Flight Computer Program Design Report

The program to run the drag system is written in Arduino, which is derived from C. To interface correctly with all of the hardware that comprises the flight computer several libraries need to first be included.

```
    #include <SPI.h>
    #include <SD.h>
    #include <Servo.h>
    #include <SparkFunMPL3115A2.h>
    MPL3115A2 altimeter;
    Servo servoOne;
    File dataFile;
```

The variables that will later be used to calculate the actions of the rocket are then initialized.

```
    long    prevApogee  = 1000,  //  Previous apogee in meters
            altitudeErr = 0.3,   //  Error in the altitude reading [1]
            prevAlt     = 0,     //  Set last loop variables
            prevAcc     = 0,
            prevVel     = 0,
            prevTime    = 0;
```

Booleans are used for many of the actions within the program, these will act like checkboxes to determine whether or not some action has be executed yet. The majority of these will begin as false and be set to true when some event occurs, they are never reset and therefore the program must be reloaded for any consecutive flights.

```
    boolean runDrag     = true,         //  Run drag system this run
            dragOpen    = false,        //  Grag system activated
            hasFired    = false,        //  If engine has been fired
            burnout     = false,        //  If engine has burned out
            apogee      = false,        //  If apogee has been reached
            test        = true;         //  Test variable
```

So as to be able to open and close the drag system multiple times as the rocket slows down, making more and more accurate estimates of the final altitude, an array of tests is set. These are formatted as each test being an array of two integers: the first being the percent of the last apogee at which to begin the test and the second being the percent of last apogee that the test is aiming to approximate.

```
    int     tests[][2]  = {
                {50,85},
                {65,75}
            };
```

Several integers are then set, the majority of which are to declare which pin is used to interface with a specific instrument. The sweep and sweeps integers are used when testing the drag systems motor at initialization.

```
    int     activeTest  = 0,            //  Currently running test
            sdPin       = 10,           //  SD Card Pin
            servoPin    = 9,            //  Servo pin
            sweep       = 0,
            sweeps      = 3;            //  Amount of initial sweeps
```

The last variables that are then initialized are strings that will be used when determining where on the SD card the current flight data will be logged.

```
    String  filename,
            extension   = ".txt";
```

With all the variables initialized, the setup program can begin to attach all the hardware interfaces.

```
void setup(){
```

This begins with the SD card, the initialization for which can be further broken into simpler steps: first the pin is attached, then the name of the current flight is determined based on previously saved flight records, finally headers are printed to help read the recorded data later.

```
    int    i = 0;
    SD.begin(sdPin);
    do{
        i++;
        filename  = "flight";
        filename += i;
        filename += extension;
    }   while(SD.exists(filename));
    dataFile = SD.open(filename, FILE_WRITE);
    dataFile.println("Time\tChange in 
time\tAltitude\tVelocity\tAcceleration");
    dataFile.println("ms\tms\tm\tm/s\tm/s*s");
    dataFile.println();
    dataFile.close();
```

Then the drag system motor is attached to the pin set previously.

```
    servoOne.attach(servoPin);
```

Finally, the altimeter is attached and its basic settings set, followed by the closing of the setup function.

```
    altimeter.begin();
    altimeter.setModeAltimeter(); //  Measures in meters
    altimeter.setOversampleRate(7);
    altimeter.enableEventFlags();
    prevAlt = altimeter.readAltitude();
}
```

After all hardware is attached and variables initialized, the loop can begin. This function is ran continuously so long as the Arduino is powered. It begins by measuring the current altitude and time since powered on.

```
void loop(){
    double altitude = altimeter.readAltitude();
    unsigned long currTime = millis();
```

The data file needs to be reopened every time