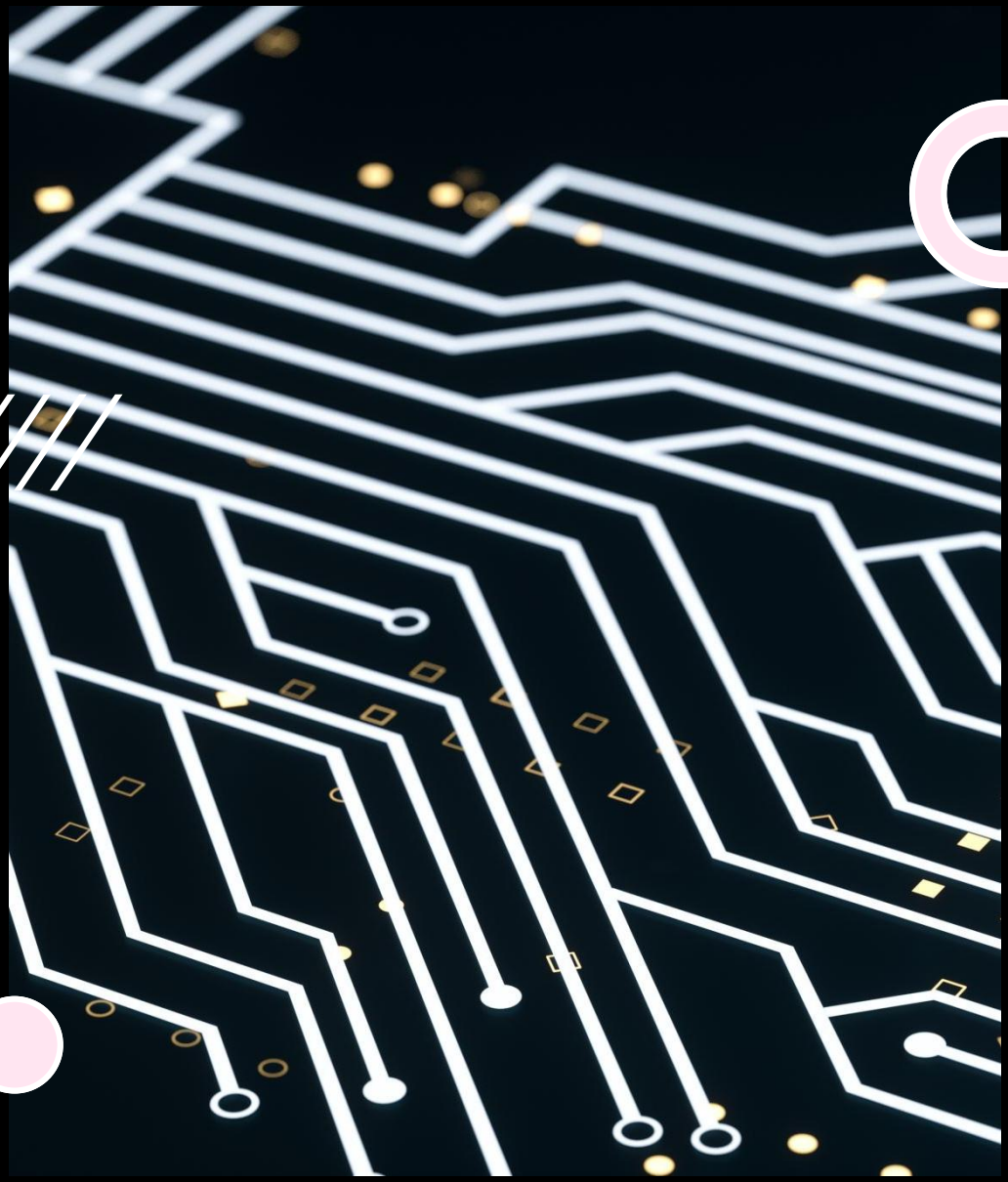
Physics of Rockets

THE SESSION WILL BEGIN SHORTLY

Presented by:
Jacob Lawson

INTRO TO AVIONICS AND FLIGHT CONTROLLERS

JACOB LAWSON



What's the point?

- This is the brain of your rocket, how it knows when and where it is and at what point to deploy the parachute. While the USA managed to get to the moon using the computing power of a calculator your rockets will require a lot more to get to 1500m.

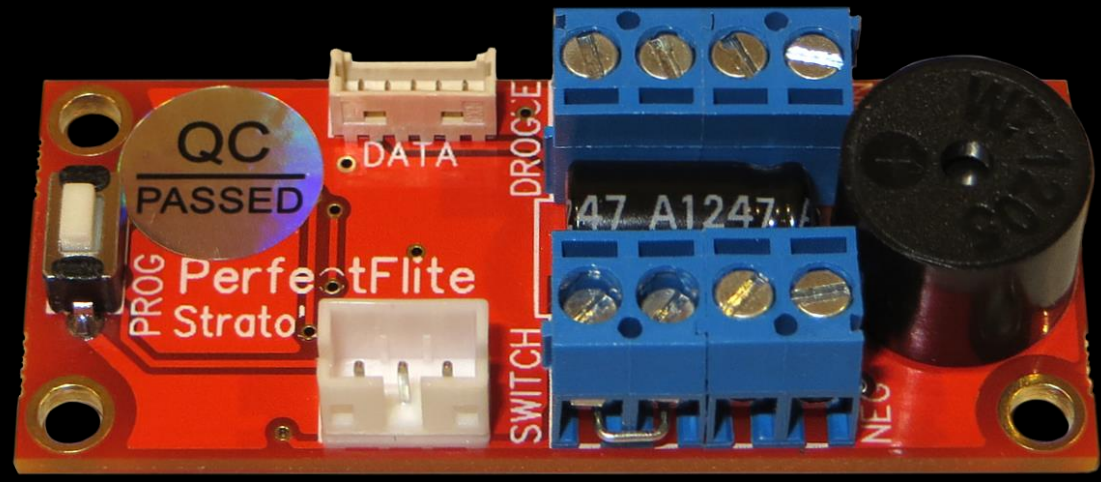
Pre-Built Options

Want to do zero work on your avionics?

