GNG1103

Deliverable F: Prototype I

3D Printer Monitoring System

Submitted by

GNG 1103, Section A3, Group 1

Magdalena Richardson, 7231925

Qaiz Mohamed, 300072323

Amesh Roy, 300073404

Midas Chin, 300066761

Lucas Hubert, 8386891

October 31, 2019

University of Ottawa

Introduction

The objective of this deliverable is to develop the first of three prototypes involved in this project. This prototype will essentially show the proof of concept and will be improved/ refined for further prototypes. Each critical component and system will be created and simple analyses on each of them will be carried out. A non - functional user interface using Ross Dashboard, a preliminary sd card holder, a function node MCU integrated with dashboard, and a basic electrical model will all be created for this prototype. This prototype allows the team to make essential observations with the functionality of the components as well as receive feedback that will be implemented.

Hardware Subsystem

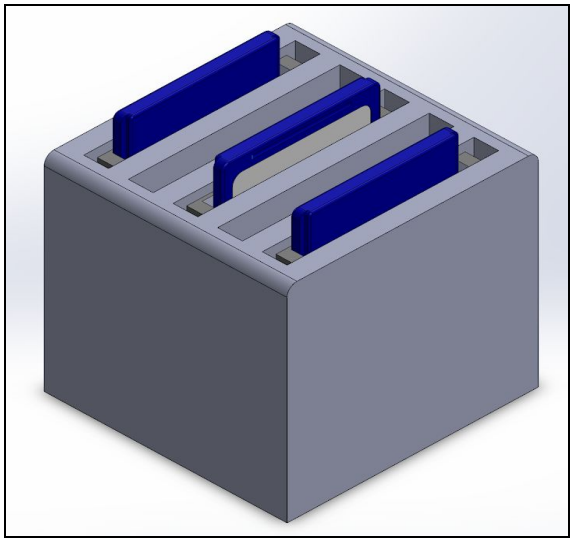


Figure 1 - 3D model of the SD card holder.

For prototype one, a basic SD card holder was modeled using SOLIDWORKS and was 3D printed in order to show proof of concept and serve as a focused prototype (Figures 1,2,3,4).

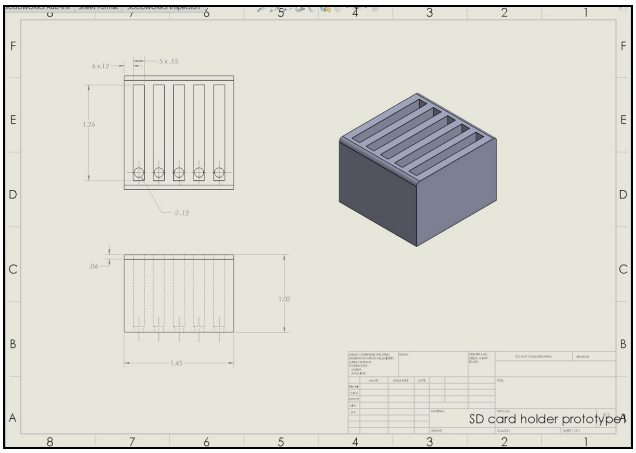


Figure 2 - Technical drawing of the first SD card holder

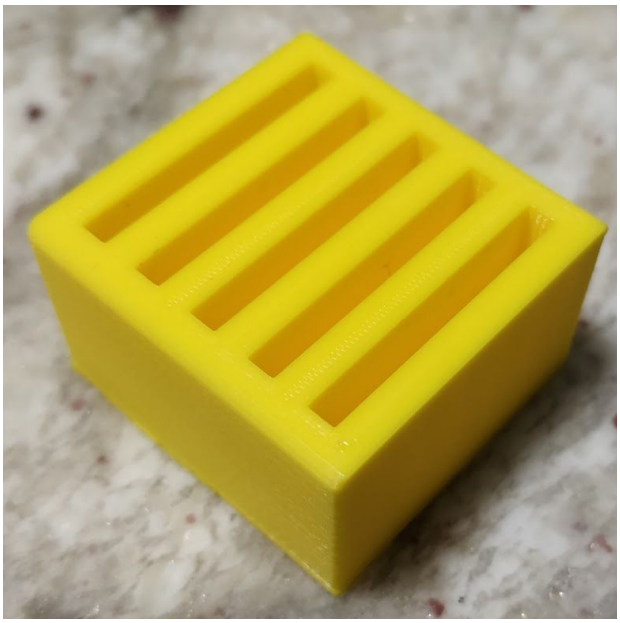


Figure 3 - The first prototype of the SD card holder.

The test objectives/ goal is to see if the other hardware components such as the SD socket fit into the card holder and how much extra room there is for wiring. The results that will be obtained will be a simple yes or no answer to whether the components fit. Using this test will allow for refinements to the 3D model in order to develop a better card holder for future prototypes. The two main criteria that will be used to test the model are the following:

1. Does the SD card socket fit within the holder?
2. If so, is there enough space to allow multiple wires to enter and exit each slot?

