Room Occupancy Sensor Team Final Report

EK210 Introduction to Engineering Design

Summer Term 1 2023

Winston Liu
Anna Tretiakova
Raphael Mok

Table Of Contents

Executive Summary

The purpose of this project is to design a device that can keep track of the number of people at all times currently in a room, display that value as people enter or leave and display the status of the room capacity using a color code. This was achieved by interfacing a set of IR proximity sensors and a 16x16 LED grid with an Arduino Uno R3. Product prototypes were modeled in Autodesk Inventor, with pieces 3D printed and assembled as well as soldered together by hand. The final product achieves most of the design objectives: It detects, stores, and displays the room occupancy, and warns users when the occupancy is reached using an arduino and sensors to detect motion through a door frame. The product was also made as user friendly as possible; there is a rotary encoder which can be used to alter the maximum occupancy of the room. In a test of 100 walk-through trials mounted at 5' off the ground, testing, the product did not make any mistakes when entering and exiting the room, ensuring consistent quality in the user's experience. All but one of the design objectives were achieved, and the objective that failed was eventually deemed unnecessary.

Introduction

Problem Statement: The purpose of this project is to design a device that can keep track of the number of people at all times currently in a room, display that value as people enter and display the status of the room capacity using a color code.

Despite the world moving on slowly from the COVID-19 pandemic, there are still many uses for room-occupancy monitors. For example, fire safety protocols may dictate that the number of people in a room may not exceed a certain number. In this project, we were approached by Prof. Ousama Amar and asked to design such an occupancy monitor. Important constraints issued by the client required a green display for below occupancy, and the product was to switch to red when the occupancy was reached or exceeded. Once at capacity, the product should display by how much room occupancy is exceeded. Furthermore, the product was required to protrude no more than two inches from the wall to which it was mounted. Most of the A detailed table of objectives, metrics, and constraints is available in **Table 1**.

Elements of Design

Several innovative elements were incorporated into the design of this product. The product is contained in a single compact package, allowing for easy 1-step installation and eliminating detector alignment issues. This was accomplished by using IR distance sensors instead of beam break sensors. We initially considered beam-break sensors, but eventually deemed that the added complexity to installation to be unacceptable. Our product contains two GP2Y0A02YK0F infrared proximity sensors with a range of 150 cm, retaining the single-package design without compromising function. To properly range-find the door, the product waits until the door frame is in a reasonable range of approximately 2 meters so as to not take a reading before the sensor is fully set up.After this, it detects what the size of the door is and subtracts 3 centimeters from it as to not trip from minor discrepancies in the detection, and sets that to be the size of the door automatically. The display consists of a 16x16 grid of WS2812b LEDs, which along with the Adafruit neopixel arduino library allowed us to efficiently control the display without sacrificing speed. Although the product features a port to interface with the arduino, users can adjust the room occupancy limit with an encoder, meaning no experience with electronics is necessary to operate the device. Internally, all components are soldered to an arduino shield, decreasing the likelihood of disconnections when jostled and ensuring that electrical components maintain a good connection to power and the arduino. The external structure features permanent mounting points on a lightweight 3d-printed frame, and the one-piece design of the product also supports the use of reclosable fasteners, which do not offer the same precision as permanent fasteners.

Final Product Description

The final product consists of two GP2Y0A02YK0F infrared proximity sensors, a 16x16 LED screen with a 1/16th inch protector screen, and an Arduino Uno R3, housed by a two-piece 3d-printed structural assembly. The enclosure has dimensions 6.75x7.625x3.25 in, and has ¼ in. thick walls and features mounting points for other components with ⅝ in. ANC 4-40 machine screws. Mounting to the door may be achieved by screwing the product to the doorframe using the mounting points, or alternatively through adhesive or reclosable fasteners. A CAD model of the final assembly is depicted in **Figure 11**.

The product features a standard 9mm 5V DC power jack, which allows it to be powered by a variety of wall adapters, and the housing allows access to the Arduino's USB port. The wiring from the sensors to the arduino consists of 22 AWG copper wire, and the wiring to the main power supply consists of 20 AWG copper wire all soldered to an arduino shield.

When the product is switched on, the distance to the door is automatically determined by reading a baseline detection level on the sensors for 20 seconds. The initial display is in "set mode," where the user may select the desired occupancy using the encoder on top of the product. When the encoder button is pressed, the product goes into "operate" mode, and functions as a room occupancy monitor. If the outer sensor is tripped first, the product adds one to the occupancy, and if the inner sensor is tripped first, the product subtracts one. The maximum room occupancy is 99, and the product will not go below zero occupancy. The operational flowchart and full code can be found in **Figures 4** and **5**.

Evaluation of Results

A series of tests were conducted on the final prototype to determine whether the key objectives were successfully met. The product meets all client-specific requirements: It is able to detect and store the number of people in a room, compare it to the set room occupancy, and display to the user whether the room occupancy has been reached, and exceeds expectations by alerting users to how much the occupancy has been exceeded. A sampling interval of 25ms is adequate to accurately detect users at reasonable movement speeds (i.e. not sprinting), and a multi-color display provides accessibility to individuals with color-deficient vision. During testing, it was found to be unfeasible to house a backup battery, due to the complexity of maintaining a reasonable battery lifespan and potential fire safety hazards. A detailed analysis addressing every objective can be found in **Table 1**, though testing revealed that the most important objectives were met or exceeded.

Lessons Learned

The design and assembly of this product has taught our group many important engineering and collaborative skills.

From a technical standpoint, the project allowed us to test and interface with various components, especially light-involved components and sensors. Due to the presence of a relatively high-current-draw LED array, new circuit designs were explored where the arduino was not the main power source for the various sensor and display components. In general, the open-ended nature of the physical design constraints allowed us the freedom to explore our ideas and test them.

