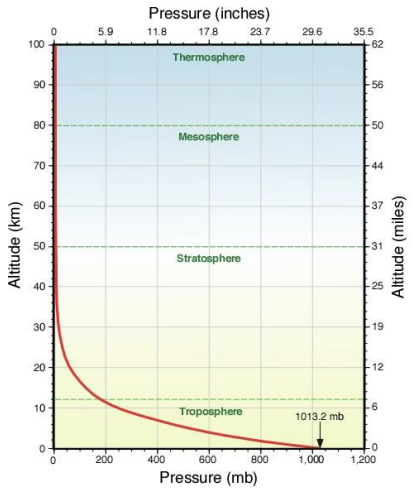
CDR Group A

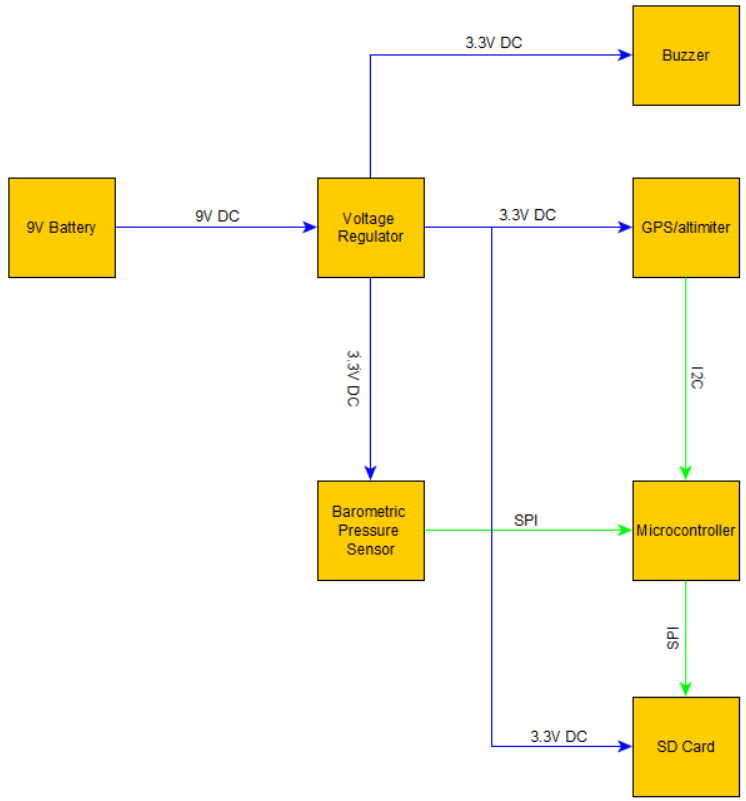
Kade, Quintin, Anurag, Cameron, Noah, Kevin