In addition to qualitative observations, some quantitative data will be recorded if possible to allow for further improvements to the model. The measurements recorded will be the amount of space left (if socket fits), or what the required dimensions will be for the future prototypes and will all be recorded in mm. The material being used for this sub component is PLA obtained from the makerspace 3D printers and will approximately cost $1.20 (refer to appendix in deliverable E for cost calculations). The only dependency for this test would be receiving the ordered parts in time therefore allowing the team to test the model.

After printing out the initial 3D model and receiving the necessary parts, a test of the first prototype was conducted. It was determined that the SD card sockets would not fit into the card holder as the slots created were not thick enough to allow the socket to fit within the slot. This can be accounted for as the initial design was made in accordance to a different type of SD card socket than the one being used. In addition, it was also determined that the hole for wires passing through would only allow for 1-2 wires to fit, however, a minimum of 3-4 are required to make connections with SD card socket. Furthermore, it was also decided through a group consensus that it would be ideal for the socket to be level with the top of the SD card holder.

After the testing of the holder it was unquestionable that further improvements were required, especially regarding the socket having to fit within the holder. To start, measurements were made and obtained about the dimensions of the socket and the space required for it to fit. The table below displays the obtained metrics.

TABLE II

Various measurements obtained from testing the prototype.

Measurement of socket	Value (mm)
Height	30.5
Thickness	2.9
Width	30

Using the metrics obtained from the item's data sheet, new dimensions for the slot holder were developed allowing for extra space within each slot. The dimensions that will be implemented are slot widths of 3.75mm with 3mm spacing in between each one, a slot depth of 31mm, and a slot length of 32mm. These new dimensions make the SD card holder 38mm in length, 36.75mm in width, and 31.5mm in height.

In addition, the other problem that needs to be fixed for the next prototype is the entrance and exit for wires that will allow 3-4 wires going in and out. A solution that will be implemented in the next prototype regarding this problem is to remove the holes on the bottom of each slot

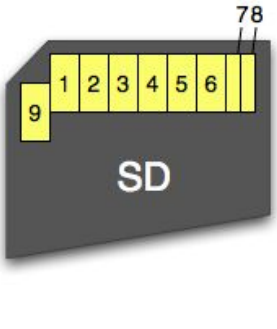
and to create thin slots running along the bottom of each main slot. Since the slots on the bottom will be small, this prevents the socket from falling out and allows for easy, direct access to the socket pins.

Going forward, many additional adjustments and add-ons will be required for the SD card holder and will be implemented in the following prototypes. One essential add-on is a housing unit for the node MCU and multiplexer that will be directly attached to the card holder. A potential idea for this is to have a hollow housing unit sitting under the card holder with slots or holes to allow for wires passing from the SD socket to the multiplexer and then to the node MCU. Currently, the most viable option to achieve this is to use interlocking jigsaw blocks that can be slid and held firmly in place. Another add-on to the current model is to have the makerspace logo displayed on the front of the card holder to make it visually appealing.

Electrical Subsystem

Background Information

In theory, we could sense the presence of the SD card by attempting to read its contents - this would be a form of 'digital sensor'. The minimum pins required to read data[1] are CLK (clock), CMD (command) and DAT0 (data) as well as VCC (power) and GND (ground). This would require running 5 wires up to the SD card, and it would also require some kind of communication protocol to ensure that we do not damage the card when we take it out. The algorithm would be much more involved than using the card detection switch, however it is a possible alternative that we have if we need it.

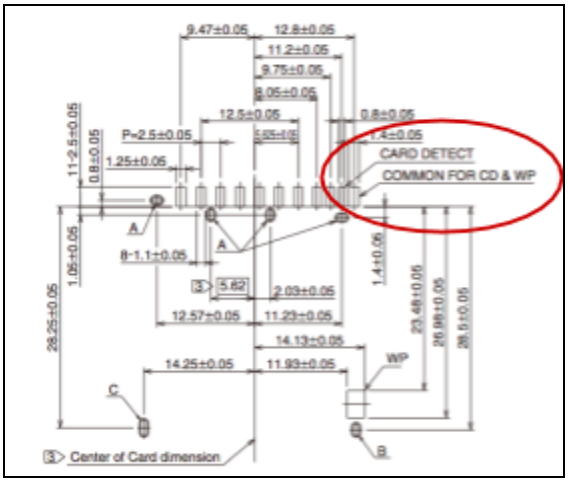


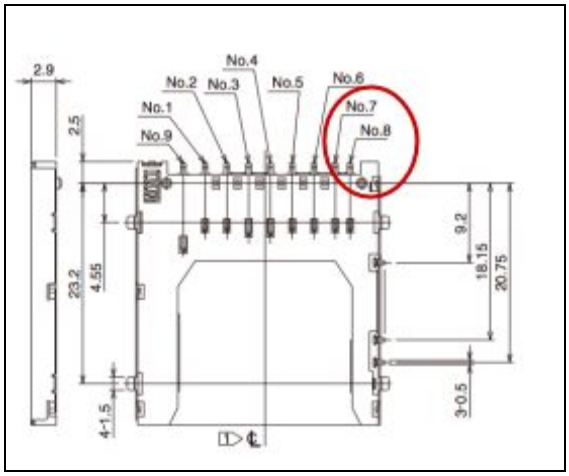
Pin	SD	SPI
1	CD/DAT3	CS
2	CMD	DI
3	VSS1	VSS1
4	VDD	VDD
5	CLK	SCLK
6	VSS2	VSS2
7	DAT0	DO
8	DAT1	X
9	DAT2	X

Figure 4 - Pin diagram of an SD card. Image from http://www.electroniccircuitsdesign.com/pinout/sd-microsd-card-pinout.html.

