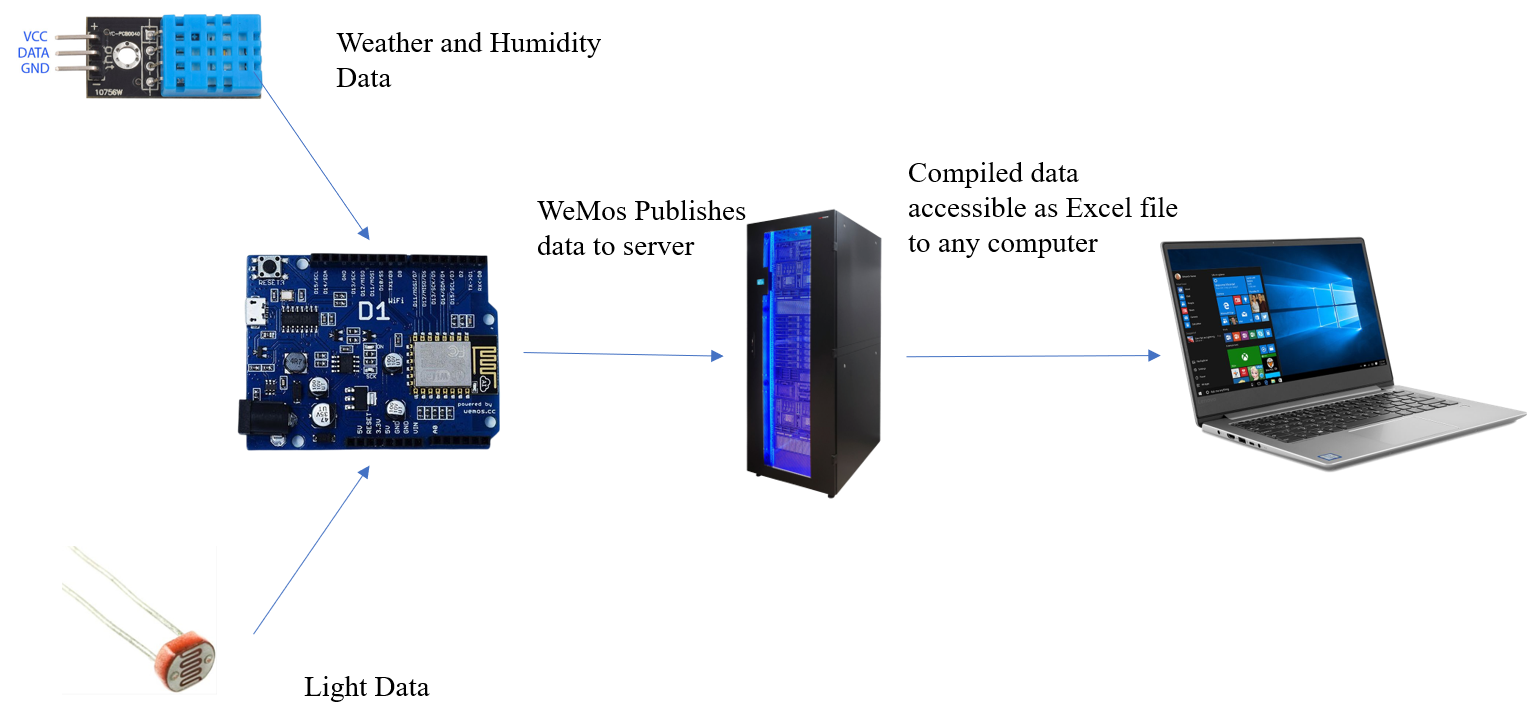
# Introduction

This report outlines the design process of team V3 throughout the process of creating a weather sensing machine that can send live temperature, humidity, and light levels of an environment to a server on it’s local network.

The original goals of the project were to have the team design and deploy a sensor that would measure the weather in its environment and send out the information over the course of a week. The data sent out by the sensor was to be able to be received and manipulated by a Virtual Instrument (VI) made in LabVIEW. To achieve this goal the team was given access to a WeMos D1 R2 Arduino Board, a shield for the board, a DHT11 sensor which measures temperature and humidity data, a mini photocell which acts like a resistor that becomes weaker when exposed to more light, resistors, a small 3D printer, a soldering iron, a small battery, and miscellaneous tools such as wires, wire-cutters, and solder.

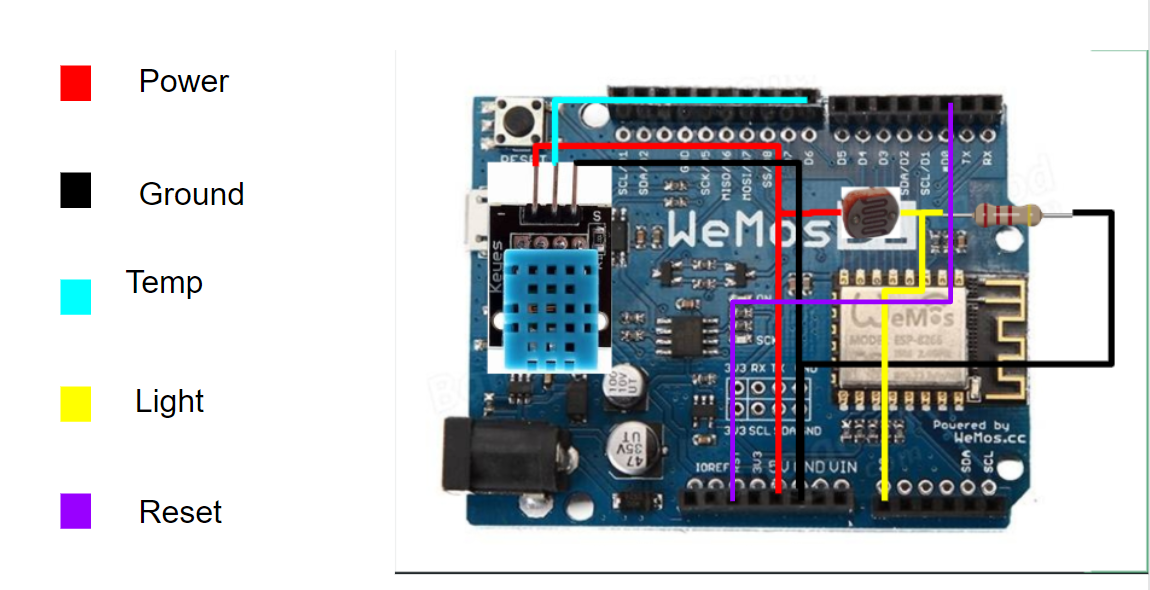
Throughout the process of designing the sensor some of the goals of the project were shifted. The battery didn’t have enough capacity to keep the WeMos board active for a full week so the team was asked to gather at least 4 days of data as opposed to the full week. LabVIEW proved to be too cumbersome for many of the teams assigned to the project to successfully reach the original goals. Because of this the team was given the option to view all the data neatly in excel instead of viewing and manipulating the data in LabVIEW.

# Design

In order to obtain the weather data the DHT11 and photocell needed to be attached to the WeMos board. From there the WeMos publishes the data to the server where it can be accessed by any computer on demand. 

# Electrical Design

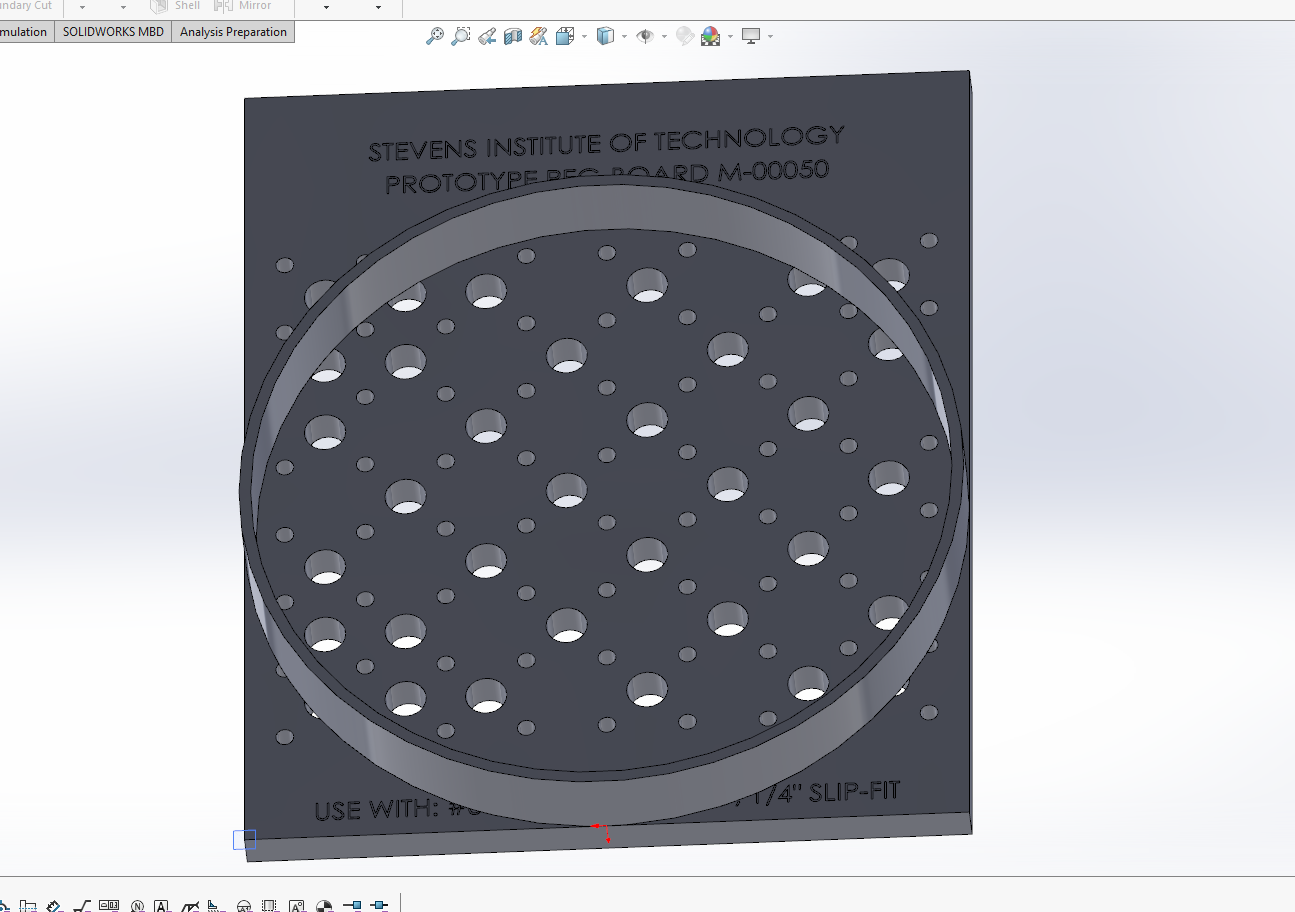
The DHT Requires an input voltage and ground to collect data. This can be provided by the Wemos’s VCC port and ground port. The Data from the DHT can go into one of the data ports of the Wemos board in order for it to get the data. The photocell resistor has a simple setup. It requires voltage as well so it is wired to the VCC of the WeMos board and then the output is wired both to a resisted and directly into the WeMos’s AO port. Lastly one of the ports (in this case D0) must be wired to the reset port on the Wemos board. This is for battery related reasons that will be discussed later in the report. The final wiring schematic is below.



