Sub meter application design challenge for metering commercial buildings

Presented to:
Kunle Adeleye
6046234587
kunle.adelaye@bchydro.com

By:
Faiza Yahya: Group Leader
Katherine McLauchlan: Group Recorder
Antonio Wang: Time Keeper
Carlo Cossette: Facilities Coordinator

April 10th, 2014

Executive Summary

A power metering system is designed to measure apparent and real power, voltage, current and power factor for all 3-phases of a building. The device wirelessly communicates using a mesh network from the measurement station to the base station. The system is low-cost, accurate, and able to maintain reliable communication.

The measurement system is composed of a hardware schematic for the voltage and current values. An Arduino Mega is used to accept the voltage and current waves and execute the measurements with appropriate calculations. The measurement station consists of the voltage and current hardware schematics and a radio module for sending the data wirelessly to the base station. The base station contains a receiving radio module and storage circuitry. Both stations are enclosed in aluminum boxes that act as Faraday Cages to eliminate noise.

A prototype is constructed for $100 per measurement device and $50 per base station. The power metering system consists of one measurement station and two receiving stations to demonstrate a mesh network. Light aluminum enclosures are required to prevent noise interference and protect against external electric fields. The result of the system therefore displays accurate results.

Table of Contents

Definitions

Measurement Device: A device that measures three phase Watt-hour electrical energy

Base Station: A device located at a distance from the measurement device and that wirelessly receives data information from every measurement devices, stores it, and interfaces with PCS to give the user feedback

Measuring System: A system consisting of a base station and one or more measurement devices. It operates in a mesh network where the measurement stations wirelessly communicates data regarding power consumption.

Measurement Point: A location where a measurement device directly measures power from the 3-phase feedlines' entry point.

Mesh Network: A wireless communication method where measurement devices are linked together downstream to the base station.

Radio Module: Device that connects to communicating circuits in order to allow wireless communications.

Feedline: Power cable located at the entry point of the facility.

Sending Station: Consists of an Arduino Mega to compute additional measurement data and wirelessly send the data to the Receiving Station.

Receiving Station: Consists of an Arduino Uno to receive the data at a remote location.

GUI: Graphical User Interface on a personal computer.

SQL: Is a structured query language used for managing the database.

UDP/TCP Ports: UDP is a User datagram protocol and TCP is transmission datagram protocol and they are needed for successful connection with SQL Server Database.

Introduction

A low cost wireless energy metering system is designed that performs basic electrical measurements and communicates wirelessly with a central base station as well as a remote location (ie laptop computer) within the building. The metering system consists of three or more substations measuring three phase Watt-hour power flow, voltage and current. A mesh networking technique is used to wirelessly transfer the measured data to each node in the system. All device communication and transmission adheres to open protocol standards. Our testbed is a large building located on the UBC campus.

The purpose of the energy metering system is to measure and analyze electrical data to obtain a better understanding of the building energy consumption. In order to improve energy efficiency, a reliable method of measuring energy usage must be established. Implementing a metering system will indirectly enhance the building systems performance. The most significant challenge in developing a high level metering system is the cost. The main objective of the project must be to limit the cost factor.

A functional block diagram of the power metering system is displayed in Figure X. The measurement station is the first circuit established in full system. It's purpose is to initially step down the voltage using high rated transformers to result in a lower and safer voltage. The measurement circuits compose voltage and current waveforms and are utilized as inputs to the sending station. The sending station consists of the microcontroller Arduino Mega to process the voltage and current waveforms and compute the additional required measurement values. An Xbee module is the used for wireless communication to send the data to the receiving station. The receiving station consists of an Arduino Uno and Xbee module to process the data received. The last component of the system is the user interface which cleanly displays the processed information on a GUI.

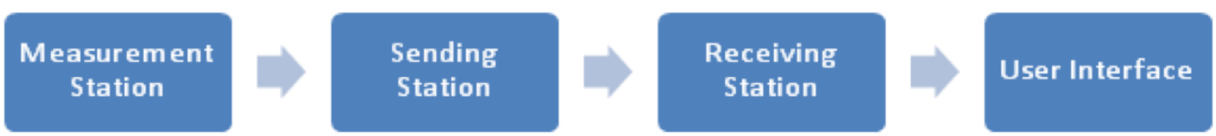


Figure 1: Functional Hardware Block Diagram

Specifications

Voltage Measurement	0-128V, 3-phase Wye
Current Measurement	35A
Frequency	60Hz
Accuracy	The ANSI C12 sets guidelines for the accuracy measurement for the metering system.In order to claim compliance, the meter must pass the accuracy tests listed in ANSI C12.1. The accuracy level for reading the meters is 1%. The section 12.1 describes further in detail the accuracy limits, test plans, and inspection procedures for the meters.
Data Communication Intervals	15mins maximum
Data Log Record	Time stamp on each measurements
Internal Clocks	All measurement devices synchronize automatically to the base station's clock
Temperature	-20 to 50 Degrees Celsius 10%-90% Humidity
Interference	Multiple metering systems must co-exist within range without interference
Data Storage (measurement)	4GB
Data Storage (Base Station)	4TB storage
Base Station Power	Internal Power Supply

Security	128-bit Advanced Encryption System
Response To Loss Of Communication	Automatic Resumption of dropped communication
Data File Service	Base station export in SQL file
User Interface	Indicators LEDs for unit status, successful power on self-test, successful communication, strength of communication signal, correct/incorrect phase connections
Security	If wireless communication cannot be achieved securely through the network, a wired solution is favourable.
Design laboratory	An access to 3-phase AC power supplies is yet to be confirmed.
Cost	Measurement stations cost average approximately $100, while base stations are in the $50 range.
Enclosure	Bent Aluminium Sheet

Figure 2: Specifications

Architectural Design: System

The measurement station is directly connected to the source being measured; this could be the feedline or a specific section of a building. This is represented by Section 1 of Figure 3. The measurement station circuitry provides the analog-to-digital conversion and wireless communication module (Section 2, Figure 3). The information is then transmitted to a second measurement unit, so that it can act as a mesh network and transfer all data to the base station wirelessly (Section 3, Figure 3). Once the data is received by the base station(Section 4, Figure 3), it is displayed on a PC through a USB connection (Section 5, Figure 3).

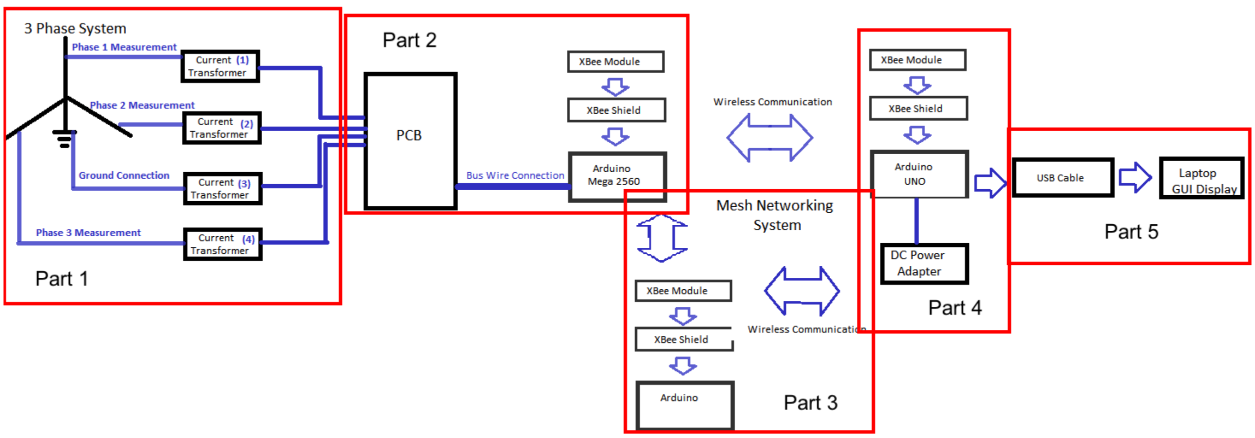


Figure 3: System Assembly Diagram

Architectural Design: Subsystem

Measurement Circuit

Our measurement device provides users with three-phase measurements of voltage, current, real power, reactive power and power factor. All these measurement values can be calculated using voltage and current waveforms. Voltage values are stepped down using transformer isolation and subsequently reduced using resistor-based voltage dividers. Current measurements are performed using current transducers (CT) by measuring the induced magnetic field around the conductor. The induced field on the secondary side is then converted to a voltage using a shunt resistor (see Figure 4). The use of CT in the design allow for users to customize the measurement according to a maximum current of their choice; after a different rating of CT is connected, a small calibration in the software takes place to make efficient use of all available data bits in the analog-to-digital conversion.

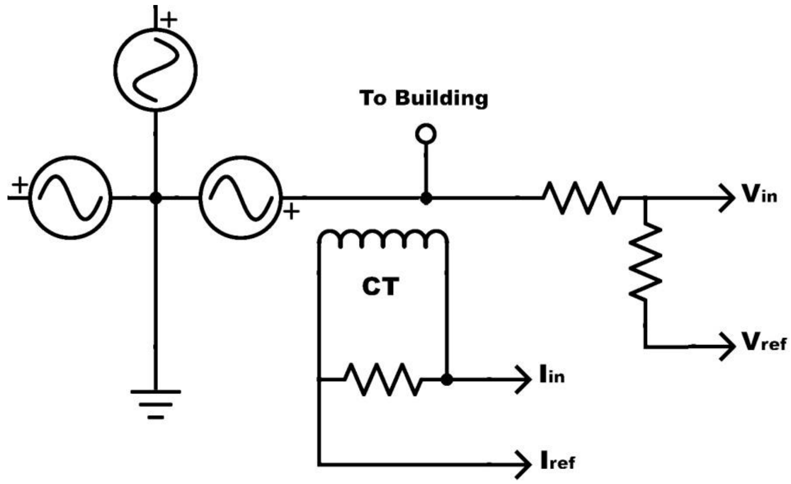


Figure 4: Voltage and Current Measurement

All our signals are given a 2.5V offset before reaching microprocessor. This offset is added using op-amp circuitry in differential configuration. The signals are then sent to the Arduino Mega's input connections to be converted to a digital signal. The power to all circuitry is obtained from one of the measured phases and total current consumption from the circuit is less than 30mA.