From a logistical standpoint, the increased complexity of this project necessitated a greater amount of organization. As a group we learned to construct various organizational structures (gantt charts, function diagrams, PCCs etc.) to ensure that the project stayed on track. Effective communication between those with different engineering backgrounds was also necessary on this front, and the project allowed us to have an experience working together with new people and constructing an effective organizational structure to accomplish our goals.

References

Last Minute Engineers, 2023, “How Rotary Encoder Works and Interface It with Arduino”, https://lastminuteengineers.com/rotary-encoder-arduino-tutorial/

Alex Reid, July 30 2019, “How to control RGB LED matrix with an Arduino”, Reid Projects, https://reid-projects.com/how-to-control-rgb-led-matrix-with-an-arduino/

Akylanator, Autodesk Instructables, “Led Matrix 16x16”, https://www.instructables.com/Led-Matrix-16x16/

Sharp, Reference sheet for GP2Y0A02YK0F Infrared Proximity Sensor Detect 20-150cm Distance Long Range, https://www.sparkfun.com/datasheets/Sensors/Infrared/gp2y0a02yk_e.pdf

Worldsemi, “Intelligent control LED integrated light source”, Reference sheet for the 16x16 LED matrix, https://cdn-shop.adafruit.com/datasheets/WS2812B.pdf

Prof. Ousama Amar, general guidance

Prof. Pavan Bhavsar, general guidance

Edward Park, general guidance

Katie Kelso, 3D printing and assembly

Appendix

Table 1: Objectives, Metrics, and Analysis

Objectives:	**Metrics:**	**Analysis Of Final Design**
Count	Whether the number is correct 0-99	Product is able to count and store the number of people. **Met requirements**
Accurate	# of people counted is accurate	Product is accurate in walking to jogging speeds, and with 5 cm distance between users. **Met requirements**
Work quickly	Occupancy count should go up or down in less than 1 second.	Number updates within 40ms of a person tripping the second sensor. **Met requirements**
Display the room occupancy	Letters less than 2 cm and greater than 1 cm in height.	Display has multiple modes with color codes. **Exceeded requirements**
Affordable	Cost is less than $200	Estimated product prototype cost was $**136.55** for a from-scratch build. **Met requirements**
Retrofittable	Should not take > 30 min to set up	Product is able to be mounted on a door in minutes using either screws, adhesive, or reclosable fasteners. **Exceeded requirements**
Color code that matches the room occupancy	Have three LEDs of different colors and each has to be in the smartphone brightness range, which is 0.6 to 2.1 lux.	Product displays green for under, red for at, and +X in red for exceeding occupancy. Product also displays purple

		for setting occupancy. Brightness meets lux requirements. **Exceeded requirements**
Light in weight	Less than 700g	Product weighs 263 grams. **Met requirements**
Compact	Less than 20cm x 20cm x 20cm	Product is 17.15cm x17.15cm x 8.25cm. **Met requirements**
Battery operated	Battery that can last 4-5 hours if not connected to an outlet	Product is powered by an outlet only due to fire safety hazards. **Failed to meet requirements**
Durable	Must withstand a fall from at least 1.5 m	Product survived a drop from 1.5m off the ground. **Met requirements**

Constraints:	**Metrics**
Price	Must be under $200
Visible	Visible from 3m away from door
Must not overly obstruct entrance/exit	Must not stick out >12in from the door
Lightweight to comfortably fit on door frame	Must not weigh more than 700g

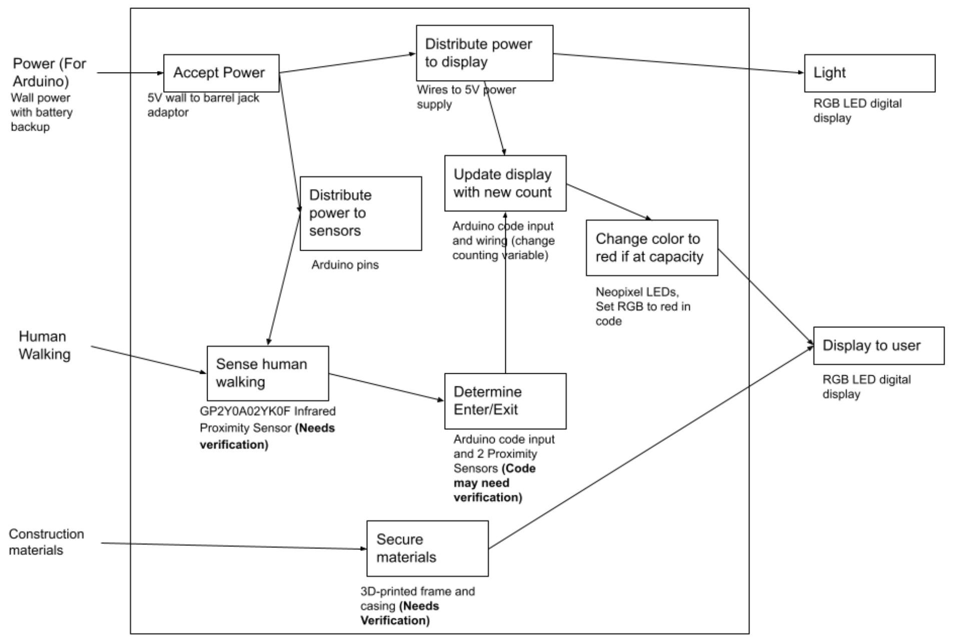
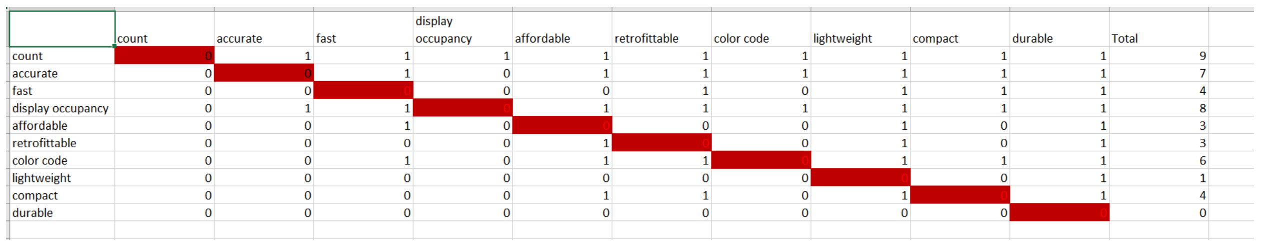