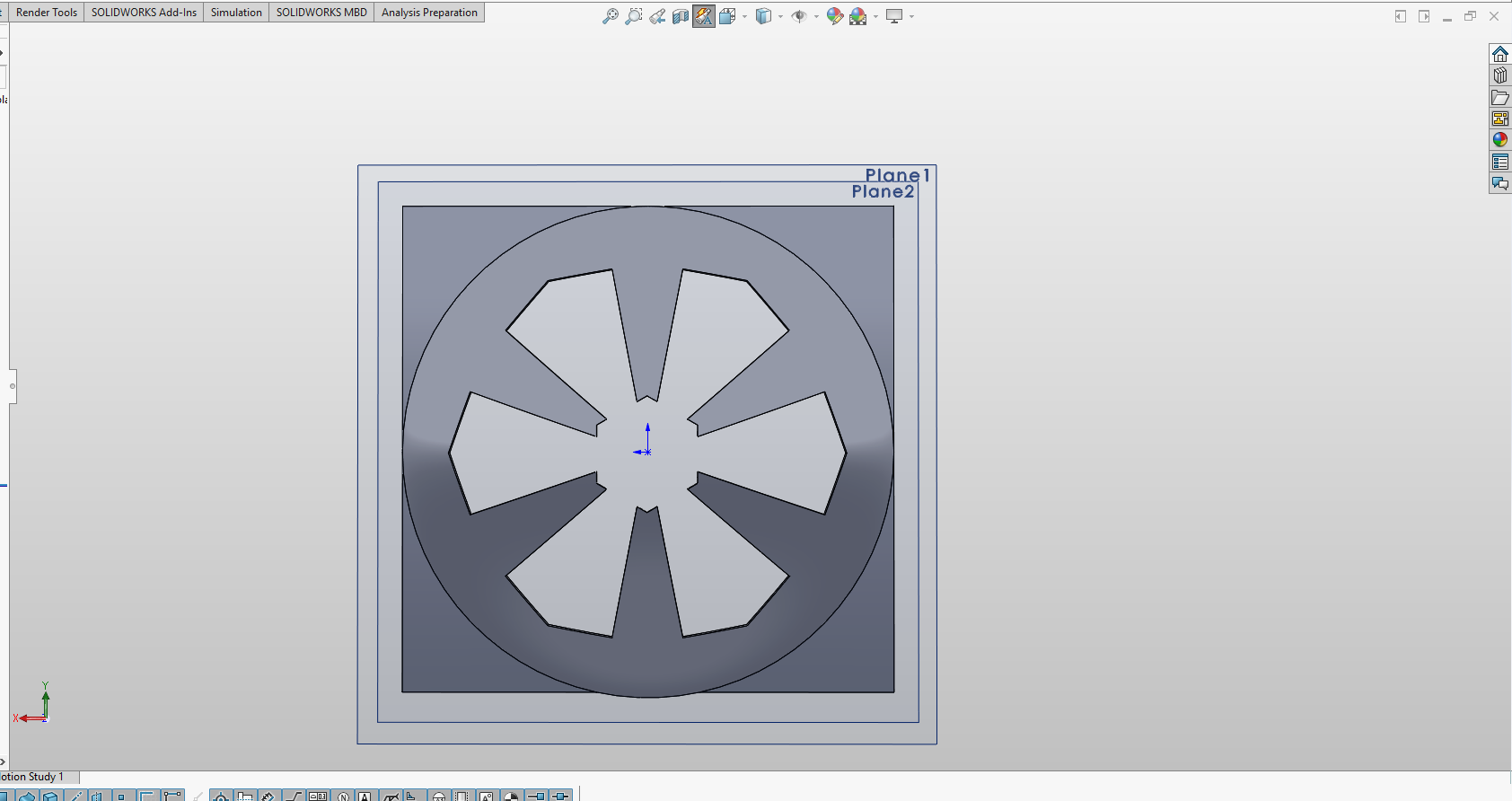
## Mechanical Design

## 

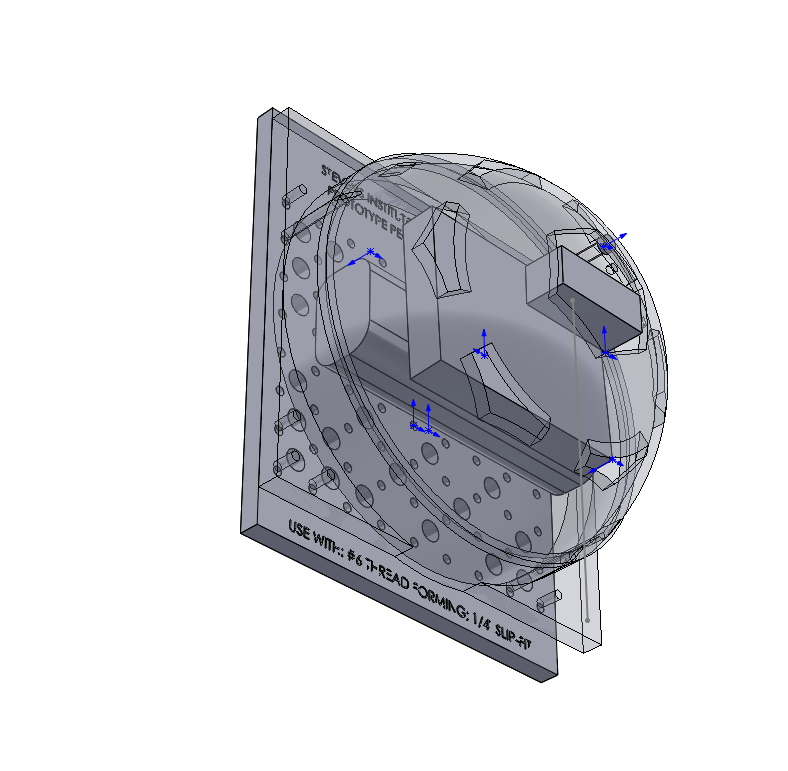
## This was the basis for our original design, we wanted to make a turtle shell that was inspired by this 3-d printed igloo.



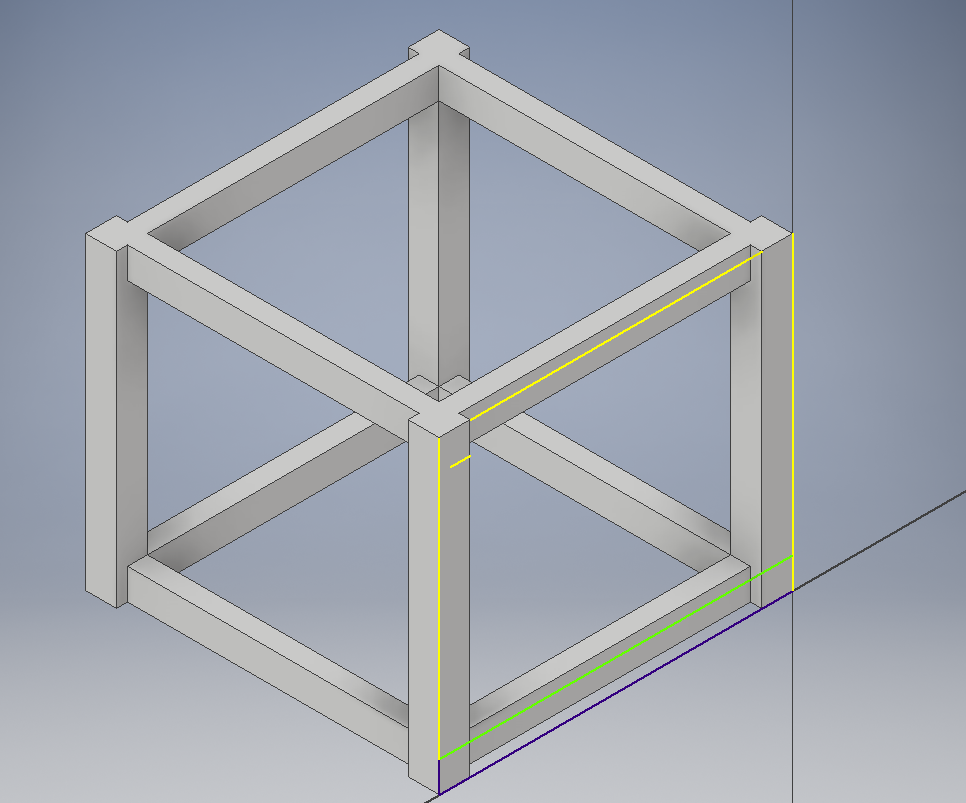
We then started with the idea that the dome would have to fit on the pegboard that we were given at the start of the project, because if it wasn’t the print would take much longer than necessary.