Microcontroller and Telecommunication

As shown in Figure 5, the input data are sampled with the 10-bit ADCs on the Arduino Mega at the left hand side. After calculations in the firmware, the calculated result including voltage, current, real power, apparent power, power factor and energy for all three phases are transmitted to the base station through the XBEE RF module. The XBee module in the base station then receives the data, and the Arduino in the base station decodes the packets and sends the data to laptop through USB cable.

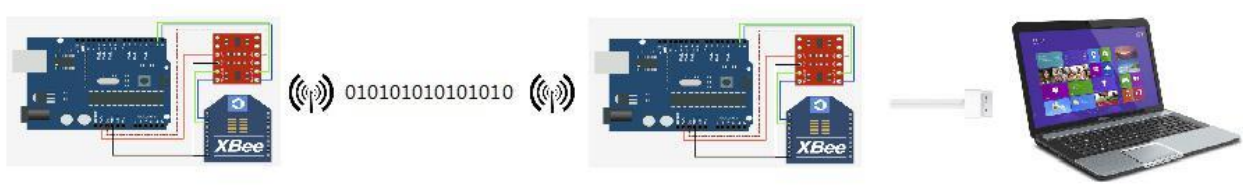


Figure 5: Microcontroller and Telecommunication Flow Chart

Measurement Point Microcontroller I/Os

The Arduino Mega’s analog pin 8 to pin 13 are the input pins that connect to the output of the measurement circuitry. The digital pin 16 to pin 21 are utilized to control the LCD screen.The main serial ports are used to communicate with the xbee module.

Firmware

In the firmware, the RMS voltage, RMS current, real power, apparent power, power factor and energy consumption are calculated with the voltage and current samples from the analog input pins. As shown in Figure 6, the calculation results are then displayed on LCD and sent to the xbee module through the serial ports. In this section, an overview of calculation, LCD display and wireless communication code is given; however, a detailed explanation is not included.

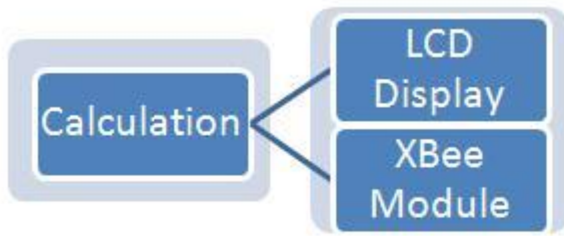


Figure 6: Firmware Flow Chart

Calculation

In the calculation section, digital filter is applied to the sample data to eliminate the 2.5V offset. After the root mean square operation with the filtered samples, the RMS voltage and current are found. The real power, apparent power, power factor and energy are then calculated upon the RMS voltage, RMS current and filtered samples.

For each calculation result, the number of samples can be set in the firmware. This parameter affects the accuracy of the calculation. The more samples, the more accurate the calculation is. However, involving too many samples for one calculation implies a longer calculation time. Using 60 cycles of samples will take at least 3 seconds for one calculation result as using 10 cycles of samples will only take 0.5 seconds. Currently, the firmware sets the number of samples to 15 cycles, so that the calculation does not take too long but still maintains accuracy.

LCD Display

The Arduino LCD display library is implemented in the firmware to control the LCD screen. The functions in the library can set the cursor position and write characters on the current position. Figure 7 demonstrates the flow chart of printing characters on LCD screen.

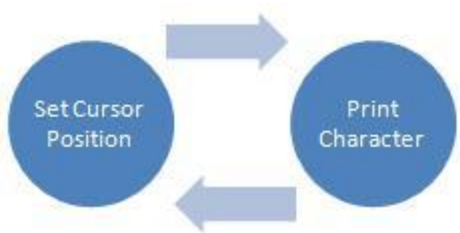


Figure 7: LCD Display Flow Chart

The scrolling function in the library is also implemented in the firmware. The LCD screen scrolls one space as the scrolling function is called once. Therefore, a “for” loop is used to make the LCD screen to scroll more than one spaces.

Wireless Communication

In the wireless communication section, the calculation data are encoded into standard transmission packets for ZigBee protocol. As shown in Figure 8, the packet is sent to the XBee module through the serial port. After the XBee module transmits the packet, an acknowledgement packet is sent back to the XBee, and is read by the Arduino serial port. By decoding the acknowledgement packet, the firmware can identify if the data packet is successfully transmitted. If the transmission is not successful, the firmware will send the data packet again until the transmission is complete.

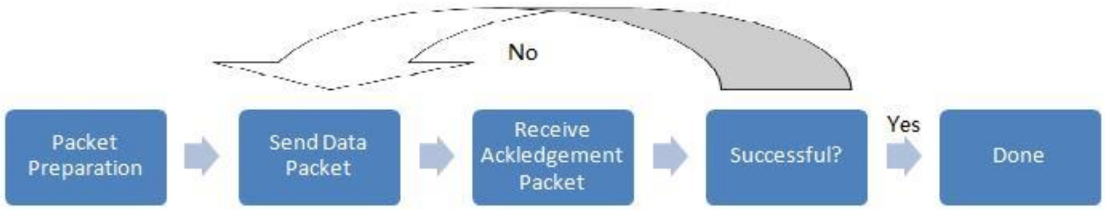


Figure 8: Wireless Communication Code Flow Chart

Mesh Network and XBee Module

Figure 9 demonstrates the small mesh network model built for this project. As the Xbee modules in the measurement points are configured as routers, the routers have two paths to communicate with the coordinator XBee, i.e. the base station. In case of the direct communication between a router and the coordinator is disrupted, the mesh network automatically establishes communication with the alternative path through the other router. As a result, the mesh network increases the transmission success rate and is able to potentially expands the transmission range by adding more XBee modules.

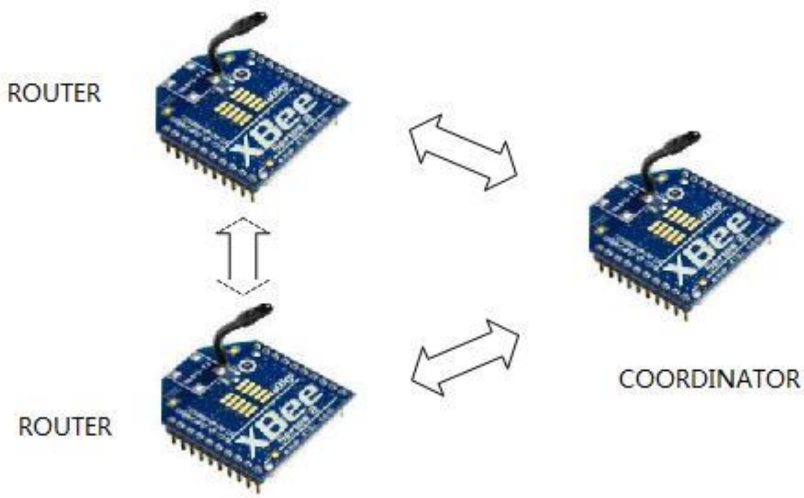


Figure 9: Mesh Network

Storage

Initially the SD card was utilized for the storage of the data. However, to automate the storage and avoid the manual task of obtaining the data from SD card, SQL Database was implemented using the Microsoft SQL Server. This gives the data storage of over 4 terabytes.The connection with SQL Database was created successfully by creating a login and enabling UDP/TCP ports for successful communication through Windows Firewall. The allocation of memory was created using the table "real storage" that can store all the three phase measurement including the timestamp of when the data is received. This storage table can be displayed on the GUI by running the SQL commands in the user input. This promotes an easier user interface by letting the user display how they wish to see the data.

Graphical User Interface (GUI)

The graphic user interface for this project was successfully implemented in Visual Basic. The purpose of the GUI is to create a user-friendly interface. One of its key features is to display the current data for the three phase system received by the base station and update the values. The Serial Port connection on the GUI enables it to connect to the arduino to receive the data wirelessly. To disconnect with the Serial Port and to terminate the serial connection the disconnect button is used.

The GUI is further integrated with SQL Server to display the stored database table. This functionality is implemented by few lines of code in Visual Basic by including the Server name and the data base used. This feature was implemented so that user can easily delete if there is wrong measurement obtained in the database by entering the identification number of the row and pressing "Delete" button. In order to start saving the data, the user can click the "Save button" which will store the data for all the three phase measurement every 5 seconds. The timestamp indicates when each data was stored. For easier identification of system functionality, there are indicators on the GUI to indicate both the successful Serial Connection and SQL Database Connection (see Figure 10).

