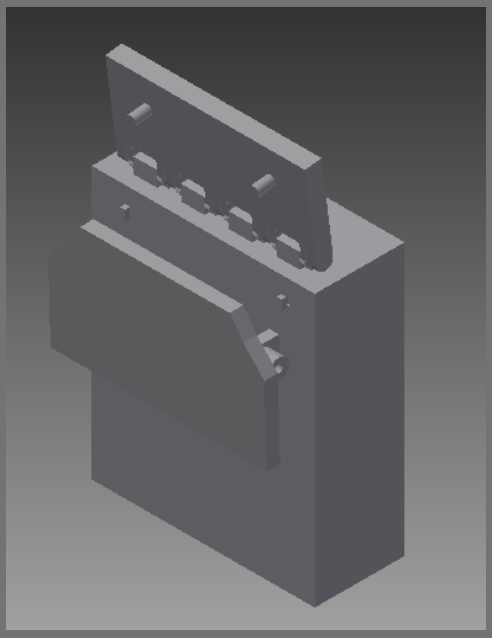
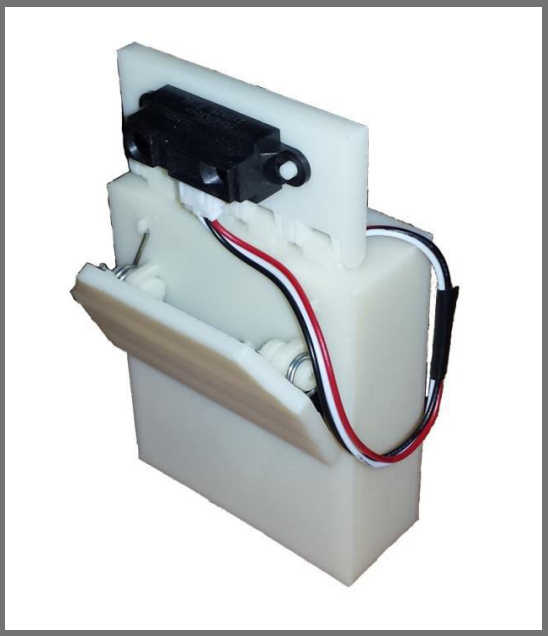
Using IR-Sensors to Limit the Screen-On Time of a Computer

Data Book

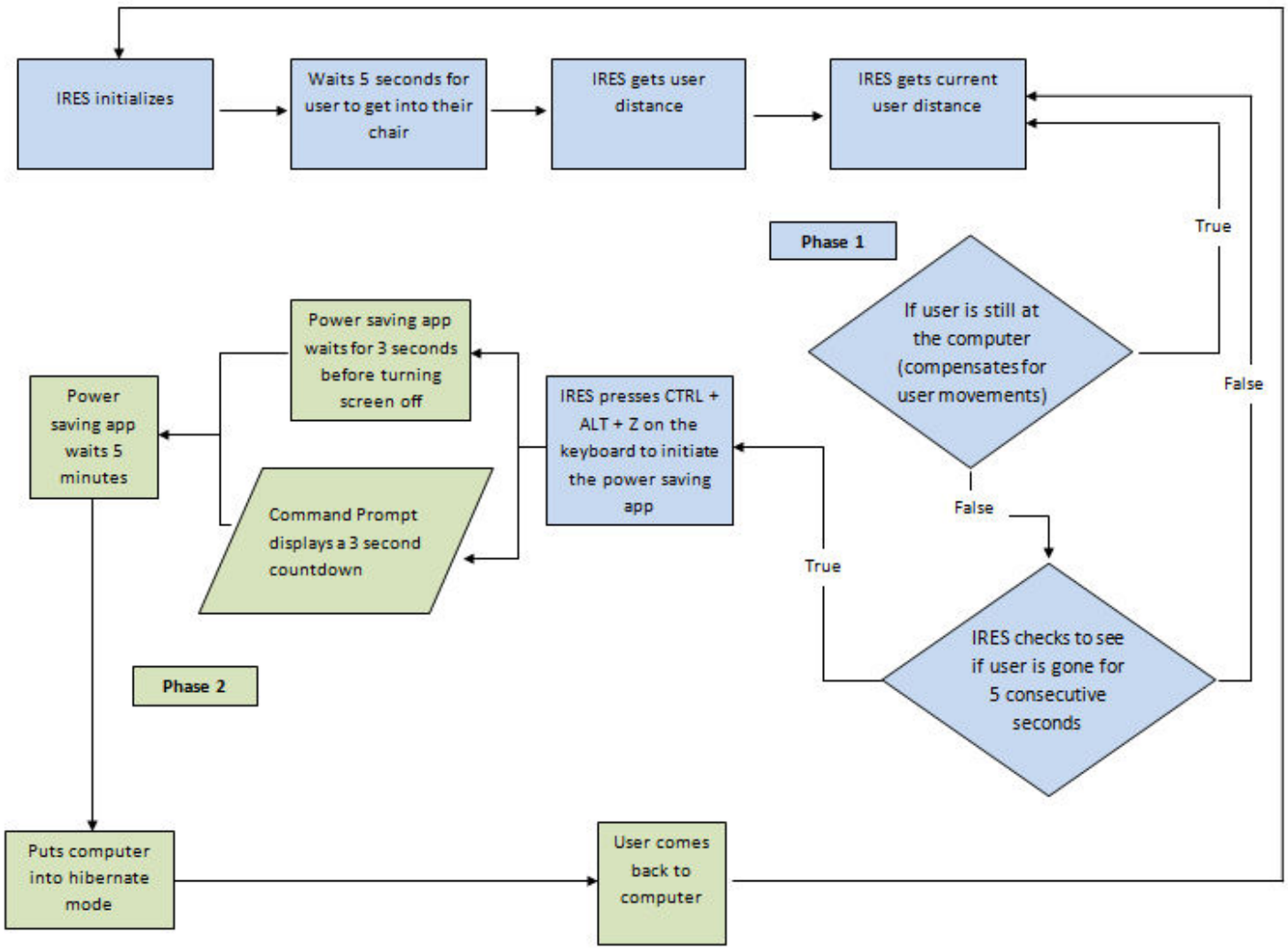




Akshath Jain

Table of Contents

IRES Algorithm



Code

Arduino

```
/*
Name: Akshath Jain
 Date: 3/14/15
 Purpose: To create a program for IRES that is GUI compatible and efficient
 */

const int IRPIN = A0;
int time = 0, userStatus = 0; //for userStatus, 0 is false, 1 is true, and 2 is left the computer
double initialUserDistance, currentUserDistance, acceptableDistance;

void setup(){
  pinMode(IRPIN, INPUT); //Sets the ir sensor as an input
  Keyboard.begin();
  Serial.begin(9600);
  delay(5000); //Waits 5 second
  initialUserDistance = getDistance();
  getAcceptableDistance();
}

void loop(){
  currentUserDistance = getDistance(); //Gets current user distance
  userStatus = 0;
  if(currentUserDistance < acceptableDistance && verifyUserDistance()) {
    keyPress();

    while(currentUserDistance < acceptableDistance){
      currentUserDistance = getDistance();

      if(currentUserDistance >= acceptableDistance && verifyUserGone())
        break;

      time++;
      printStats();
    }
  }
  time++; //keeps track of time
  printStats();
}

boolean verifyUserDistance(){ //return true if user is gone for 5 consecutive seconds
  userStatus = 1;

  for(int i = 0; i < 5; i++){ //This loop tests to make sure that the user is gone, makes sure for 5 seconds
    currentUserDistance = getDistance(); //Gets user distance

    if(currentUserDistance >= acceptableDistance){ //Checks to see if user is at the computer
      userStatus = 0;
      return false; //If the user is still at the computer
    }

    time++;
    printStats();
  }
  userStatus = 2;
```

```
  return true;
}

boolean verifyUserGone(){
  userStatus = 1;

  for(int i = 0; i < 5; i++){
    currentUserDistance = getDistance(); //Gets user distance

    if(currentUserDistance < acceptableDistance){
      userStatus = 2;
      return false; //If user is still gone
    }

    time++;
    printStats();
  }
  userStatus = 0;
  return true;
}

double getDistance(){ //gets the user's distance from the ir sensor
  double temp = 0;

  for(int i = 0; i < 1000; i++){ //This loop will get a thousand readings
    temp += analogRead(IRPIN);
    delay(1);
  }
  return temp/1000; //And this will return the average of those thousand readings
}

double getAcceptableDistance(){
  double v = initialUserDistance * 5.0/1023, d;
  d = 23.709*pow(v, -1.179); //voltage to distance
  d += 20;//this allows the user to move back for 30 cm, equivalent to leaning back.
  acceptableDistance = 14.363*pow(d, -0.843) * 1023.0/5;   //distance to voltage
}

void keyPress(){ //This function initiates the keypresses required to turn the screen off
  char ctrlKey = KEY_LEFT_CTRL; //This is how the Arduino recognizes the control key
  char altKey = KEY_LEFT_ALT; //This is how the Arduino recognizes the alt key
  char zKey = 'z'; //This is how the Arduino recognizes the z key
  Keyboard.press(ctrlKey); //Arduino presses the control key
  delay(10); //Delays for 10 milliseconds
  Keyboard.press(altKey); //Arduino presses the alt key
  delay(10);
  Keyboard.press(zKey); //Arduino presses the z key
  delay(10);
  Keyboard.releaseAll(); //Arduino releases all the key's that are being pressed (control + alt + z)
}

void printStats(){
  Serial.print(initialUserDistance);
  Serial.print(' ');
  Serial.print(acceptableDistance);
  Serial.print(' ');
  Serial.print(currentUserDistance);
  Serial.print(' ');
  Serial.print(time);
  Serial.print(' ');
  Serial.print(userStatus);
  Serial.print(';');
}
```

Screen Control App

```
/*
Name: Akshath Jain
Date: 11/27/14
Purpose: To control my computer screen
*/

#include <iostream>
#include <Windows.h>
#include <PowrProf.h>
#include <time.h>
#pragma comment(lib, "PowrProf.lib")
using namespace std;

int main()
{
    system("color 0a");

    cout << " ------------------------" << endl;
    cout << "|Automatic Screen Control|" << endl;
    cout << "|      Akshath Jain      |" << endl;
    cout << " ------------------------" << endl;

    cout << "\nShutting down in: ";
    for (int i = 3; i >= 0; i--){
        cout << i << "\b"; //Backspace
        Sleep(1000); //Waits 1 seconds, total of 4 seconds
    }
    SendMessage(HWND_BROADCAST, WM_SYSCOMMAND, SC_MONITORPOWER, (LPARAM)2); //Turns screen off
    Sleep(300000); //waits 5 minutes
    SetSuspendState(true, true, true); //hibernates computer
}
```

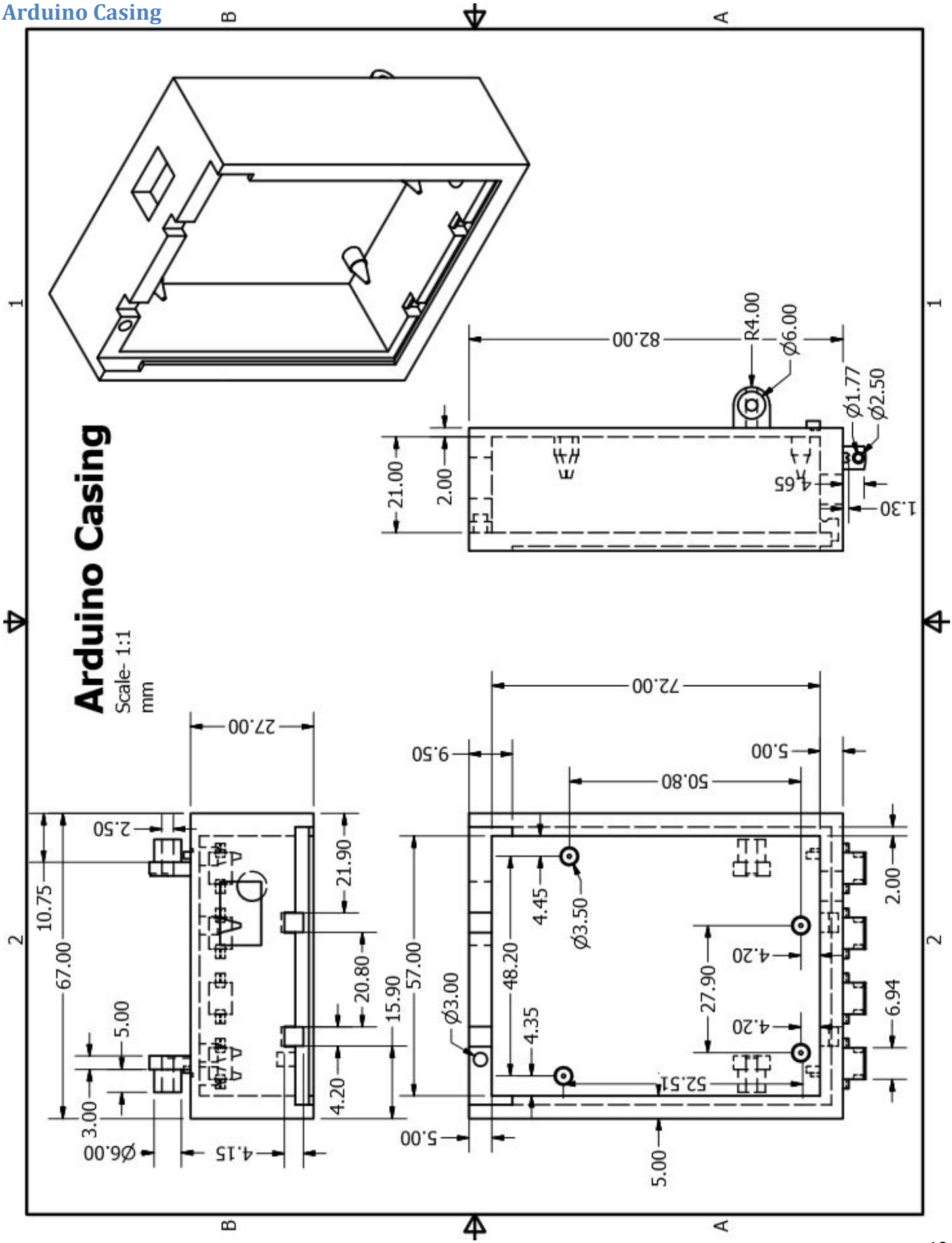
Graphical User Interface (GUI)

```
/*
Name: Akshath Jain
 Date: 3/14/15
 Purpose: Creating a GUI for IRES
 */
import processing.serial.*;

Serial port;
PFont font;
int size = 5;
String IRESData[] = {"00", "00", "00", "0", " "};
String headings[] = {"Initial User Distance: ", "Acceptable User Distance: ", "Current User Distance: ", "Total Time: ", "User Status: "};
String subHeadings[] = {" cm", " cm", " cm", "", ""};
int userStatusColor = #ffffff, totalTimeInSeconds, currentTimeInSeconds = 120000;

void setup(){
  size(700, 500);
  port = new Serial(this, "COM3", 9600);
  port.bufferUntil(';'); //specifies how often serialEvent() will iterate
  font = loadFont("Calibri-50.vlw");
  textFont(font);
}

void draw(){
  if(totalTimeInSeconds > currentTimeInSeconds){
    if((mouseX > 290 && mouseX < 420) && (mouseY > 175 && mouseY < 235)){ //if mouse is over the dismiss button
      showMessage(#67676F, #F5F5F5); //calls function show message
      if(mousePressed == true){ //if mouse is pressed
        currentTimeInSeconds = totalTimeInSeconds + 10000; //reminds the user to take a break
      }                                                    //once every 10 minutes after 2 hours
    }
    else
      showMessage(#000000, #ffffff);
  }
  else{
    setBackground();
    for (int i = 0; i < size; i++){
      if (i < 2) //makes text white
        fill(255, 255, 255);
      else //makes this text black
        fill(0, 0, 0);
      text(headings[i] + IRESData[i] + subHeadings[i], 10, (65 + i * 100));
    }
  }
}

void serialEvent(Serial port){
  char endStop[] = {' ', ' ', ' ', ' ', ';'};
  for (int i = 0; i < size; i++){
    IRESData[i] = port.readStringUntil(endStop[i]);
    IRESData[i] = IRESData[i].substring(0, IRESData[i].length() - 1);
  }
  fixDistance();
  fixTime();
  fixUserStatus();
}

void setBackground(){
  background(0, 0, 0); //sets the bacground to black
  fill(255, 255, 255); //makes a white rectangle (x,y,width,height)
  rect(0, 200, 700, 200);

  fill(userStatusColor); //makes a colored rectangle depending on where the user is
  rect(0, 400, 700, 100);
}

void showMessage(int buttonColor, int textColor){
  background(76,77,85);

  fill(76,77,85);
  rect(0,0,700,500);
```

