Alvin, ART385, Blink Without Delay, 3/31/20

Statement: Use Blink Without Delay or MSTimer to create 'simultaneous' multiple outputs (and/or read multiple sensors (for example: apply 'Fading' to all 3 channels of an RGB LED))

Please include a simple **ART385 Design Document, which should be saved as a PDF and include the following sections:**

Document Info

Restate the Assignment

Hand-drawn or digitally drawn schematic diagram (that shows how your components are connected)

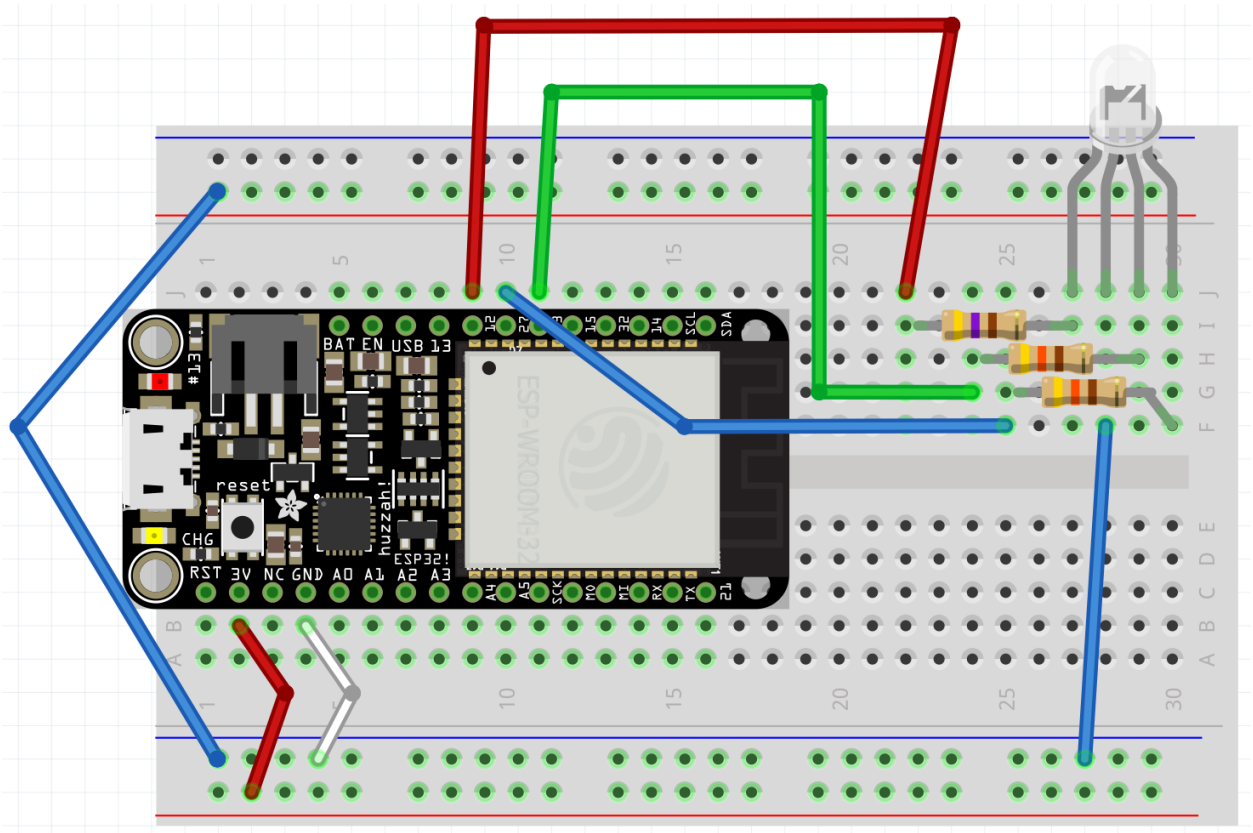
Reflections

You will be assessed on code legibility (specifically the comments), proper structure for the state machine, GitHub management, experimentation, the look-and-feel of the design document itself and following all the instructions.