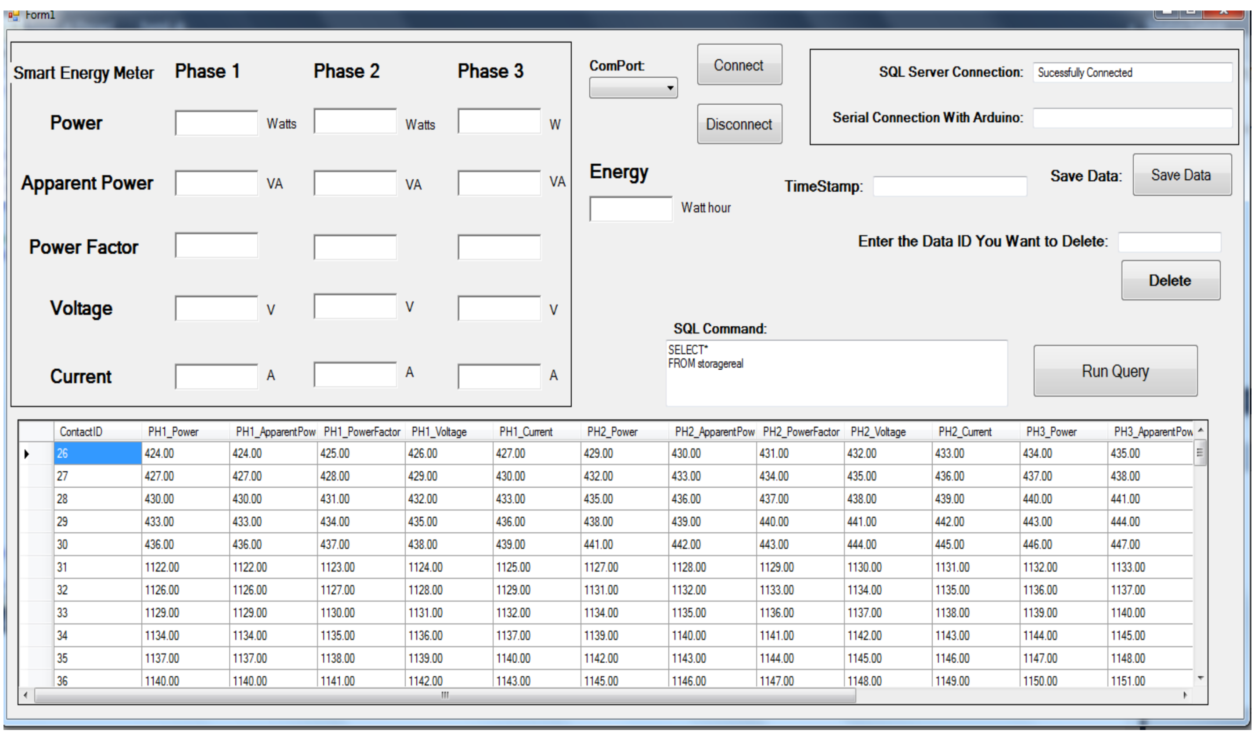


Figure 10: GUI

Formal Testing

Testing was performed using existing meters. The Electrical and Computer Engineering department labs at the University of British Columbia have a variety of multimeters and oscilloscope, which were used to compare with our measurement station's output data. We also compared the accuracy of our results with a single-phase commercial power meter. Throughout the calibration process, the measurement device was observed over several hours to ensure that measurements were consistent over time.

Risk Management

Working on a project involving wall-outlet voltages and currents presents a high risk of electrocution. In order to greatly reduce those risks, a few additions were made to the measurement device. The use of transformers on the voltage measurement branches provided galvanic isolation and eliminated the presence of high voltages on the circuit. Fuses were added to the voltage branches to limits the incoming currents to 0.5A, in case of an accidental short on the circuit. Following CSA standards for electrical units, all jack and plug connections on the voltage branches were replaced with strain relief bushings, providing a secure connection to wall voltages.

Client Hand-Off Protocol

Our client has requested a copy of the project's final report, along with a video demonstrating performance of the measurement unit. The final product is not intended to be used, but rather serve as a proof of concept regarding the software implemented.

Conclusion

The sub-meter system was successful in all components of the project. The measurement hardware circuitry simulated accurate voltage and current waveforms that were sent to the microprocessor. Wireless communication was achieved with Xbee modules between the sending and receiving stations. Mesh networking principles were demonstrated using three nodes.

The project was a valuable experience in the design, implementation, and testing of the system that involved several discrete hardware and software components. The selection of microcontroller proved to be a large drain in design time. Different designs were considered before an acceptable one was reached.

Ultimately the system accomplished it's primary goal of displaying energy information to a user in a clean and accessible way. It is an extremely useful application where the user can see how much energy an appliance consumes, and further understand the effects of leaving an appliance on or forgetting to turn the lights off when they are not required. Understanding energy consumption is the first step to reducing consumption.

Configuring the microcontroller from the user’s end would be a useful feature regarding further product development. In addition, expanding the GUI to a mobile application would make the information more accessible and user friendly. A mobile application brings the information closer to the user and therefore consumption can be monitored more diligently.

Appendices

Appendix A: Circuit Schematic

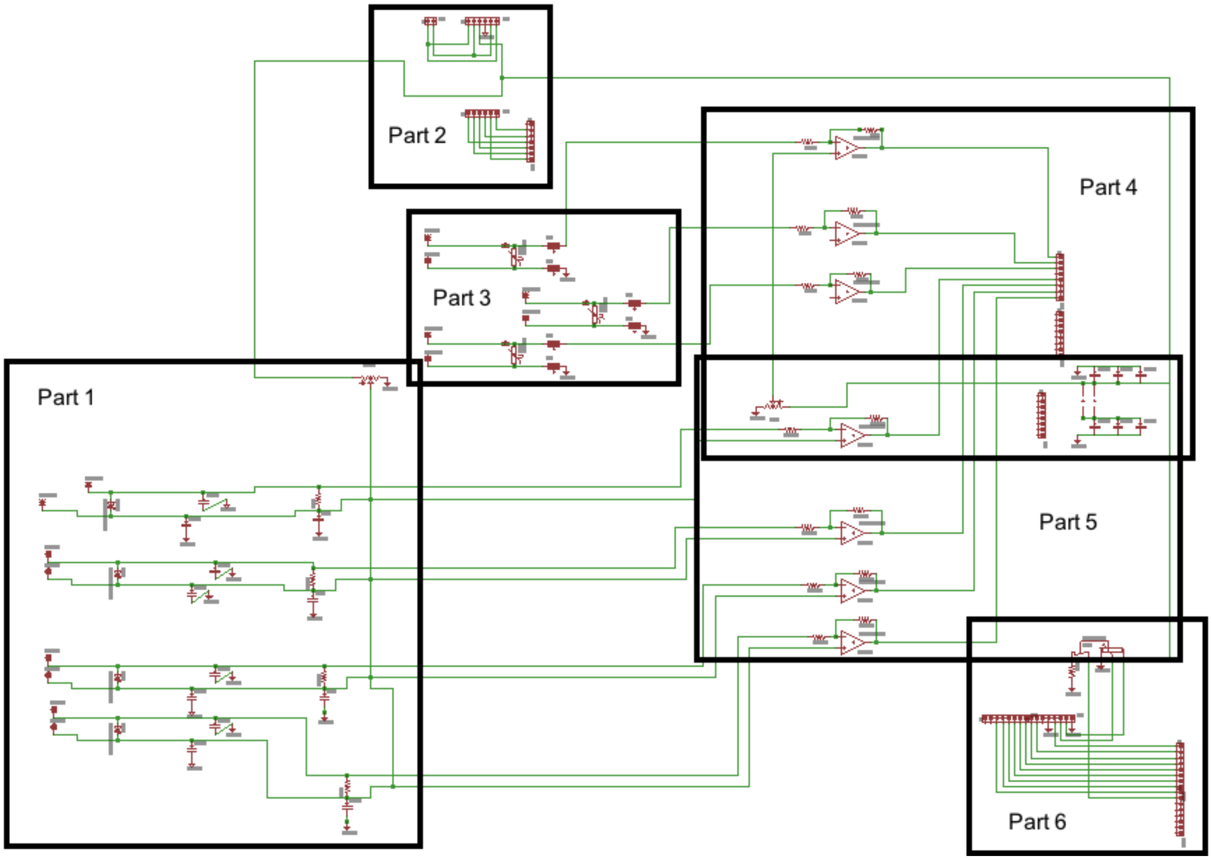


Figure 11: Circuit Schematic

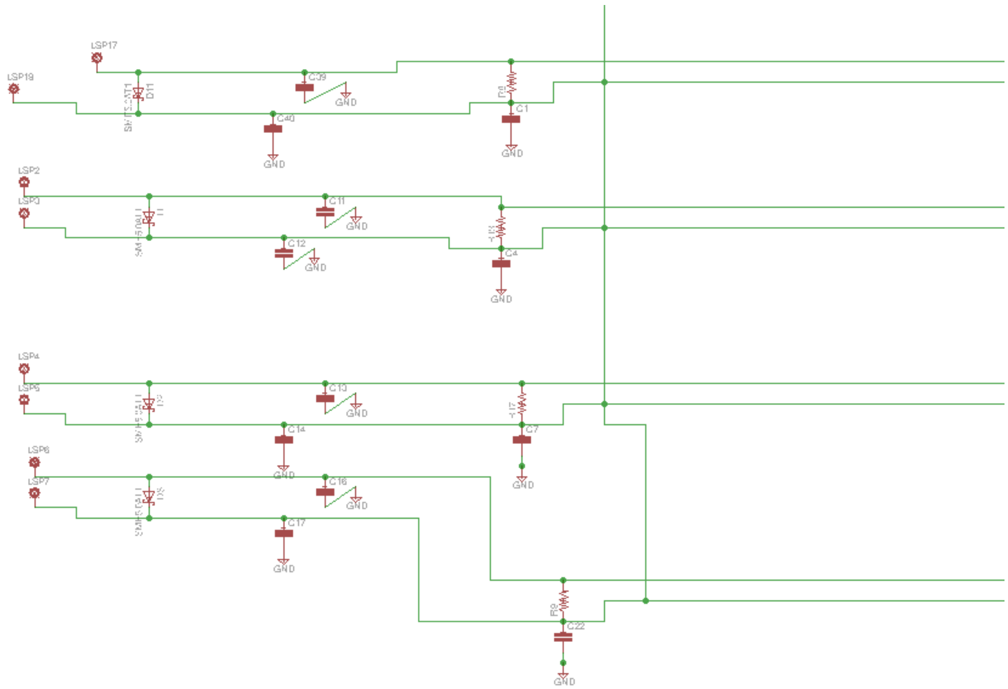


Figure 12: Circuit Schematic (Part 1)

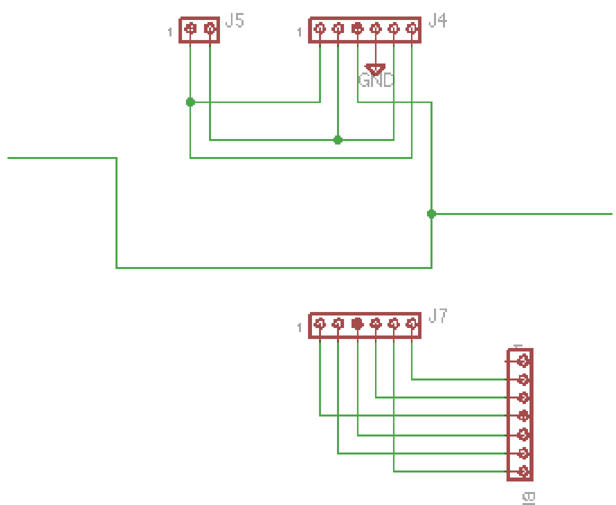


Figure 13: Circuit Schematic (Part 2)