Power (For Arduino)
Wall power with battery backup
Accept Power
5V wall to barrel jack adaptor
Distribute power to display
Wires to 5V power supply
Light
RGB LED digital display
Distribute power to sensors
Arduino pins
Update display with new count
Arduino code input and wiring (change counting variable)
Change color to red if at capacity
Neopixel LEDs, Set RGB to red in code
Human Walking
Sense human walking
GP2Y0A02YK0F Infrared Proximity Sensor (Needs verification)
Determine Enter/Exit
Arduino code input and 2 Proximity Sensors (Code may need verification)
Display to user
RGB LED digital display
Construction materials
Secure materials
3D-printed frame and casing (Needs Verification)

Figure 1: Glass box diagram for the product



	count	accurate	fast	display occupancy	affordable	retrofittable	color code	lightweight	compact	durable	Total
count	0	1	1	1	1	1	1	1	1	1	9
accurate	0	0	1	0	1	1	1	1	1	1	7
fast	0	0	0	0	0	1	0	1	1	1	4
display occupancy	0	1	1	0	1	1	1	1	1	1	8
affordable	0	0	1	0	0	0	0	1	0	1	3
retrofittable	0	0	0	0	1	0	0	1	0	1	3
color code	0	0	1	0	1	1	0	1	1	1	6
lightweight	0	0	0	0	0	0	0	0	0	1	1
compact	0	0	0	0	1	1	0	1	0	1	4
durable	0	0	0	0	0	0	0	0	0	0	0

Figure 2: Pairwise comparison chart for primary objectives

Table 2: Morph chart for the product

Functions/ Means	**1**	**2**	**3**	**4**
Accept power	Wall plug	12V lipo battery	9V dry cell	
Detect enter/exit room	Beam Break sensor	Ultrasonic Sensor	Laser TOF sensor	Pressure plate
Keep track of # people in room	Arduino	Raspberry PI		

Power the means of detection	Arduino	Raspberry PI	Voltage regulator	Separate battery
Sense when the detector Trips	Arduino	Raspberry PI	Plug in Computer	
Light LEDs depending on # people	Arduino	Raspberry PI	Plug in Computer	
Display the occupancy count	7 segment display	LEDs on breadboard	LCD	Neopixel display
Secure components together/to entrance	Wood	Sheet steel	Polycarbonate /acrylic	Tape
Distinguish between entering and exiting	dual beam break sensors	TOF sensor (to measure relative velocity)		

Table 3: Final list of parts with costs

Product Name with link to where it was purchased	**Cost**
Elegoo UNO	$16.99
GP2Y0A02YK (Proximity Sensors)	$18.88
KY-040 360 Degrees Rotary Encoder	$9.99
BTF-Lightning WS2812B ECO RGB 16x16 Display	$17.88
5V 6A Power Supply	$14.59
3D print (205g * 1.5 (print time cost) * $0.0148/gram)	$4.59
Heated Inserts	$10.65
Arduino Shield	$14.11

4-40 Screws	$10.10
22 AWG Assorted Color Wire	$11.98
1/16th in. Plexiglass	$6.79
TOTAL	**$136.55**