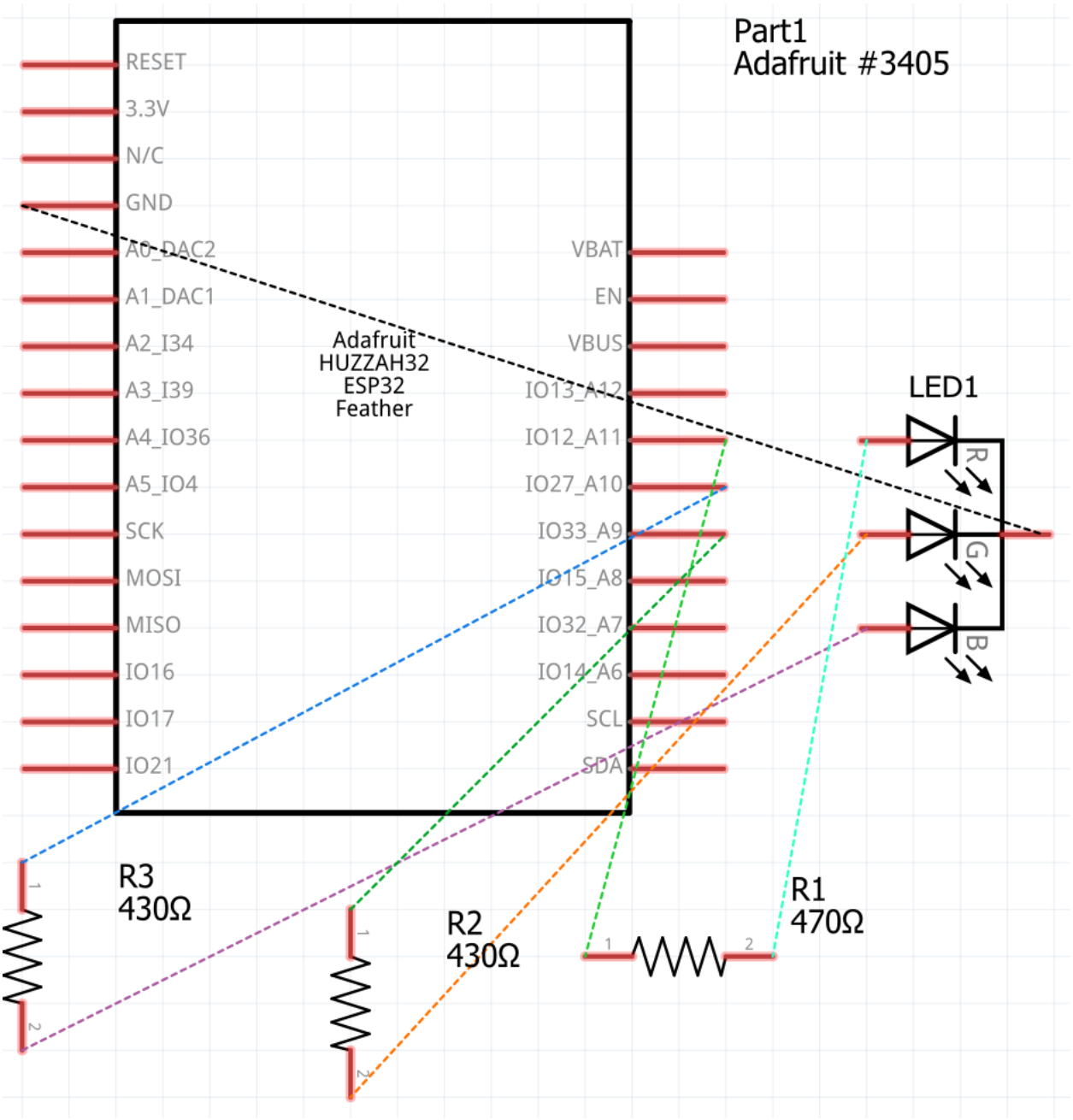
The GitHub repo should be Called **"BlinkWithoutDelay"** and should include a README.md and LICENSE file and the Art385 Design Document (called DesignDocument.pdf).

Hand Drawn Sketches:

