Circuit Board Development

Yue Li

Jonathan Kosir

Instructor: James Leonard

ECE 449

Abstract:

The objective of this Senior Design Project was to create functional prototype circuit boards (PCBs) which helped aid in higher functionality as well “clean up” a ground robot. In doing so we could show the advantages of using a PCB in complicated projects compared to wires and breadboards which can create “rat’s nests”. The PCBs included all circuitry so that users do not need to wire these components, which reduces errors and can be more easy to debug and use. The first circuit for the ground robot is composed of two PCBs, one is for main Arduino, L293D and motors, another is for the mini Arduino pro and ultrasonic sensors. When we combine the two boards into one circuit the Arduino UNO can control all the components. This allows the ground robot to use its ultrasonic sensors in all four directions to change routine by itself when it senses barriers. The second circuit for the robot was a tiny voltage amplifier. This circuit takes three, two double A, battery packs and attaches each one to a voltage amplifier. These amplifiers are then put in parallel and attached to the motors of the robot. Each mini circuit takes the given voltage of the battery pack and amplifies it to five volts. This amplification allows for the motors to receive a higher current and in return making the motors in the robot run longer and more powerful while draining the batteries down as low as possible.

Table of Contents

Introduction:

Today, many students who major in electrical engineering wire their projects by hand using breadboards. In simple projects, this is a quick and convenient way to create the circuit. However, for more complicated and embedded system projects, the process of wiring the entire circuit by hand costs a large amount of time. This is notably true for when a mistake is made. Tracing the wires and debugging the circuit can take extensive time, especially if the wiring of the circuit is messy. The focus of our senior design is how to overcome this issue. The solution we decided on is prototype circuit boards or PCBs. In the PCB, all the circuits have already been designed and printed out to the board. This means users do not need to worry about connection problems. Therefore, we took an old ground robot which had been created in a previous capstone and improved it. The improvements we made focused around removing the wiring on the robot, which previously caused numerous problems, and replacing them with our created PCBs. Designing these PCBs for the robot allowed us to show the advantages stated above compared to a hand wired circuit. While we attacked, this plan our advisor found out he was having an issue with the motors not using the batteries as efficiently as he wished. The motors would only use about one sixth of the battery and then stop working. He asked us to create a lightweight PCB which would amplify the batteries and in return run more current to the motors. This amplifier would allow for the motors to drain the batteries much further than they were previously. This was just one more small addition in the process of improving the robot.

Background:

PCB Creation

Currently, various components are used in designing projects. For example, our ground robot uses a H bridge, a laser, and ultrasonic range sensors. However, when creating projects even when many components are used, most people use breadboards to wire them. This can be very bulky and create quite a mess. This is especially seen in more complex projects. Breadboards and hand wired circuits can create many problems especially in projects where smaller volume and weight is required. The complicated wiring that comes hand-in-hand with these projects can make it difficult to detect error. This can result in taking more time and resources to fix problems and complete the project. The ground robot which we

worked on had a L293D, LEDs, resistors, and four ultrasonic sensors. When the breadboard was used to wire this the space it took up was vast, not only was it bulky but heavy too. This weight caused the robot to move slower and the motors to work harder, making it much more in-efficient. That's why we think that using PCBs has many advantages, especially for complicated projects such as this.

Voltage Amplifier

As stated above many circuits do not use batteries until they are completely empty. This is a huge problem and can be a huge expense, and damaging for the environment. When people use these batteries and they stop working in the electronics they are using they assume they are dead and throw them away. These batteries easily could be far from dead but the circuit needs more current from the batteries than they can supply as they begin to run down. One way to fix this issue is to build in a voltage amplifier into the circuit. The amplifier can increase the voltage and current coming from the batteries providing enough to run the circuit, even though the battery is providing less power than the circuit needs. This allows the circuit to drain the batteries almost all the way down. Using a voltage amplifier and creating a PCB to mount our components on allows us to mount a voltage amplifier to our battery packs. This will drain our batteries all the way down and still maintains a lightweight and compact electronic package. Therefore, we did not just create something which barely affects the weight or size of our robot, and save us money on buying new batteries, but we also created an item that can help improve the environment and reduce our battery costs.

Research:

Voltage Amplifier

For the voltage amplifier, we had to find a solution which allowed us to use as much of the battery as possible, but still stay lightweight. We already knew to stay lightweight the circuit must be designed by creating our PCB board as well as keeping our components to a minimum. Therefore, we focused on how we would amplify the batteries to create enough current for the motors and effectively drain the batteries. This process took us through many different ideas, each idea, had to be researched, developed, and then analyzed to see if it would be the correct solution for our problem. The first thing we focused our research on was a "joule thief". A joule thief effectively uses a toroidal coil and a transistor to amplify a voltage source. First electricity flows through the first coil to the base. This slowly opens the collector-emitter channel and allows electricity to flow into the second coil. This electricity growth

creates a magnetic field. This field increases the electricity in the first coil. This increase in electricity goes to the base and causes the collector-emitter channel to open a little more. This, causes the magnetic field to grow even more as well as the electricity in the first coil. This process repeats until the transistor enters saturation and cannot increase any more. This stops the increase of electricity in the second coil and causes the electrical current in the first coil to slowly begin to dissipate. In return, this causes the electricity to the base of the transistor to lower slowly closing the transistors collector-emitter channel to close until it closes completely. This causes a large amount of energy to be stored in the coil. All this energy must go somewhere and is released causing a huge voltage spike. This spike exits through the part of the circuit which is connected to the second coil, typically through a diode. This process repeats around 50,000 times a second causing an average gain in voltage.

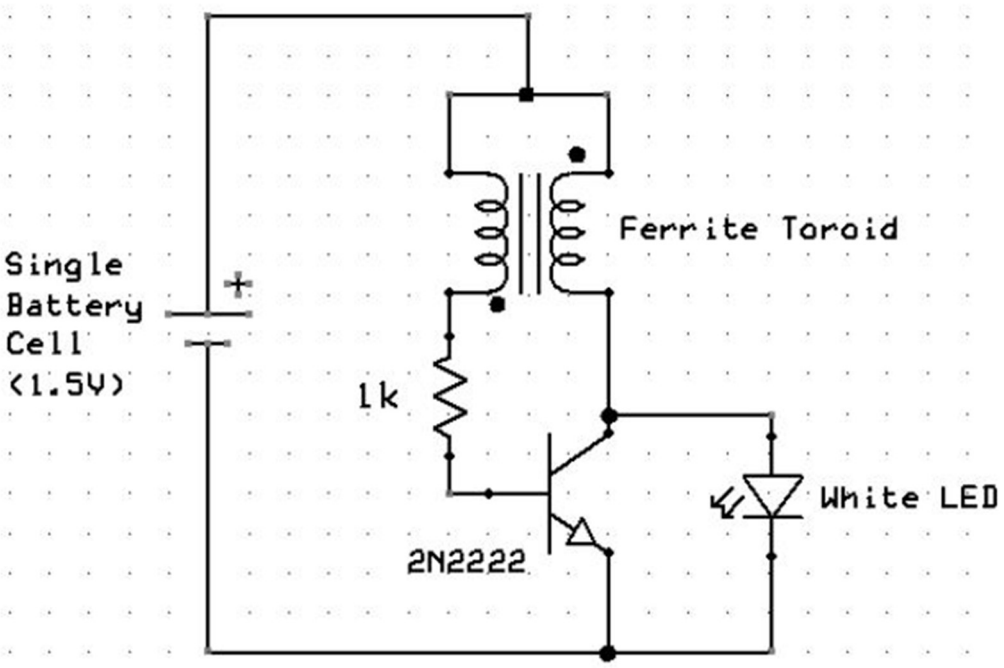


Figure 1: Joule Thief Schematic

In the process of looking at joule thief and its partner super joule thief we came upon what is being called the scavenger circuit.

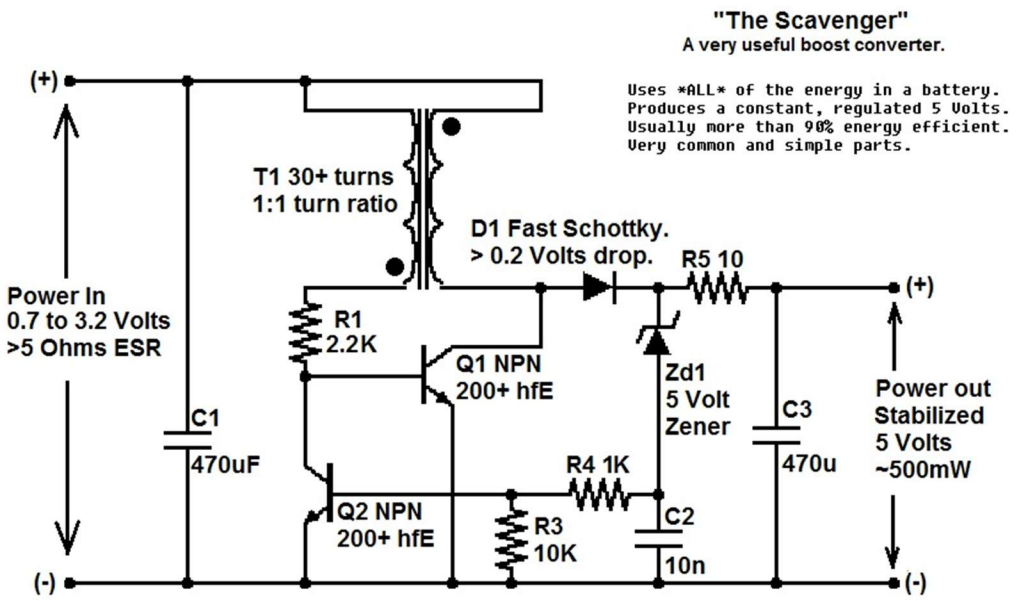


Figure 2: The Scavenger Circuit Schematic

This circuit uses a Joule Thief, extra transistor, and a voltage regulator. The joule thief in this circuit works the same way as stated previously. Then the extra transistor makes the opening and closing of the transistor used in the joule thief happen more abruptly. This strong switch allows the electricity to hit the toroid much harder effectively "slamming" it. This slamming causes the magnetic field in the grow much stronger which allows for the voltage gain to be much higher.

In this circuit once this electricity from the toroid is released it goes into a voltage regulator. Since the joule thief creates super high spikes of electricity at very quick intervals it can be dangerous to more sensitive electronics. The regulator levels out the voltage so there is just a steady voltage increase being released at the voltage out point.

After finding this circuit and reading more about it we decided to build it, and analyze it to see if it would fit our needs. In the section below you can review our data and what information we gained from our analyses. In the end, we decided the circuit was to unstable for our needs. When there were lower loads on the circuit the circuit would not give us the voltage out that we would have liked and in return not supplying the current we would have needed for the circuit. We figured that this stability problem may have been able to be fixed by attaching the circuit to an Arduino and allowing the Arduino to control the transistor and its pulse width modulation to receive more steady results. We decided to not pursue this path do to our second fall back of this circuit. We decided it was simply too bulky. The toroid was much too big and there were to many components to be as lightweight as we would have liked.

Therefor in the end we decided to pursue the route of an online purchased voltage amplifier component. Our only issue with this idea is we could not find a component which would give, what we believed, to be our ideal voltage, seven volts. The circuit could take a battery almost all the way to empty with load not affecting it to severely. Still, as noted, the amplification did not reach our needs. So now we were stuck at a road block. Continue trying to design and improve our heavier, bulkier, circuit, which gave us the volts out we wanted but didn't handle loads the way we would have liked. Or go with the small compact circuit which did a great job amplifying small voltages and handled loads very effectively. The decision we made was go with the lightweight store bought version. With this purchase, we decided to create three circuits to aid this component and put three voltage amplifiers, each singly attached to a battery pack, and put these three battery packs and amplifiers in parallel. The results we receive from this last circuit combination was exactly what we needed.

Data Analysis:

The bottom six figures are graphs of data collected when analyzing the scavenger circuit. This data was captured by putting different load resistors on the circuit, for each new load resistor we put in many different input voltages. We collected frequency of how often Q1 switched. This was done by attaching an oscilloscope to the base of Q1. We also analyzed what our output voltage, over the load resistor using a multimeter, at each new input voltage. We then analyzed the data by recording our received data in individual files per the load resistor. We then analyzed this data by graphing our data using a MATLAB program we created. The first five figures show our voltage in vs our frequency and voltage out. As you can see the lower the voltage the harder the circuit had to work to try and raise the voltage to the seven volts we needed. The frequency of the transistor increased and once the transistor couldn't switch quick enough the voltage out drops substantially.

