GNG1103

Design Project User Manual

3Detect - A 3D Printer Monitoring System

Submitted by:

Team Nandos, A15

Erin Cox, 300051053

Carly Dawe, 300060960

James Lu, 300060342

Lynne Ngo, 300068874

Sophia Ruberto, 300052105

December 7, 2019

University of Ottawa

Abstract

CEED needs an automated system that is easy to use and cost-efficient whilst also alerting employees of the current status of the printers. It uses a (passive infrared sensor) motion sensor, a nodeMCU, and Dashboard to monitor the status of a 3D printer and inform users about the said status. A motion sensor is connected to a nodeMCU and sends the nodeMCU the data it receives, when it detects motion and when it does not detection motion. The nodeMCU takes this data and relays it to Dashboard, through the code in the Dashboard, the interface changes to alert CEED employees when a print is completed. The motion sensor is placed behind the 3D printer and angled towards the spool of filament. The spool of filament turns when the printer is printing, this motion is detected by the motion sensor and the information is relayed to the dashboard interface that the user sees. Once the spool stops turning, the dashboard will display pick up to let users know that a print is done and needs to be picked up. This method of motion detection also allows for detection of jams in the nozzle. The wheel will not be turning so it notifies to pick up the print, when the user goes to pick up the print they will find that it has failed and can clear the print bed and free up the machine for the next person to use. The motion sensors will change the colour of the boxes to match the status but CEED employees can also manually change the status of the box if needed using the buttons on the Dashboard. This report will cover how the prototype is made, all the materials needed for the prototype, how to use the prototype,how to maintain the prototype, and steps for the future.

Table of Contents

List of Tables

List of Figures

List of Acronyms

Acronym	Definition
MDF	Medium-density fibreboard
Arduino Node MCU	- (MCU)- Node MicroController Unit
V	voltage
USB	Universal Serial Bus
CEED	Center for entrepreneurship of engineering and design
UI	User Interface
PIR	Passive Infrared Radiation

Table 1 - Acronyms

1 Introduction

CEED provides a space and equipment for students to be creative. Students can use the Makerspace, Makerlab, and Brunsfield to create personal projects and bring their entrepreneurial ideas to life. Makerspace is also available to the public on Sundays, With the large amounts of people who use Makerspace, CEED employees are busy ensuring that all the equipment is running properly and the space is clean for new incoming people to work at. The time they use to constantly check the 3D printers to see if prints are done is time that can be spent on other things such as helping people. Therefore, CEED needs an automated system that is easy to use and cost-efficient whilst also alerting employees of the current status of the printers. 3Detect is that automated system. Using a PIR motion sensor, a nodeMCU, and Dashboard, 3Detect will monitor the status of 3D printers and inform the CEED employees of the status of the 3D printer through a Dashboard interface so they know when a print is completed.

2 How the Prototype is Made

This section of the manual will inform the user how to create the prototype, as well as supply a list of components, their prices, and the needed equipment. Instructions for both the mechanical components and software components are provided.

2.1 Mechanical and Software Components

2.1.1 BOM (Bill of Materials)

Material	Cost	Link
⅛ 12 inch x 24 inch MDF	$2.50	https://makerstore.ca/shop?keywords=mdf&olsPage=products%2Fmdf&page=2
NodeMCU	$17.00	Available from Makerstore
Female to Male Jumper Cables	$1.00	https://makerstore.ca/shop?keywords=jumper%20cables&olsPage=products%2Fjumper-cables-per-10&page=2
Female to Female Jumper Cables	$6.99	https://www.amazon.ca/gp/product/B019SX71TC/ref=ppx_yo_dt_b_asin_title_o06_s00?ie=UTF8&psc=1
Battery Clip	$13.49	https://www.amazon.ca/gp/product/B00O5B5MVW/ref=pp

		x_yo_dt_b_asin_title_o02_s00?ie=UTF8&psc=1
9 Volt Battery	$6.99	https://www.canadiantire.ca/en/pdp/duracell-copper-top-alkaline-9v-battery-0653017p.html#srp
PIR Motion Sensor	$10.99	https://www.amazon.ca/gp/product/B019SX6ZR6/ref=ppx_yo_dt_b_asin_title_o09_s00?ie=UTF8&psc=1

Table 2 - Bill of Materials

2.1.2 Equipment list

Equipment	**Use**
Star head Screwdriver	Adjust sensitivity and delay Time
Micro USB to USB Cable	Connect battery pack to NodeMCU
Battery Pack	Power NodeMCU
Epoxy	Adhere the box together
Laser Cutter Machine	Cut box pieces

Table 3 - Equipment Lists

2.1.3 Instructions

The instructions on how to create the final product are included in the sections below.

2.1.3.1 NodeMcu and PIR Motion Sensor Setup including corresponding Arduino Code

1. Download Arduino from the following link https://software.intel.com/en-us/get-started-arduino-install
2. Open arduino and change board to “NodeMCU 1.0 (ESP-12E Module)”