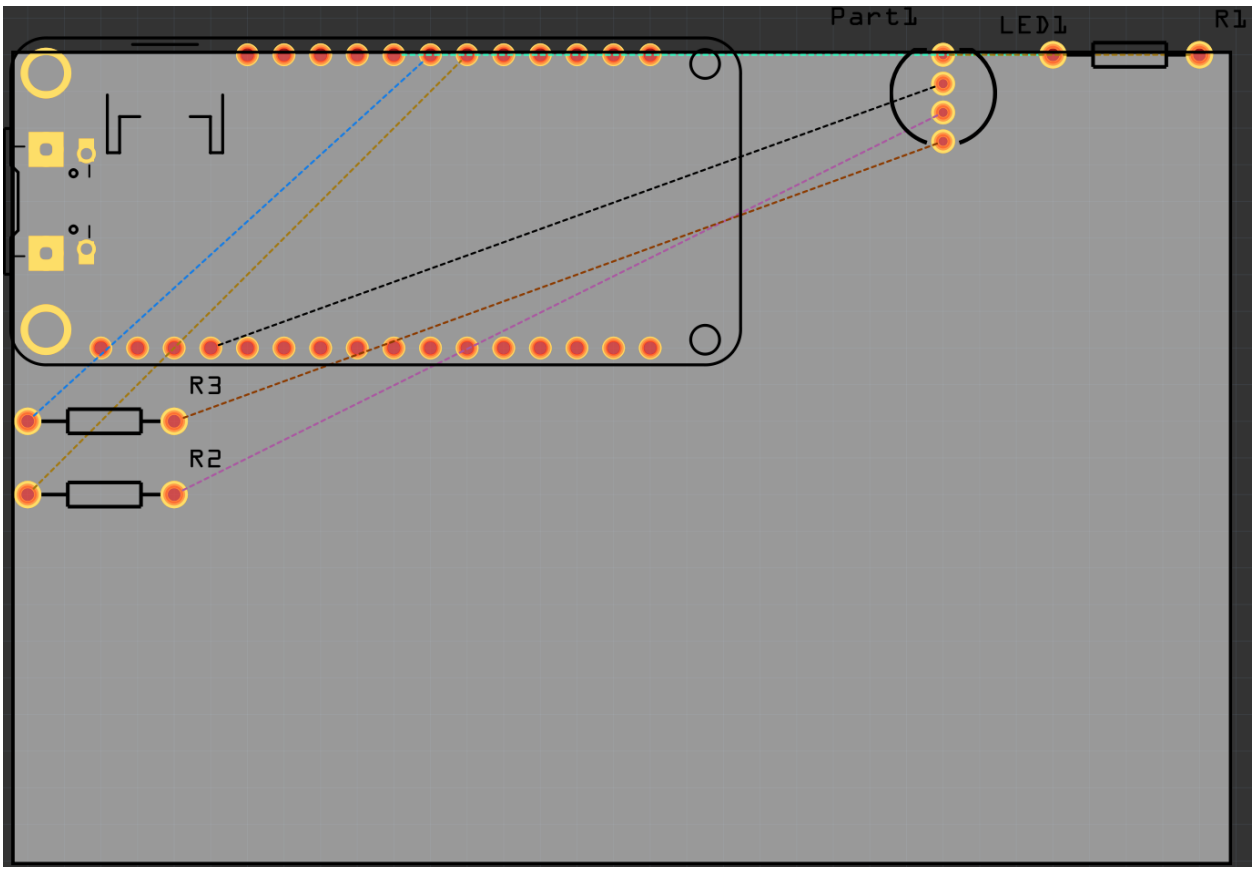
Breadboard:



Schematic:


Part1
Adafruit #3405
RESET
3.3V
N/C
GND
A0_DAC2
A1_DAC1
A2_I34
A3_I39
A4_IO36
A5_IO4
SCK
MOSI
MISO
IO16
IO17
IO21
VBAT
EN
VBUS
IO13_A12
IO12_A11
IO27_A10
IO33_A9
IO15_A8
IO32_A7
IO14_A6
SCL
SDA
Adafruit
HUZZAH32
ESP32
Feather
LED1
R
G
B
R3
430Ω
R2
430Ω
R1
470Ω

PCB:



Observed Patterns/Loops Before Figuring Out How to Reset Counter:

R

G

B

G

R

B

R

G

B

G

R

B

Loop #	Red	Green	Blue
1			
2	**R**		
3			
4		**G**	
5			
6			**B**
7			
8		**G**	

9			
10	**R**		
11			
12			**B**
13			
14	**R**		
15			
16		**G**	
17			
18			**B**
19			
20		**G**	
21			
22	**R**		
23			
24			**B**

Reflections: This was the first assignment I had real trouble with. For one, I had no idea what the directions meant. Once I finally figured out that what the instructions were calling for, the next part was coding…which turned out to be *much* more difficult than I thought. For one, I had to figure out *which* sample code to use; whether the RGB_PWM, BlinkWithoutDelay (Hint: This one!) or the Fade worked the best.

Of course, I had to find out after much debugging that I had wired my board wrong. And then after that, I had to figure out how to trigger my different lights. My biggest problem was trying to turn off any lights I was not using. After *even* more trial and error, I settled on using a counter coupled with a modulo operator.

True, the fact that it was a modulo meant that the RGB pattern was randomized after the first three patterns. But I could not get how to turn off the lights I did not want on. With only two lights, I could get this to work flawlessly. With three lights however, it took some trial and error, and even then, it was somewhat flawed. The lights light up on every other loop (High at first, low on second), so the best I can figure out is a general pattern and the loop pattern. Essentially, for every odd loop, at least one light will light up. I changed the original LED state to "HIGH" and had color increment by one so that the LEDs will light up on every even numbered loop. This allowed me to better track my color and made sure that the lighting pattern was at least consistent, as seen in my notes above.

Finally, I figured out how to reset my color to 0 after every blue light. It originally did not work when I tried to use a for loop, probably because not only did I not figure out the pattern, it also did not reset color correctly. Just one line, and now my lights are cycling on and off in a pattern.

For Project 2, I would most likely use the same board setup, except I would also add a button and the LDR. Of course, it turns out that wiring the board turned out to be the easy part for me in this class. If I follow the rules and color-code my wires correctly, my board should work. It is the coding that is kicking my butt; I consider myself an average coder at best. And if I lack any means to double-check my work on an actual board, I fear that the rest of the semester is going to be miserable for me.