Photo Sensing Circuit & Orientation Program: Operations Manual

Team:

Trenton Black

Esmael Abdullah

Dakota Hanks

Fahad Esmael

Andres Rodrigues

Date: July 24th, 2018

Client: Dr. Michael Shafer

Instructor: Dr. Sarah Oman

Graduate Teaching Assistant: Amy Swartz

Table of Contents

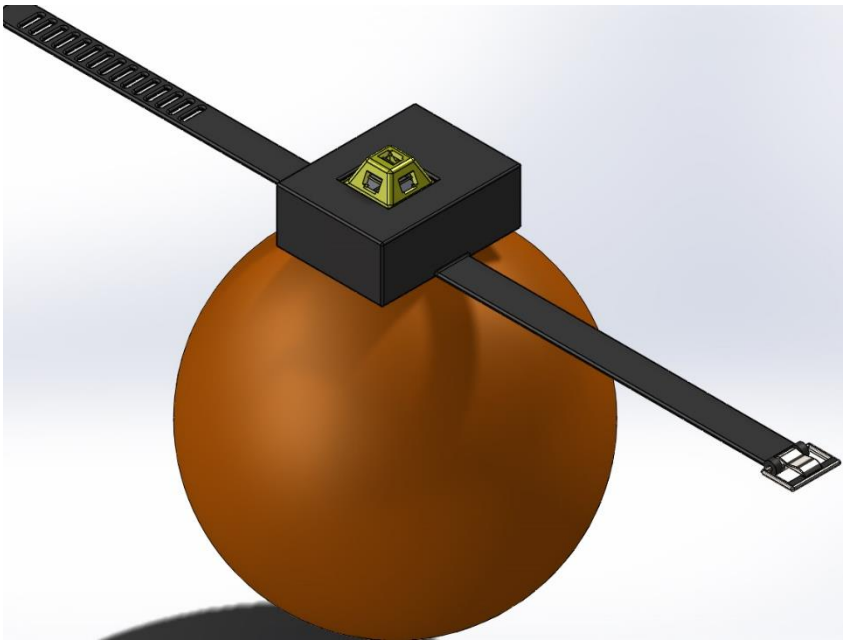


Figure 1: Size Comparison of Final Assembly to a Full-Size Men's basketball 9.5in in Diameter

1 Introduction

The following manual instructs proper set-up, operation and data processing into the simulation program for the photo-orientation-circuit. The Calibration section will outline set-up protocol for establishing an origin with the circuit for clear results translation by the simulation afterward. Operations discusses the physical execution of the code inside of the Arduino, although all processes are run by a script, understanding code initiation, watchdog timing intervals and data retrieval are essential. To conclude, directions on transferring the data from Arduino into the MATLAB environment and how MATLAB indexes the data for reference will be provided.

2 Assembly

Here a brief description of the components required for assembly and instruction on assembly is provided. A diagram of the final design with number labeling balloon that will provide reference in the text for clear part identification and assembly.

Parts

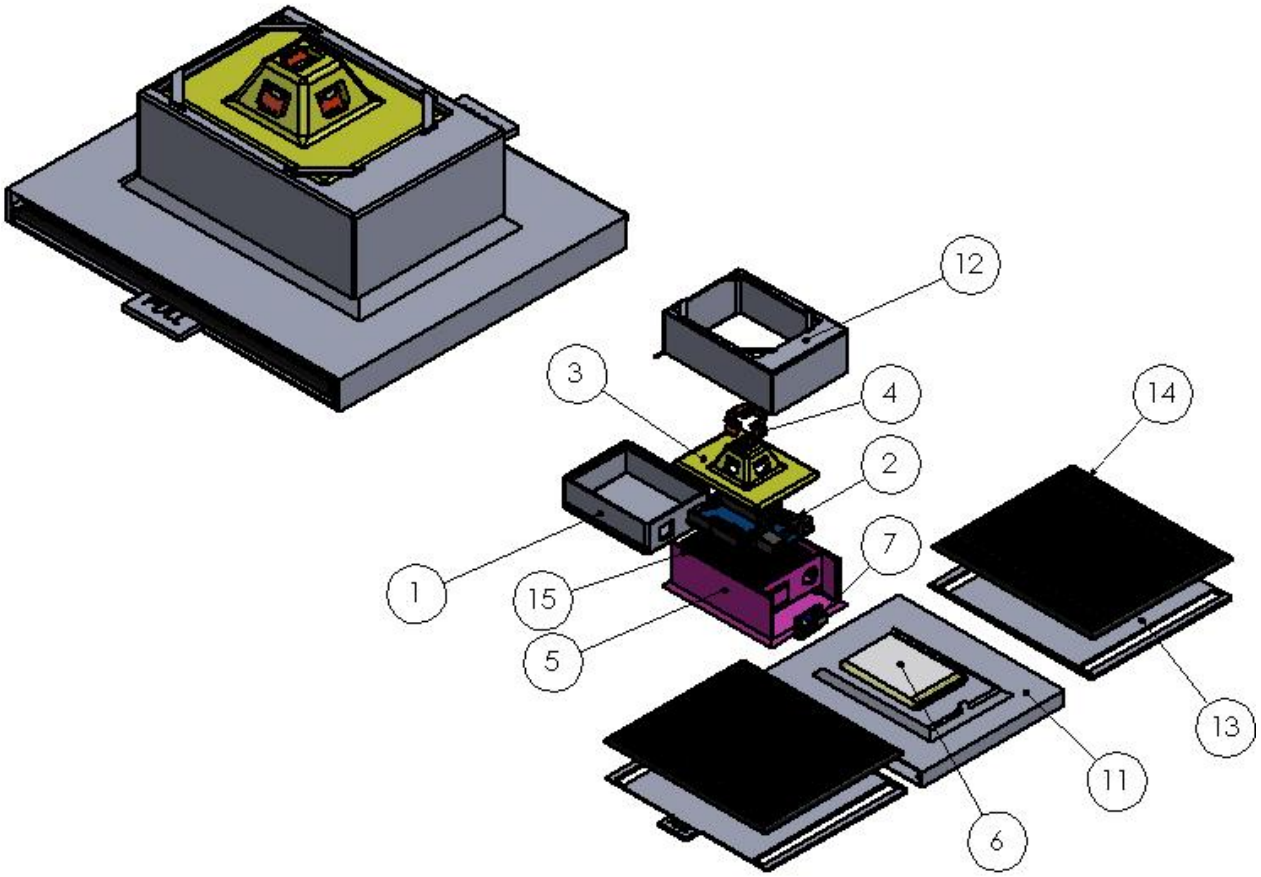
Although all assembly parts are provided with description in figure 2. There are various materials and smaller subsystems used in the manufacturing of the product are delivered below.

Circuit Components

- 22-gauge wire (A)
- 2x Solderable Bread Boards (B)
- 1x 10k Ohm Resistor (C)
- 2x 10μF Capacitors (D)
- 1x N-CP MOSTFET (E)
- 1x Micro USB (F)
- 1x 5.5mm Barrel Jack (G)

Simple Photo Circuit

1. Slip the Arduino Uno R3(2) into the housing(1) leading the computer jack through the square hole first and settling it into place inside the housing which will align the barrel jack(G) power supply to a secondary port.
2. Slip the wired ambient light sensors(4) into the 65° array(3), bottom first then apply pressure to the upper portion providing a slight interference fit.
3. Before installing the array(3) lid into the Arduino housing(1) configure the position of all the wires(A) fitting them to the Arduino serial connection pins, remove the Arduino(2) and place the wires securely into the wire slotting board(15) and push the board into the array(3) for a snug fit.
4. Align the array(3) with the Arduino(2) housing by geometry and the placement of serial pins over the Arduino and slide it into the housing making sure the bottom of the lid is securely mated to the top of the Arduino Housing(1).



