**Quintin Lopez-Scarim**
**February 23rd, 2024**
**23533621**

# Report – Drivers for Embedded Systems

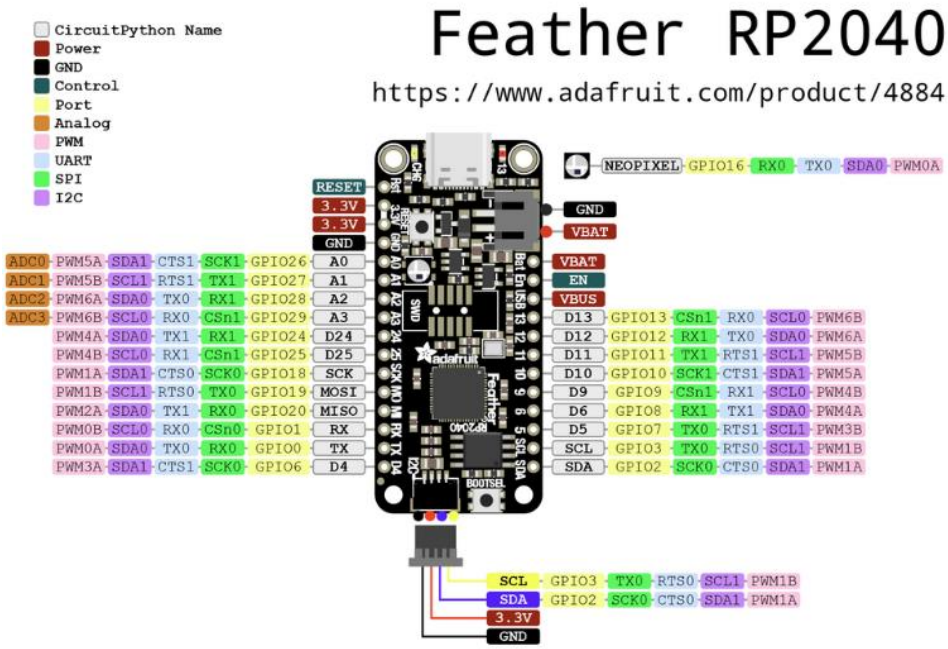
## Introduction

For this lab, hardware in the form of the Adafruit expansions for the rp2040 – and of course the rp2040 itself – were utilized. This includes a matrix of LED's and a board with an accelerometer embedded in it. This lab teaches and re-enforces the methods and build methods that can be utilized by a microprocessor to communicate with hardware expansions, and how to write sufficient and useful drivers for them.