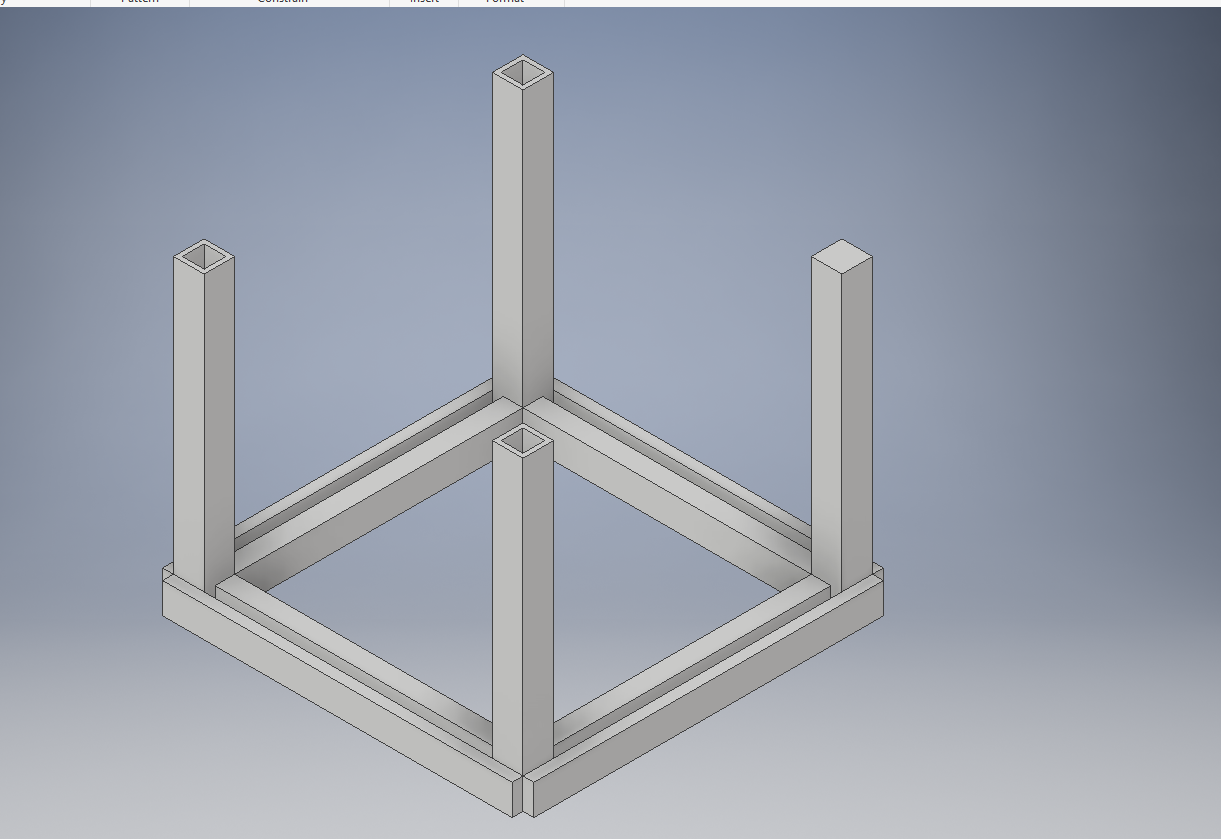
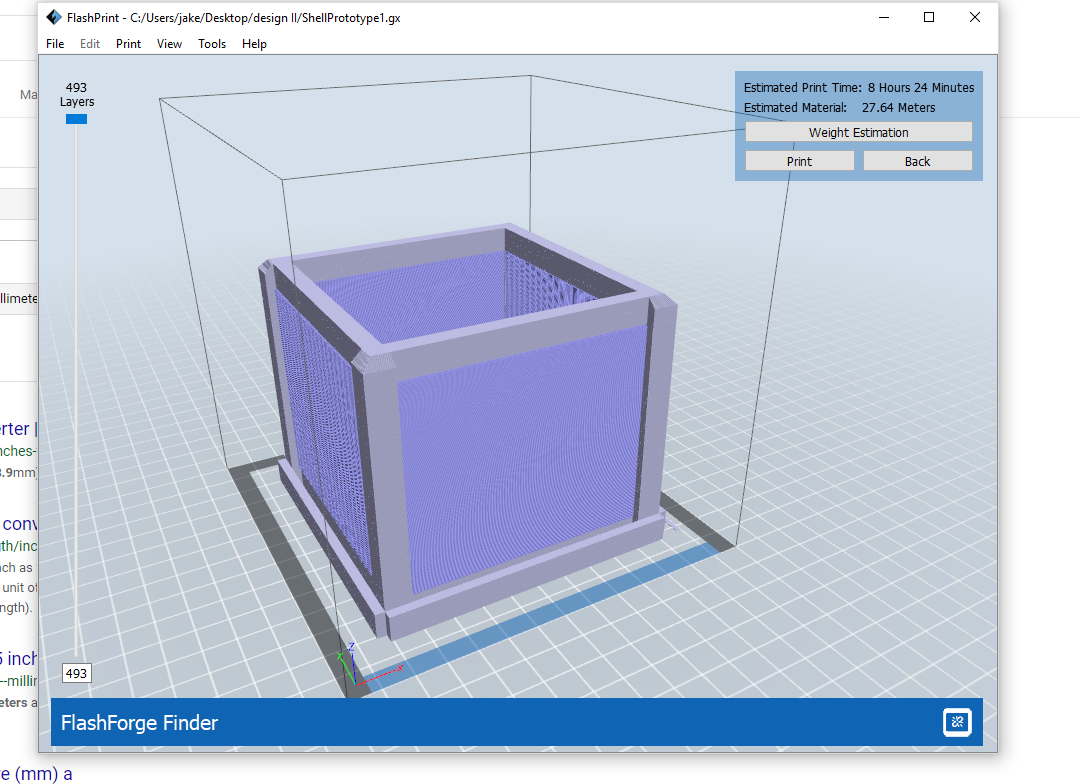
This is a picture during the solidworks prototyping when we were trying to figure out how we could make all of the components fit and work under the enclosure that we were designing. The slots and the hexagonal center were supposed to allow a location for the battery to slide through while also allowing the temperature, humidity, and light to be accessible from within the enclosure. We also cut the holes in a symmetrical pattern so that the dome would still be stiff enough to protect the sensitive components, such as the wemos board and electrical components, if for some reason the container fell.



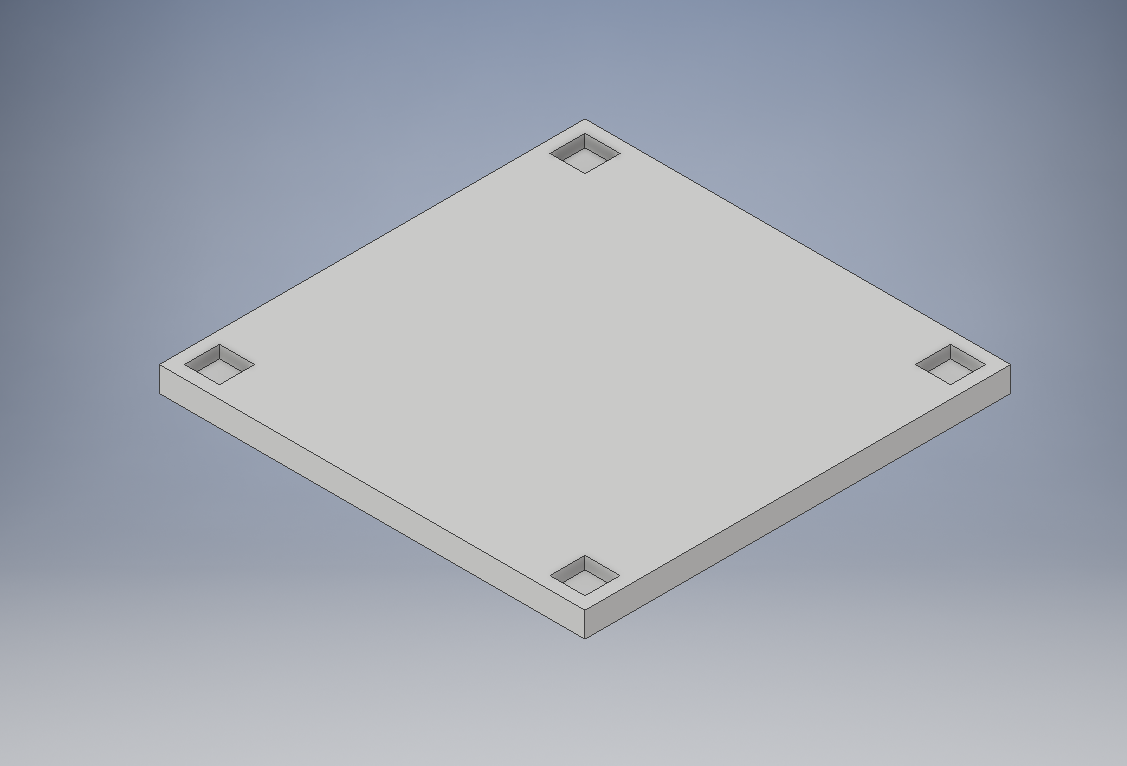
After making more design tweeks to where the platform would be too large for the peg board, the dome would be raised off the board twice what we originally planned, and the new design for the dome, we decided to scrap this idea because we were running out of time and we weren’t close enough to a functional prototype to feel comfortable that we would get the project done by the deadline.