ITEM NO.	PART FILE NAME	DESCRIPTION	QTY.
1	Arduino Housing	Arduino Housing 3x3x1.13	1
2	Arduino UNO	Arduino Uno R3 2.7x2.1x.59	1
3	Array3_65	Solar Array at 65D 3x3x.68	1
4	Ambient Light Sensor	Ambient Light Sensor .39x.39x.07	5
5	Battery_Tray	Battery Tray 4.4x3.9x2.0	1
6	Li-Po_Batt	Lithium Ion Polymer Battery 3.7V 2Ah 2.76x1.94x.22	1
7	ARDUINO MINI LIPO USB 1905	Adafruit Li-Po Charger 0.93x0.7x0.56	1
11	Solar Tray	Solar Tray 6.13 x 5.25 x .86 in	1
12	Mounting_Cover(2)	Solar & Battery Tray Cover 4.26 x 1.56 x 3.0 in	1
13	Panel Tray	Panel Tray 5.95 x 5.58 x .1	2
14	Solar_Panel(2)	ALLPOWERS 2.5W 5V .5Ah 5.91x5.12x.12	2
15	Wire Slot	Wire Slot Grid 2.73 x 2.0 x .08	1

Figure 2: Exploded Assembly with numerically identified parts

Battery & Solar Power Supply

5. Connect the Battery(6) to the Li-Po charger(7) using the pale white clip connection.
6. Now the solar tray(11) is ready to be slid into place using the lower edge of the battery tray(5), slide the pieces together until the back of battery tray is securely against the closed end of the solar tray's flange.
7. Place the solar panels(14) in the panel trays(13) and make sure to run the wire through the long holes running the length of the tray.
8. The board of the Adafruit Li-Po charger(7) will have 4 holes labelled 5V, GND, and Batt. These holes correspond to positive 5v output, ground, ground, and Voltage/ current input. Ensure that the positive end of the solar cells connect to the hole labelled Batt and the negative side to the GND hole closest to the Batt hole.
   (NOTE: This may be difficult to do, utilizing a piece of string or stiff wire as a guide to pull it through would be beneficial.)
9. Then connect a wire(A) from the 5V output of the li-po charger to a 5V input pin of the Arduino(2). Also connect the GND of the li-po charger(7) to the -V or GND input of the Arduino(2).
10. Use a multimeter to test is current is flowing to the Arduino. An analog switch can be added before the solar cells to manually stop charging. (NOTE. User will know that battery(6) is charging if the green led is on.)
11. Now the solar tray(11) is ready to be slid into place using the lower edge of the battery tray(5), slide the pieces together until the back of battery tray is securely against the closed end of the solar tray's flange. (Make sure all wires are safely inside the battery compartment)
12. Place the barrel jack from the Li-Po circuit in the round hole of the battery tray(5).
13. Slide the Arduino housing(1) into the battery tray(5) using the extruded runners to guide a secure fit ensuring proper mating of the Arduino's computer connection to the protected hole and the barrel jack(G) power supply to coincident hole in the battery tray.
14. Now that the circuit is complete, a covering will be equipped over the entire assembly locking into position the Solar Array(3), Arduino Housing(1), Battery Tray(5) and Solar Tray(11).
15. Finally, place the panel trays(13) into their slots on either side and return them to a resting (compressed) position wherein the unit attains its smallest volume.

3 Calibration

The code set with a specific set of parameters for identifying each of the solar sensors equipped based on their respective angular positions as a result it is important to align the device to a specific known orientation to begin all proceeding measurement from to collect organized data the following instructions will provide the proper procedure.

Arduino: Solar Azimuth & Zenith Calibration

1. Connect the Arduino Uno R3 to your computer using the USB connection.
2. Open the Arduino program interface command window.
3. Before, running the program, set the light sensing device on a level surface, using a flash light shine the light directly at a corner between two sensors roughly 45º from horizontal as well as from the diagonal sensors. (NOTE: Make sure at least 3 sensors are illuminated)

4. As you move the light around the sensor, note the points at which the azimuth is zero and the Zenith is maximum at 90º
5. Add or subtract from each reading until you are getting accurate reading the device.
6. Stop the run and clear the serial monitor, before running a real test adjust the pause inside the loop to increase or decrease the amount of data computed.
7. Now that your program is calculating correctly, double check the onboard time dependent calculator, input your time of day and quickly run-stop the program to receive a value. Double check the result on https://www.esrl.noaa.gov/gmd/grad/solcalc/azel.html using their solar position calculator for validation.
8. Now you are ready for outdoor calibration.
9. Set the device on a level surface pointing the previously noted 0º sensor towards relative south, begin a quick run-stop to collect a maximum of five data points.
10. If the results are within ±10º then the calculator will be able to accurately track orientation by logging the position of the sun in the background.

Arduino: Watch Dog Timer Calibration

The watchdog program, is intended to allow the Arduino to go into power-saving or a version of sleep-mode while checking the inputs for any variations that would require the Arduino to wake. As such, the watchdog will be set to sleep during night operations, and awake in the mornings, with checks every 8 seconds to make sure that no inputs go unnoticed. This carries over into the daytime, since the sun does not move quickly, it was suggested by the team that 8 second intervals for gathering data would suffice to create an 'orientation profile'. Here is how you can customize the watch dog timer to your specifications the Code is shown in figure 4.

1. Open the Arduino sleep program.
2. At the beginning of the if statement after the delay change the 20 to the desired time to sleep. The value must be in military time.
3. Next after the for loop change the value to the time you would like the Arduino to wake up.
4. Find the number of seconds between to two times that you had set then divide that number by 8.
5. Take the number that was calculated in step four and insert that number in the line that starts the for loop equal or greater then I.

time_sleep

```
void myWatchdogEnable(const byte interval)
  {
  MCUSR = 0;                          // reset various flags
  WDTCSR |= 0b00011000;               // see docs, set WDCE, WDE
  WDTCSR =  0b01000000 | interval;    // set WDIE, and appropriate delay

  wdt_reset();
  set_sleep_mode (SLEEP_MODE_PWR_DOWN);
  sleep_mode();            // now goes to Sleep and waits for the interrupt
  }


void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
}

void loop() {
  Serial.print(hour);
  Serial.print(":");
  Serial.print(minute);
  Serial.print(":");
  Serial.println(sec);
  sec = sec + 1;
  if (sec == 60)
  {
    minute = minute + 1;
    sec = 0;
  }
  if (minute == 60)
  {
    hour = hour + 1;
    minute = 0;
  }
  delay (1000);
  if (hour == 20)
  {
    for (int i=0; i <= 5399; i++) //sleeps for 12 hours
    {

      myWatchdogEnable (0b100001);  // 8 seconds
    }
    hour = 8;
  }
```

Figure 4: Watchdog operations algorithm

MATLAB: Simulation Calibration

Calibrating the MATLAB code to ensure proper function and no breaks in the code from faulty values is relatively straightforward and simple. Create a list of values whose results are easily known and look for the correct outputs and the resulting visual translation of the outputs into movement in the 3D plot. Figure 6, represent the basic lines of code that will be observed and manipulated to perform the calibration.

1. Under '%Measured Zenith and Azimuth Angles' there is an option to either have randomly generated values between the operating bounds of 90-360^{o} or to manually input desired values. Input a single value or an array for the values [45 60 90 135 180 15 360]
2. Likewise, repeat step one for '%Calculated Zenith and Azimuth Angles' [360 15 180 135 90 60 45]
3. No run script, the out puts thereof should correspond to rotations
   (Insert Image of Command Line with outputs)

4. With movements of the 3D model such moving in the series of:
   (Insert Image series of screen shots of 3D plot moving for each orientation)
5. Compare the results of the code to the results discussed in steps 4 & 5.

```
%First Loop
if j==k
%Operating Bounds
a = 0;
bz = 90;
ba = 360;
%Measured Zenith and Azimuth angles
Zm = (bz-a).*rand(1,1) + a;
Am = (ba-a).*rand(1,1) + a;
%Zm = 90;
%Am = 0;

%Calculated Zenith and Azimuth angles
Za = (bz-a).*rand(1,1) + a;
Aa = (ba-a).*rand(1,1) + a;
%Za = 90;
%Aa = 45;
 end

 %All other loops
 if j<k

%Measured Zenith and Azimuth angles
Zm = Za;
Am = Aa;

%Calculated Zenith and Azimuth angles
Za = (bz-a).*rand(1,1) + a;
Aa = (ba-a).*rand(1,1) + a;
 end
```

Figure 5: MATLAB calibration code for manual inputs

4 Operation

It is important to know how to turn the device on, how the code sequence will operate while circulating inside of the loop, and what problems may occur during operation that will cause errors and breaks inside the loops. As such the built-in timing function Watchdog, will operate as a power saving mode throughout the day to conserve power and take measurements are a rate consistent with collecting enough information for a profile cataloguing the orientation of the object as it is rotated about the X, Y and Z axis.