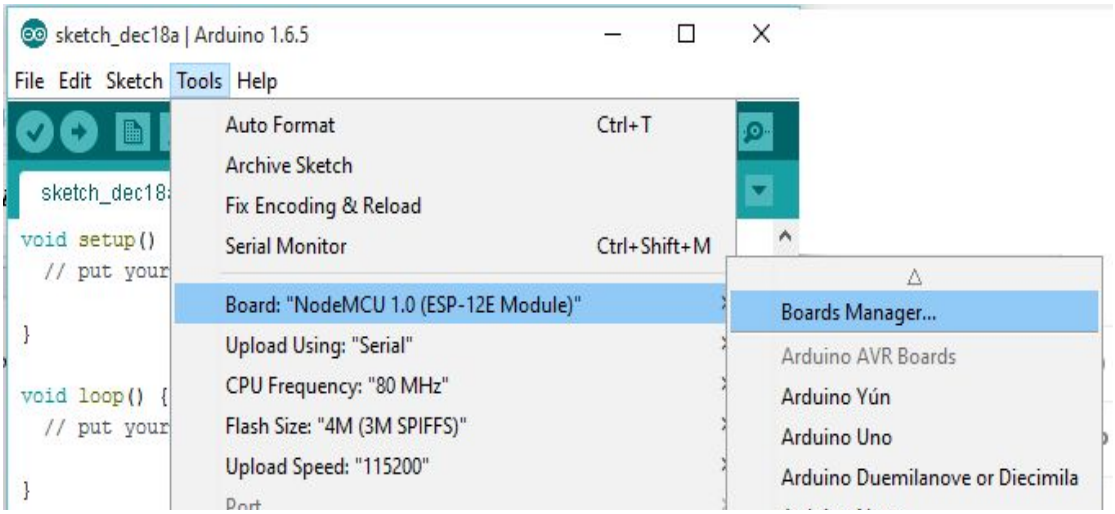

Figure 1 - Board Change

3. Initialize parameters sensor and state.

```
int sensor = D2;
int state = LOW;
```

Figure 2 - State Initialization

4. Create a setup and include the pinmode and Serial.begin. Ensure Baud rate matches serial monitor of 9600.

```
void setup() {

  pinMode(sensor, INPUT);
  Serial.begin(9600);
```

Figure 3 - Set Up Code

5. Create a loop to determine if the state is high or low. Include a time step of a 1000 to have a 1 second delay.

```
void loop() {
  long in = digitalRead(sensor);
  if(in == HIGH){
    //Serial.println("1");
    //delay(1000);
    if (state==LOW)
    {
      state=HIGH;
      Serial.println("1");
    }
```

Figure 4 - Code Loop

6. Use a female to female jumper cable to connect output on PIR motion sensor to pin D2

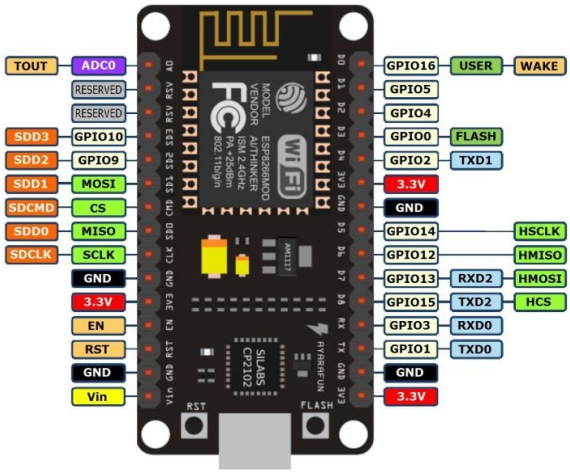

Figure 5 - Pinout for NodeMCU

7. Attach the ground wire on battery clip to board on a ground pin (represented by a GND on board) using a female to male jumper cable
8. Attach voltage on battery clip to the voltage on PIR motion sensor using a female to male jumper cable

Figure 6 - PIR Motion Sensor Pinout

9. Attach another ground pin on NodeMCU to ground pin on PIR motion sensor

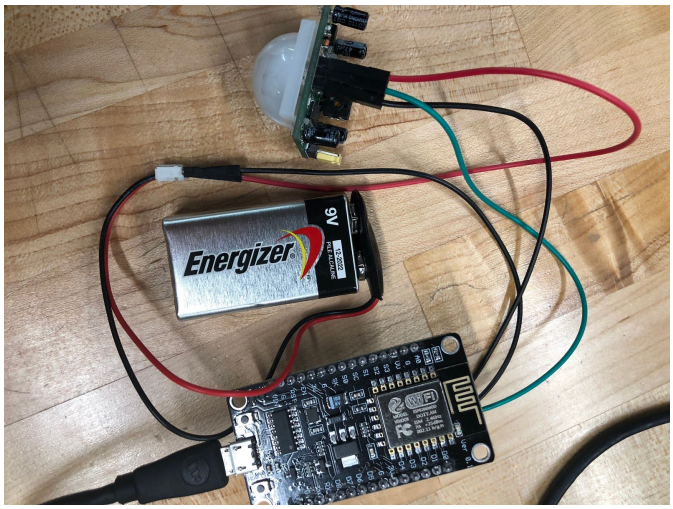

Figure 7 - Final Assembly

2.1.3.2 Listener Server Code

1. In Dashboard you create a text entry parameter and name it. In this case we named it ardMessage.
2. Make sure its initial value is “N/A” and the display type is “Read Only”.
3. On the dashboard itself double click on the panel to open up the general edit window and in the meta folder create a listening server using the “<listener/>” button.
4. In the listener menu ensure you check start automatically and select the listener as a server and give it a port number.
5. In delimiter type (TCP) select New Line from the drop down menu. Then add a task and write in the code.

```
If(event.getEventType() == 1) {
        Var rawData = event.getBytesAsString();
        Ogscript.debug(rawData);
Params.setValue('ardMessage',0,rawData);
```