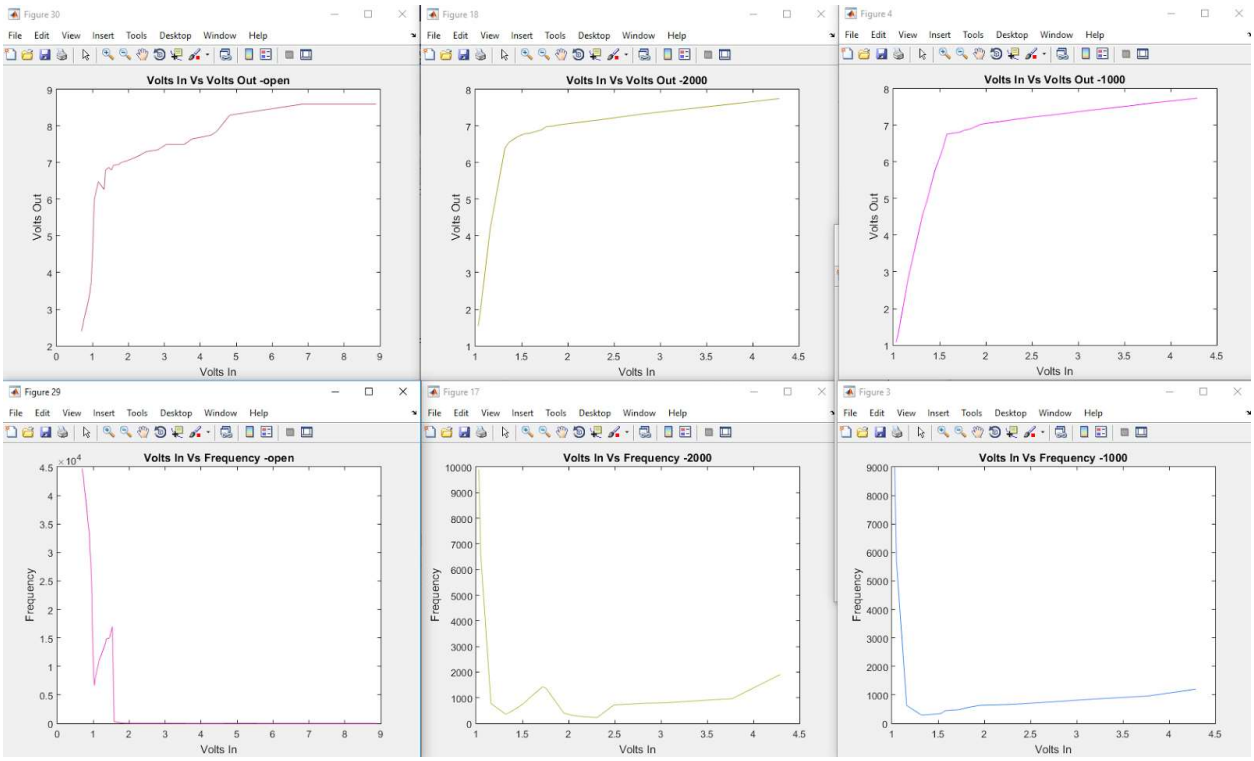


Figure 3: Graphs Scavenger Circuit Analysis with Open Load to 1K Ohms

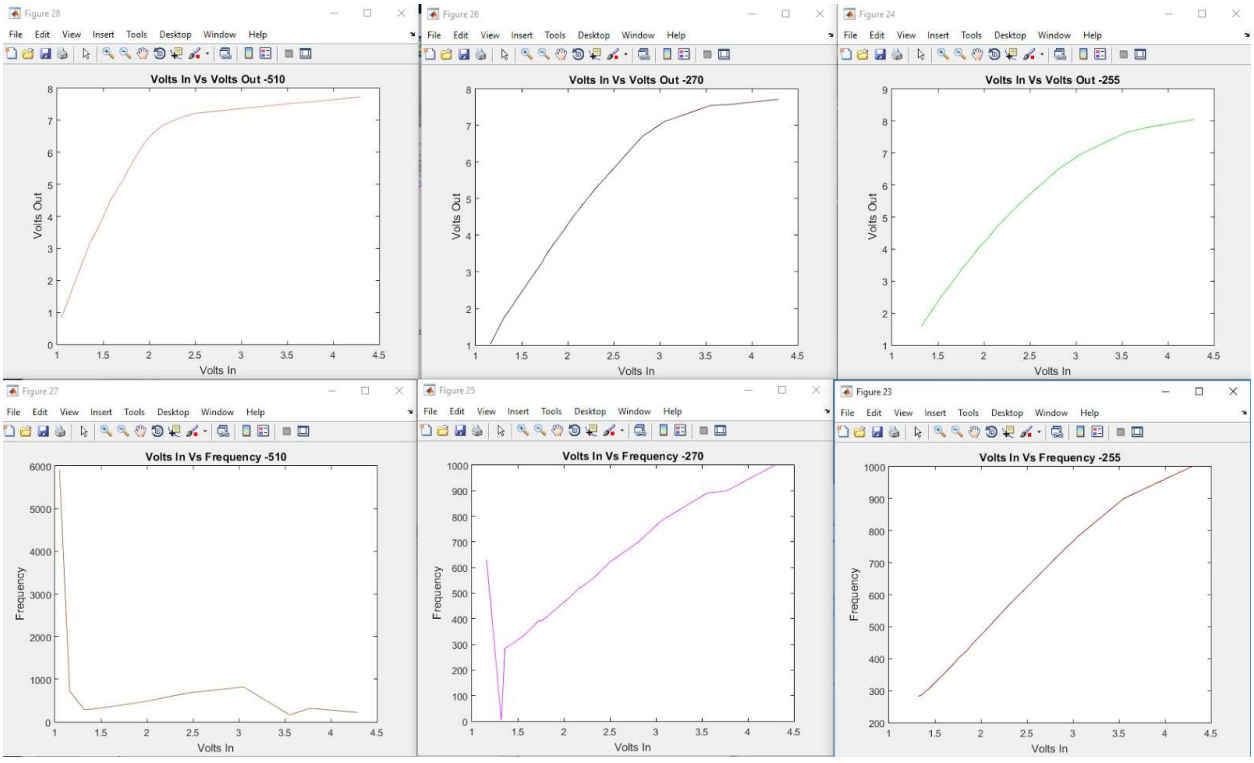


Figure 4: Graphs Scavenger Circuit Analysis with 510 to 1K Ohms load

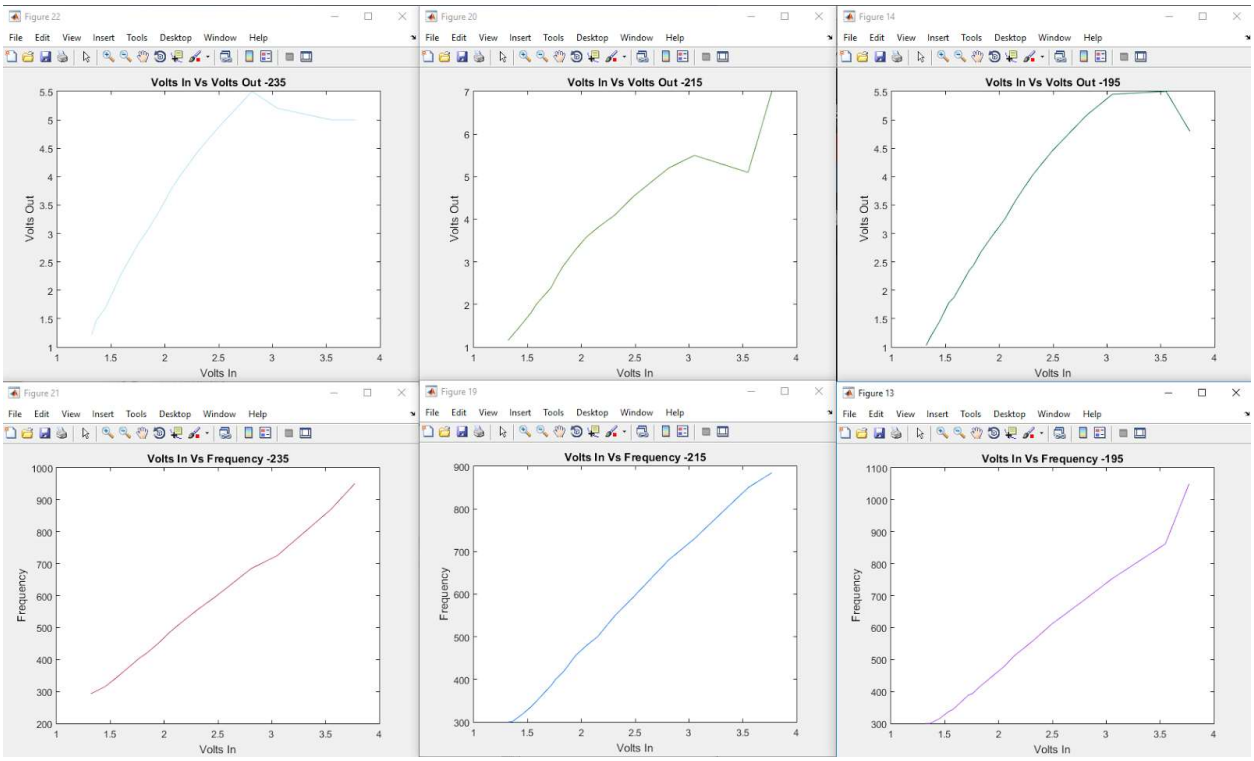


Figure 5: Graphs Scavenger Circuit Analysis with 235 to 195 Ohms Load

As you can see when the load resistor is around 165 Ohms the circuit begins to break down. We do not see the voltage increase that we saw in the previous graphs. This means that we are not getting nearly enough current over our load resistor to satisfy our needs. This resistance in much higher than we anticipated and would not meet the criteria we would have liked.

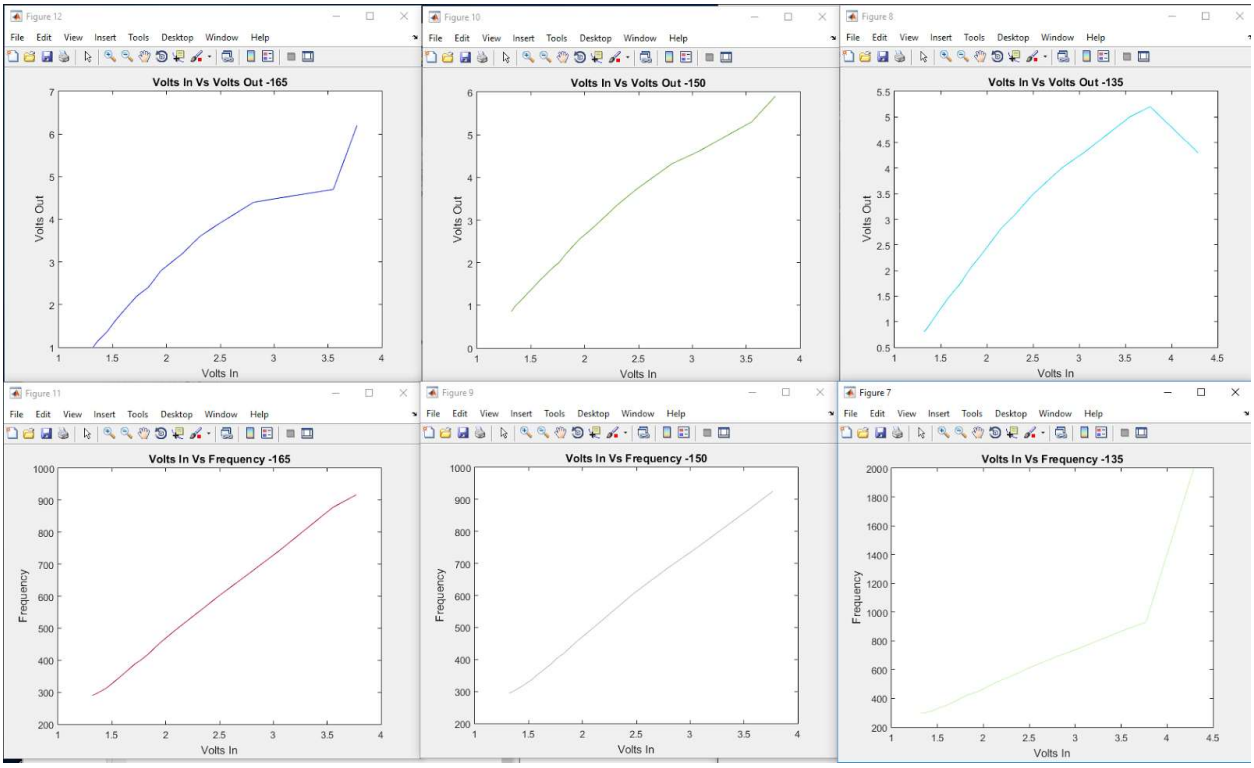


Figure 6: Graphs Scavenger Circuit Analysis with 165 to 135 Ohms Load

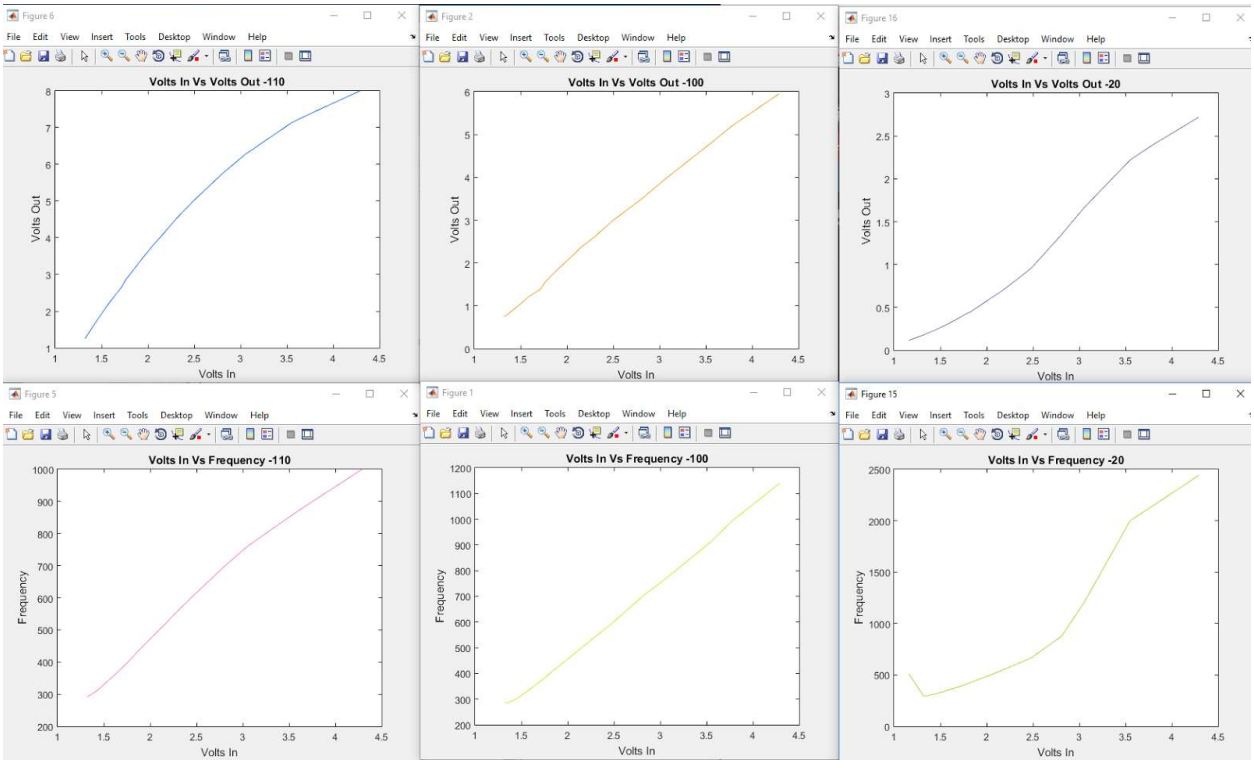


Figure 7: Graphs Scavenger Circuit Analysis with 110 to 20 Ohms load

The figure below shows our Thevenin Voltage, Resistance, and Norton Equivalents. To get these results we analyzed all our data over all our load resistors. Our Thevenin voltage vs our voltage in looks very good in this graph. We get more than enough voltage out compared to our voltage in. The problem with this circuit is noticed when we look at the Norton current. We would like to maintain well over .1 amps. Though this value is dropped below this threshold around 2.75 volts in. 2.75 volts is the value of having our two batteries in parallel drained about ¼ of the way down. This clearly is not acceptable.

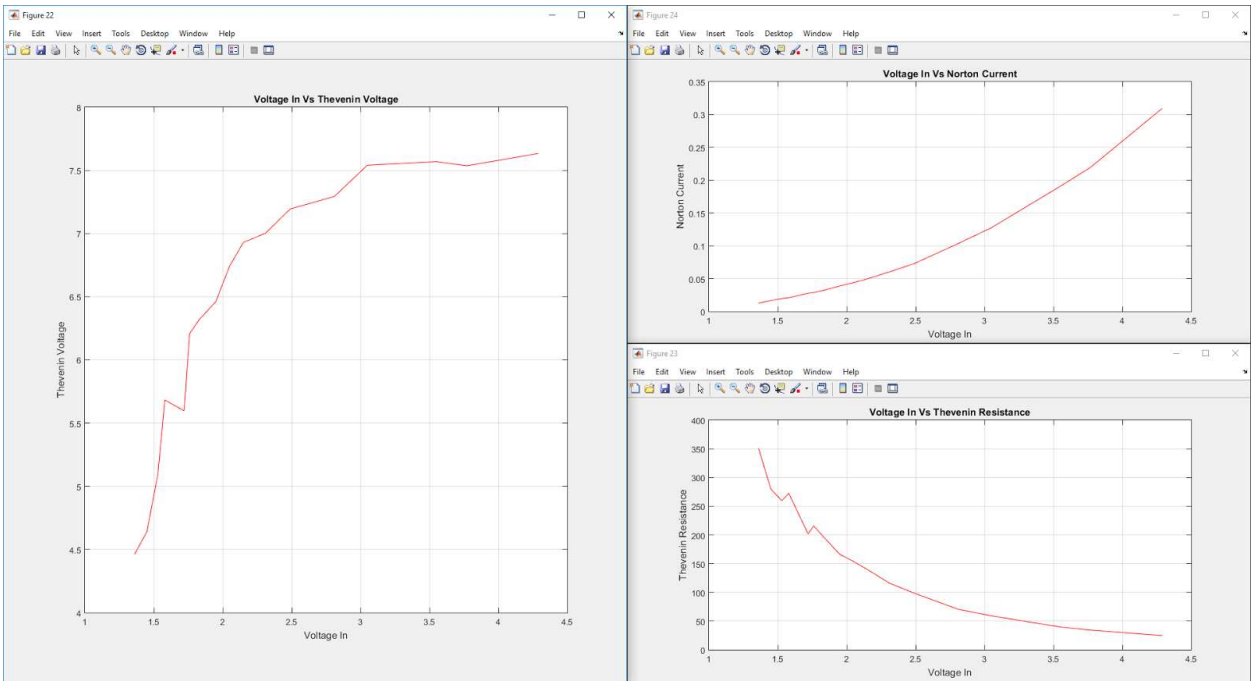


Figure 8: Graphs Scavenger Circuit Analysis Norton and Thevenin Equivalents

The data below is just a small portion of data collected over around 20 hours for the store-bought voltage amplifier. The solid blue lines are the voltage in, the dotted blue line is the voltage out, and the red line is the current. The way our circuit worked the voltage amplifier took voltage from our batteries and the voltage out from the amplifier is connected to a relay. One part of the switch on the relay is connected to load resistors with the resistance of 33.9 Ohms, the other part of the switch is just floating with no load. We then have an Arduino connected to the voltage in, the voltage out before the switch, so we can get the value out when it is loaded and not loaded, then lastly to the relay. The Arduino reads the values through its serial input, as well as controls the switch which switched about every minute. The serial pushes these values every 5 seconds to MATLAB and then MATLAB graphs these values in real time, as well as prints them out to a file just in case the program fails and loses we lose the graph.

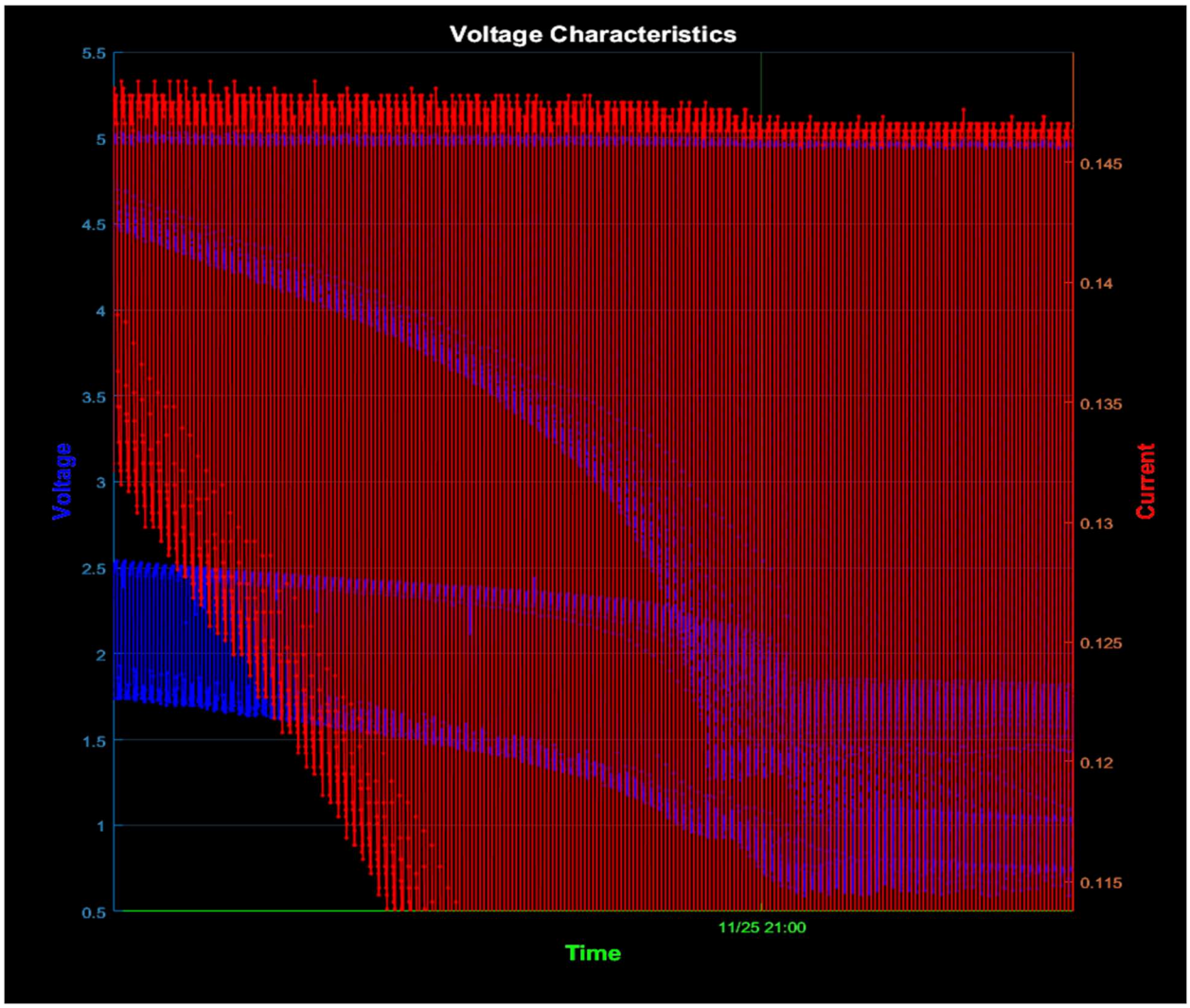


Figure 9: Real Time MATLAB analysis of store bought voltage amplifier zoomed in

Looking at these two graphs we can see the output of the store-bought component is much better than the values we were getting from the Scavenger circuit we made. Our voltage clearly drops when we put on an extremely small load resistance but maintains 5 volts without any load even when the input voltage gets down to around 1.5. At the lowest point our voltage amplifier in given around .6 volts in but still more than doubles it to around 1.5 volts out. In the top figure, we cannot see how low our current goes, but with no load our current is at around .145 amps which is much more than the other circuit gave us which was at most around 14 milliamps. Also, the circuit is much less affected to load. We look at the figure below which shows a more zoomed in shot and we see that the value of the current drops to around 34 milliamps with a super low load on it which is much better than our previous circuit.