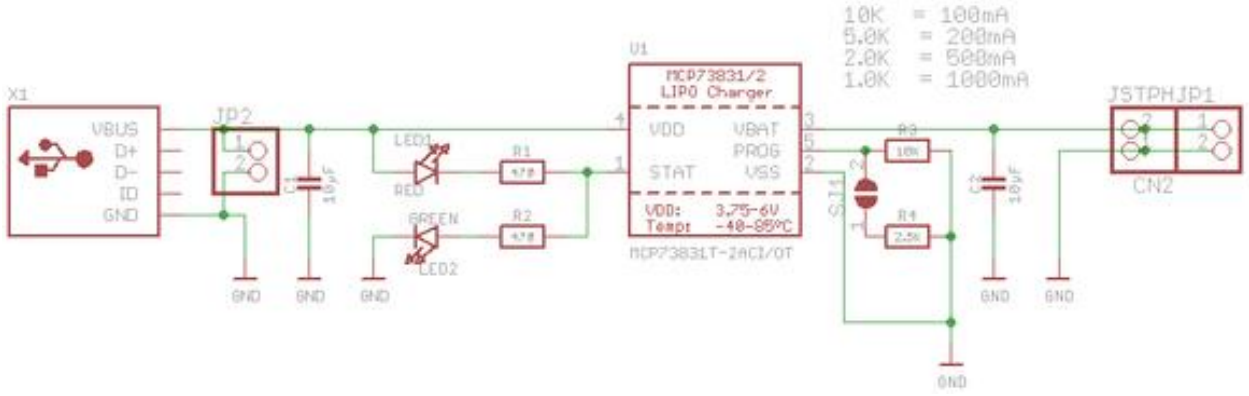


Figure 6: Solar Battery charging and bypass circuit schematic

5 Data Processing

Although, the way the device currently transmits data is not technically a form of collection, the data will be compiled via the serial monitor of the Arduino. The data output is tailored to address the required inputs of the simulation program, such that once the data from the serial monitor is exported via .txt file and uploaded to MATLAB the information provided can be directly processed by MATLAB.

Export Arduino Serial Data .txt File

1. Run the Arduino program and open the serial monitor.
2. Terminate the program while leaving the serial monitor open.
3. Copy the data that is inside the serial monitor and paste it into a txt file.

Import .txt Serial Data to MATLAB

1. Verify your file name string inside of the apostrophe for variable 'Filename'. Shown in Figure 7.
2. Make sure the .txt file is in the same folder location as the Complete Animation Program.

```
%Import Arduino .txt Data
Filename = 'Test_Data.txt';
Data = import(Filename);

%Origin
O = [5 5 0];

%Data Counter
k = numel(Data);
j= k+1;
n=1;

while j > 0
```

Figure 7: Import data .txt file into MATLAB pre-delimited from the Arduino Serial monitor

3. Make sure the solar data corresponding to the solar Azimuth and Zenith is being properly indexed from data the first row is zenith data and the second is azimuth data. Designated by the first number in parenthesis (1,#) & (2,#) illustrated by Figure 8.

```
while j > 0

 %Start View
 Zm = 45;
 Am = 0;

 %Arduino Data Arrays
 Za = Data(1,1:k);
 Aa = Data(2,1:k);
```

Figure 8: Data array indexs outside of If statements

4. The second if statement references the information on the solar calculator to build an initial profile of the orientation from the sun, since it is assumed the sun moves very slowly through the sky with respect to several seconds it is only used once for comparison. Designated by the difference in suffix of (m & a) measured and actual in figure 9.

```
%Second Loop
if j==k
%Operating Bounds
a = 0;
bz = 90;
ba = 360;

%Measured Zenith and Azimuth angles
%Zm = (bz-a).*rand(1,1) + a;
%Am = (ba-a).*rand(1,1) + a;
Zm = Zm(1,1);
Am = Am(1,1);

%Calculated Zenith and Azimuth angles
%Za = (bz-a).*rand(1,1) + a;
%Aa = (ba-a).*rand(1,1) + a;
Za = Za(1,1);
Aa = Aa(1,1);
 end
```

Figure 9: The final IF statement will run through the rest of the data points changing to the new orientation and saving the previous using single rowed data arrays.

5. The third if statement is different from the first and second since it begins to save the previous orientation as the second vector that will be used to compare the new orientation and determine the movement that has elapsed. As shown in Figure 10. the counter (n) increases by one each time thus moving to the next data column on the next iteration of the while loop.

```
 %All other loops
 if j<k

%Measured Zenith and Azimuth angles
Zm = Za;
Am = Aa;

%Calculated Zenith and Azimuth angles
%Za = (bz-a).*rand(1,1) + a;
%Aa = (ba-a).*rand(1,1) + a;

Za = Za(1,n+1);
Aa = Aa(1,n+1);
```

Figure 10:Calling data from outside of the data index using the variable names from the original program.

6 APPENDIX A: 3D Printed Circuit Component Housings

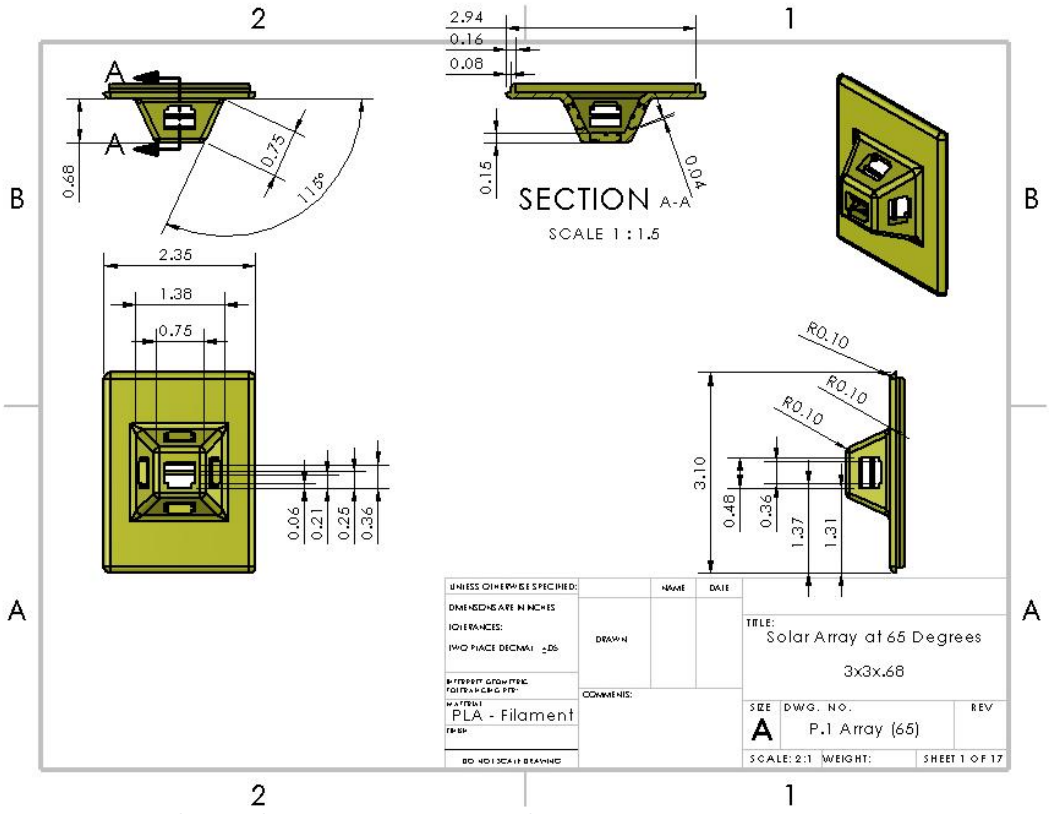


Figure 11: Ambient Light Sensor Array mounting plate.