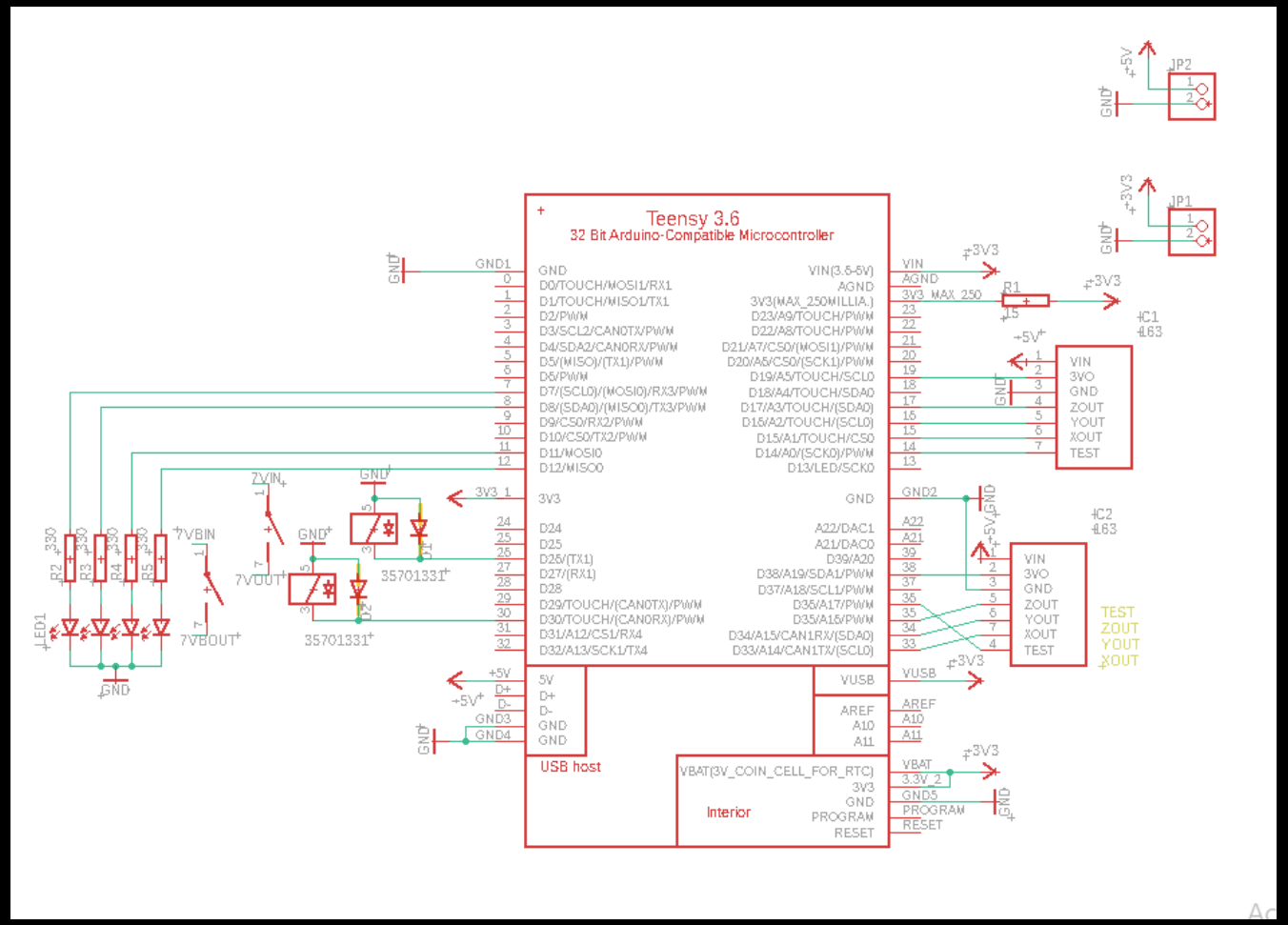
Buy a prebuilt

- No circuit designing
- No coding
- No fuss

PerfectFlite Stratologger CF

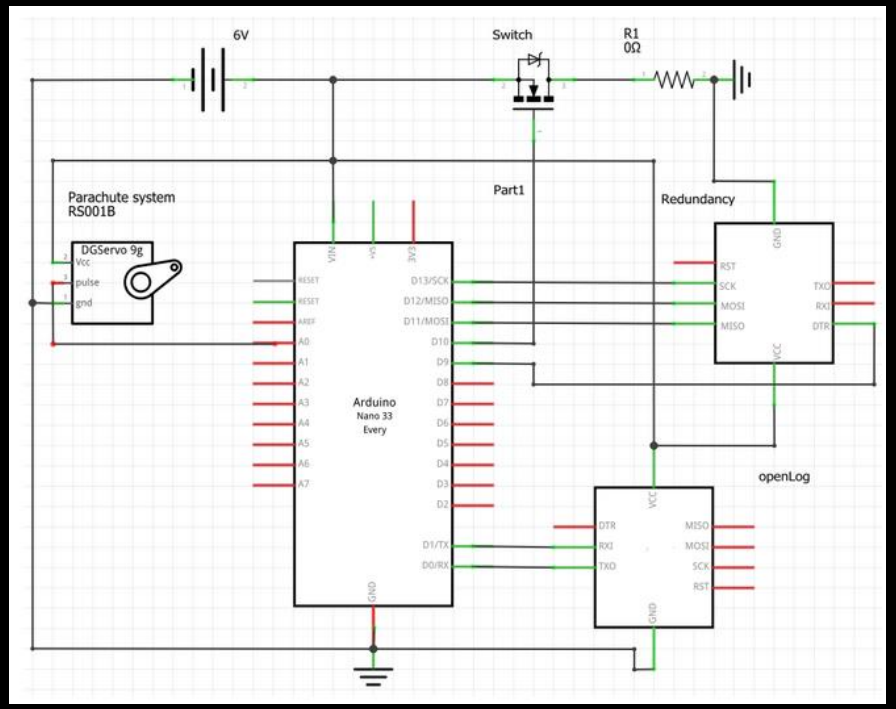