Figure 8 Dashboard Listener Code

6. Use the arduino template code WiFiClientBasic
7. Add the name and password from your internet to STASSID and STAPSK
8. Delete any existing client.println(“”) from the code
9. Replace the template code with your own.

```
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 the request to the server
  delay(1000);
  float temperByte = analogRead(A0);
  Serial.println(temperByte);
  float converted = temperByte/1024;
  Serial.println(converted);
  converted = converted*2200/(1-converted);
  Serial.println(converted);
  converted = (2715.2 - converted)/57.2;
  Serial.println(converted);
  client.print(converted);
  client.print("\n");

  Serial.println("closing connection");
  client.stop();

  Serial.println("wait 5 sec...");
  delay(5000);
}
```

Replace this stuff with your own code

Figure 9 Template Code With Indication

10. Ensure when placing Arduino code into the template the state and sensor declarations are global.
11. Copy and paste your existing code into the void setup loop and void loop.
12. Ensure your port number matches the one you created on the dashboard.
13. To see our final Arduino code please refer to appendix B

2.1.3.3 Dashboard User Interface and Code

This section will provide instructions to create the dashboard interface and code the buttons behind it. This interface only shows an accurate status for one printer, specifically "Printer1." The following is an image of the finished UI

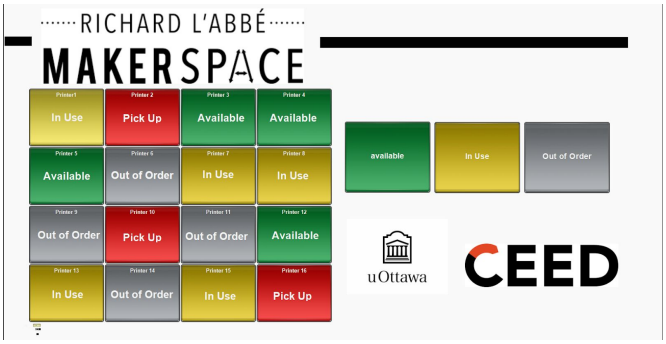

Figure 10 Dashboard Interface

The source code for the UI is in appendix A, for further reference.

1. Download dashboard using the following link https://www.rossvideo.com/support/software-downloads/dashboard/
2. Create a new file, go to file, new, new customPanel File. Save the file as a recognizable name

3. Click Finish
4. Click PanelBuilder Edit Mode
5. Create a Table with the amount of printers you have, click “fill with buttons.”
6. Double click a button, name it printer #1, go to style, then font, then text alignment and then click top.
7. Do the same for all other printers
8. Create 3 large buttons, using the “button” button. Name them “Available” “In Use” and “Out of Order.”
9. Create 3 parameters. Name them “text”, “arduino” and “button”. The text parameter should be a text type, the arduino and button parameters should be number type.
10. Align the text parameter on top of the button “Printer 1.”
11. Put the following 3 codes into the tasks area in the arduino parameter.

```
5  if (params.getValue('arduino', 0)  ==  0)
6  {
7     if (params.getValue('button', 0)  ==  0) { ogscript.setStyle("printer", "bg#009933");
8  } else {
9     if (params.getValue('button', 0)  ==  2) { ogscript.setStyle("printer", "bg#999EA3");
10    } else { if (params.getValue('button', 0)  ==  3) { ogscript.setStyle("printer", "bg#FF0000");
11    } else { ogscript.setStyle("printer", "bg#FFEC39");
12    } } } } else { params.setValue('button', 0, 3);
13    ogscript.setStyle("printer", "bg#FFEC39");
14 }
```

Figure 11 Colour Dashboard Code

```
5  if (params.getValue('arduino', 0)  ==  0)
6  {
7     if (params.getValue('button', 0)  ==  0) { params.setValue('text', 0, "Available");
8  } else {
9     if (params.getValue('button', 0)  ==  2) { params.setValue('text', 0, "Out of Order");
10    } else { if (params.getValue('button', 0)  ==  3) { params.setValue('text', 0, "Pick Up");
11    } else { params.setValue('text', 0, "In Use");
12    } } } } else { params.setValue('text', 0, "In Use");
13 }
```

Figure 12 Text Dashboard Code

```
5  params.setValue('arduino', 0, params.getValue('ardMessage', 0));
```

Figure 13 Parameter Transfer Code

12. Do the same for the button parameter using the following code.

```
5  if (params.getValue('arduino', 0)  ==  0)
6  {
7     if (params.getValue('button', 0)  ==  0) { params.setValue('text', 0, "Available");
8  } else {
9     if (params.getValue('button', 0)  ==  2) { params.setValue('text', 0, "Out of Order");
10    } else { if (params.getValue('button', 0)  ==  3) { params.setValue('text', 0, "Pick Up");
11    } else { params.setValue('text', 0, "In Use");
12    } } } } else {
13 }
```

Figure 14 Button Parameter Code

13. Double click the “Available” button, insert the following code into the tasks. Do the same for “In Use” and “Out of Order”

```
params.setValue('button', 0, 0);
```

Figure 15 Available Button Code

```
5 params.setValue('button', 0, 1);
```

Figure 16 In Use Button Code

```
5 params.setValue('button', 0, 2);
```

Figure 17 Out of Order Button Code

14. Double click the background, go to style, set a background colour using “Background Colour” this UI is white.
15. To add photos, create a basic canvas, go to style for that canvas and choose a background URL, Click fit in background fill.
16. To create the black aesthetic outline bar, create a basic canvas and colour it black.
17. Colour the other Printer buttons by double clicking and setting a background colour. (This colour will not change, only the printer one button is functional)
18. Add text on top of the printer buttons by creating a label and placing it on top of the button. (This text will not change, only printer one text is functional)