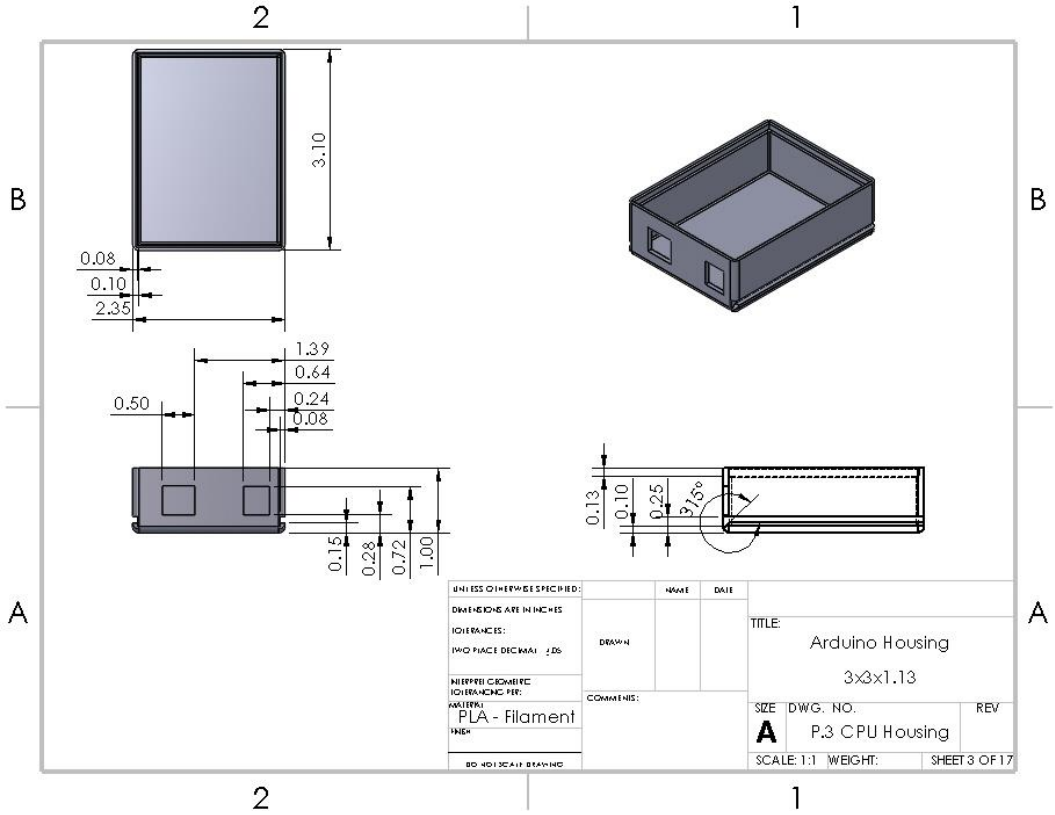


Figure 12: Arduino Housing

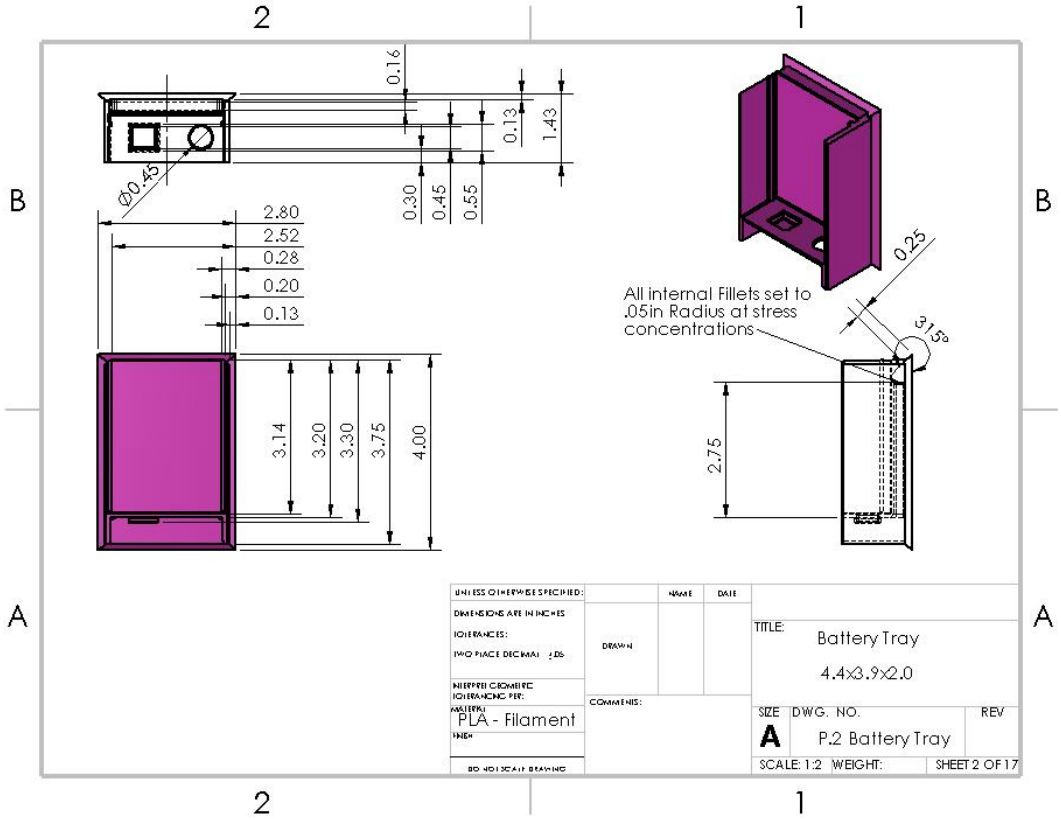


Figure 13: Battery Tray, for storing and housing the battery and Li-Po charger including the facilitation of Sub-Assemblies.

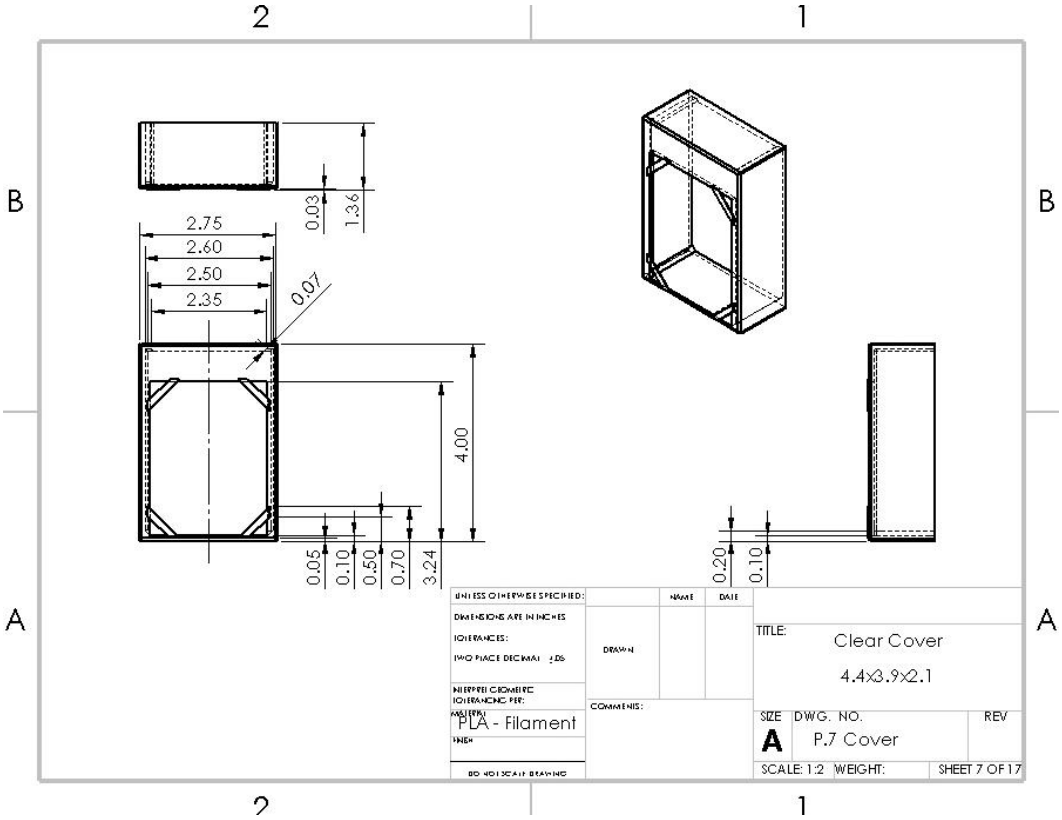


Figure 14: Simple Covering for Battery Tray and Photo-Sensory circuit.