CUSTOM BUILT

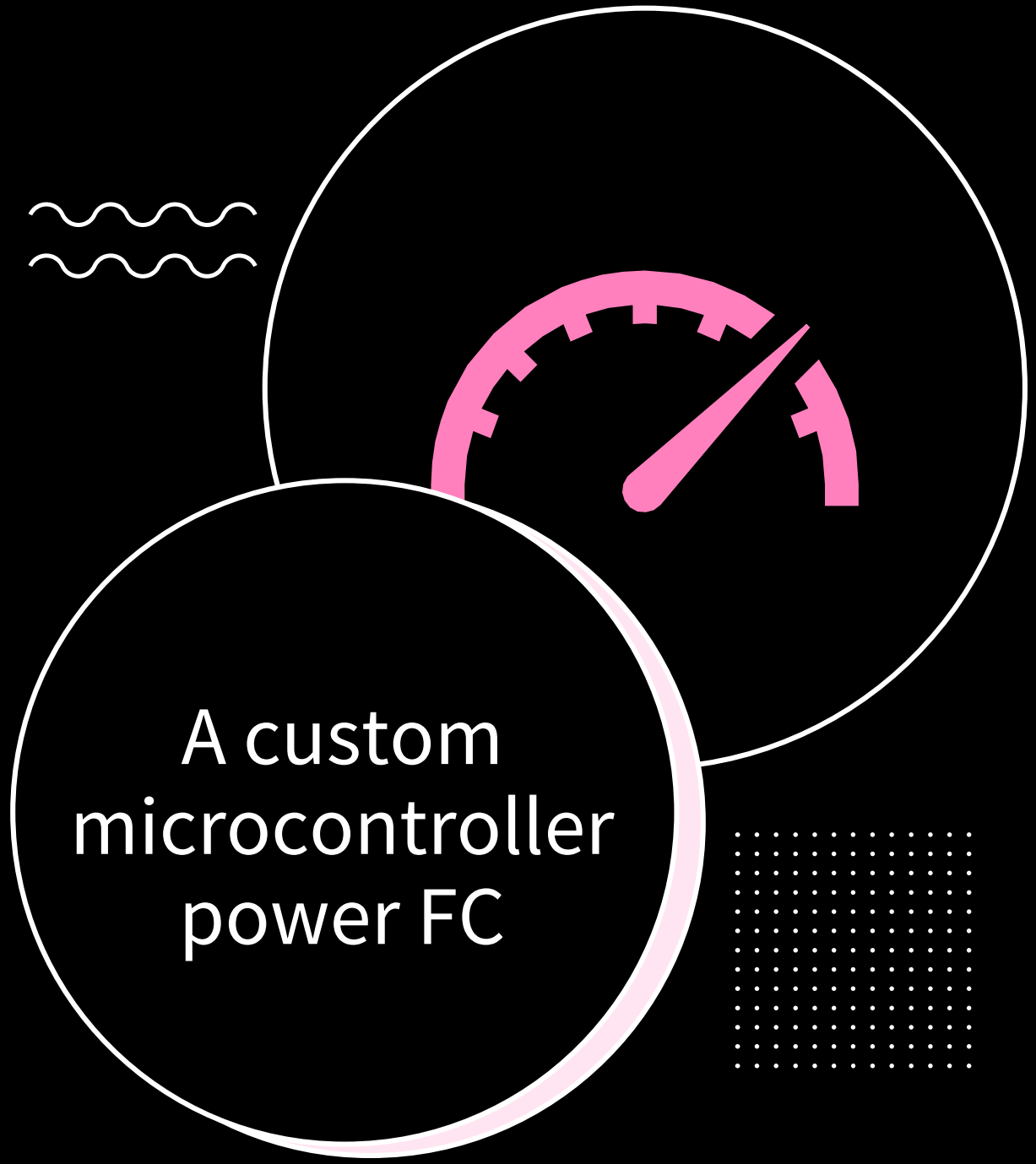


By designing our own circuit using off the shelf components we can create a far more intelligent avionics board