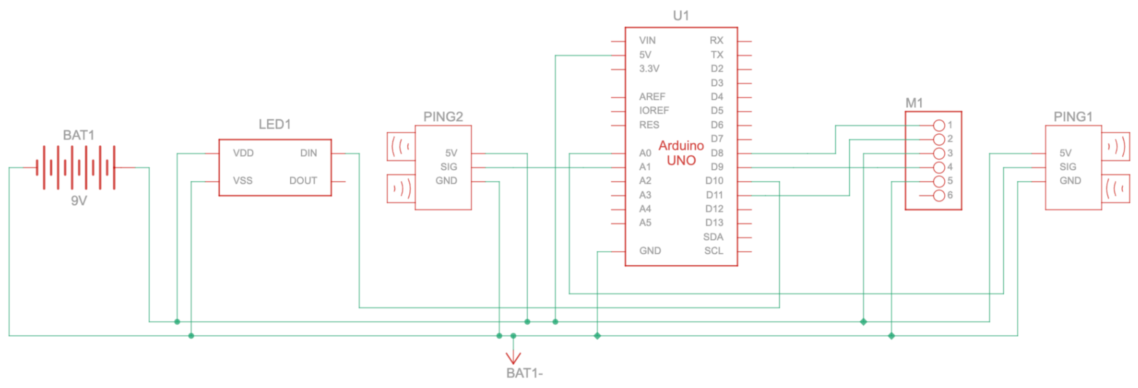


Figure 3: Electronics Sketch

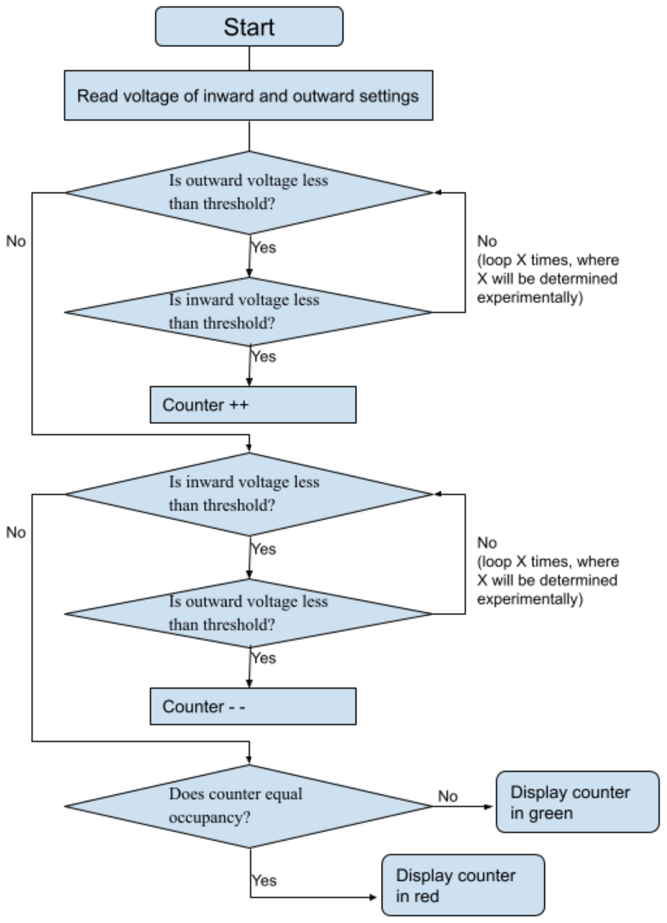

Start
Read voltage of inward and outward settings
Is outward voltage less than threshold?
No
Yes
No (loop X times, where X will be determined experimentally)
Is inward voltage less than threshold?
Yes
Counter ++
Is inward voltage less than threshold?
No
Yes
No (loop X times, where X will be determined experimentally)
Is outward voltage less than threshold?
Yes
Counter - -
Does counter equal occupancy?
No
Display counter in green
Yes
Display counter in red

Figure 4: Code Flowchart

```
#include <Adafruit_NeoPixel.h>
#include <Adafruit_NeoMatrix.h>
#include <Adafruit_GFX.h>
int counter=0,setter=0; //creates counter
int doorLen=0; //defines length of door (cm)
int maxPpl=0;
#define numCounts 20 //wait time before ignoring an input from only one sensor
#define delayTime 25 //delay between each sensor check
#define PIN 10
#define CLK 8
#define DT 9
#define SW 11
int check=0,set=0,pplSet=0; //set makes sure the counter doesnt overcount
int currentStateCLK,lastStateCLK; //settings for the encoder
unsigned long lastButtonPress=0;

Adafruit_NeoMatrix matrix = Adafruit_NeoMatrix(16, 16, PIN,
NEO_MATRIX_BOTTOM + NEO_MATRIX_RIGHT +
NEO_MATRIX_COLUMNS + NEO_MATRIX_ZIGZAG,
NEO_GRB + NEO_KHZ800);//initializing matrix for screen

const uint16_t colors[] = {matrix.Color(255, 0, 0), matrix.Color(196, 180, 84),
matrix.Color(0, 255, 0)}; // red yellow green

void setup() {
pinMode(CLK,INPUT);
pinMode(DT,INPUT);
pinMode(SW,INPUT_PULLUP);
Serial.begin(9600);
matrix.begin();
matrix.setBrightness(20);
lastStateCLK=digitalRead(CLK);
}

void loop() {
bool alreadyCounted=false; //check to make sure sensor isn't messed with
float senOut = analogRead(A0);
float senIn = analogRead(A1);

//loop checks how far the max distance from the wall and sets that to be the door
size
while((pow(senIn/1000,-1.1904)*10)>200&&setter==0){
delay(delayTime);
senIn=analogRead(A1);
}
if (setter==0){
doorLen=((pow(senIn/1000,-1.1904)*10)-3);
setter=1;
}
Serial.println(doorLen);
//more screen setting stuff
matrix.fillScreen(0);
matrix.setCursor(2,5);
matrix.setTextColor(colors[2]);
matrix.drawCircle(7, 8, 7, matrix.Color(0,0,255));
matrix.setTextSize(1);

//loop allows the user to set the max number of people with the encoder
while (pplSet%2==0){
matrix.fillScreen(0);
matrix.setCursor(2,5);
matrix.setTextColor(matrix.Color(160,32,240));
matrix.setTextSize(1);
```

```
currentStateCLK=digitalRead(CLK);
if (currentStateCLK!=lastStateCLK){
if (digitalRead(DT)!=currentStateCLK){