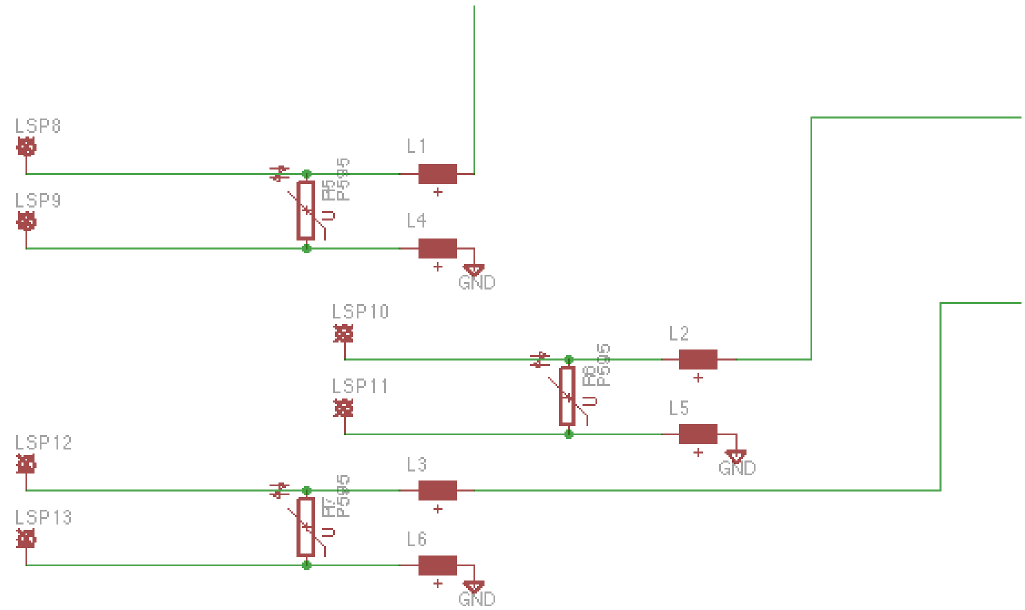


Figure 14: Circuit Schematic (Part 3)

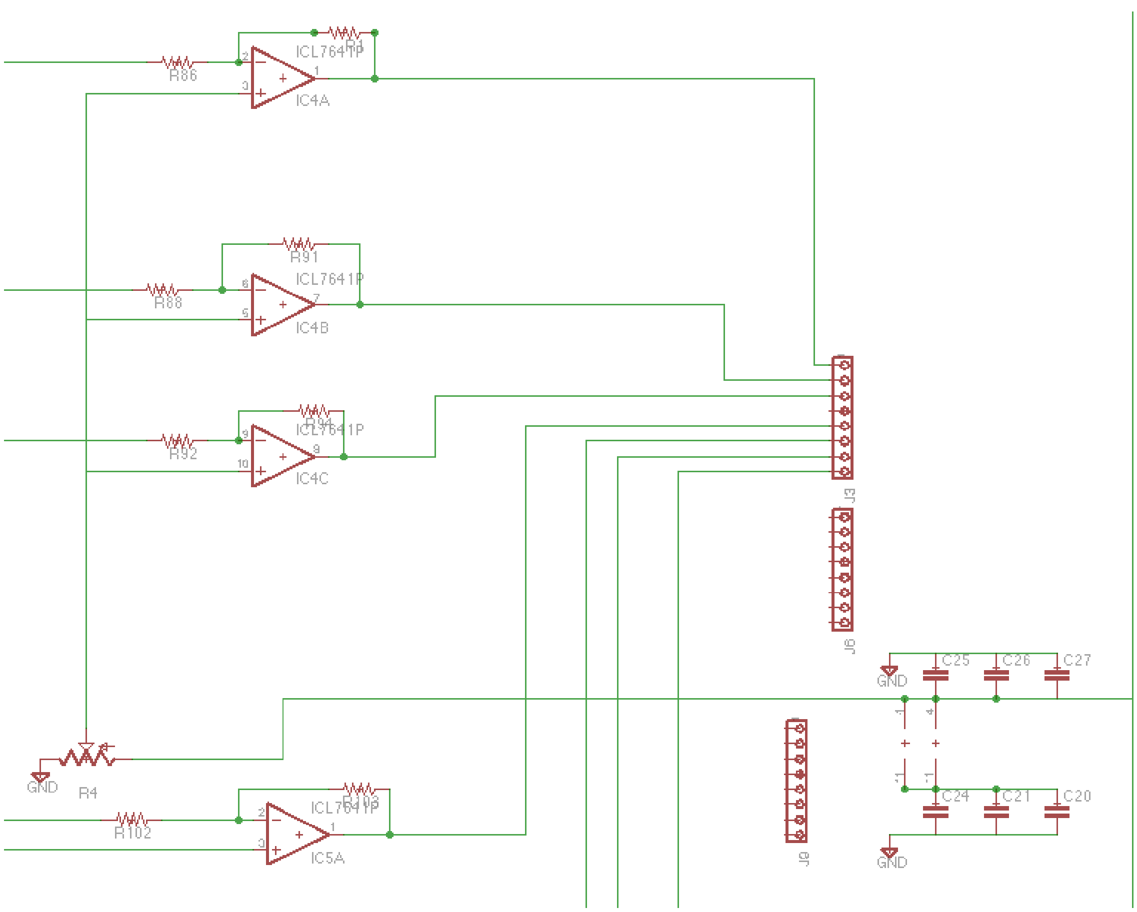


Figure 15: Circuit Schematic (Part 4)

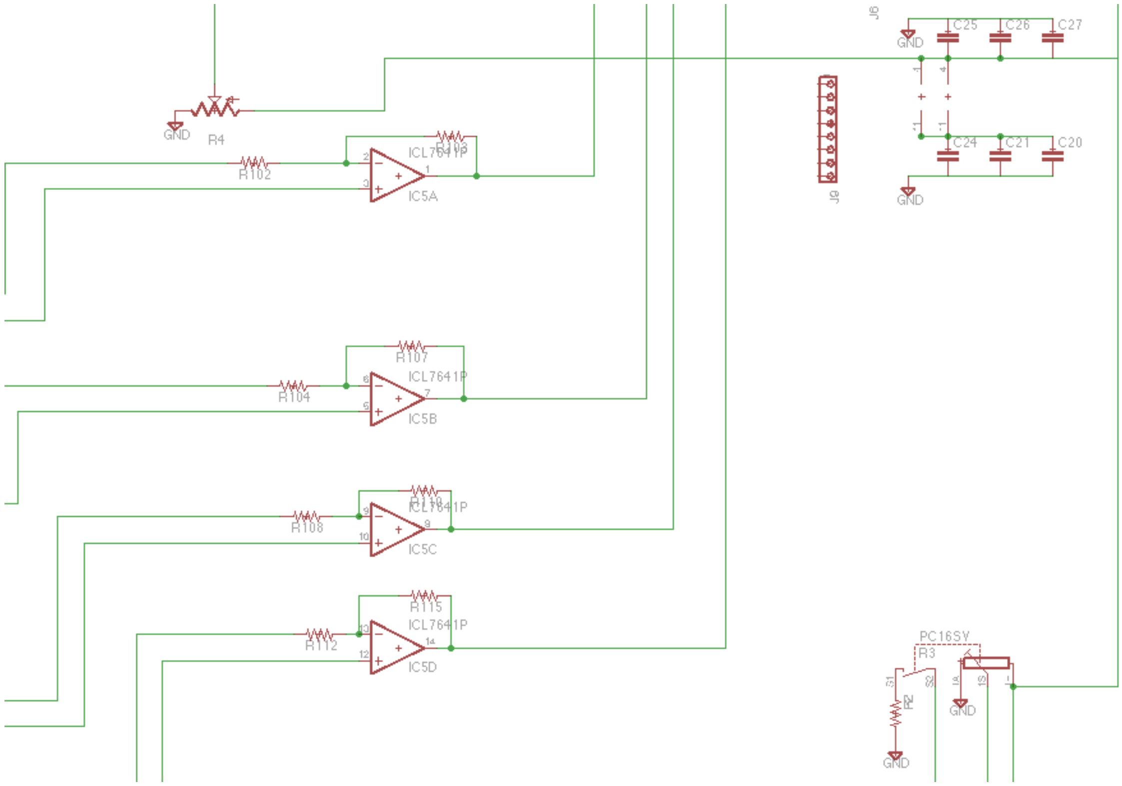


Figure 16: Circuit Schematic (Part 5)

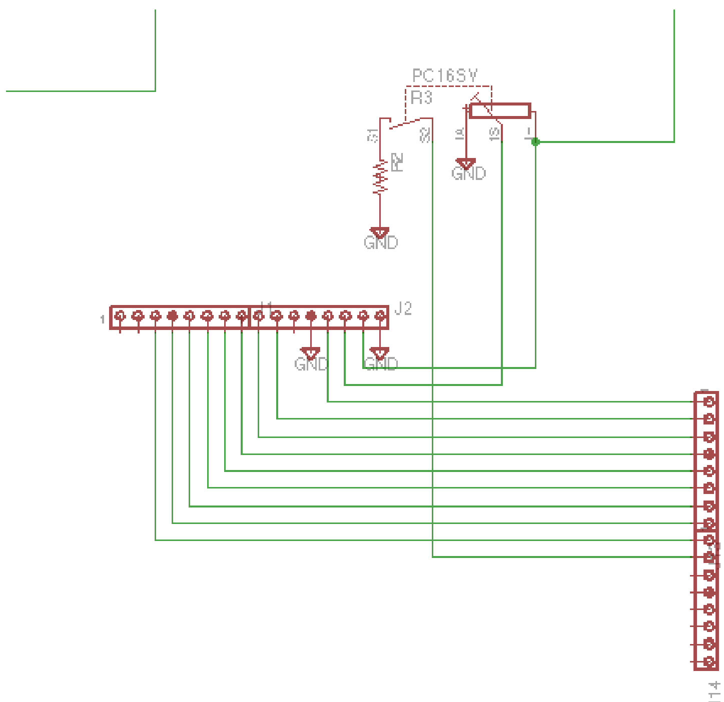


Figure 17: Circuit Schematic (Part 6)