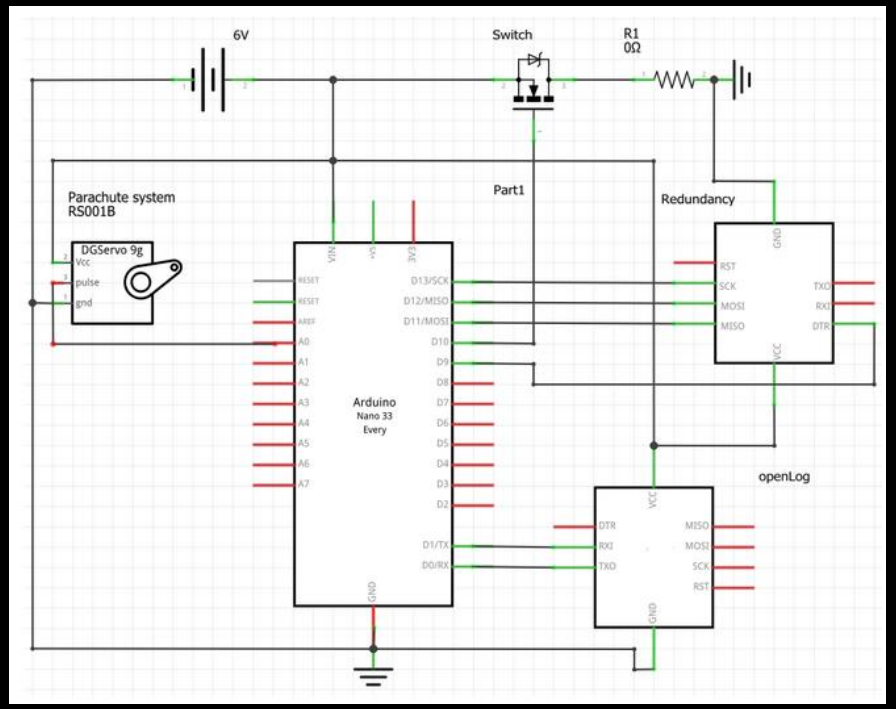
Shown here is an early version of the board schematic for our rocket VESNA



SO WHAT ARE YOU GOING TO BE BUILDING?



- Capable of
  - Power on
  - Parachute deployment
  - Gyroscope measurements
  - Altitude measurements
- https://www.youtube.com/watch?v=R5yhD7-Nu-k
- https://www.youtube.com/watch?v=7PF2xnGsxDs&t=5s



HOW ARE YOU GOING TO LEARN HOW TO BUILD THIS?

Through the follow key stages

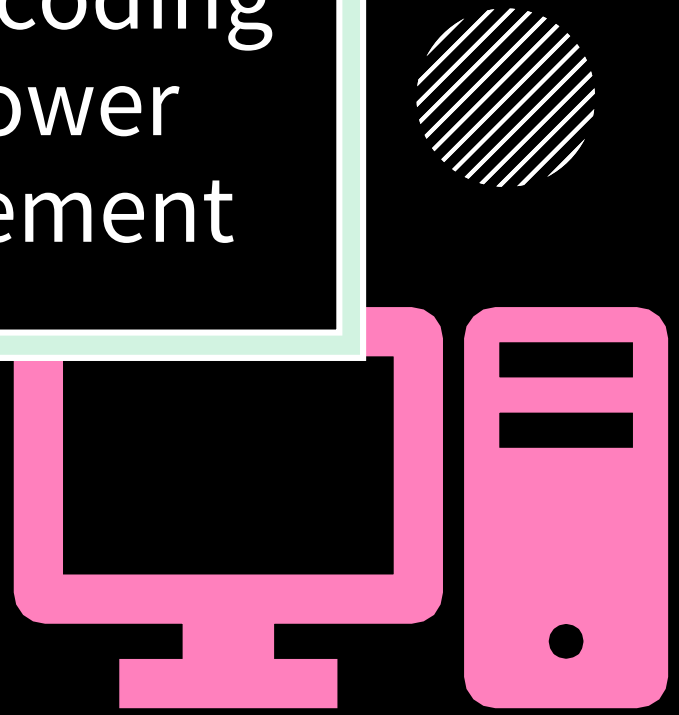
1: Introduction to circuitry, Arduino coding and power management

