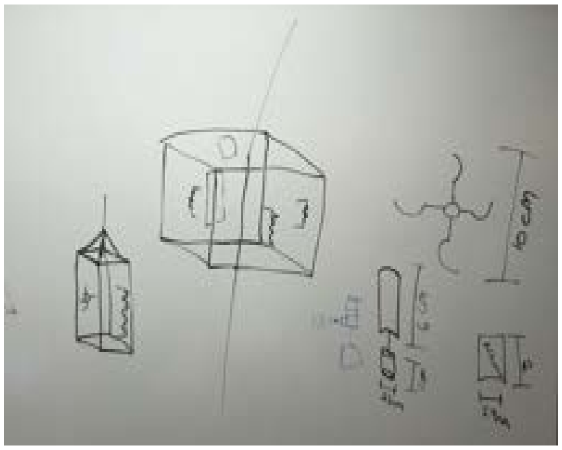
- Arduino Nano
- Balloon Shield
- Arduino Accelerometer
- Jumper Wires
- PLA Wind Turbines
- Aluminum Tape
- Foam Board
- ACS712 Current Sensor
- 5pcs Voltage Sensor DC 0~25V
- DC 0.1-5.5V Small Motor Blades Generator



Note: The Arduino Nano board is not directly compatible with the arduino uno balloon shield. Custom parts were made to allow for connection.

Drawings (Rev B)





Preliminary Sketches

The preliminary sketches were designed for a four turbine payload, later this idea changed into a 3 turbine payload.

Interior of Final Payload Powered by 5V Battery

Payload Exterior with two PLA printed turbines

3.0 Code

Using the same logic in the functional block diagram Megan Robards, aided by Casiniro DeFigueiredo, built the code used for the final launch. This software was completed using Arduino Code.

```
#include <SoftwareSerial.h>

/* This program is Designed for High-Atmosphere Balloon. This program reads data
from three generators(motors) and Gyroscope store is temporarily before saving the data
to a .txt
 * on a SD card.
 *
 * 1st PROGRAMMER: CASIMIRO M DEFIGUEIREDO  || 2nd PROGRAMMER:
MEGAN ROBARDS
 * DATE: 03/28/2023
 * VERSION 1.0
 *
 *
______________________________________________________________________________
__________NOTICE______________________________________________________________
____________________________________
 * - AMPMETERS have a small variance from 0mA to 185mA (Assume all Read values
below 200mA is 0)
 *

*_____________________________________________________________________________
____________HARDWARE__________________________________________________________
________________________________________
 * 3x Ampmeter
 * 3x Voltmeter
 * 1x Gryoscope
 * 1x Arduino Nano
 * 1x Arduino Nano Breakout board
 * 1x Arduino Uno Balloon sheild.
 *

*_____________________________________________________________________________
____________CONNECTION________________________________________________________
_______________________________________
 * - Each Ampmeter requires 5v to Vcc and GND connection
 * - Each Voltmeter requires GND connection
 *----------------------- PINOUTS ---------------------------------------------------
 *  PIN  | CONNECTION
 *-------|---------------------
 *   A0  | Ampmeter(1) Sig
 *   A1  | Voltmeter(1) Sig
 *-------|---------------------
 *   A2  | Ampmeter(2) Sig
 *   A3  | Voltmeter(2) Sig
 *-------|---------------------
 *   A4  | Gyroscope X-axis sig
 *   A5  | Gyroscope Z-axis sig
 *-------|---------------------
 *   A6  | Ampmeter(3) Sig
 *   A7  | Voltmeter(3) Sig
 */

//-----------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------
//                                    INCLUDES_&_DEFINE
//-----------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------

//Ampmeters pins
#define Amp1        A0
#define Amp2        A2
#define Amp3        A6

//Voltmeter pins
#define Volt1       A1
#define Volt2       A3
#define Volt3       A7

//Gyroscope pins
#define Gyro_X      A4
#define Gyro_Z      A5

//SENSOR VALUES
#define mVperAmp    0.180
#define ACSoffset   2.500
#define ACSerror    0.150
#define Resistor_1  30000.000
#define Resistor_2  7500.000


//Change Values
#define NumGens     3

SoftwareSerial mySerial (2,3);

//-----------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------
//                                    CLASS & STRUCT
//-----------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------

class mData {
```

```
protected:
  float amp;
  float volt;

public:
  mData(){
    amp = 0.000;
    volt = 0.000;
  }

  float AMP(){
    return amp;
  }
  void AMP(float a){
    amp = a;
  }
  float VOLT(){
    return volt;
  }
  void VOLT(float v){
    volt = v;
  }
  float Watt(){
    return amp * volt;
  }
};

class Time {

  private:
    int Hours;
    int Min;
    int Sec;
    int mill;

  public:
    Time() {
      Hours = 0;
      Min = 0;
      Sec = 0;
      mill = 0;
    }
    int HOUR() {
      return Hours;
    }
    int MINUTES() {
      return Min;
    }
    int SECOUNDS() {
      return Sec;
    }
    int MILLISECONDS() {
      return mill;
    }
    void ReadTime(){
      long temp = millis();
      Sec = temp / 1000;
      Min = temp / 60000;
      Hours = temp/ 3600000;
      while(temp >= 1000) {
        temp -= 1000;
      }
      mill = temp;
      while(Sec >= 60) {
        Sec -= 60;
      }
      while(Min >= 60) {
        Min -= 60;
      }
    }

};

//-----------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------
//                                                  GLOBAL_VARIABLE_&_FUNCTIONS
//-----------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------

//FUNCTIONS
float AmpSensor(byte);                                                              //
float VoltSensor(byte);

//GLOBALS
//Data handling
mData Data[NumGens];
mData High = mData();
Time time = Time();

// Accelerometer X
  int accelX;
  float accelXVolt;
  float accelXG;

// Accelerometer Z
  int accelZ;
  float accelZVolt;
  float accelZG;

//ports
byte Ammeters[] = {Amp1, Amp2, Amp3};
byte Voltmeters[] = {Volt1, Volt2, Volt3};

//
  unsigned int SampleSet = 1;


//----------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------
//                                                          MAIN_PROGRAM
//----------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------

void setup() {
  Serial.begin(9600);
  mySerial.begin(9600);

  for(byte i = 0; i < 1; i++)
    Data[i] = mData();

}

void loop() {
  Serial.print("      ** Sample Set: "); Serial.print(SampleSet); Serial.println(" **");
  for (byte s = 0; s < 10; s++) {

    Serial.print("-------------------------NEW SAMPLE "); Serial.print(s);
Serial.println("-----------------------------");
    //READ Generators
    for (int i = 0; i < NumGens; i++) {
      //Serial.print("aPort: ");
      //Serial.println(Ammeters[i]);
      //Serial.print("vPort: ");
      //Serial.println(Voltmeters[i]);
      //Ammeter
      Data[i].AMP(AmpSensor(Ammeters[i]));
      //Voltmeter
      Data[i].VOLT(VoltSensor(Voltmeters[i]));
      if (Data[i].VOLT() > High.VOLT())
        High.VOLT(Data[i].VOLT());
      if (Data[i].AMP() > High.AMP())
        High.AMP(Data[i].AMP());
      Serial.print("**HIGHEST VOLT**  ");
      Serial.println(High.VOLT(), 3);
      Serial.print("**HIGHESt AMP **  ");
      Serial.println(High.AMP(), 3);
    }

    //Read Accelarometer
    accelX = analogRead(A4);
    accelXVolt = accelX * (5.0 / 1023);
    accelXG = (accelXVolt - (3.3 / 2)) / (0.330);

    accelZ = analogRead(A5);
    accelZVolt = accelZ * (5.0 / 1023);
    accelZG = (accelZVolt - (3.3 / 2)) / (0.330);


    //Read Time
    time.ReadTime();

    //Print Time
    Serial.print("--------------------");
    Serial.print("Time (Hr:Min:Sec.MS): ");
    Serial.print(time.HOUR());
    Serial.print(":");
    Serial.print(time.MINUTES());
    Serial.print(":");
    Serial.print(time.SECOUNDS());
    Serial.print(".");
    Serial.println(time.MILLISECONDS());

    //Print Data
    for (byte i = 0; i < NumGens; i++) {
      Serial.print("* Motor ");
      Serial.print(i + 1);
      Serial.println(" *");
      delay(10);
      Serial.print("Volt: ");
      Serial.println(Data[i].VOLT(), 3);
      delay(10);
      Serial.print("Amp: ");
      Serial.println(Data[i].AMP(), 3);
```