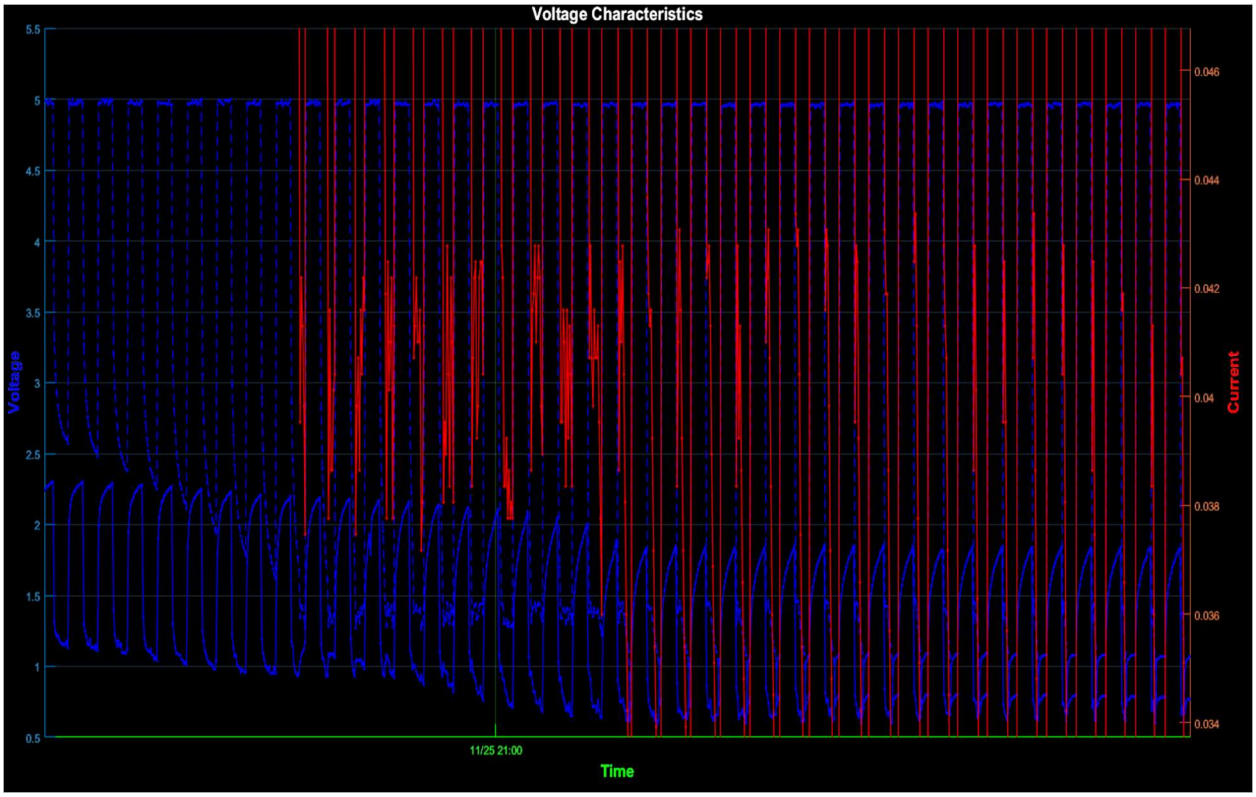


Figure 11: Real Time MATLAB analysis of store bought voltage amplifier zoomed in

After creating the PCB, we ran the analysis again to double check everything turned out successfully. As you can see in the figure below the results we obtained from the PCB are almost identical to the results we received from just the test of the component.

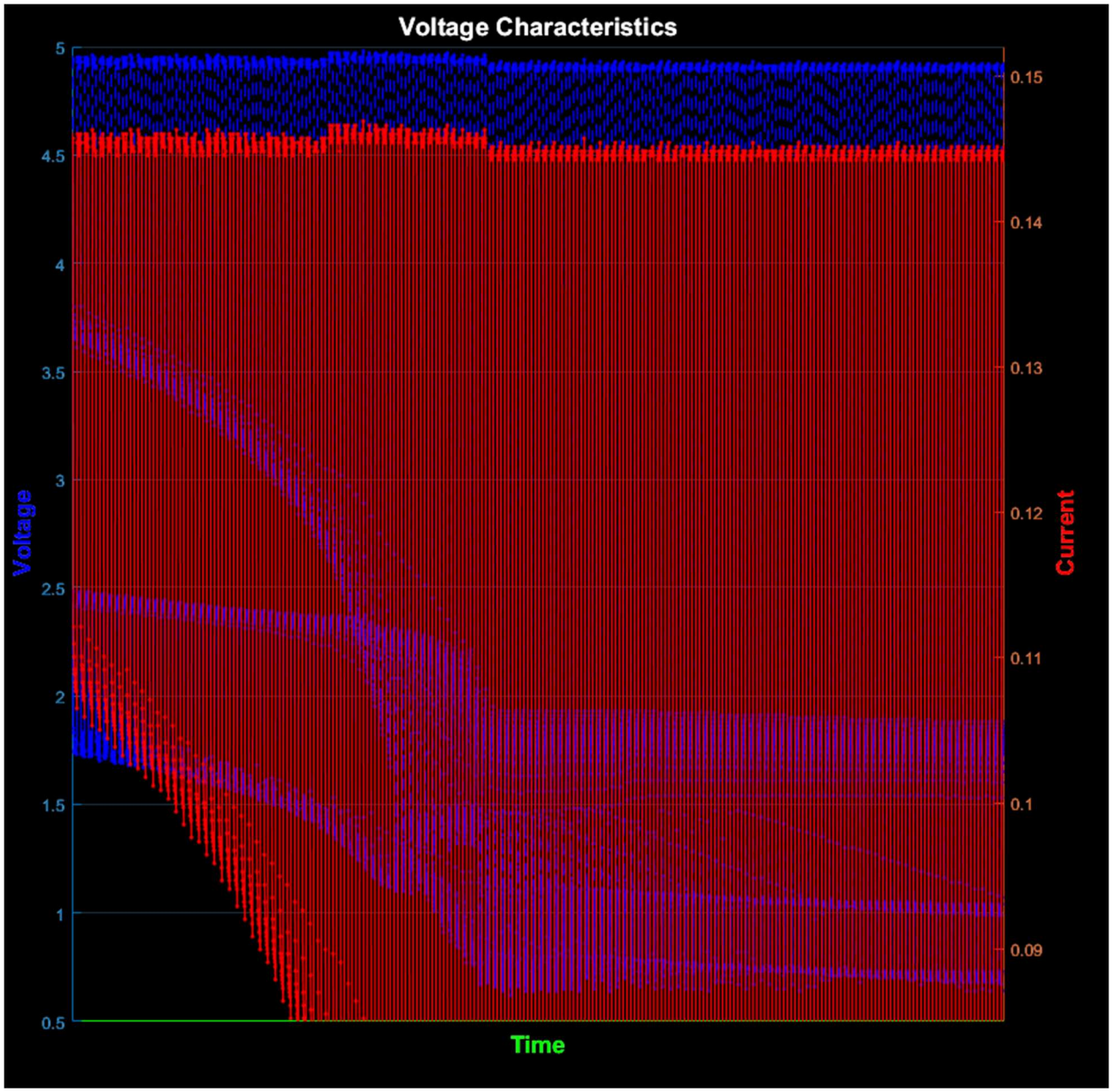


Figure 12: Real Time MATLAB analysis of PCB with voltage amplifier

Lastly, we analyzed this PCB with two C batteries in series. With this we obtained much better results. As you can see in the two graphs below our voltage was almost identical but the C batteries gave us much better current. This current is about the same as the double As at the beginning but lasts much longer. Putting two packs of C batteries in series would most definitely give us the results we are looking for.

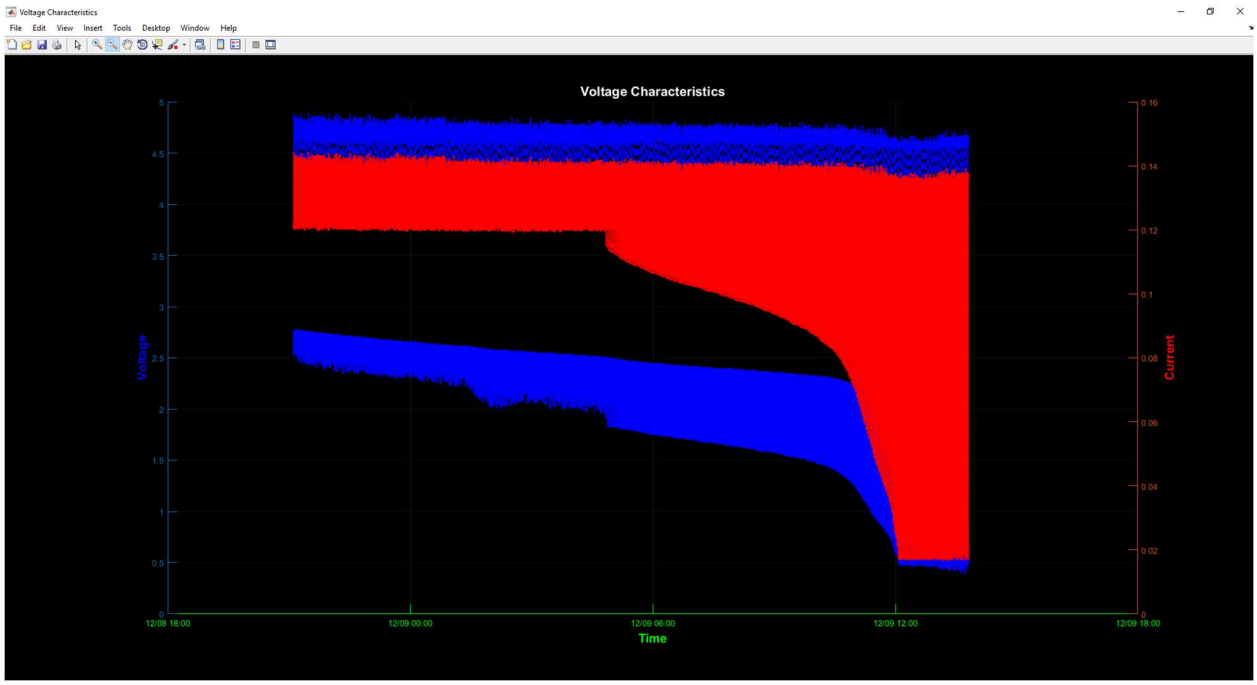


Figure 13: MATLAB Analysis of C Batteries Attached to PCB

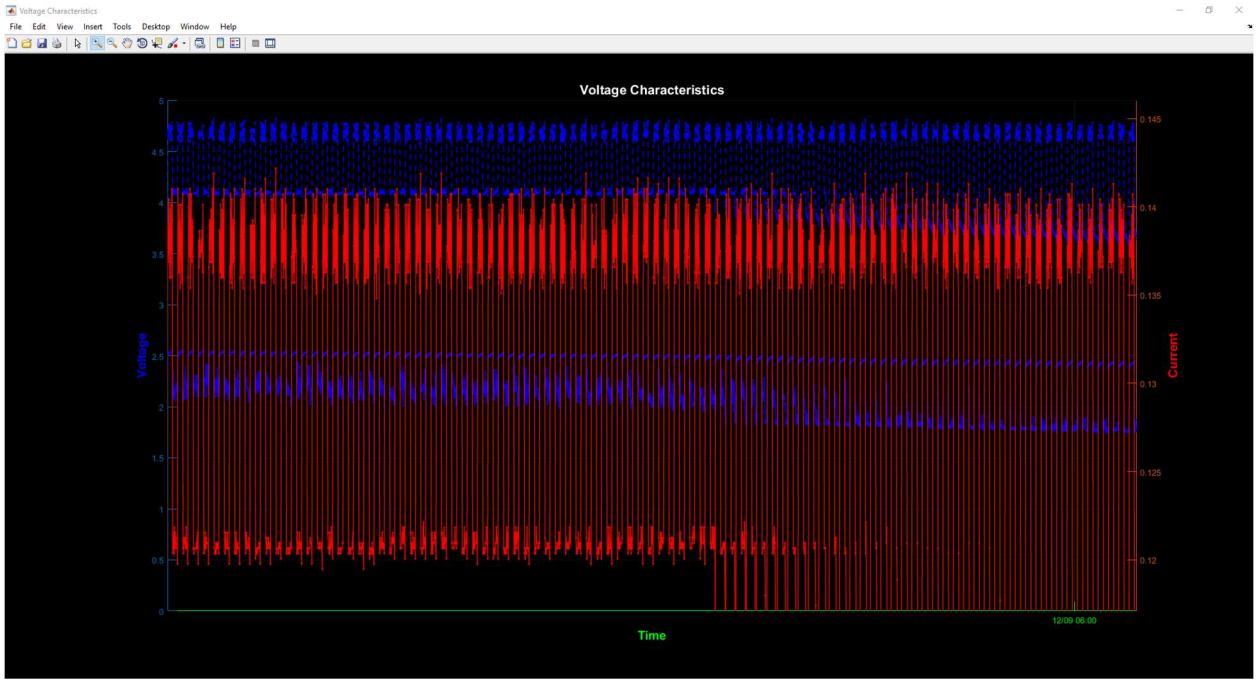


Figure 14: MATLAB Analysis of C Batteries Attached to PCB

PCB Design Process

A brief overview of the PCB's design process is below. First, we need to design our board in a PCB creation software, in our case, we used Eagle. This PCB design must then be sent to our ISO Pro software. The next step is to print out the board using the QCJ5 printing machine. Lastly you must then solder all components onto the circuit board. The details of the process are the following:

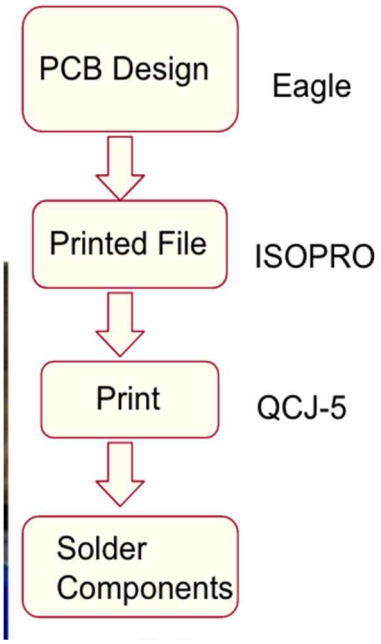


Figure 15: Design Process Flow Chart of PCB

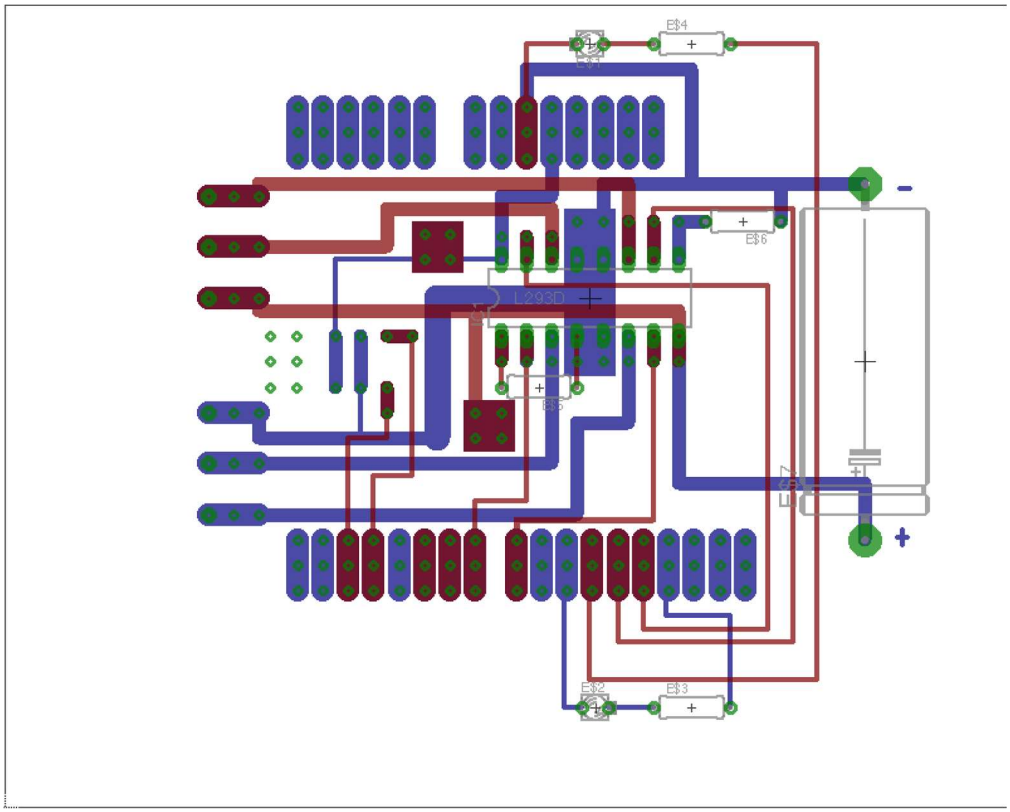


Figure 16: Eagle Board Diagram

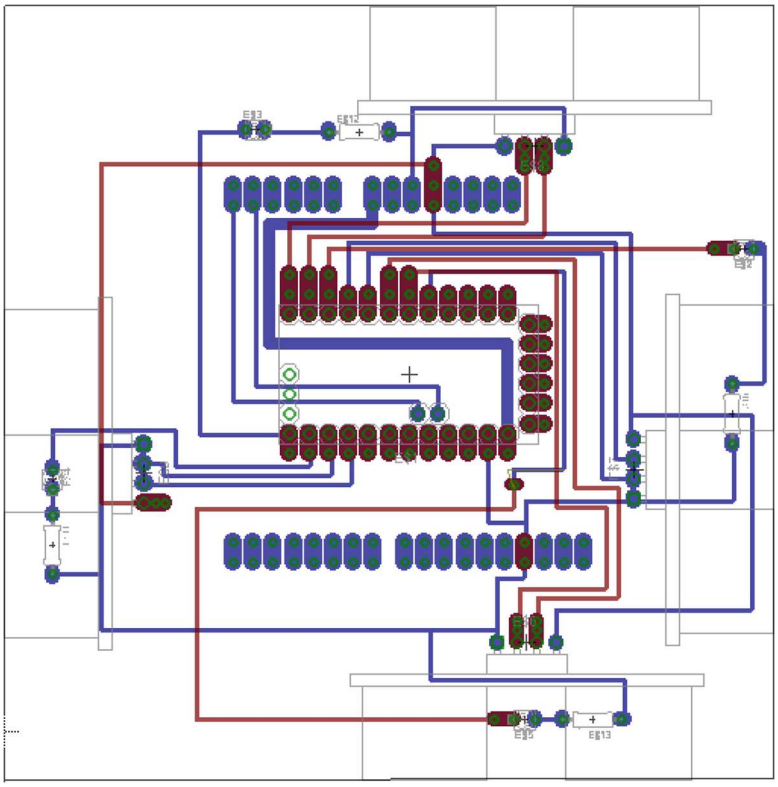


Figure 17: Eagle Board Diagram

First you must design your desired circuit in a PCB creation software. This must be designed in a way that minimizes your design as well as meets your needs. In doing this you must think about each side of the board. The figure above shows our PCB design, the blue circuit is for the solder layer and the red circuit is for the component layer. Our robotic project is composed of two circuit boards: The first board is for motor control, and the second board is for a second Arduino, four ultrasonic sensors, and four LEDs. When we design our Eagle files we need to pay attention to not let two wires cross, that we have a solder layer wire at each attachment point of a component, and that we try to minimize feed through vias.

