Failsafe mechanism (Force sensors)

1. Read Piezo resistive sensors for force reading above a certain threshold
2. If 1 of 2 reads below threshold, continue loop
3. If 0 0f 2 read below threshold, continue loop
4. If 2 of 2 read below threshold, stop loop, enter Error deceleration loop

Acceleration throttle control

1. Obtain current speed if possible (in RPMs)
    a. If not use last throttle data send to ESC to estimate speed
2. Read user input value for speed from app connection
3. Divide difference between input speed and current speed
4. Sequentially write increasing or decreasing throttle commands to the ESC
    a. Space throttle commands by a delay set by acceleration parameters

Error “Watchdog” loop to enter deceleration

1. Deceleration loop entered
2. Obtain current speed if possible (in RPMs)
    a. If not use last throttle data send to ESC to estimate speed
    b. Do not allow entrance to loop if speed is 0
3. Sequentially send deceleration commands to the ESC down to 0 speed
4. Flash LEDs red/yellow

Front, Rear, Right, Left, LED control

1. Read user input from app connection
2. Send PWM to MOSFETs to control respective LED strips or send to NeoPixels

Connectivity loss failsafe

1. Continuously check connection
2. If connection is good, allow all loops to proceed
3. If connection is bad or broken, enter Error deceleration loop

Base cases for deceleration error loop

ESC detects low battery and enters its own braking, separate from microcontroller

Broken connection from app

User enters error loop through app command

2 of 2 Piezo resistive sensors read below threshold, indicating rider has fallen off

```
Initialization of variables
Variables for speed/throttle
Variables for LED color
Declare ESC Servo object
Variables for sensors

void setup ()
Attach ESC to correct pin
Attach LEDs to correct pins
Start Serial for debugging

void loop () (main loop)
Call startup function if all conditions are met
        Set speed to 0
Call run loop
        run loop loops within itself, only exits if base/error conditions are met
```

Run loop

```
Run forever
Read all sensor data
Read all throttle/speed data
Read throttle command from app, reassign to desired speed variable every loop
    Calculate difference between current speed and desired speed
     Write first step above/below current speed (fixed step size)
Delay fixed time period (probably 50-200ms)
If any bases cases are met (sensors, connection, user command)
    Call deceleration error loop
    break (Will break to main loop, which will loop around to startup function)
```

Startup

```
While 1 > 0
Check to make sure all base cases are ok and board is ready to go
    Connection
    Kill switch not flipped
    Sensors read ok
If yes, set speed to 0, cool LED sequence proceeds to run loop
If no, continue looping
Delay 500ms
```

Deceleration

```
Called by run loop with 1 arg passed by value, last sent speed
While speed > 0
Read last sent speed
    Calculate difference between current speed and 0
     Write first step below current speed (use fixed step size)
    Store this last sent speed, used by loop for next write to ESC
```

Delay fixed time period (probably 50-200ms)

Break to *run loop* (which will then break to main loop) after speed goes to 0