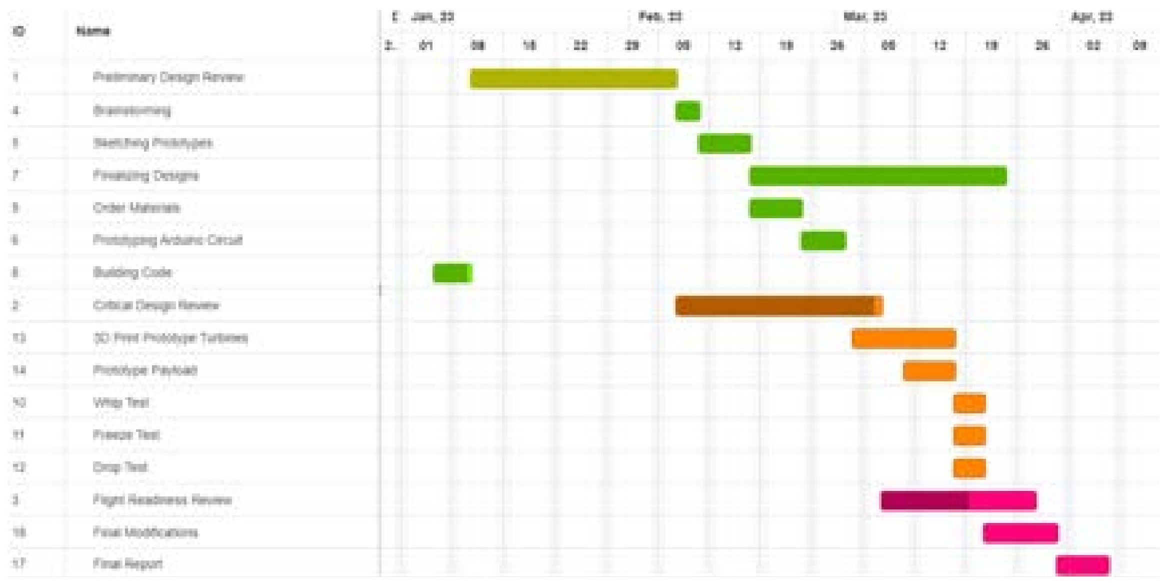
```
      delay(10);
      Serial.print("Watt: ");
      Serial.println(Data[i].Watt(), 3);
      delay(10);
    }
    Serial.print("Accel X-axis: ");
    Serial.println(accelXG, 3);
    Serial.print("Accel Z-axis: ");
    Serial.println(accelZG, 3);

    delay(1000);
  }
  Serial.println();
  SampleSet++;
  delay(14000);
}

    Serial.write(mySerial.read());
    {
      mySerial.println("Hello again, OpenLog!");
  delay(1000);
    }
}



//-----------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------
//                                                   FUNCTIONS
//-----------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------

/* Function NAME    |    AmpSensor
 * VERSION          |    1.0
 * DATE             |    03/28/23
 *
 * DESCRIPTION
 *    This function reads the amp sensor connected to "pin" and returns the value after
calculating the data.
 *
 * INPUTS
 * - pin | byte | pin for the ampmeter signal
 *
 * RETURNS
 * - float | Amps
 */
```

```
float AmpSensor(byte pin){

  //Variables
  float Voltage = 0.000;
  float Amperage = 0.000;
  float RawValue = analogRead(pin);

  //MATH
  //Serial.print("READ VALUE: "); Serial.println(RawValue, 3);
  Voltage = ((RawValue * 5.000) / 1024.000);
  //Serial.print("Voltage: "); Serial.println(Voltage, 3);
  Amperage = abs((Voltage - ACSoffset) / mVperAmp) - ACSerror;
  if(Amperage < 0)
    Amperage = 0;
  //return
  return Amperage;
}

/* Function NAME    |    VoltSensor
 * VERSION          |    1.0
 * DATE             |    03/28/23
 *
 * DESCRIPTION
 *    This function reads the volt sensor connected to "pin" and returns the value after
calculating the data.
 *
 * INPUTS
 * - pin | byte | pin for the voltmeter signal
 *
 * RETURNS
 * - float | Amps
 */
float VoltSensor(byte pin){

  //Variables
  float Voltage_out = 0.000;
  float Voltage_in = 0.000;
  int RawValue = analogRead(pin);

  //MATH
  Voltage_in = (RawValue * 5.000) / 1024.000;
  Voltage_out = Voltage_in / (Resistor_2 / (Resistor_1 + Resistor_2));

  //return
  return Voltage_out;
}
```