Appendix B: Gerber Files

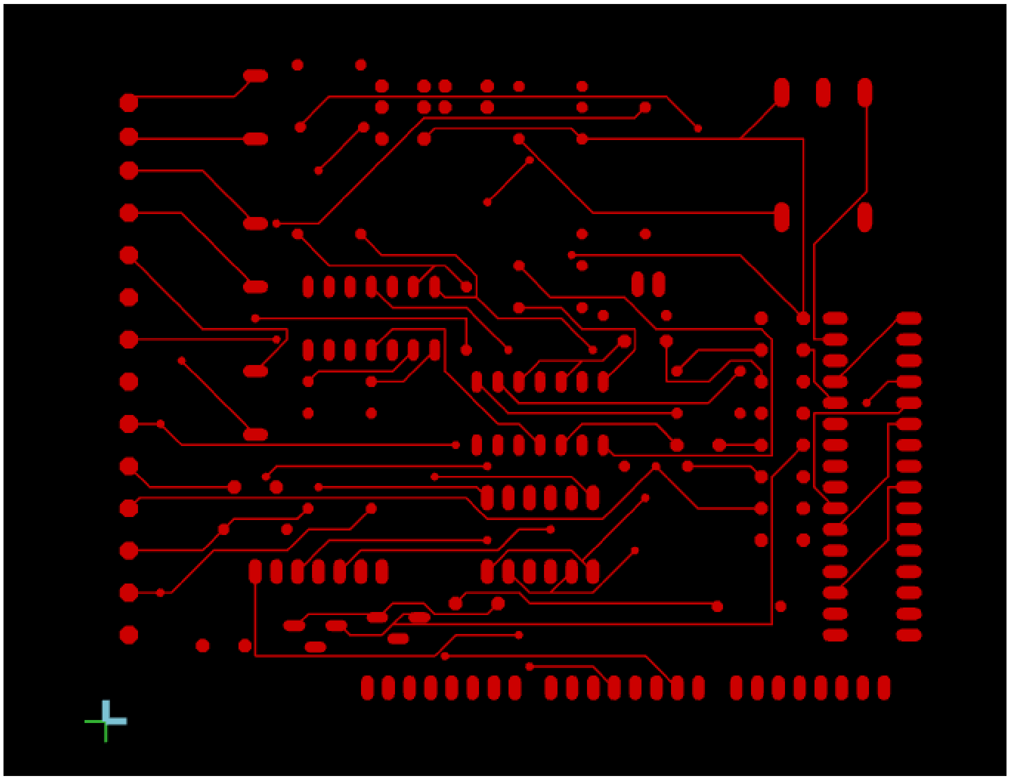


Figure 18: Bottom Solder Mask

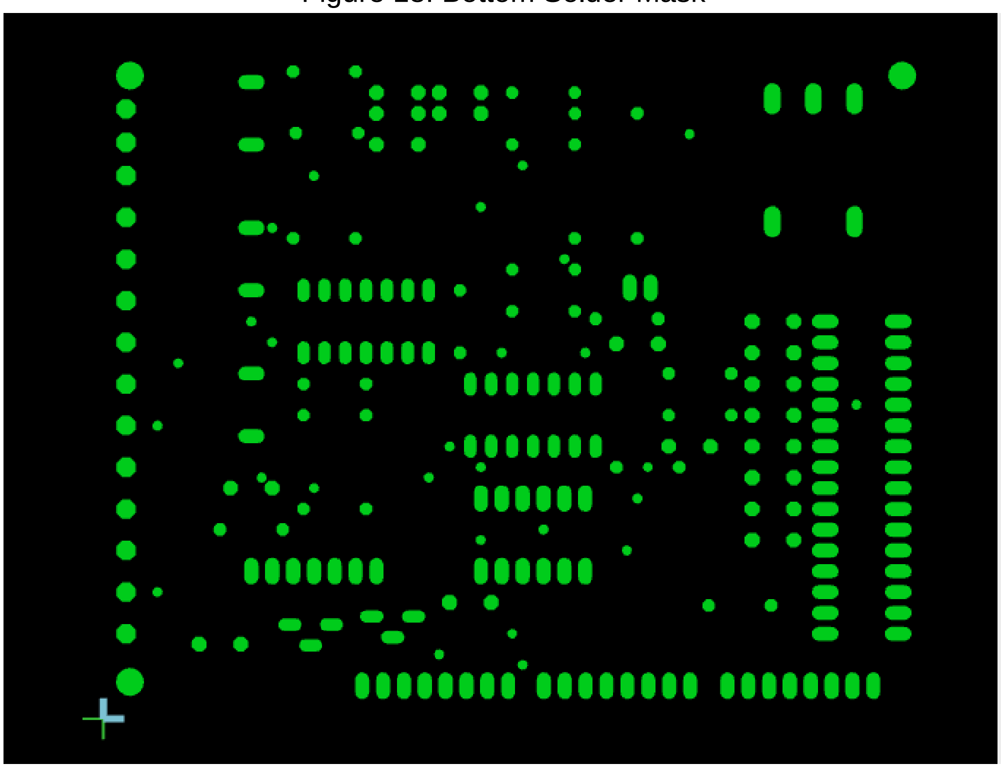


Figure 19: Bottom Holes Mask

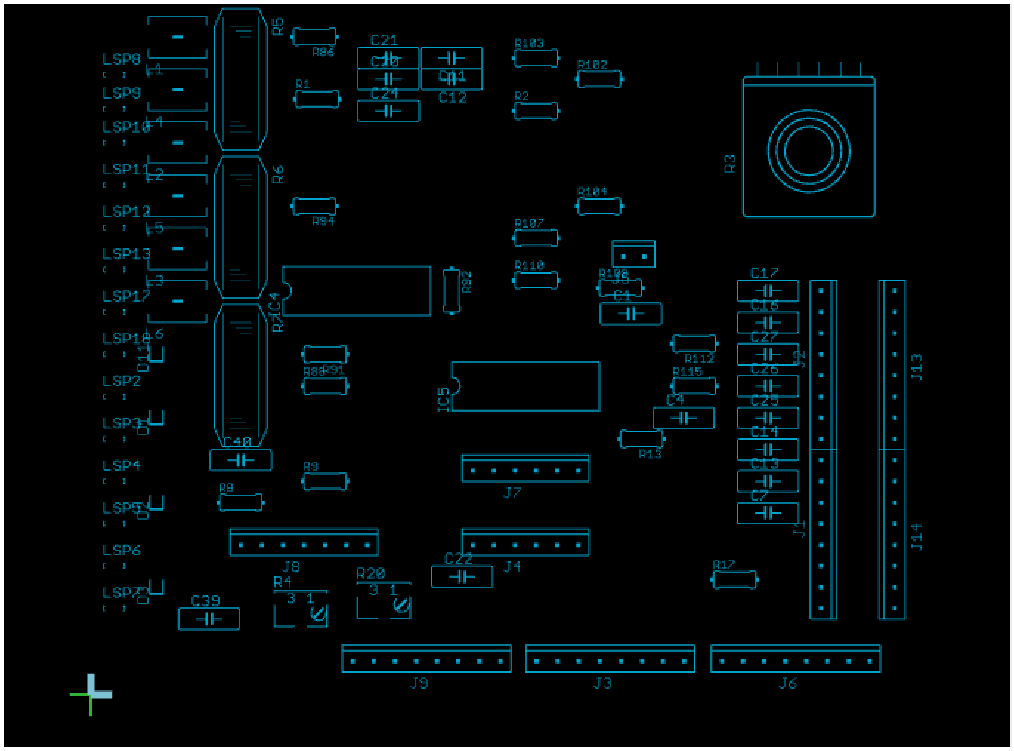


Figure 20: Top Silk Mask

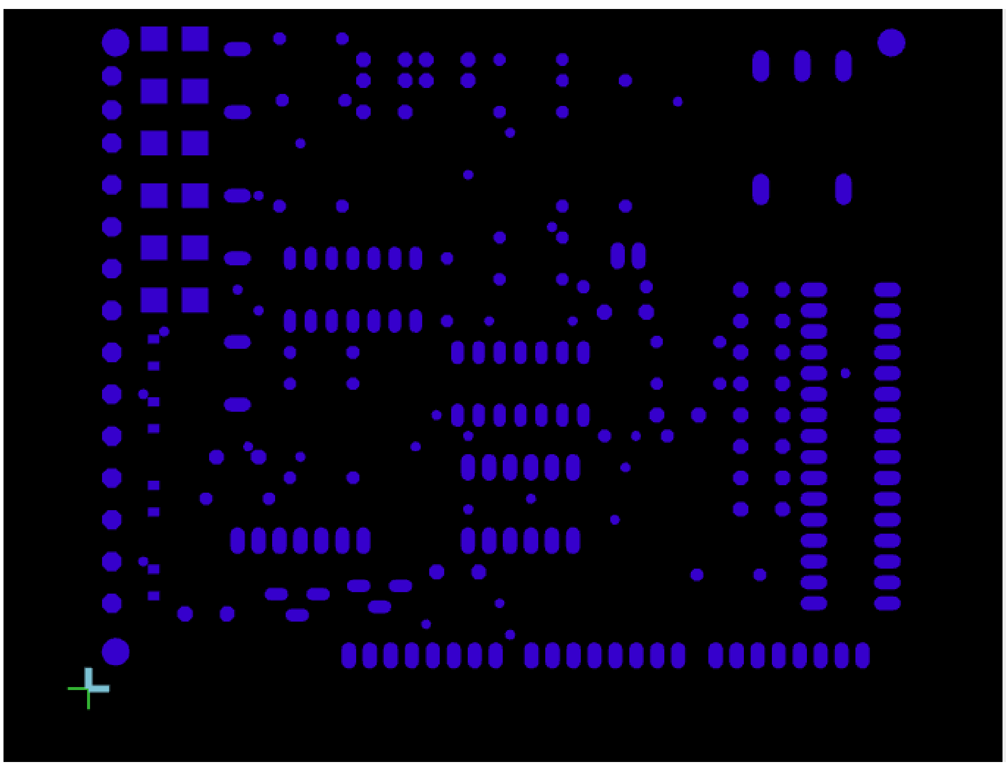


Figure 21: Top Holes Mask

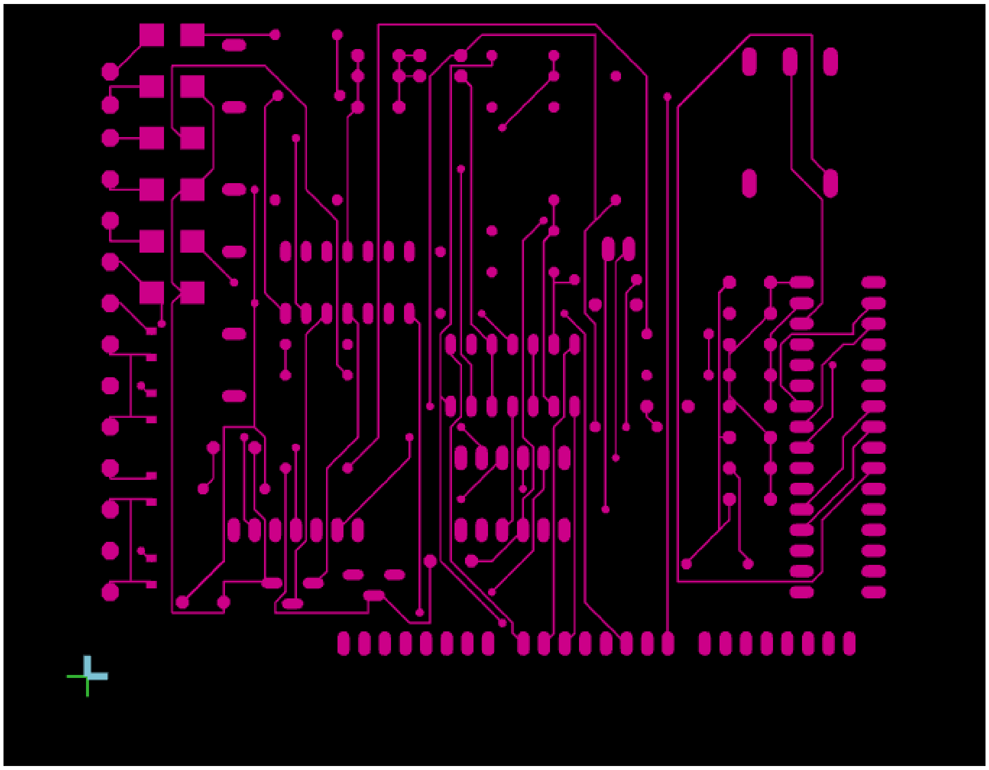


Figure 22: Top Solder Mask

Appendix C: Bill of Materials

Bill Of Materials									
Project:	#6 - Sub Meter Application Design								
Date:	April 1st, 2014								
Revision:	#3								
Prepared by:	Carlo Cossette								
Item	Label	Manufacturer	Manufacturer #	Quantity	Supplier	Supplier Part #	Description	Unit Price	Total
1									
2	TR1-TR4	LEM USA	LTSR 15-NP	4	Digi-key	398-1022-5-ND	Sensor Curr 15A	$21.44	$85.76
3	IC4, IC5	Texas Instrument	TL074CN	2	Digi-key	296-1777-5-ND	Quad Op Amp	$0.62	$1.24
4	System (Part 2, 3, 4)	Arduino	--	4	Lee's	10997	Arduino Mega	$33	$132
5	System (Part 2, 3, 4)	Sparkfun	XB24-Z7WIT-004	2	Lee's	11108	Xbee 2mW Antenna	$27	$54
6	System (Part 2, 3, 4)	SparkFun	WRL-10854	2	Lee's	11105	Arduino Xbee Shield	$26.95	$53.9
7	System (Part 2, 3, 4)	SparkFun	BOB-00544	2	Lee's	15037	MicroSD Transflash Breakout Board	$12	$24
8	J2, J14	--	--	2	Lee's	21240	1x40 Header SIP	$1	$2
9	L1-L6	Panasonic	EXC-ML20A390U	8	Digikey	P10191TR-ND	Inductor EXCML20A	$0.1	$0.83
10	D1,D2,D3,D11	Bourns	SMAJ5.0CA	4	Digikey	SMAJ5.0CABCT-ND	SMAJ5.0CA	$0.42	$1.68
11	R1, R91, R94	Stackpole Electronics	CFM14JT1K20	3	Digikey	S1.2KQTR-ND	1.2K Resistor	$0.08	$0.24
12	R102, R104, R108, R112	Stackpole Electronics	CFM14JT3K30	4	Digikey	S3.3KQCT-ND	3.3K Resistor	$0.08	$0.32
13	R86, R88, R92	Stackpole Electronics	CFM14JT11K0	3	Digikey	S11KQCT-ND	11K Resistor	$0.08	$0.24
14	R103, R107, R110, R115	Stackpole Electronics	CFM14JT51K0	4	Digikey	S51KQCT-ND	51K Resistor	$0.08	$0.32
15	R5, R6, R7	EPCOS	S20K275	3	Digikey	495-1417-ND	s20k275 varistor	$1.06	$3.18
16	R8, R9, R13, R17	Stackpole Electronics	CF14JT100R	4	Digikey	CF14JT100RTR-ND	10 ohm Resistor	$0.08	$0.48
17	R4, R20	Bourns Inc.	3296W-1-102LF	2	Digikey	3296W-102LF-ND	1K Trimpot	$2.31	$4.62
18	C1, C2, C4, C5, C7, C8, C11, C12, C13, C14, C16, C17, C39, C40	Vishay BC Components	K470J15C0GF5TL2	14	Digikey	BC1009CT-ND	47pF Ceramic Capacitor	$0.34	$4.76
19	C3, C6, C9, C10, C15, C18, C38	Vishay BC Components	K153K10X7RF5UH	7	Digikey	BC2671CT-ND	15nF Capacitor	$0.32	$2.24
20	C29, C30, C31, C32, C33, C34, C35	Panasonic	EEA-GA1H4R7B	7	Digikey	P15838CT-ND	4.7uF Capacitor (50V)	$0.35	$2.45
21	C36, C37	Vishay BC Components	K120J15C0GF5TL2	2	Digikey	BC1002CT-ND	12pF Capacitor	$0.34	$0.68
22	C19, C20, C21	Rubycon	UPW2F2R2MPD1T	1	Digikey	493-4725-1-ND	2.2uF Capacitor (315V)	$0.62	$0.62

Appendix D: Expenditure Forms

Capstone Project Purchasing Expense Claim Form

Academic Year: 2013-2014 Course Number: EECE469 Group number: Project 6