maxPpl--;
}else{
maxPpl++;
}
}
lastStateCLK=currentStateCLK;
// detects click and switches back to the regular sensor if it occurs
int btnState=digitalRead(SW);
if (btnState==LOW){
if (millis()-lastButtonPress>50){
pplSet++;
}
lastButtonPress=millis();
}
if (maxPpl<0){
maxPpl=0;
}
if (maxPpl>99){
maxPpl=99;
}
Serial.println(maxPpl);
matrix.print(maxPpl);
matrix.show();
delay (1);
}
int btnState=digitalRead(SW);
if (btnState==LOW){
if (millis()-lastButtonPress>50){
pplSet++;
}
lastButtonPress=millis();
}

matrix.fillScreen(0);
matrix.setCursor(2,5);
if(counter>=maxPpl){
matrix.setTextColor(colors[0]);
matrix.drawCircle(7,8,7, matrix.Color(255,172,28));
if (counter==maxPpl){
matrix.drawLine(2, 3, 26, 27, matrix.Color(255, 127, 28));
matrix.drawLine(13,14,30,31,matrix.Color(0,0,0));
}
}else{
```

```
matrix.setTextColor(colors[2]);
matrix.drawCircle(7, 8, 7, matrix.Color(0,0,255));
}
matrix.setTextSize(1);

//Serial.println(senOut);
//Serial.println(pow(senOut/1000,-1.1904)*10);
//loop that checks whether the current person going through was already counted and
resets it
if((pow(senOut/1000,-1.1904)*10)>doorLen&&(pow(senIn/1000,-1.1904)*10)>doorLen){
set=0;
}
if (set==0){
//checks if the outer sensor was tripped
if((pow(senOut/1000,-1.1904)*10)<doorLen){
//runs loop until inner sensor is tripped
while((pow(senOut/1000,-1.1904)*10)<doorLen){
delay(delayTime);
senOut = analogRead(A0);
senIn = analogRead(A1);
if ((pow(senIn/1000,-1.1904)*10<doorLen)){
counter++;
alreadyCounted=true;
break;
}
}
//waits for a bit in case someone stops directly on the sensor
if (alreadyCounted == false){
for (int i=0;i<=20;i++){
delay(delayTime);
senOut = analogRead(A0);
senIn = analogRead(A1);
if(pow(senIn/1000,-1.1904)*10<doorLen){
counter++;
break;
}
}
}
//sets set to 1 so the machine doesn't overcount
set=1;
}
//checks if the inner sensor was tripped and repeats first sensor checks
else if((pow(senIn/1000,-1.1904)*10)<doorLen){
while((pow(senIn/1000,-1.1904)*10)<doorLen){
delay(delayTime);
senOut = analogRead(A0);
senIn = analogRead(A1);
if ((pow(senOut/1000,-1.1904)*10<doorLen)){
counter--;
alreadyCounted=true;
break;
}
}
if (alreadyCounted == false){
for (int i=0;i<=20;i++){
delay(delayTime);
senOut = analogRead(A0);
senIn = analogRead(A1);
if(pow(senOut/1000,-1.1904)*10<doorLen){
counter--;
break;
}
}
}
set=1;

}
}
//sets counter to 0 if it ends up in the negative
if (counter<0){
counter=0;
}
if (counter>maxPpl){
matrix.print('+'+String(counter-maxPpl));
}else matrix.print(counter);
matrix.show();
//delays to restart the loop over again
delay(delayTime);
}
```