[1] Pin information from https://electronics.stackexchange.com/questions/188999/card-detect-pin-of-sdhc-interface.

On the other hand, the card socket we used is equipped with a mechanical card detector as well, which will short-circuit pins 7 & 8 when there is a card visible.





Figures 5 & 6 - Technical drawings of the mechanical card detection switch. Image from https://www.digikey.ca/product-detail/en/DM1B-DSF-PEJ(22)/HR846CT-ND/559986.

Test Plan

This was the first feasibility test of our electronic subsystem where we learned more about how we will develop our final product. We need a sensor that will tell our microcontroller if a 3D printer is being used, so we chose to sense the presence of SD cards in a card holder similar to the one currently in use at Makerspace. The physical sensor is the SD card socket, which can sense using one of two methods: it can sense digitally by reading data from the card, or it can sense mechanically by using the card-detection switch on the socket (which creates a short-circuit when the card is in the socket). As the mechanical detection method requires only 2 wires compared to the 5 wires required for digital detection, our first prototype is a proof-of-concept combined with a visual diagram of the circuit. This type of prototype allows us to work on several facets of the product at once: by designing the circuit first, we can plan our physical housing component and plan our software, and by testing out a single component at a time we can easily troubleshoot and find alternatives if necessary.

The goal of this test was to ensure that this method 1) accurately detects the presence of the SD card, and 2) is reliable in all of the card sockets that we purchased. This would tell us if we could continue developing our product according to our current plan as it would inform how we program our microcontroller and how we design the circuitry of our system. We were prepared for the possibility that our results would show us that the card socket would not be a viable sensor, either because the mechanical switch was unreliable or because the sockets we purchased were poor quality. If that is true, we would have had to re-evaluate other forms of sensing (ex: digital sensing, external sensing) or consider other models of card sockets. We

determined our test would be a success if we could create a positive detection result 100% of the time on at least 5 cards, which is what we will need for a final product prototype.

The procedure for this test is as follows:

1) Build an accurate circuit diagram. We used the program 'Fritzing' (which is a PCB manufacturer) because their software contains a big library of pre-designed parts that look and connect exactly like many common electronic components, including the NodeMCU and an SD card socket.

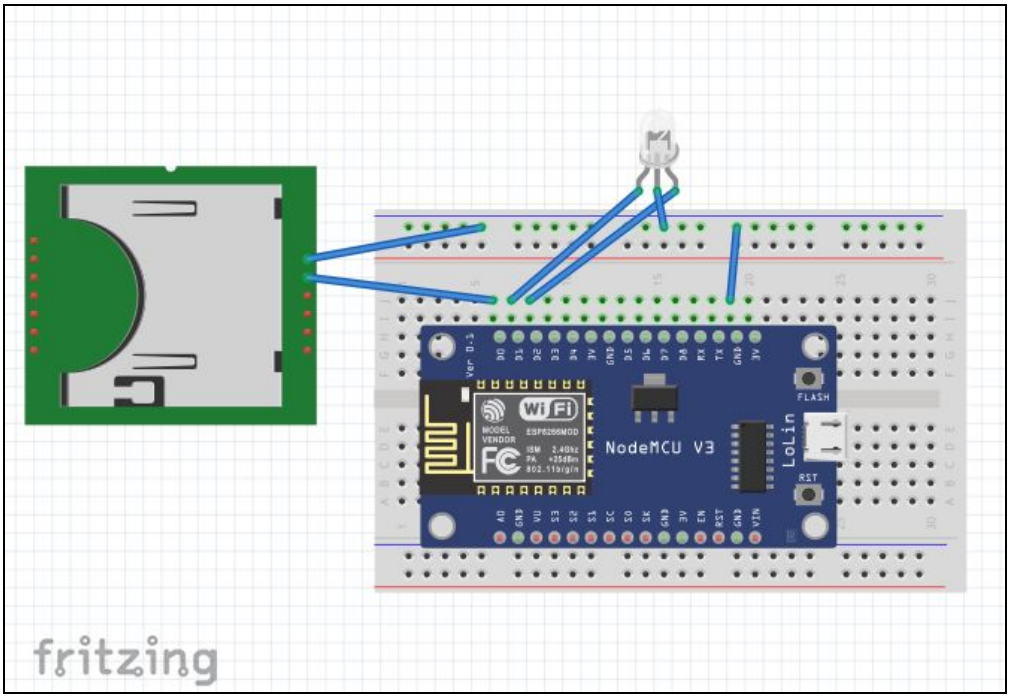


Figure 7 - Basic schematic of the SD card sensing method.

