GNG1103

Project Deliverable K

Design Project User Manual

Faculté de génie
Faculty of Engineering

Submitted by:

Team 12

Maya Salame, 300052824

Benoit Coote, 6832207

Kai-Chieh Chen, 300138078

Ugo Guitard Labelle, 0300049017

Saad Kamal, 300071981

December 7th, 2019

University of Ottawa

Table of Contents

List of Figures

List of Tables

List of Acronyms

Acronym	Definition
PIR	Passive Infrared
PLA	Polylactic Acid
CEED	Centre for Entrepreneurship & Engineering Design
STEM Building	“Science Technology Engineering Mathematics” Building
BOM	Bill of Material
IDE	Integrated Development Environment
PCB	Printed Circuit Board
USB	Universal Serial Bus

1.0 Introduction

The goal of our project is to develop a machine monitoring system that allows users of the MakerSpace to efficiently determine the availability of 3D printers. By integrating Ross Video DashBoard video to a PIR motion sensor, users will be able to determine whether a printer is in use, available, or out of order.

The objective of the final prototype of our project was to have a fully functional system that can be easily duplicated for all the 3D printers in the MakerSpace. For the final prototype, a real-time monitoring system that can be remotely displayed on DashBoard was the primary focus. By creating a host network for the signal and encoding the nodeMCU, we were able to establish a means of communication for the network. The integration of Ross Video DashBoard software and a hardware hard casing device allowed an enhanced user experience. An aesthetically pleasing and easy to interpret interface on DashBoard received signaling from the nodeMCU in order to accurately display the relevant information.

Over a three month time frame, we were able to develop a reliable and precise channel of communication from a PIR motion sensor to DashBoard and relevant printer information to be displayed remotely over a Wifi signal. The testing included starting and stopping prints to evaluate the response of our system and to create a realistic interference around the sensor in order to adjust the delay and sensitivity as well as minimise the noise and false positive signal.

Based off of the client meetings with CEED and the MakerSpace, relevant information and limitations were provided to our project that directed the stream of our 3D printer motion sensor. The project has to be implemented through the Ross Dashboard application. It has to be user-friendly and improve the experience of the CEED users and staff. The budget for this project is $100. The project must be implementable within the already limited available spaces in the CEED. The DashBoard software can control Arduino. The CEED users and staff can be contacted by the students to a certain extent and the CEED may be visited by the students. The Ross Video Dashboard University has a wide range of resources and tutorials on how to use the software and its capabilities.

Our team strived to obtain constructive and applicable feedback for our ideas, verify the general design's feasibility, and to analyze the critical systems and subsystems' integration; to ultimately minimize or eliminate any risk and uncertainty. Finally, a satisfactory stopping criteria for our iterative testing was considered, such that our defined metrics are all appropriately measured and an acceptable fidelity of our design is met.

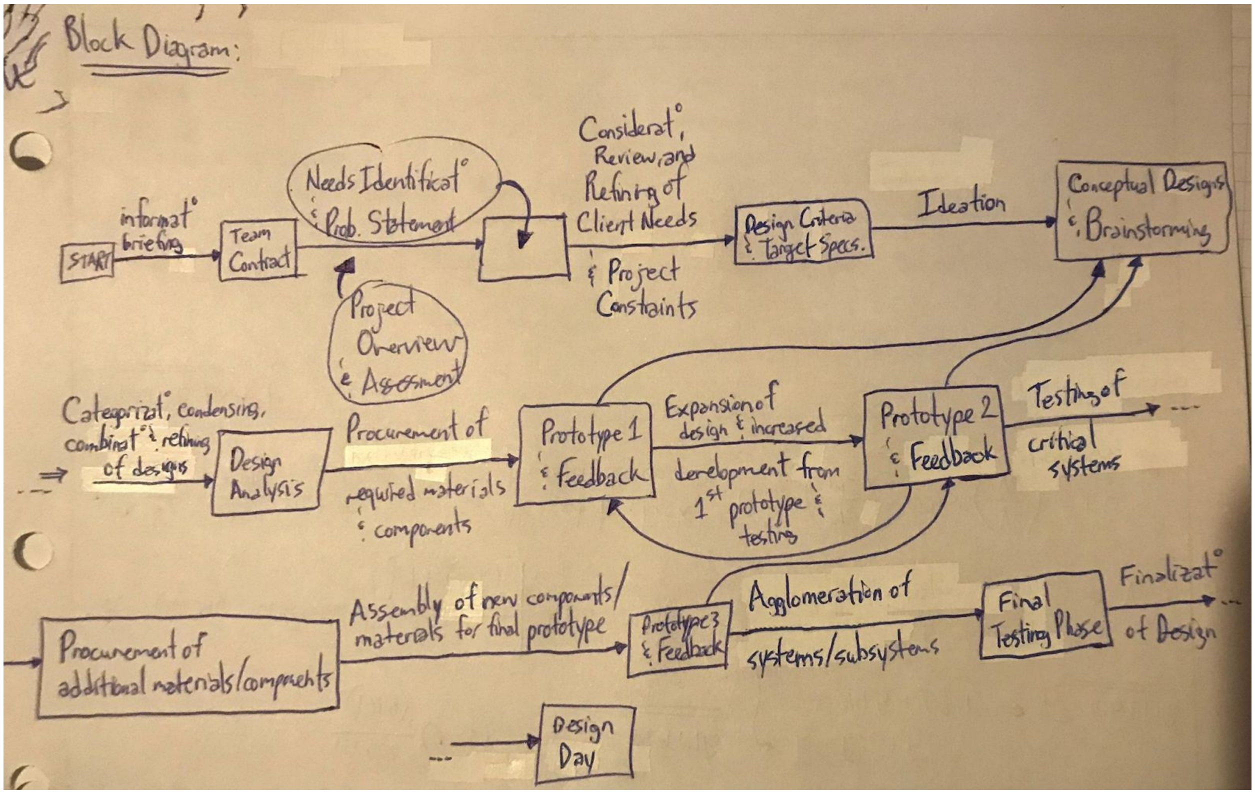

Block Diagram:
START
informat° briefing
Team Contract
Needs Identificat° & Prob. Statement
Project Overview & Assessment
Considerat°, Review, and Refining of Client Needs
& Project Constraints
Design Criteria & Target Specs.
Ideation
Conceptual Designs & Brainstorming
Categorizat°, condensing, combinat° & refining of designs
Design Analysis
Procurement of required materials & components
Prototype 1 & Feedback
Expansion of design & increased development from 1st prototype & testing
Prototype 2 & Feedback
Testing of critical systems
Procurement of additional materials/components
Assembly of new components/materials for final prototype
Prototype 3 & Feedback
Agglomeration of systems/subsystems
Final Testing Phase
Finalizat° of Design
Design Day