4.0 Management

The management of Peak Power was a great success in that three of us took the lead in different areas of the build. Since this team was a late entry, Megan took the lead in the beginning because this was her second year in this program. Once the team had an idea of what we were doing and what was expected, Megan took charge of the programming and circuitry, Isaak led the payload structure and propellers build, Daniel took the supply and ordering lead, and Karina helped with the paper portion of the project. For the most part, the process went without any management issues and the experiment was successful in collecting data.

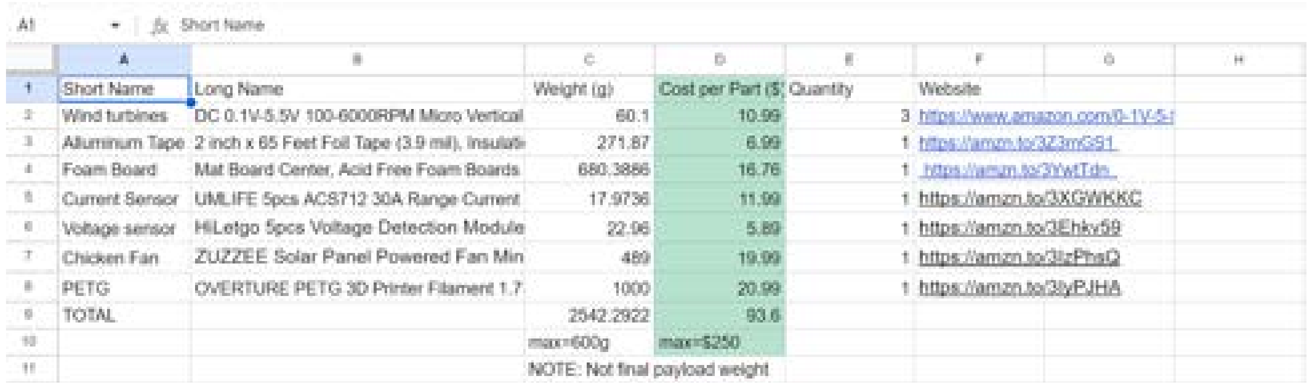
Chart 1:Gantt Chart



5.0 Budget

A total of 93 dollars were spent on supplies for this payload, 157 dollars less than the max budget of 250 dollars. This is displayed in the chart below.