2.1.3.4 - Laser-cut MDF Box

1. Identify and measure the components that must go in the box
2. Determine an appropriate size for the box that will allow it to contain all the components
3. Go to https://www.makercase.com/
4. Enter in the dimensions determined in step 2
5. Ensure to select “Finger” under “Edge Joints”
6. Download the box plan
7. Open the box plan in Inkscape
8. Under “Fill”, select “No paint”
9. Under “Stroke paint”, select “Flat colour” and “RGB”. Ensure that R:0, G:0, B:0, A:255
10. Under “Stroke style”, ensure that the width is 0.001 in

11. Add a circle in the front piece for the motion sensor to stick out
12. Save the file as a PDF
13. Use the PDF and a laser cutting machine to cut the MDF
14. Use epoxy to adhere the box together

3 How to Use the Prototype

This prototype is able to sense the motion of a 3D printer using a PIR motion sensor, the information is sent to dashboard and will alert employees of the print's status. The motion sensor will send 1's and 0's through a listener server to a parameter called "ardMessage" in dashboard. 1's mean there is motion while 0's mean there is no motion. An employee must click one of the three buttons on the interface to further decide the status of the printer. If, for example, the button corresponding to the printer turns red, meaning there is no motion, the employee must then pick up the print and then click the "Available" button. The button corresponding to the printer will turn green and the text will change to "Available." The same functions can be done for the "Out of Order" button and the "In Use" button.

The employee must place the laser cut box behind a 3D printer, with the motion sensor pointed at the spool of filament. The box must contain the motion sensor, batteries, and the setup Node MCU. The Arduino Node MCU must be connected by jumper cables to the PIR motion sensor by the input pin and the ground pin, while the external 9V battery pack must be connected to the power of the PIR motion sensor and the Arduino Node MCU. To properly hook up the Arduino with the motion sensor, you want to have three different jumper wires (black, green and red). First, take the black wire and connect from the ground on the PIR motion sensor to the ground on the NodeMCU. Then take a green wire and input it in the middle slot of the PIR motion sensor, connect the other end of the wire to the input pin chosen on the NodeMcu. Finally, take the red wire to the red wire input on the 9V battery clip. Finally, take an extra wire and connect that to the black battery clip wire input and connect the other end to the ground on the nodemcu, but this time on the other ground pin of the other side you previously placed. See section 2 for specific instructions on how to set this up. Dashboard must be installed on the desktop computer in use, see section 2.1.3.3 for specific instructions on how to create the dashboard interface. A listener server must be implemented to send signals from the nodeMCU to the dashboard. This can be done using the instructions in section 2.1.3.2.

4 How to Maintain the Prototype

1. Only attach pin inputs from motion sensors to pins on NodeMCU D1,D2,D5,D6.According to the GPIO pin diagram of the Arduino NodeMCU, these 4

inputs are free and will not affect your nodeMCU. As we learned from past experiences, we accidentally attached our input to D7 and this actually fried our NodeMCU.

2. If you are planning to use AKURU PIR motion sensors, you must have an external battery that can at least pump out 5V to efficiently power the motion sensor.
3. Motion sensors must be kept at temperatures of -20° to +80° Celsius.
4. Do not spill liquids, or food on NodeMCU. Do not touch the microchip of the Nodecmu, rather hold the sides to not damage the Arduino.
5. Test all motion sensors and wires bought online's current voltage with a voltmeter to make sure they are able to function properly.
6. If you notice that the microchip on the Arduino NodeMCU is getting very hot, this is a problem, and immediately disconnect wires connected to a battery pack and motion sensor. Then to test if the Arduino is fried or not, unplug all wires connected except for the microUSB to USB cable connected to the computer, and try to upload an arbitrary code, if the NodeMCU does not light up or upload code, your Arduino NodeMCU is fried.
7. If applicable, add a fan inside the box to prevent overheating of the Arduino Node MCU.
8. Do not attach the PIR motion sensor to a power source that is greater than 12V, this will harm the motion sensors.
9. Make sure that all wires are connected properly in the proper inputs, ground, Vin and input, inputting wires to the incorrect port can damage the Arduino NodeMCU.
10. Do not drop the Nodemcu or motion sensor, do not drop in whirlpools, do not toss in hurricanes, do not expose to aggressive lightning, finally do not leave the nodeMCU in the cold Canadian winter storms.

5 Conclusions and Recommendations for Future Work

Our problem statement was CEED needs an automated system that is cost-efficient, easy to use whilst also alerting employees of the current status of the prints. PIR motion sensors connected to proper external batteries will be able to track the motion of the turning spool. This detection will help the Arduino NodeMCU receive data pertaining to the 3D printer. The Dashboard will then change colors which will relate to the current printer status, red for pick up, yellow for in use, green for available, and grey for out of order. To conclude, our project uses the PIR motion sensor to track the motion of the spool turning behind the 3D printer in MakerSpace. The Arduino NodeMCU will then change its State to HIGH or LOW, and then send a "1" and "0" to the dashboard. The Dashboard will then take the "1" and "0" and change the button color to the corresponding printer-state color. For future work, try to buy multiple PIR motion sensor, to be able to track the motion of multiple printers. Then change your Dashboard code to accommodate for the tracking of multiple printers and their statuses. We recommend that

all wires are connected to the proper port on the Arduino NodeMCU, also, check the required voltage that runs for the specific motion sensor you have purchased.

6 Bibliography

1- Hunter, Robert. “Listener Tutorial with NodeMCU.” *Brightspace*, Robert Hunter, 21 Nov. 2019, https://uottawa.brightspace.com/d2l/le/content/116873/viewContent/2488425/View.