Figure X: Block Diagram of Our Process

1.1 Problem Identification

It was brought to our team's attention, by the CEED MakerSpace clients, that a solution is firmly desired for an efficient, innovative way to enhance machine monitoring of the equipment and tools provided in the CEED MakerSpace complexes. Our project design is based on the development of a user-friendly and remote way to view the availability of 3D printers in real time. This is to be enabled through the use of a PIR motion sensor that detects printer movement and then relays this message to its neighbouring, and attached, nodeMCU, which ultimately sends the message to the DashBoard panel, through a wireless hotspot network, for users and employees to consult for their personal and/or collective 3D printing needs. This line of communication between the PIR motion sensor, the nodeMCU, and the DashBoard panel allows for a seamless connection of the user to a single and straightforward custom panel, which displays all relevant printer information. By using a DashBoard viewing platform, we are able to display a user-friendly interface via a circuit system using the aforementioned nodeMCU device. By combining and integrating both the hardware and software aspects of our design, our group will be able to provide a solution to a common user problem. To establish the functionality of this design, our group utilized the method of iterative testing; for our focused and physical prototype. For this method, a prototyping test plan was generated that would allow for our team to organize each iterative step as well as to introduce a methodical solution process.

1.1.1 Importance

Having met with the client, at the beginning of the design project, in order to gather an understanding of the issues that they have been facing at the CEED MakerSpace complex in the STEM building as well as their needs, which they believe to be potential alternative solutions, our team was able to develop an idea of the importance of this prototype design. While this is true, our team continued to appreciate and understand the overall worth of such a design, which is ultimately geared towards providing an ease of access of the MakerSpace for both its users as well as the employees.

For the CEED MakerSpace employees, this innovative design will prove to be beneficial since it will provide them with an organizational tool for the MakerSpace, such that the availability, real-time occupancy, and/or downtime information of the 3D printers shall be readily available. With this valuable information literally at their fingertips, they will now be able to efficiently direct users within the MakerSpace as well as to determine the total usage of the 3D printers within this space.

1.2 Fundamental User Needs

Our team has been in unanimous agreement that users of the MakerSpace complex, these being, for the most part, students, are extremely engaged and active individuals whose time is invaluable and should therefore not be needlessly wasted. This being said, over the course of the design project, our team has continuously understood and related to the dire need of such a product. Essentially, from the point of view of the users, they are currently faced with the perplexing predicament of deciding whether or not they should commit to a trip onto campus to carry out a 3D print. This is problematic since they do not currently have a way from which they can remotely determine the availability of the MakerSpace's 3D printers; from their own individual spaces. However, by implementing our '*3D printer machine monitoring system*', the users will thereafter have a platform from which they can observe the availability and running time of these machines in real-time; thus allowing them to make an informed decision regarding the allocation of their time.

1.3 Product Differentiation

Our product is different in the way that the users can only look at the display while CEED employees can control it. This reduces user errors and CEED employees can change the display status manually when needed. It is also cheap ($3/sensor), easy to move between machines, and easy to install. As there are many printers in MakerSpace, the low price and mobility of the hardwares are essential.

The system is maintenance free, easy to install and duplicate, it requires no input from the user and the only manipulation required by the staff is to manually input the out of order command in case of printer malfunction. It requires no user training, and does not rely on the user for any inputs, therefore limiting the human error factor.

The low cost of the hardware and that all the enclosure can be 3D printed, make it very cost effective for the client. The sensor chosen for the device allow a good range of adjustments that can be fined tune to give a very reliable status of the system.

1.3.1 Similar Products

We had looked at a few available products for solution.

1.3.1.1 Eve Wireless Motion Detector

Home monitoring device that can integrate to Apple Home kit and has customizable sensitivity settings.

1.3.1.2 ESR PIR Sensor

Installation sensor specially designed for indoor components such as lights. It also has adjustable sensitivity and time step, and automatic day/night function.

1.3.1.3 Octoprint

Management software that can control and monitor 3D printers via internet connection.

1.3.2 Our Product

Out product uses a small and simple PIR sensor to monitor the status of a 3D printer.

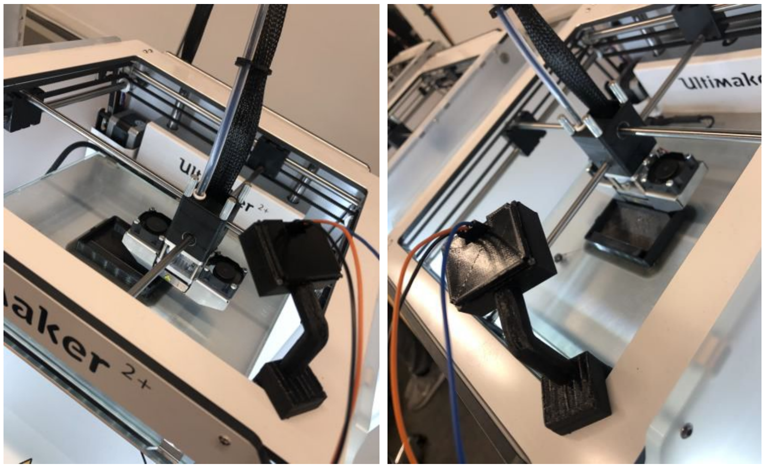


Figure X: Angled Views of Mounted PIR Motion Sensor on the 3D Printer

1.4 Main Function

The main *purpose* of our team's final prototype is essentially to provide users of the CEED MakerSpace complex with a real-time monitoring system that would displayed on a user-friendly Ross Video DashBoard display panel, from which they can access remotely. The users, as well as the CEED MakerSpace employees will undoubtedly benefit from this product since its main *function* is to detect the availability of 3D printers by means of a PIR motion sensor that is mounted on the outer edge of the printer (sensor's line of sight is oriented inwards) and connected to a nodeMCU device. Once the sensor detects motion within the 3D printer, it will relay a response to the nodeMCU and subsequently, through a Wifi hotspot network, will relay the message to a configured DashBoard display panel. It is from this DashBoard panel that users will be able to determine how many 3D printers are *in use*, *available*, or *out of order*. Ultimately, depending on the information displayed on the DashBoard panel, users will be in a better position to make an informed decision on whether they should commit to a trip on campus to carry out their prints.