2) Procure a multimeter that has 'Continuity' mode as one of its functions, where it will beep when both ends are short-circuited. Keep an SD card and a card socket close by.

Figure 8 - Detecting current in our SD sockets.

3) Connect the probes of the multimeter to the two pins corresponding to the correct sensing method. For the sensing method to work, the multimeter must beep when there is an SD card in the slot, and it must not beep when there is no SD card in the slot (assuming the probes are constantly attached to their respective pins and are not touching each other).

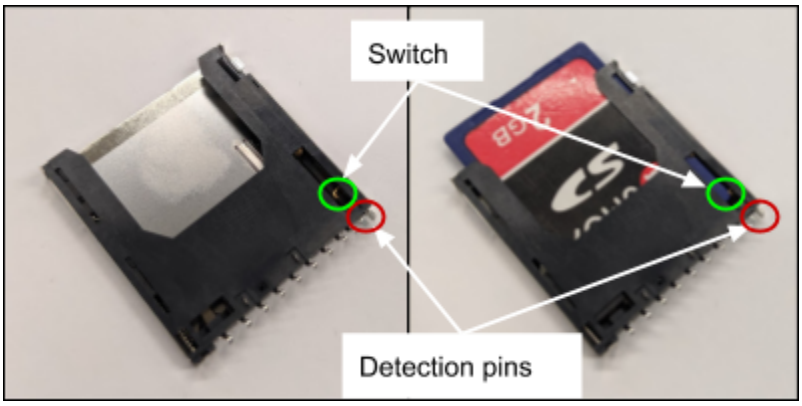


Figure 9 - Locating the card detection pins in our SD sockets.

The primary metric of this test is resistance, however it is a fairly binary test - in continuity mode, the multimeter either detects a ‘high’ resistance (meaning there is negligible current moving through the contacts) or a ‘low’ resistance (meaning almost all the current is moving through the contacts).

With our 10 purchased sockets, we are observing the multimeter and recording 10 data points which correspond to either PASS / FAIL for each socket. The cost of these 10 sockets is $45 (sourced from Digikey.ca). We also need $35 worth of supplies ($15 for an SD card and $20 for a multimeter (sourced from Amazon.ca)), but we borrowed these from the Makerspace. This test requires approximately 2 hours of time (1.5 hours to create diagrams and 0.5 hours to

conduct the multimeter test). Since we are using supplies from the Makerspace, we can conduct it anytime the Makerspace is open.

As we can see from Fig. GANTT, this test must be completed before any further planning can be done with our housing subsystem, our software subsystem, or our electronic subsystem. As such, it must be completed by the end of Week 1 of prototyping (by Oct. 31, 2019) so that Week 2 of prototyping (where we intend to perform functional tests of our developed algorithms, designs, etc.) can roll-out on schedule.

Test Results

After performing our test on our prototype, we determined that we have chosen a good sensing method for our device and we can proceed using the circuit from Fig. 7.

TABLE I

The card socket succeeds as a reliable sensor as long as we are using the pins on the mechanical switch and we are careful to avoid the defective one.

Sensing Method	**Card Socket Index**									
	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
Test Result	FAIL	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS

The next prototype will involve putting all wiring onto a prototype breadboard and creating an algorithm for this design on the NodeMCU. This will also involve integrating the multiplexers and creating an algorithm for multiple slots. This will likely be done with the help of online documentation of other people using multiplexers in Arduino projects[2][3]. We may also find that we need to add in a pull-up or pull-down resistor on the data pin[4].

Backend Software Subsystem

For the first prototype of the backend software, a proof of concept had to be established in order to prove that the ESP8266 NodeMCU could successfully pair and communicate with the Ross Dashboard software. First, a wired connection between the NodeMCU and a laptop was

[2] Example using our exact shift register - https://playground.arduino.cc/Code/ShiftRegSN74HC165N/

[3] Example using a larger multiplexer - https://www.instructables.com/id/MegaMUX-32-Channel-Multiplexer-Tutorial/

[4] Procedure for integrating a pull-up resistor - https://www.electronics-tutorials.ws/logic/pull-up-resistor.html

required in order to program it using the Arduino IDE to connect through WiFi. The initial setup was performed, including ensuring the laptop was correctly paired with the NodeMCU and Arduino IDE was uploading to the correct port. The next step was to connect the NodeMCU to a mobile hotspot broadcasting from the laptop on a network band of 2.4GHz. In order to verify the connection, the NodeMCU printed a message in the serial monitor (figure 2) so that we could tell if it was connected. Once the connection was established, the local IP address of the connection was used in the Ross Dashboard software to test sending a signal to the NodeMCU which would then send back a customized message which would also be printed in Ross Dashboard (figure 3). The future steps for this subsection is to see if information such as voltage readings from the NodeMCU can be passed to a laptop and what that information looks like and eventually transferring that information into the Dashboard UI.