2: Introduction to designing a Flight Controller schematic in Autodesk Eagle

3: Creating your avionics board in Autodesk Eagle

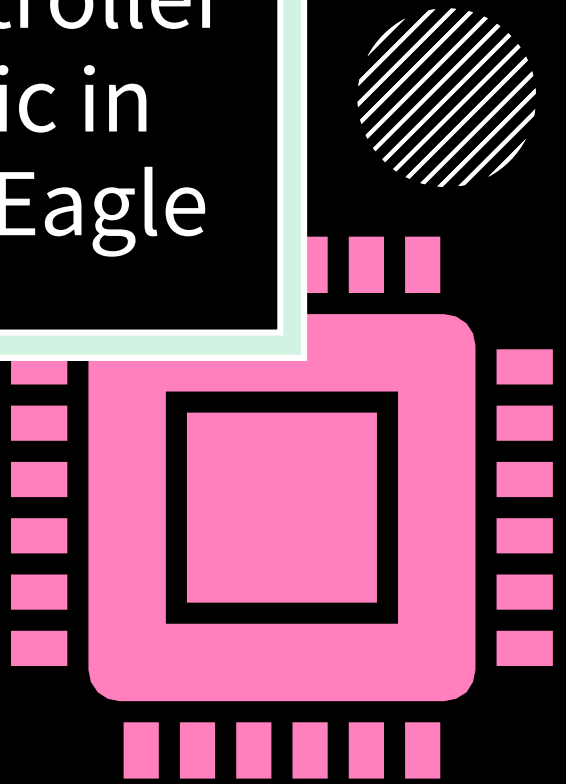
4: Programming your Flight Controller

1: Introduction to circuitry, Arduino coding and power management



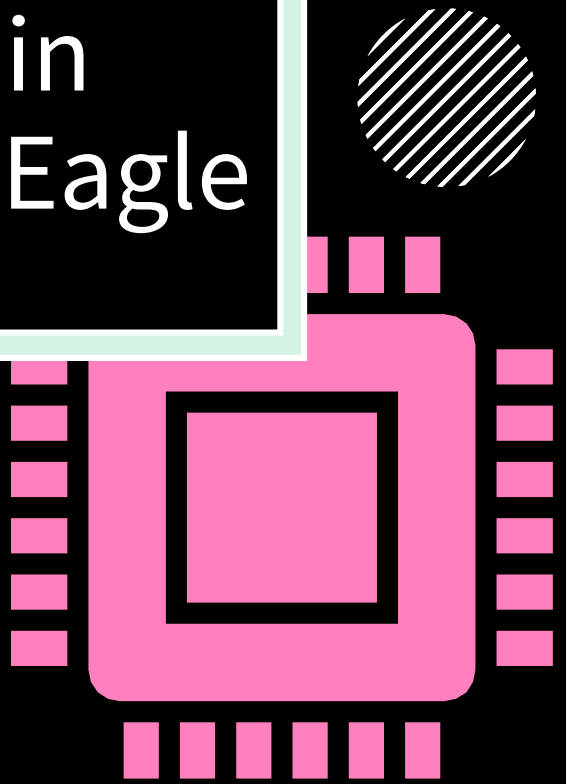
- This training session will cover the basics
  - Building a basic circuit with an online simulator
  - How to make your circuit components interact with each other using Arduino
  - An explanation on power management and stability

2: Introduction to designing a Flight Controller schematic in Autodesk Eagle



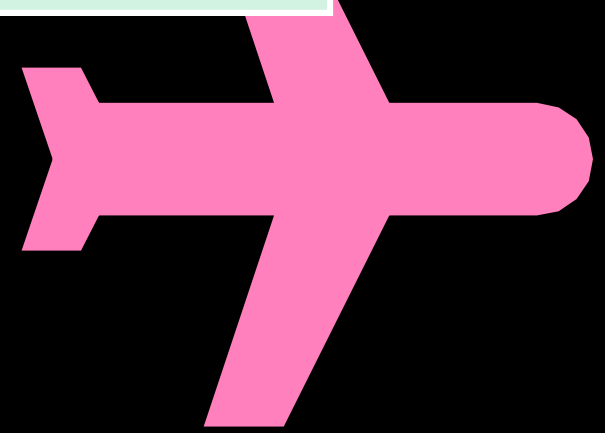
- Designing a circuit using individual components can be daunting, but Eagle simplifies it right down, and we will explain
  - Finding components for your circuit
  - Ensuring pin requirements of your components are met
  - How to be efficient with your design

3: Creating your avionics board in Autodesk Eagle



- We will then take your schematics and use them to design a PCB board
  - Sizing out your board
  - Auto-routing and doing yourself
  - Creating a compact and safe design

4: Programming your Flight Controller



- Then its time to write your code. This is what makes your rocket's avionics truly intelligent and as such it can end up being what makes or breaks your rocket.
  - Accelerometer integration
  - Parachute launching
  - A successful flight

What will you need?

1: Introduction to circuitry, Arduino coding and power management

- Online Arduino simulator
- E.g. TinkerCAD

2: Introduction to designing a Flight Controller schematic in Autodesk Eagle

- Autodesk Eagle

3: Creating your avionics board in Autodesk Eagle

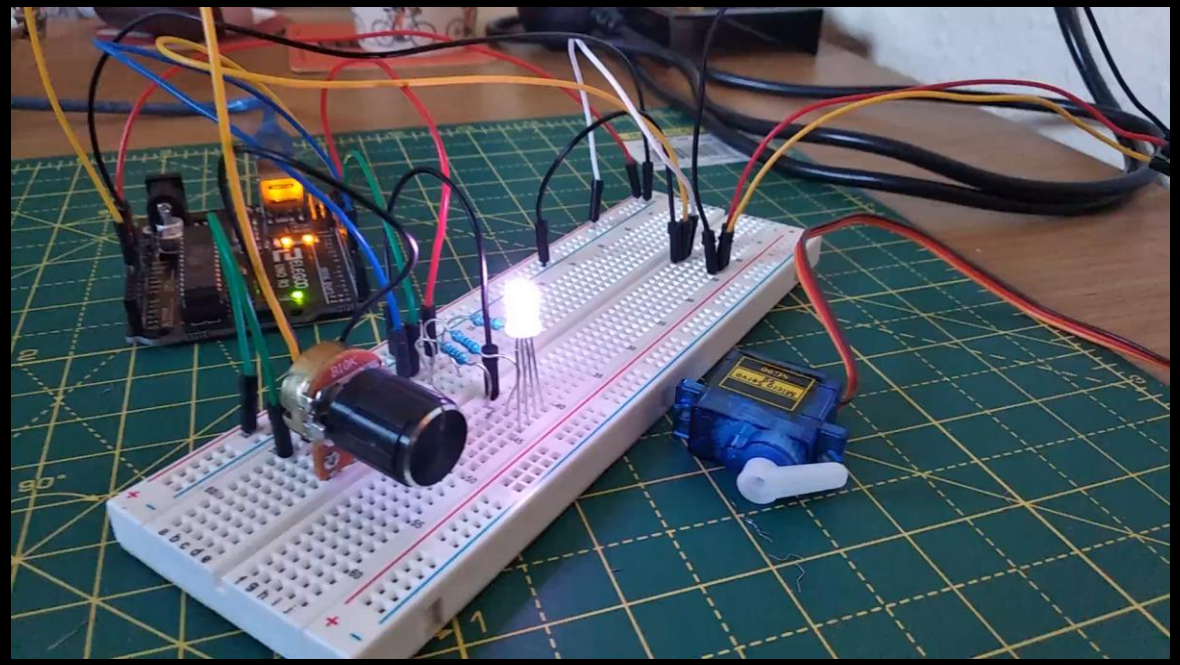
- Autodesk Eagle

4: Programming your Flight Controller

- Arduino coding program

All required programs are subject to change. We will make sure you are told which programs you'll definitely need the week beforehand in the lesson announcement email.