Science Mission

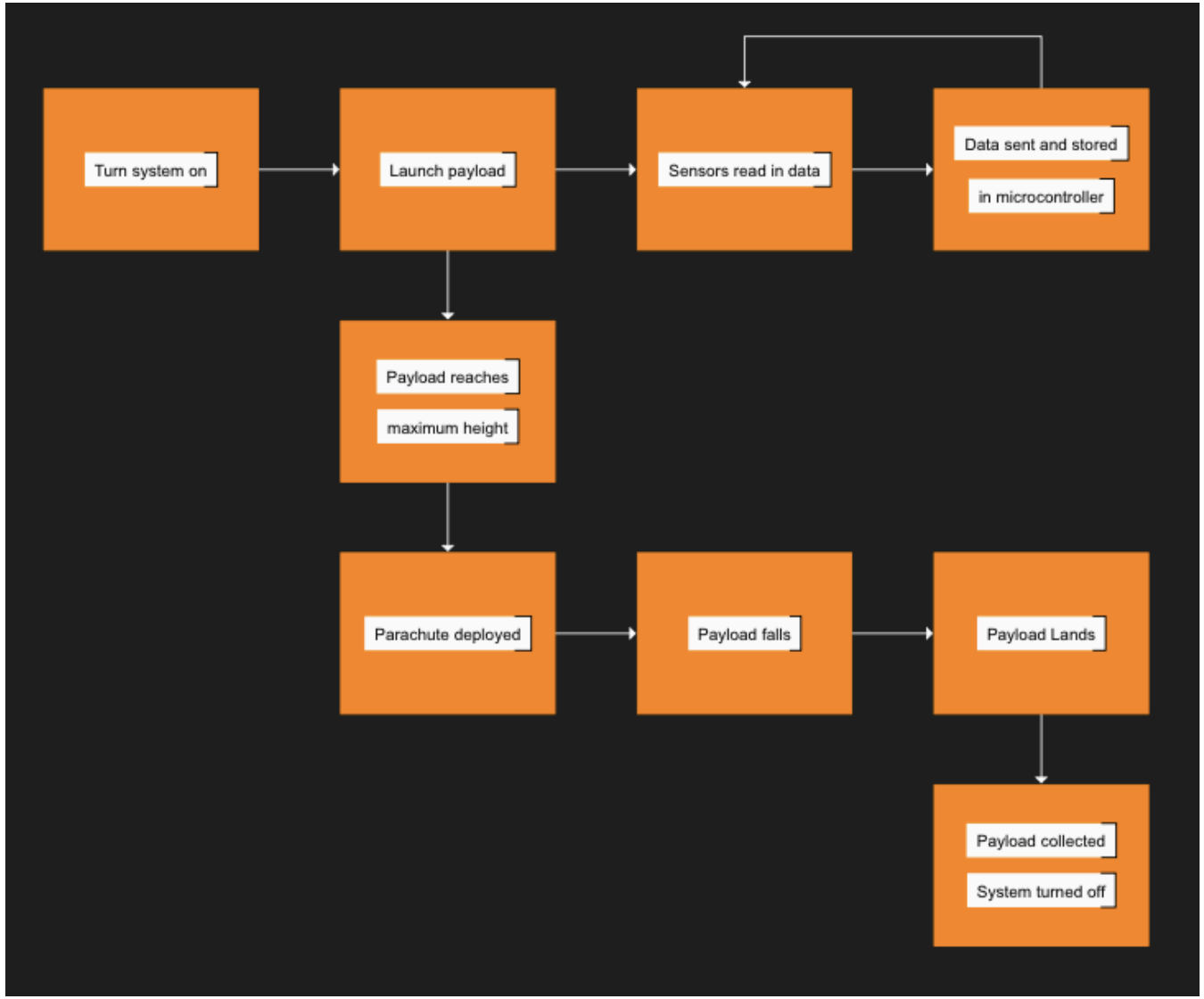
- To retrieve different barometric data points at various altitudes
- Barometric pressure will be collected with a barometric sensor
- Altitude will be collected with a GPS sensor
- Data will be used in further experiments that require changes in barometric pressure



System Block Diagram



Con-Ops


Turn system on
Launch payload
Sensors read in data
Data sent and stored
in microcontroller
Payload reaches
maximum height
Parachute deployed
Payload falls
Payload Lands
Payload collected
System turned off