The Eagle files are converted to a print file, called Gerber files, using the Eagle CAM process. This is then imported into ISO Pro to produce three main separate files: the component layer file, solder layer file, and drill file. The component file is the top layer of board, the solder file is bottom layer, and the drill file is the location of the holes. The motor and ultrasonic sensor's printed file is shown below.

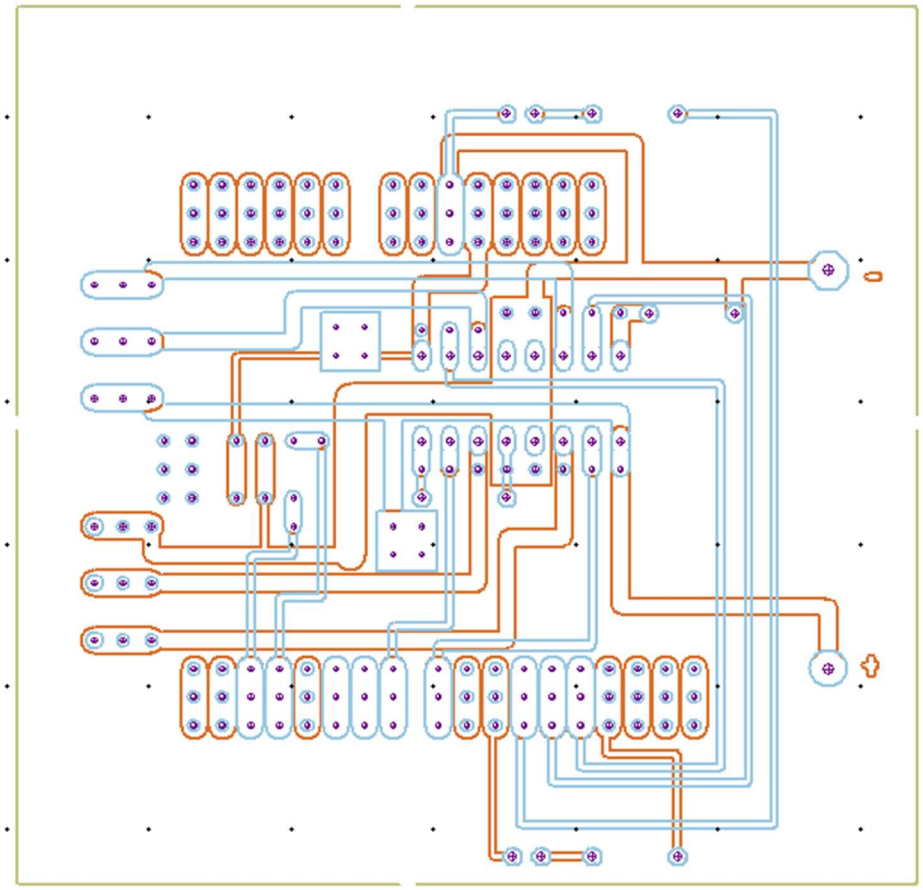


Figure 18: ISO Pro PCB Diagram

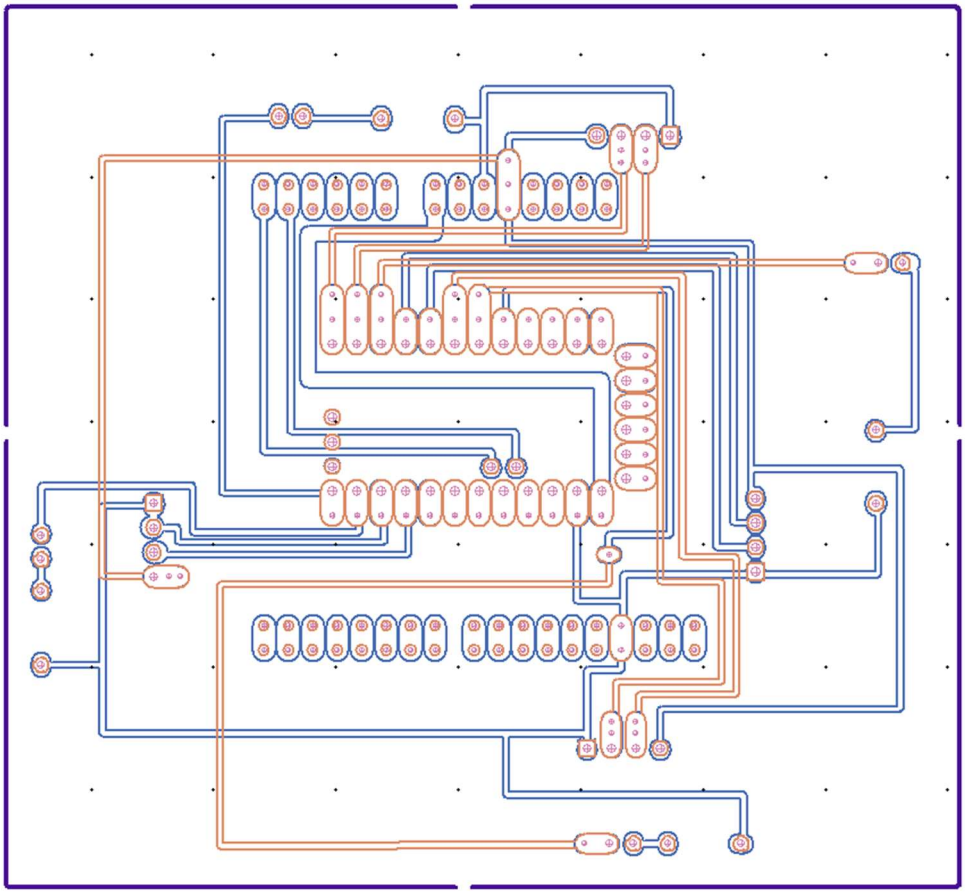


Figure 19: ISO Pro PCB Diagram

As the above two figures show, the points are the holes needed to be drilled in the board and the lines are the traces that machine will cut along so that a circuit will formed. Because the surface of that board is copper, the machine cuts isolation lines to along the traces to form the wires on the board.

We use the QC-J5 milling machine to print out this circuit board, for drilling and milling, there are many drill sizes available so one of the challenges is to pick the proper size for each component to give the best soldering performance. In addition, users can also set the drilling depth appropriate for different board materials. We can set the specific position of our PCB in the ISO Pro software to get the most use of our blank board. To take the circuit board out, users need to design a board outline to be cut along like the purple line in ultrasonic sensor board.

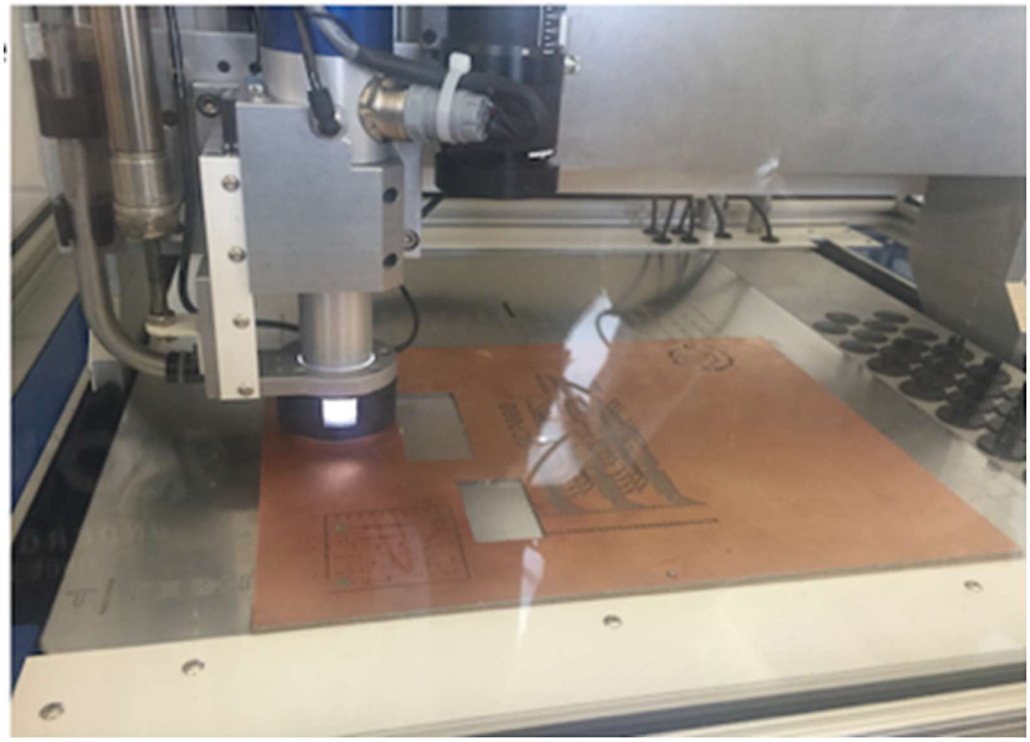


Figure 20: QC-J5 milling machine

Results:

After we soldered all components in the printed board, it is shown as figure below:

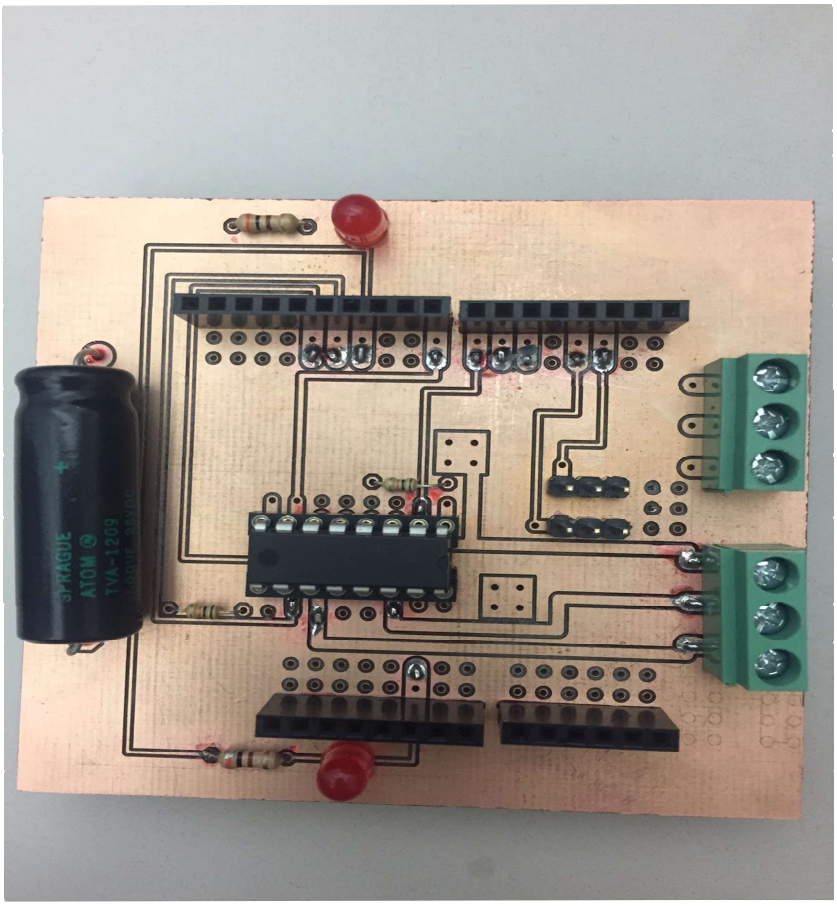


Figure 21: PCB One

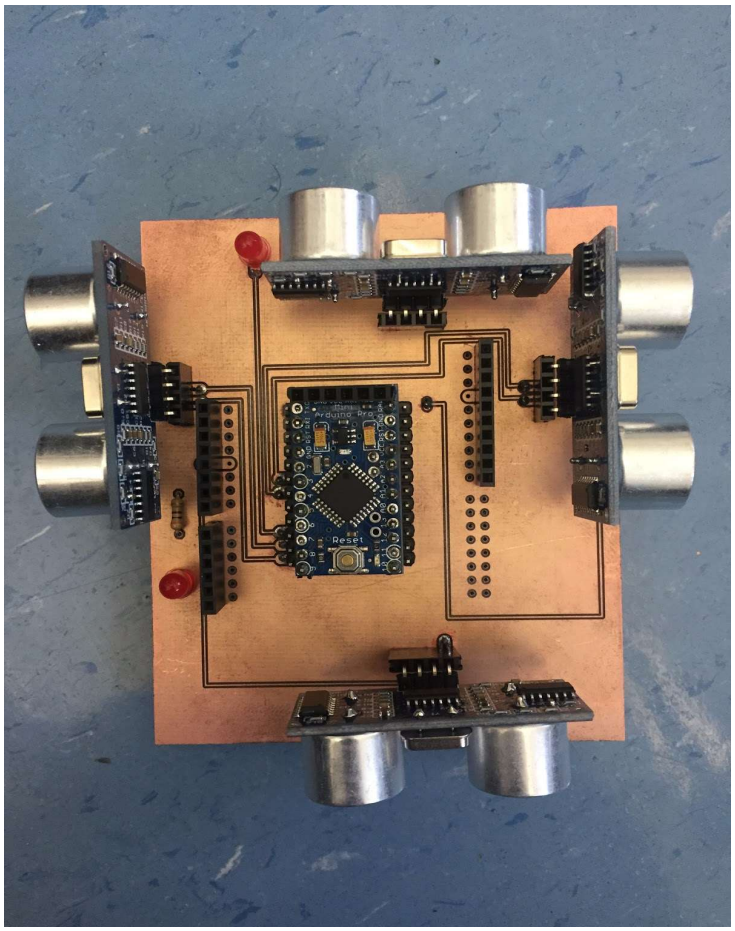


Figure 22: PCB One

This is the board of motor, the middle chip is L293D used to control motor, the two lines in the upper and bottom of board and the headers that plug into the Arduino UNO since the Arduino UNO is our main microprocessor. In the right of the image are two three-hole screw headers. The two center holes are for power supply and the top two holes are for the left DC motor and the bottom two holes are for the right DC motor.

The above figure is the ultrasonic sensor board. The four-ultrasonic sensors point in four directions is used to detect barriers from every direction. The middle chip is a mini Arduino pro, which is used to control ultrasonic sensor and LEDs. This Mini Arduino Pro is controlled by the Arduino UNO by I2C communication using two lines between the Arduino Uno and the mini Arduino Pro for serial communication.

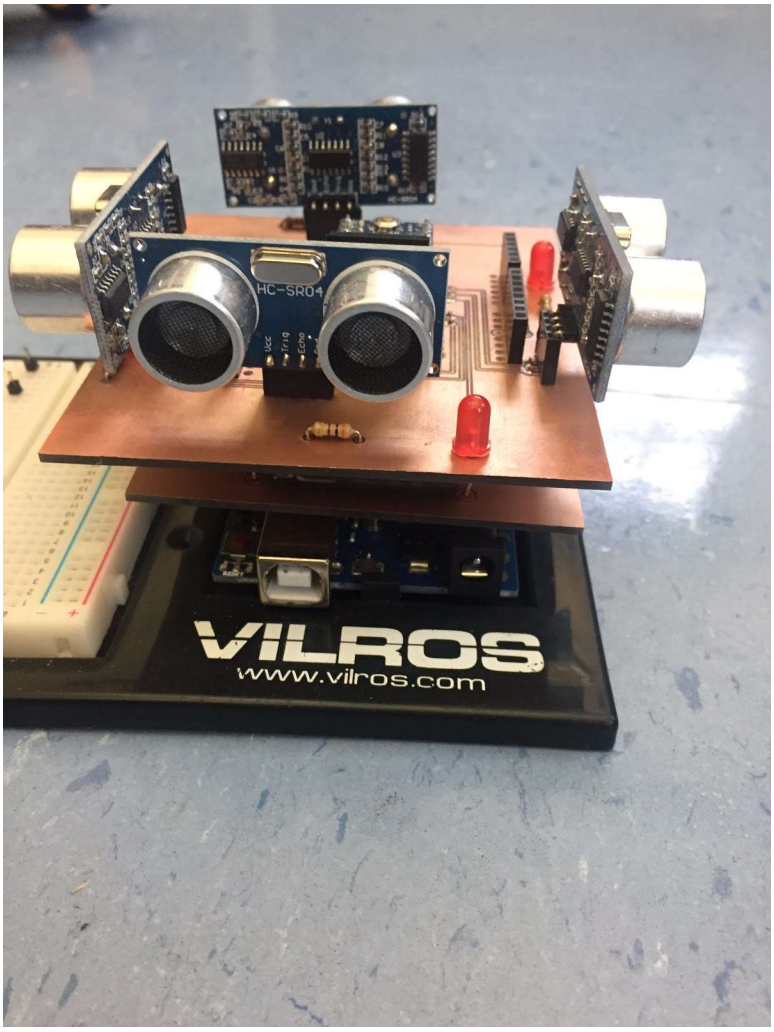


Figure 23: PCB One and Two on An Arduino

The above image is our results of the ground robot, the bottom layer is Arduino UNO so that it could control every component in this project. As you can see there are no extra wires in whole project which reduces much more space and weight.

Future Work

Though we finished all our main goals for this semester there is always more to be done. We did a great job in learning how to use the PCB machine and could do much more work over a shorter amount of time. There is still need to work more on the voltage amplifier. We did not completely get multiple battery packs in parallel or series to be analyzed. As well a circuit which created a higher current and voltage would be ideal. Another thing we would begin to look at creating PCBs for the gyrocopter which

our advisor is working on. Also, we would like to continue to teach people how to work with the machine now that we have the knowledge. The group below us will be working with it and it would be very beneficial if we would take the time to teach them.

Conclusion

In conclusion, our project improved functionality of the ground robot as well as make it lighter, and easier to debug in the future. The two images below are a before and after of the robot. As you can see the top picture is very cluttered and a mess. If a single wire were to accidentally be moved it would be very difficult to find or remember where it should be placed. Also, if any errors happened it would be very difficult to debug.