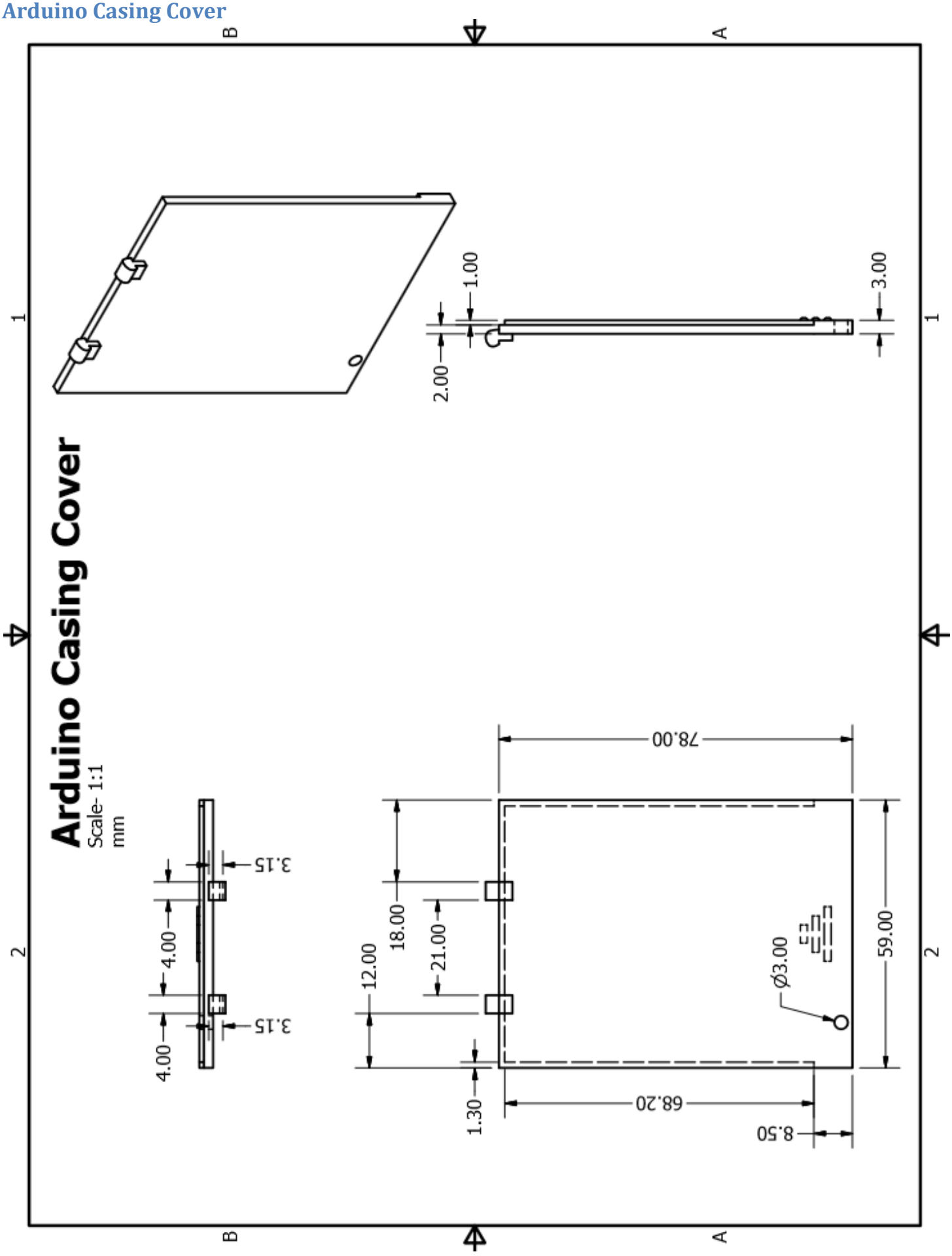
```
  fill(255,255,255);
  text("Time to take a break!", 145, 150);

  fill(buttonColor);
  rect(290, 175, 130, 60); //x,y,length,width

  fill(textColor);
  textSize(30);
  text("Dismiss", 310, 213);

  textSize(50);
}
void fixDistance(){
  double v, d;
  for (int i = 0; i < 3; i++){
    v = Double.parseDouble(IRESData[i]) * 5.0/1023; //converts the Arduino's [0,1023] into voltage from [0,5]
    d = 23.709*pow((float)v, -1.179); //voltage to distance in cm
    IRESData[i] = Integer.toString((int)(d + .5)); //puts data back into array IRESData and
  }                                                //rounds distance, converts to integer, and turns it to a string
}

void fixTime(){
  int h, m, s;

  s = Integer.parseInt(IRESData[3]);
  totalTimeInSeconds = s; //total time in seconds, stored to check if user has been working too long

  m = s / 60; //converts total seconds into the total number of whole minutes
  s %= 60; //converts total seconds into minutes and seconds, disregarding minutes because this is modular division
  h = m/60; //converts the total number of minutes into total number of whole hours
  m %= 60; //converts total minutes into minutes, it disregards hours because this is modular division

  int t[] = {h, m, s};
  String time[] = {"", "", ""};

  for (int i = 0; i < 3; i++){
    if (t[i] >= 0 && t[i] <= 9)
      time[i] = '0' + Integer.toString(t[i]); //converts an int to a string and adds a 0 in front of it
    else
      time[i] = Integer.toString(t[i]); //converts an int to a string
  }
  IRESData[3] = time[0] + ':' + time[1] + ':' + time[2]; //puts fixed data back into array IRESData at index 3
}

void fixUserStatus(){
  int status = Integer.parseInt(IRESData[4]);
  switch(status){
  case 0:
    userStatusColor = #00D817;
    IRESData[4] = "At Computer";
    break;
  case 1:
    userStatusColor = #F7DF00;
    IRESData[4] = "Verifying User";
    break;
  case 2:
    userStatusColor = #EA0C0C;
    IRESData[4] = "User is Gone";
    break;
  }
}
```