2- Lopez, george. *Pintrest-Motion Sensor*, Google, 11 Oct. 2011, https://www.google.com/search?rlz=1C1CHBF_enCA809CA809&biw=1280&bih=610&tbm=isch&sa=1&ei=QXXgXfzqOMjEsAW-nqr4Cg&q=pir+motion+sensor+pinout&oq=pir+motion+sensor+pinout&gs_l=img.3..0.1657.3637..4514...0.0..0.127.1023.0j9.....0....1..gws-wiz-img.....0i67j0i8i30.8hMOXoAb6io&ved=0ahUKEwi8j9Wpo47mAhVIIqwKHT6PCq8Q4dUDCAc&uact=5#imgrc=Rn5LJyCxgPmsEM:

3- Jayakumar, Magesh, and Instructables. “Quick Start to Nodemcu (ESP8266) on Arduino IDE.” *Instructables*, Instructables, 30 Sept. 2017, https://www.instructables.com/id/Quick-Start-to-Nodemcu-ESP8266-on-Arduino-IDE/.

4- Pelayo, Roland. “NodeMCU Pinout Reference.” *Microcontroller Tutorials*, Microcontroller Tutorials, 2 May 2019, https://www.teachmemicro.com/nodemcu-pinout/.

APPENDICES

APPENDIX A: The pride and joy Dashboard code. This code took the “1” and “0” and converted them into different colours on the dashboard interface.

```
<abs contexttype="opengear" id="_top" keepalive="true" style="bg#FAF9F9;"
virtualheight="1000" virtualwidth="1700">

  <meta>

    <listener autostart="true" delimitertype="newline" listenport="21888">

      <task tasktype="ogscript">if(event.getEventType()==1){

  var rawData = event.getBytesAsString();
```

```
  ogscript.debug(rawData);

  params.setValue('ardMessage',0,rawData);

}</task>

    </listener>

    <params>

      <param access="1" constraint="0.0;3.0;0.0;3.0;1" constrainttype="INT_STEP_RANGE"
name="arduino" oid="arduino" precision="0" type="INT32" value="0" widget="spinner"/>

      <param access="1" maxlength="0" name="ardMessage" oid="ardMessage"
type="STRING" value="0" widget="text-display"/>

      <param access="1" constraint="0.0;2.0;0.0;2.0;1" constrainttype="INT_STEP_RANGE"
name="button" oid="button" precision="0" type="INT32" value="2" widget="spinner"/>

      <param access="1" maxlength="0" name="text" oid="text" type="STRING" value="Out of
Order" widget="label"/>

    </params>

  </meta>

  <param constraint="0.0;1.0;0.0;1.0;1" constrainttype="INT_STEP_RANGE" expand="true"
height="6" id="a" left="66" name="arduono input" oid="arduino" precision="0" top="890"
width="30">

    <task tasktype="ogscript">/*! block
id=1022,1021,1048,1050,1052,1049,1051,1053,1068,1069,1067,1070,1066,1071 !*/

if (params.getValue('arduino', 0)  ==  0)

{

        if (params.getValue('button', 0)  ==  0) { ogscript.setStyle("printer", "bg#009933");

} else {

        if (params.getValue('button', 0)  ==  2) { ogscript.setStyle("printer", "bg#999EA3");
```

```
	} else { if (params.getValue('button', 0) == 3) { ogscript.setStyle("printer",
"bg#FF0000");

	} else { ogscript.setStyle("printer", "bg#FFEC39");

	} } } } else { params.setValue('button', 0, 3);

	ogscript.setStyle("printer", "bg#FFEC39");

}

/*!!

 &lt;block id="1022" type="if" x="283" y="10" w="268" INPUT1="ID:1021"
OPERATION="equals" INPUT2="0" TRUE="ID:1048" FALSE="ID:1066" IGNORE="" /&gt;

&lt;block id="1021" type="param__top&amp;amp;arduino (arduino)[0]" x="10" y="10"
w="243" SET="" /&gt;

&lt;block id="1048" type="if" x="1127" y="60" w="268" INPUT1="ID:1050"
OPERATION="equals" INPUT2="0" TRUE="ID:1052" FALSE="ID:1049" IGNORE="" /&gt;

&lt;block id="1050" type="param__top&amp;amp;button (button)[0]" x="854" y="60" w="243"
SET="" /&gt;

&lt;block id="1052" type="ogscript_setstyle" x="1996" y="110" w="243" ID="printer"
STYLE="bg#009933" /&gt;

&lt;block id="1049" type="if" x="1698" y="150" w="268" INPUT1="ID:1051"
OPERATION="equals" INPUT2="2" TRUE="ID:1053" FALSE="ID:1068" IGNORE="" /&gt;

&lt;block id="1051" type="param__top&amp;amp;button (button)[0]" x="1425" y="150"
w="243" SET="" /&gt;

&lt;block id="1053" type="ogscript_setstyle" x="2308" y="187" w="243" ID="printer"
STYLE="bg#999EA3" /&gt;

&lt;block id="1068" type="if" x="2052" y="330" w="268" INPUT1="ID:1069"
OPERATION="equals" INPUT2="3" TRUE="ID:1067" FALSE="ID:1070" IGNORE="" /&gt;

&lt;block id="1069" type="param__top&amp;amp;button (button)[0]" x="1725" y="343"
w="243" SET="" /&gt;
```

<block id="1067" type="ogscript_setstyle" x="2345" y="355" w="243" ID="printer" STYLE="bg#FF0000" />

<block id="1070" type="ogscript_setstyle" x="1657" y="451" w="243" ID="printer" STYLE="bg#FFEC39" />

<block id="1066" type="param_setvalue" x="530" y="324" w="318" PARAM="[root, Params for GNG.grid, Not In Menu, button (button)]" VALUE="3" next="ID:1071" />

<block id="1071" type="ogscript_setstyle" x="10" y="567" w="243" ID="printer" STYLE="bg#FFEC39" />

!!*/

/*!!<checksum>689bbb21d20de9cc0a041ec5a468f5aa</checksum>!!*/</task>

<task tasktype="ogscript"/>

<task tasktype="ogscript">/*! block id=1062,1061 !*/

```
params.setValue('arduino', 0, params.getValue('ardMessage', 0));
```

/*!!