Chart 2: Budget



A1 | fx Short Name

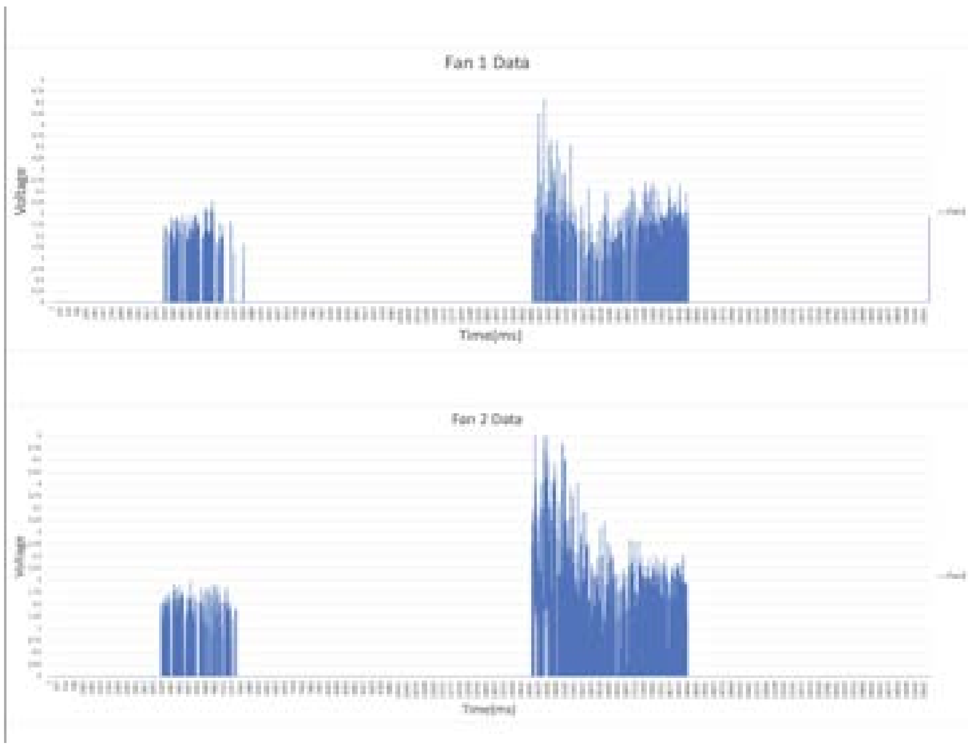
	A	B	C	D	E	F	G	H
1	Short Name	Long Name	Weight (g)	Cost per Part ($	Quantity	Website		
2	Wind turbines	DC 0.1V-5.5V 100-6000RPM Micro Vertical	60.1	10.99	3	https://www.amazon.com/0-1V-5-		
3	Aluminum Tape	2 inch x 65 Feet Foil Tape (3.9 mil), Insulati	271.87	6.99	1	https://amzn.to/3Z3mGS1		
4	Foam Board	Mat Board Center, Acid Free Foam Boards	680.3886	16.76	1	https://amzn.to/3YwtTdn		
5	Current Sensor	UMLIFE 5pcs ACS712 30A Range Current	17.9736	11.99	1	https://amzn.to/3XGWKKC		
6	Voltage sensor	HiLetgo 5pcs Voltage Detection Module	22.96	5.89	1	https://amzn.to/3Ehkv59		
7	Chicken Fan	ZUZZEE Solar Panel Powered Fan Min	489	19.99	1	https://amzn.to/3IzPhsQ		
8	PETG	OVERTURE PETG 3D Printer Filament 1.7	1000	20.99	1	https://amzn.to/3iyPJHA		
9	TOTAL		2542.2922	93.6				
10			max=600g	max=$250				
11			NOTE: Not final payload weight					

6.0 Test Plan and Results

There was a problem with the computer hardware saving to an SD card. It was found after many hours and many people contributing their experiences to discover that the OpenLog came wired backwards from the factory. This issue did not allow for any pretests to be accurately completed and were not recorded because the OpenLog was not found to be the problem until the night before the launch. This means that this payload flew blind with no known potential problems to fix. Upon inspection of the initial 3D printed fans, it was found that two were printed with weak points and it was decided to reprint two of the three propellers. These new propellers were made of PLA the second time and proved to be much stronger than the first two. Although, the second set took many times to reprint due to an incorrectly installed filament spool.

7.0 Expected Results

The expected mission results are that the propellers will stop spinning after a certain point due to a vacuum. Before and after the propellers enter the vacuum, they will produce power due to air pressure and wind. It will produce the most energy shortly after they re-enter into air pressure. It is also expected that propeller A will be the most efficient based on the initial designs and internet searches. Our data should be similar to those from past projects, but will show the most efficient propeller design rather than just show propeller action throughout flight.



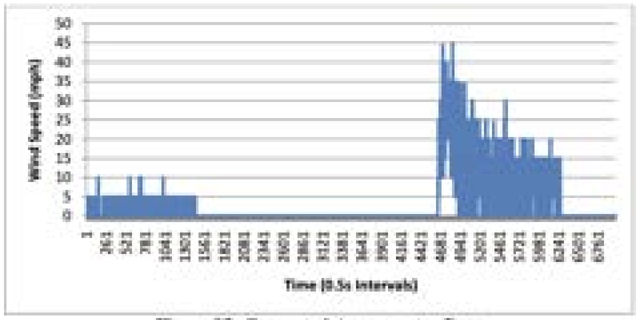


Figure 27: Converted Anemometer Data

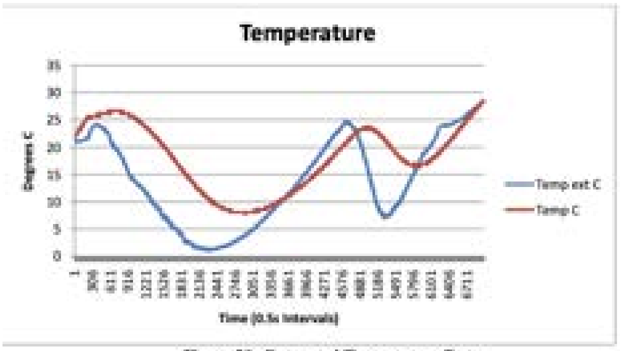
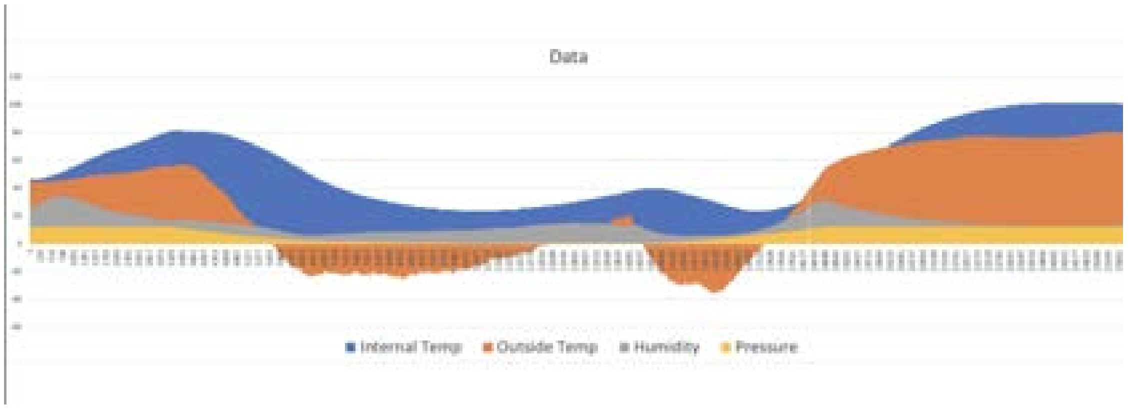