Working Drawings

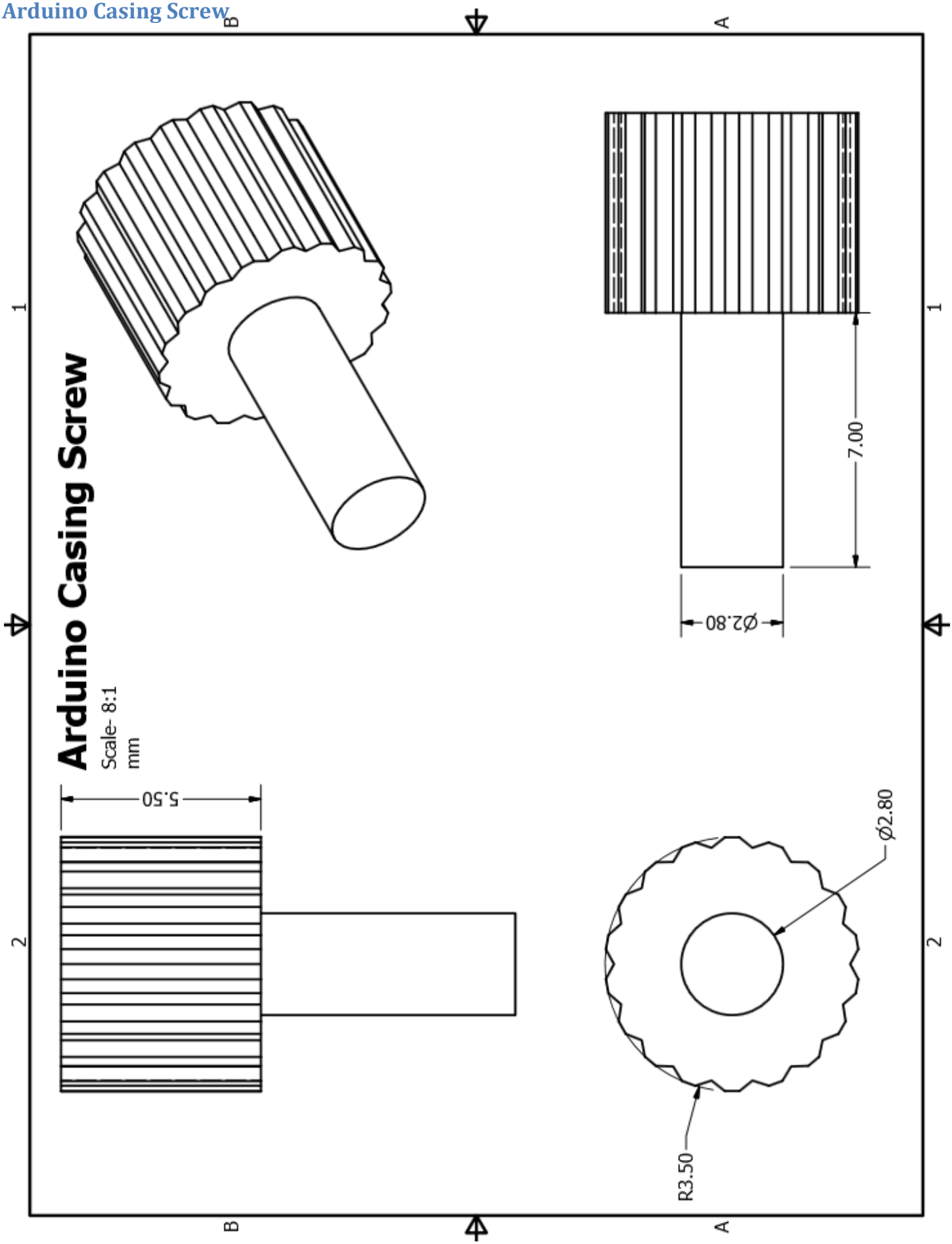
Arduino Casing



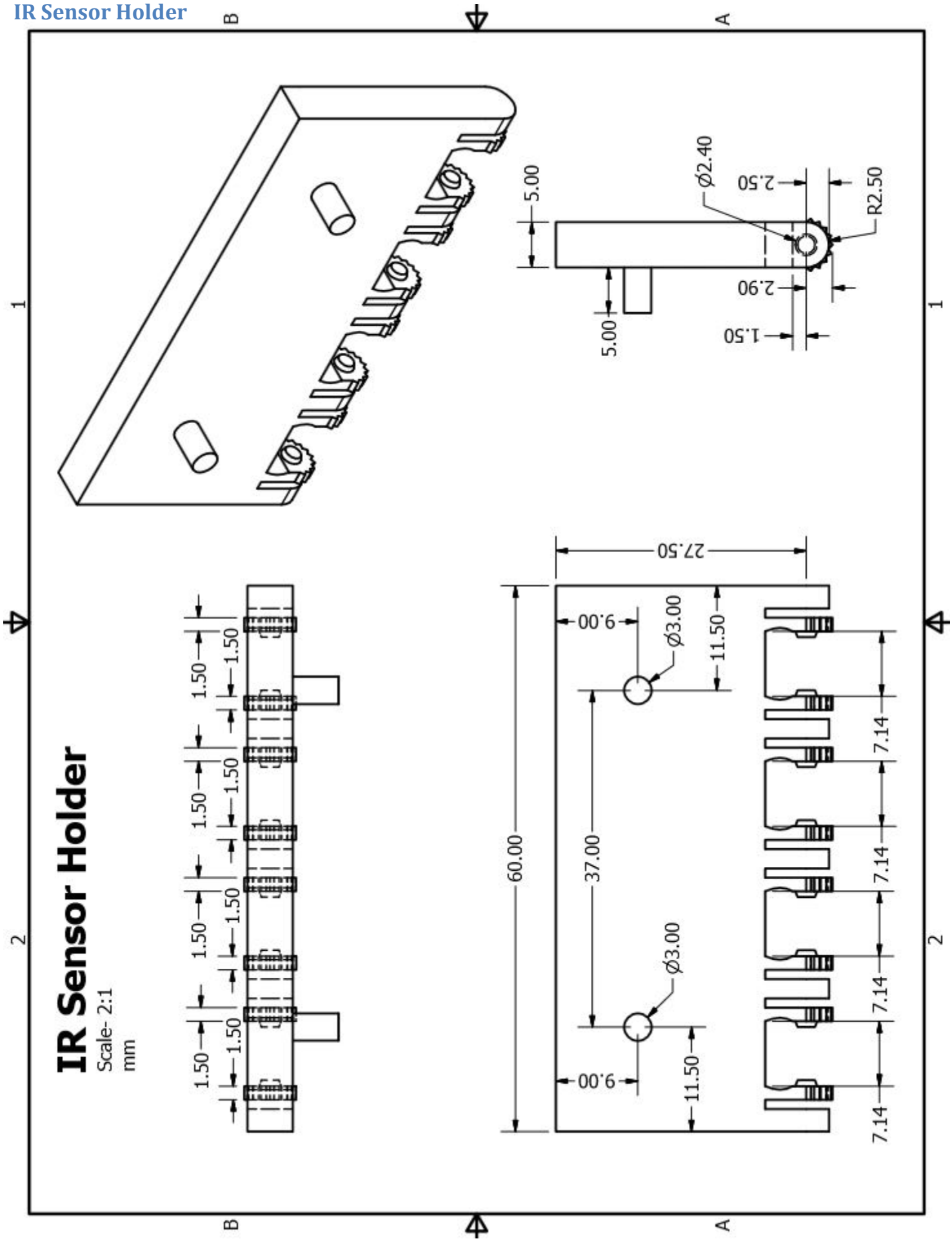
Arduino Casing Cover



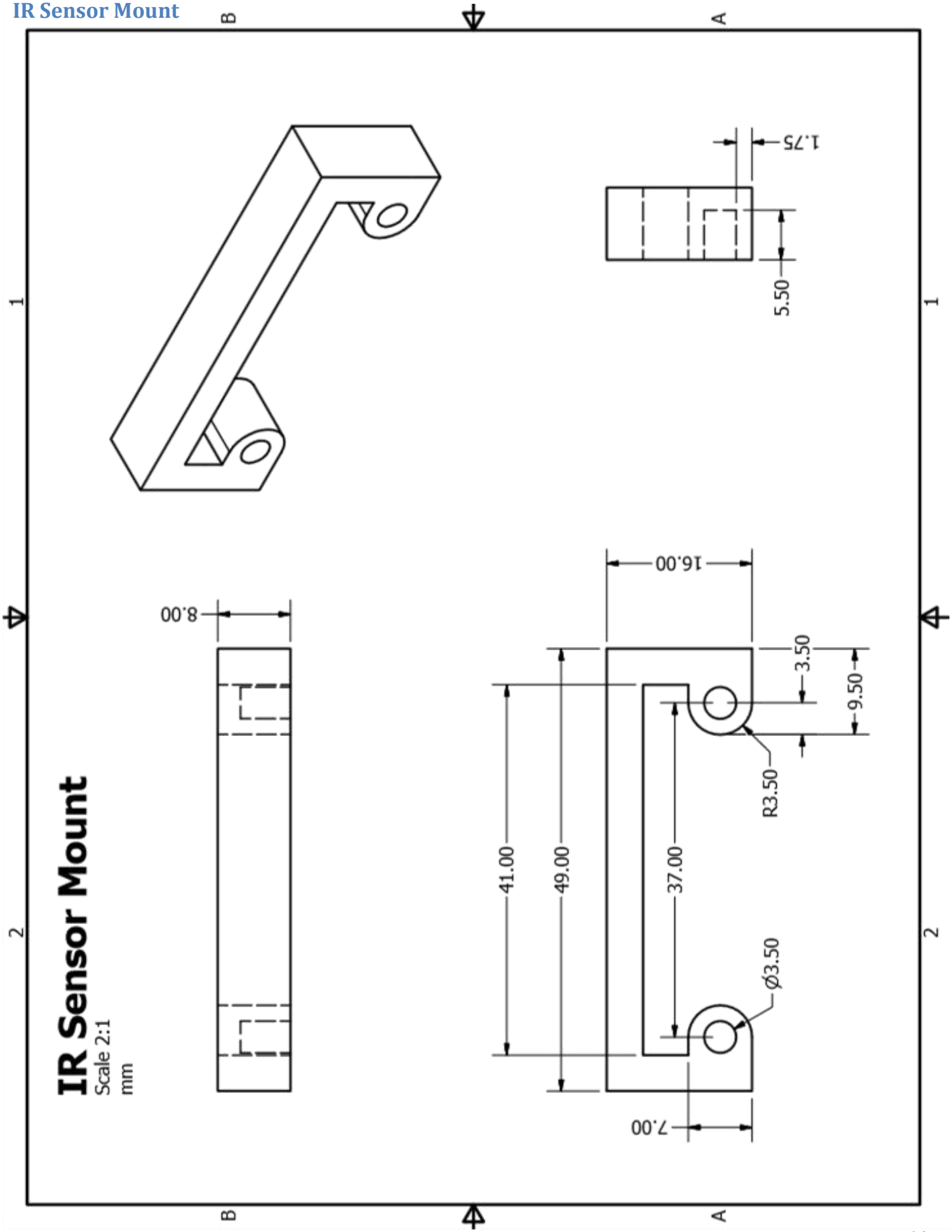
Arduino Casing Screw



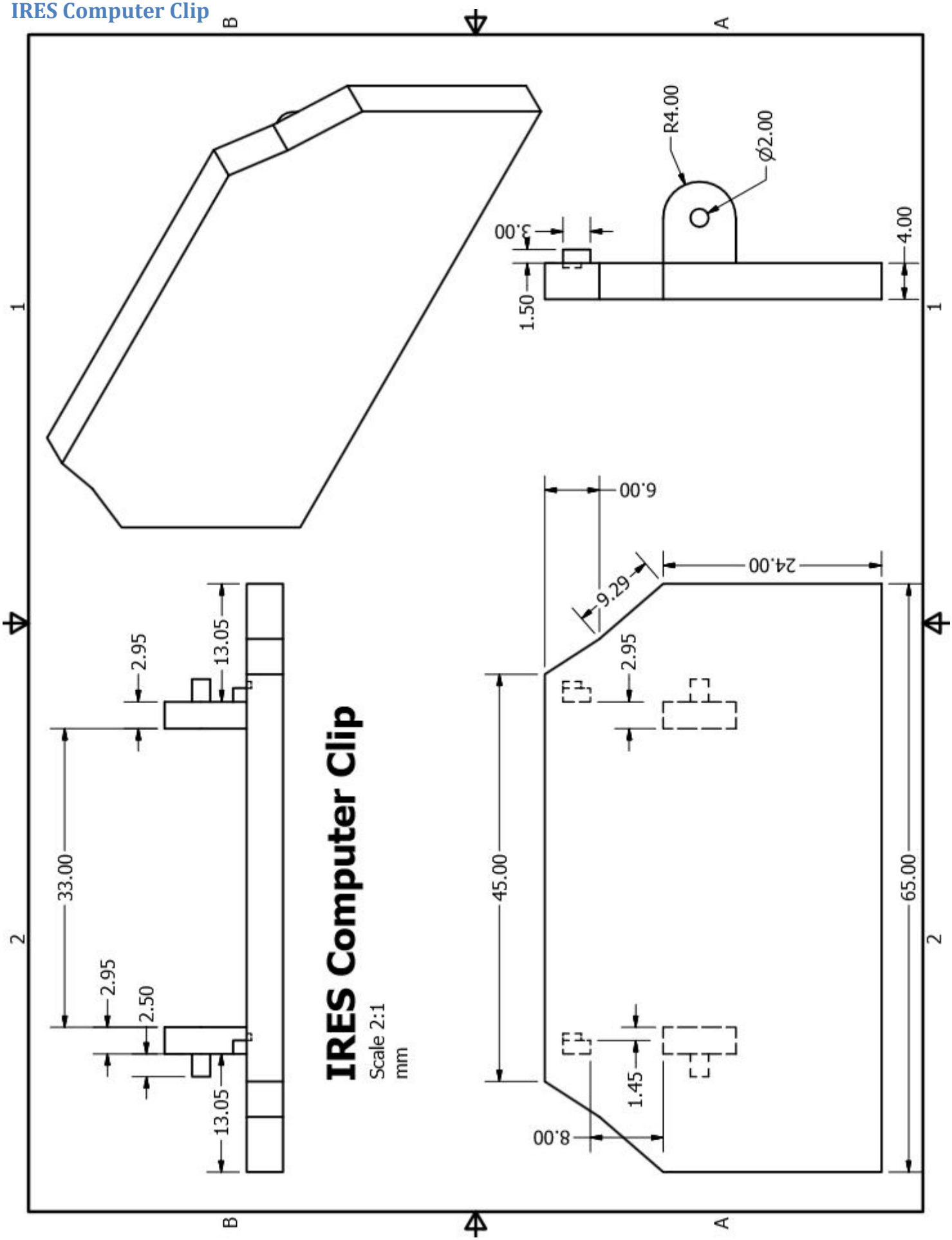
IR Sensor Holder



IR Sensor Mount

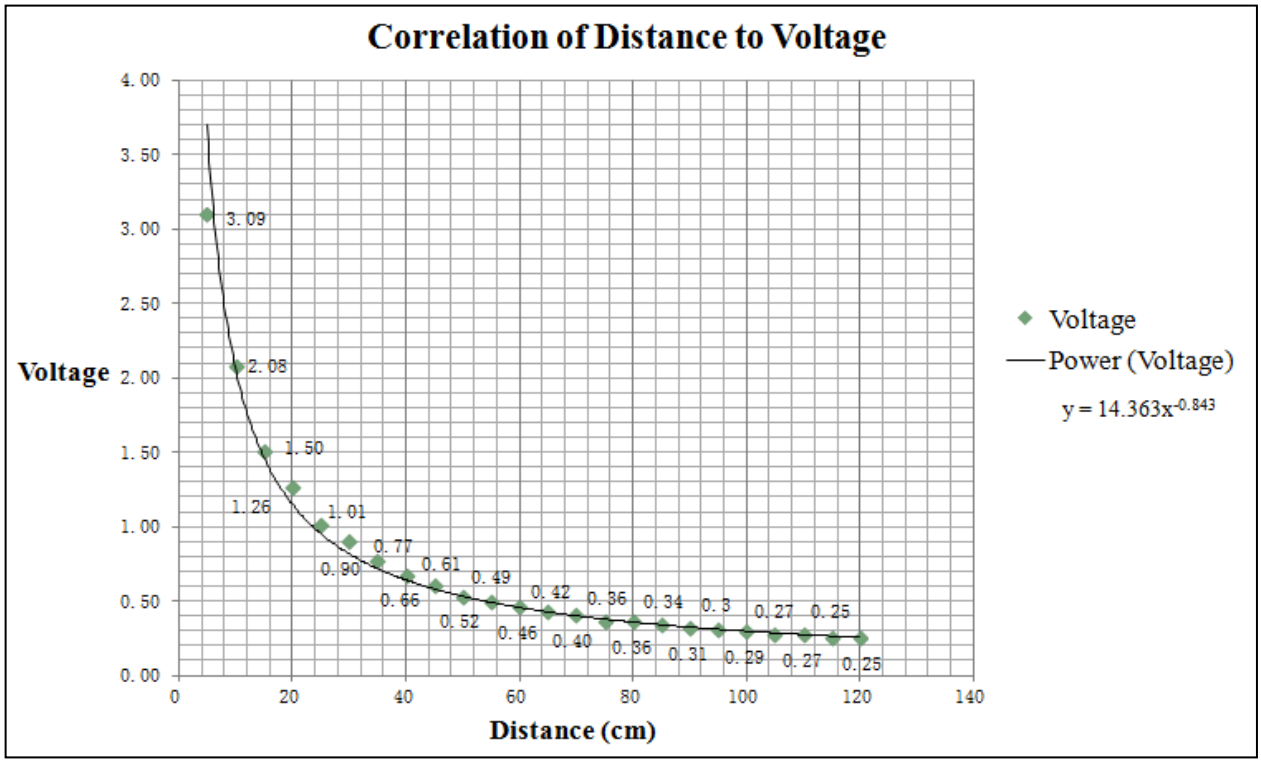


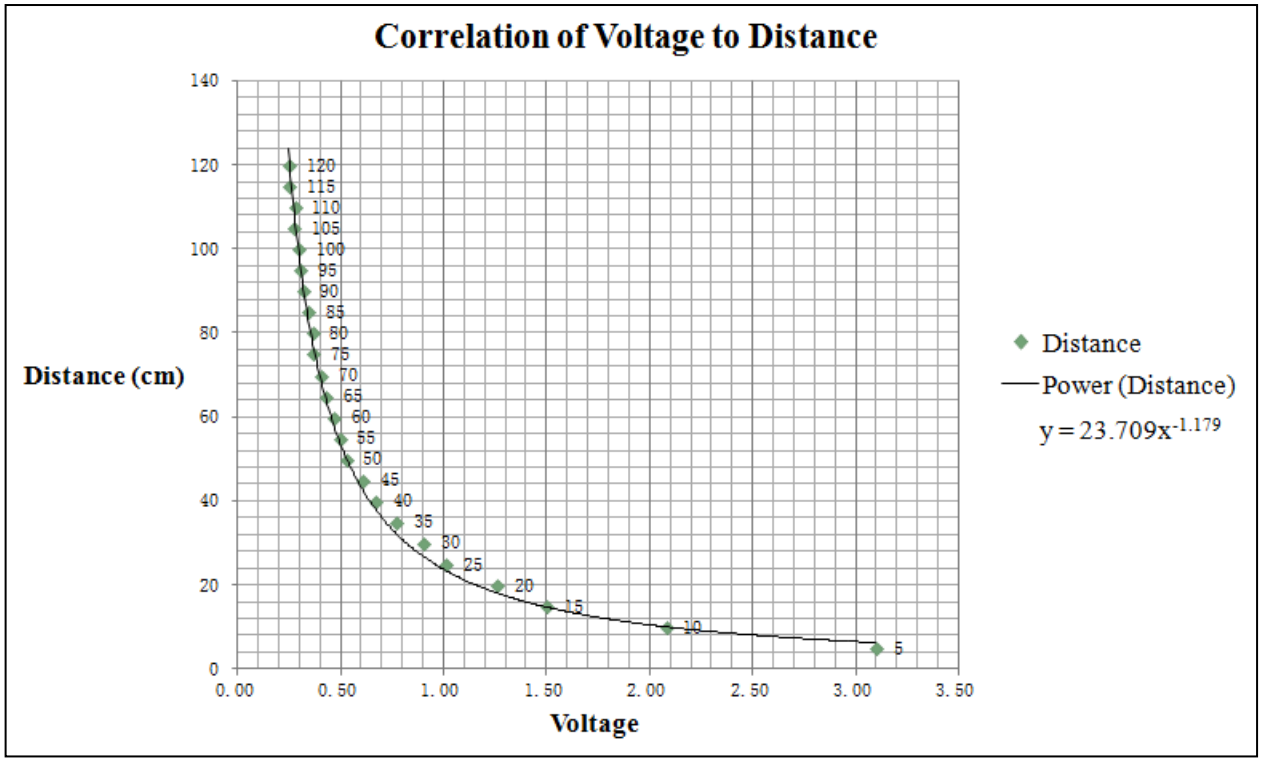
IRES Computer Clip



IR-Sensor Output

IR Sensor Output	
Distance (cm)	Voltage
5	3.09
10	2.08
15	1.50
20	1.26
25	1.01
30	0.90
35	0.77
40	0.66
45	0.61
50	0.52
55	0.49
60	0.46
65	0.42
70	0.40
75	0.36
80	0.36
85	0.34
90	0.31
95	0.30
100	0.29
105	0.27
110	0.27
115	0.25
120	0.25





Questionnaire

Proctor's Name:
Participant's Name:
Date:

Computer Usage

1. Do you use a computer daily?
2. How long do you use it for on average?
3. Do you primarily use a desktop or a laptop?
    a. (If laptop) Is your laptop plugged in and charging while you work?
        i. (If yes) Why do you keep your laptop plugged in? Is it because of a short battery life?
            1. (If yes) Would you benefit if your battery life was better, giving you more versatility in places you are able to work?

Power Saving Techniques

4. Are you aware of any power-saving techniques used by your computer (i.e. screen turns off, computer goes to sleep, etc.)?
    a. (If Yes) How long until the screen times out? (With absolute certainty)
    b. How long until your computer goes to sleep? (With absolute certainty)

Work Habits

5. When you work, do you ever take short breaks away from your computer? This can include anytime that you are away from your computer.
    a. (If yes) For how long do you work at a time, and then take a break?
    b. What is the duration of your break?

Questionaire Results

	Computer Usage						Power Saving Techniques			Computer Habits		
	1	2	3	3a	3ai	3ai1	4	4a	4b	5	5a	5b
	Y/N	Hours	Laptop (L)/ Desktop (D)	Y/N	Convenience (C)/ Short Battery Life (SBL)/ Other (O)	Y/N	Y/N	Minutes	Minutes	Y/N	Hours	Minutes
Person												
1	Yes	10	L	Yes	C	Yes	Yes	10	15	Yes	2	10
2	Yes	10	L	Yes	SBL	Yes	Yes	5	10	Yes	1	45
3	Yes	10	L	Yes	C	Yes	No	5	15	Yes	2	10
4	Yes	10	L	Yes	C	Yes	Yes	15	30	Yes	1	5
5	Yes	10	D				No	0	0	Yes	3	30
6	Yes	10	L	Yes	C	Yes	Yes	5	15	Yes	1	10
7	Yes	10	L	Yes	C	Yes	Yes	5	10	Yes	0.5	5
8	Yes	8	L	Yes	C	Yes	Yes	5	15	Yes	2	5
9	Yes	10	L	Yes	C	Yes	Yes	10	15	Yes	4	5
10	Yes	8	L	Yes	C	Yes	Yes	5	15	Yes	1	32.5
11	No	1/2	D				No	0	0	No		
12	Yes	8	L	Yes	SBL	Yes	No	5	15	Yes	1.75	12.5
13	Yes	8	L	Yes	C	Yes	Yes	5	15	Yes	1	10
14	Yes	8	D				Yes	0	0	Yes	2.50	12.5
15	Yes	8	L	Yes	C	Yes	Yes	5	15	Yes	1.25	7.5
Average		**9.14**						**5.33**	**12.33**		**1.71**	**14.29**

IRES Test Results

Tests	Start			End			Time Elapsed	Variables			
	Starting Percentage	Starting Battery Level		Ending Percentage	Ending Battery Level			Applications Open			IR-Sensor
	%	mAh	mWh	%	mAh	mWh	min	Microsoft Excel	BatteryMon	Microsoft Word	Enabled
Control											
1	100.00%	3792	43457	75.60%	2868	32867	60	X	X		
2	100.00%	3770	43457	76.00%	2866	33034	60	X	X		
3	100.00%	3770	43457	75.60%	2851	32856	60	X	X		
Average	**100.00%**	**3777**	**43,457.00**	**75.73%**	**2862**	**32,919**	**60**				
Test Case 1											