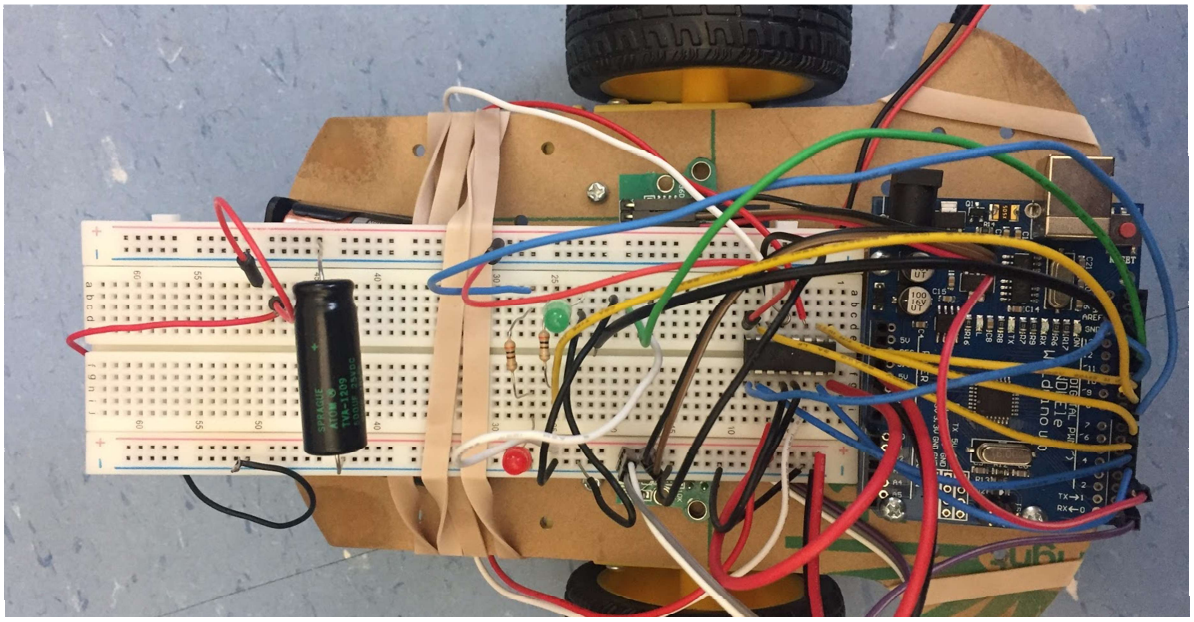


Figure 24: Ground Robot Before Image

Though the image below is not on the wheels and motors of the robot as seen in the image above, it is all the very important parts: wires, breadboard, Arduino, and all components. The below image adds an entire other shield which is not seen in the picture above with four ultrasonic sensors. Even with this entire other shield the bottom product looks cleaner and much easier to debug.

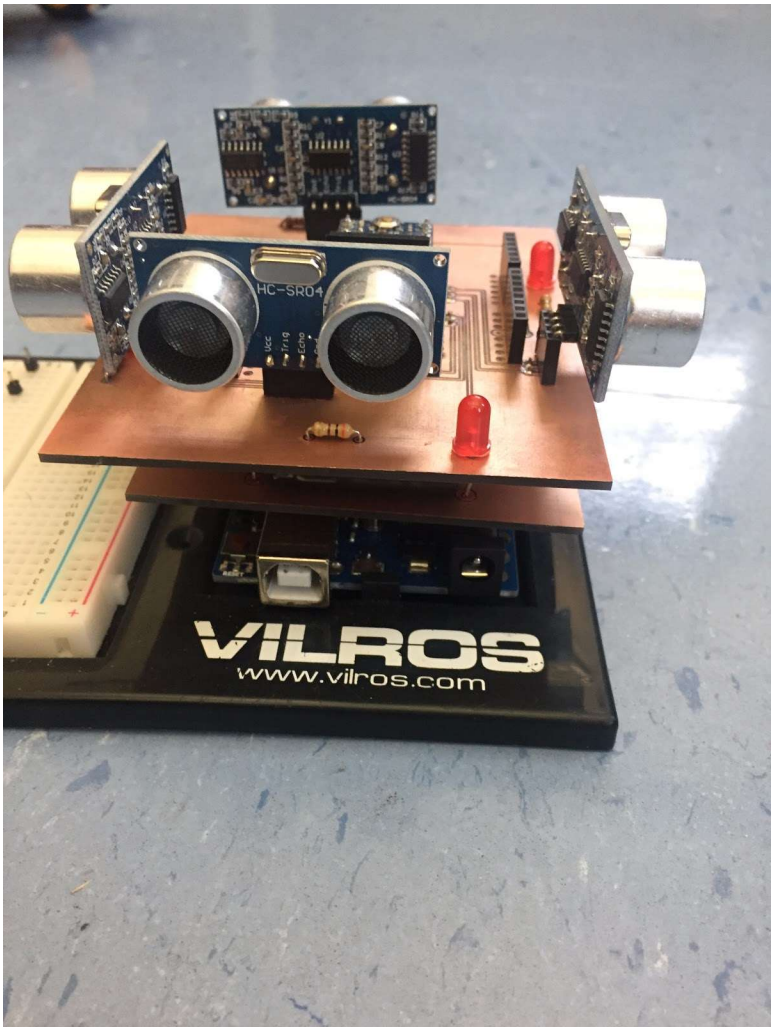


Figure 25: Arduino And Arduino Shields After Image

The figure below is the voltage amplifier we created for the battery packs. As you can see it is very much minimized using the PCB it is small and simply attaches to the back of the battery pack. The switch turn on and off the amplifier and the batteries and then the screw terminals without the wires going in is our output.

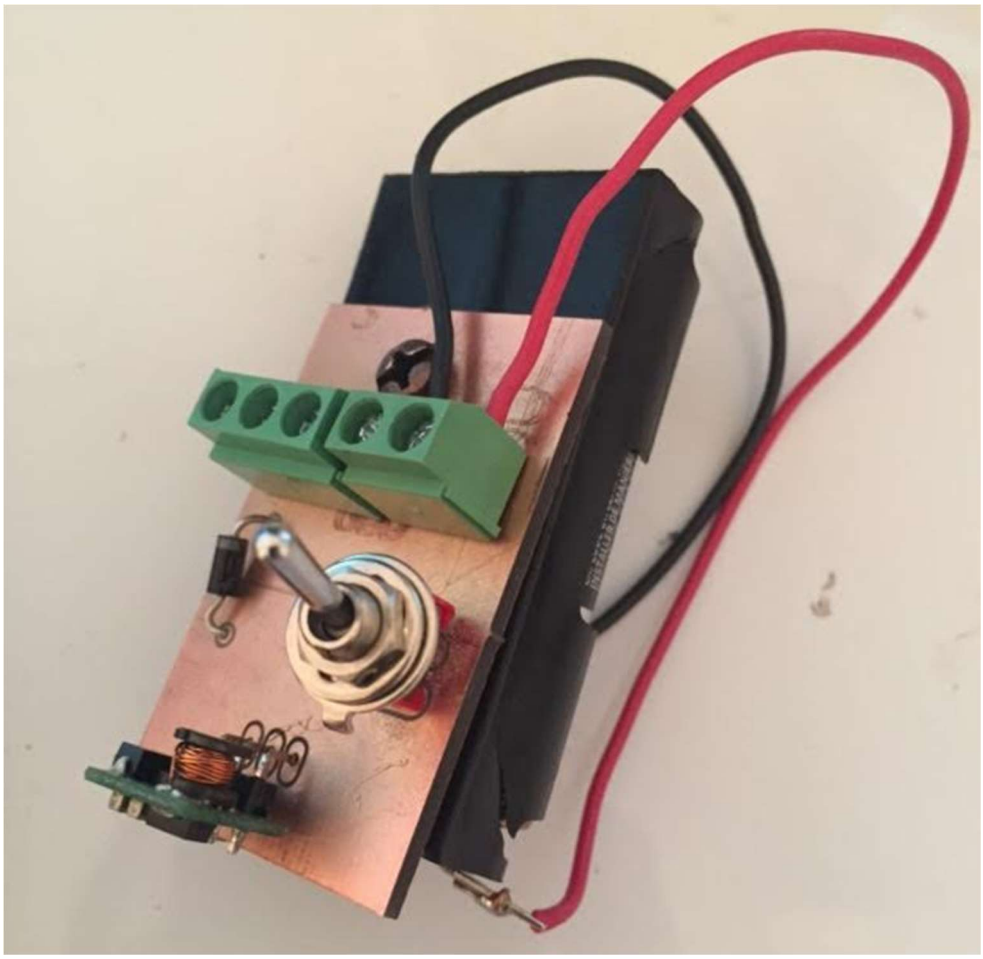


Figure 26: Voltage Amplifier Shield on a battery pack

In the end the robot was very much minimized and improved. We hit most of our goals and learned how to use the PCB machine to create very in depth circuits. The robot will continue to be improved and more work with the PCB machine will be done. We will be able to guide people in use of the machine if needed as well as use our new knowledge in our future endeavors.

References

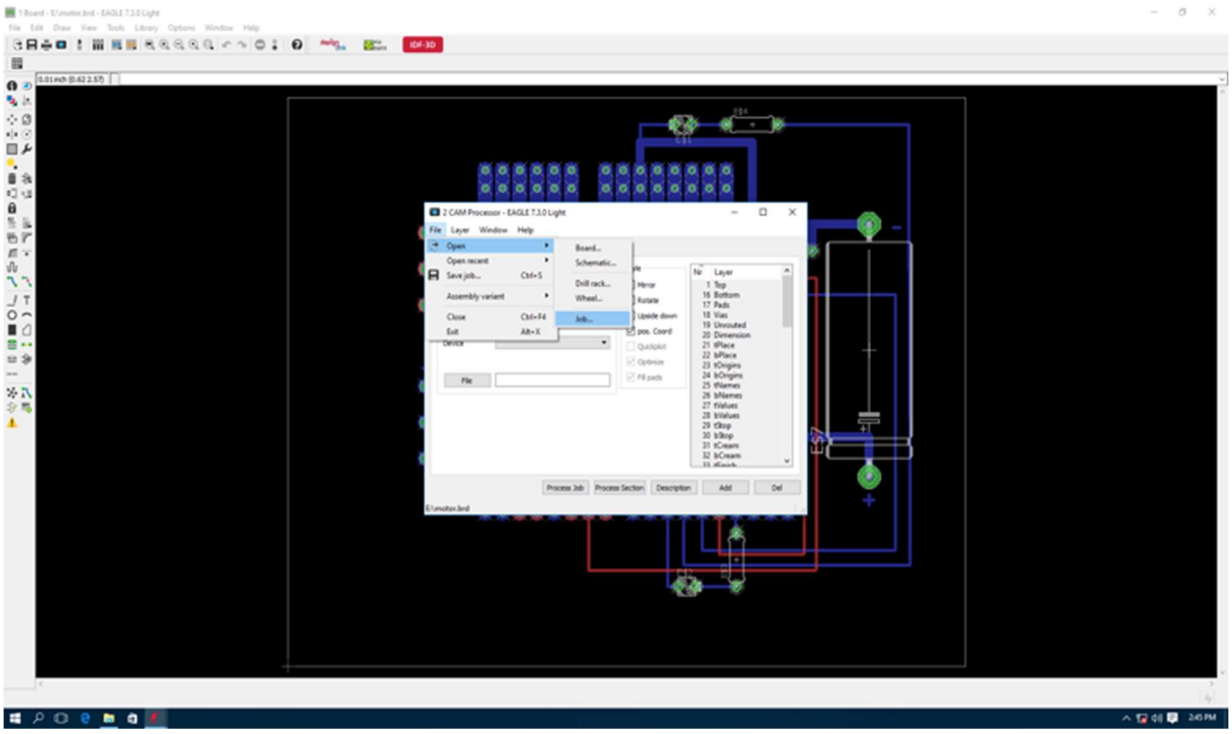
Oskay, Windell. "Weekend Projects with Bre Pettis: Make a Joule Thief." *Evil Mad Scientist Laboratories*. Evil Mad Scientist Laboratories, 2 Nov. 2007. Web. 10 Dec. 2016.

""The Scavenger" A Joule Thief Inspired Boost Regulator." *Electro Tech Online*. Electro Tech Online, 9 June 2013. Web. 10 Dec. 2016.

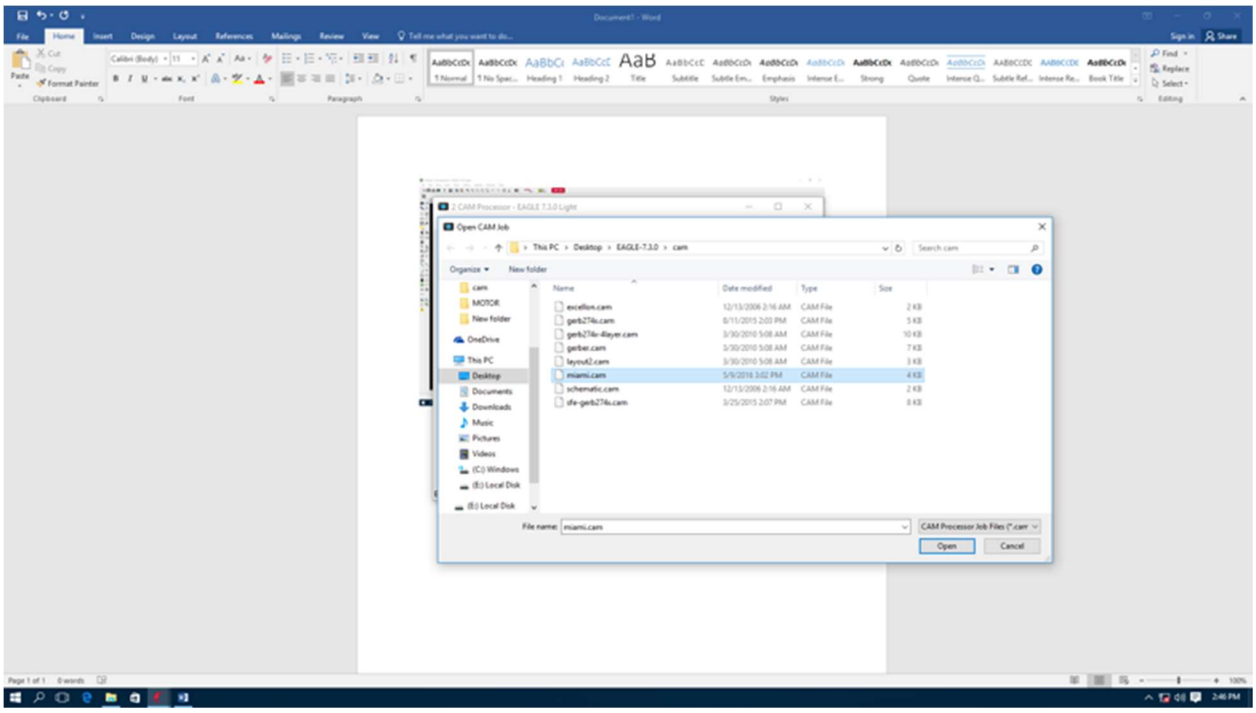
Appendices

Procedure for Printing a PCB

1. First you must design your circuit board in a PCB design software such as Eagle. (For this part, we will be explaining how to cam your board using Eagle since it is the software used in the past.) In Eagle once your board is designed you must go to the main menu and click CAM process. A window will pop up like the image below shows.

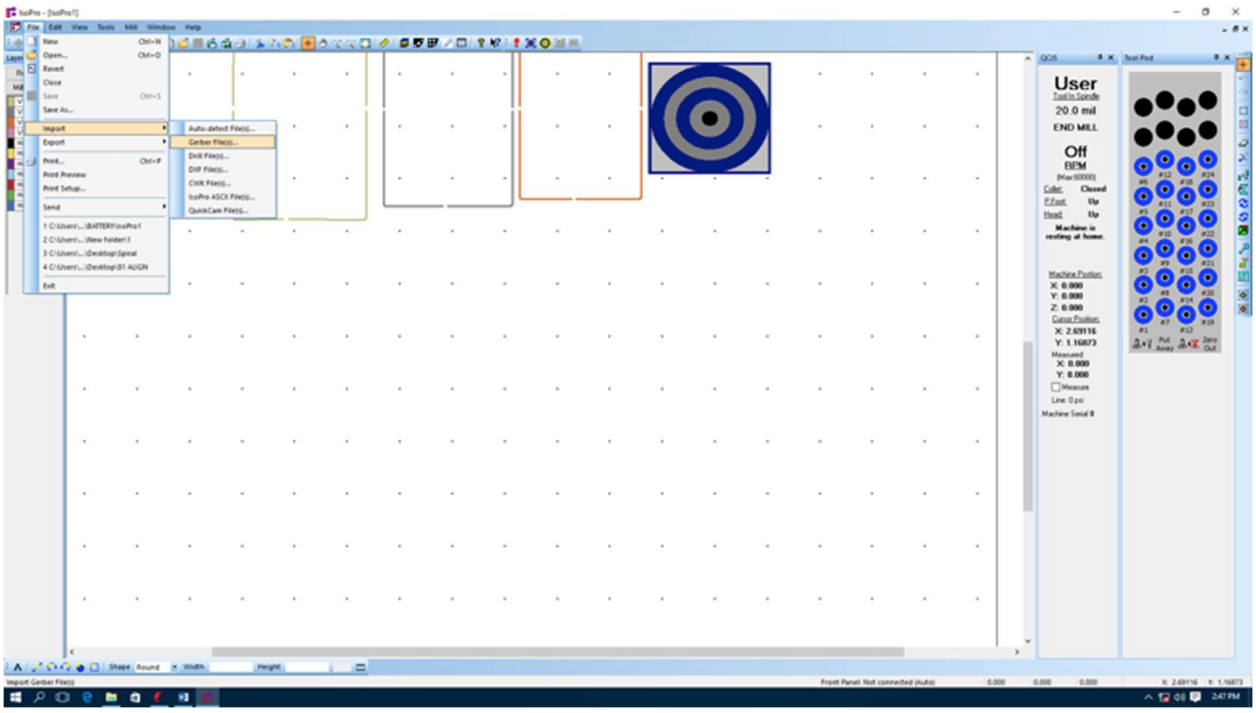


2. Click file--open--job, you will need to select the Miami.cam file.

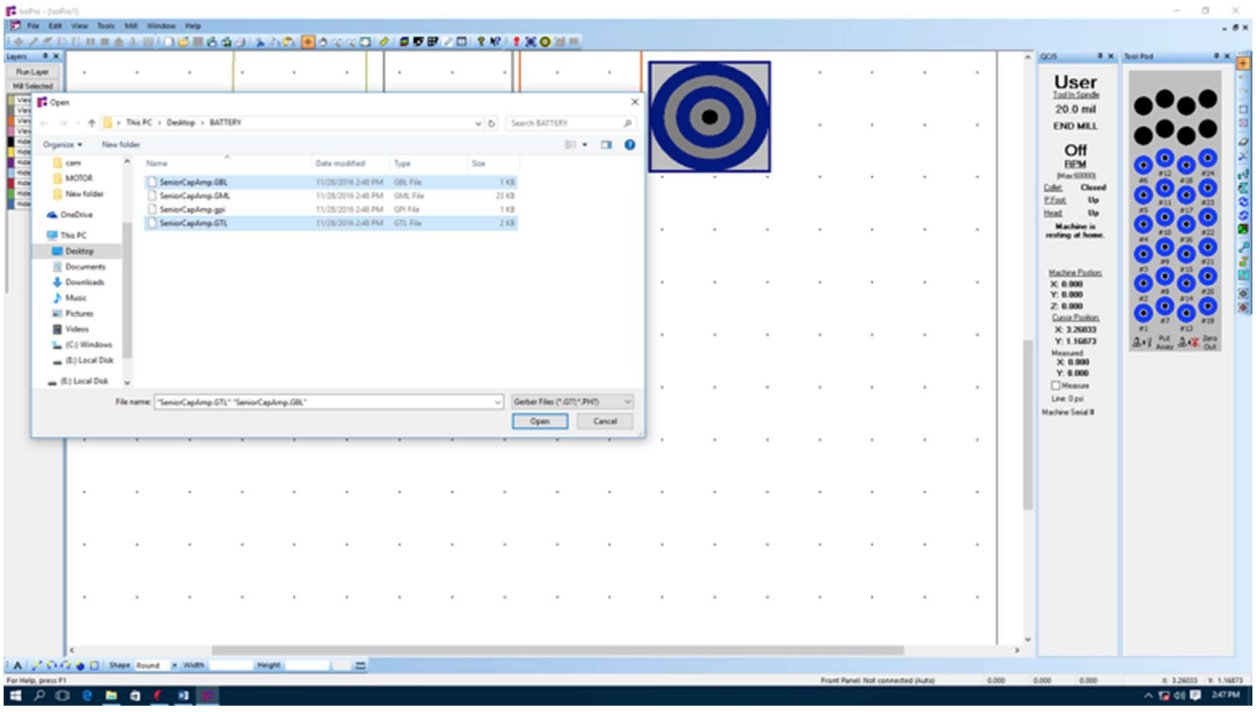


3. After you click open it will then ask you where you would like to save your files that the CAM Process will create. This is your choice but remember where you saved these files, it will be needed in the next part.