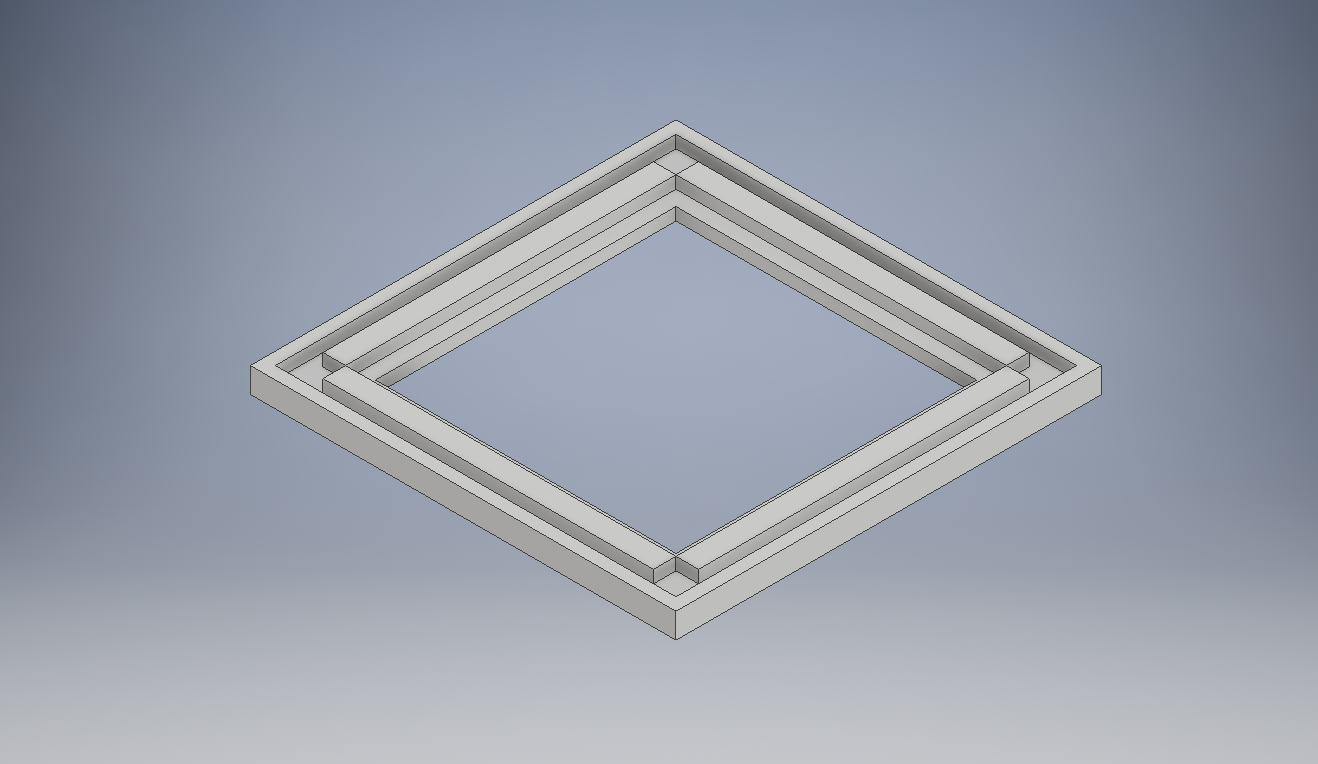
Then we started on a design of a box with plexiglass sides. Originally the box was going to be printed as a whole piece and the top was going to just be a piece of plexiglass that could be removed by unscrewing it. As bad of an idea as that seems now, the only reason why we did not do that was because the print time for the box was 8 hours and 24 minutes long which was way longer than any of our group mates wanted to wait for a piece of plastic to print. As you can see in the picture below it was going to take so long because all of the empty space between the bottom ledge and the top ledge needed to be supported by supports.



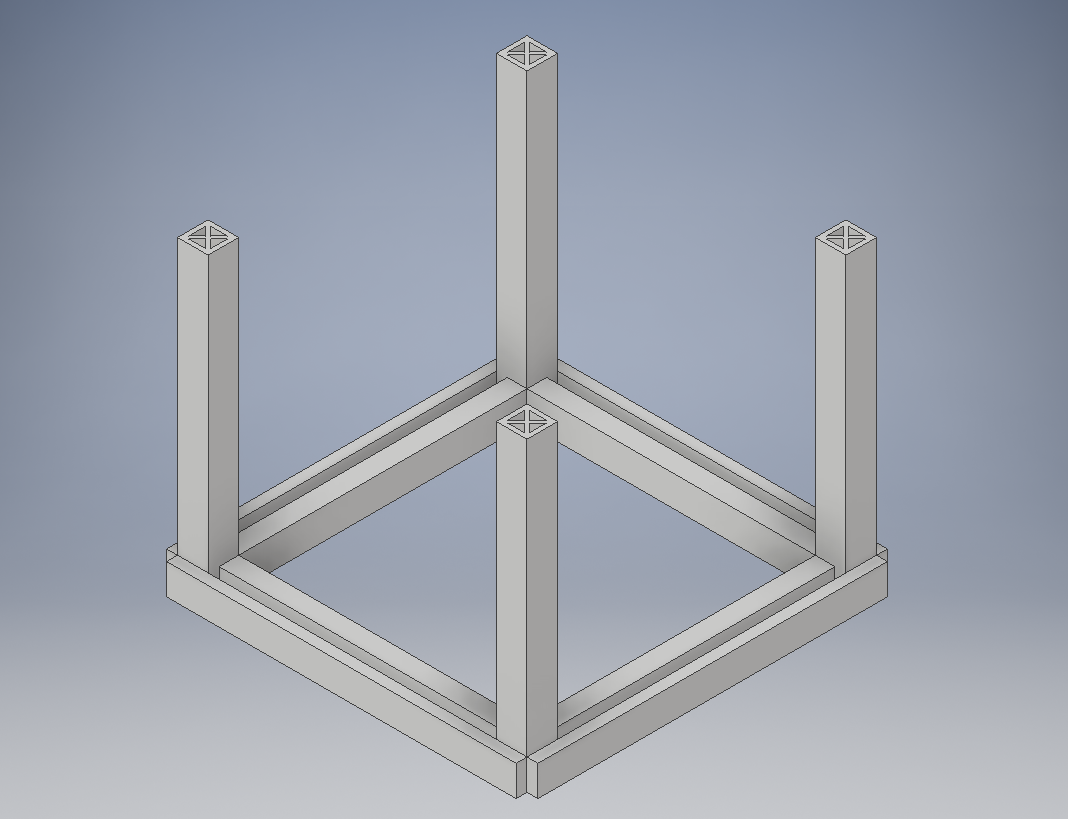
We then concluded that we could make a design that was comprised of two pieces, a bottom with pillars that would connect to a top via a friction fit. We also kept with the idea that all of the sides would have plexiglass so that the light sensor would get reliable readings and that the enclosure would look cool with visible components for everyone who laid eyes on it to see.



The above image is the preliminary design of the top which would friction fit to the pillars on the bottom. This was a rough design to get the dimensions to a point where we could start designing a more elaborate top.



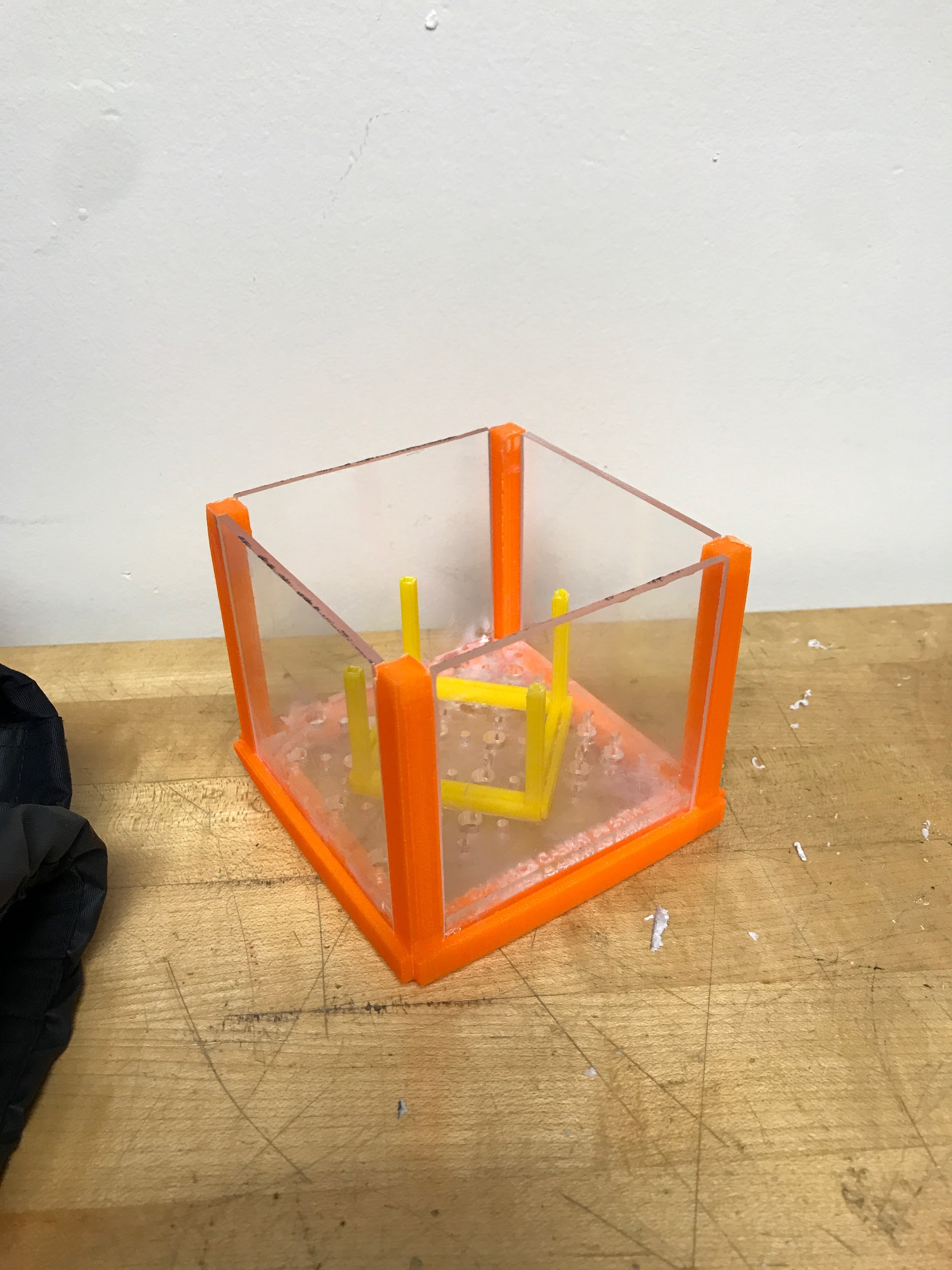
The above image is our final design of the top. The lip around the edge is so that the top not only locks into the pillars but also locks the plexiglass onto the sides. The inner indentation is there so a piece of plexiglass can be super glued on the corners in that indention. We purposely placed the indention on the underside of the top so that the people who would view the enclosure would not see the cuts on the plexiglass because, using the tools that we had it was nearly impossible to achieve clean cuts.