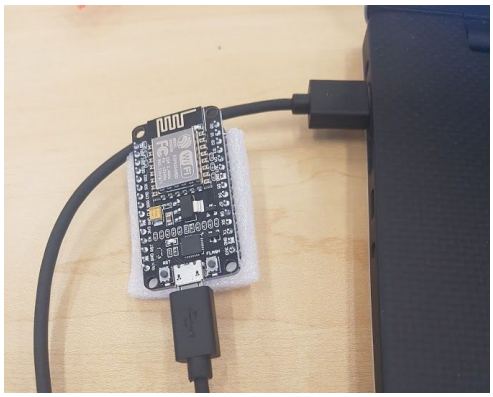


Figure 1. Wired connection between NodeMCU and laptop

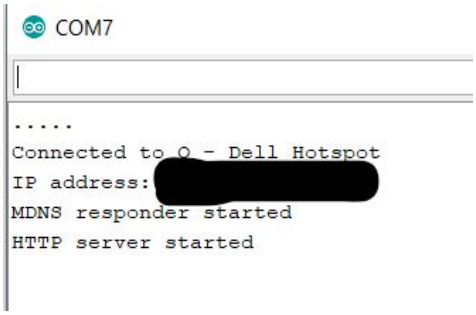


Figure 2. Serial monitor proving connection

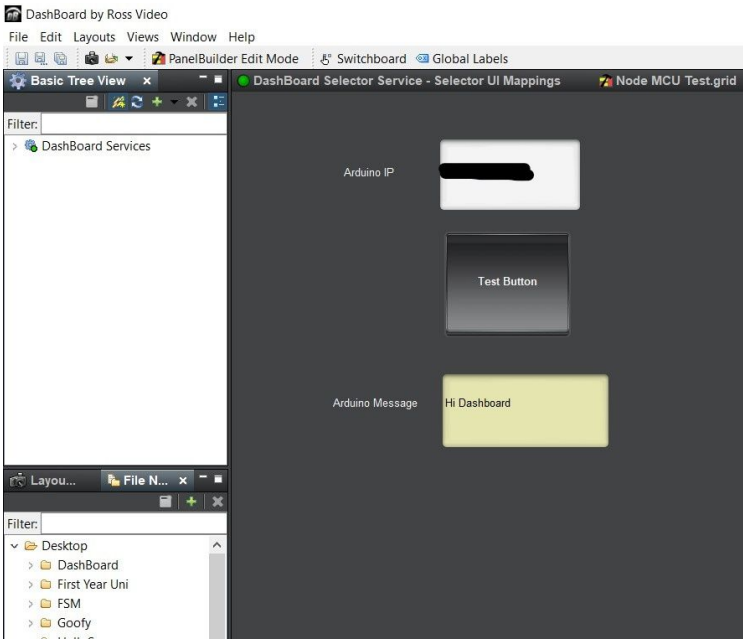


Figure 3. Dashboard Test UI

Find below the code used to carry out the above tests

```
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>
#include <ESP8266mDNS.h>

#ifndef STASSID
#define STASSID "Q - Dell Hotspot"
#define STAPSK  "nodemcutesting"
#endif

const char* ssid = STASSID;
const char* password = STAPSK;

ESP8266WebServer server(80);

void handleRoot() {
  server.send(200, "text/plain", "hello from esp8266!");
}

void listenAndRespond() {
  if(server.uri() == "/HiArduino"){
    server.send(200, "text/plain", "Subtle Flex");
  } else {
    server.send(200, "text/plain", "Wut");
  }
}

void setup(void) {

  Serial.begin(9600);
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  Serial.println("");

  // Wait for connection
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.print("Connected to ");
  Serial.println(ssid);
  Serial.print("IP address: ");
 Serial.println(WiFi.localIP());

  if (MDNS.begin("esp8266")) {
    Serial.println("MDNS responder started");
  }

  server.on("/", handleRoot);

  server.on("/inline", []() {
    server.send(200, "text/plain", "this works as well");
  });

  server.onNotFound(listenAndRespond);

  server.begin();
  Serial.println("HTTP server started");
}

void loop(void) {
  server.handleClient();
  MDNS.update();
}
```

Frontend UI Subsystem

Background Information

One of the requirements to having a successful project is developing a product that satisfies the sponsors for the project. Our sponsor for this project is Ross Video, and we're required to use their software as the basis to our design. Thus, integrating Ross Video's Dashboard is an integral component to our project. Prototype 1 UI involves the creation of many low fidelity skeleton designs that can provide alternative ideas and let the customer decide which design is preferred. This process will help shape the "High-visual-fidelity" design in prototype 2. Prototype 1 has a few different design ideas and organisational tools as well as different placements for buttons and visual cues. Prototype 1 was also done in Dashboard as a way to help familiarize with the system's capabilities and observe how certain ideas may appear in practice versus theory. It is also used to help gather feedback for each variation to help finalizing the design for the "High-visual-fidelity" prototype. Also, most of prototype 1's variations came about from the original design presented in fig #. First Official Design of UI.