1	100.00%	3783	43457	75.10%	2840	32623	60	X	X		X
2	100.00%	3778	43457	75.20%	2840	32667	60	X	X		X
3	100.00%	3615	41614	75.40%	2725	31369	60	X	X		X
Average	**100.00%**	**3725**	**42,842.67**	**75.23%**	**2802**	**32,220**	**60**				
Test Case 2											
1	100.00%	3590	43457	93.80%	3368	40770	60	X	X		
2	100.00%	3611	43457	91.70%	3313	39871	60	X	X		
3	100.00%	3665	43457	87.30%	3119	37929	60	X	X		
Average	**100.00%**	**3622**	**43,457.00**	**90.93%**	**3266.67**	**39,523.33**	**60**				
Test Case 3											
1	100.00%	3573	43457	97.80%	3492	42480	60	X	X		X
2	100.00%	3561	43457	98.10%	3493	42624	60	X	X		X
3	100.00%	3561	43457	98.20%	3497	42680	60	X	X		X
Average	**100.00%**	**3565**	**43,457.00**	**98.03%**	**3494**	**42,594.67**	**60**				
Test Case 4											
1	100.00%	3788	43457	73.90%	2800	32123	60	X	X	X	
2	100.00%	3802	43457	73.30%	2788	31168	60	X	X	X	
3	100.00%	3629	41614	72.80%	2641	30292	60	X	X	X	
Average	**100.00%**	**3739.67**	**42,842.67**	**73.33%**	**2743.00**	**31,194.33**	**60**				
Test Case 5											
1	100.00%	3696	43457	84.50%	3122	36708	60	X	X	X	X
2	100.00%	3692	43457	84.50%	3119	36719	60	X	X	X	X
3	100.00%	3517	41614	85.50%	3007	35587	60	X	X	X	X
Average	**100.00%**	**3635.00**	**42,842.67**	**84.83%**	**3082.67**	**36,338.00**	**60**				
Test Case 6											
1	100.00%	3818	43457	71.90%	2744	31224	60	X	X	X	
2	100.00%	3793	43457	73.60%	2793	32001	60	X	X	X	
3	100.00%	3652	41614	71.20%	2600	29626	60	X	X	X	
Average	**100.00%**	**3754.33**	**42842.67**	**72.23%**	**2712.33**	**30950.33**	**60**				

Overall Statistics

Overall Statistics				
12:3 Laptops to Desktops	Status Qou	IRES	Savings	IRES Energy Reduction
	ΔmAh	ΔmAh	mAh	%
Desktops	915.67	71.00	844.67	92.25%
Laptops	355.33	71.00	284.33	80.02%
Average w/ 12:3	**467.40**	**71.00**	**396.40**	**84.81%**
Desktops w/ Usage	1034.00	552.33	481.67	46.58%
Laptops w/ Usage	988.67	552.33	436.33	44.13%
Average w/ 12:3	**997.73**	**552.33**	**445.40**	**44.64%**

Energy Statistics

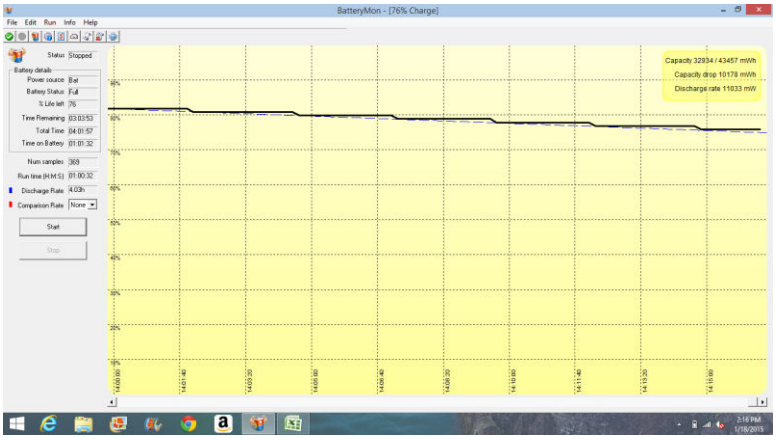
Savings Statistics									
Energy Statistics								Savings	
	# of Computers	Energy Consumption	Computer Usage	Energy Cost	Work Days	CO_2		IRES Optimal	IRES w/ Usage
	Computers (million)	kW (w/ 12:3)	Hours	¢ / kWh	Days	lbs		%	
Yes	289.89	0.12	9.14	12.46	261	1.855		84.81%	44.64%
No	20.71	0.12	0.5	12.46	121	1.855		84.81%	44.64%

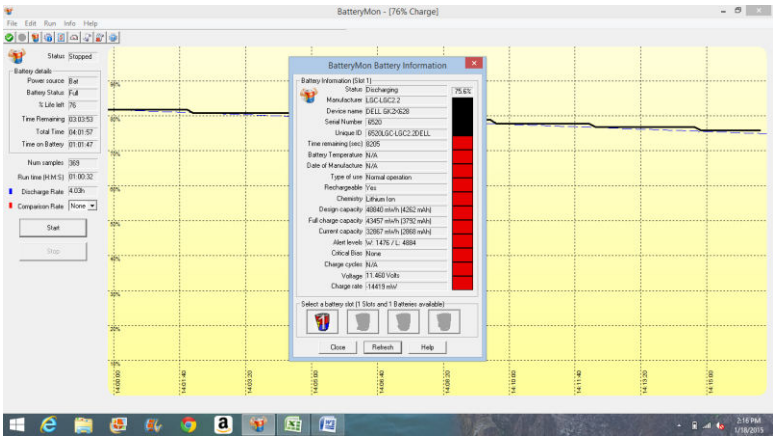
Control

Control						
	1	100.00%	3792	43457	75.60%	2868	32867
	2	100.00%	3770	43457	76.00%	2866	33034
	3	100.00%	3770	43457	75.60%	2851	32856
Average		**100.00%**	**3777**	**43,457.00**	**75.73%**	**2862**	**32,919**

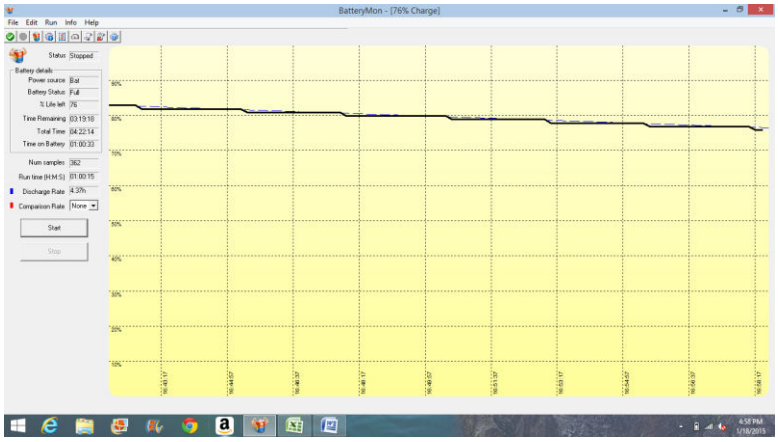
The control test was designed to test how much energy is required to run my laptop (Dell Inspiron) for 60 minutes. Going through with the test, I had three trials, which I then took the average of to find the overall net decrease. Note that power-saving techniques used by the computer were not enabled during this test; it was just running my laptop for 60 minutes without any interruption. At the end of the control, I found out that it takes 915 mAh (Starting mAh – Ending mAh) to run my computer for 60 minutes.