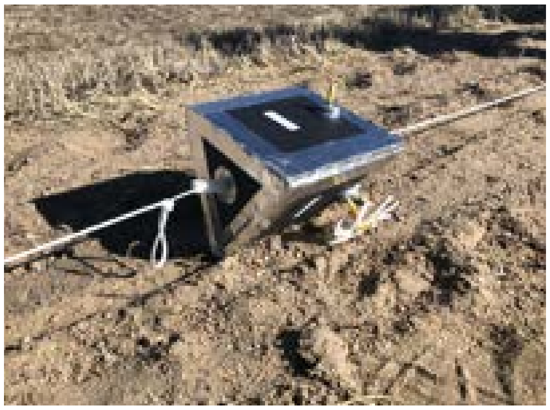
Figure 28: Converted Temperature Data



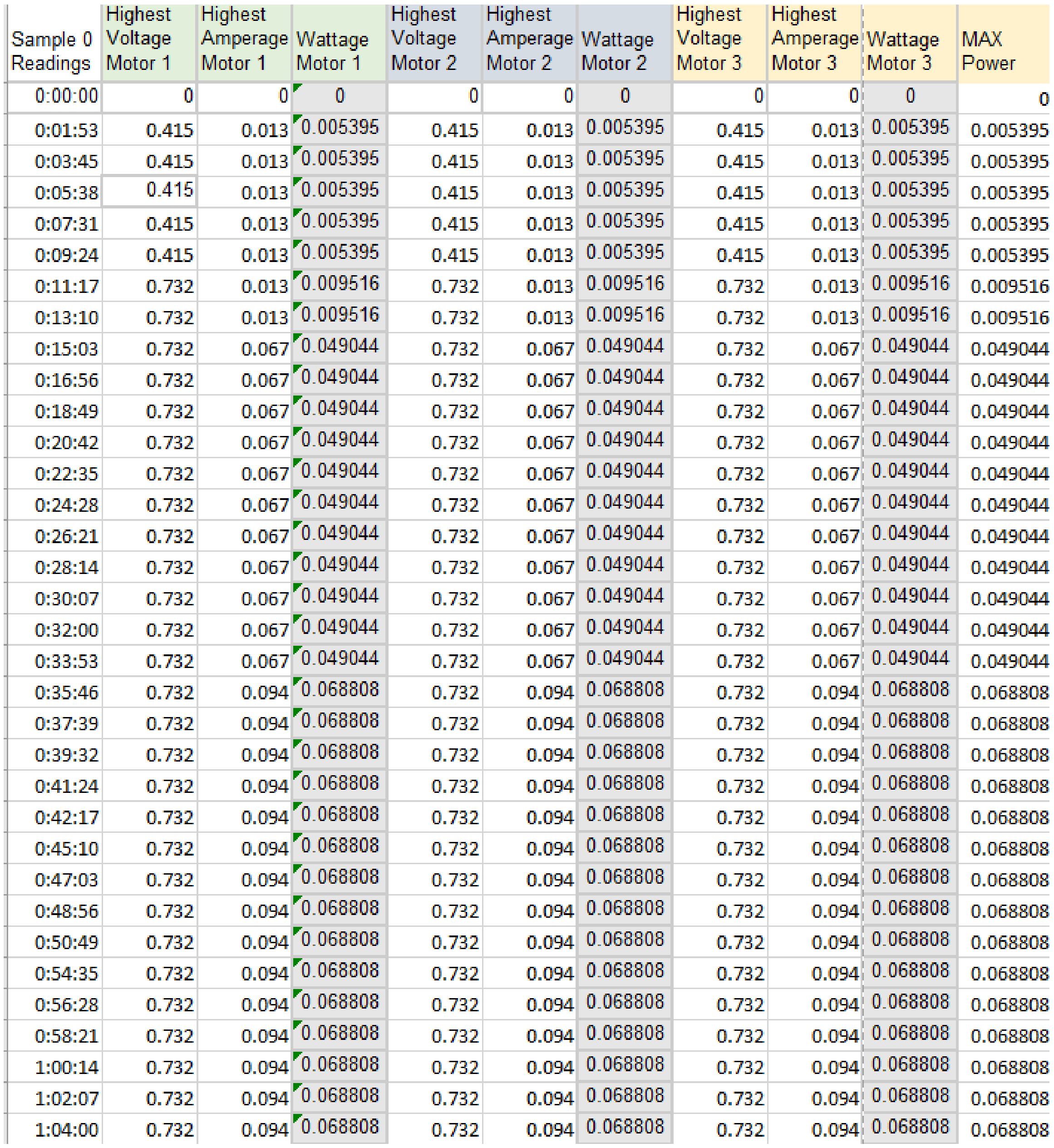
Wind Turbines as Payload Power
https://www.colorado.edu/center/spacegrant/sites/default/files/attached-files/Wind_Turbines_as_Payload_Power.pdf
Alternate Energy Sat
https://www.colorado.edu/center/spacegrant/sites/default/files/attached-files/CSU_DesignD_Final_revd.pdf