2.0 How the Prototype is Made

Throughout this section, our team will explain in detail how the prototype was built, which shall include, where appropriate, design considerations as well as relevant calculations. To facilitate the understanding of how our prototype that was built and its various components, this section has been categorized according to the three principal aspects of our design, these being mechanical, electrical, and software. Additionally, for ease of implementation and comprehension for potential future designers, these three main categories have each been further subdivided in terms of their respective list of equipment, and finally thorough instructions on how it has been assembled.

2.1 Bill of Materials (BOM)

A detailed list of all the parts and materials that were required for this design project have been compiled and tabulated in **Table.X** [below]. As can be seen, the detailed list comprises the name of the materials utilized, the cost of each given material as per its respective unit price, the total number of units that were needed for the completion of our prototype designs, and the total cost associated with our design; which is based on the summation of the total costs of each type of material.

Table.X: Detailed tabulation of our team's prototype design's Bill of Materials (BOM).

Material	Cost Per Unit ($CAD)	No. of Units	Total Cost ($CAD)
PIR Motion Sensor	~ 3	3	~ 9
Arduino UNO Card	15 ~ 35	2	~ 50
3D Printing PLA Filament	0	N/A	0
Female-to-Female Wires	N/A	6	N/A
Fastening Hardware	~ 5	1	~ 5
nodeMCU device	N/A	1	N/A
PIR Motion Sensor Mount	N/A	1	N/A
nodeMCU Encasement Mount	N/A	1	N/A
Electrical Tape	N/A	1	N/A
Micro USB Cable	N/A	1	N/A

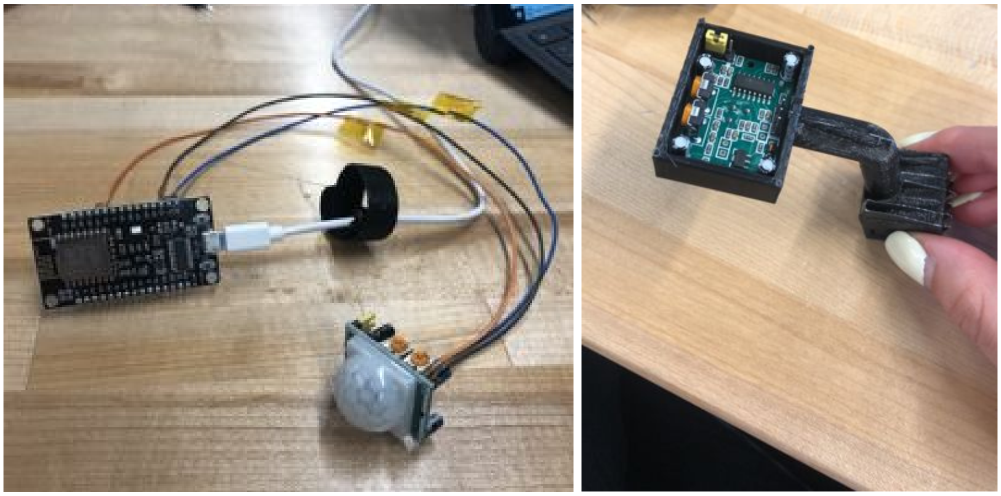


Figure X: Prototype I

2.2 Mechanical

Particularly, in our team's case, a relatively small portion of our prototype's design was attributed to be mechanically oriented. This design project was inherently a more electrical and software challenge, however certain areas of the design and subsequently their realization and implementation proved to be crucial seeing as, in this area, a number of components fabricated were in fact essential in order to meet certain constraints and metrics, which were established in the early stages of the design. Furthermore, these constructed mechanical components also led to an aesthetically pleasing final prototype; this also being an exceedingly valued aspect of the prototype in general.

2.2.1 Equipment List

The equipment and components that were required for the mechanical portion of our prototype's design are a re-statement of those listed in the BOMs, however they are listed below for a future designer's ease of understanding. The equipment and components utilized included:

- nodeMCU device,
- Female-to-female wires,
- Micro USB cable,

- 3D-printed mount for PIR motion sensor,
- 3D-printed nodeMCU encasement mount,
- PIR motion sensor,
- Two (2) functional laptops (one for network hotspot & one for Arduino & DashBoard coding/integration aspects), and
- Electrical tape.

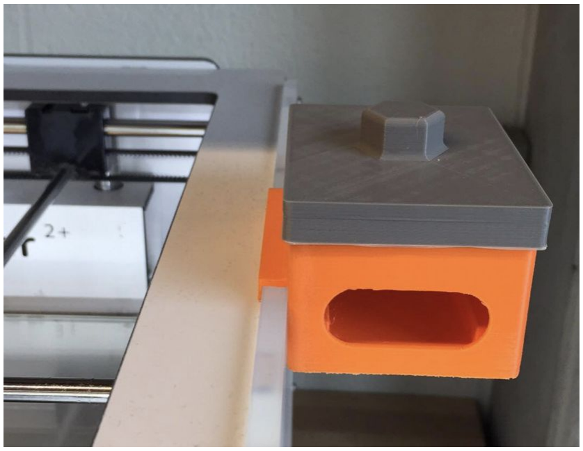
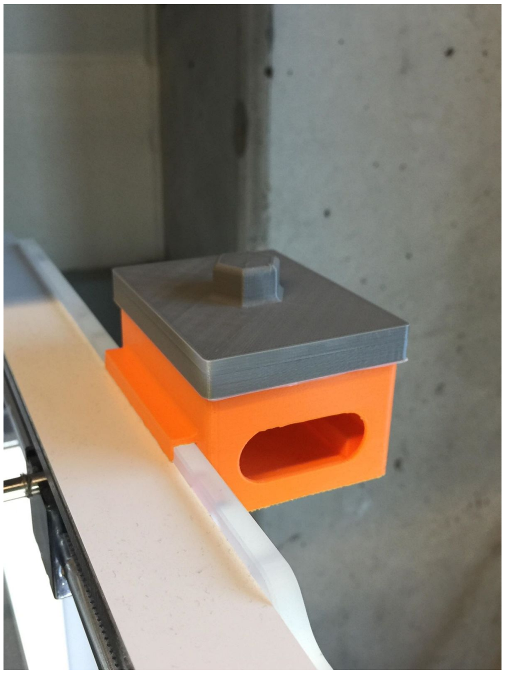


Figure.Y: nodeMCU 3D printed encasement mount.