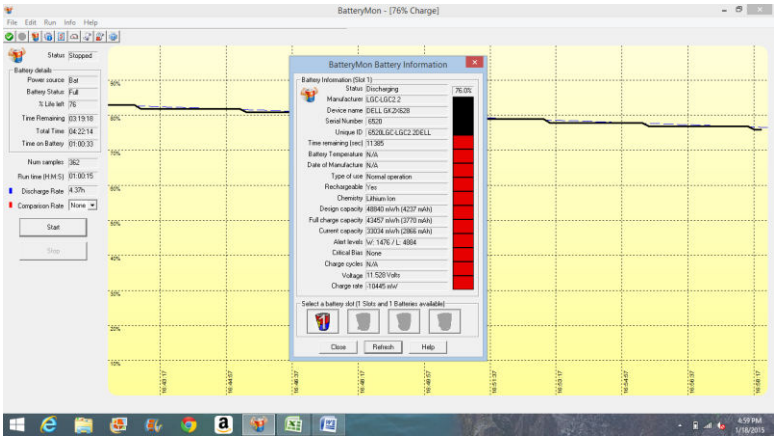
Control – Trial 1



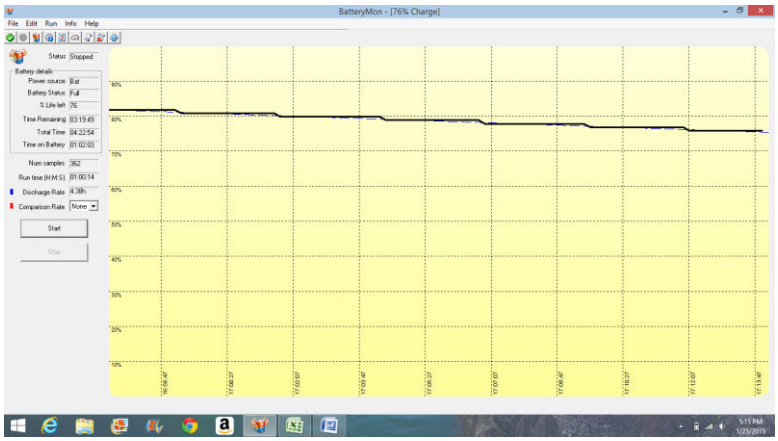


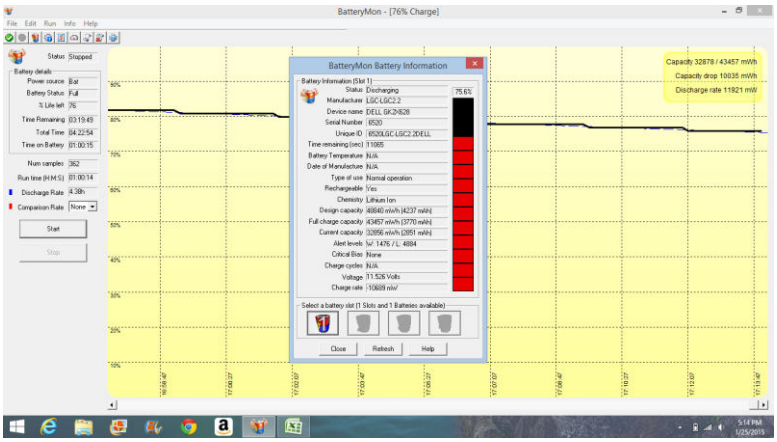
Control – Trial 2





Control – Trial 3



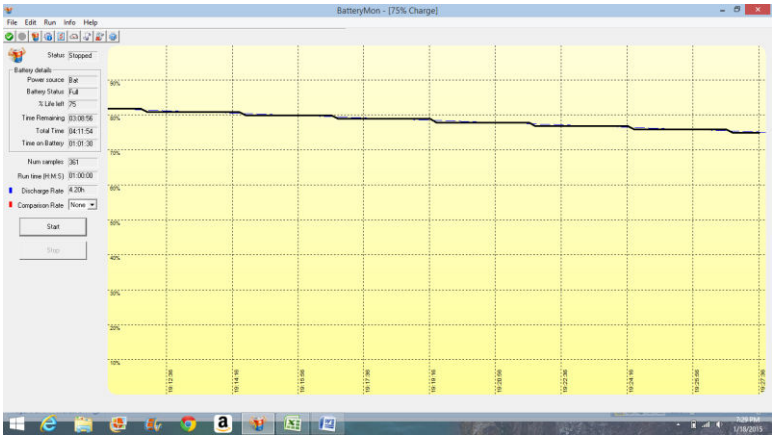


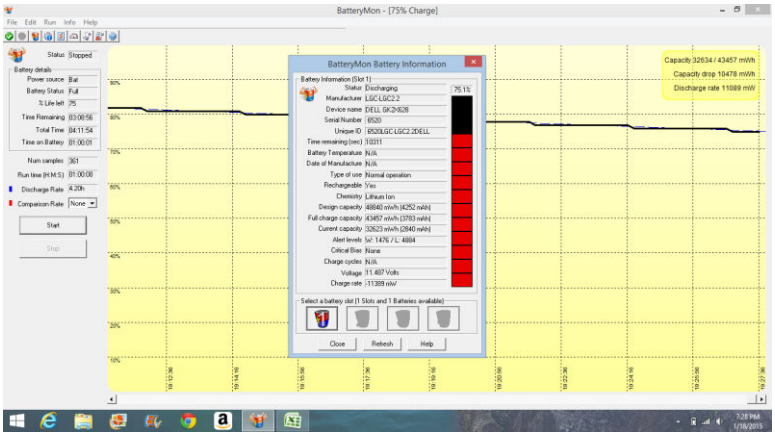
Test Case 1

Test Case 1						
1	100.00%	3783	43457	75.10%	2840	32623
2	100.00%	3778	43457	75.20%	2840	32667
3	100.00%	3615	41614	75.40%	2725	31369
Average	**100.00%**	**3725**	**42,842.67**	**75.23%**	**2802**	**32,220**

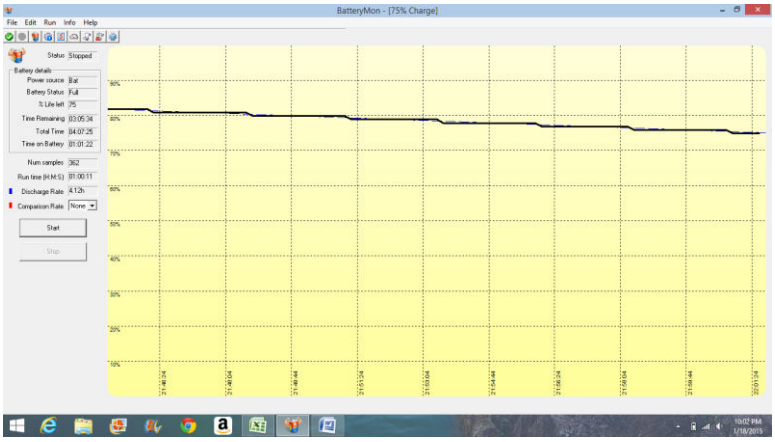
Test Case 1 was designed to test how much energy IRES takes in a one hour time period. In order to do this, IRES was set up with the same program that it normally uses, except that the cable I used only had power and ground connections, which enabled IRES to run for 60 minutes without any data transfer, meaning that power saving techniques would not be activated. In addition, while testing this, the computer's default power saving techniques were turned off. At the end of my trials, I found out that IRES only takes 8 mAh to operate, which is a minuscule number compared to the amount of power it takes to power a laptop for an hour. To arrive at this result, I used the change in (Δ) mAh from this test, and subtracted it from the ΔmAh from the control. This got me how much energy it takes to power IRES for 60 minutes.

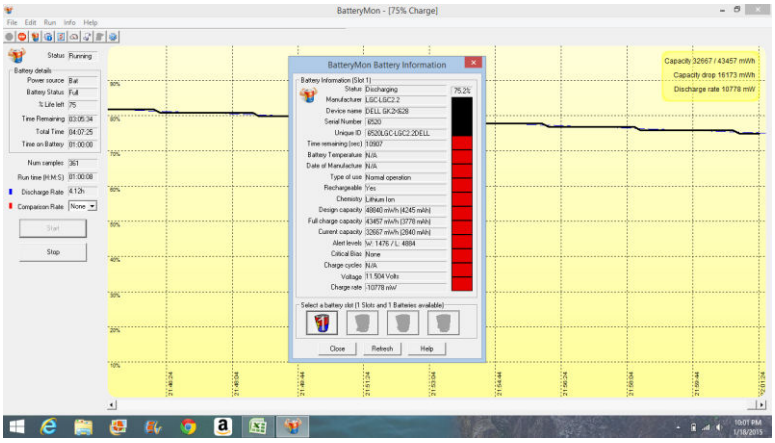
Test Case 1 – Trial 1



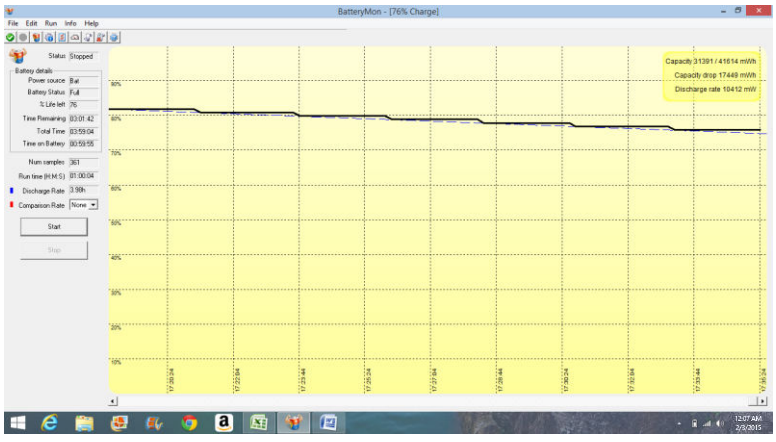


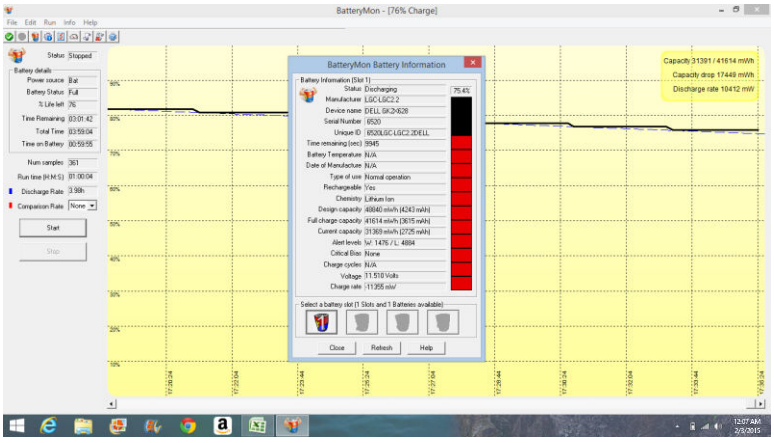
Test Case 1 – Trial 2





Test Case 1 – Trial 3



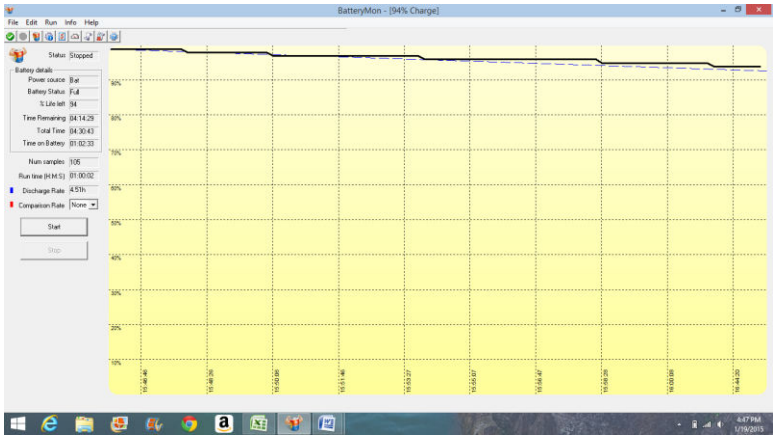


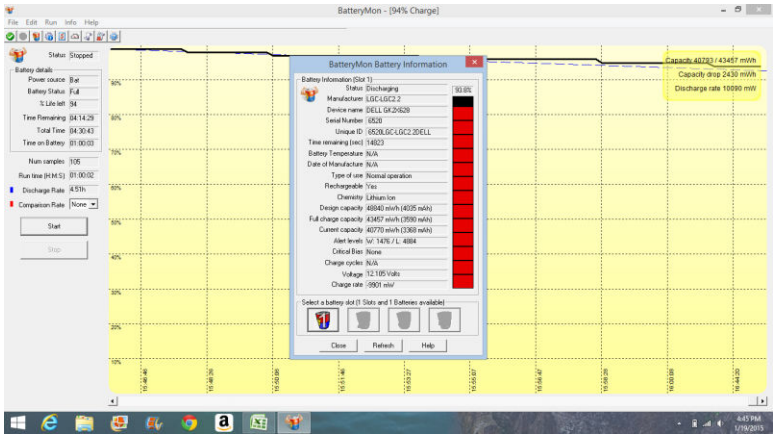
Test Case 2

Test Case 2						
1	100.00%	3590	43457	93.80%	3368	40770
2	100.00%	3611	43457	91.70%	3313	39871
3	100.00%	3665	43457	87.30%	3119	37929
Average	**100.00%**	**3622**	**43,457.00**	**90.93%**	**3266.67**	**39,523.33**

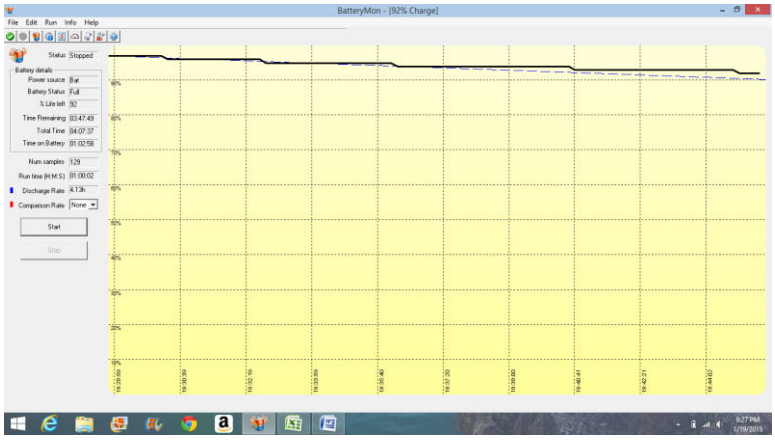
Test Case 2 was designed to test how efficient a computer is with default power saving techniques. With the default techniques enabled, the computer goes to sleep after 5 minutes of inactivity, and then 10 minutes after that (for a total of 15 minutes) the computer goes into a sleep mode, which is like the hibernate mode IRES uses, except that it does not turn the computer completely off, resulting in more power being used. After three 60 minute trials, I found out that it takes 355.33 mAh (ΔmAh from start to finish) to power my computer for 60 minutes with the default power saving techniques.