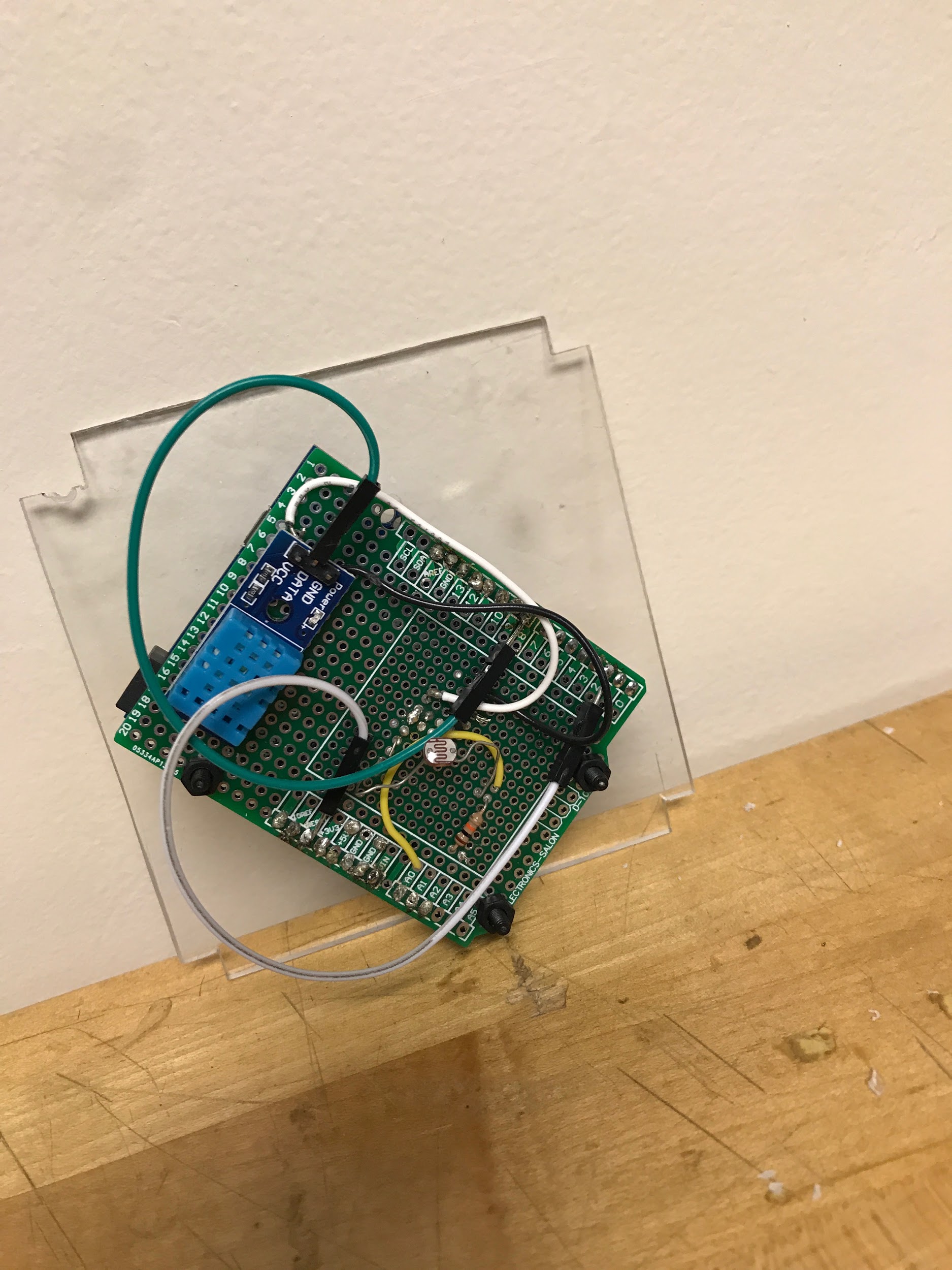
This is the final design of the bottom portion of the enclosure. The slots on the floor portion are to hold the plexiglass in place. The pillars have the x formation in them because in order to get the greatest strength with the least print time a symmetrical triangular design that was designed into the pillars was the best way to go. The print time on the piece with the generated honeycomb structure was the shortest at a time of 3 hours and 45 minutes but also had the weakest strength, the generated triangular structure was the longest print time at a time of 8 hours and 41 minutes, and the self - designed symmetrical triangular design shown above took 4 hours and 33 minutes to print. The corner indents on the base of the bottom piece were there just in case we were going to need a way of mounting this to the pegboard, we were going to have a lockdown mechanism that would fit in those corners and lock onto the top edges of the base allowing a total lock in all directions.



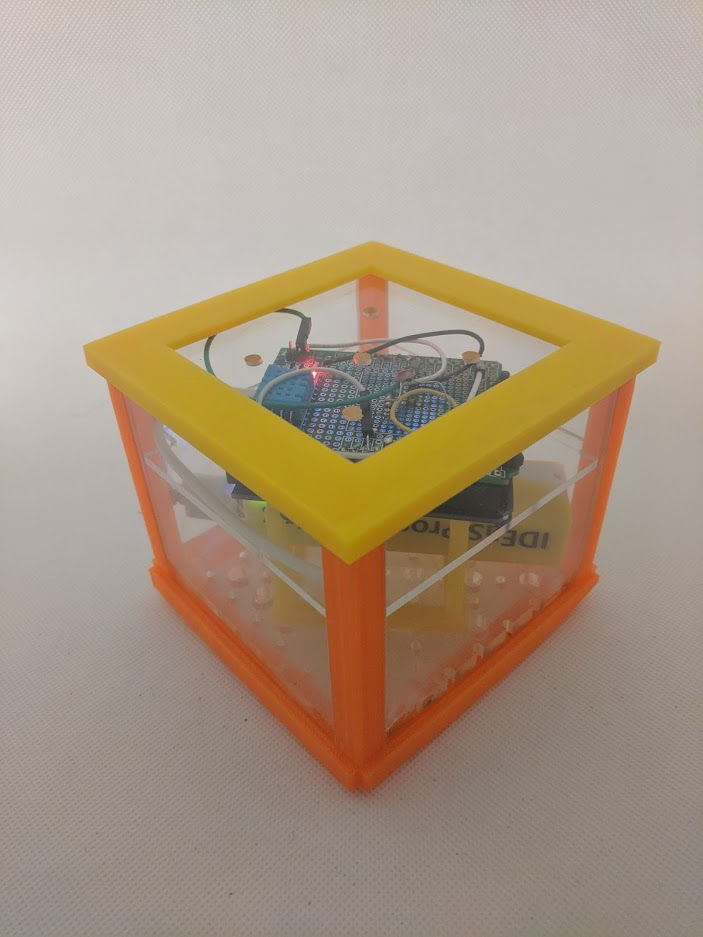
This is a final image of the top with the plexiglass installed. The holes are cut in the plexiglass to allow the temperature and humidity sensor to have accurate readings of the room that the box was deployed in.



This image is the completed bottom portion of the box with some extra components that were added in order to allow all of the electrical components to be in a proper spot so that they could all work properly and would not move if the box fell. We ended up cutting the bottom pegboard into a square with notches in the corners so that it could slide down into the box and sit on the pre-existing ledges so that the battery would have somewhere to sit, and so there would just in general be a floor to add to the rigidity of the piece. We also used our prototype print of the bottom of the box to be a spacer to allow the wemos board and electrical components (shown below) that was placed on a piece of plexiglass, cut in a similar fashion to the pegboard, to sit on top of it so that we had a sturdy platform to mount the components to. Otherwise the board would have just been floating around in a box and if it were dropped all of the components could have broken.



Above is a screenshot of the completed electrical design of the WeMos board wired up to all the components. All wires are soldered down to help with the boards structural integrity. For two of the wires the team used the wires that came with the WeMos boards. One of which is the one that connects the data signal from the DHT11 to the WeMos board. This was done because the pin connector for these wires is built into the DHT11 which made it simpler to just use what was already built in. The second of these wires connects the reset port to the D0 port on the WeMos board. This needs to be disconnected any time new code needs to be uploaded onto the WeMos board so the WeMos wires were used because they are easier to solder and desolder.