2.2.3 Instructions

Pertaining to the construction of the mechanical/physical portion of our prototype, our team wanted to, most prominently, ensure that the motion sensor along with its encasement, once mounted on the 3D printer, would effectively and consistently relay a reliable signal to the nodeMCU, therefore the final positioning as well as mounting angle were strongly considered. Similar considerations were carried over when designing the encasement box for the nodeMCU device; the latter would also be securely but comfortably fitted to one of the outer edges of the 3D printer, which can be seen in **Figure.XX** and **Figure.XX2**.



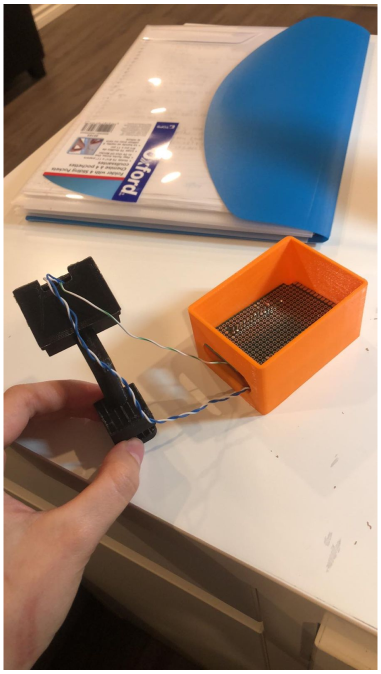


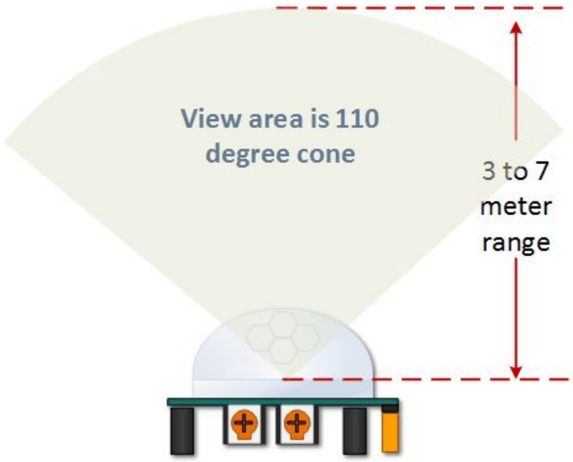
Figure.XX: nodeMCU 3D printed encasement mount.

Figure.XX2: nodeMCU 3D printed encasement mount without the top cover; to exhibit the secure and snug fit of the device within the mounting encasement.

To begin with, it should be noted that materials such as the nodeMCU, female-to-female wires, PIR motion sensor, USB connector, laptops, and the electrical tape were not built by our team but acquired through online purchases and already readily available from one of our team members.

Next, our team developed a design of the PIR motion sensor mount through the use of the SolidWorks software and in doing so would persevere to accommodate multiple design metrics such as the budget, weight, constructability, and size. These metrics were met in such a way that the 3D printers could be utilized free of charge, the final 3D printed mount was extremely lightweight, it can be effortlessly implemented as well as maintained (easy to remove and reattach), and is significantly small in size, respectively.

Once the SolidWorks design of the PIR motion sensor mount was of satisfactory quality, our team proceeded to transfer it to an SD card, which would later be inserted into one of the available 3D printers in the CEED MakerSpace complex for its construction. During the iterative testing process of the functionality and continuity of the sensor’s response, as mentioned earlier we were forced to re-evaluate certain aspects of the design such as the dimensions of the ‘slot’ that would clamp onto the 3D printer’s ledge, the angle in which the sensor would be directed into the 3D printer (focus angle), in addition to the sensor’s optimal viewing angle. In the end, we determined a suitable clamp length such that the mount could be securely mounted onto the 3D printer, but in such a way that it would not diminish the ease in which it could be removed for maintenance purposes. We also found that the optimal *viewing angle* was that of ~ 15-30 degrees and the optimal *focus angle* was approximately 80 degrees from the vertical, the former was not an original setting provided with the PIR motion sensor and therefore required a manual adjustment. The *viewing angle* is represented in **Figure.123** [below] and the *focus angle* may be more or less visualized from **Figure.789** [below].



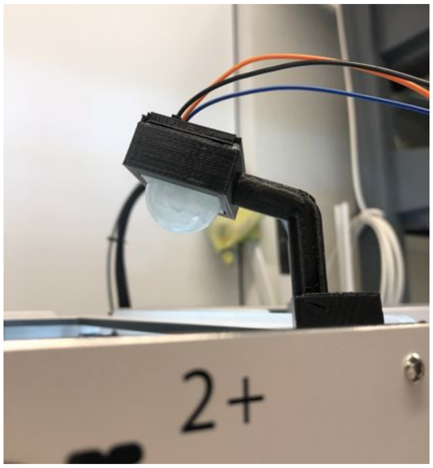


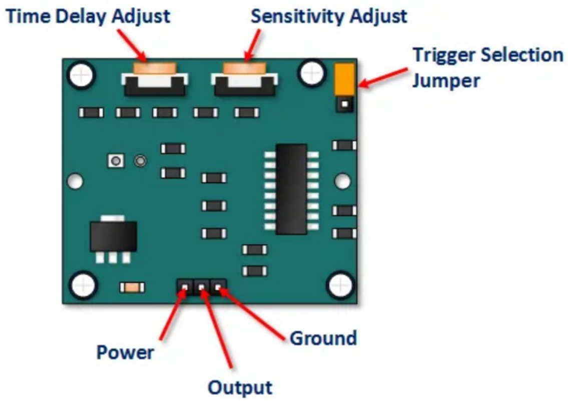
Figure.123 (left) and **Figure.789** (right), respectively.

Regarding the nodeMCU encasement mount, similar considerations were thought out to ensure that it could be easily but securely mounted onto the 3D printer ledge, which can be seen in **Figure.Y** [above]. Similarly, multiple testing iterations and 3D prints were carried out in order to achieve the desired mounting style.

The biggest constraint for the design was to securely attach the sensor without impeding the printer’s belt. After a few re-calibration, the team was able to securely attach the mount with a *2 mm tolerance* for the moving parts under the mount. Due to the somewhat odd shape of the mount, the sensor had to be printed at the smallest available filament size to allow for sufficient support for the part, without damaging it. The sensor was pointed towards the back of the printer to minimise the interference with the foot traffic in front of the printer.