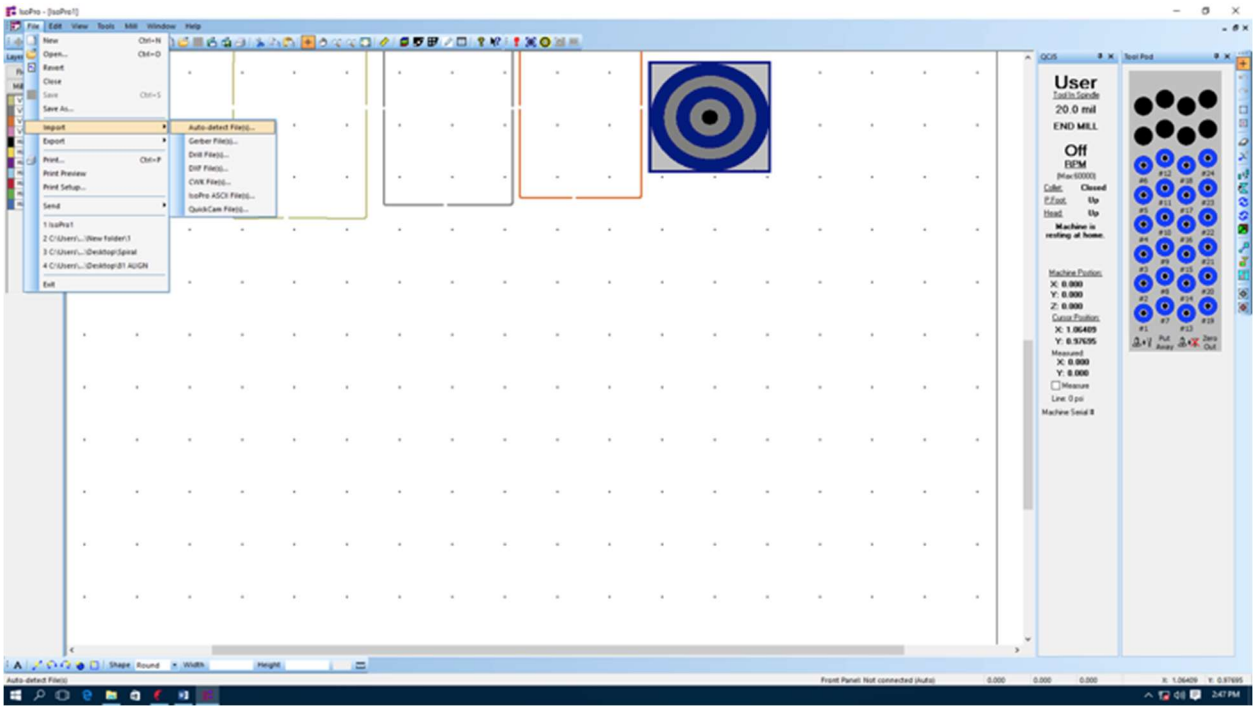
4. Open ISOPRO in computer in lab 275, make sure you have the login and password. A teacher or faculty member who is giving you access to the machine should know it. When the program is opened click file open. This should open a screen with one file choice choose that file. This will then bring up outlines from previous circuits. This image should match the material in the PCB machine. After this, click File--import--Gerber file.



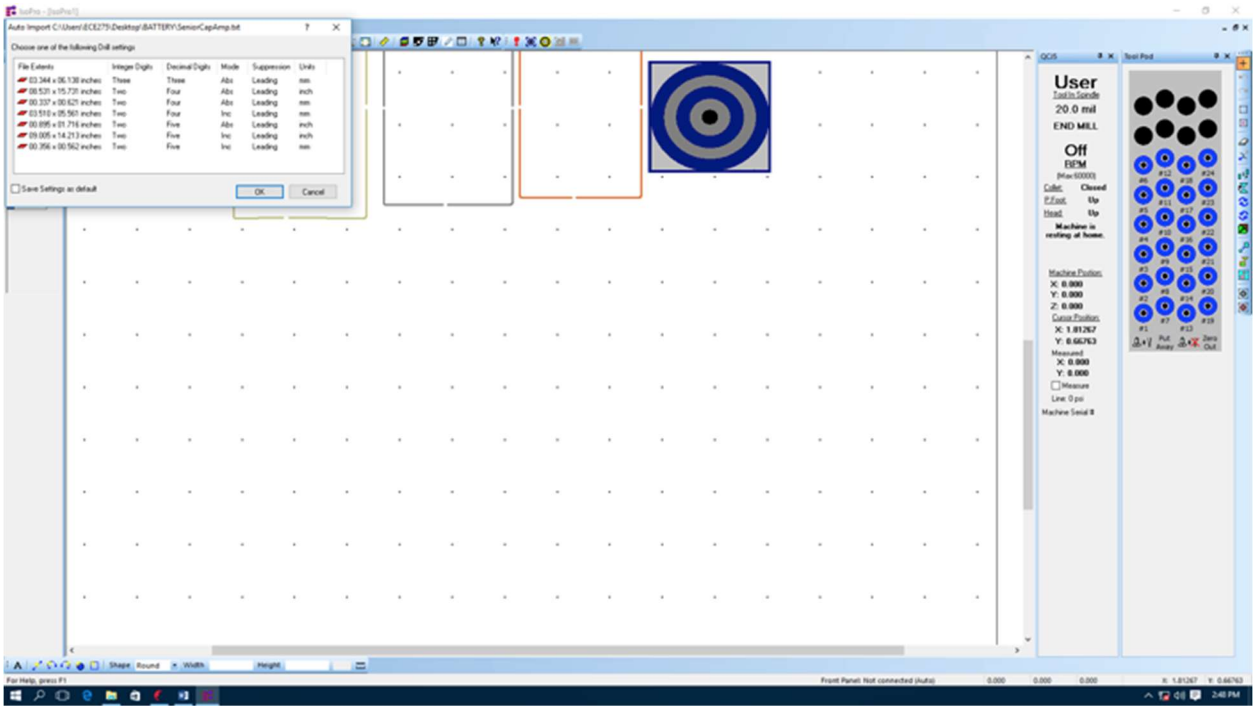
5. Now find where you saved the files when you ran the CAM process. After you navigate to this are you must import two files: The GBL file which represent bottom layer, and the GTL file which represents the top layer.



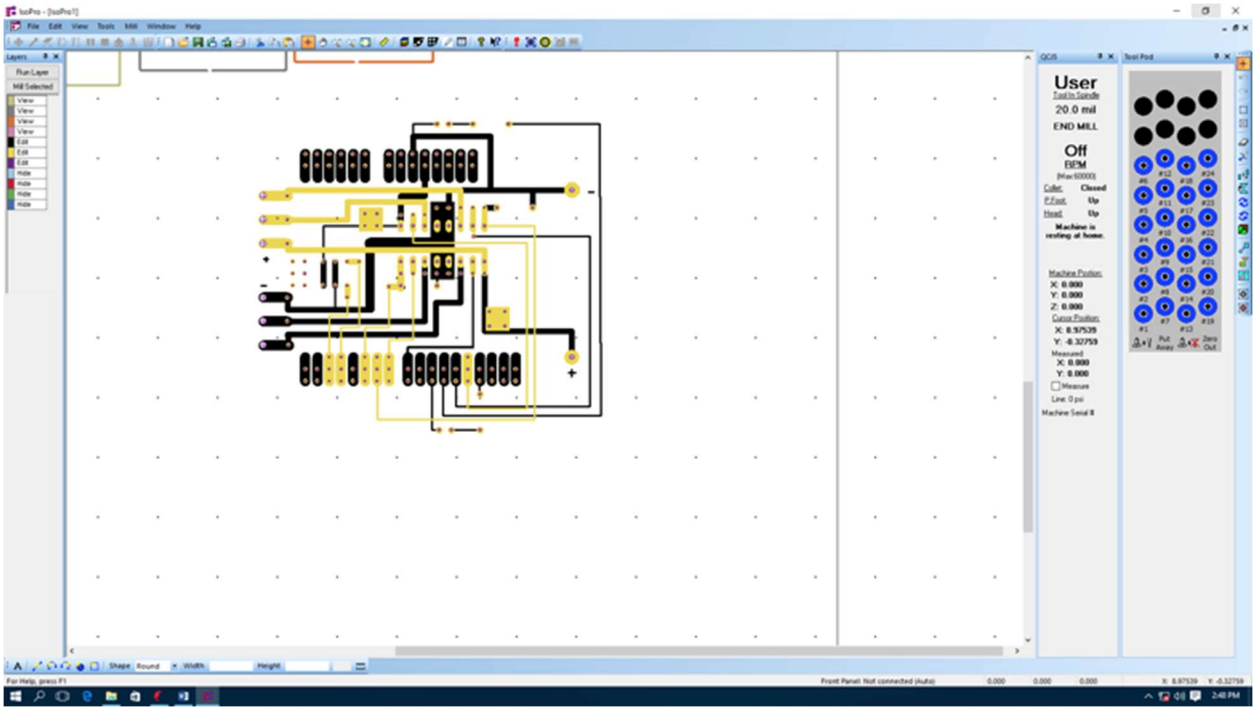
6. Next, click File--import--auto detect file. Many different files will pop up. You must choose your drill file. The extension should .txt.



7. Before you import the drill file ISO Pro will ask you to choose your drill sizes. This is something you must choose per the project you are working on.



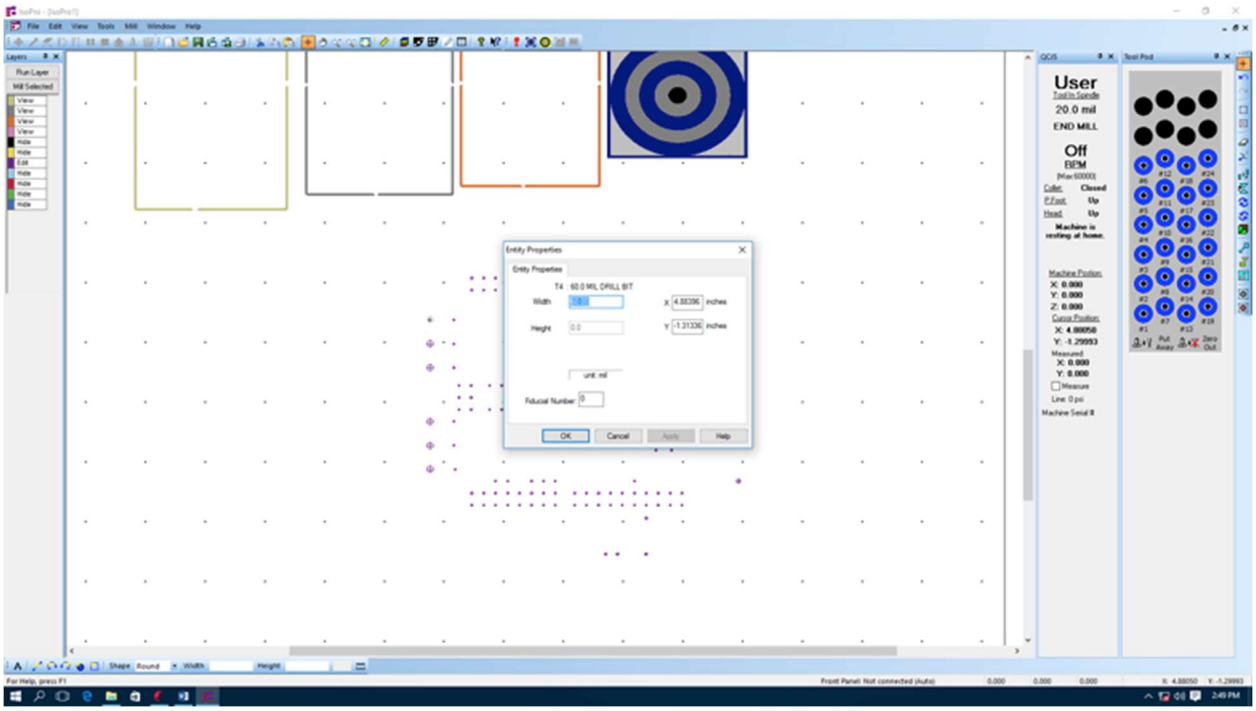
8. After importing all the appropriate files ISO Pro will create layers for them. The program should now look like the image below. (Do not worry if the colors do not match. Your colors may be different each time)



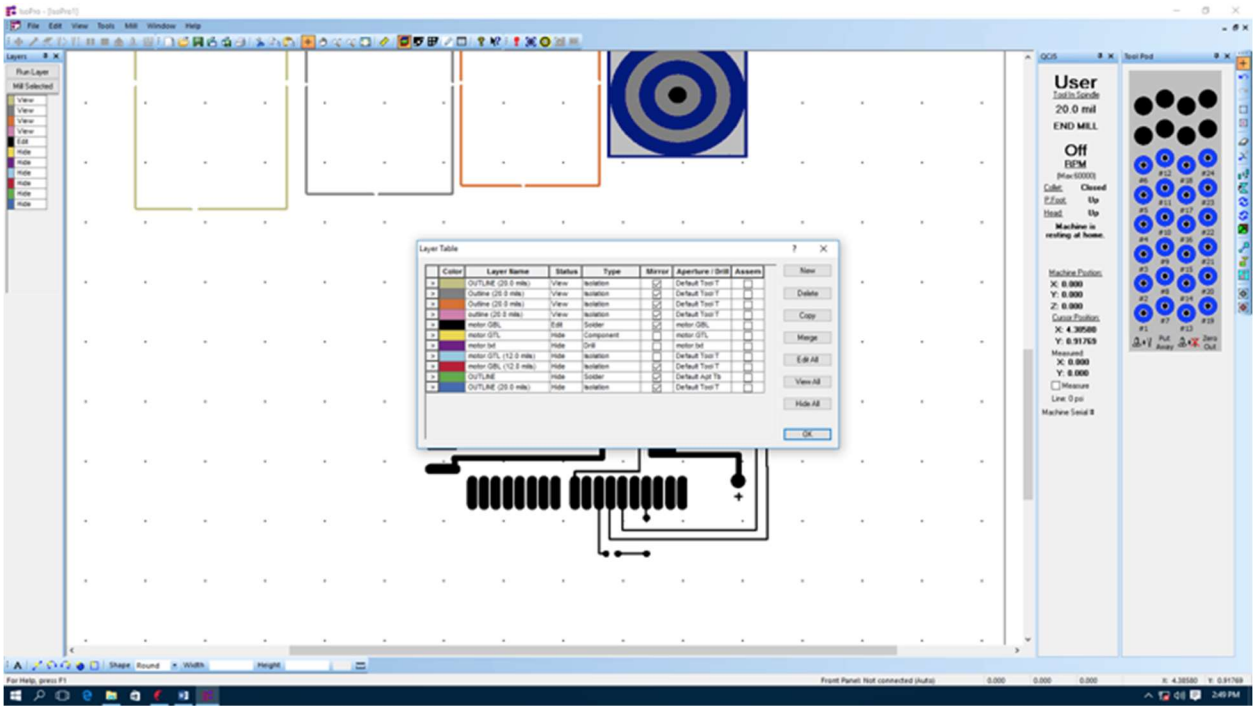
9. The bar on the left-hand side of ISO Pro is your layers. Each layer has a color which represent it. Per what you would like to do you can toggle these layers from, edit, view, and hide. Simply just click on the layer in that bar and it will toggle through these three options.

10. Now you must adjust which hole size you would like to be drilled for each hole. When you created your PCB in Eagle the hole sizes will match the component you chose. In most cases these hole sizes wouldn't have to be changed, just the correct drill would have to be loaded into the machine as it prompts you. Though, our drills are limited, so we do not have all the correct drill bits to match these sizes. This means, we must change each hole size to the closest bit size we have. For instance, if the hole is a 44 but we only have a 40 bit, we must change the drill size to 40. To do this simply right click on the drill hole

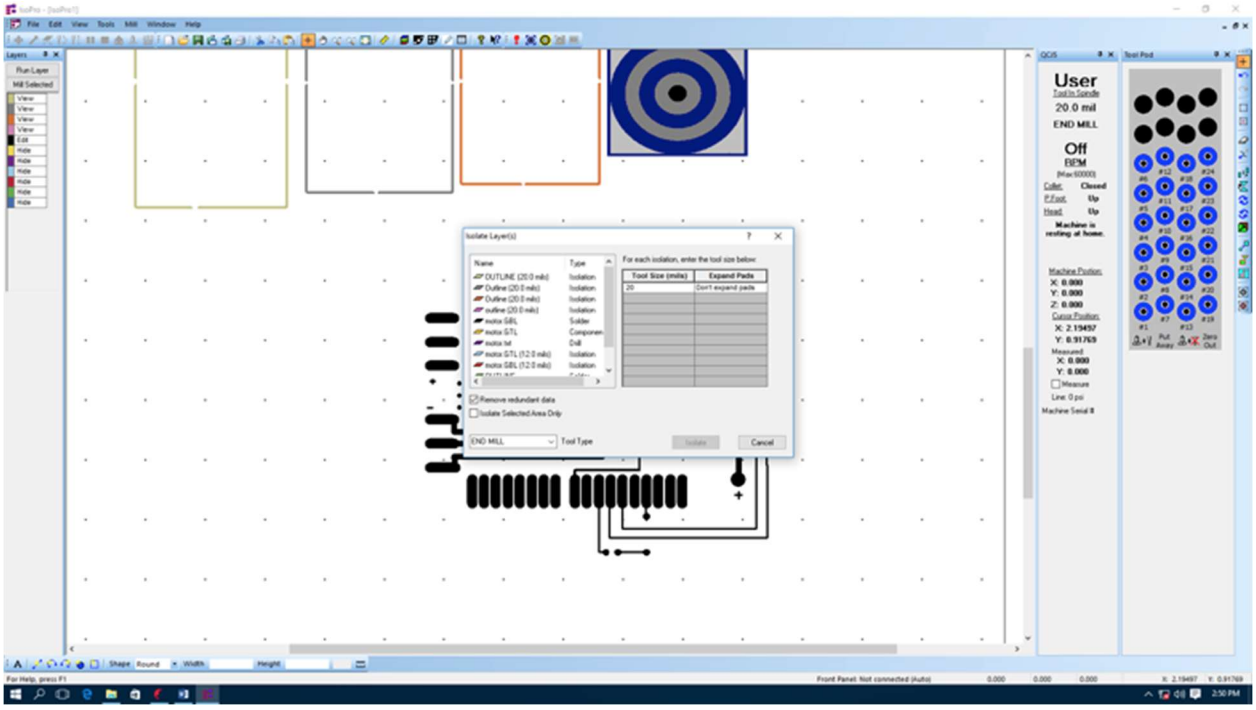
and choose properties. A window will pop up, just change the width to the appropriate size.