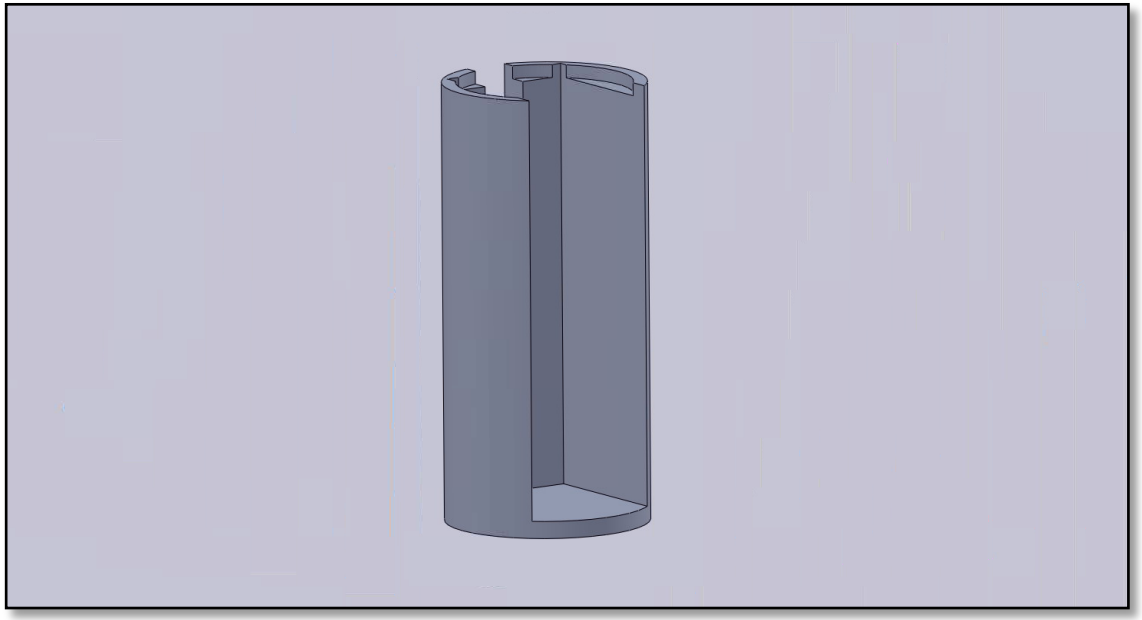
Complete Parts List

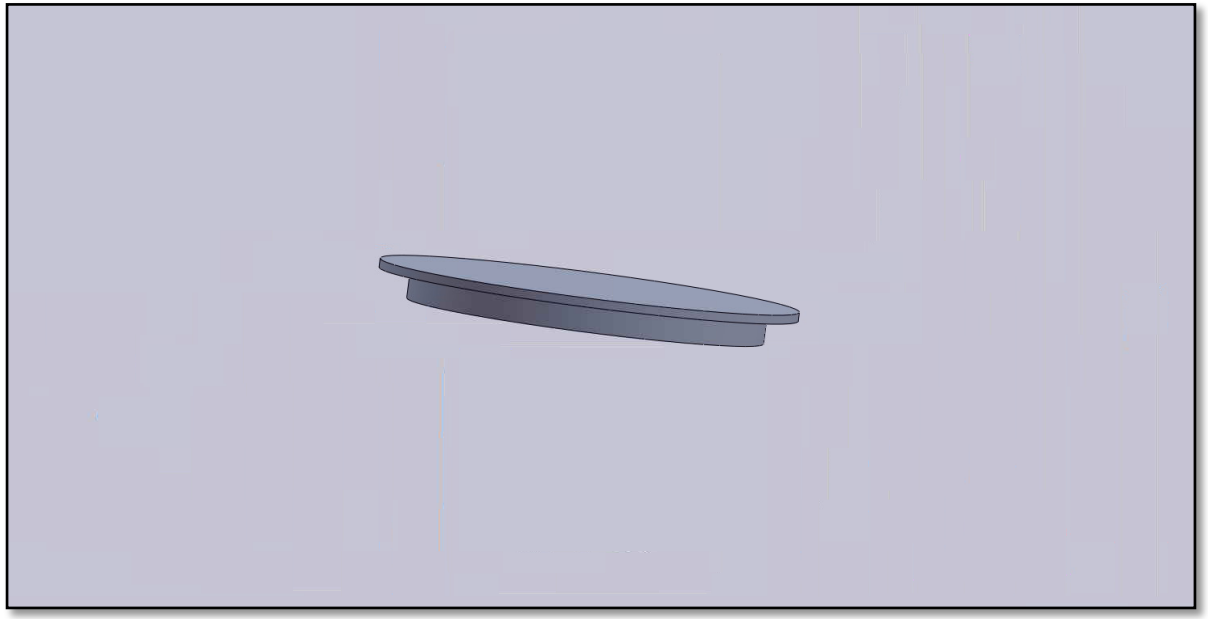
Part Name	Purpose	Adafruit Product ID
Adafruit Feather 32u4 Basic Proto	Collect and store data from sensors	2771
9V DC Battery	Power sensors	1321
Buzzer	Makes sound to find payload	1536
Parachute	Slows down speed of payload	N/A
GPS sensor	Collect data for altitude and satellite tracking	746
Barometric sensor	Collect atmospheric pressure data	4494
3.3V voltage regulator	Protect sensors	2165
MicroSD card breakout board+	Programming	254

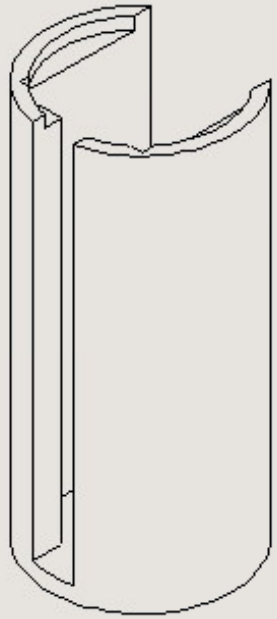
Mass & Cost Estimates

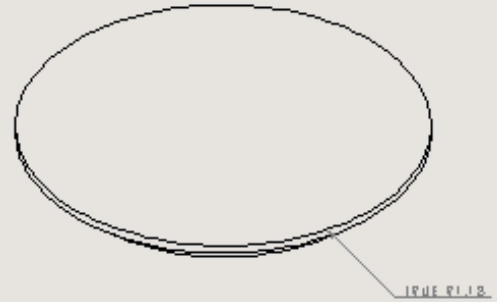
Part	Mass	Cost
Microcontroller - Adafruit Feather 32u4 Basic Proto	4.8g	$19.95
Adafruit Ultimate GPS Breakout	8.5g	$39.95
Adafruit DPS310 Precision Barometric Pressure / Altitude Sensor	10g	$6.95
Alkaline 9V Battery	45.33g	$1.50
3.3V Buzzer	4g	$0.95
Parachute	52.6g	$17.80
Voltage Regulator	9g	$1.25
MicroSD card breakout board+	3.43g	$7.50
TOTAL:	**137.66g**	**$95.85**