2.3 Electrical

The electrical connections for the prototype are fairly simple. The sensor use the 3V and ground connection from the nodeMCU to the sensor. The output of the sensor was connected to the pin 13 as per our code (correspond to pin D4 on the node MCU). The node MCU has 3 power supply that could be used (1 x 5V, 2x 3V) to power sensors. The sensor can accept all three of them. Also, by a fairly simple modification of the code, more sensors could be fitted on the same node MCU.

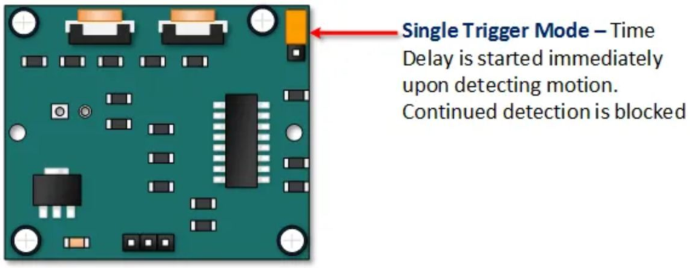


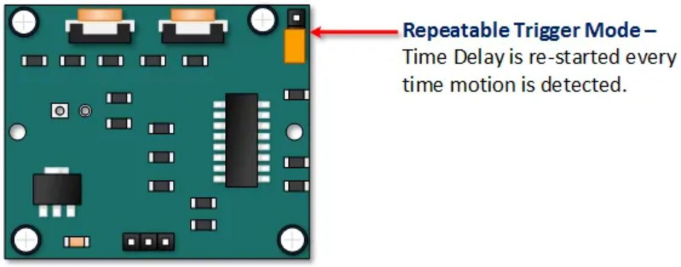
2.3.1 Equipment List

- HC-SR501 PIR Motion Sensor
- NodeMCU
- 3-D printer
- Small gauge copper wire
- Soldering iron and solder
- PCBs

2.3.2 Instructions

The sensor’s default position is the single trigger mode, in order to get a usable signal that will not interfere with the timer, the jumper should be moved to a repeatable trigger mode.


Single Trigger Mode – Time Delay is started immediately upon detecting motion. Continued detection is blocked


Repeatable Trigger Mode – Time Delay is re-started every time motion is detected.

Finally, when installing the sensor for the first test, the delay should be adjusted to a minimum value and the sensitivity should be set to the maximum. Once tests have been conducted the delay can be increased and the sensitivity decreased as required. Once the connections are made, the nodeMCU is to be connected to a PC and through the Arduino IDE the following code is to be compiled and uploaded:

```
/*
        This sketch sends a string to a TCP server, and prints a one-line response.
        You must run a TCP server in your local network.
        For example, on Linux you can use the following command: nc -v -l 3000
*/
#include <ESP8266WiFi.h>
#include <ESP8266WiFiMulti.h>
#ifndef STASSID
#define STASSID "your network name"          //Insert your network name in between brackets
#define STAPSK  "your password"                      //Insert your password between brackets
```

```
#endif
const char* ssid          = STASSID;
const char* password      = STAPSK;
const char* host = "123.123.123.1";                    //Enter the IP address of the host
const uint16_t port = 8888;
ESP8266WiFiMulti WiFiMulti;
void setup() {
  Serial.begin(115200);
                                                       //We start by connecting to a WiFi network
  WiFi.mode(WIFI_STA);
  WiFiMulti.addAP(ssid, password);
  Serial.println();
  Serial.println();
  Serial.print("Wait for WiFi... ");
  while (WiFiMulti.run() != WL_CONNECTED) {
        Serial.print(".");
        delay(500);
  }
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
  delay(500);
}
void loop() {
  Serial.print("connecting to ");
  Serial.print(host);
  Serial.print(':');
  Serial.println(port);
                                                 // Use WiFiClient class to create TCP connections
  WiFiClient client;
  if (!client.connect(host, port)) {
        Serial.println("connection failed");
```

```
            Serial.println("wait 5 sec...");
            delay(5000);
            return;
  }
                                                                    //This will send a request to the server
  delay(1000);
  int inputPin = 2;
  int pirState = LOW;
  int val = 0;
  pinMode(inputPin, INPUT);
 val = digitalRead(inputPin);
  {
            if(val == HIGH){
            if(pirState == LOW)
            {
            client.println("1"); //Status on
            pirState = HIGH;
            }
            } else {
            {
            client.println("0"); //Status off
            pirState == LOW;
            }
  }
  }
//read back one line from server
  Serial.println("receiving from remote server");
  String line = client.readStringUntil('\r');
  Serial.println(line);
  Serial.println("closing connection");
  client.stop();
  Serial.println("wait 5 sec...");
  delay(5000);                            }
```

This code is designed to send the input on pin 13 to the preset listener port on DashBoard. The user needs only to replace the *network name* "your network name"; *password* "your password"; and *IP address*, in between quotation marks. In this code, just as in DashBoard, the *port 8888* has been selected for the listener port but could be modified to suit the users requirements or preference.

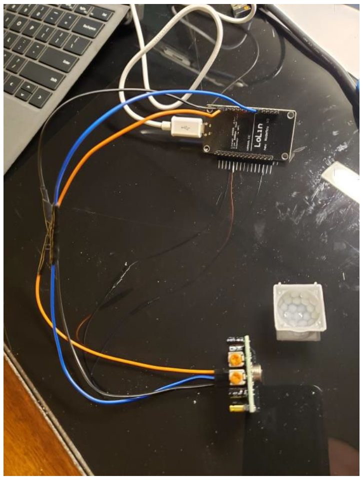
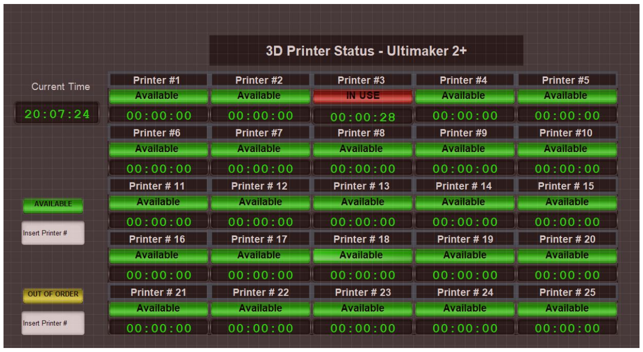


Figure.YY: Basic electrical configuration of prototype device.

2.4 Software

Throughout the iterative processes of the various prototype stages, inherent to a design project such as this, our team quickly came to realize that the software development and integration was undoubtedly paramount to its end-stage full functionality. With this in mind, we collectively collaborated to challenge this design feature in order to guarantee the functionality of our design's most critical system.