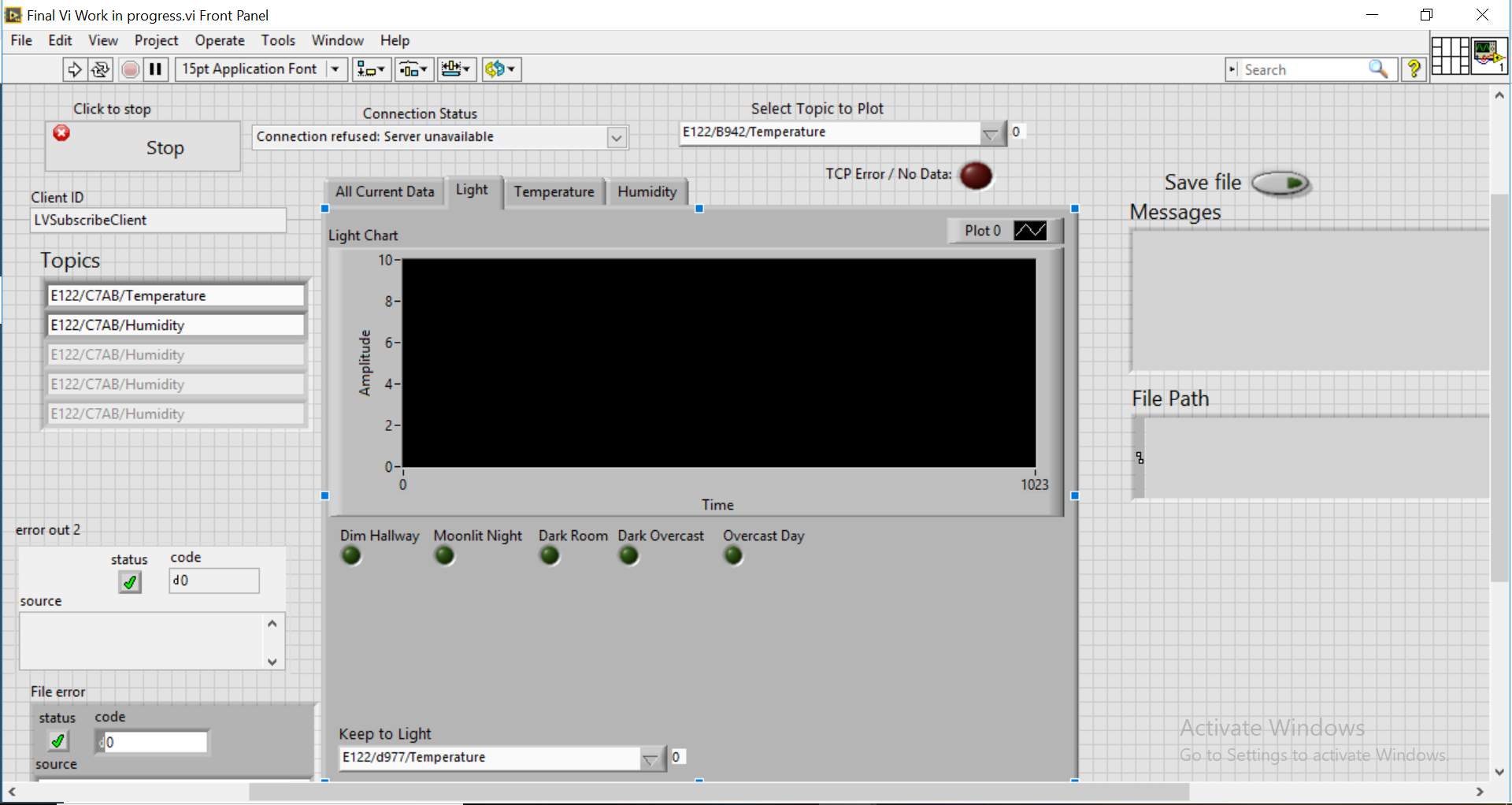


This is the final product of our plexiglass box with all of the components in it including the four plexiglass side panels, the plexiglass top, the plexiglass pegboard, the 3-d printed bottom and top, the wemos, light sensor, humidity and temperature sensor, battery, battery wire, and prototype for the bottom 3d printed piece.

In order to design the original design of the turtle shell I used solidworks, but seeing it was taking a long time and deciding that I should use a program I am more comfortable with as a group we decided that Autocad Inventor was the program that would be used to design our new idea. The box was completely designed within two class periods and completely printed within one more class period.

In order to cut the plexiglass we used the band saw because it was the only saw that had a guide on it which would allow us to get the cleanest cuts out of all of our options.

## Software Design



## This is a screenshot of the VI used to record progress during deployment. We did not use a Labview VI for analysis because we didn’t fully understand the program and were short on time. Instead, we only modified the data output for the subscription VI given to us so that it would display the data of all the sensors

## 

As can be seen above, we only added the old code to display ranges from the other weeks.



Above is the code that determines the pattern that the WeMos “sleeps”. In order to conserve battery, the sensor must go into a low power mode between the times it collects data. The WeMos naps for 12 seconds and then goes into sleep mode for intervals of 10 minutes. Once it wakes up it collects and sends data to the server.

# Deployment & Performance

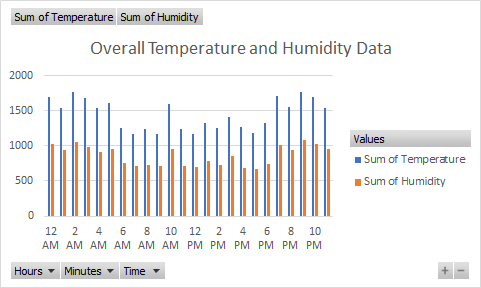
Due to the sensor’s bad battery, the sensor only lasted 3 days and was redeployed the following class for more data. The battery running out was caught by the VI displayed above. Other than the battery issue, there was no other issues with the deployment and performance.

# Environmental Investigation

Due to the placement of our team’s sensor, the experiment performed was to find out who was using room 011 of EAS over the weekend by recording the fluctuating light levels of the room over time. The objective was to find any time the room was in use when it was not supposed to so the information can be sent cv v to the right authorities to catch anyone using the room without permission.

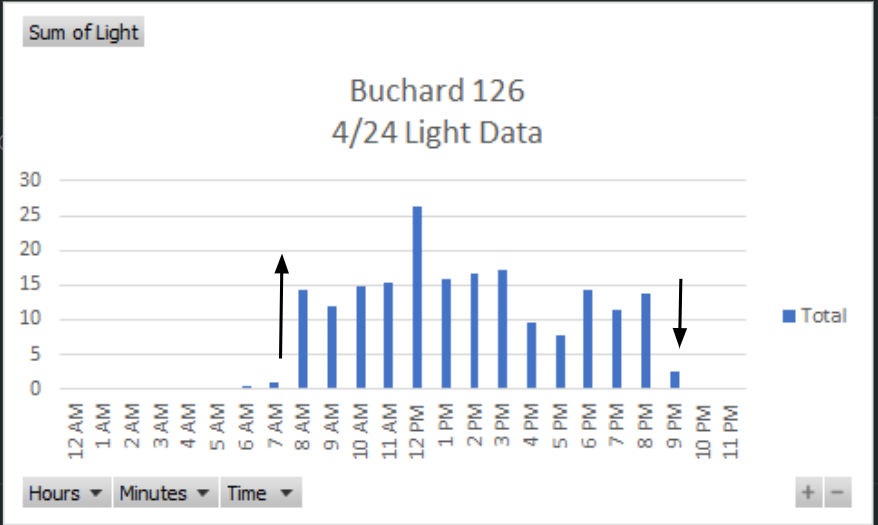
The lights in room 011 are motion activated, meaning any time someone enters the room, the lights go on. No one is able to enter the room without the lights turning on, and the lights only turn on when someone enters the room. Therefore, the light levels of the room indicate whether the room is in use or not. The sensor was placed in the back of the room, meaning any light from the sun that could have affected the light sensor’s results could not reach that far, meaning the only readings taken are from the lights triggered by people walking in the room.

The method of presenting the data is via pivot charts in Excel. Pivot charts are able to present the sums of data received over each hour. Because it is a sum, the precise data of what was recorded every data point is lost. Also, pivot charts take the sum of all data points within a certain hour when creating the chart, meaning any extraneous data will make some values mistakenly higher than other values. Our experiment had extraneous data from between 6 PM to 5 AM. To correct for this, the data is presented by day. Due to the unreliability of the light sensors and the difference in the amount of data points taken by each group’s sensor, the ranges for them are purposefully broad. The only ranges are lights on at above 5 cumulative volts and lights off below 5 cumulative volts. The ranges will be different per sensor looked at as well, being dependent on the highest and lowest values of that sensor.

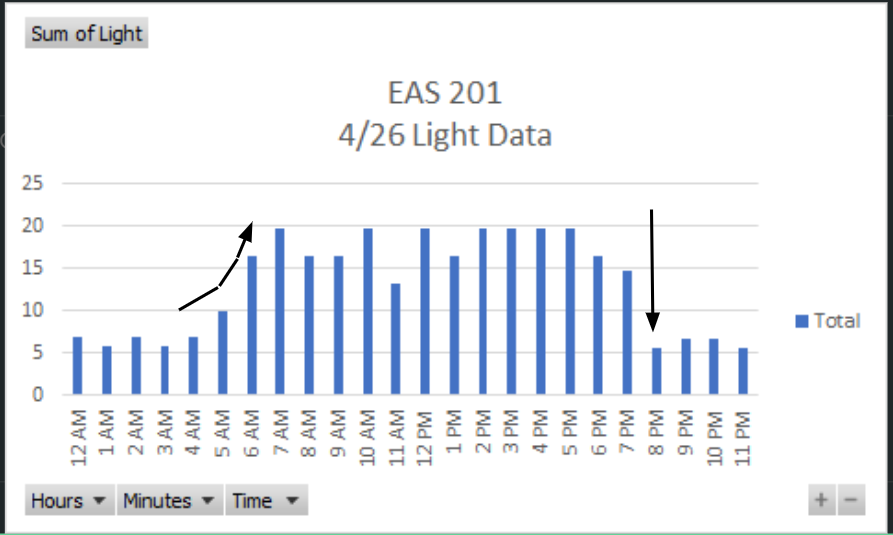


Activity of the room will not be determined using the temperature and humidity data because save for the previously mentioned artificially high readings on the ends and the weird spike at 10 AM, the readings stay consistent, giving no information about the activity of the surroundings. The 10 AM spike may be caused by a failure of the sensors because when a spike in temperature occurs, there is a corresponding decrease in humidity. Because that trend isn’t seen and there is no other information on what may cause the spike, the most likely cause of the spike is instrumentation failure, we think.