8.0 Launch and Recovery

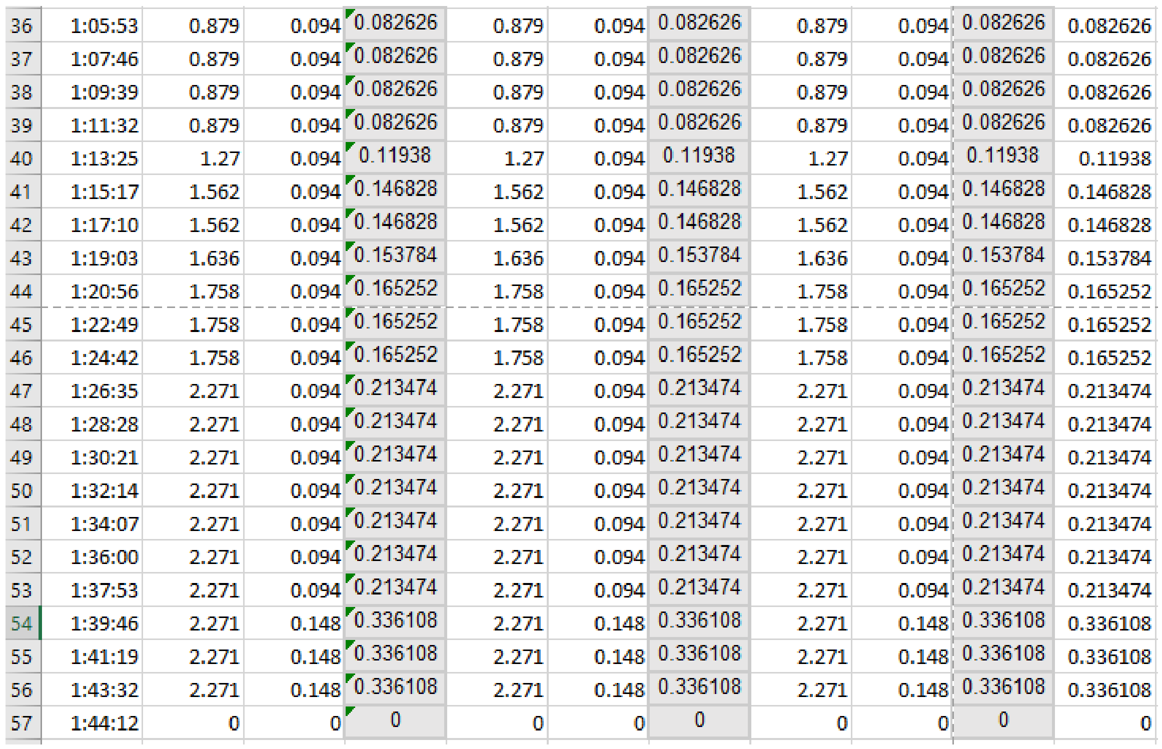
On the launch day, Isaak Sanders was the handler who turned on the payload before the flight and released it as the weather balloon was released. Subsequently, the team went with the whole demosat group to retrieve the payload. The payload was safely retrieved, with one propeller that snapped off during the landing. Then, the data was taken off of the SD card that was recording inside of the payload. On that day, the weather was ideal, and everything went according to plan except for the balloon bursting early causing it to fly to about 60,000 feet instead of the expected 120,000 feet.