Figure 5: Arduino code

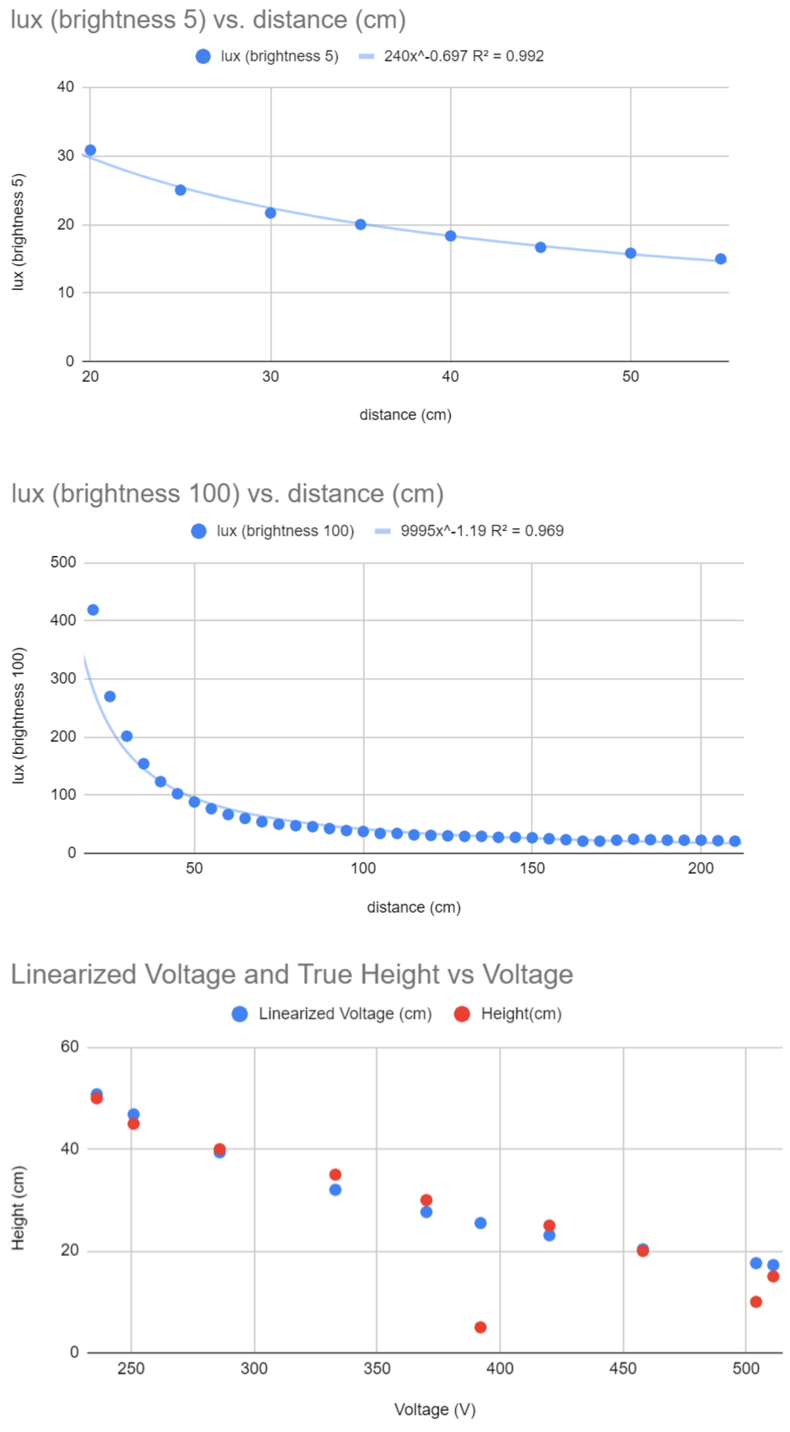


Figure 6: Brightness testing results

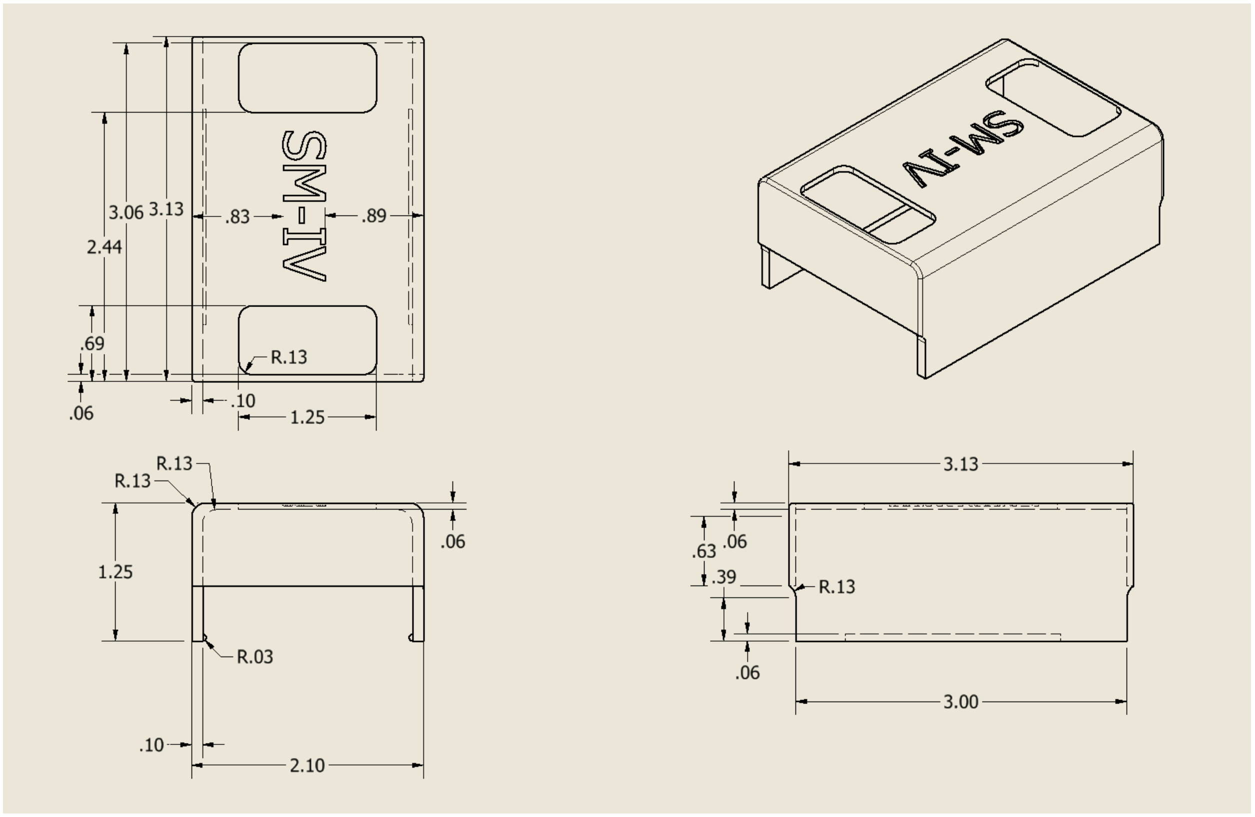


Figure 7: CAD Drawing of Cover

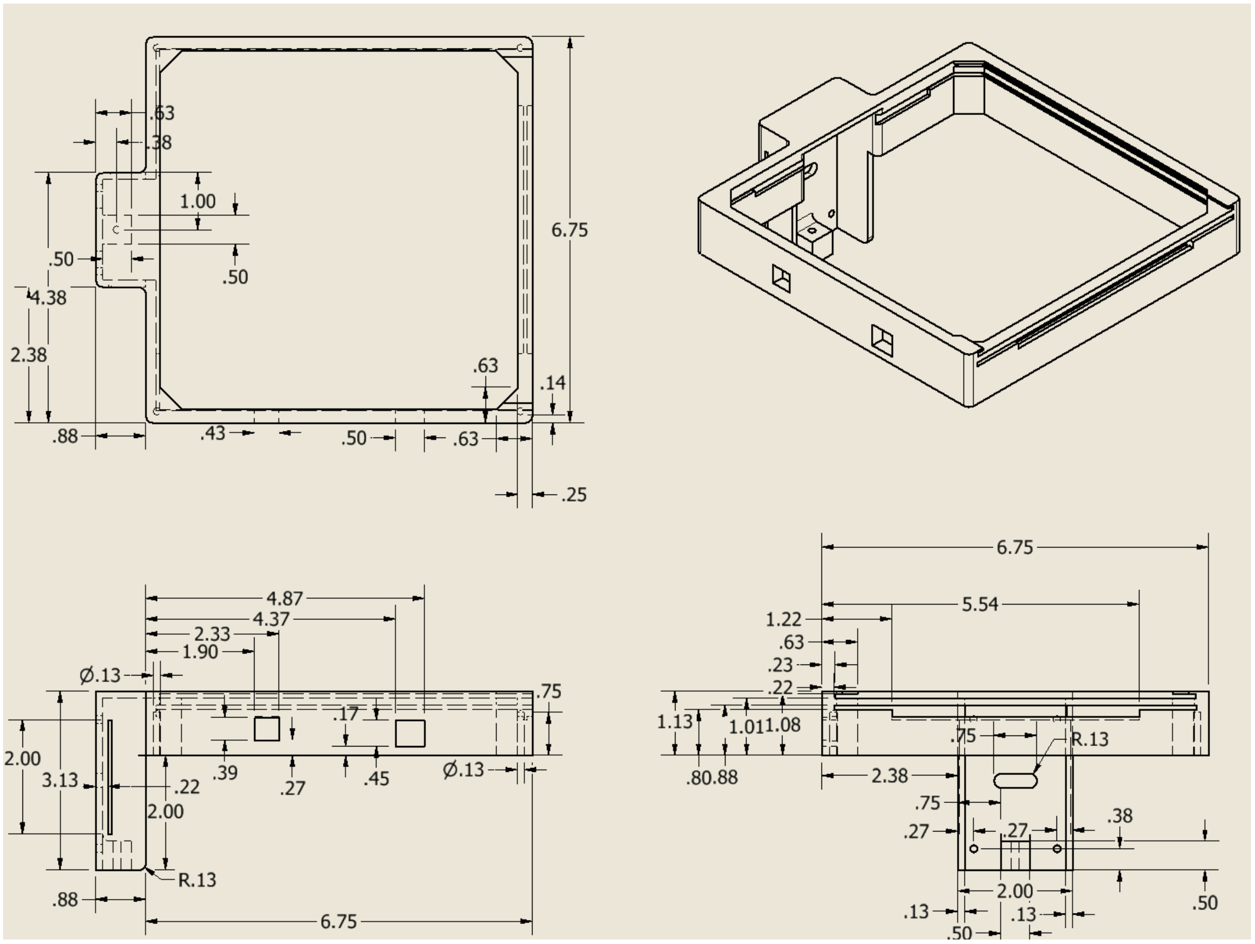