<block id="1062" type="param__top&arduino (arduino)[0]" x="10" y="192" w="243" SET="ID:1061" />

<block id="1061" type="param__top&ardMessage (ardMessage)[0]" x="10" y="100" w="243" SET="" />

!!*/

/*!!<checksum>96f3431c003ca3199bb967f579bfa7c4</checksum>!!*/</task>

<task tasktype="ogscript">/*! block id=1007,1005,1008,1006,1009,1010,1016,1013,1014,1015,1017,1018,1019 !*/

```
if (params.getValue('arduino', 0)  ==  0)

{

        if (params.getValue('button', 0)  ==  0) { params.setValue('text', 0, "Available");
```

```
} else {

        if (params.getValue('button', 0)  ==  2) { params.setValue('text', 0, "Out of Order");

        } else { if (params.getValue('button', 0)  ==  3) { params.setValue('text', 0, "Pick Up");

        } else { params.setValue('text', 0, "In Use");

        } } } } else { params.setValue('text', 0, "In Use");

}

/*!!

 &lt;block id="1007" type="if" x="283" y="10" w="268" INPUT1="ID:1005"
OPERATION="equals" INPUT2="0" TRUE="ID:1008" FALSE="ID:1019" IGNORE="" /&gt;

&lt;block id="1005" type="param__top&amp;amp;arduino (arduino)[0]" x="10" y="10"
w="243" SET="" /&gt;

&lt;block id="1008" type="if" x="854" y="60" w="268" INPUT1="ID:1006"
OPERATION="equals" INPUT2="0" TRUE="ID:1009" FALSE="ID:1010" IGNORE="" /&gt;

&lt;block id="1006" type="param__top&amp;amp;button (button)[0]" x="581" y="60" w="243"
SET="" /&gt;

&lt;block id="1009" type="param_setvalue" x="1723" y="110" w="318" PARAM="[root,
Params for GNG.grid, Not In Menu, text (text)]" VALUE="Available" /&gt;

&lt;block id="1010" type="if" x="1425" y="150" w="268" INPUT1="ID:1016"
OPERATION="equals" INPUT2="2" TRUE="ID:1013" FALSE="ID:1014" IGNORE="" /&gt;

&lt;block id="1016" type="param__top&amp;amp;button (button)[0]" x="1152" y="150"
w="243" SET="" /&gt;

&lt;block id="1013" type="param_setvalue" x="2642" y="200" w="318" PARAM="[root,
Params for GNG.grid, Not In Menu, text (text)]" VALUE="Out of Order" /&gt;

&lt;block id="1014" type="if" x="2344" y="240" w="268" INPUT1="ID:1015"
OPERATION="equals" INPUT2="3" TRUE="ID:1017" FALSE="ID:1018" IGNORE="" /&gt;
```

<block id="1015" type="param__top&button (button)[0]" x="2071" y="240" w="243" SET="" />

<block id="1017" type="param_setvalue" x="3338" y="290" w="318" PARAM="[root, Params for GNG.grid, Not In Menu, text (text)]" VALUE="Pick Up" />

<block id="1018" type="param_setvalue" x="2990" y="330" w="318" PARAM="[root, Params for GNG.grid, Not In Menu, text (text)]" VALUE="In Use" />

<block id="1019" type="param_setvalue" x="10" y="446" w="318" PARAM="[root, Params for GNG.grid, Not In Menu, text (text)]" VALUE="In Use" />

!!*/

/*!!<checksum>b7d1aa597700c8e9ea57ebf61de44a85</checksum>!!*/</task>

</param>

<table height="643" left="64" top="231" width="784">

<tr>

<button buttontype="push" colspan="1" fill="both" id="printer" name="Printer1" rowspan="1" style="txt-align:north;" weightx="1.0" weighty="1.0">

<task tasktype="ogscript"/>

</button>

<button buttontype="push" colspan="1" fill="both" name="Printer 2" rowspan="1" style="bg#FF0000;txt-align:north;" weightx="1.0" weighty="1.0"/>

<button buttontype="push" colspan="1" fill="both" name="Printer 3" rowspan="1" style="bg#009933;txt-align:north;" weightx="1.0" weighty="1.0"/>

<button buttontype="push" colspan="1" fill="both" name="Printer 4" rowspan="1" style="bg#009933;txt-align:north;" weightx="1.0" weighty="1.0"/>