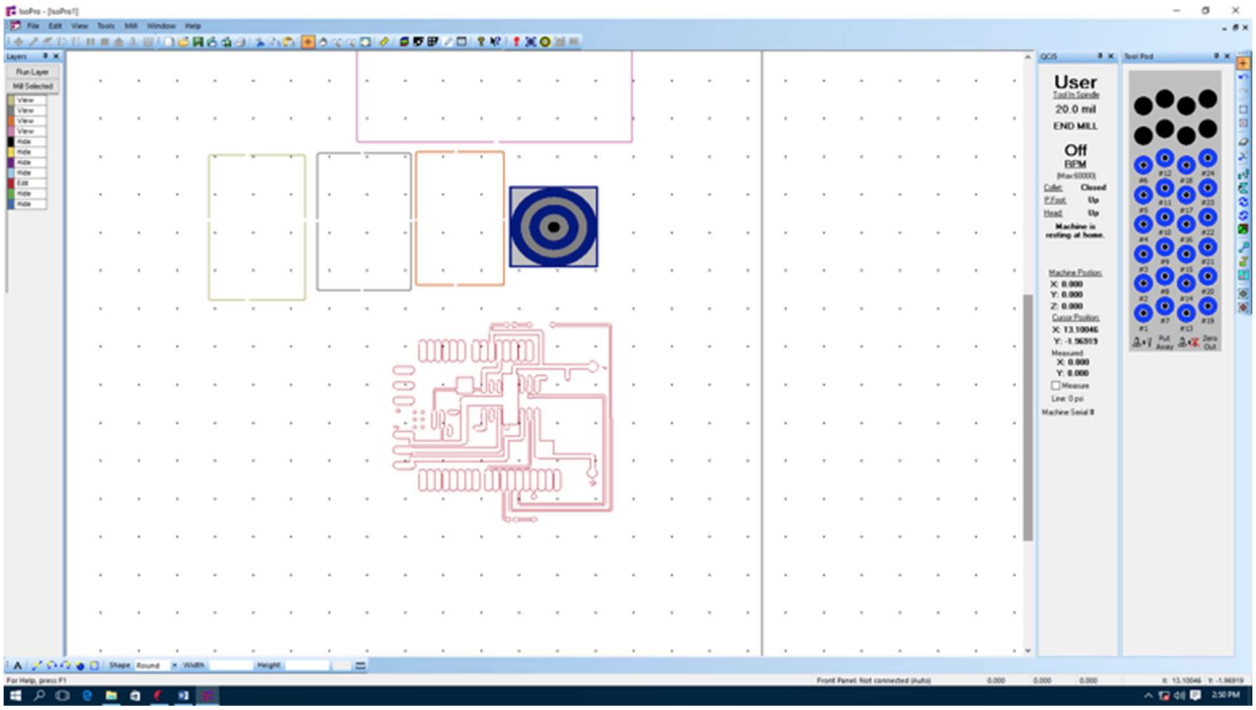
11. Now you must change your bottom layer to solder, as well as make sure it is mirrored. To do this click the layer button in the horizontal toolbar at the top of the program (It is the button that looks like colorful squares layered on top of each other). Find your bottom layer (the one labeled GBL) and change it from component to solder. While you are here, double check that the mirrored check box next to it is checked.



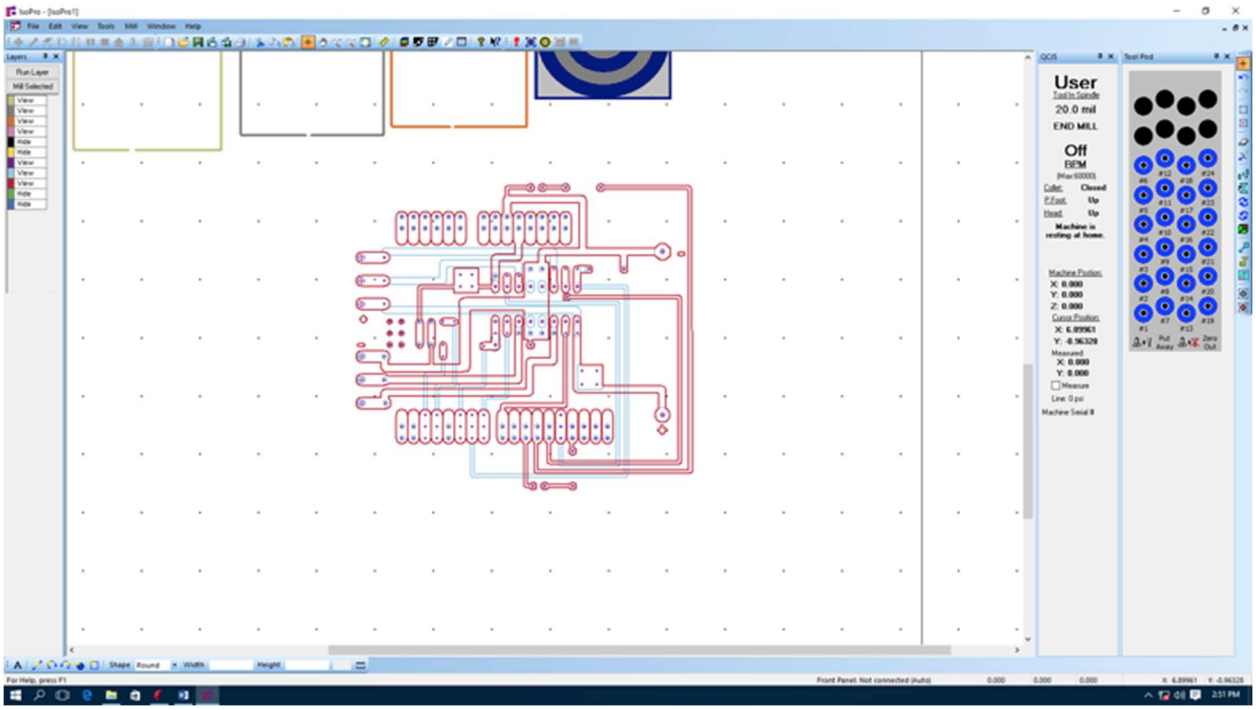
12. Next step is to isolate the top and bottom layer. Click Tool--Isolate, choose the right files you want to isolate (GBL and GTL), you can change the tool size per various situations.



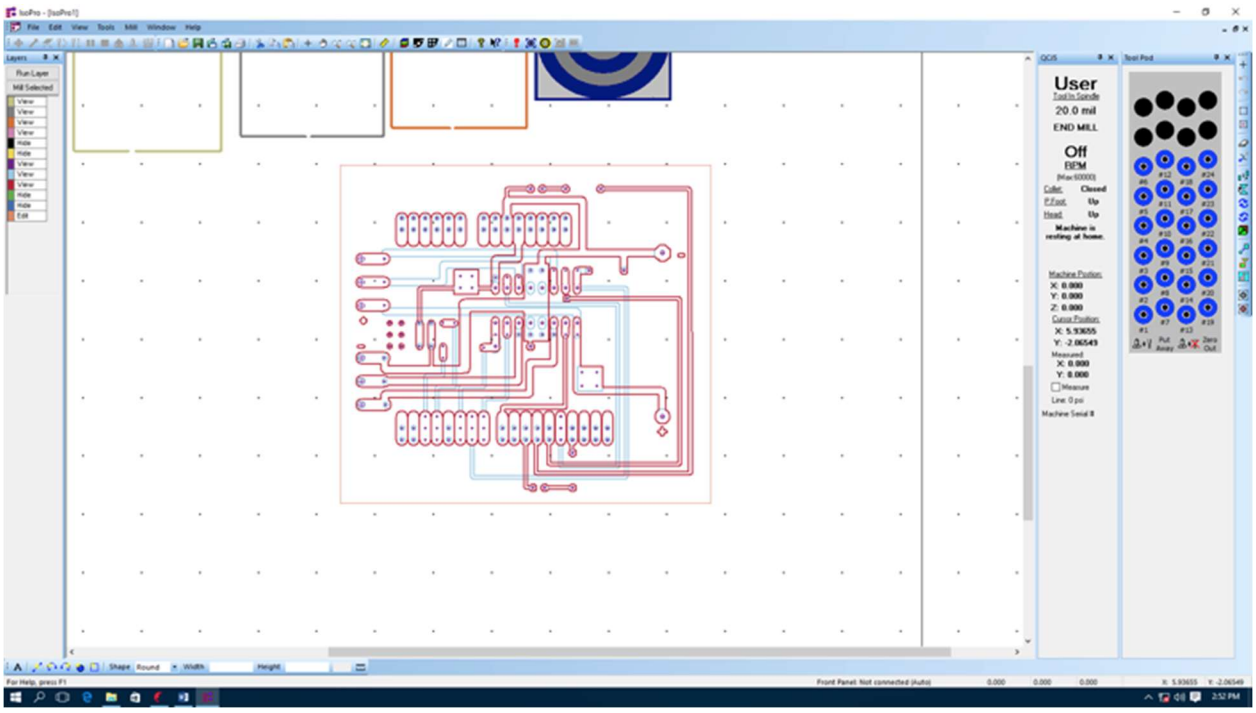
13. After you isolate the top and bottom layer ISO Pro will create an isolation layer for both. When the machine runs later the outline of this isolation is what the machine will trace to create the circuit. You can see this outline in the image below.



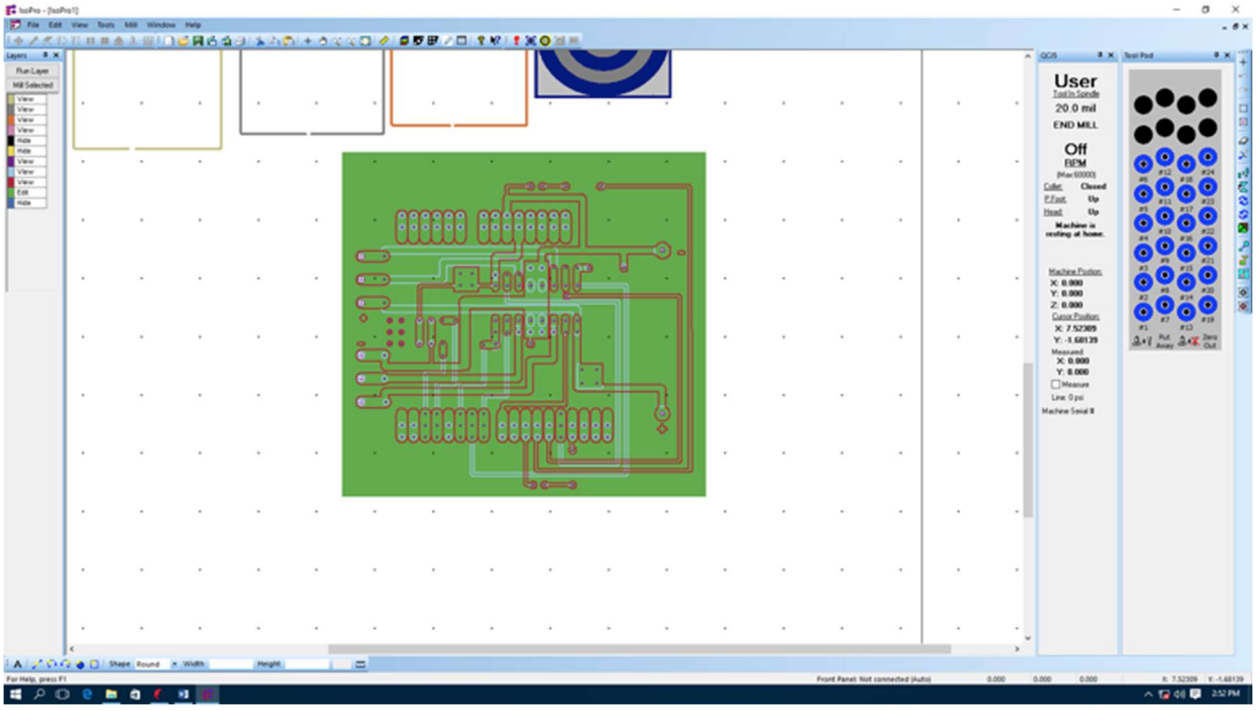
14. The completed PCB design in ISO Pro is below. This image includes the top layer, the bottom layer, and the drill files. All three of these are needed to print the PCB.



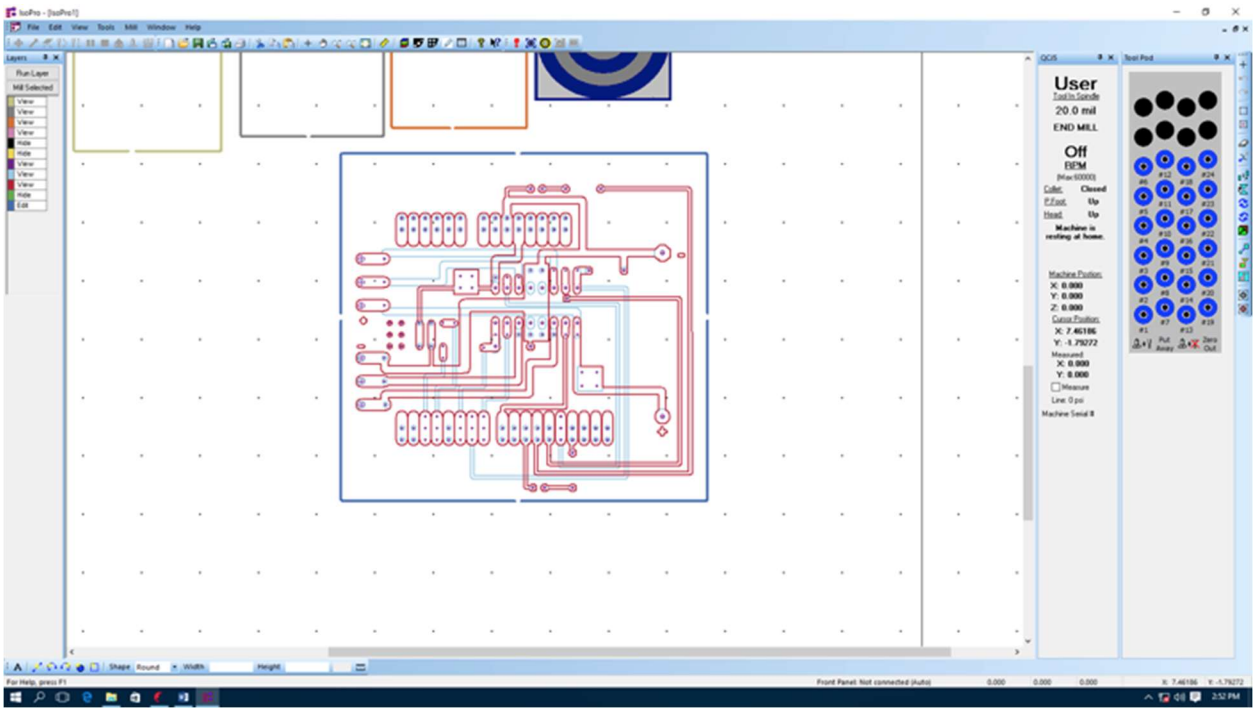
15. Before you print the PCB, you must create an outline around your circuit. The machine will use this outline to cut your circuit out of the bigger piece of material. To create this outline first create a new layer. This can be done in the layer menu where we changed the bottom layer to solder. You will see a bottom to create new layer, click this button. Name this layer what you would like (typically outline is used). After you name it make sure this layer is solder just like the bottom layer. You can now exit the layer screen. Now you must outline your circuit by pressing the rectangle tool in the bottom left tool menu. Outline your circuit with this tool.



16. Now you must select the outline. You can do this by simply clicking the outline, or by toggling all the layers to view. Then toggle the outline layer to edit and select all. Once this is done choose Edit--Convert to polygon to get following image.

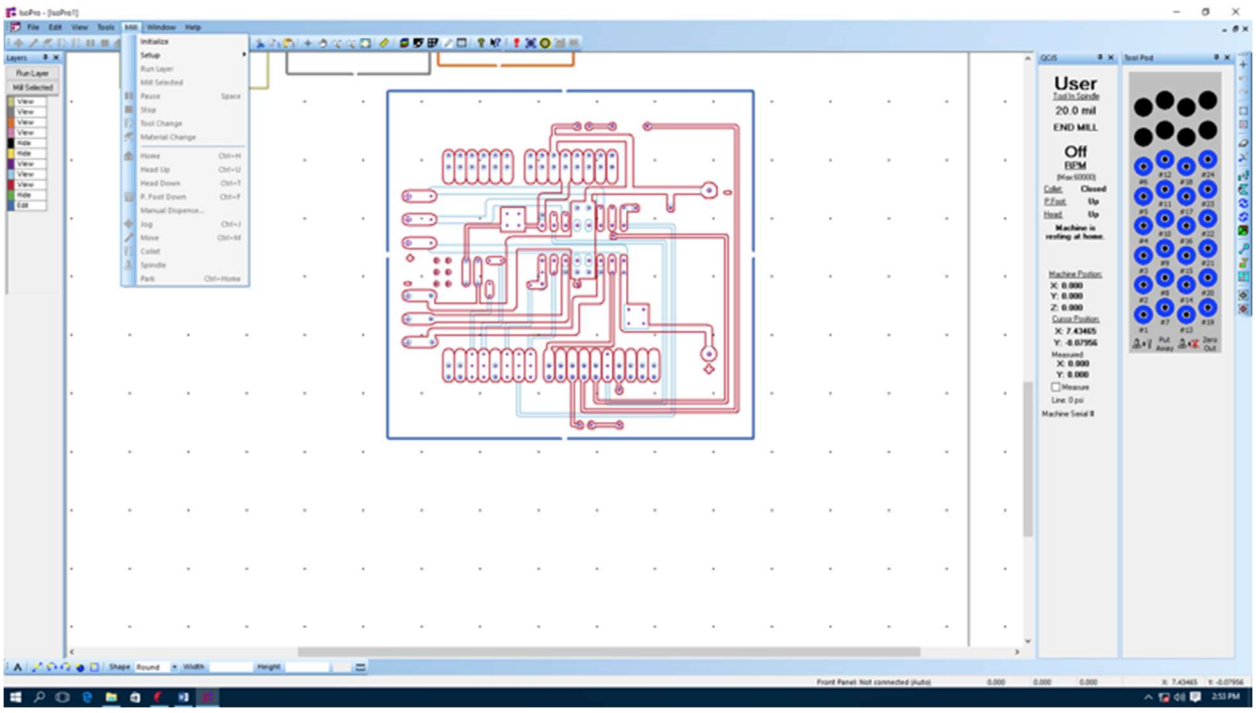


17. Now isolate the outline, by following the same steps used for the top and bottom layer. The only difference will be that you will be choosing the outline layer (or whatever you named it).