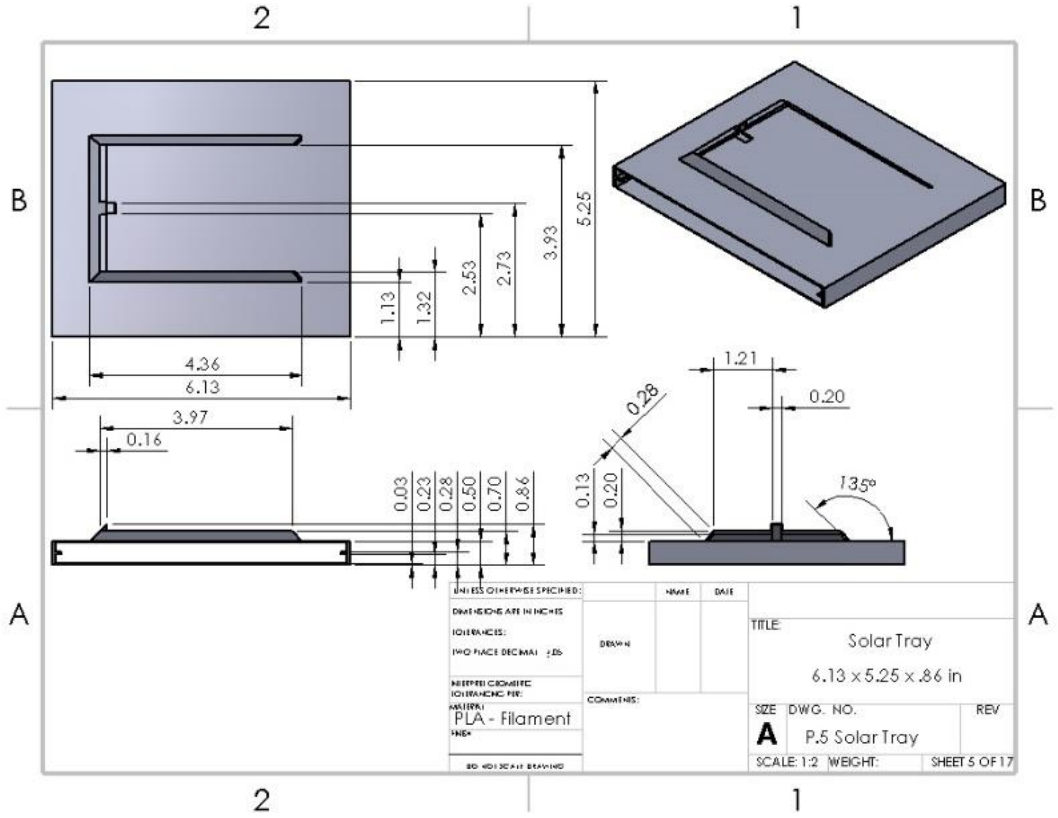


Figure 15: Solar Tray, will attach to the bottom of the Battery tray and will hold and facilitate connection between solar panels and Li-Po charger as well as low volume Panel storage.

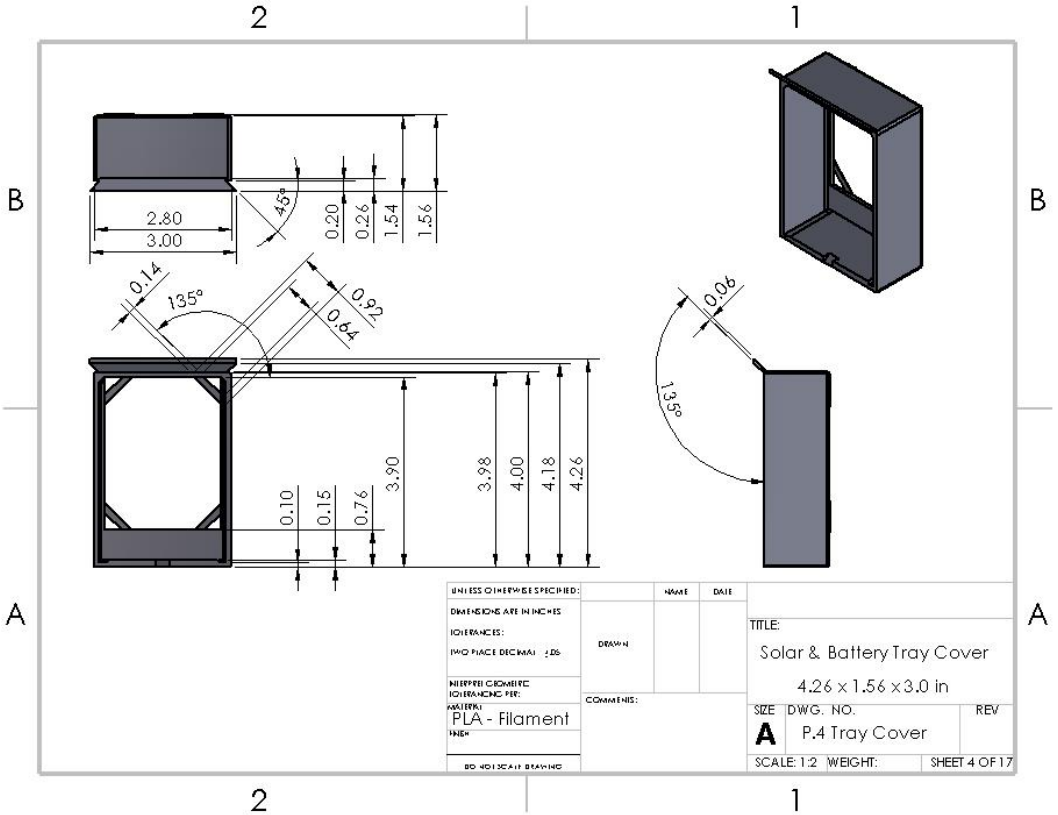


Figure 16: Solar Covering, Fixes the Position of the Photo-Sensing Circuitry housing, Battery Tray and Solar Tray.