</tr>

<tr>

```
        <button buttontype="push" colspan="1" fill="both" name="Printer 5" rowspan="1"
style="bg#009933;txt-align:north;" weightx="1.0" weighty="1.0"/>

        <button buttontype="push" colspan="1" fill="both" name="Printer 6" rowspan="1"
style="bg#999EA3;txt-align:north;" weightx="1.0" weighty="1.0"/>

        <button buttontype="push" colspan="1" fill="both" name="Printer 7" rowspan="1"
style="bg#EBCA00;txt-align:north;" weightx="1.0" weighty="1.0"/>

        <button buttontype="push" colspan="1" fill="both" name="Printer 8" rowspan="1"
style="bg#EBCA00;txt-align:north;" weightx="1.0" weighty="1.0"/>

    </tr>

    <tr>

        <button buttontype="push" colspan="1" fill="both" name="Printer 9" rowspan="1"
style="bg#999EA3;txt-align:north;" weightx="1.0" weighty="1.0"/>

        <button buttontype="push" colspan="1" fill="both" name="Printer 10" rowspan="1"
style="bg#FF0000;txt-align:north;" weightx="1.0" weighty="1.0"/>

        <button buttontype="push" colspan="1" fill="both" name="Printer 11" rowspan="1"
style="bg#999EA3;txt-align:north;" weightx="1.0" weighty="1.0"/>

        <button buttontype="push" colspan="1" fill="both" name="Printer 12" rowspan="1"
style="bg#009933;txt-align:north;" weightx="1.0" weighty="1.0"/>

    </tr>

    <tr>

        <button buttontype="push" colspan="1" fill="both" name="Printer 13" rowspan="1"
style="bg#EBCA00;txt-align:north;" weightx="1.0" weighty="1.0"/>

        <button buttontype="push" colspan="1" fill="both" name="Printer 14" rowspan="1"
style="bg#999EA3;txt-align:north;" weightx="1.0" weighty="1.0"/>

        <button buttontype="push" colspan="1" fill="both" name="Printer 15" rowspan="1"
style="bg#EBCA00;txt-align:north;" weightx="1.0" weighty="1.0"/>
```

<button buttontype="push" colspan="1" fill="both" name="Printer 16" rowspan="1" style="bg#FF0000;txt-align:north;" weightx="1.0" weighty="1.0"/>

</tr>

</table>

<button buttontype="push" height="198" id="available" left="879" name="available" style="bg#009933;font:bold;size:Big;" top="321" width="223">

<task tasktype="ogscript">/*! block id=1025,1022 !*/

params.setValue('button', 0, 0);

/*!!

<block id="1025" type="param__top&button (button)[0]" x="182" y="111" w="243" SET="ID:1022" />

<block id="1022" type="integer" x="10" y="100" w="168" VALUE="0" />

!!*/

/*!!<checksum>bb6288c168b29d4db3241ead701bc278</checksum>!!*/</task>

</button>

<button buttontype="push" height="198" id="available" left="1110" name="available" top="322" width="223"/>

<button buttontype="push" height="198" id="use" left="1110" name="In Use" style="bg#EBCA00;font:bold;size:Big;" top="322" width="223">

<task tasktype="ogscript">/*! block id=1026,1024 !*/

params.setValue('button', 0, 1);

/*!!

<block id="1026" type="param__top&button (button)[0]" x="217" y="145" w="243" SET="ID:1024" />

<block id="1024" type="integer" x="18" y="144" w="168" VALUE="1" />

!!*/

/*!!<checksum>f581c1aa32a7be98619b603fea990f52</checksum>!!*/</task>

</button>

<button buttontype="push" height="198" id="out" left="1342" name="Out of Order" style="bg#999EA3;font:bold;size:Big;" top="323" width="223">

<task tasktype="ogscript">/*! block id=1030,1026 !*/

params.setValue('button', 0, 2);

/*!!

<block id="1030" type="param__top&button (button)[0]" x="211" y="95" w="243" SET="ID:1026" />

<block id="1026" type="integer" x="10" y="100" w="168" VALUE="2" />

!!*/

/*!!<checksum>ab7578361b47f1487e71255ee6e4ad2e</checksum>!!*/</task>

</button>

<abs height="207" left="87" style="bg-u:makerspace%20logo.png;bg-fill:fit;" top="12" width="699"/>

<abs height="26" left="2" top="89" width="70">

<abs bottom="0" left="0" right="0" style="bg#000000;" top="0"/>

</abs>

<abs height="30" left="816" style="bg#030202;" top="90" width="795"/>

<abs height="273" left="1190" style="bg-u:CEED.png;bg-fill:fit;" top="580" width="396"/>

<abs height="241" left="885" style="bg-u:uottawa.png;bg-fill:fit;" top="572" width="261">