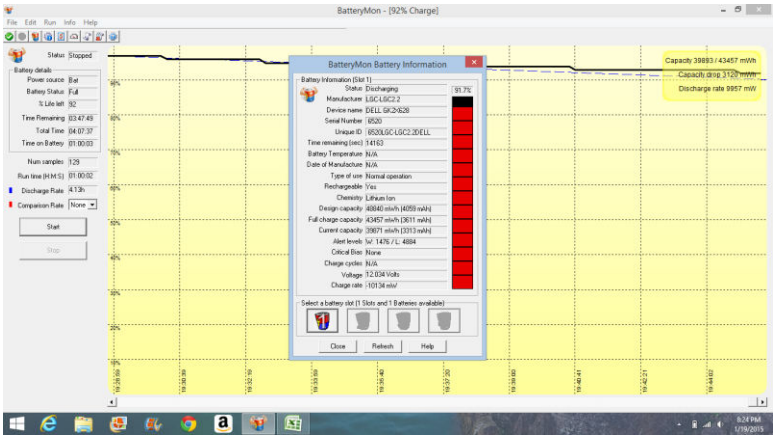
Test Case 2 – Trial 1



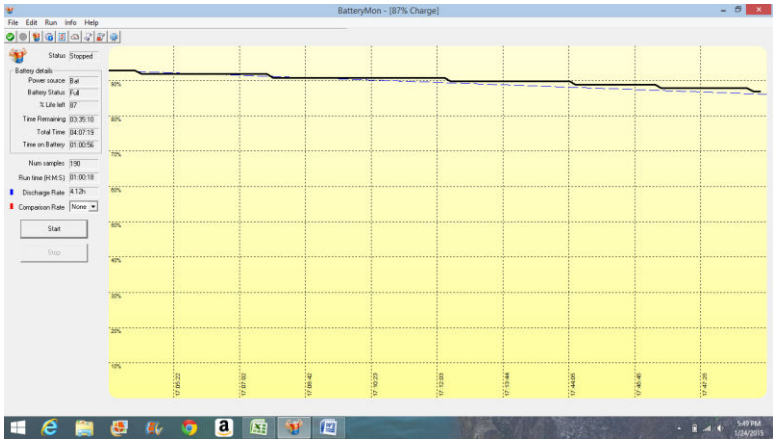


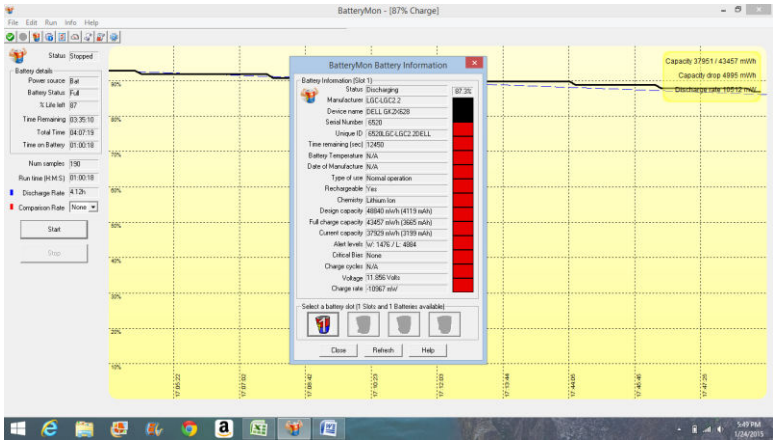
Test Case 2 – Trial 2





Test Case 2 – Trial 3



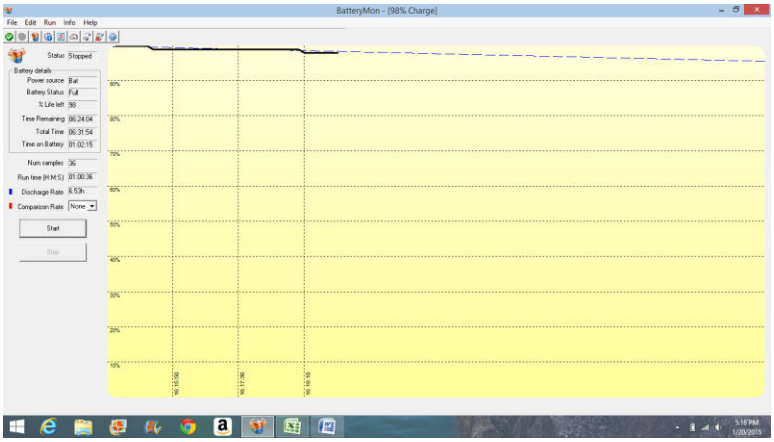


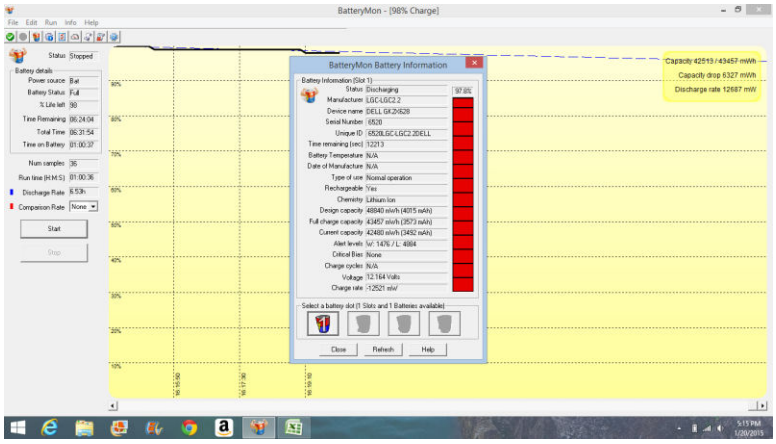
Test Case 3

Test Case 3						
1	100.00%	3573	43457	97.80%	3492	42480
2	100.00%	3561	43457	98.10%	3493	42624
3	100.00%	3561	43457	98.20%	3497	42680
Average	**100.00%**	**3565**	**43,457.00**	**98.03%**	**3494**	**42,594.67**

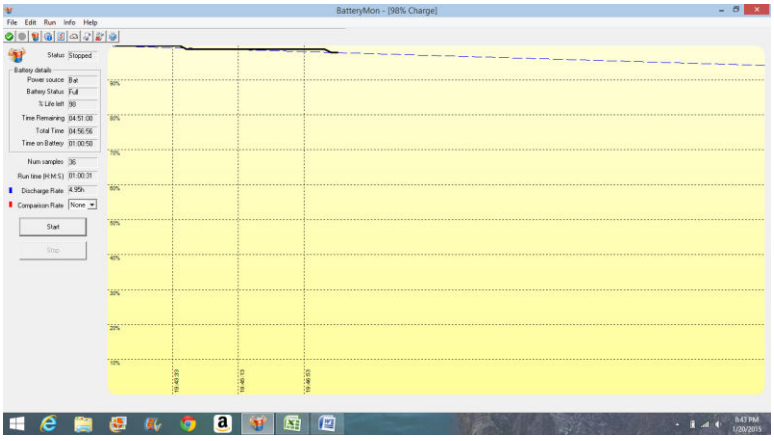
Test Case 3 was to test how efficient IRES is. In this test, the computer's default power saving techniques were turned off, so as not to interfere with IRES, and IRES and all of its components were enabled. After the three, 60 minute trials, I found out that using IRES, it takes only 71 mAh (ΔmAh from start to finish) to run a computer using IRES, showing that this new power saving solution is already significantly better than traditional power saving techniques.

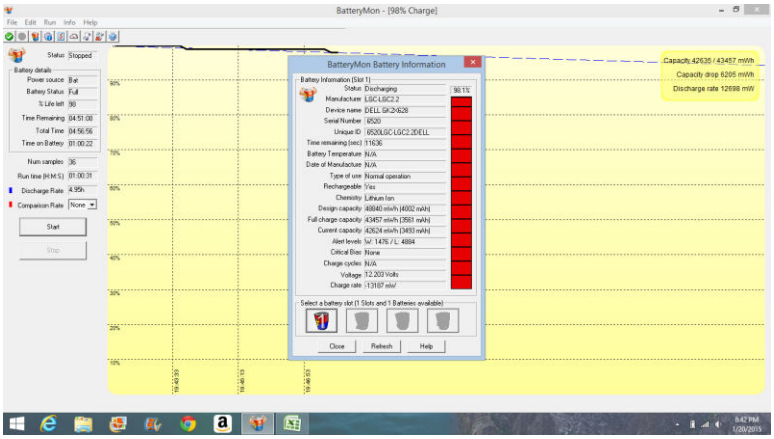
Test Case 3 – Trial 1



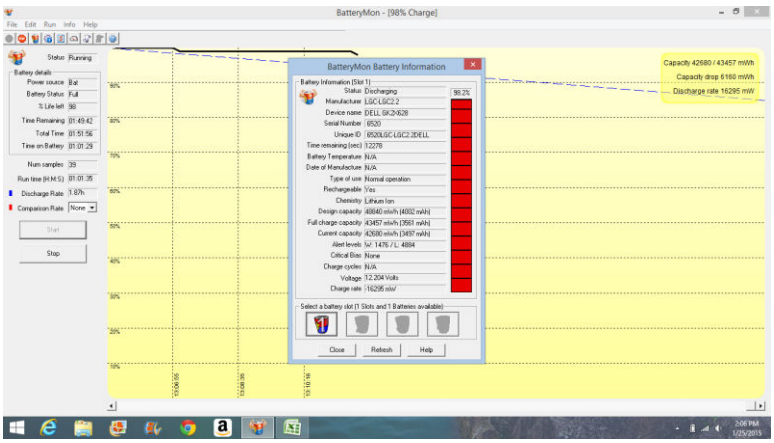


Test Case 3 – Trial 2





Test Case 3 – Trial 3

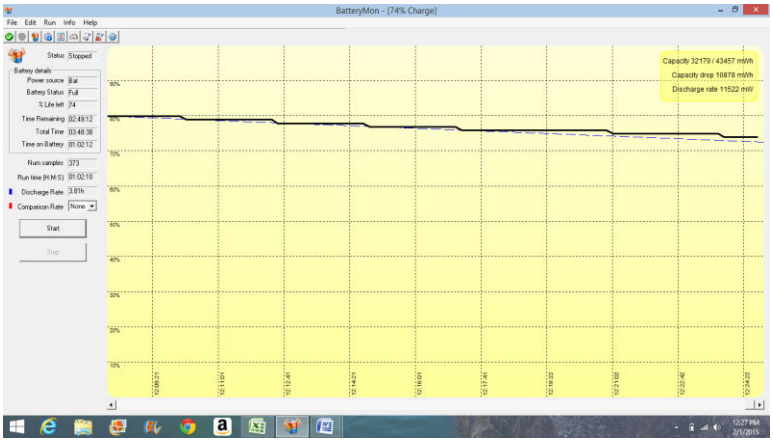


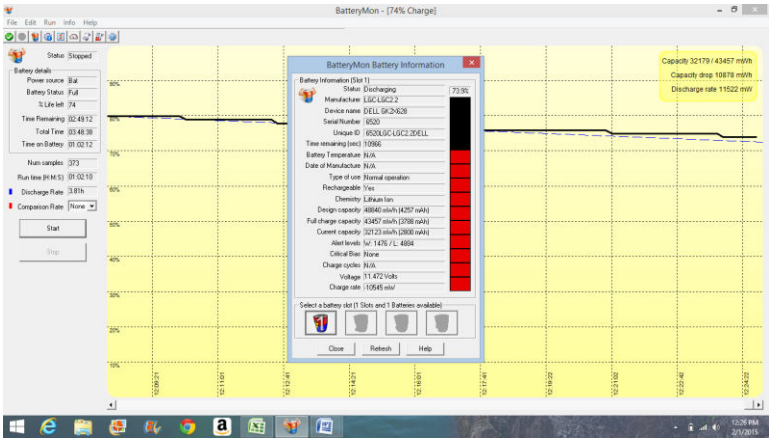
Test Case 4

Test Case 4						
1	100.00%	3788	43457	73.90%	2800	32123
2	100.00%	3802	43457	73.30%	2788	31168
3	100.00%	3629	41614	72.80%	2641	30292
Average	**100.00%**	**3739.67**	**42,842.67**	**73.33%**	**2743.00**	**31,194.33**

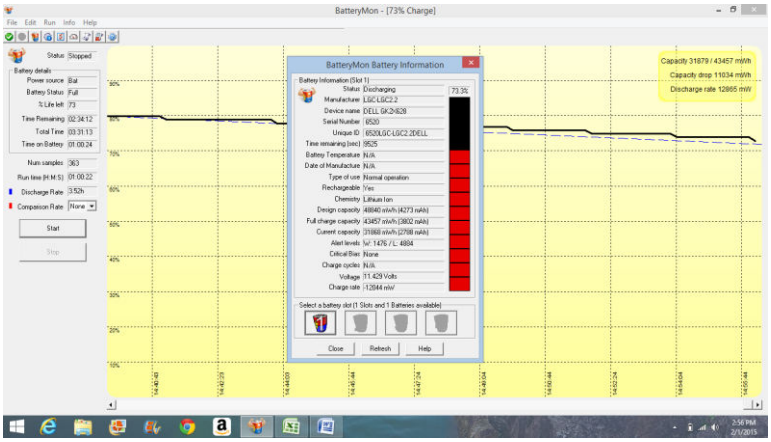
Test Case 4 was designed to test how efficient a laptop is with actual usage habits. Using the 15:15 work to break ratio and three, 60 minute trials, I determined that a laptop uses 988.67 mAh (ΔmAh from start to finish) of electricity (this amount is including the Arduino). The applications open in this were Microsoft Excel, BatteryMon, and Microsoft Word. The laptop's default power saving techniques are set to turn the screen off after 5 minutes of inactivity, and then put the laptop into a sleep mode after another 10 minutes of inactivity (for a total of 15 minutes). I used the laptop's default techniques because the average results from my survey were 5.33 minutes, and 12.33 minutes. The options on the laptop are in five minute increments, so for consistency, I chose the default settings. From this test I determined that it takes 988.67 mAh (ΔmAh from start to finish) of electricity to run a laptop with actual usage habits for 60 minutes.