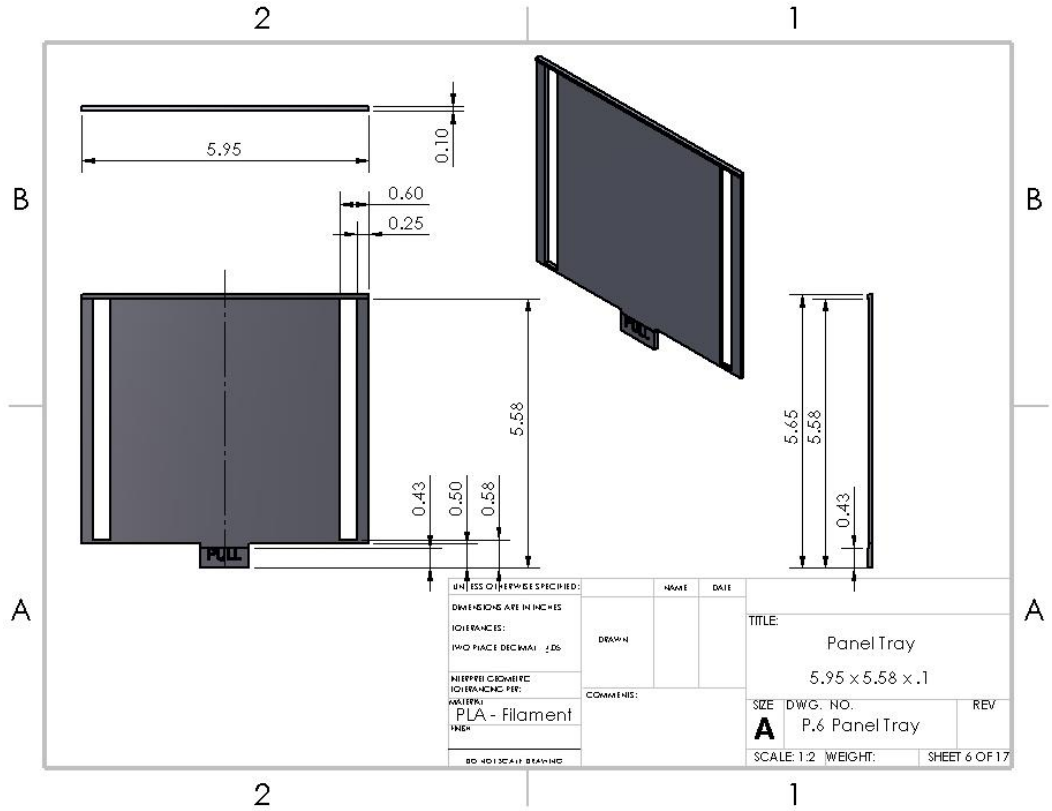


Figure 17: Panel Tray, these will hold the Solar Panels for easy accessibility and between storage and deployment.

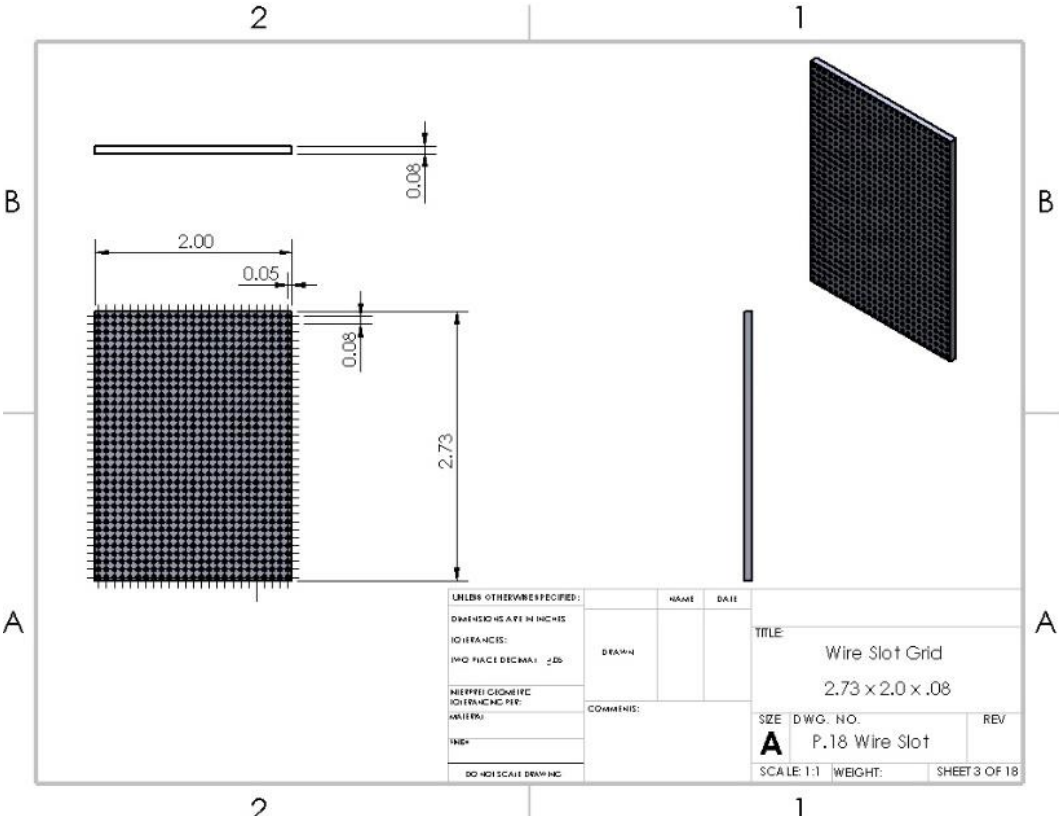


Figure 18: Universal Wire slotter, making good serial connections to the Arduino from the Ambient Light Sensors is difficult, there for you can preset the position of your wires from the Ambient Light sensors for simple plug-n-play usage.