ARDUINO CODING



Potentionmeter_V3_SERVO §

```
#include <Servo.h>  // required library for the commands used below

// RGB LED output ports
#define BLUE 3
#define GREEN 5
#define RED 6

Servo myservo;  // create servo object to control a servo
// twelve servo objects can be created on most boards

int pos = 0;    // variable to store the servo position
int POTPIN = A0;  // potentionmeter input at pin A0;
int POTreadValue; // read value variable
int DataFreq1;    // user defined rate of data collection
int DataFreq2;    // rate of servo control
float Voltage;    // true voltage value across potentionmeter
int redValue;     // rgb RED value
int greenValue;   // rgb green value
int blueValue;    // rgb BLUE value
int ColourCycle = 255;

void setup() {

  // servo setup
  Serial.begin(9600);
myservo.attach(9);  // attaches the servo on pin 9 to the servo object
  myservo.write(pos);
  delay(1500);

  // set pins to output
  pinMode(RED, OUTPUT);
  pinMode(GREEN, OUTPUT);
  pinMode(BLUE, OUTPUT);

  // write to pins
  digitalWrite(RED, LOW);
  digitalWrite(GREEN, LOW);
  digitalWrite(BLUE, LOW);
  analogWrite(BLUE, 100);
  analogWrite(GREEN, 50);
  analogWrite(RED, 150);

  // put your setup code here, to run once:
  pinMode(POTPIN, INPUT);
  Serial.begin(9600);

  Serial.println("Potentionmeter test!");

  Serial.println("Please enter Frequency of data collection");
  Serial.println("Values of 20+ are suitable");
  while(Serial.available()==0){
      // makes the code wait until a number is entered
  }
  DataFreq1 = Serial.parseInt();
  DataFreq2 = 5 * DataFreq1;

  for (pos = 0; pos <= 180; pos += 1) { // goes from 0 degrees to 180 degrees
    // in steps of 1 degree
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);                       // waits 15ms for the servo to reach the position
  }
  for (pos = 180; pos >= 0; pos -= 1) { // goes from 180 degrees to 0 degrees
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);                       // waits 15ms for the servo to reach the position
  }
}

void loop() {
  // put your main code here, to run repeatedly:

  delay(DataFreq1);

  POTreadValue = analogRead(POTPIN);
  Voltage = POTreadValue/204.6;
  // Serial.println(Voltage);
  delay(DataFreq1);

  if ( Voltage > 4.00){

      blueValue = Voltage * 51;
      analogWrite(BLUE, blueValue);
      analogWrite(GREEN, 0);
      analogWrite(RED, 0);
      delay(DataFreq1);

      myservo.write(POTreadValue/5.7);  // tell servo to go to position in variable 'pos'
      delay(15);                        // waits 15ms for the servo to reach the position

      Serial.println(Voltage);
  }
  if ( Voltage < 1.00){

      greenValue = Voltage * 51;
      analogWrite(GREEN, greenValue);
      analogWrite(RED, 0);
      analogWrite(BLUE, 0);
      delay(DataFreq1);

      myservo.write(POTreadValue/5.7);  // tell servo to go to position in variable 'pos'
      delay(15);                        // waits 15ms for the servo to reach the position

      Serial.println(Voltage);
  }
  if ( Voltage > 1.00 && Voltage < 4.00){

      redValue = Voltage * 51;
      analogWrite(RED, redValue);
      analogWrite(GREEN, 0);
      analogWrite(BLUE, 0);
      delay(DataFreq1);

      myservo.write(POTreadValue/5.7);  // tell servo to go to position in variable 'pos'
      delay(15);                        // waits 15ms for the servo to reach the position

      Serial.println(Voltage);
  }

}
```

KNOW YOUR PINS

Your system needs to except data from multiple inputs and control multiple outputs, all in a fraction of a second.

Potentionmeter_V3_SERVO

```
#include <Servo.h>  // required library for the commands used below

// RGB LED output ports
#define BLUE 3
#define GREEN 5
#define RED 6

Servo myservo;  // create servo object to control a servo
// twelve servo objects can be created on most boards

int pos = 0;    // variable to store the servo position
int POTPin = A0;  // potentionmeter input at pin A0;
int POTreadvalue; // read value variable
int DataFreq1;     // user defined rate of data collection
int DataFreq2;     // rate of servo control
float Voltage;      // true voltage value across potentionmeter
int redValue;       // rgb RED value
int greenValue;     // rgb green value
int blueValue;      // rgb BLUE value
int ColourCycle = 255;

void setup() {

  // servo setup
  Serial.begin(9600);
  myservo.attach(9);  // attaches the servo on pin 9 to the servo object
  myservo.write(pos);
  delay(1500);

  // set pins to output
  pinMode(RED, OUTPUT);
  pinMode(GREEN, OUTPUT);
  pinMode(BLUE, OUTPUT);

  // write to pins
  digitalWrite(RED, LOW);
  digitalWrite(GREEN, LOW);
  digitalWrite(BLUE, LOW);
  analogWrite(BLUE, 100);
  analogWrite(GREEN, 50);
  analogWrite(RED, 150);

  // put your setup code here, to run once:
  pinMode(POTPin, INPUT);
  Serial.begin(9600);

  Serial.println("Potentionmeter test!");
```