Test Case 4 – Trial 1

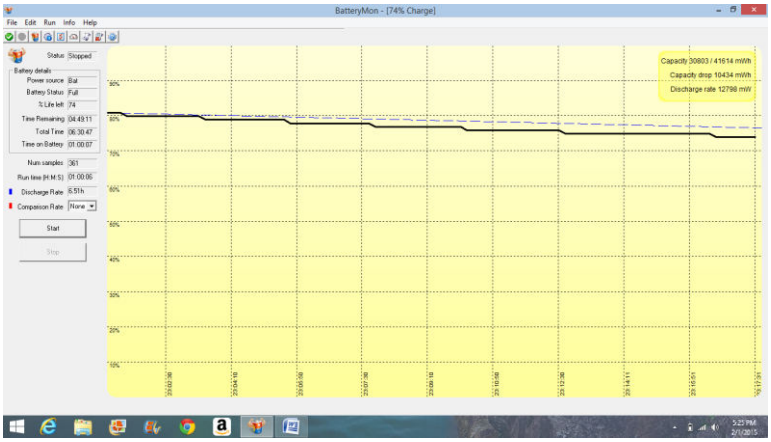


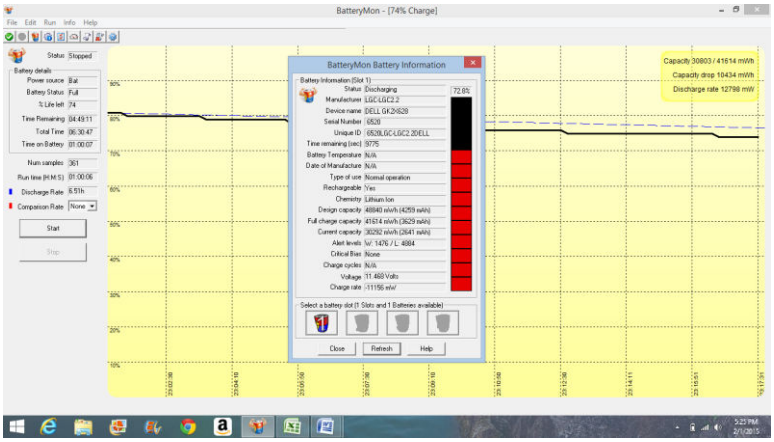


Test Case 4 – Trial 2



Test Case 4 – Trial 3



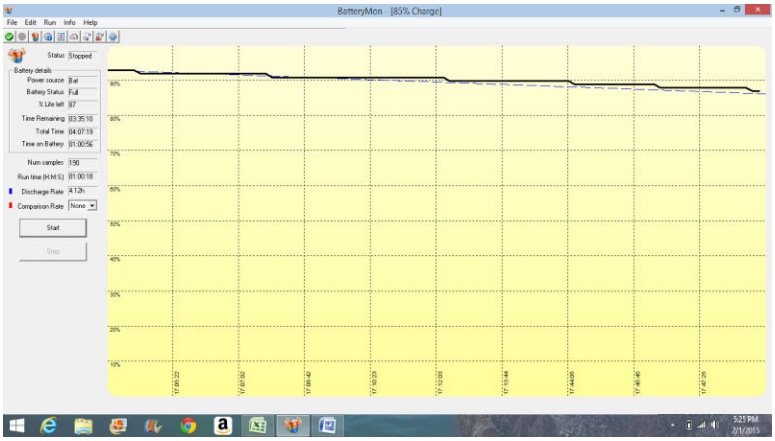


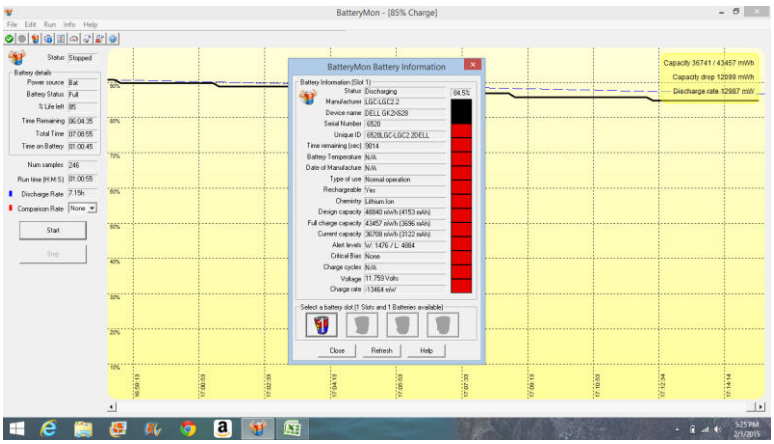
Test Case 5

Test Case 5						
1	100.00%	3696	43457	84.50%	3122	36708
2	100.00%	3692	43457	84.50%	3119	36719
3	100.00%	3517	41614	85.50%	3007	35587
Average	**100.00%**	**3635.00**	**42,842.67**	**84.83%**	**3082.67**	**36,338.00**

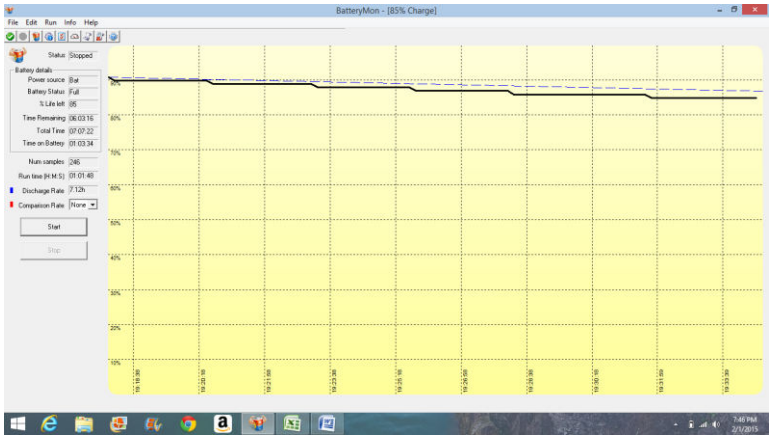
Test Case 5 was designed to test how efficient IRES is with actual usage habits. For this test, I used the same Arduino program as Test Case 4, except in this I disabled the computer's default power saving settings, and instead programmed to Arduino to initiate my power saving app after 15 minutes of work. Also, whenever the computer entered into hibernate mode, I had to manually restart it every thirty minutes because the Arduino was no longer on. Just like Test Case 4, it had the same applications running, Microsoft Excel, BatteryMon, and Microsoft Word, and like the other tests, it was also 60 minutes with three trials. Unlike Test Case 4, when I calculated the overall efficiency, I did not add the power of the Arduino back on because using the Arduino in this test just takes the place of using IRES. From this test I determined that it takes 552.33 mAh (ΔmAh from start to finish) of electricity to run a computer with IRES with actual usage for 60 minutes.

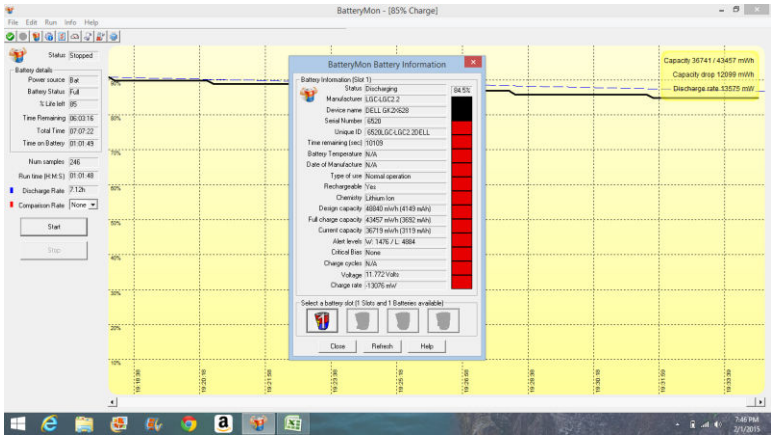
Test Case 5 – Trial 1



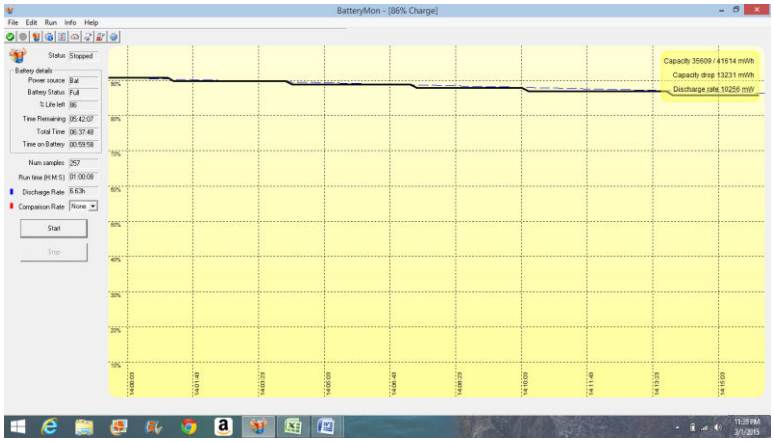


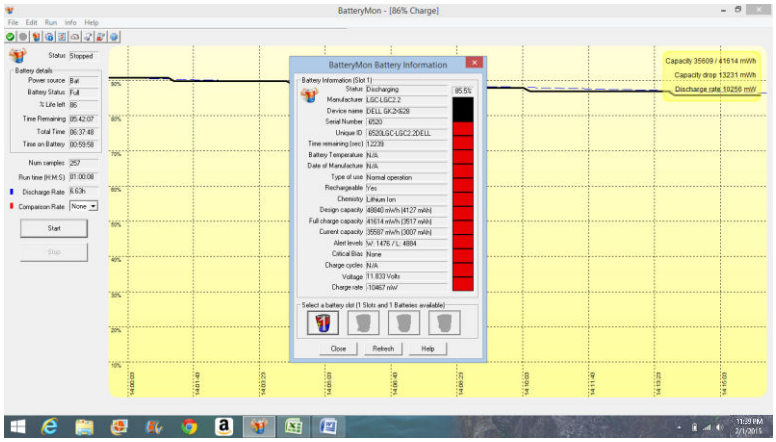
Test Case 5 – Trial 2





Test Case 5 – Trial 3



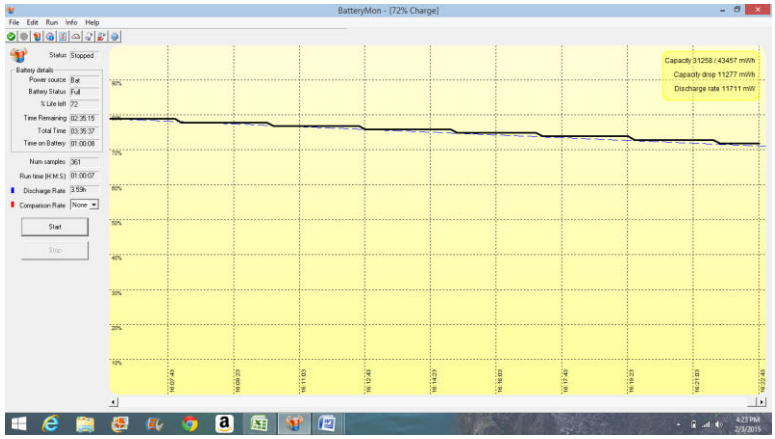


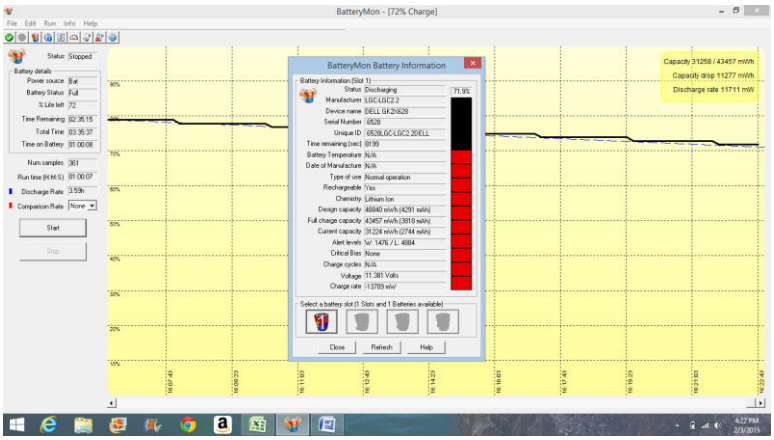
Test Case 6

Test Case 6						
1	100.00%	3818	43457	71.90%	2744	31224
2	100.00%	3793	43457	73.60%	2793	32001
3	100.00%	3652	41614	71.20%	2600	29626
Average	**100.00%**	**3754.33**	**42842.67**	**72.23%**	**2712.33**	**30950.33**

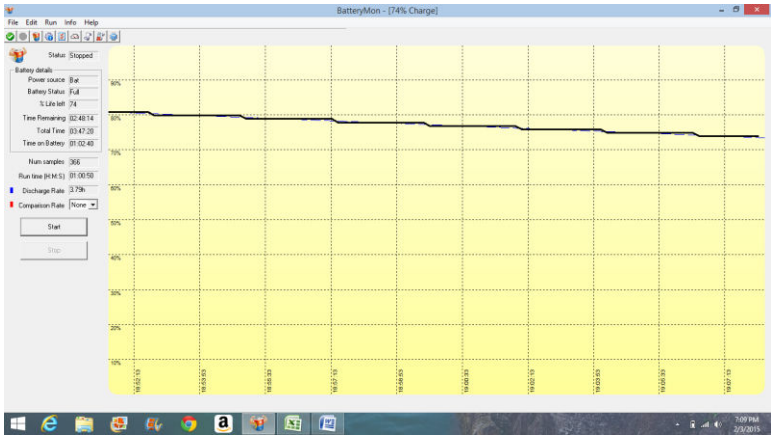
Test Case 6 was designed to test the efficiency of a desktop with actual usage. The usage was simulated using the same Arduino program used in Test Case 4, with the 15 minutes of work, writing “Power saving device test” proceeded by a 15 minute break. There were no power saving settings enabled for this because desktops do not require power saving, as they are connected directly into a power source, but when desktops do have power saving settings, they are usually just a screen saver that is activated around 30-40 minutes of inactivity, which is beyond the parameters for testing. Like the other Test Cases, there were three, 60 minute trials with Microsoft Excel, BatteryMon, and Microsoft Word open. Like Test 4, I will be adding the amount of energy the Arduino took during testing while determining overall efficiency. From this test, I determined that it takes 1034 mAh (ΔmAh from start to finish) of electricity to run a desktop for one hour with actual usage.