CAD Model of Housing









Payload Insert

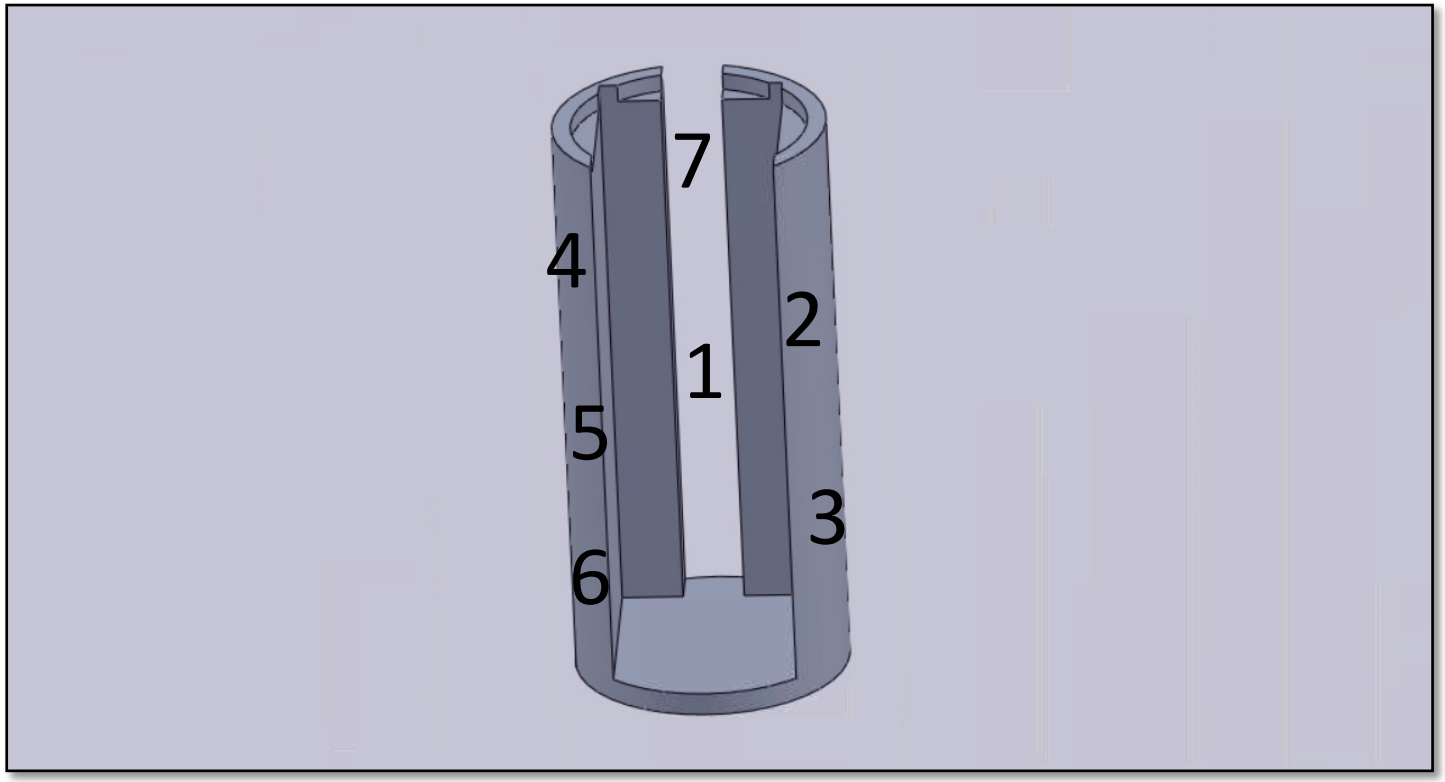
A2

Housing Fabrication Plan

- We will be 3D-printing the housing
  - Cheaper, weighs less, and easier to 'customize'
- We will use 60% infill
- It is recommended to use this % for functional prints, especially since the housing needs to be strong and light
  - We may print a few smaller pieces first (such as the lid) and see how well it holds up
  - Because of the thinness of the walls, a larger percent might make sense so there isn't too much space within the walls

Explanation of how electronics will be integrated into the structure

- Microcontroller at 1
- GPS sensor at 2
- Barometric sensor at 3
- 9V battery at 4
- Buzzer at 5
- External SD Card at 6
- Voltage Regulator at 7



Assembly & Testing Plan

1. Order parts
   - Adafruit, maybe other website, depending on availability
2. Print housing
   - Via Penn State
   - We will be testing the infill percentage at this stage
     - Print a small portion first
   - We settled on 60% but we may move to a higher %
3. Put electronics together on board (As per previous slide)
   - Any Center of Gravity issues will be addressed here
4. Test electronics
5. Once computer is connected, run programming test
6. Put board into housing, test full system