Figure 8: CAD Drawing of Case

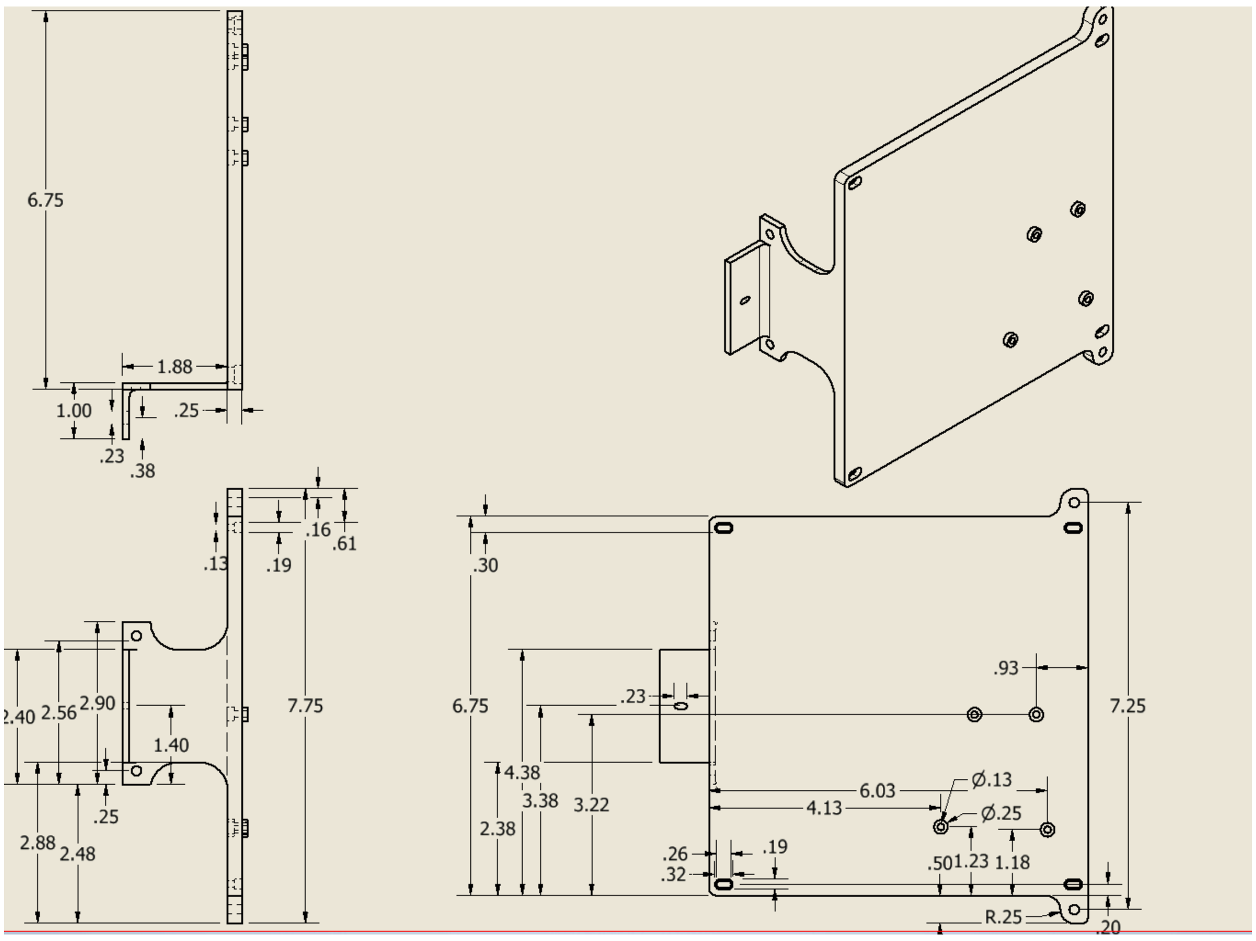
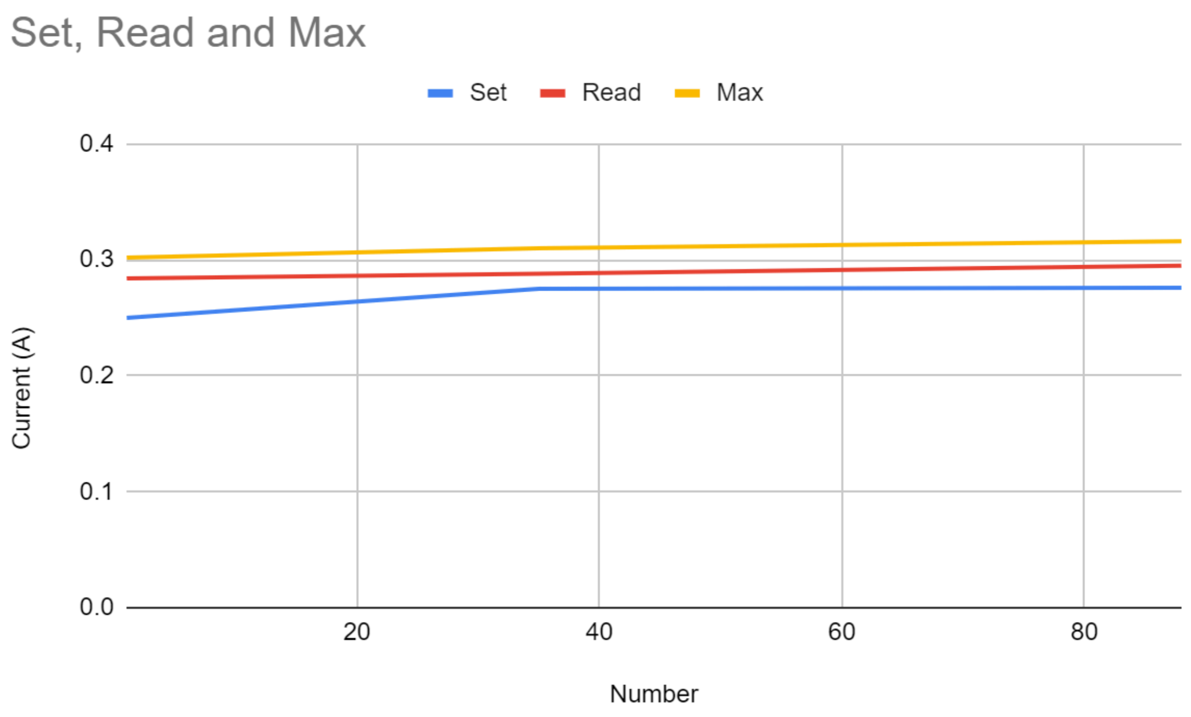


Figure 9: CAD Drawing of Lid



- Arduino max draw: 200mA
- No more than ⅓ of WS28128b on, drawing 60mA each (5.1A at absolute maximum)