Invoice Num. [ece.ubc.ca #192590] Date 6-Apr-14

Payable To Carlo Cossette Student or Employee Num. 20555116

Mailing Address 207-6385 Hawthorn Lane

Vancouver, BC V6T1Z4

Expenses Claimed

DATE	DESCRIPTION	CATEGORY	AMOUNT	CURRENCY	RATE	SUBTOTAL
2013-Oct-11	Lee's Electronics	Development Parts	$301.17	CAD	1.0000	$301.17
2013-Oct-11	Digi-Key	Current Sensors and ICs	$159.55	CAD	1.0000	$159.55
2013-Oct-12	Futurlec	SMD Adapter Boards	$28.00	USD	1.03535	$28.99
2013-Nov-04	Digi-Key	Passive Circuit Components	$108.08	USD	1.0415	$112.56
2013-Nov-13	Digi-Key ***REFUND***	REFUND	$(91.67)	CAD	1.0000	$(91.67)
2013-Nov-16	Lee's Electronics	Launchpad board	$14.56	CAD	1.0000	$15.07
2014-Jan-08	Lee's Electronics	Arduino Mega	$67.20	CAD	1.0000	$67.20
			$-	CAD	1.0000	$-
			$-	CAD	1.0000	$-
			$-	CAD	1.0000	$-
					TOTAL	$592.88

Put Speedchart # here : LNFE

Signature of Submitter

By signing this claim form, I assert:

(1) that this is the first and only time that these expenses have been / will be claimed;

(2) that these expenses have been incurred in accordance with all applicable UBC and granting agency policies; and

(3) that I understand that the Finance Clerk may make adjustments to the amounts claimed in order to meet UBC or granting agency policies.

Name Carlo Cossette Date 2014-01-07

Signature

Capstone Project Purchasing Expense Claim Form

Academic Year: 2013-2014 Course Number: EECE469 Group number: Project 6

Invoice Num. [ece.ubc.ca #199680] Date 4-Apr-14

Payable To Carlo Cossette Student or Employee Num. 20555116

Mailing Address 207-6385 Hawthorn Lane
Vancouver, BC V6T1Z4

Expenses Claimed

DATE	DESCRIPTION	CATEGORY	AMOUNT	CURRENCY	RATE	SUBTOTAL
2013-Oct-24	Lee's Electronics (passive circuit components)	Passive Circuit Components -F	$22.90	CAD	1.0000	$22.90
2013-Oct-24	Staples (Micro SD card)	Development Parts -F	$21.49	CAD	1.0000	$21.49
2013-Oct-30	Lee's Electronics (LCD display)	Passive Circuit Components -F	$16.80	USD	1.0000	$16.80
2014-Feb-01	Lee's Electronics (Communication Devices)	Development Parts -A	$66.25	USD	1.0000	$66.25
2014-Mar-03	Lee's Electronics (Transformers)	Passive Components	$15.68	CAD	1.0000	$15.68
2014-Mar-12	Seeedstudio (China)	PCB Manufacturing	$29.76	USD	1.1135	$33.14
			$-	CAD	1.0000	$-
			$-	CAD	1.0000	$-
			$-	CAD	1.0000	$-
			$-	CAD	1.0000	$-
					TOTAL	$176.26

Put Speedchart # here : **LNFE**

Signature of Submitter

By signing this claim form, I assert:

(1) that this is the first and only time that these expenses have been / will be claimed;

(2) that these expenses have been incurred in accordance with all applicable UBC and granting agency policies; and

(3) that I understand that the Finance Clerk may make adjustments to the amounts claimed in order to meet UBC or granting agency policies.