Changes Since PDR

- Added 9 V battery to power sensors
  - It can provide more power
- Added a voltage regulator
  - Better understanding of how the electronics will work
- Updated CAD Model
  - Include lid, small details
- Added SD Card Breakout Board
  - This was needed to store data

Power Draw Breakdown

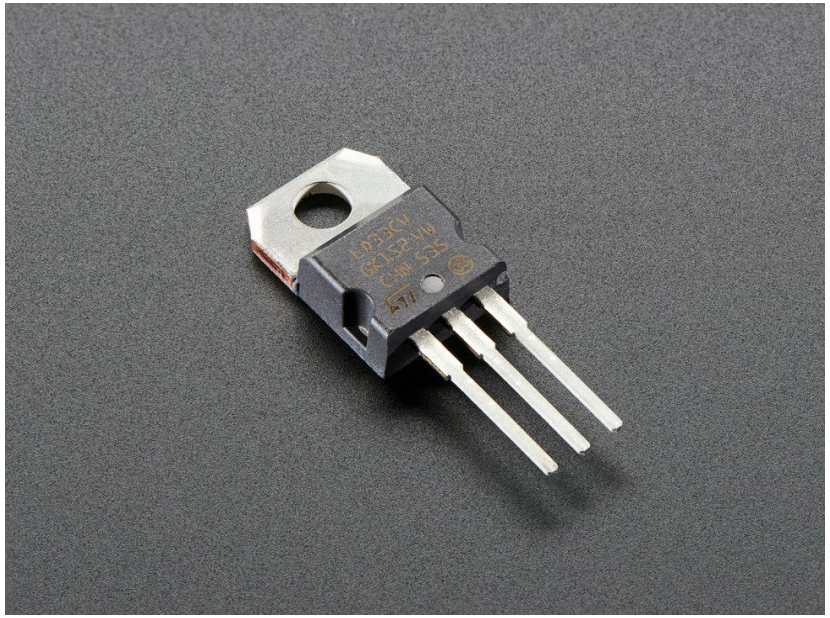
- GPS – .020 A * 3.3V = 0.07W
- Pressure – 3.6 microA * 3.3V = 1.188e-5 W
- Microcontroller - .5 A * 3.3 V = 1.65 W
- SD Card Breakout Board – 3.3V * 150 mA = 0.495W

Energy Source State

- 9V DC Battery
  - 500mAh
  - 4.5 Wh
  - Used to power both sensors, sd card breakout board, and microcontroller