<abs left="200" top="140"/>

<abs height="3" left="191" top="145" width="1"/>

```
  </abs>

  <param editable="false" expand="true" height="31" left="201" name="ardmessage"
oid="ardMessage" style="fg#0C0505;" top="915" width="124">

    <task tasktype="ogscript"/>

    <task tasktype="ogscript"/>

    <task tasktype="ogscript">/*! block id=1057,1060 !*/

params.setValue('arduino', 0, params.getValue('ardMessage', 0));

/*!!

 &lt;block id="1057" type="param__top&amp;amp;arduino (arduino)[0]" x="10" y="100"
w="243" SET="ID:1060" /&gt;

&lt;block id="1060" type="param__top&amp;amp;ardMessage (ardMessage)[0]" x="10"
y="284" w="243" SET="" /&gt;

!!*/

/*!!&lt;checksum&gt;ce6f7ad0b0b04fa0970dbb6dd832258f&lt;/checksum&gt;!!*/</task>

  </param>

  <param constraint="0.0;3.0;0.0;3.0;1" constrainttype="INT_STEP_RANGE" expand="true"
height="6" left="75" name="button" oid="button" precision="0" top="903" width="15">

    <task tasktype="ogscript">/*! block
id=1007,1005,1008,1006,1009,1010,1016,1013,1014,1015,1017,1018 !*/

if (params.getValue('arduino', 0)  ==  0)

{

        if (params.getValue('button', 0)  ==  0) { params.setValue('text', 0, "Available");

} else {

        if (params.getValue('button', 0)  ==  2) { params.setValue('text', 0, "Out of Order");
```

```
        } else { if (params.getValue('button', 0)  ==  3) { params.setValue('text', 0, "Pick Up");

        } else { params.setValue('text', 0, "In Use");

        } } } } else {

}

/*!!

 &lt;block id="1007" type="if" x="283" y="10" w="268" INPUT1="ID:1005"
OPERATION="equals" INPUT2="0" TRUE="ID:1008" FALSE="" IGNORE="" /&gt;

&lt;block id="1005" type="param__top&amp;amp;amp;arduino (arduino)[0]" x="10" y="10"
w="243" SET="" /&gt;

&lt;block id="1008" type="if" x="854" y="60" w="268" INPUT1="ID:1006"
OPERATION="equals" INPUT2="0" TRUE="ID:1009" FALSE="ID:1010" IGNORE="" /&gt;

&lt;block id="1006" type="param__top&amp;amp;amp;button (button)[0]" x="581" y="60" w="243"
SET="" /&gt;

&lt;block id="1009" type="param_setvalue" x="1723" y="110" w="318" PARAM="[root,
Params for GNG.grid, Not In Menu, text (text)]" VALUE="Available" /&gt;

&lt;block id="1010" type="if" x="1425" y="150" w="268" INPUT1="ID:1016"
OPERATION="equals" INPUT2="2" TRUE="ID:1013" FALSE="ID:1014" IGNORE="" /&gt;

&lt;block id="1016" type="param__top&amp;amp;amp;button (button)[0]" x="1152" y="150"
w="243" SET="" /&gt;

&lt;block id="1013" type="param_setvalue" x="2642" y="200" w="318" PARAM="[root,
Params for GNG.grid, Not In Menu, text (text)]" VALUE="Out of Order" /&gt;

&lt;block id="1014" type="if" x="2344" y="240" w="268" INPUT1="ID:1015"
OPERATION="equals" INPUT2="3" TRUE="ID:1017" FALSE="ID:1018" IGNORE="" /&gt;

&lt;block id="1015" type="param__top&amp;amp;amp;button (button)[0]" x="2071" y="240"
w="243" SET="" /&gt;

&lt;block id="1017" type="param_setvalue" x="3338" y="290" w="318" PARAM="[root,
Params for GNG.grid, Not In Menu, text (text)]" VALUE="Pick Up" /&gt;
```

<block id="1018" type="param_setvalue" x="2990" y="330" w="318" PARAM="[root, Params for GNG.grid, Not In Menu, text (text)]" VALUE="In Use" />

!!*/

/*!!<checksum>d6ab484687393b36e136af3d40773a11</checksum>!!*/</task>

</param>

<param expand="true" height="74" left="38" oid="text" style="txt-align:center;font:bold;size:Bigger;" top="273" widget="label" width="252"/>

<label height="45" left="211" name="Out of Order" style="txt-align:center;font:bold;size:Bigger;" top="446" width="294"/>

<label height="45" left="21" name="Out of Order" style="txt-align:center;font:bold;size:Bigger;" top="610" width="294"/>

<label height="45" left="211" name="Out of Order" style="txt-align:center;font:bold;size:Bigger;" top="446" width="294"/>

<label height="45" left="402" name="Out of Order" style="txt-align:center;font:bold;size:Bigger;" top="614" width="294"/>

<label height="45" left="209" name="Out of Order" style="txt-align:center;font:bold;size:Bigger;" top="773" width="294"/>

<label height="28" left="278" name="Pick Up" style="txt-align:center;font:bold;size:Bigger;" top="299" width="159"/>

<label height="28" left="278" name="Pick Up" style="txt-align:center;font:bold;size:Bigger;" top="626" width="159"/>

<label height="28" left="665" name="Pick Up" style="txt-align:center;font:bold;size:Bigger;" top="787" width="159"/>

<label height="51" left="451" name="Available" style="txt-align:center;font:bold;size:Bigger;" top="286" width="219"/>

<label height="51" left="638" name="Available" style="txt-align:center;font:bold;size:Bigger;" top="286" width="219"/>

<label height="51" left="53" name="Available" style="txt-align:center;font:bold;size:Bigger;" top="445" width="219"/>

<label height="51" left="643" name="Available" style="txt-align:center;font:bold;size:Bigger;" top="608" width="219"/>

<label height="83" left="480" name="In Use" style="txt-align:center;font:bold;size:Bigger;" top="425" width="148"/>

<label height="83" left="672" name="In Use" style="txt-align:center;font:bold;size:Bigger;" top="429" width="148"/>

<label height="83" left="91" name="In Use" style="txt-align:center;font:bold;size:Bigger;" top="754" width="148"/>

<label height="83" left="485" name="In Use" style="txt-align:center;font:bold;size:Bigger;" top="759" width="148"/>

</abs>

APPENDIX B: The Arduino Code

```
#include <ESP8266WiFi.h>

#include <ESP8266WiFiMulti.h>


#ifndef STASSID

#define STASSID "DESKTOP-9EL40QG 8813"

#define STAPSK  "J45s:245"

#endif


int sensor = 13;

int state = LOW;
```

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

const char* host = "10.192.66.57";
const uint16_t port = 21888;

ESP8266WiFiMulti WiFiMulti;

void setup() {
  Serial.begin(9600);
  pinMode(sensor, INPUT);
  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);

   WiFiClient client;

    if (!client.connect(host, port)) {
    Serial.println("connection failed");
    Serial.println("wait 5 sec...");
    delay(5000);
```

```
  return;
 }
   int state = digitalRead(sensor);
 if(state == HIGH){
  //digitalWrite (Status, HIGH);
  Serial.println("1");
  client.println("1");
  delay(1000);
 }
  else{
  Serial.println("0");
  client.println("0");
  delay(1000);
}
}
```