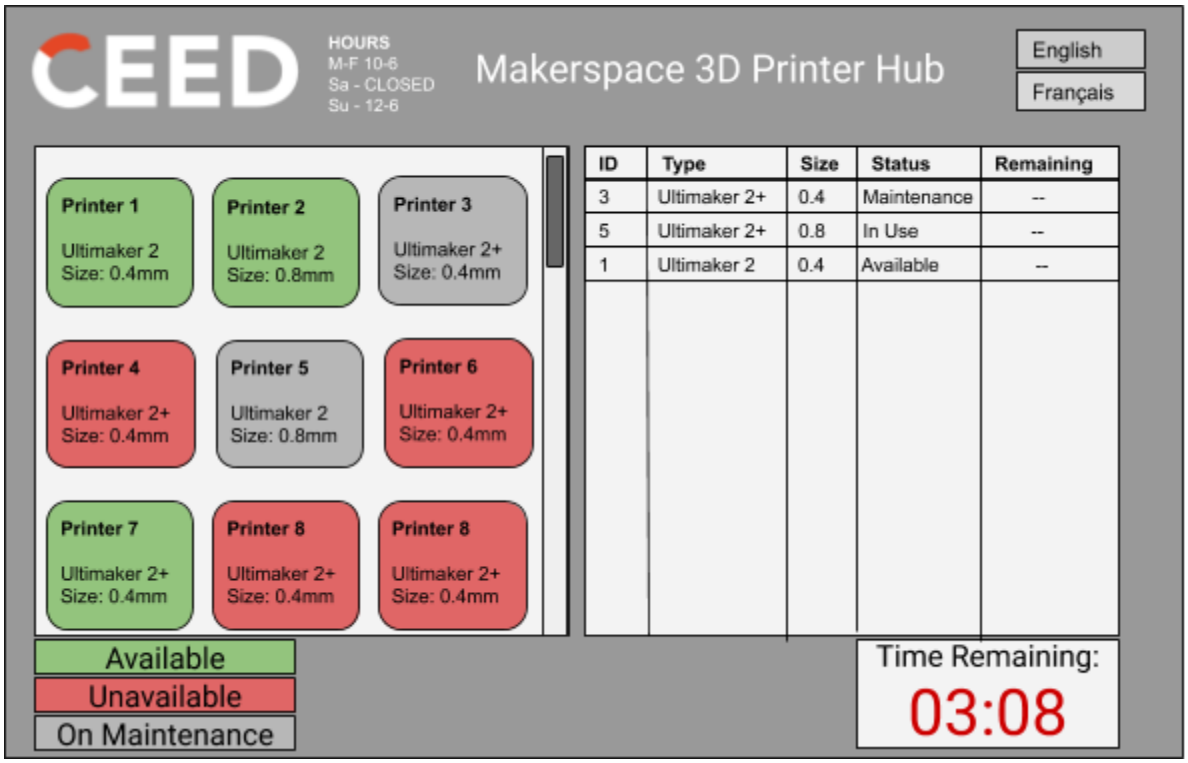


Fig 10 - First Official Design of UI.

The prototype 1 designs with varying levels of depth done in dashboard are as follows:

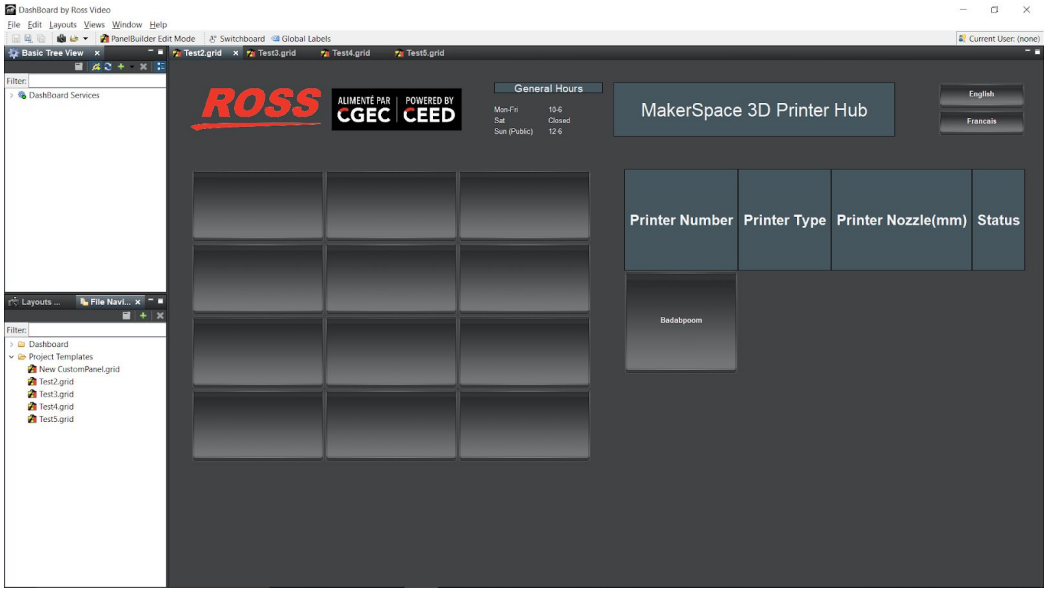


Fig 11: First iteration with default settings from Dashboard and original placements of icons based on the first design. Buttons automatically fill the table. Logos were also placed to observe how full the screen might look.