7 APPENDIX B: Circuit Components

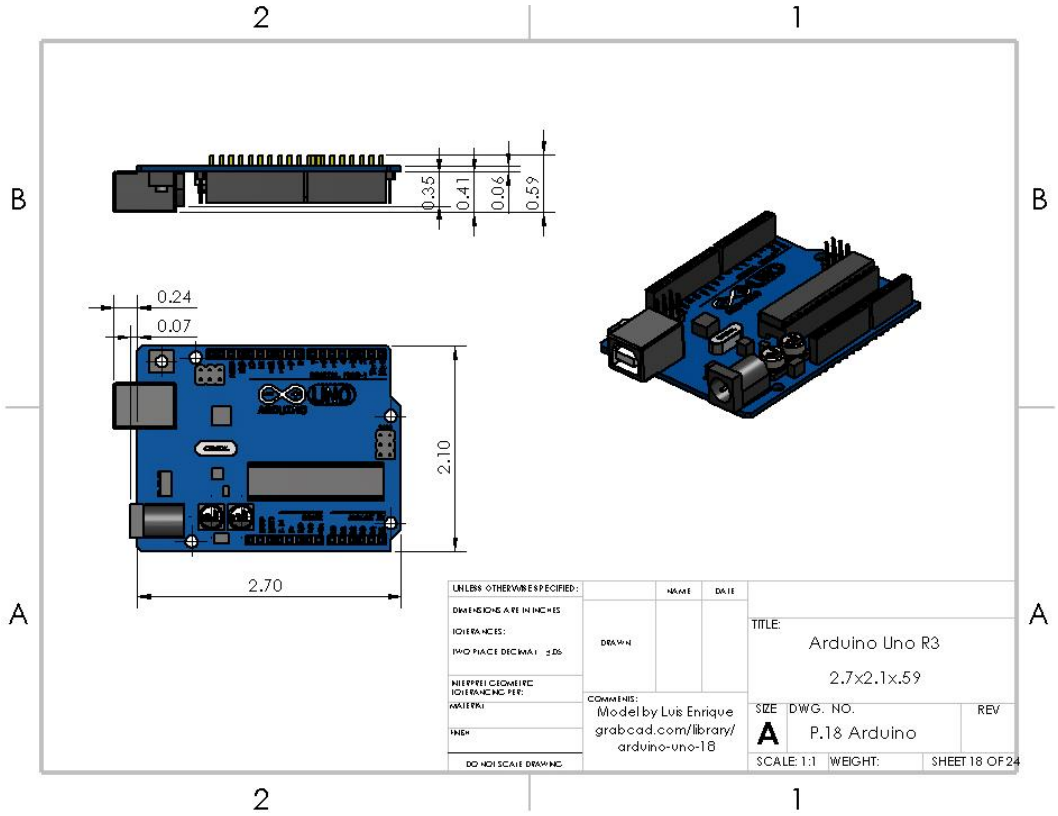


Figure 20: Arduino Uno R3

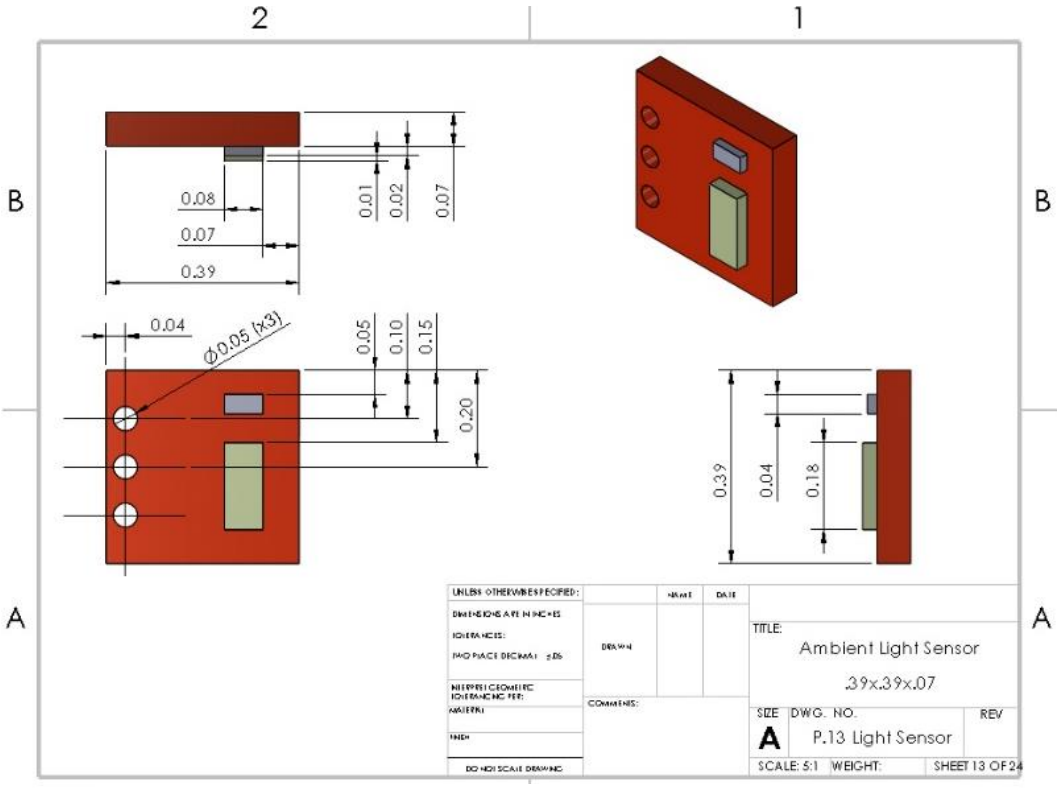


Figure 19: Sparkfun Ambient Light Sensor Breakout (x5)

