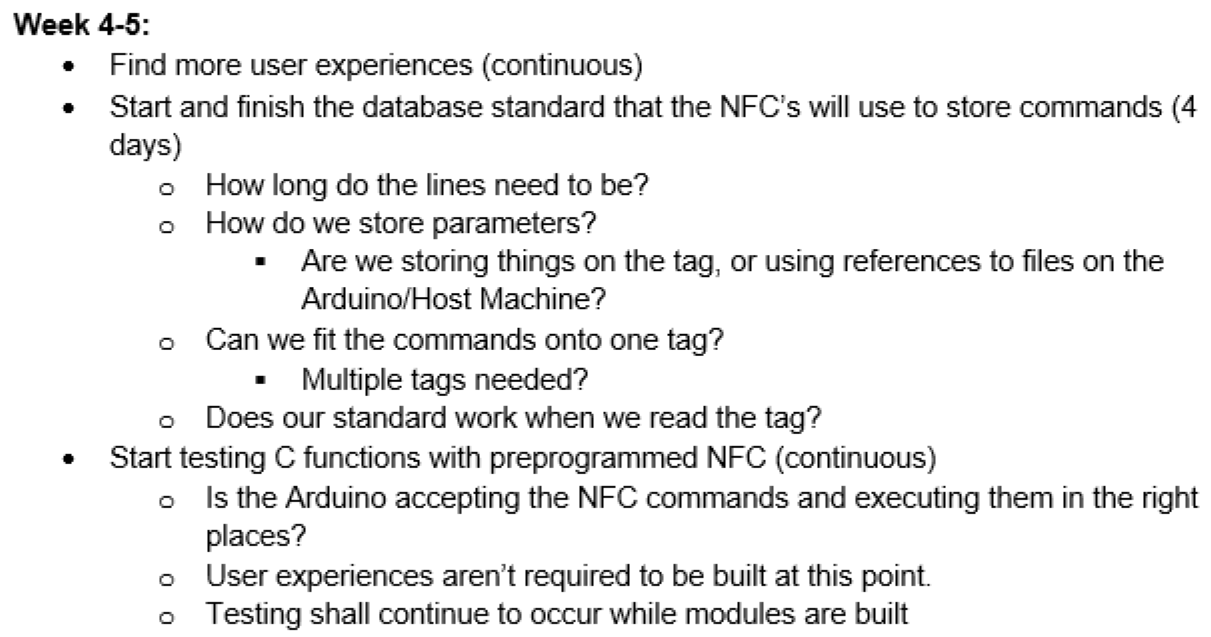
Computer-Driven Experiences with Familiar Objects Week 4

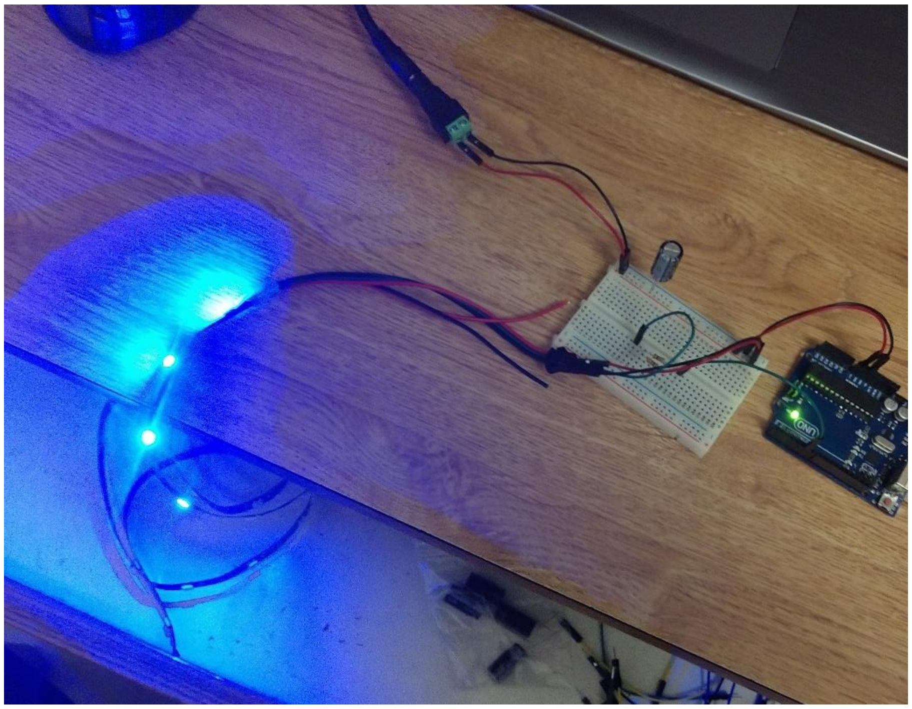


Week 4-5:

- Find more user experiences (continuous)
- Start and finish the database standard that the NFC's will use to store commands (4 days)
    - How long do the lines need to be?
    - How do we store parameters?
        - Are we storing things on the tag, or using references to files on the Arduino/Host Machine?
    - Can we fit the commands onto one tag?
        - Multiple tags needed?
    - Does our standard work when we read the tag?
- Start testing C functions with preprogrammed NFC (continuous)
    - Is the Arduino accepting the NFC commands and executing them in the right places?
    - User experiences aren't required to be built at this point.
    - Testing shall continue to occur while modules are built

As stated in my previous research journal, the user experiences are on hold temporarily until I meet with my stakeholder. I will continue to work on my own experience, which is centered around playing and recording audio, as well as playing some videos I have recorded over the years as a percussionist. The main reason I'm stumped with the experiences is that I don't really know the questions to ask people to get meaningful info about making the user experiences.

In terms of the NFC database, my commands need to be 144 bytes, or 144 characters, long not including header bytes that are used to separated the NDEF records. Each individual command will be stored in its own NDEF record, and I can use URI (i.e. ftp://blah/blah or http://blah.blah/blah) records with custom prefixes to denote commands and store the arguments in the actual URI text while simultaneously keeping byte count as low as possible. There's also the bonus of me being able to link files and their locations directly using the URIs. With this knowledge, I have been programming the C functions that apply to the LED lights that I've ordered. Even though the experiences are on hold I know that the LED lights are the largest, most expensive, and most versatile module, since they will be used in every experience anyways, so it's good to get that module working ASAP. I've bought 5 meters of RGB LEDs and all the requisite passive components and power supplies to run it and the Arduino. You can't really see it in the photo, but the light travels through the strip and then switches to a different color and travels again, I call it a "chase" effect, with the lights chasing each other down the strip.



It might not make a difference to anyone just looking at it, but the Arduino in this photo is running my C++ library right now, I used one the functions I wrote to power the LEDs using the Adafruit NeoPixel library as a base. I've made a few dozen commits already, and as shown above, I've been able to get the CDEFO library in a working state such that I can test the functions I've written so far. The prototypes that matter so far have been written, as can be seen in the GitHub commits. Getting to this point took me about 5 hours this week, and looking at documentation about the LEDs and requisite hardware contributed an extra 2 hours.

I can also verify that there aren't any weird quirks using Arduino and Visual Studio IDE at the same time. There were a few issues in the beginning involving the lack of RAM in the Arduino. I found that when I used multiple instances of the NeoPixel class the Arduino would run out of RAM and crash, so I needed to use pointers instead of making new instances every time I try to run the LEDs. Also, I've decided to make everything a static method for the CDEFO class because it's the easiest to understand for someone trying to implement it, and again I don't have to worry about RAM running out as easily. There is a chance I might need to buy a larger Arduino if the lack of RAM really becomes an issue.