3D Printer Status - Ultimaker 2+

Current Time

20:07:24

Printer #1 Available 00:00:00	Printer #2 Available 00:00:00	Printer #3 IN USE 00:00:28	Printer #4 Available 00:00:00	Printer #5 Available 00:00:00
Printer #6 Available 00:00:00	Printer #7 Available 00:00:00	Printer #8 Available 00:00:00	Printer #9 Available 00:00:00	Printer #10 Available 00:00:00
Printer # 11 Available 00:00:00	Printer # 12 Available 00:00:00	Printer # 13 Available 00:00:00	Printer # 14 Available 00:00:00	Printer # 15 Available 00:00:00
Printer # 16 Available 00:00:00	Printer # 17 Available 00:00:00	Printer # 18 Available 00:00:00	Printer # 19 Available 00:00:00	Printer # 20 Available 00:00:00
Printer # 21 Available 00:00:00	Printer # 22 Available 00:00:00	Printer # 23 Available 00:00:00	Printer # 24 Available 00:00:00	Printer # 25 Available 00:00:00

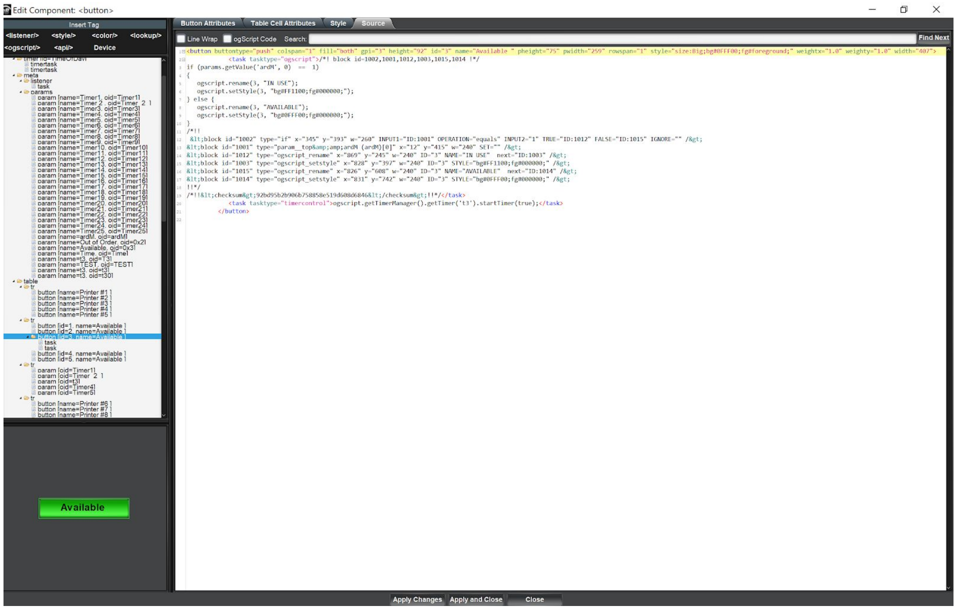
Figure.XY: Sample operational dashboard panel with implemented timer functions.

2.4.1 Equipment List

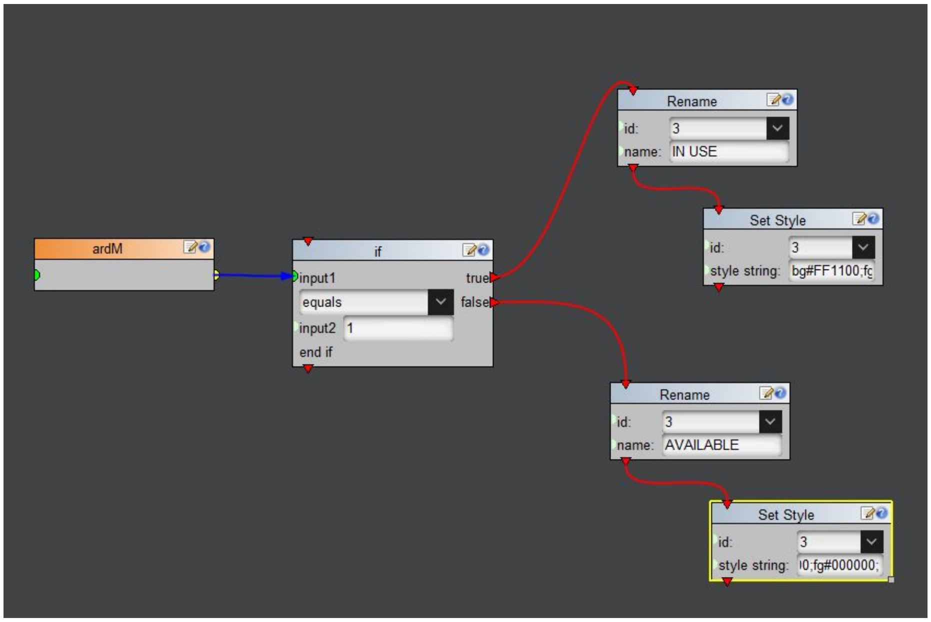
The equipment list for this particular section is non-existent seeing as it is the *software* portion of our team's prototype design.