- Actual testing: drew .315 Amps of current at maximum (#88, capacity)

Figure 10: Power Calculations

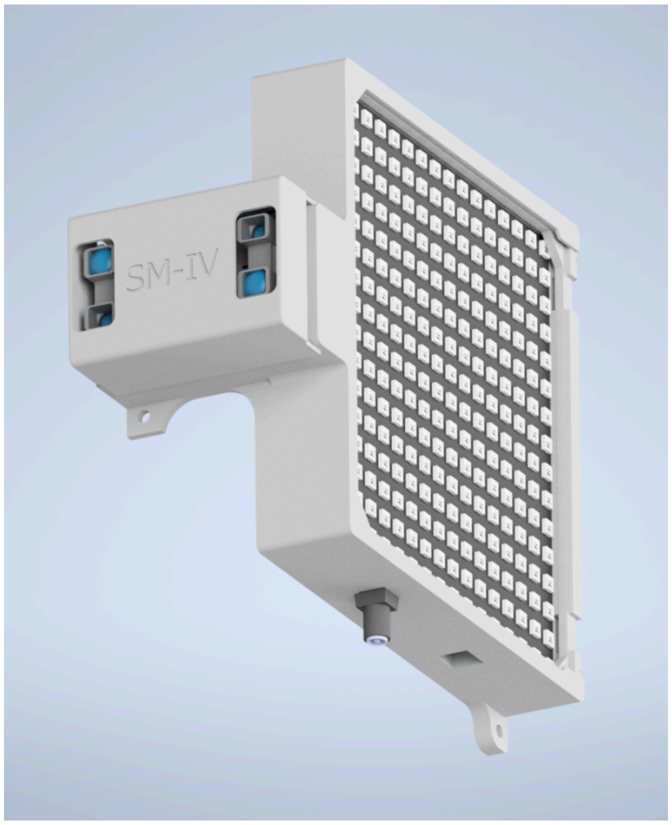


Figure 11: CAD model of the completed assembly