VARIABLE CONDITIONS

Your launch site is yet to be decided, so make sure to use user defined variables. This way your avionics can be easily modified

Potentionmeter_V3_SERVO

```
void setup() {

  // servo setup
  Serial.begin(9600);
  myservo.attach(9);  // attaches the servo on pin 9 to the servo object
  myservo.write(pos);
  delay(1500);

  // set pins to output
  pinMode(RED, OUTPUT);
  pinMode(GREEN, OUTPUT);
  pinMode(BLUE, OUTPUT);

  // write to pins
  digitalWrite(RED, LOW);
  digitalWrite(GREEN, LOW);
  digitalWrite(BLUE, LOW);
  analogWrite(BLUE, 100);
  analogWrite(GREEN, 50);
  analogWrite(RED, 150);

  // put your setup code here, to run once:
  pinMode(POTPin, INPUT);
  Serial.begin(9600);

  Serial.println("Potentionmeter test!");

  Serial.println("Please enter Frequency of data collection");
  Serial.println("Values of 20+ are suitable");
  while(Serial.available()==0){
        // makes the code wait until a number is entered
  }
  DataFreq1 = Serial.parseInt();
  DataFreq2 = 5 * DataFreq1;

  for (pos = 0; pos <= 180; pos += 1) { // goes from 0 degrees to 180 degrees
    // in steps of 1 degree
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);                       // waits 15ms for the servo to reach the position
  }
  for (pos = 180; pos >= 0; pos -= 1) { // goes from 180 degrees to 0 degrees
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);                       // waits 15ms for the servo to reach the position
  }
}

void loop() {
```

NOT JUST LIKE THE SIMULATIONS

You can simulate your flight on OpenRocket as many times as you like, its still not the real thing.

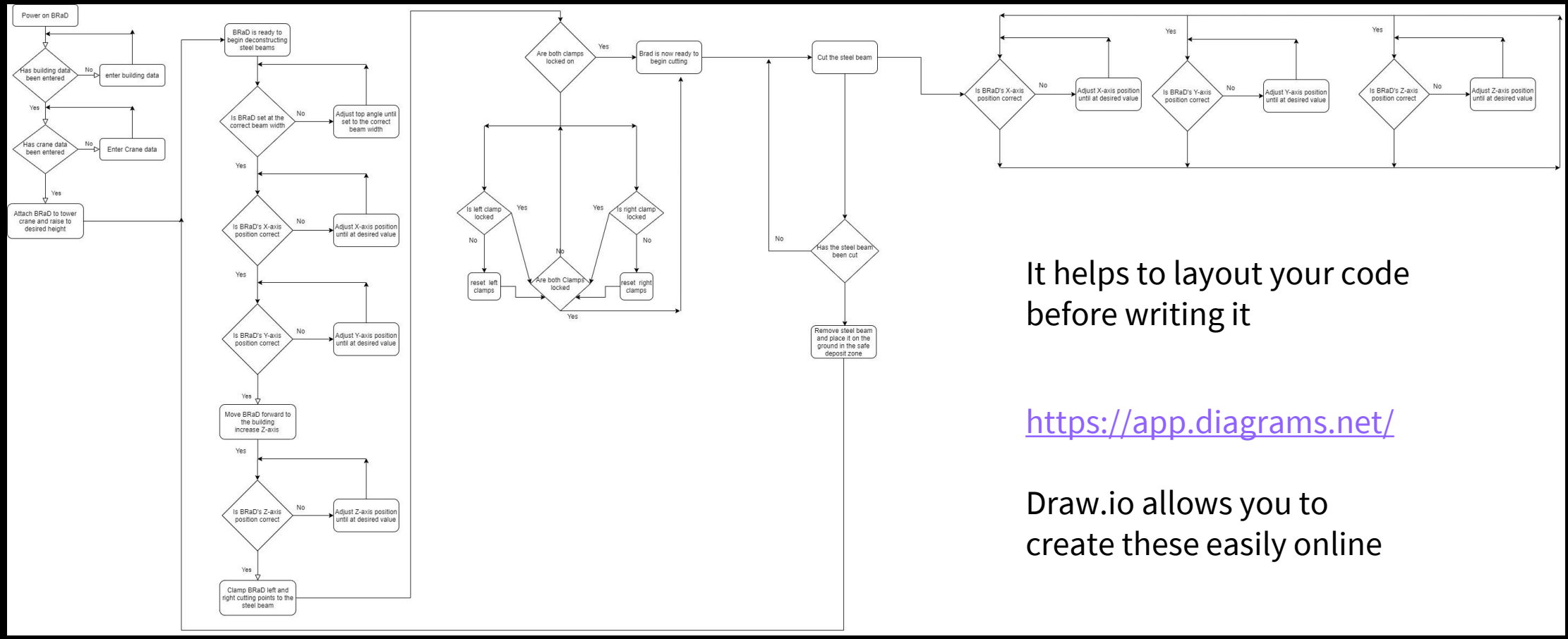
Your avionics must be able to use sensor data to decide when to change state.

There’s no point releasing your parachute before apogee just because the expected time to apogee has passed.