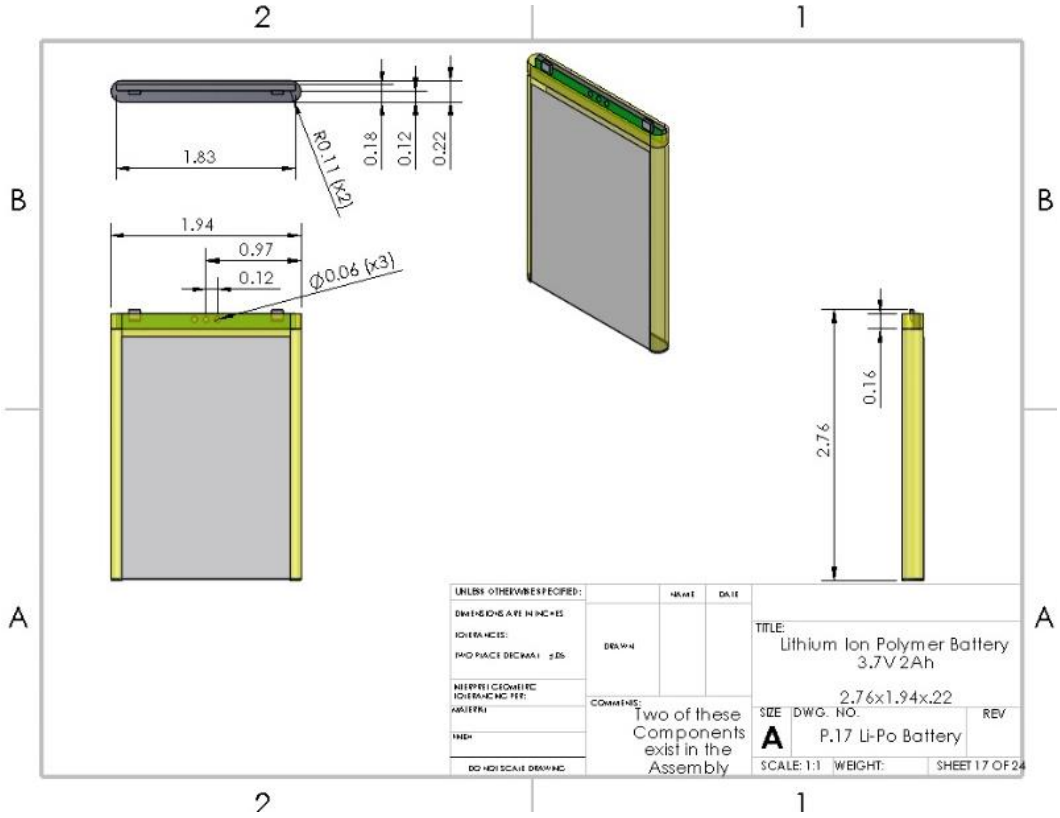


Figure 21: Lithium-Ion Polymer Battery 3.7V at 2Ah

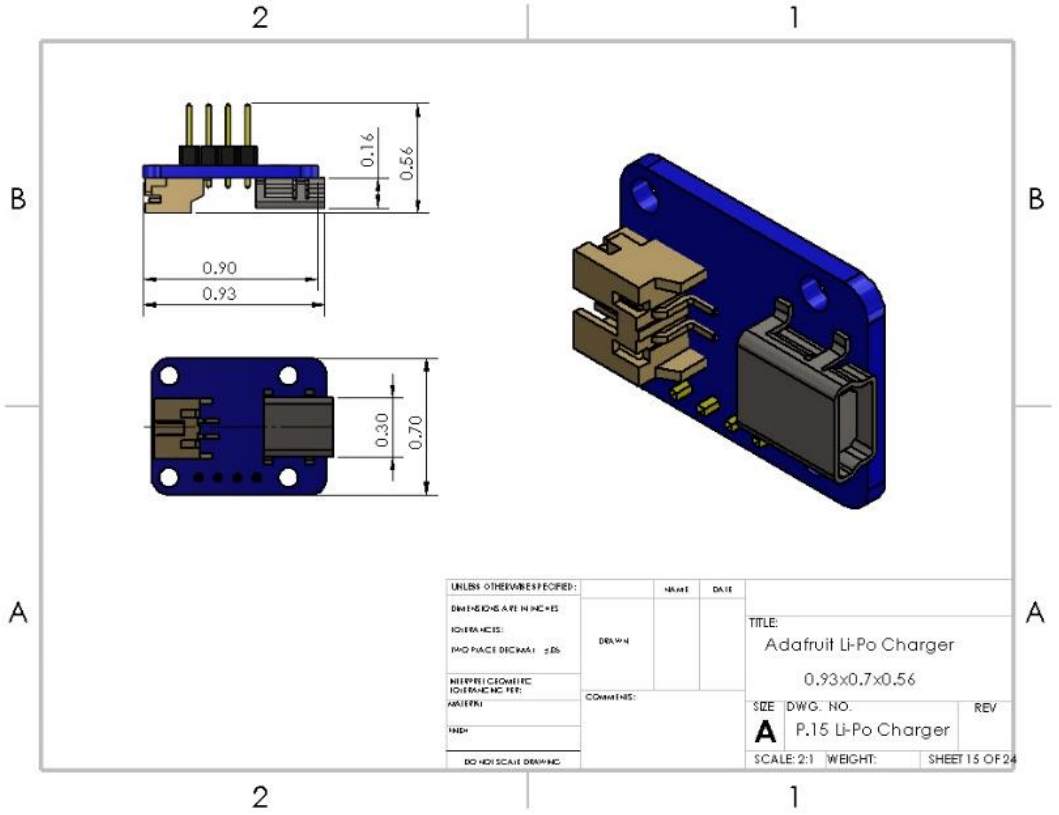


Figure 22: Adafruit Li-Po Charger

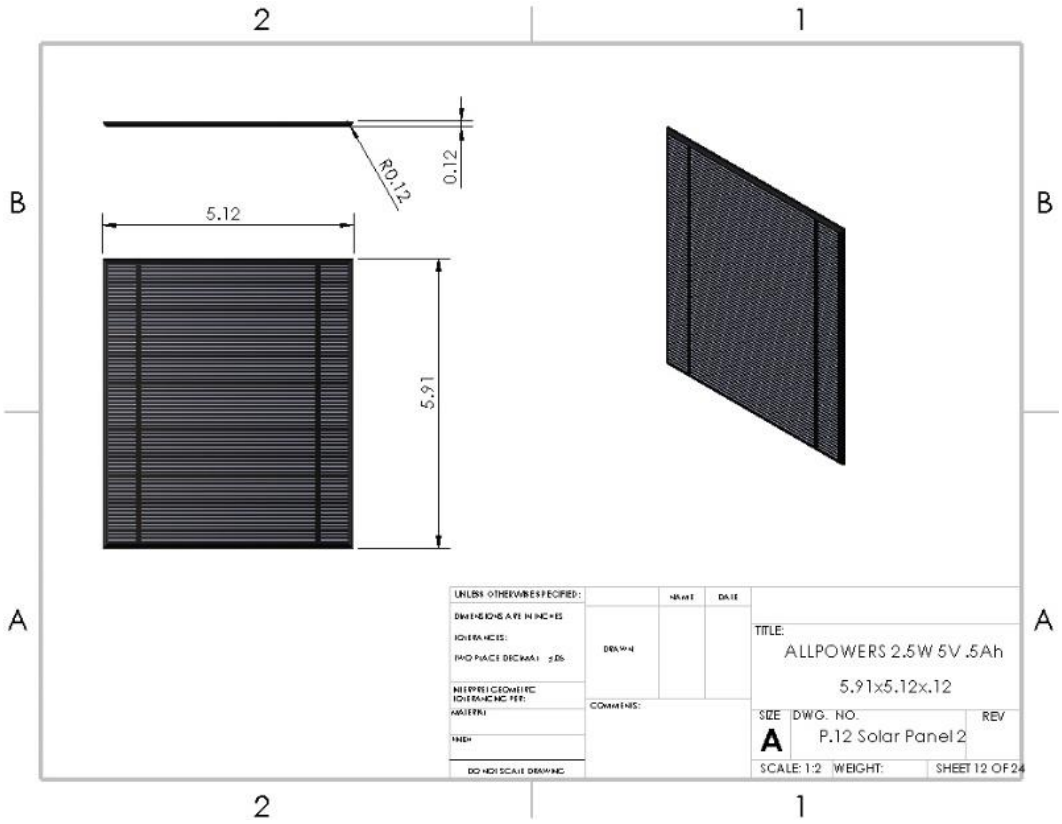


Figure 23: AllPOWERS 2.5W – 5V at .5Ah solar Panels (x2)

8 APPENDIX C: Assemblies & Exploded Views

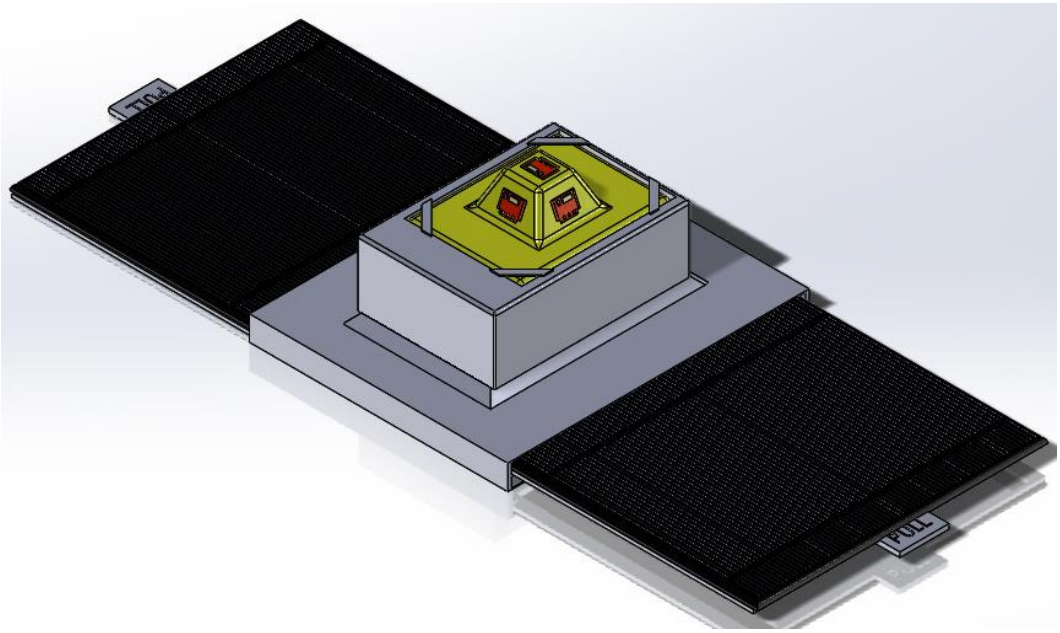


Figure 24: Complete assembly with Solar Panels Deployed from Solar Tray.

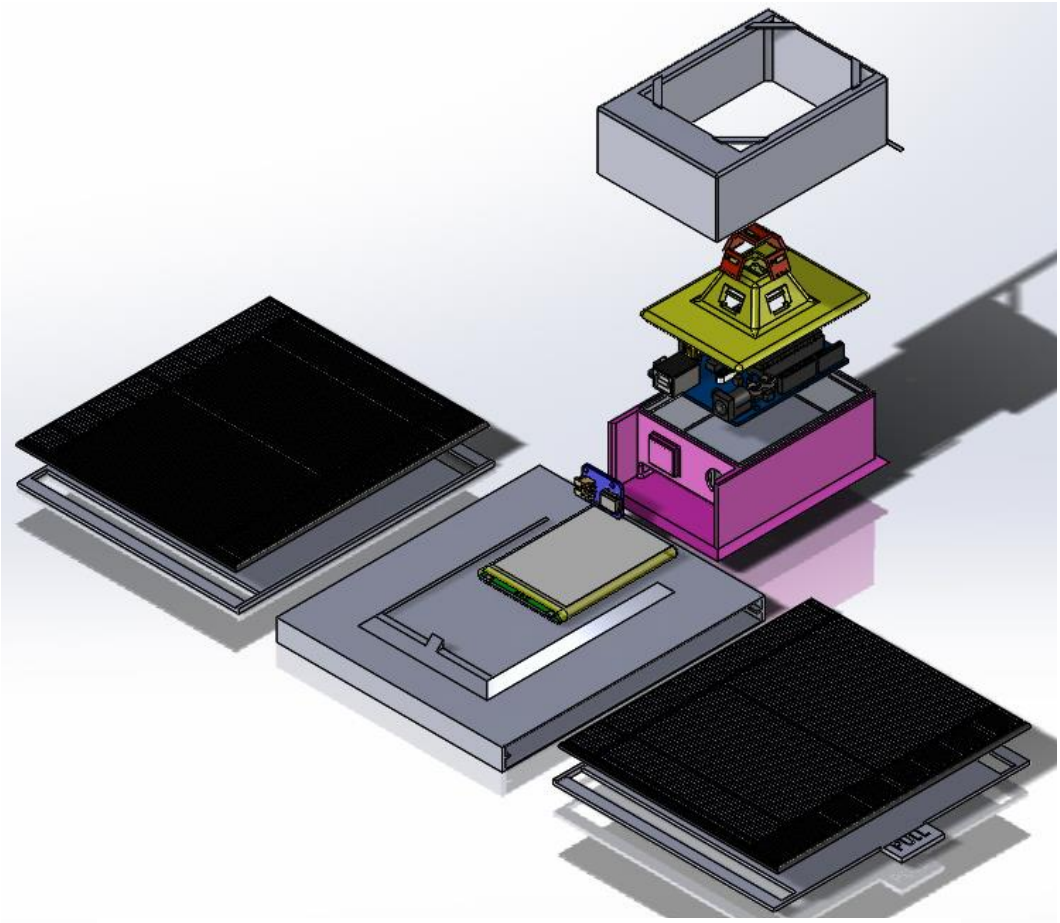


Figure 25: Exploded view of complete assembly.