2.4.2 Instructions

The software aspect of this project is divided into two major components: *Arduino* and *DashBoard*. Starting with the DashBoard component of the project, several key components must be established. Among these, the *main three* include: setting up a listener server, creating parameters to hold values sent to the server, and creating tasks that implement these values. To create a basic listener server on DashBoard, it is advised to follow the "*Tutorial 2 - Listen for Sensor Value*" (Knox and Hunter 2019). Once this tutorial is completed, you will have successfully set up a listener server and created a single parameter to hold a value from that server. With these basic components, you can create a new task on a button, which will implement the use of the parameter to change the button's text as well as colour; using block programming.



```
<button buttontype="push" colspan="1" fill="both" gpi="3" height="92" id="3" name="Available " pheight="75" pwidth="259" rowspan="1" style="size:Big;bg#0FFF00;fg#foreground;" weightx="1.0" weighty="1.0" width="407">
            <task tasktype="ogscript">/*! block id=1002,1001,1012,1003,1015,1014 !*/
if (params.getValue('ardM', 0)  ==  1)
{
   ogscript.rename(3, "IN USE");
   ogscript.setStyle(3, "bg#FF1100;fg#000000;");
} else {
   ogscript.rename(3, "AVAILABLE");
   ogscript.setStyle(3, "bg#0FFF00;fg#000000;");
}
/*!!
 &lt;block id="1002" type="if" x="345" y="393" w="260" INPUT1="ID:1001" OPERATION="equals" INPUT2="1" TRUE="ID:1012" FALSE="ID:1015" IGNORE="" /&gt;
&lt;block id="1001" type="param__top&amp;amp;ardM (ardM)[0]" x="12" y="415" w="240" SET="" /&gt;
&lt;block id="1012" type="ogscript_rename" x="869" y="245" w="240" ID="3" NAME="IN USE"  next="ID:1003" /&gt;
&lt;block id="1003" type="ogscript_setstyle" x="828" y="397" w="240" ID="3" STYLE="bg#FF1100;fg#000000;" /&gt;
&lt;block id="1015" type="ogscript_rename" x="826" y="608" w="240" ID="3" NAME="AVAILABLE"  next="ID:1014" /&gt;
&lt;block id="1014" type="ogscript_setstyle" x="831" y="742" w="240" ID="3" STYLE="bg#0FFF00;fg#000000;" /&gt;
!!*/
/*!!&lt;checksum&gt;92bd95b2b906b758858e519d608d6846&lt;/checksum&gt;!!*/</task>
            <task tasktype="timercontrol">ogscript.getTimerManager().getTimer('t3').startTimer(true);</task>
         </button>
```



if: input1 equals input2 1, end if

Rename — id: 3, name: IN USE

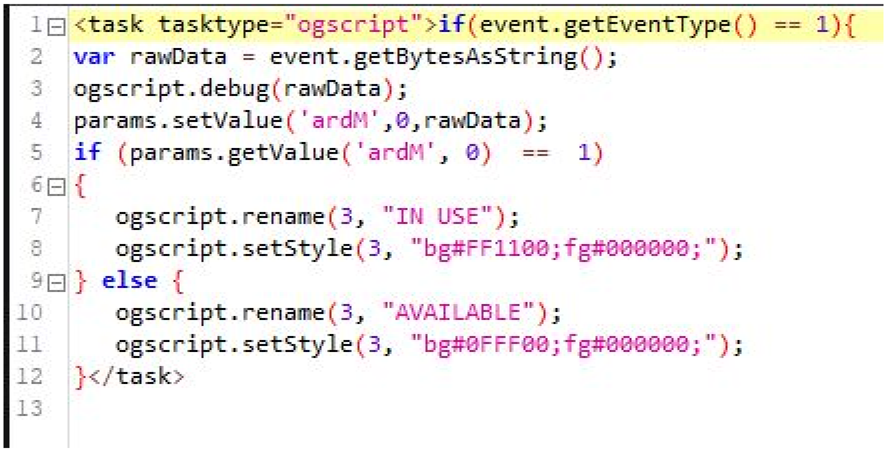
Set Style — id: 3, style string: bg#FF1100;fc

Rename — id: 3, name: AVAILABLE

Set Style — id: 3, style string: 0;fg#000000;

Figure.XY: OG script and block coding for button task modification

The next step is to automate this process so that the button does not need to be pressed physically to update its value. To do this we have taken advantage of the constantly updating listener server to modify values instead of using a button press. Each button can be given a unique ID, which is used to reference itself outside of the code. By implementing this button ID with the corresponding code that checks our incoming parameter, we have successfully created a way that automates the button style, text or any other attribute at the same rate as the parameter is being updated. An example code:



```
<task tasktype="ogscript">if(event.getEventType() == 1){
var rawData = event.getBytesAsString();
ogscript.debug(rawData);
params.setValue('ardM',0,rawData);
if (params.getValue('ardM', 0)  ==  1)
{
   ogscript.rename(3, "IN USE");
   ogscript.setStyle(3, "bg#FF1100;fg#000000;");
} else {
   ogscript.rename(3, "AVAILABLE");
   ogscript.setStyle(3, "bg#0FFF00;fg#000000;");
}</task>
```

With these core steps, the panel can be extended as needed. Other details such as creating timer parameters and linking them to specific buttons can be done with the same type of code and can be used to sync buttons or even to start any other event.

3.0 How to Use the Prototype

In this section, our team will attempt to guide potential future designers on how to use our device by means of introducing the ways in which our prototype proved to best suit our own personal client needs and specifications. Moreover, a systematic breakdown of the prototype's usage will be presented in the upcoming sub-sections; once again, this has been methodically structured for the benefit of the reader. This section will elaborate on the prototype's functions and how it works; guidelines to ensure it is used and operated in a safe manner; and the steps required to effectively install both the physical and electrical/software components of the prototype.

3.1 Prototype Functions

Essentially, the primary function of this prototype is to capture movement of the 3D printer's articulating arms through the use of a PIR motion sensor that will then relay the response to the neighbouring and attached nodeMCU device, which will send a message to DashBoard via a wireless hotspot network; ultimately, providing users and employees of the CEED MakerSpace complex with real-time information regarding the total availability or usage of the 3D printers.

Additionally, the DashBoard panel has been configured in such a way that not only can users gain insight as to the availability/unavailability of the 3D printers but it can also, if one or several are having problems and are entirely unavailable for maintenance purposes, be represented as “out of order” on the DashBoard display panel.

3.1.1 How It Works

Our team's final prototype was designed in such a way that all of the various aspects and components could be seamlessly integrated into one fully functioning device; whether they be mechanical, electrical, or software oriented. This was effectively accomplished by means of adjusting the PIR motion sensor's range and sensitivity according to the client's specific design needs; properly wiring and soldering the required electrical components to the nodeMCU device; generating suitable and descriptive coding blocks within the Arduino integrated development environment (IDE); establish a functional, aesthetically pleasing, and user-friendly DashBoard display panel(s); and setting up a reliable and continuous line of communication between the nodeMCU device and DashBoard via a wireless hotspot network.

As it has been briefly mentioned in previous sections, the functionality of the prototype as a whole relies on the interdependent integration and communication between all of these components and design aspects; which shall now be elaborated upon. To begin with, having set the sensor settings to be in a repeated trigger mode, adjusted the PIR motion sensor's sensitivity dials for a range of 3 - 7 meters, and reduced the time delay to approximately 3 seconds, the sensor is now ready to be placed into its 3D printed stand and subsequently mounted onto the inner top right edge of the 3D printer. In doing so, the PIR motion sensor is now accustomed to meet our design needs in the most optimal manner. From this, once the PIR motion sensor detects movement within the printer, it will send a "motion detected" response to the connected, and also mounted, nodeMCU device. Next, the message will be sent to the custom-built DashBoard display panel via a personalized wireless hotspot network. Ultimately, it is this detection of *motion detected* (returns 1) or *motion ended* (returns 0) from the PIR motion sensor that will adjust the displayed message on the DashBoard panel, for users and employees to see, respective to a particular 3D printer.