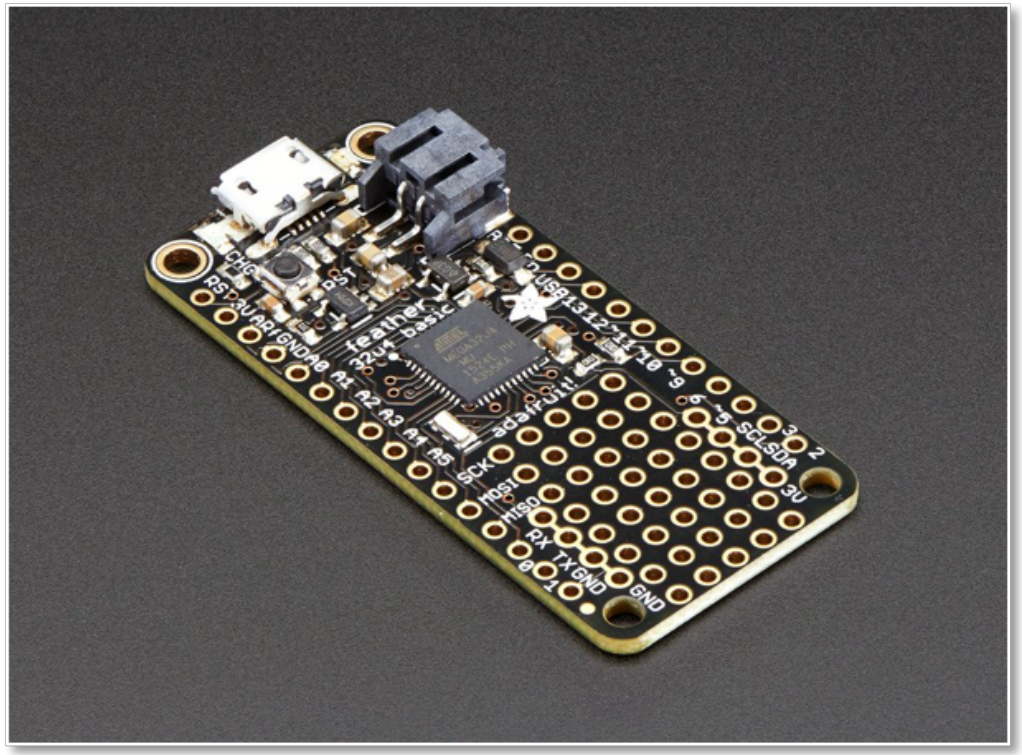
Voltage and Current Regulators

- 3.3V 800mA Linear Voltage Regulator

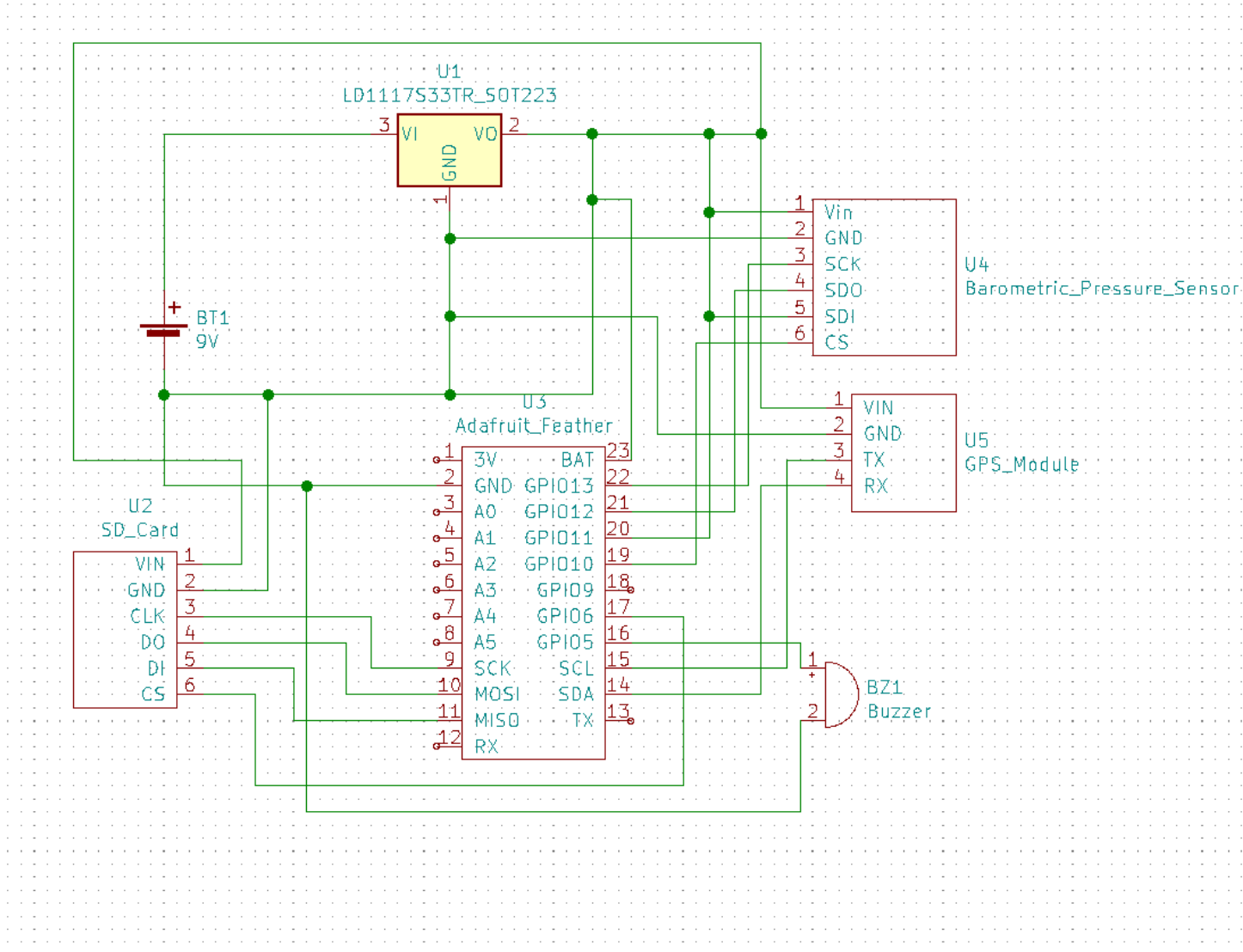


Choice of Microcontroller

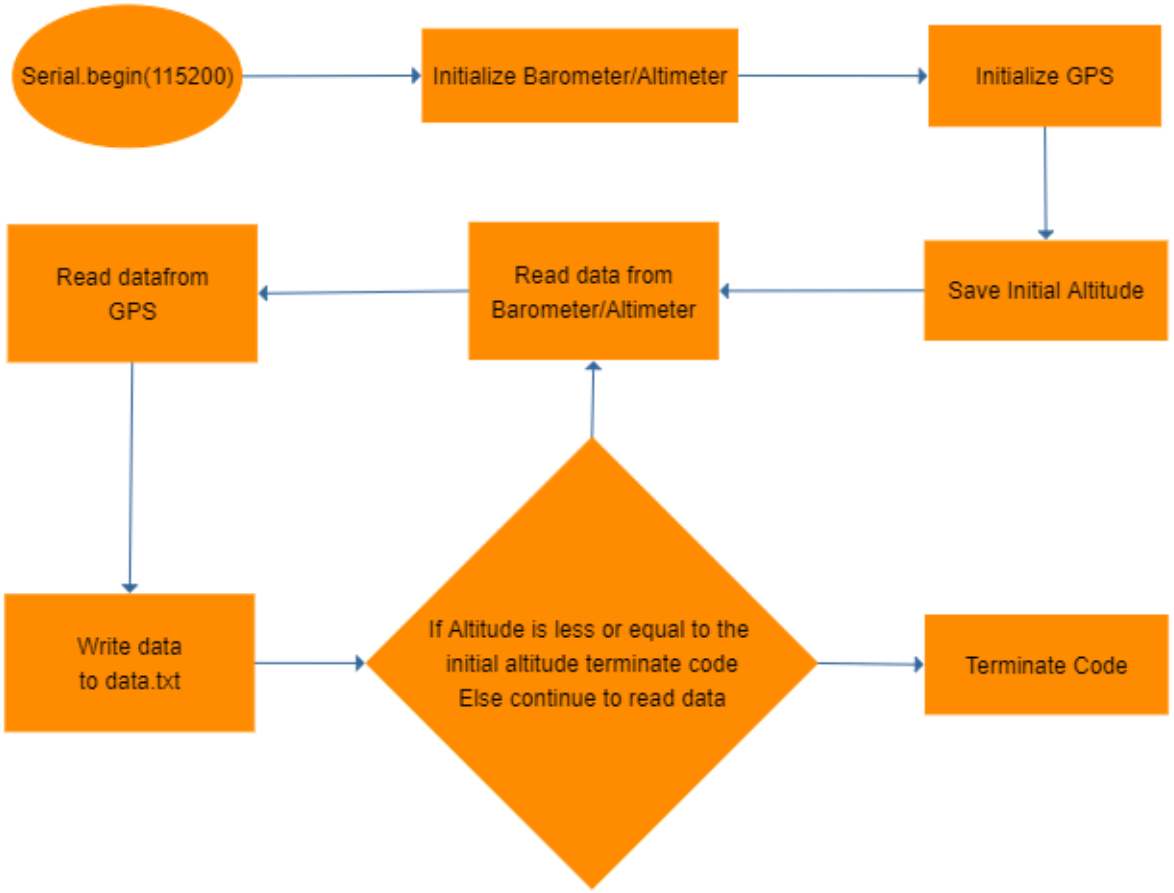
- Our microcontroller is the Adafruit Feather 32u4 Basic Proto
  - Dimensions: (51mm x 23mm x 8mm)
  - Weight: 4.8g
  - Price: $19.95
- Works with our coding skill (C++)



Schematic


U1
LD1117S33TR_SOT223
VI
VO
GND
BT1
9V
U2
SD_Card
VIN
GND
CLK
DO
DI
CS
U3
Adafruit_Feather
3V
GND
A0
A1
A2
A3
A4
A5
SCK
MOSI
MISO
RX
BAT
GPIO13
GPIO12
GPIO11
GPIO10
GPIO9
GPIO6
GPIO5
SCL
SDA
TX
U4
Barometric_Pressure_Sensor
Vin
GND
SCK
SDO
SDI
CS
U5
GPS_Module
VIN
GND
TX
RX
BZ1
Buzzer

Software Con-Ops



Initializing Setup

```
17 void setup() {
18   // put your setup code here, to run once:
19   Serial.begin(115200);
20
21   findBar();
22   setupGPS();
23
24   dps.configurePressure(DPS310_64HZ, DPS310_64SAMPLES);
25   dps.configureTemperature(DPS310_64HZ, DPS310_64SAMPLES);
26 }
```

```
37 void findBar(){
38   while (!Serial) delay(10);
39   Serial.println("DPS310");
40   if (! dps.begin_I2C()) {             // Can pass in I2C address here
41   //if (! dps.begin_SPI(DPS310_CS)) {  // If you want to use SPI
42     Serial.println("Failed to find DPS");
43     while (1) yield();
44   }
45   Serial.println("DPS OK!");
46 }
```