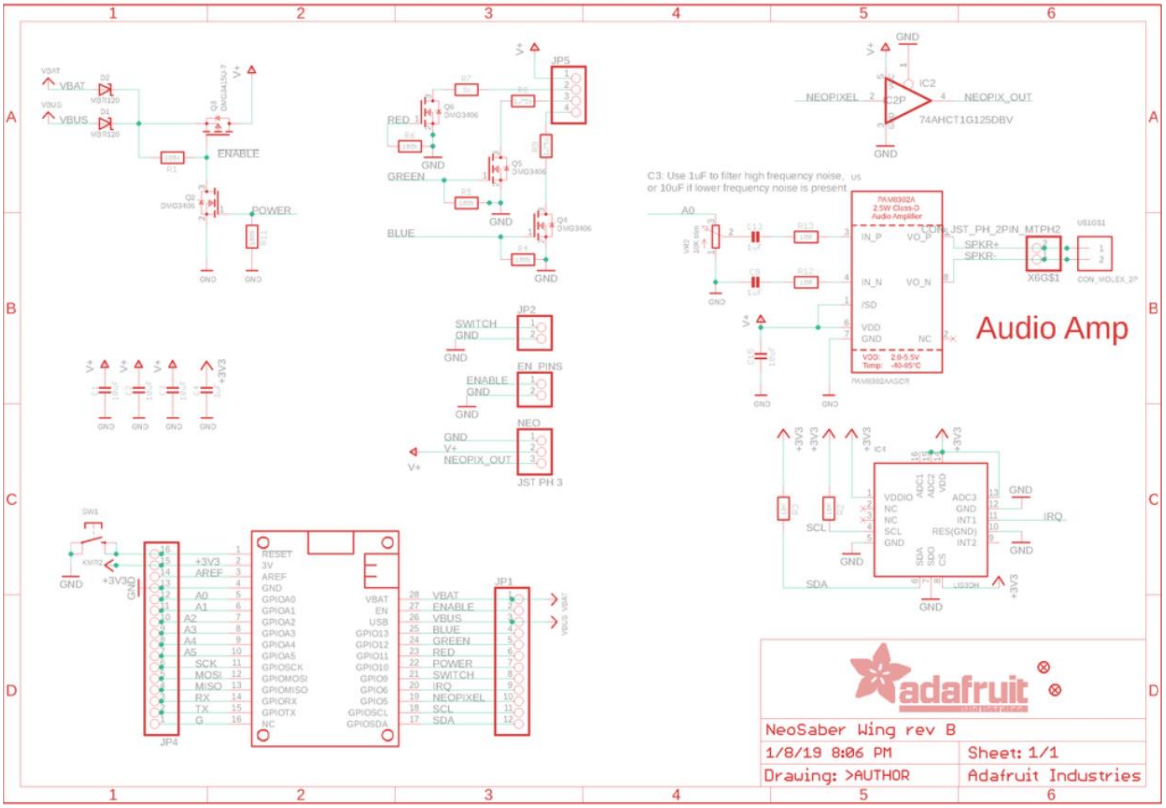
## Design

The design for this lab was oriented around the typical build system described by the quick start manual for the rp2040. It requires importing modules from the pico-SDK and manipulating a Cmake build system to utilize them. There's the added step of using external files, such as ws2812.pio, of which a header file can be made by using certain Cmake commands. The structure of the files is typical of a Cmake library, with a source and generated directory. The actual code structure of the files is rather rudimentary, utilizing SDK functions to call I2C writing, or else re-using code defined in the .pio file. The main level file has polling of the accelerometer readings, with a delay to increase the readability of the output LED's – without it variance in the accelerometer's reading flashes LED's often, which is hard to look at. The only issue that I had was not understanding how the Neomatrix works, with it needing a matrix maintained in the background and every LED being written to every time.

As for knowing which I2C module and pins to use the following schematics were utilized:




Schematic 1, courtesy of: https://learn.adafruit.com/adafruit-feather-rp2040-pico/pinouts



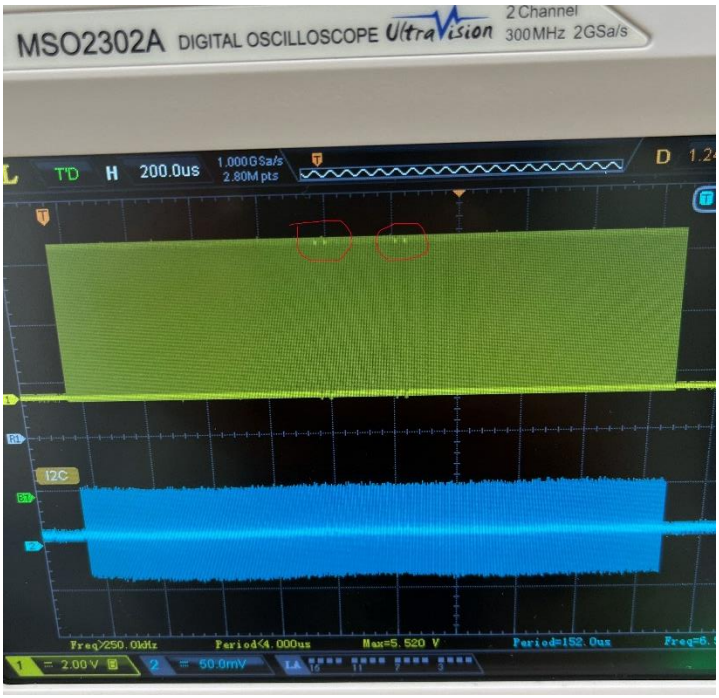

Schematic 2 courtesy of: https://learn.adafruit.com/adafruit-prop-maker-featherwing/downloads

The connection between SCL, SDA, and the accelerometer shown in schematic one and two was important. Additionally in the second schematic, the POWER connection to D10 was of note for supplying constant voltage to the Neomatrix, and the NEOPIXEL line also lets us know to use D5 for communication.

**I2C reading:**


**Comments**: This I2C protocol starts low, but this is likely a result of writing_while_blocking which reserves the line. Other than this irregularity the frame is normal and reads from the x register.

**Neomatrix reading:**




**Comments:**

This protocol uses duty cycles to write, and the four irregularities circled are the protocol writing the 4 green LED's by using a larger duty cycle.

**Conclusion**

The final product functions as expected, with some interesting I2C readings - likely a result of using write_to_blocking and the microprocessor trying to keep the line. I would personally have added a directory for header files for clarity. One of the biggest difficulties I had with this lab was the hardware – I attempted to utilize Lead-free solder with a low heat soldering iron, which led to some very questionable connections, and eventually my Neopixel's lines eventually even snapped. This was fixed by using the better soldering setup in lab, but it showed to me the importance of good equipment when assembling hardware.