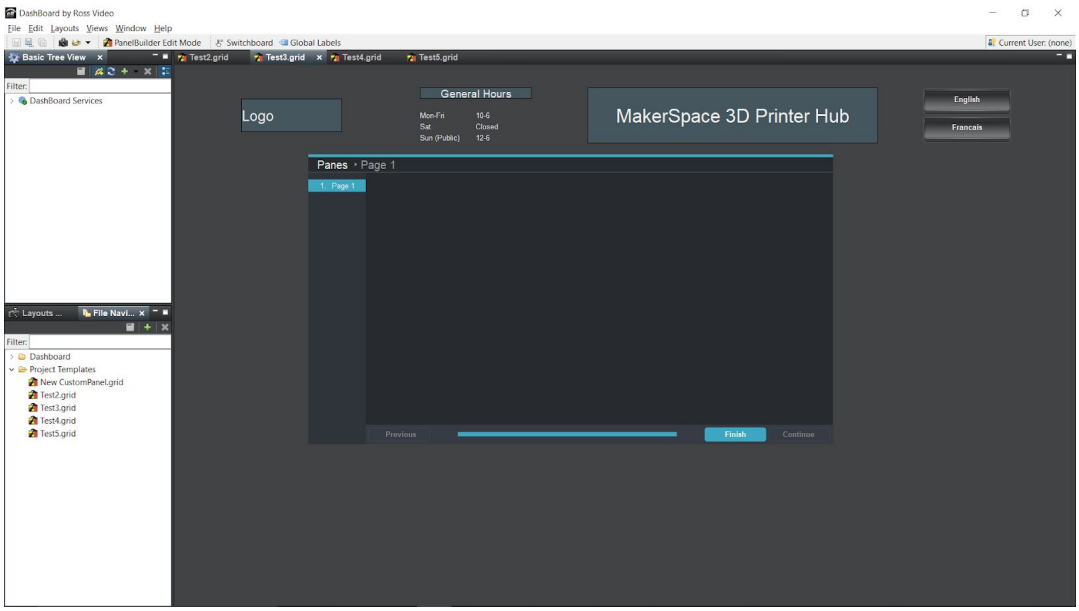


Fig 12: Second Iteration with a more barebones aesthetic and replacing the original design that had two panels with a single panel with a pages to allow the user to flip through the pages to find their desired information.

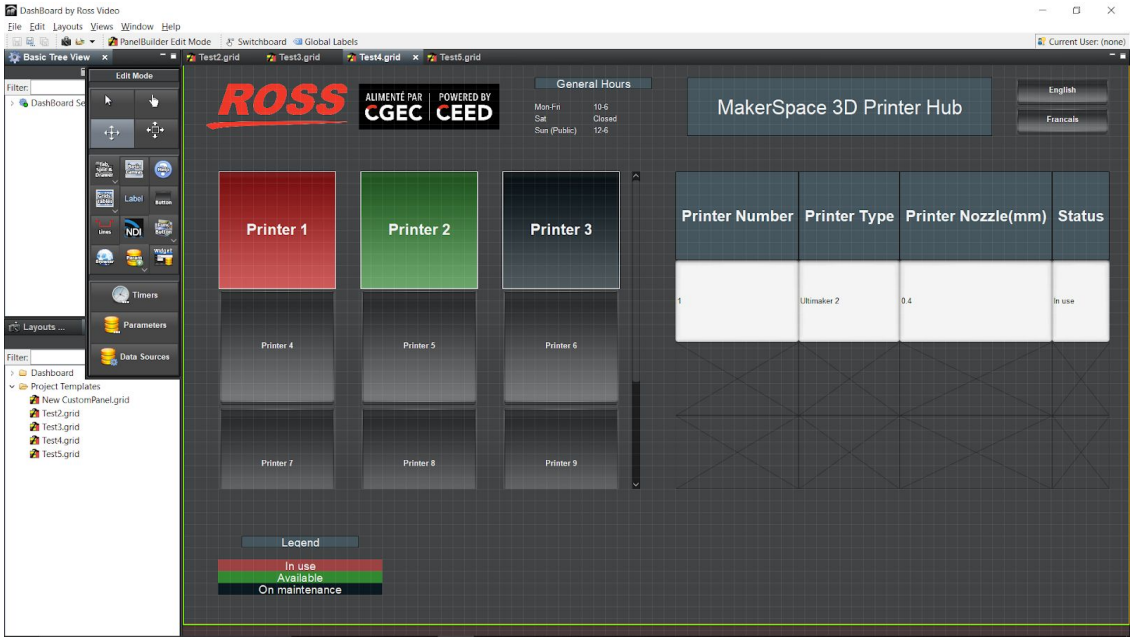


Fig 13: Third Iteration with the original design with the buttons manually spaced out and the additional implementation of the scroll wheel. Further inclusion of certain details like color legend and the beginning phase of the table design.