Initializing Setup - Continued

```
67 void setupGPS(){
68   GPS.begin(0x10);  // The I2C address to use is 0x10
69   // uncomment this line to turn on RMC (recommended minimum) and GGA (fix data) including altitude
70   GPS.sendCommand(PMTK_SET_NMEA_OUTPUT_RMCGGA);
71   // uncomment this line to turn on only the "minimum recommended" data
72   //GPS.sendCommand(PMTK_SET_NMEA_OUTPUT_RMCONLY);
73   // For parsing data, we don't suggest using anything but either RMC only or RMC+GGA since
74   // the parser doesn't care about other sentences at this time
75   // Set the update rate
76   GPS.sendCommand(PMTK_SET_NMEA_UPDATE_1HZ); // 1 Hz update rate
77   // For the parsing code to work nicely and have time to sort thru the data, and
78   // print it out we don't suggest using anything higher than 1 Hz
79
80   // Request updates on antenna status, comment out to keep quiet
81   GPS.sendCommand(PGCMD_ANTENNA);
82 }
```

Communications Setup

```
28 void loop() {
29   // put your main code here, to run repeatedly:
30   readBar();
31   pre_printGPS();
32   printGPS();
33 }
```

```
46 void readBar(){
47   sensors_event_t temp_event, pressure_event;
48
49   while (!dps.temperatureAvailable() || !dps.pressureAvailable()) {
50     return; // wait until there's something to read
51   }
52
53   dps.getEvents(&temp_event, &pressure_event);
54   Serial.print(F("Temperature = "));
55   Serial.print(temp_event.temperature);
56   Serial.println(" *C");
57
58   Serial.print(F("Pressure = "));
59   Serial.print(pressure_event.pressure);
60   Serial.println(" hPa");
61
62   Serial.println();
63 }
```

Communications Setup - Continued

```
82 void pre_printGPS(){
83   char c = GPS.read();
84   // if you want to debug, this is a good time to do it!
85   if (GPSECHO)
86     if (c) Serial.print(c);
87   // if a sentence is received, we can check the checksum, parse it...
88   if (GPS.newNMEAreceived()) {
89     // a tricky thing here is if we print the NMEA sentence, or data
90     // we end up not listening and catching other sentences!
91     // so be very wary if using OUTPUT_ALLDATA and trying to print out data
92     Serial.println(GPS.lastNMEA()); // this also sets the newNMEAreceived() flag to false
93     if (!GPS.parse(GPS.lastNMEA())) // this also sets the newNMEAreceived() flag to false
94       return; // we can fail to parse a sentence in which case we should just wait for another
95   }
96 }
```

Communications Setup - Continued

```
98  void printGPS(){
99    if (millis() - timer > 2000) {
100     timer = millis(); // reset the timer
101     Serial.print("\nTime: ");
102     if (GPS.hour < 10) { Serial.print('0'); }
103     Serial.print(GPS.hour, DEC); Serial.print(':');
104     if (GPS.minute < 10) { Serial.print('0'); }
105     Serial.print(GPS.minute, DEC); Serial.print(':');
106     if (GPS.seconds < 10) { Serial.print('0'); }
107     Serial.print(GPS.seconds, DEC); Serial.print('.');
108     if (GPS.milliseconds < 10) {
109       Serial.print("00");
110     } else if (GPS.milliseconds > 9 && GPS.milliseconds < 100) {
111       Serial.print("0");
112     }
113     Serial.println(GPS.milliseconds);
114     Serial.print("Date: ");
115     Serial.print(GPS.day, DEC); Serial.print('/');
116     Serial.print(GPS.month, DEC); Serial.print("/20");
117     Serial.println(GPS.year, DEC);
118     Serial.print("Fix: "); Serial.print((int)GPS.fix);
119     Serial.print(" quality: "); Serial.println((int)GPS.fixquality);
120     if (GPS.fix) {
121       Serial.print("Location: ");
122       Serial.print(GPS.latitude, 4); Serial.print(GPS.lat);
123       Serial.print(", ");
124       Serial.print(GPS.longitude, 4); Serial.println(GPS.lon);
125       Serial.print("Speed (knots): "); Serial.println(GPS.speed);
126       Serial.print("Angle: "); Serial.println(GPS.angle);
127       Serial.print("Altitude: "); Serial.println(GPS.altitude);
128       Serial.print("Satellites: "); Serial.println((int)GPS.satellites);
129     }
130   }
131 }
```

Software Testing Plan

- Data testing will be done to confirm that the code yields the desired results
- Once in the lab, we will hook up the barometer and GPS to the microcontroller using a breadboard and run the code to ensure that the microcontroller is receiving signals from the sensors
- Confirm that the data gets written to the text file in the sd card.