It should be prominently noted that, due to the difficulties faced when attempting to establish a hotspot network to DashBoard and the fact that this connection is undoubtedly paramount to the functionality of our design, our team took the initiative of meeting with our course professor, Dr. David Knox. Significant realizations and advancements were made during the meeting, this being the proper setup of a line of communication from our hotspot to DashBoard; essentially the debugging of the communication problem with our nodeMCU. It was found, during this meeting, that our 2.4GHz wifi ad hoc network was properly set up, all devices were connected on the same network (verified by utilizing "*IPconfig*" and "*ping*" to determine the IP addresses of our devices), and that both ends of the link between our Arduino sketch and the DashBoard block information were using the same port number. However, the two devices persisted to not talk to each other. We discovered that the Windows software firewall on the PC (Windows Defender) was in fact blocking the unexpected network traffic. Once we configured that software firewall to allow traffic to and from DashBoard, a line of communication was effectively generated.

3.2 Safe Operation

Guidelines pertaining to the safe operation of our prototype is straightforward and logical to most common users. There are basically no significant hazards or dangers related to the functioning and operation of our prototype device, however an unlikely and potentially hazardous scenario is worth mentioning.

Although it is more oriented towards the construction and integration of the nodeMCU device with the printed circuit board (PCB), potential dangers revolve around the process of combining the two components together by means of soldering. The noteworthy hazards inherent to soldering include burns and minor shocks. In order to avoid these unwanted outcomes, one should ensure that all safety precautions are followed. The safety guidelines that our team resorted to can be found in the course's "*Soldering Laboratory Manual*" provided through the University of Ottawa's virtual campus BrightSpace page (Knox 2019).

3.3 Prototype Installation

Due to the simplicity of our prototype’s design and operation, the steps required to effectively install both the physical and electrical/software components of the prototype are more or less straightforward.

With regard to the physical aspect of our prototype, once the wire configuration for the nodeMCU device is adequate, future designers need only to power the nodeMCU device with a micro USB cable. A 6V USB power outlet or a 120 to 6V USB block connected to the wall can be used. It can be connected to a laptop as well, when attempting to test and debug the Arduino-related coding blocks, in order to establish a reliable, continuous, and accurate response of the information transmitted from the PIR motion sensor to the nodeMCU device. This will also allow the user to see the output of the sensor on the serial monitor in the Arduino IDE.

The sensor should be mounted on the front right quadrant of the printer. The back left and front left corners are to be avoided since these will interfere with the printer operation. The back right corner is also not recommended since any motion in front of the sensor is undesirable and will trigger the sensor. The nodeMCU box can be conveniently mounted close to the sensor with the provided hook mount on the side of the printer.

4.0 Prototype Maintenance

There is no maintenance to be done on the device. Once installed it will run for the lifetime of the components. If a component was to fail, the faulty component is to be identified and replaced.

5.0 Conclusions & Recommendations

Through the use of multiple iterations of prototyping, our team was able to successfully target critical systems and subsystems that are integral to our project to test functionality and its overall compatibility with the final design. Our first prototype only included a general layout of what systems were to be targeted and how they should interact, while the second prototype allowed us to take several of these key ideas and implement them in order to acquire experimental data. Among these implementations, systems such as DashBoard, motion sensing, and the coding required for both were considered to be critical; as our final design is contingent on their functionality. For the second prototype, we were able to confirm the use of our specific PIR motion sensor utilizing a Arduino code to display whether our motion was detected, as an object moved into its range. Our third and final prototype was developed to send data signals to DashBoard that determines the availability of 3D printers, in a real-time fashion.

The challenges that proved to significantly impede the advancement of this project were essentially the hardware components of the design. Initially, our challenges involved the construction of the PIR motion sensor's casing, that is to be mounted on the corner of the printer, by means of the SolidWorks software and 3D printers, respectively; the establishment of the hotspot network to the DashBoard to initiate the listener server; creating an automatic loop of relevant printer information gathered into the network. There were some difficulties connecting the network to DashBoard and starting the listener server, however, we were successful in connecting the nodeMCU to the Wifi network. A hotspot connection with a 2.4GHz network band was the alternative option that our team decided to pursue, given the fact that the UOdevice network would continuously fail. Our recommendation for this challenge are to ensure the PC Firewall is disabled and also uses a 2.4GHz network band; not 5GHz. Challenges were overcome by doing online research, watching instructive tutorials such as "*Tutorial 2 - Listen for Sensor Value*", contacting those more knowledgeable (Ross DashBoard technical support members, Dr. Knox, etc). The final challenges we faced were difficulties soldering the hardware components due to lack of reliable uOttawa equipment and personal expertise. Finally, the lessons that we collectively as well as individually learned, from the challenges we faced, include:

- *Organization of time*
    - The challenges we faced were primarily due to lack of time to re-design, or re-implement certain aspects of our project. Given nobody in our group has prior coding/software experience, we spent the majority of our project time attempting

    to connect the nodeMCU device, through a listener server or hotspot, to DashBoard. Had we have started sooner, and gotten DashBoard to properly communicate, we may have been able to elevate our project; more so in terms of its functionality and Design Day presentation.
  - Towards the end of the project, a requirement was set to solder the wiring of the hardware components of our PIR motion sensor and nodeMCU device. After soldering, the nodeMCU was entirely dysfunctional. If this was done sooner, a contingency plan to purchase a new nodeMCU device, or desolder, would have been pursued and later implemented.
- *<u>Soldering experience</u>*
  - One of our greatest challenges was soldering the wiring components together without having an impact on the functionality of the nodeMCU. For our group, there was no prior hands-on experience with soldering, therefore we were subject to error. Moving forward, we will ensure we have a high-quality soldering iron and a group member who has gained, or already has, adequate soldering experience.

6.0 Bibliography

- Knox, D., and Hunter, R. (2019). *DashBoard with Arduino - Tutorial 2: Listen for Sensor Value*.
- Knox, D. (2019). *Soldering Laboratory Manual*.

APPENDIX I: Design Files

Design files are made available at:

https://makerepo.com/uguit022/gng1103-a12-3d-printer-monitoring-system