9.0 Results, Analysis, and Conclusions

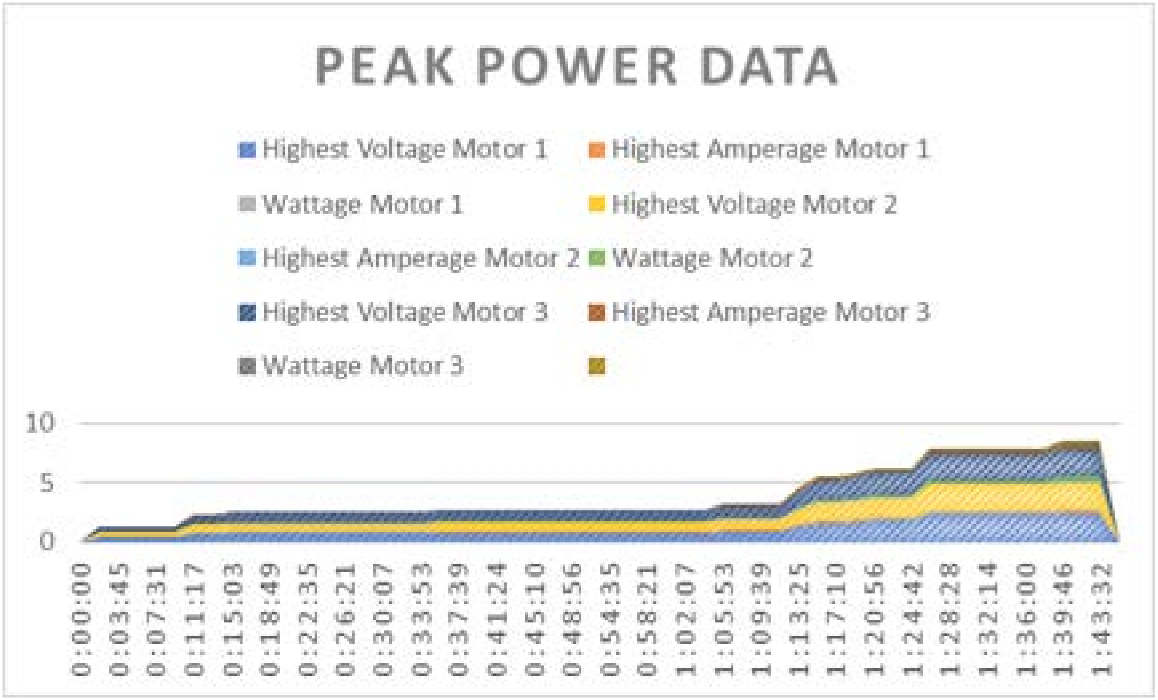


Sample 0 Readings	Highest Voltage Motor 1	Highest Amperage Motor 1	Wattage Motor 1	Highest Voltage Motor 2	Highest Amperage Motor 2	Wattage Motor 2	Highest Voltage Motor 3	Highest Amperage Motor 3	Wattage Motor 3	MAX Power
0:00:00	0	0	0	0	0	0	0	0	0	0
0:01:53	0.415	0.013	0.005395	0.415	0.013	0.005395	0.415	0.013	0.005395	0.005395
0:03:45	0.415	0.013	0.005395	0.415	0.013	0.005395	0.415	0.013	0.005395	0.005395
0:05:38	0.415	0.013	0.005395	0.415	0.013	0.005395	0.415	0.013	0.005395	0.005395
0:07:31	0.415	0.013	0.005395	0.415	0.013	0.005395	0.415	0.013	0.005395	0.005395
0:09:24	0.415	0.013	0.005395	0.415	0.013	0.005395	0.415	0.013	0.005395	0.005395
0:11:17	0.732	0.013	0.009516	0.732	0.013	0.009516	0.732	0.013	0.009516	0.009516
0:13:10	0.732	0.013	0.009516	0.732	0.013	0.009516	0.732	0.013	0.009516	0.009516
0:15:03	0.732	0.067	0.049044	0.732	0.067	0.049044	0.732	0.067	0.049044	0.049044
0:16:56	0.732	0.067	0.049044	0.732	0.067	0.049044	0.732	0.067	0.049044	0.049044
0:18:49	0.732	0.067	0.049044	0.732	0.067	0.049044	0.732	0.067	0.049044	0.049044
0:20:42	0.732	0.067	0.049044	0.732	0.067	0.049044	0.732	0.067	0.049044	0.049044
0:22:35	0.732	0.067	0.049044	0.732	0.067	0.049044	0.732	0.067	0.049044	0.049044
0:24:28	0.732	0.067	0.049044	0.732	0.067	0.049044	0.732	0.067	0.049044	0.049044
0:26:21	0.732	0.067	0.049044	0.732	0.067	0.049044	0.732	0.067	0.049044	0.049044
0:28:14	0.732	0.067	0.049044	0.732	0.067	0.049044	0.732	0.067	0.049044	0.049044
0:30:07	0.732	0.067	0.049044	0.732	0.067	0.049044	0.732	0.067	0.049044	0.049044
0:32:00	0.732	0.067	0.049044	0.732	0.067	0.049044	0.732	0.067	0.049044	0.049044
0:33:53	0.732	0.067	0.049044	0.732	0.067	0.049044	0.732	0.067	0.049044	0.049044
0:35:46	0.732	0.094	0.068808	0.732	0.094	0.068808	0.732	0.094	0.068808	0.068808
0:37:39	0.732	0.094	0.068808	0.732	0.094	0.068808	0.732	0.094	0.068808	0.068808
0:39:32	0.732	0.094	0.068808	0.732	0.094	0.068808	0.732	0.094	0.068808	0.068808
0:41:24	0.732	0.094	0.068808	0.732	0.094	0.068808	0.732	0.094	0.068808	0.068808
0:42:17	0.732	0.094	0.068808	0.732	0.094	0.068808	0.732	0.094	0.068808	0.068808
0:45:10	0.732	0.094	0.068808	0.732	0.094	0.068808	0.732	0.094	0.068808	0.068808
0:47:03	0.732	0.094	0.068808	0.732	0.094	0.068808	0.732	0.094	0.068808	0.068808
0:48:56	0.732	0.094	0.068808	0.732	0.094	0.068808	0.732	0.094	0.068808	0.068808
0:50:49	0.732	0.094	0.068808	0.732	0.094	0.068808	0.732	0.094	0.068808	0.068808
0:54:35	0.732	0.094	0.068808	0.732	0.094	0.068808	0.732	0.094	0.068808	0.068808
0:56:28	0.732	0.094	0.068808	0.732	0.094	0.068808	0.732	0.094	0.068808	0.068808
0:58:21	0.732	0.094	0.068808	0.732	0.094	0.068808	0.732	0.094	0.068808	0.068808
1:00:14	0.732	0.094	0.068808	0.732	0.094	0.068808	0.732	0.094	0.068808	0.068808
1:02:07	0.732	0.094	0.068808	0.732	0.094	0.068808	0.732	0.094	0.068808	0.068808
1:04:00	0.732	0.094	0.068808	0.732	0.094	0.068808	0.732	0.094	0.068808	0.068808