Potentionmeter_V3_SERVO

```
void loop() {
  // put your main code here, to run repeatedly:

  delay(DataFreq1);

  POTreadvalue = analogRead(POTPin);
  Voltage = POTreadvalue/204.6;
  //  Serial.println(Voltage);
  delay(DataFreq1);

  if ( Voltage > 4.00){

      blueValue = Voltage * 51;
      analogWrite(BLUE, blueValue);
      analogWrite(GREEN, 0);
      analogWrite(RED, 0);
      delay(DataFreq1);

      myservo.write(POTreadvalue/5.7);  // tell servo to go to position in variable 'pos'
      delay(15);                        // waits 15ms for the servo to reach the position

      Serial.println(Voltage);
  }
  if ( Voltage < 1.00){

      greenValue = Voltage * 51;
      analogWrite(GREEN, greenValue);
      analogWrite(RED, 0);
      analogWrite(BLUE, 0);
      delay(DataFreq1);

      myservo.write(POTreadvalue/5.7);  // tell servo to go to position in variable 'pos'
      delay(15);                        // waits 15ms for the servo to reach the position

      Serial.println(Voltage);
  }
  if ( Voltage > 1.00 && Voltage < 4.00){

      redValue = Voltage * 51;
      analogWrite(RED, redValue);
      analogWrite(GREEN, 0);
      analogWrite(BLUE, 0);
      delay(DataFreq1);

      myservo.write(POTreadvalue/5.7);  // tell servo to go to position in variable 'pos'
      delay(15);                        // waits 15ms for the servo to reach the position
```

Coding Flow Diagrams

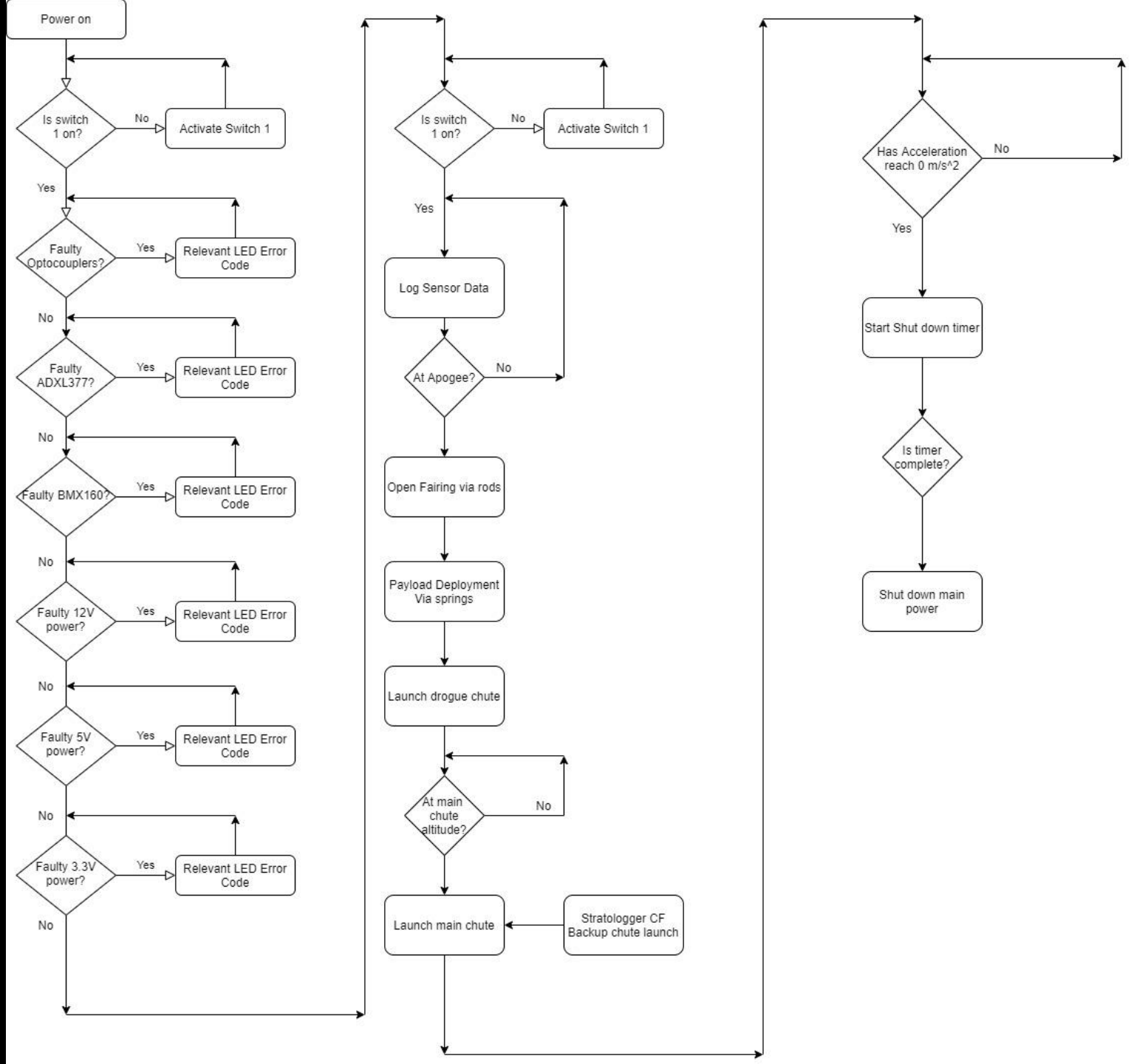


It helps to layout your code before writing it

https://app.diagrams.net/

Draw.io allows you to create these easily online

A Basic Flow Diagram For VESNA



Task

- Use Draw.io to create a coding diagram of the payload you created at the end of Wednesday’s design session.
- You don’t need to overcomplicate it. Just the general outline of your data gathering process.