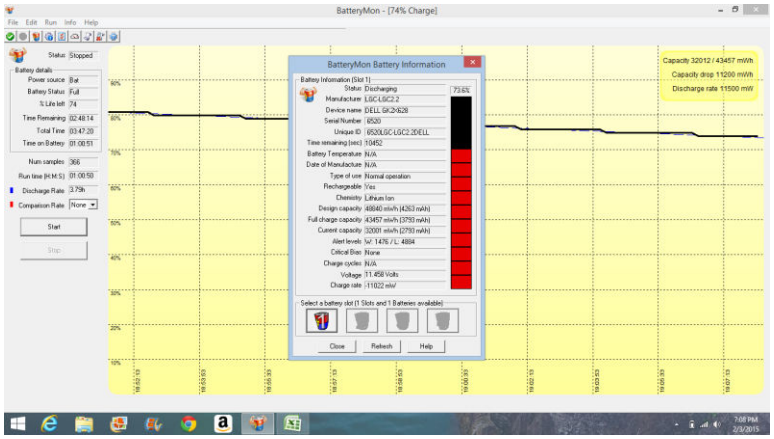
Test Case 6 – Trial 1



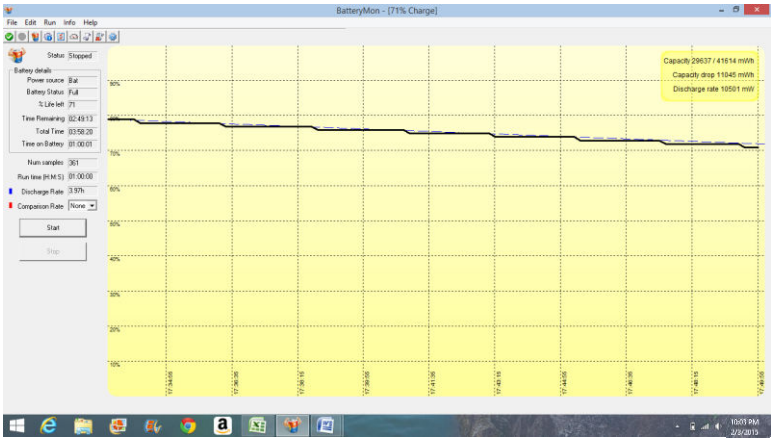


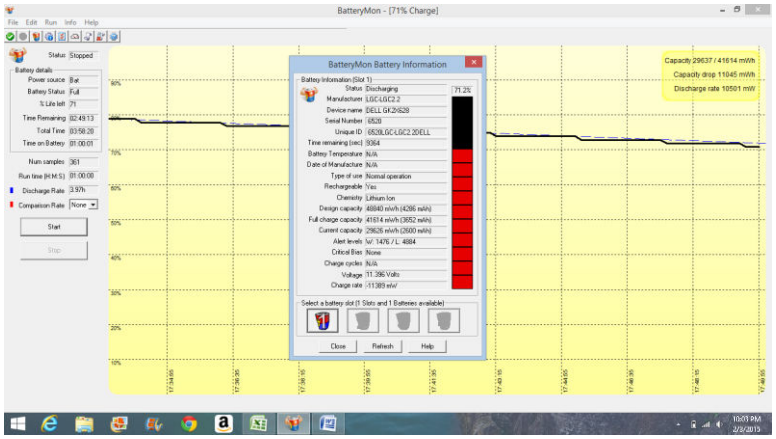
Test Case 6 – Trial 2





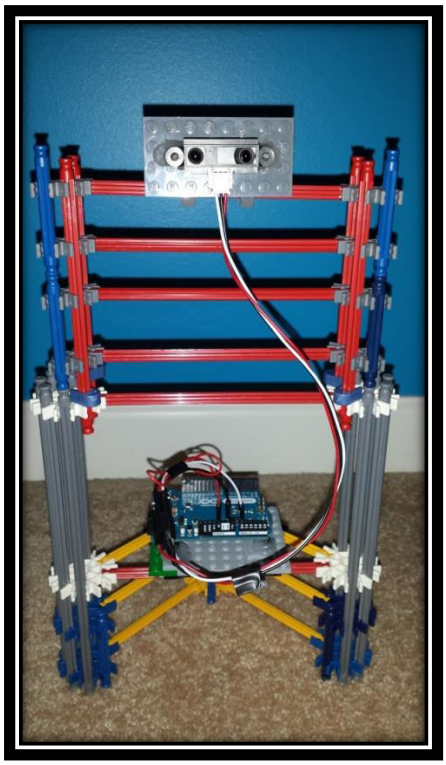
Test Case 6 – Trial 3



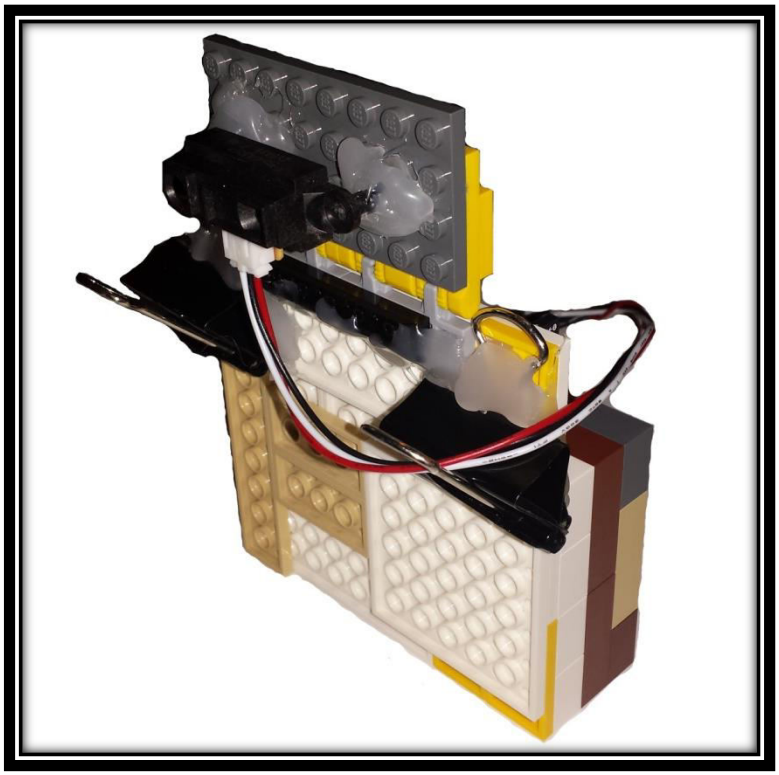


Pictures

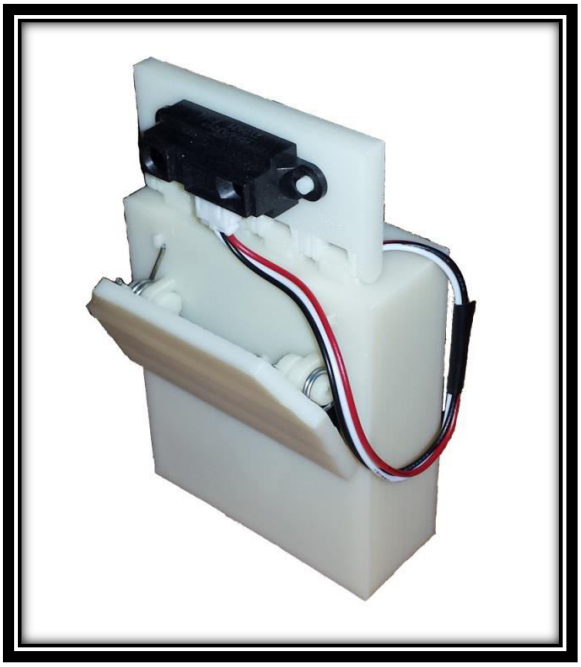
IRES Versions



IRES v1

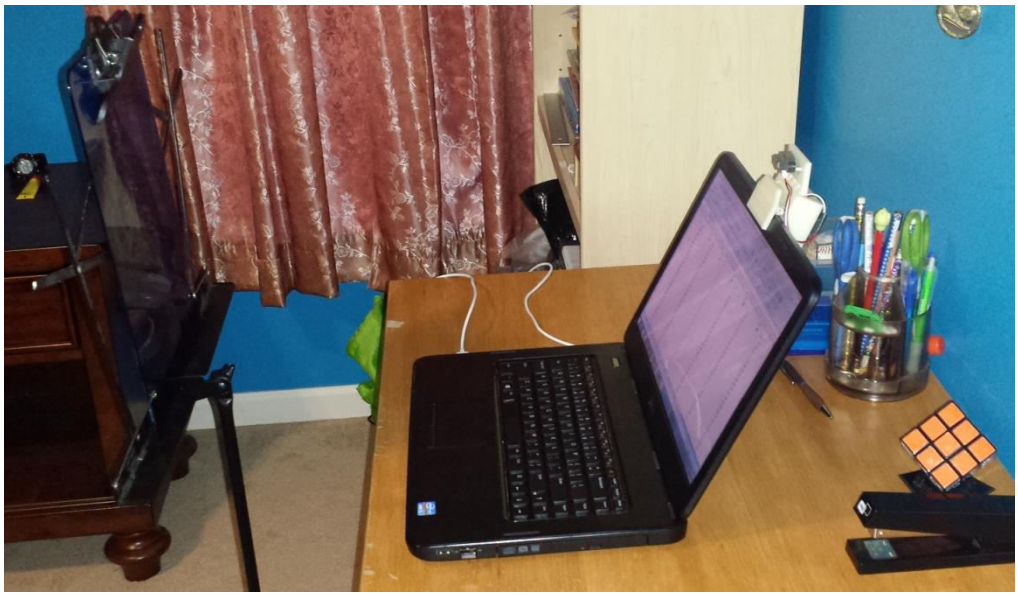


IRES v2

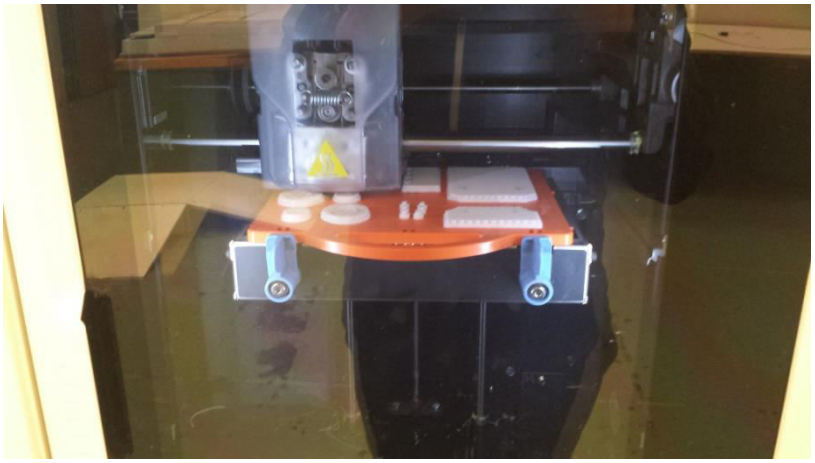


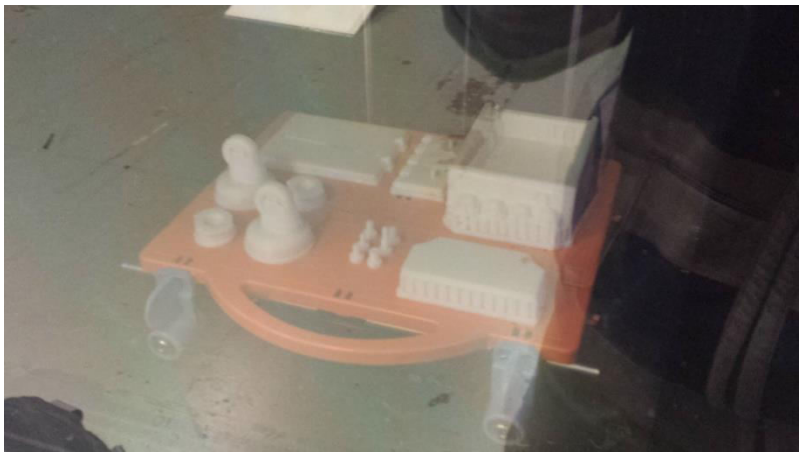
IRES v3

Converting Voltage to Distance



3D-Printer





uPrint 3D Printer

Sodium-Hydroxide Bath