36	1:05:53	0.879	0.094	0.082626	0.879	0.094	0.082626	0.879	0.094	0.082626	0.082626
37	1:07:46	0.879	0.094	0.082626	0.879	0.094	0.082626	0.879	0.094	0.082626	0.082626
38	1:09:39	0.879	0.094	0.082626	0.879	0.094	0.082626	0.879	0.094	0.082626	0.082626
39	1:11:32	0.879	0.094	0.082626	0.879	0.094	0.082626	0.879	0.094	0.082626	0.082626
40	1:13:25	1.27	0.094	0.11938	1.27	0.094	0.11938	1.27	0.094	0.11938	0.11938
41	1:15:17	1.562	0.094	0.146828	1.562	0.094	0.146828	1.562	0.094	0.146828	0.146828
42	1:17:10	1.562	0.094	0.146828	1.562	0.094	0.146828	1.562	0.094	0.146828	0.146828
43	1:19:03	1.636	0.094	0.153784	1.636	0.094	0.153784	1.636	0.094	0.153784	0.153784
44	1:20:56	1.758	0.094	0.165252	1.758	0.094	0.165252	1.758	0.094	0.165252	0.165252
45	1:22:49	1.758	0.094	0.165252	1.758	0.094	0.165252	1.758	0.094	0.165252	0.165252
46	1:24:42	1.758	0.094	0.165252	1.758	0.094	0.165252	1.758	0.094	0.165252	0.165252
47	1:26:35	2.271	0.094	0.213474	2.271	0.094	0.213474	2.271	0.094	0.213474	0.213474
48	1:28:28	2.271	0.094	0.213474	2.271	0.094	0.213474	2.271	0.094	0.213474	0.213474
49	1:30:21	2.271	0.094	0.213474	2.271	0.094	0.213474	2.271	0.094	0.213474	0.213474
50	1:32:14	2.271	0.094	0.213474	2.271	0.094	0.213474	2.271	0.094	0.213474	0.213474
51	1:34:07	2.271	0.094	0.213474	2.271	0.094	0.213474	2.271	0.094	0.213474	0.213474
52	1:36:00	2.271	0.094	0.213474	2.271	0.094	0.213474	2.271	0.094	0.213474	0.213474
53	1:37:53	2.271	0.094	0.213474	2.271	0.094	0.213474	2.271	0.094	0.213474	0.213474
54	1:39:46	2.271	0.148	0.336108	2.271	0.148	0.336108	2.271	0.148	0.336108	0.336108
55	1:41:19	2.271	0.148	0.336108	2.271	0.148	0.336108	2.271	0.148	0.336108	0.336108
56	1:43:32	2.271	0.148	0.336108	2.271	0.148	0.336108	2.271	0.148	0.336108	0.336108
57	1:44:12	0	0	0	0	0	0	0	0	0	0

Flight Data



As shown by the data tables, the voltage only continues to increase, especially towards the end of the flight. From this the team can infer that the payload never made it high enough to reach the vacuum of space because there was never a pause in data or an output of zero. Additionally, the voltage rapidly increases towards the end of the flight signifying that the payload is descending. The weight of the balloon and the direction of motion, coupled with gravity, would increase the speed of the object, thus allowing for fast speeds of wind to flow through the turbines. Most likely because of a coding or hardware problem, all of the motor data points are identical between the different turbines.

10.0 Ready for Flight

Our payload was mostly intact on retrieval except for one propeller breaking at the motor connection; this would require a new connection and could not be repaired. While it is a complete connection replacement it would be a relatively inexpensive and quick repair that would require a new part glued on by gorilla glue. All other parts of the payload would fly again. The difficulty of this repair is that it may happen again since it was simply a weak part and not a weakness in our plan. The payload would just need to have a new battery, the above repairs done, and stored in a dry location.

11.0 Conclusions and Lessons Learned

From this experiment, we learned that testing before launch is crucial since the data we got was not recorded correctly because of the direction the propeller spun. Due to the limited time it was difficult to foresee this being a problem. If we could do it again we would change the board since there were issues because of how it was built. This would have saved time and made the process run smoother. Additionally we would have started the 3D printing earlier because it took a long time to print out all three propellers. And if there were issues it took even longer to get a better version and there were printing errors.

In conclusion the data was not complete but with the data we did get we were able to see the voltage generated by the propellers. As shown in 8.0 the data allows us to see where the balloon was and when it started to fall to the ground.

The ending data points for the three different turbines are identical, most likely because of a hardware or wiring issue thus concluding our experiment semi-successful and our mission a failure. Because of the identical data point the Peak Power team was not able to come to a conclusion about which turbine design is ideal for conducting the best power, however, the team is able to find the range of power produced from their results. With further experimentation on the arduino, and a better understanding of the arduino code, then the team will be able to perform this experiment again and come to a reasonable conclusion.

12.0 Message to Next Year

To any students next year or in future years who hope to do a similar project we suggest that you check all hardware with a breadboard to make sure that the wiring was done correctly at the factory. This gave us, especially Megan, a massive headache and a bit of stress. 3D printing should be one of the first things to complete because issues can arise from improper printing or previously unnoticed printing errors. Another is to make sure to leave enough time for testing because catching any errors in data early can help prevent any errors during launch. If the data does not look right to what you are comparing it, also double check things are working correctly. Additionally, just because parts are made by professionals does not mean that they are always 100% correct. The Openlog SD provided to us by Sparkfun Tx and Rx pins were switched and as a result, the team was not able to retrieve and test data until they figured out the problem the night before the launch.