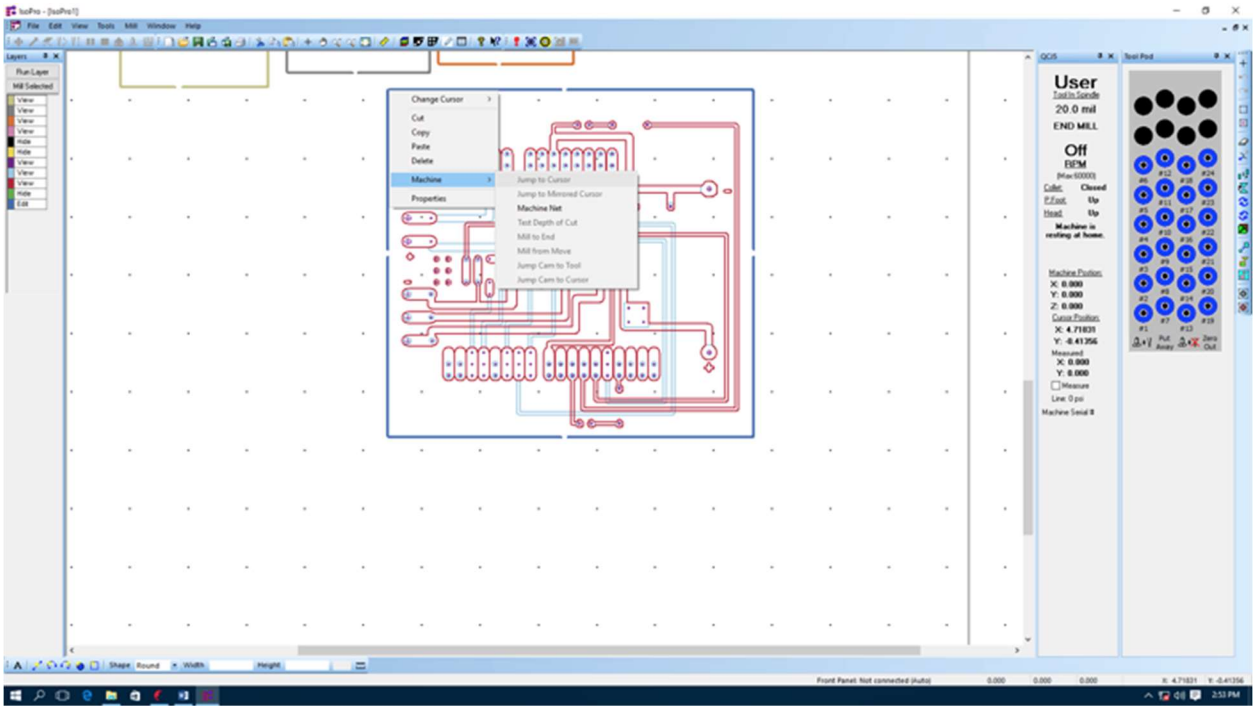


18. Using tool Clip in the right tool menu, to cut some gap in the outline (Make sure all layers are on view except for the outline layer, this one should be on edit. If other layers are on edit you can clip those as well).

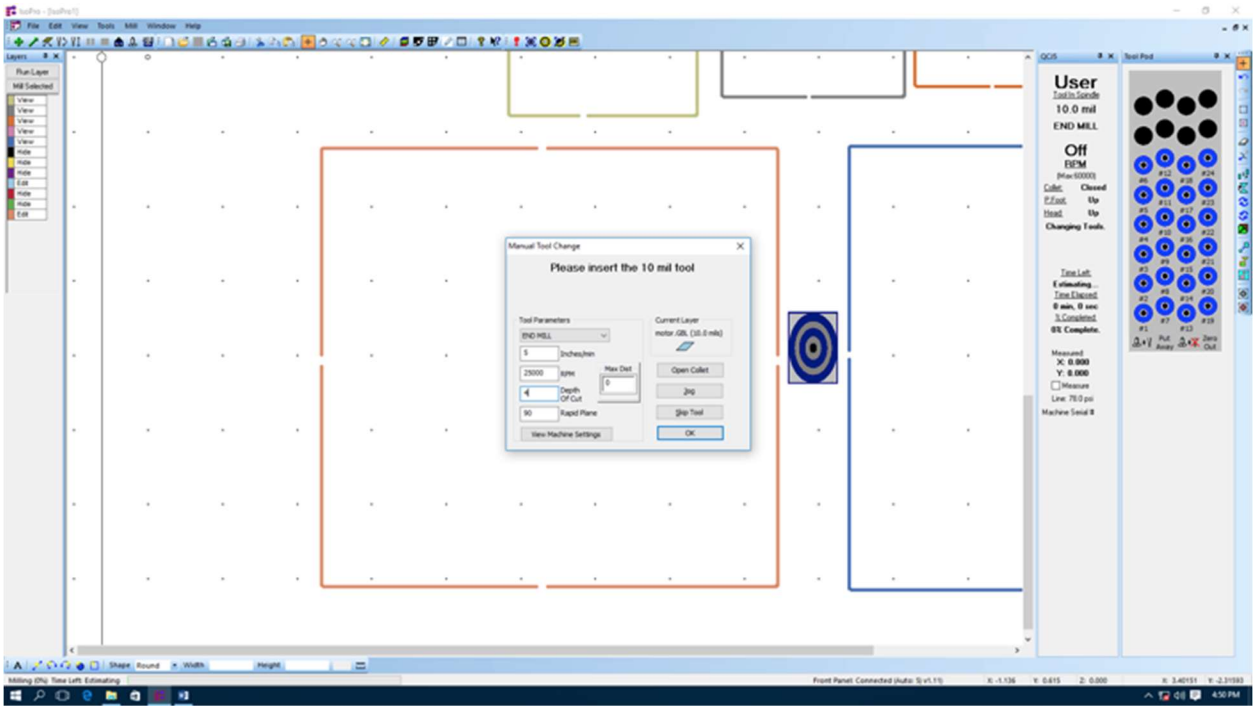
19. Now you can finally turn on the machine. This can be done by pressing the big white button on the bottom right hand side of the machine. When you turn on the machine make sure you also turn on the air valve behind it (It is a level connected to tubing, you can't miss it.) Sometimes the machine and the program won't connect. If this happens make sure you save your file and exit out. Just re-open the program and file with the machine on and it should connect no problem.



20. Now make sure that your board is positioned on the screen to waste as least material as possible. To do so place closest to the other outlines in the program. Then make sure the bullseye is right next to your circuit. You can move it simply by clicking on it. Now make sure your board is in the right position Check your printed board position by right click the corner, click machine---jump to cursor, the machine will move to this position. Make sure the machine is in the spot you would like for the PCB to be printed at.



21. After making sure the position of your board is okay, change the pressure to 10 20 20. These buttons are on the machine itself in the front. Simply use the up and down arrows to reach the correct pressure. Once this is done then you can run your first layer. First, you want to drill your holes. To do this click Mill---run layer, choose the drill file. A pop up menu will appear that looks like the image below. Make sure the depth is at 75, everything else should stay the same. Then click okay. A screen will now popup to prompt you to insert a specific drill size. Do so by using the tool button on the front of the machine. Once the drill bit is in press the machine press okay. You will follow this process for each layer. It is vital that you re-initialize the machine after each layer is printed. For milling, the parameters are shown below.



22. When you completed the component side, flip the board, and make sure the isolation of the bottom layer and the outline are both mirrored in layer menu. Use the same procedure as above to print this layer. Then when you print the outline, change the depth to 75.

Code for the Arduino to run real time MATLAB analysis of voltage amplifier

```
/*
 ReadAnalogVoltage
 Reads an analog input on pin 0, converts it to voltage, and prints the result to the serial monitor.
 Graphical representation is available using serial plotter (Tools > Serial Plotter menu)
 Attach the center pin of a potentiometer to pin A0, and the outside pins to +5V and ground.
```

```
 This example code is in the public domain.
*/

/* This code was modified by Jonathan Kosir
*  It flips our relay in our circuit as well
*  as prints out the voltage we would like
*  matlab to read through the serial port
*/
// the setup routine runs once when you press reset:
void setup() {
 // initialize serial communication at 9600 bits per second:
 Serial.begin(9600);
 pinMode(2, OUTPUT);
}
 int count = 0;
int  flip = 0;
// the loop routine runs over and over againorever:
void loop() {
 count = count + 1;
 if(count >= 12){
   if(flip == 0){
     flip = 1;
   }
   else{
   flip = 0; }
   count = 0;
 }
 if(flip == 1){
   digitalWrite(2, LOW);
 }
 else
 {
 digitalWrite(2, HIGH);
 }
```

```
// read the input on analog pin 0:
int sensorValue = analogRead(A0);
int sensorValue1 = analogRead(A1);
// Convert the analog reading (which goes from 0 - 1023) to a voltage (0 - 5V):
float voltage = sensorValue * (5.0 / 1023.0);
float voltage1 = sensorValue1 * (5.0 / 1023.0);
// print out the value you read:
Serial.println(voltage);
Serial.println(voltage1);
delay(5000);
}
```

MATLAB code for Thevenin and Norton equivalents

```
% Jonathan Kosir Miami University
% December 1st 2016
% Thevenin and norton equivalents graphed
% by reading input and output voltage and resistance from file
% must name files by the load resistance
% file data must be in three columns (If using freq - only two columns if
% not- frequency read can be used to graph input vs frequency or so on)
% Voltage input column one
% Voltage out column two
% Frequency column three
% All values must be double with no char values

%Read all data files in folder
% GetAllFiles is a function found online deveopers name uknown
fileList = getAllFiles('C:\Users\Jon Kosir\Documents\Documents\Senior Design Fall
2016\capstone\Small Voltage Amp');
filenNameLength = 91;
%Turn on grid in graph
grid on
```

```
%Inizilize how many data points take
NumVoltsIn = 25;

% Initilize number of files/ The number of load resistances which were
% analyzed
NumLoadResistors = length(fileList);

% Loop through all load resistors data
for k=1:NumLoadResistors

% Create a matrix with all the load resistances by reading file names
test = char(fileList(k));
number = test(fileNameLength:end-4);

% Take all data fraom resistance files
M = dlmread(test);
VoltageIn = M(1:NumVoltsIn,1);
VoltageOut = M(1:NumVoltsIn,2);
%Freq = M(1:size,3);

% Pre allocate matrices
if(k == 1)
voltageInM = zeros(NumVoltsIn,1);
voltageOutM = zeros(NumVoltsIn,1);
voltageDiffM = zeros(NumVoltsIn,1);
end

% Create matrices for every resistance file
VoltAdd = VoltageIn(1:NumVoltsIn,1);
VoltOutAdd = VoltageOut(1:NumVoltsIn,1);
Resistance(k,1) =  str2double(number);
voltageInM = [voltageInM VoltAdd];
voltageOutM = [voltageOutM VoltOutAdd];
end
```

```
% Delete uneeded data
voltageInM = voltageInM(:,2:end);
voltageOutM = voltageOutM(:,2:end);
voltageDiffM = voltageOutM - voltageInM;

%Not used for thevenin and norton program
for k=1:NumLoadResistors
  for j=1:NumVoltsIn
i(j,k) = voltageOutM(j,k) / Resistance(k,1);
%iM = [iM i];
  end
end
%-----------------------------------------------

% Do math
xM = [1 1 1];
for k=1:NumVoltsIn
thisNum = NumVoltsIn -1;
A(1:NumLoadResistors, 1) = 1;
negativeCurrent = -i;
tem = negativeCurrent(k,:);
A(:,2) = tem;
B = voltageOutM(k,:)';
X = A\B;
X = X';
C = X(1)/X(2);
X = [X C];
xM = [xM ; X];
end

% Graph thevenin and norton data
% If program changed slightly can be used to graph other data from
% files
```

```
xM = xM(2:26,:);
ThevV = xM(:,1);
ThevR = xM(:,2);
NortI = xM(:,3);
figure();
plot(VoltageIn, ThevV, 'red');
xlabel('Voltage In');
ylabel('Thevenin Voltage');
titleName = strcat('Voltage In Vs Thevenin Voltage');
title(titleName);
clear title;
grid on;
figure();
plot(VoltageIn, ThevR, 'red');
xlabel('Voltage In');
ylabel('Thevenin Resistance');
titleName = strcat('Voltage In Vs Thevenin Resistance');
title(titleName);
clear title;
grid on;
figure();
plot(VoltageIn, NortI, 'red');
xlabel('Voltage In');
ylabel('Norton Current');
titleName = strcat('Voltage In Vs Norton Current');
title(titleName);
clear title;
grid on;
--------------------------------------------------------------------------------

%Get all files function
function fileList = getAllFiles(dirName)

 dirData = dir(dirName);      %# Get the data for the current directory
```

```
dirIndex = [dirData.isdir];  %# Find the index for directories
fileList = {dirData(~dirIndex).name}';  %'# Get a list of the files
if ~isempty(fileList)
  fileList = cellfun(@(x) fullfile(dirName,x),...  %# Prepend path to files
                     fileList,'UniformOutput',false);
end
subDirs = {dirData(dirIndex).name};  %# Get a list of the subdirectories
validIndex = ~ismember(subDirs,{'.','..'});  %# Find index of subdirectories
                                     %#   that are not '.' or '..'
for iDir = find(validIndex)                  %# Loop over valid subdirectories
  nextDir = fullfile(dirName,subDirs{iDir});    %# Get the subdirectory path
  fileList = [fileList; getAllFiles(nextDir)];  %# Recursively call getAllFiles
 end

end
```

Code for serial read in analysis for store bought voltage component

```
%%real time data plot from a serial port
% This matlab script is for ploting a graph by accessing serial port data in
% real time. Change the com values and all variable values accroding to
% your requirements. Dont forget to add terminator in to your serial device program.
% This script can be modified to be used on any platform by changing the
% serialPort variable.
% Author: Moidu thavot.

% File Modified By Jonathan Kosir
% Can Read in voltage in voltage out
% With resistance caluclate current over load
% Graphs all three values
% Must be ran with matlab 2016

%withought amplifier = 21.178Ohms current  = .1473
%with = 33.9 cuurent .14877
```

```
% Create the serial object
serialPort = 'COM3';
serialObject = serial(serialPort);
fopen(serialObject);

% Set the instrument in remote mode
fprintf(serialObject,'SYSTEM:REMOTE');

% Variables
time = now;
voltage = 0;
voltage2 = 0;
resistance = 33.9;
current = 0;

% Set up the figure window
fileName = fopen('C:\Users\Jon Kosir\Desktop\testFile.txt','w');
figureHandle = figure('NumberTitle','off',...
  'Name','Voltage Characteristics',...
  'Color','black','Visible','off');

% Set axes
axesHandle = axes('Parent',figureHandle,...
  'YGrid','on',...
  'YColor','blue',...
  'XGrid','on',...
  'XColor','green',...
  'Color','black');

hold on;

% Create Plot handle for Left Y Axis
yyaxis left;
```

```
plotHandle = plot(axesHandle,time,voltage,'Marker','.','LineWidth',1,'Color','blue');
plotHandle2 = plot(axesHandle,time,voltage2,'Marker','.','LineWidth',1,'Color','blue');

% Create Plot handle for Right Y Axis
yyaxis right;
plotHandle3 = plot(axesHandle,time,current,'Marker','.','LineWidth',1,'Color','red');



% Create xlabel
xlabel1 = xlabel('Time','FontWeight','bold','FontSize',14,'Color','green');

% left Create ylabel
yyaxis left;
ylabel('Voltage','FontWeight','bold','FontSize',14,'Color','blue');

% right create ylabel
yyaxis right;
ylabel('Current','FontWeight','bold','FontSize',14,'Color','red');

xlim(axesHandle,[min(time) max(time+0.001)]);

%pl =  plt(time,[voltage; voltage2; current;],'Xlim',xlim,...
%  'LabelX','Time','LabelY',{'Voltage' 'Current'});

% Create title
title('Voltage Characteristics','FontSize',15,'Color','white');

% Set the time span and interval for data collection
stopTime = '10/22 21:53';
timeInterval = 0.005;
% Collect data
count = 1;
while ~isequal(datestr(now,'mm/DD HH:MM'),stopTime)
```

```
  % Collect data from arduinos serial
  time(count) = datenum(clock);
  fprintf(serialObject,'MEASURE:VOLTAGE:DC?'); % To measure current the command is
MEASURE:CURRENT:DC?
  voltage(count) = fscanf(serialObject, '%f');%#ok<SAGROW>
  voltage2(count) = fscanf(serialObject, '%f');%#ok<SAGROW>

  % Calculate current
  current(count) = voltage2(count)/resistance;%#ok<SAGROW>

  %pl = plt('edit',time,[voltage; voltage2; current;]);

  % Plot left Y axis
  yyaxis left;
  set(plotHandle,'YData',voltage,'XData',time);
  set(plotHandle2,'YData',voltage2,'XData',time);

  %Plot right Y axis
  yyaxis right;
  set(plotHandle3,'YData',current,'XData',time);

  set(figureHandle,'Visible','on');

  datetick('x','mm/DD HH:MM');

  % Used to print data to file in case program throws error and
  % Graph is lost
  fprintf(fileName, '%f\t', voltage(count));
  fprintf(fileName, '%f\t', voltage2(count));
  fprintf(fileName, '%f\r\n', current(count));
  pause(timeInterval);
  count = count +1;

end
```

```
% Put the instrument in local mode
fprintf(serialObject,'SYSTEM:LOCAL');

% Clean up the serial object
fclose(fileName);
fclose(serialObject);
delete(serialObject);
clear serialObject;
```