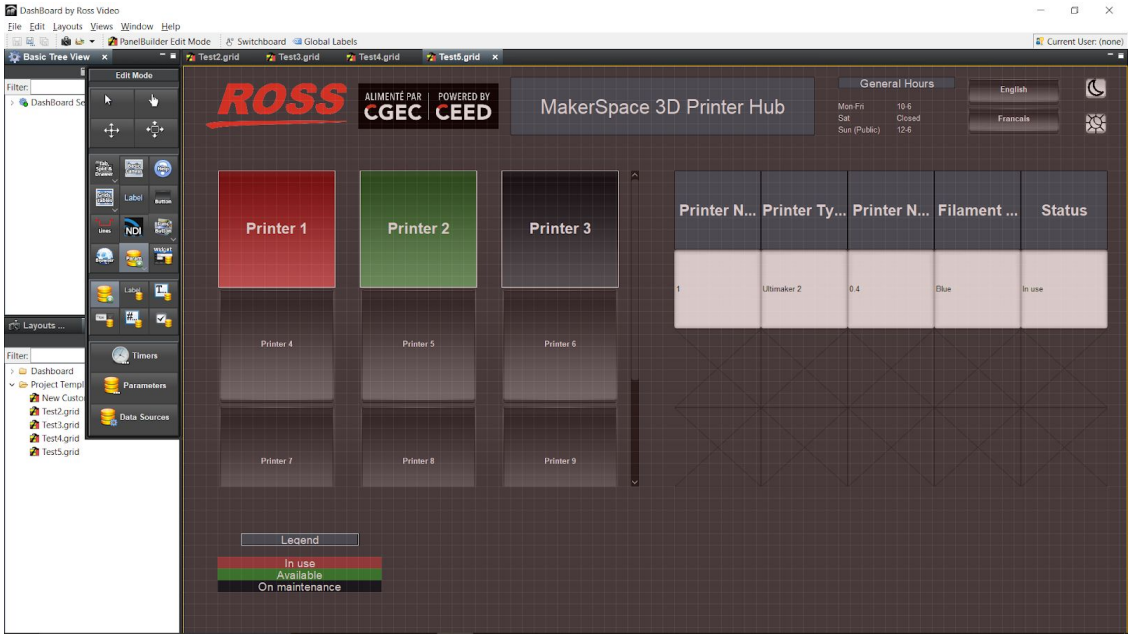


Fig 14: Fourth Iteration based on the third iteration design with the buttons manually spaced out and the additional implementation of the scroll wheel. Using customer feedback, the top bar was rearranged and new features may be implemented such as light vs dark mode. Furthermore, a new column was added with the filament color based on user requests.

Test Plan

This was the first feasibility test of our UI, so it was a stretch to figure out the capabilities of Dashboard and how to use it to its fullest given our current level of knowledge. This process involved the basic design of UI, deepening our understanding of Dashboard, and receiving feedback from users to better understand what information should be visible and how it should be presented. Using this data, we better understand what the user likes and needs. For example: the fourth iteration used user feedback to implement some of its additional features. In the case of test failure or test feedback, the only way for this test to fail is we are unable to implement the requested feature of if we don't receive feedback at all.

The procedure for this test is as follows:

Survey students in CEED Makerspace on their experience with 3D printers.

1. Begin by introducing myself and the project and start with open questions.
2. Show the Iterated Design Templates on Dashboard and ask the following questions:
    a. Which UI is the most visible (Ranked from 1-3)
    b. Which UI is the most clean and pretty to look at (Ranked from 1-3)
    c. Which UI is the easiest to understand (Ranked from 1-3)
    d. Scrollable vs not scrollable set-up?
    e. Any recommendations for things to change or add.

Table III
Test Results from UI

User	Proto -type	Most Visible	Most Clean	Easily Understood	Scrollable vs not scrollable	Recommen-dations
Bernadette 2nd year biomedical sciences student	1	2	1	2	Yes, scrollable is prefered	-Scrollable feature for all -Add filament color
	2	1	2	2		
	3	3	2	2		
	4	3	3	3		
Anat 2nd year Biochem student	1	1	2	1	Yes, scrollable is prefered	-Light mode vs Dark Mode -Modify Text Colours
	2	2	1	2		
	3	2	2	3		
	4	3	3	3		
Essraa A CEED employee	1	1	1	2	Yes, scrollable is prefered	-Add themes -Link Scroll features together.
	2	2	2	1		
	3	2	2	3		
	4	3	3	3		

UI results analysis

As the prototypes were iterated through and a with a few general designs created and tested through the general public. It seems that certain features are more likeable and more user-friendly than others. It seems that having a scrollable feature would be essential as well as the overall colouring and theme of the UI must be modified to fit the needs of its users. Having the table separated in clean and organized manner was also preferred. Taking these ideas and continuing to iterate the design over design #4 seems like the ideal plan to proceed.

Conclusion

We have had a very successful first round of prototyping. All of our proof-of-concepts have fit within our expectations and we are on schedule for our second round of prototyping, where we will be doing more work on functional capabilities.

Moving forward, we will need to do more collaborative work to bring our individual prototyping work together. This will likely involve increasing the amount of times we meet out of our regular lab sessions and our weekly team meetings as we combine the electrical and software subsystems, the electrical and hardware subsystems, and the two software subsystems.