To test that activity can be taken to show activity in a room, we can analyze data from other sensors to see if we can measure whether there was anyone in the room.

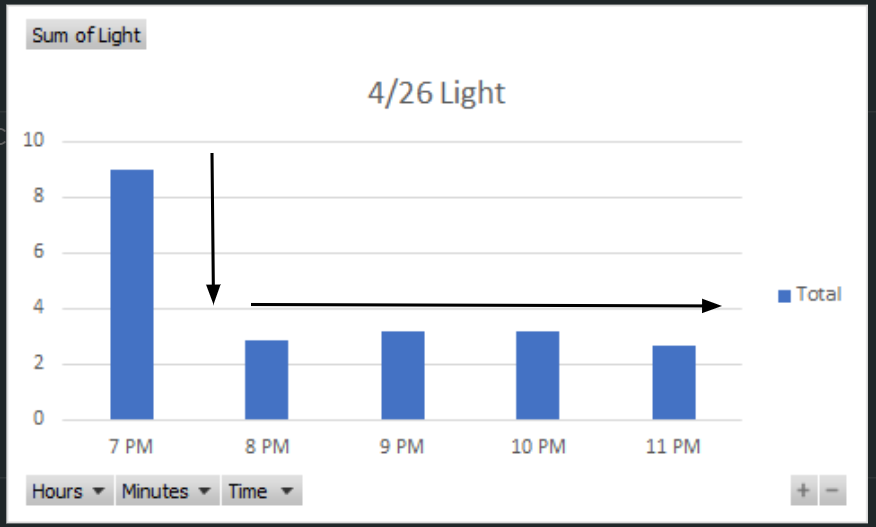


Sunrise on 4/24 was at 6:04 according to Google. Because it recorded barely any data at that point, it can be surmised that the sensor was in a position where it couldn’t get direct sunlight. The small amount of voltage read may have been caused by light refraction putting some light in the room. The sharp increase of voltage at 8 AM is most likely caused by the lights being turned on, which also corresponds to the start of classes at 8 AM, meaning the room was most likely occupied at this time. There is a similar trend at the end of the day at 9 PM, where there is a sharp decrease in voltage, indicative of the lights being turned off. This also corresponds to when the last class ends at 9 PM. The spike may be cause by the light from the sun coming in at the exact right angle.

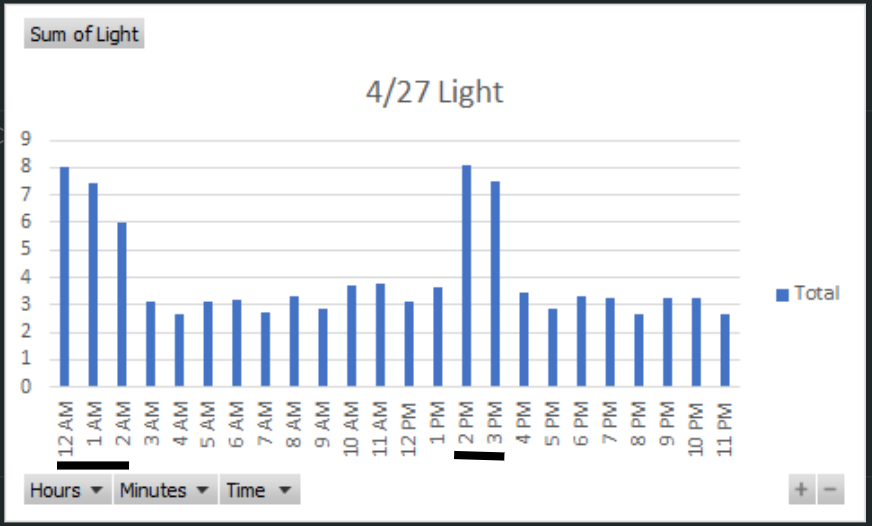


For this sensor, the gradual increase in voltage at the beginning could be indicative of the sun gradually reaching around a corner of a window or the like as it rises and moves across the sky. Because the light fluctuates a lot, we can guess that the sensor was placed at a window where the sun could affect readings. For example, sunset was at 7:47 PM, which could explain why there is a small decrease in voltage at that time as compared to the rest of the data. Then, there is a somewhat consistent reading from 7 AM to 7 PM which could be when the lights are on. There is a sharp decrease in voltage at 8 PM, which is indicative of the lights being turned off. This shows at least that there were people to turn off the lights at 8 PM. It gives less information than the other sensors, but it is still something.

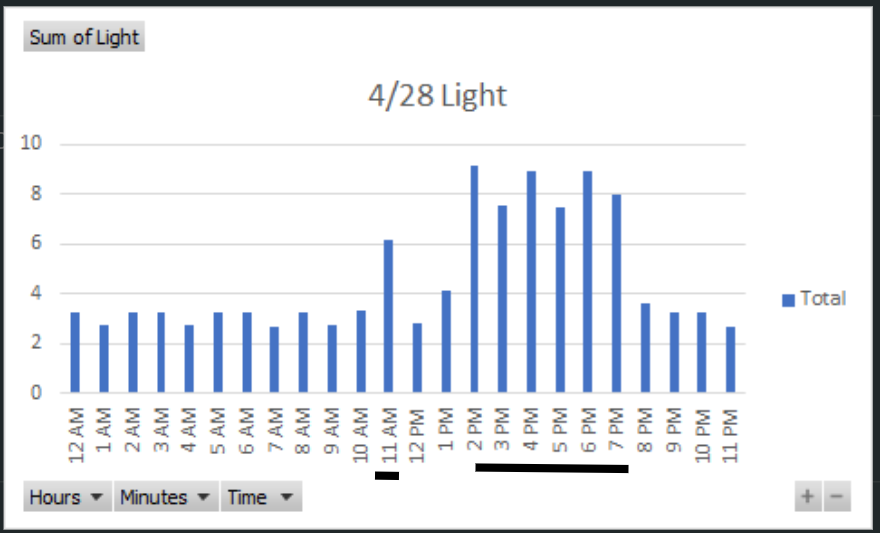
Now we will analyze the data received from our sensors. This section will be split up by day to analyze each of the fluctuations of the light by day and to preserve the relative accuracy of the data. Our data with our pivot charts will be provided for more in depth data if is needed.



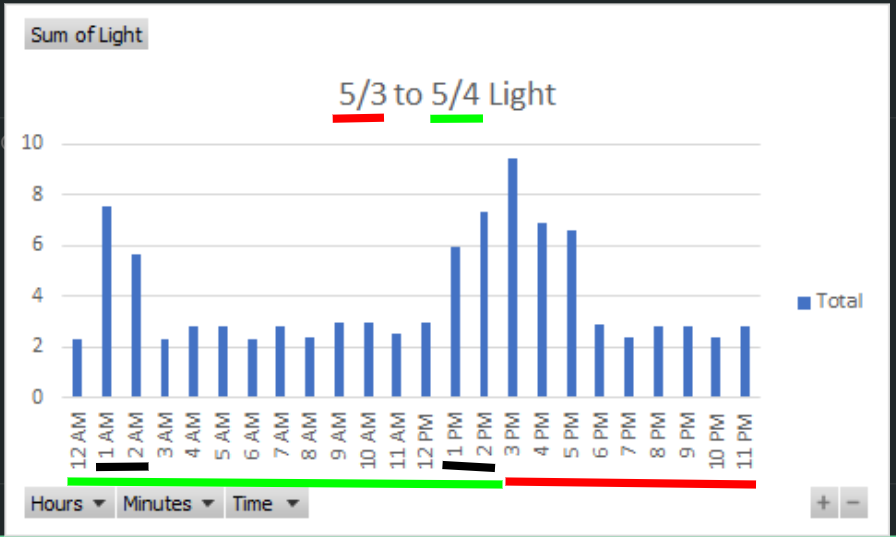
This is the Friday our sensor was deployed. The data before this is unusable because the sensor was being tested. As you can see, there is a sharp decrease in light levels at 8 PM, which is presumably when the last person left the room and the lights turned off. There are general ranges for on and off, barring any irregularities. The general ranges are between 2 and 4 cumulative volts for lights off and between 8 and 10 cumulative volts for lights on. However, spikes and sudden decreases can happen, which is why the aforementioned range of above or below 5 cumulative volts is used.



This data set is on a Saturday. As seen above, there are spikes at 12 AM to 2 AM and from 2 PM to 3 PM. This means that someone was in the room at these times. It is possible that someone came into the room during these times to 3d print something, as the 2-3 hour period is enough to print one or two big parts or more depending on how many machines used. However, there is a possibility that these are the times the janitor comes in to clean. Since we have data on another weekend, the rest of the analysis will be done then. If a similar pattern to this is seen, then it is more likely that the janitor just comes in at these times. If not, then the person in the room at those times is most likely not a janitor and could be a student using the machines



This data set is on a Sunday. There is a small anomaly at 11 AM. Since the cumulative voltage is smaller than normal, it means the lights were only on for a portion of the hour. It is possible that someone just walked passed the room to reach somewhere further in and triggered the lights. The lights take a while to turn off, but not an hour, which corroborates the idea that someone just walked by. The other times when the lights are on are between 2 PM and 7 PM which is a 5 hour period. Our sensor did not record another sunday, so it is unclear whether this is a normal occurrence or not. However, it is unlikely that it is a janitor because it it very unlikely that a janitor takes 5 hours to clean a room. There could be a class or meeting at this time, but we think that is also unlikely because it is a Sunday and it is unlikely that the school would schedule anything on a sunday. It is possible that someone came in to 3d print a 5-hour part during non school hours, or the person came to use the other tools and machinery for a weekend project. Overall, it is unclear on what caused that 5 hour period, but it could be a student using the room when they shouldn’t be. However, further investigation using other means would be required, because this isn’t enough data to make a complete conclusion.



