Aspiration Assisted Motor-Driven Biopsy Needle

Report: 11-14-2024

1. **Work Planned to be Completed:**

The primary focus of this week was (1) turning the RP2040 into a Mass Storage Class (MSC) device with read and write capabilities. Additionally, (2) I want to have a chosen battery and battery charging system for the device to act in a handheld mode. (3) Another important thing I want to begin is moving over the rest of the current circuit to use the Adafruit Feather RP2040 instead of the Arduino R3’s development board.

1. The USB MSC is a set of protocols that enables a host (PC device) to storage device file transfers. The general idea is that the RP2040 could somehow be made to act like a flashdrive. The RP2040 does not come with built-in non-volatile memory aside from its Flash which is used to store programs typically and is very limited. Therefore, there would need to be another form of storage ideally an SD card and the necessary components to interface it with the development board. Additionally, I would need a library that enables me to write and read from this external storage. After getting this working, I would also need another library (TinyUSB) to make this arrangement act as a MSC.
2. Battery selection is important if we are seriously considering the device’s handheld operation. I need to further investigate available products or any possible alternative power sources.
3. Less important is moving all the current circuitry to function with a new embedded device. I don’t expect that this will be too difficult although it could be good time consuming. It also does introduce a good starting point for some new documentation.*.*
4. **Work Completed:**

For this week I really wanted to get a functional SD card working with the RP2040 which entailed reading and writing capabilities.

A circuit board with a wire

Description automatically generated  
***Figure 1:*** *Current setup to interface microSD card with the RP2040.*

For this to be possible, first I needed to acquire (1) a compatible microSD card and (2) an available adapter that operated on the 3.3V supplied by the development board. SD cards come in many different types such as SD, SDHC, SDXC, SDUC, etc. Each of these occupies a different range of memory i.e. 2GB, 32GB, 2TB but the most important thing was the format of their filesystem. Most libraries that would enable the MSC capability are designed around FAT16/FAT32 formatting in order to maximize compatibility. Most of the smaller sized SD cards are automatically formatted to FAT32 which is the standard for anything 32GB or smaller. Larger cards tend to use exFAT which is not widely supported by libraries but can typically be reformatted to FAT32 which requires a SD card slot or adapter on a PC machine.

The next focus was the adapter which would allow the RP2040 to interface with the microSD using SPI. Many SD cards communicate with a newer and more efficient SDIO protocol which is particularly prevalent among the exFAT and larger variants. However, most general SD cards regardless of their default communication protocol, support SPI. Another consideration was the input voltage to this device as many embedded boards output 5V which could potentially fry the adapter if it doesn’t have the necessary stepdown circuitry. This was not a problem though as the RP2040 outputs 3.3V.

After ensuring that everything was completed hardware-wise, all that had to be done was finding the right libraries and writing the code. In order to utilize the SD card, I configured SPI0 on the RP2040 for communication and used the SdFat library for interfacing. This allows me to write and read to the SD card with the libraries file objects.

Next, was getting this to work as an automated filesystem. Normally we would be able to write to the SD card but actually retrieving the data for our purposes would require removing the card, inserting it into an adapter and then inserting it into a PC assuming there is an available port for it. Instead, I wanted it so that just by plugging in the device via USB, we could easily access all test files. In order to do this, I needed to use the TinyUSB library. I created a Mass Storage Class object which I then configured to read and write directly from the Windows filesystem, a typically complex process that was greatly abstracted. More details on how it operates will be added in a future document.

A screenshot of a computer

Description automatically generated

***Figure 2:*** *The MSC device in action.*

This week was particularly busy due to exams so unfortunately I was unable to complete as much as I would have liked to on the battery front. However, I did do some investigation into an idea from my discussion with Carsten last week: that being utilizing a motor that demands a lower voltage but has the potential to handle the same loads. The general idea is that this would enable me to use a voltage booster (~6V) and a single cell LiPo which would significantly simplify the recharging process and cost. This was particularly enticing as the Adafruit Feather RP2040 board actually comes with a recharging circuit. Additionally, it is far easier to find a charging component/manager without balancing requirements. I did some very simple electrical power calculations but it seems that the current demands would be too high for available boosters.

A paper with text and images

Description automatically generated with medium confidence

***Figure 3:*** *Investigating the plausibility of using the voltage booster assuming motors with similar efficiencies.*

1. **Future Work:**

Now that there is a functioning MSC I would like to get some clarification on how the output information should actually be formatted (i.e. csv organization). Additionally, I’d like to get the sensors, motors, and inputs interfaced ASAP with this new board. Battery selection has taken a lot longer than anticipated and I might need more time to be sure about what to purchase and if there are other available options I haven’t fully investigated. Importantly this means that we could look into stronger motors that would make this design more practical. Completing the PCB library is also now a more pressing task than before.

**Current CEN3907C Due Dates**

|  |  |
| --- | --- |
| *Assignment* | *Due Date* |
| Pre-Alpha Build | 10/25 |
| Driver Lab | 11/1 |
| Elected Lab | 11/15 |
| Design Plan Revision | 11/1 |
| Prototype Presentation | 12/3 |
| Design Prototype | 12/4 |
| Presentation Reviews | 12/3 |