Name Carlo Cossette Date 2014-04-04

Signature

Appendix E: Measurement Firmware

Measurement device arduino code

```
#include <EmonLib.h>
#include <XBee.h>
#include <LiquidCrystal.h>

//************************CONSTANT DEFINITION*******************************************
//general parameters
#define NUM_PAR  5  //number of parameters (V,I,P,pf)
#define TIME_DELAY 500
//measurment paramteres
#define NUM_ACROSS 100  //proportional to number of samples for 1 measurement.
#define VOLTAGE_CAL 68.43
#define CURRENT_CAL 11.28  //10.75//11.53//11.57

//communication parameters
#define NUM_TX_FRAME_BYTE 14//number of bytes in a transmit frame (without the data
bytes)
//LCD parameters
#define PIN_BUTTON2 15
#define DISPLAY_LENGTH 12
//*******************************END****************************************************

//**************************VARIABLE DECLARATION****************************************
XBee xbee= XBee();         //Xbee variable

float data[NUM_PAR*3+1];    //data
static double energy;       //energy
static int loopcount=0;
int flag=0;
unsigned long start;

void displaylcd (float var[]);
LiquidCrystal lcd(16, 17, 18, 19, 20, 21);

void setup(){
  Serial.begin(9600);
  xbee.setSerial(Serial);

  pinMode(PIN_BUTTON2, INPUT);
  lcd.begin(16, 2);
  energy = 0;
  digitalWrite(10,LOW);
```

```
  start=millis();
}

void loop(){
  //digitalWrite(10,LOW);
  // unsigned long start=millis();
  //**********************************************************************************
  EnergyMonitor emon1;        //Emon variable
  //3 phase V I P measurement
  for (int i=0;i<3;i++){
    emon1.voltage(i+8,VOLTAGE_CAL);          // Voltage: input pin, calibration,  68.43
    emon1.current(i+11,CURRENT_CAL );        // Current: input pin, calibration. 12.94

    emon1.phase(0);            //phase caliberation

    emon1.calcVI(NUM_ACROSS,2000);         // Calculate all. No.of half wavelengths
(crossings), time-out
    energy += emon1.realPower * (millis()-start)/1000.0/3600.0;//W*h

    start=millis();

    data[NUM_PAR*i]     = emon1.realPower;        //extract Real Power into variable
    data[NUM_PAR*i+1]   = emon1.apparentPower;    //extract Apparent Power into [
    data[NUM_PAR*i+2]   = emon1.powerFactor;      //extract Power Factor into Variable
    data[NUM_PAR*i+3]   = emon1.Vrms;             //extract Vrms into Variable
    data[NUM_PAR*i+4]   = emon1.Irms;             //extract Irms into Variable
  }
  data[NUM_PAR*3]  = energy;

  Serial.println("Phase\tRP\tAP\tPF\tV\tI\tenergy");
  for(int i=0;i<3;i++){
    Serial.print("Phase");
    Serial.print(i+1);
    Serial.print("\t");
    for(int j=0;j<NUM_PAR;j++){
      Serial.print(data[NUM_PAR*i+j]);
      Serial.print("\t");
    }
    if(i==2)
      Serial.print(data[NUM_PAR*3]);
    Serial.println();
  }
```

```
  displaylcd(data);//display lcd data

  XBeeAddress64 addr64 = XBeeAddress64(0x00000000,0x00000000);//Set up address
  ZBTxRequest zbTx;//packet variable
  ZBTxStatusResponse txStatus = ZBTxStatusResponse();//response status variable
  uint8_t payload[sizeof(data)];//payload

  while(flag==0){

    for(int i=0;i<sizeof(data)/sizeof(float);i++)
      for(int j=0;j<sizeof(float);j++)
        payload[i*sizeof(float)+j] =((byte *)&data[i])[j];      //transform float type to byte to store data
in payload
    zbTx=ZBTxRequest(addr64,payload,sizeof(payload));

    xbee.send(zbTx);

    xbee.readPacket(1000);
    if(xbee.getResponse().isAvailable()){
      flag=1;
      if (xbee.getResponse().getApiId() == ZB_TX_STATUS_RESPONSE){
        xbee.getResponse().getZBTxStatusResponse(txStatus);
        Serial.print("delivery status: ");
        Serial.println(txStatus.getDeliveryStatus());
      }
      else if (xbee.getResponse().isError()) {
        Serial.println();
        Serial.print("Error reading packet.  Error code: ");
        Serial.println(xbee.getResponse().getErrorCode());
      }
      delay(500);

    }
    else{
      Serial.println("No response");
      delay(5000);
    }

  }
  flag=0;
}


void displaylcd (float var [])
```

```
{
  int state_button2 = digitalRead(PIN_BUTTON2);


  lcd.setCursor(DISPLAY_LENGTH*0,0);
  lcd.print("P1=      W");
  lcd.setCursor(DISPLAY_LENGTH*0+3,0);
  lcd.print(var[0]);

  lcd.setCursor(DISPLAY_LENGTH*1,0);
  lcd.print("P2=      W");
  lcd.setCursor(DISPLAY_LENGTH*1+3,0);
  lcd.print(var[NUM_PAR*1]);

  lcd.setCursor(DISPLAY_LENGTH*2,0);
  lcd.print("P3=      W");
  lcd.setCursor(DISPLAY_LENGTH*2+3,0);
  lcd.print(var[NUM_PAR*2]);

  lcd.setCursor(DISPLAY_LENGTH*0,1);
  lcd.print("PF1=      ");
  lcd.setCursor(DISPLAY_LENGTH*0+4,1);
  lcd.print(var[2]);

  lcd.setCursor(DISPLAY_LENGTH*1,1);
  lcd.print("PF2=      ");
  lcd.setCursor(DISPLAY_LENGTH*1+4,1);
  lcd.print(var[NUM_PAR*1+2]);

  lcd.setCursor(DISPLAY_LENGTH*2,1);
  lcd.print("PF3=      ");
  lcd.setCursor(DISPLAY_LENGTH*2+4,1);
  lcd.print(var[NUM_PAR*2+2]);


  // scroll 13 positions (string length) to the left
  // to move it offscreen left:
  if(HIGH == state_button2)
  {
    for (int positionCounter = 0; positionCounter < 15; positionCounter++) {
      // scroll one position left:
      lcd.scrollDisplayLeft();
      // wait a bit:
```

```
    delay(500);
  }

  // scroll 29 positions (string length + display length) to the right
  // to move it offscreen right:
  for (int positionCounter = 0; positionCounter < 20; positionCounter++) {
    // scroll one position right:
    lcd.scrollDisplayRight();
    // wait a bit:
    delay(500);
  }
 }


}
```

Base Station Arduino Code

```
#include <XBee.h>

float data[16];
char temp[sizeof(float)];
XBee xbee = XBee();
XBeeResponse response = XBeeResponse();
// create reusable response objects for responses we expect to handle
ZBRxResponse rx = ZBRxResponse();
ModemStatusResponse msr = ModemStatusResponse();

void setup()
{
 //This intinializes the serial port
 Serial.begin(9600);
 xbee.setSerial(Serial);

}

void loop()

{

  xbee.readPacket();

  if (xbee.getResponse().isAvailable()) {
    // got something
```

```
 //Serial.print("xbee.getResponse().isAvailable()");

if (xbee.getResponse().getApiId() == ZB_RX_RESPONSE) {
  // got a zb rx packet
 // Serial.print("ZB_RX_RESPONSE");
  // now fill our zb rx class
  xbee.getResponse().getZBRxResponse(rx);

  /*if (rx.getOption() == ZB_PACKET_ACKNOWLEDGED) {
      // the sender got an ACK
      flashLed(statusLed, 10, 10);
  } else {
      // we got it (obviously) but sender didn't get an ACK
      flashLed(errorLed, 2, 20);
  }*/
  // set dataLed PWM to value of the first byte in the data
  //analogWrite(dataLed, rx.getData(0));
} else if (xbee.getResponse().getApiId() == MODEM_STATUS_RESPONSE) {
  xbee.getResponse().getModemStatusResponse(msr);

  // the local XBee sends this response on certain events, like association/dissociation

  /*if (msr.getStatus() == ASSOCIATED) {
    // yay this is great.  flash led
    flashLed(statusLed, 10, 10);
  } else if (msr.getStatus() == DISASSOCIATED) {
    // this is awful.. flash led to show our discontent
    flashLed(errorLed, 10, 10);
  } else {
    // another status
    flashLed(statusLed, 5, 10);
  }*/
} else {
      // not something we were expecting
  //flashLed(errorLed, 1, 25);
};
for(int i=0;i<rx.getDataLength()/sizeof(float);i++){
  for(int j=0;j<sizeof(float);j++){
    temp[j]=rx.getData(i*sizeof(float)+j);
  }
  data[i]=*((float*)temp);
  Serial.print(data[i]);
  Serial.print(",");
}
```

```
      Serial.println();
    } else if (xbee.getResponse().isError()) {
      Serial.print("Error reading packet.  Error code: ");
      Serial.println(xbee.getResponse().getErrorCode());
    }


   //sd(data);
  // displaylcd(data);


 //Serial.println(data);
  }
```