This data set was taken on a Friday and a Saturday. The red portions are Friday and the green portions are Saturday. At the beginning of the red portion on Friday, the lights are on because of the Engineering design class we are a part of. It decreases as it should be at 5 when class ends and the last person leaves. On Saturday, there is activity on 1 to 2 AM and 1 AM to beyond. The activity at 1 AM and 1 PM are around the same time as the previous Saturday, indicating that this is most likely when the janitor comes in.

Overall, we found out when the janitor most likely comes in on saturday, and some potential points of interest on other times. Further investigation must be taken to find out who was actually there at those times, but this is a good start and hopefully helps in catching who is using the room without permission.

# Attachments

The following is the code used on the WeMos Board:

/\*

E122 - Baseline Wemos Firmware - Version 0.3-

Updated: April 1 2019

Integrated from components picked off the net by Prof.KP.

This version serves as starting point for integration of Low Power Mode

\*\*\*\*\*\*\*\*/

/\* Installation of drivers and other set up needed - Check CANVAS...

To install the ESP8266 board, (using Arduino 1.6.4+):

- Add the following 3rd party board manager under "File -> Preferences -> Additional Boards Manager URLs":

http://arduino.esp8266.com/stable/package\_esp8266com\_index.json

- Open the "Tools -> Board -> Board Manager" and click install for the ESP8266"

- Select your ESP8266 in "Tools -> Board"

\*/

#include <ESP8266WiFi.h>

#include "PubSubClient.h"

#include "info.h"

#include "WiFiManager.h"

#include "DHT.h"

#include "WeMosSleep.h" //include sleep function library

//The ESP8266 recognizes different pins than what is labelled on the WeMos D1

#if defined(d1) //Defines Wemos D1 R1 pins to GPIO pins

#define D0 3

#define D1 1

#define D2 16

#define D8 0

#define D9 2

#define D5 14

#define D6 12

#define D7 13

#define D10 15

#endif

#if defined(d1\_mini) //Defines Wemos D1 R2 pins to GPIO pins

#define A0 0

#define D0 16

#define D1 5

#define D2 4

#define D3 0

#define D4 2

#define D5 14

#define D6 12

#define D7 13

#define D8 15

#endif

//Set up the DHT11 (temperature/humidity sensor)

#define DHTPIN D6 //PLUG THE DHT 11 ONLY INTO D5, D6, D7 on either D1-R1 or D1-R2 Board.

// DO NOT PLUG INTO OTHERS AS THE MAPPING IS NOT THE SAME. DHT11 WILL BURN.

#define DHTTYPE DHT11

// CONFIGURATION SETTINGS ....BEGIN

//Wifi Settings

//const char\* ssid = "DLabsPrivate1";

//const char\* password = "L3tsM@keSometh1n";

const char\* ssid = "Stevens-Media";

const char\* password = "Stevens1870";

//MQTT Settings

const char\* mqtt\_server = "155.246.18.226";

const char\* MQusername = "jojo";

const char\* MQpassword = "hereboy";

//MQTT Publish Topics

//XXXX will automatically be replaced by last 4 digits of MAC address

char\* MQtopic1 = "E122/B942/Temperature";

char\* MQtopic2 = "E122/B942/Humidity";

char\* MQtopic3 = "E122/B942/Light";

//Note that since this is a real Wemos board -- it runs forever as opposed to

// the fakemos -- http://www.dmi.stevens.edu/fakemos/

//create class instance for DHT-11 temperature/humidity sensor

DHT dht(DHTPIN,DHTTYPE);

//create class instance for WiFi connection

WiFiClient espClient;

WeMosSleep sleep;

//create class instance to retrieve MAC address

info board\_info;

//create class instance for MQTT Client

PubSubClient client(espClient);

//declare variables

char msg1[20],msg2[20],msg3[20]; //value strings for temperature, humidity and light

float temp, hum, light; //values for temperature, humidity and light

//controls connections

bool connectedToWiFi = false;

bool connectedToMqtt = false;

int InvalidCtr = 0;

void setup\_wifi() {

delay(10);

// We start by connecting to a WiFi network

Serial.println();

Serial.print("Connecting to "); Serial.println(ssid);

//connect to WiFi

WiFi.begin(ssid, password);

//output a "." every 500ms until connection established

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

connectedToWiFi = true;

randomSeed(micros()); //insure random call later on is random

Serial.println(""); Serial.println("WiFi connected");

//output IP address assigned by WiFi access point/router

Serial.print("IP address: "); Serial.println(WiFi.localIP());

}

//If a MQTT topic is subscribed to this function will be invoked when a topic

// is received from the MQTT server.

//For now the LED can be turned on and off

void callback(char\* topic, byte\* payload, unsigned int length) {

//print topic and payload(value)

Serial.print("Message arrived [");

Serial.print(topic);

Serial.print("] ");

for (int i = 0; i < length; i++) {

Serial.print((char)payload[i]);

}

Serial.println();

// Switch on the LED if an 1 was received as first character

if ((char)payload[0] == '1') {

digitalWrite(BUILTIN\_LED, LOW); // Turn the LED on (Note that LOW is the voltage level

// but actually the LED is on; this is because

// it is acive low on the ESP-01)

} else {

digitalWrite(BUILTIN\_LED, HIGH); // Turn the LED off by making the voltage HIGH

}

}

void mqttConnect() {

// Loop until we're reconnected

while (!client.connected()) {

Serial.print("Attempting MQTT connection...");

// Create a random client ID

String clientId = "ESP8266Client-";

clientId += String(random(0xffff), HEX);

// Attempt to connect to MQTT server

if (client.connect(clientId.c\_str(),MQusername,MQpassword)) {

Serial.println("connected");

// Once connected, publish an announcement...

//client.publish(MQtopic1, "00000");

// ... and resubscribe ---- Dont subscribe KP

// client.subscribe("inTopic");

// #KP - No announments --- No Subscribes.

} else {

Serial.print("failed, rc=");

Serial.print(client.state());

Serial.println(" try again in 5 seconds");

// Wait 5 seconds before retrying

delay(5000);

}

}

connectedToMqtt = true;

}

//replace XXXX in MQTT topic with last 4 digits of MAC address

void insertMAC(char\* Topic){

int posX = 0;

int strLen = strlen(Topic); //get length of topic

//find first X in topic string, ie "E122/XXXX/Temperature"

for(int i=0; i<strLen; i++){

if(Topic[i]=='X'){posX = i; break;}

}

if(posX > 0){ //if 'X' found

//replace XXXX with last 4 digits of MAC address

Topic[posX]=board\_info.mac()[12]; //00:00:00:00:x0:00

Topic[posX+1]=board\_info.mac()[13]; //00:00:00:00:0x:00

Topic[posX+2]=board\_info.mac()[15]; //00:00:00:00:00:x0

Topic[posX+3]=board\_info.mac()[16]; //00:00:00:00:00:0x

}

}

//any code in this function runs once at startup

void setup() {

pinMode(BUILTIN\_LED, OUTPUT); // Initialize the BUILTIN\_LED pin as an output

Serial.begin(115200); //serial monitor must be set to this baud rate

while (!Serial)

{

};

Serial.println("");

Serial.println("Wemos POWERING UP ......... ");

sleep.setNapSeconds(12); // sets Nap time to 12 seconds

sleep.setSleepMinutes(10);// sets Sleep Time to 15 minutes

//if it is time to wake, the code after the following function will run, otherwise system will go to sleep

sleep.checkWake();

Serial.print("Mac Address:"); Serial.println(board\_info.mac());

insertMAC(MQtopic1); //replace "XXXX" with last 4 digits of MAC address

insertMAC(MQtopic2); //replace "XXXX" with last 4 digits of MAC address

insertMAC(MQtopic3); //replace "XXXX" with last 4 digits of MAC address

if(connectedToWiFi == false){

setup\_wifi(); //start WiFi

}

dht.begin(); //initialize DHT-11 temperature/humidity sensor

delay(100);

client.setServer(mqtt\_server, 1883); //MQTT server IP address and port

client.setCallback(callback); //specify function for MQTT subscribe callbacks

}

//code in this function runs repeatedly

void loop()

{ //start loop

if(connectedToMqtt == false)

{

//check if connected to MQTT server, if not try to reconnect

if (!client.connected())

{

mqttConnect();

}

//run MQTT tasks

client.loop();

}

readSensors(); //read temperature, humidity and light sensors

// put any additional code you wish to run repeatedly here

} //end loop

void readSensors(){

//read temperature from DHT-11 sensor, true = return Farenheit

temp = dht.readTemperature(true);

//read humidity from DHT-11 sensor

hum = dht.readHumidity();

//if(temp > 1000) InvalidCtr++;

//Serial.print("InvalidCtr="); Serial.println(InvalidCtr);

//read light sensor at A0 connection, value will be 0 to 1023

//(float) is a typecast to convert the returned integer to a float

light = (float) analogRead(A0);

//convert 0 to 1023 value to voltage

light = light/310; //ADC value max / Vmax = 1024/3.3 = 310

//convert values to strings

// 20 specifies max string size

// %d specifies an integer (non-fraction) conversion

// (int) is a "type cast" to convert a float to an integer

snprintf (msg1, 20, "%d", (int) temp);

snprintf (msg2, 20, "%d", (int) hum);

snprintf (msg3, 20, "%5.3f", light); //5.3 for "0.000" format

//output MQTT Topic and Message to serial monitor for diagnostics

Serial.print("Published :" ); Serial.print(MQtopic1);

Serial.print(" with value: " ); Serial.println(msg1);

client.publish(MQtopic1, msg1); //publish temperature to MQTT server

Serial.print("Published :" ); Serial.print(MQtopic2);

Serial.print(" with value: " ); Serial.println(msg2);

client.publish(MQtopic2, msg2); //publish humidity to MQTT server

Serial.print("Published :" ); Serial.print(MQtopic3);

Serial.print(" with value: " ); Serial.println(msg3);

client.publish(MQtopic3, msg3); //publish light voltage to MQTT server

Serial.println("");

delay(2000); //wait to publish data again

sleep.sleep();

}