Appendix F: User Interface Code (GUI)

```
Imports System.IO.Ports
Imports System.IO
Imports System.Threading

Public Class Form
```

```
    Dim myPort As Array
    Delegate Sub myMethodDelegate(ByVal [text] As String)
    Dim myDelegate As New myMethodDelegate(AddressOf ProcessReading)
    Dim SQL As New SQLControl
    Private Sub Form1_Load(sender As System.Object, e As System.EventArgs) Handles MyBase.Load
        'has a connection then return sucessfully connected

        myPort = IO.Ports.SerialPort.GetPortNames()
        ComboBox1.Items.AddRange(myPort)
        If SQL.HasConnection = True Then
            SQLServerCon.Text = " Sucessfully Connected"
        Else
            SQLServerCon.Text = " Connection Failed"

        End If
    End Sub

Private Sub cmdQuery_Click(sender As System.Object, e As System.EventArgs) Handles cmdQuery.Click
        If txtQuery.Text <> "" Then
            If SQL.HasConnection = True Then
                'run query for what ever is on the textbox'
                SQL.RunQuery(txtQuery.Text)

                'At least there is one table in our damn data set
                If SQL.SQLDataset.Tables.Count > 0 Then
                    DGVData.DataSource = SQL.SQLDataset.Tables(0)

                End If
            End If
        End If
    End Sub
    Private Sub SerialPort1_DataReceived(ByVal sender As System.Object, ByVal e As
System.IO.Ports.SerialDataReceivedEventArgs) Handles SerialPort1.DataReceived
        Dim str As String = SerialPort1.ReadLine()
        Invoke(myDelegate, str)
    End Sub


    Private Sub Connect_Click(sender As System.Object, e As System.EventArgs) Handles Connect.Click
        Try
            SerialPort1.BaudRate = "9600"
            SerialPort1.PortName = ComboBox1.Text
```

```
            'SerialPort1.PortName = "COM4"
            SerialPort1.Parity = Parity.None
            SerialPort1.StopBits = StopBits.One
            SerialPort1.DataBits = 8
            SerialPort1.Handshake = Handshake.None
            SerialPort1.Open()
            If SerialPort1.IsOpen Then
                SerialCon.Text = "Connection Sucessful"
                Connect.Visible = False
                Disconnect.Visible = True
            Else
                SerialCon.Text = "Connection Failed"
            End If

        Catch
            SerialPort1.Close()
        End Try
    End Sub

    Private Sub Disconnect_Click(sender As System.Object, e As System.EventArgs) Handles
Disconnect.Click
        Try
            SerialPort1.Close()
            Connect.Visible = True
            Disconnect.Visible = False
            Exit Sub
        Catch
            MessageBox.Show("Some kind of problem.")
        End Try
    End Sub

    Sub ProcessReading(ByVal input As String)
        On Error Resume Next
        P1_Power.Refresh()
        P1_I.Refresh()
        P1_V.Refresh()
        P1_PF.Refresh()
        P1_AP.Refresh()

        P3_I.Refresh()
        P3_V.Refresh()
        P3_PF.Refresh()
```

```
        P3_AP.Refresh()
        P3_Power.Refresh()
        P2_I.Refresh()
        P2_V.Refresh()
        P2_PF.Refresh()
        P2_AP.Refresh()
        P2_Power.Refresh()
        Energy.Refresh()


        Dim s() As String = Split(input, ",")
        P1_Power.Text = s(0)
        P1_AP.Text = s(1)
        P1_PF.Text = s(2)
        P1_V.Text = s(3)
        P1_I.Text = s(4)
        Energy.Text = s(15)
        P2_Power.Text = s(5)
        P2_AP.Text = s(6)
        P2_PF.Text = s(7)
        P2_V.Text = s(8)
        P2_I.Text = s(9)
        P3_Power.Text = s(10)
        P3_AP.Text = s(11)
        P3_PF.Text = s(12)
        P3_V.Text = s(13)
        P3_I.Text = s(14)


    End Sub



    Private Sub cmdSave_Click(sender As System.Object, e As System.EventArgs) Handles cmdSave.Click
        Dim index As Integer = 0
        Do While index <= 10
            Dim tm As System.DateTime
            tm = Now
            TimeStamp_txt.Text = tm
            SQL.AddMember(P1_Power.Text, P1_AP.Text, P1_PF.Text, P1_V.Text, P1_I.Text, P2_Power.Text,
P2_AP.Text, P2_PF.Text, P2_V.Text, P2_I.Text, P3_Power.Text, P3_AP.Text, P3_PF.Text, P3_V.Text,
P3_I.Text, Energy.Text, TimeStamp_txt.Text)
```

```
            SQL.Delay(5)
            index += 1
        Loop
    End Sub

    Private Sub cmdDeleteUser_Click(sender As System.Object, e As System.EventArgs) Handles
cmdDeleteUser.Click
        If txtDeleteUser.Text <> "" Then 'if blank
            If MsgBox("Do you really wish to delete" & txtDeleteUser.Text & "?", MsgBoxStyle.YesNo) =
MsgBoxResult.Yes Then
                SQL.DataUpdate("DELETE FROM storagereal WHERE ContactID= '" & txtDeleteUser.Text & "' ")
            End If
        End If
    End Sub


End Class
```

Appendix G: SQL Control Code (SubCode all the function here called by the main code)

```
Imports System.Data.Sql
Imports System.Data.SqlClient
```

```
Public Class SQLControl
    Public SQLCon As New SqlConnection With {.ConnectionString =
"Server=(local);Database=SUBMETER;User=FaizaCool;Pwd=************;"}
    Public SqlCmd As SqlCommand
    Public SQLDA As New SqlDataAdapter
    Public SQLDataset As New DataSet

    'boolean expression to see if it sucessfully opens and closes'
    Public Function HasConnection() As Boolean
        Try
            SQLCon.Open()

            'if it is sucessful lets close that connection'
            SQLCon.Close()

            'if open and close returns successfully return true'
            Return True

        Catch ex As Exception
            'error box if the connection fails
            MsgBox(ex.Message)

            'otherwise return false
            Return False
        End Try

    End Function

    Public Sub RunQuery(Query As String)
        Try
            SQLCon.Open()
            'new instance of sql command to call upon the query'
            SqlCmd = New SqlCommand(Query, SQLCon)

            'Load Sql records for the data grid'
            'set our data adapter to refernce our sql commands'
            SQLDA = New SqlDataAdapter(SqlCmd)

            'creates a new dataset
            SQLDataset = New DataSet

            'fill our data set with whatever comes out of the sql data adapter
```

```
            'fills that dataset with data
            'pipe this into our data grid
            SQLDA.Fill(SQLDataset)

            'if it is sucessful lets close that connection'
            SQLCon.Close()

        Catch ex As Exception
            MsgBox(ex.Message)

            'needs to kow if the above try has any errors at all'
            If SQLCon.State = ConnectionState.Open Then
                SQLCon.Close()
            End If

        End Try
    End Sub

    Public Sub AddMember(PH1_Power As String, PH1_ApparentPower As String, PH1_PowerFactor As
String, PH1_Voltage As String, PH1_Current As String, PH2_Power As String, PH2_ApparentPower As
String, PH2_PowerFactor As String, PH2_Voltage As String, PH2_Current As String, PH3_Power As String,
PH3_ApparentPower As String, PH3_PowerFactor As String, PH3_Voltage As String, PH3_Current As
String, Energy As String, Time_Stamp As String)
        Try


            Dim strInsert As String = "INSERT INTO storagereal (PH1_Power, PH1_ApparentPower,
PH1_PowerFactor, PH1_Voltage,PH1_Current,
PH2_Power,PH2_ApparentPower,PH2_PowerFactor,PH2_Voltage,PH2_Current,PH3_Power,PH3_Appare
ntPower, PH3_PowerFactor, PH3_Voltage,PH3_Current, Energy,Time_Stamp) " & _
                                "VALUES ( " & _
                                "'" & PH1_Power & "'," & _
                                "'" & PH1_ApparentPower & "'," & _
                                "'" & PH1_PowerFactor & "'," & _
                                "'" & PH1_Voltage & "'," & _
                                "'" & PH1_Current & "'," & _
                                "'" & PH2_Power & "'," & _
                                "'" & PH2_ApparentPower & "'," & _
                                "'" & PH2_PowerFactor & "'," & _
                                "'" & PH2_Voltage & "'," & _
                                "'" & PH2_Current & "'," & _
                                "'" & PH3_Power & "'," & _
```

```
                        "'" & PH3_ApparentPower & "'," & _
                        "'" & PH3_PowerFactor & "'," & _
                        "'" & PH3_Voltage & "'," & _
                        "'" & PH3_Current & "'," & _
                        "'" & Energy & "'," & _
                        "'" & Time_Stamp & "') "


      ' MsgBox(strInsert)

      SQLCon.Open()
      SqlCmd = New SqlCommand(strInsert, SQLCon)

      SqlCmd.ExecuteNonQuery() 'for insert update and delete

      SQLCon.Close()


    Catch ex As Exception
      MsgBox(ex.Message)

    End Try

  End Sub

  Public Sub Delay(ByVal dblSecs As Double)

    Const OneSec As Double = 1.0# / (1440.0# * 60.0#)
    Dim dblWaitTil As Date
    Now.AddSeconds(OneSec)
    dblWaitTil = Now.AddSeconds(OneSec).AddSeconds(dblSecs)
    Do Until Now > dblWaitTil
      Application.DoEvents() ' Allow windows messages to be processed
    Loop

  End Sub

  Public Function DataUpdate(Command As String) As Integer
    Try
      SQLCon.Open()
      SqlCmd = New SqlCommand(Command, SQLCon) ' type any command query
```

```
            'how many records are gonna be changed it will say that below
            Dim ChangeCount As Integer = SqlCmd.ExecuteNonQuery() 'execute nonquery but also produce a
value


            SQLCon.Close()

            Return ChangeCount
        Catch ex As Exception
            MsgBox(ex.Message)

        End Try

        Return 0

    End Function

End Class
```

Appendix H: Storage Code

The storage code to create the database in a folder named "SUBMETER" and a table called “realstorage”.

```
/////////////////////////////////////////////////////////////////
CREATE DATABASE SUBMETER
GO

USE SUBMETER
GO

CREATE TABLE storagereal
(ContactID INT IDENTITY PRIMARY KEY,
PH1_Power VARCHAR(250) NULL,
PH1_ApparentPower VARCHAR(250) NULL,
PH1_PowerFactor VARCHAR(250) NULL,
PH1_Voltage VARCHAR(250) NULL,
PH1_Current VARCHAR(250)NULL,
PH2_Power VARCHAR(250) NULL,
PH2_ApparentPower VARCHAR(250) NULL,
PH2_PowerFactor VARCHAR(250) NULL,
PH2_Voltage VARCHAR(250) NULL,
PH2_Current VARCHAR(250)NULL,
PH3_Power VARCHAR(250) NULL,
PH3_ApparentPower VARCHAR(250) NULL,
PH3_PowerFactor VARCHAR(250) NULL,
PH3_Voltage VARCHAR(250) NULL,
PH3_Current VARCHAR(250)NULL,
Energy VARCHAR(250)NULL)

select* from storagereal

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

Appendix I: Enclosure Drawings

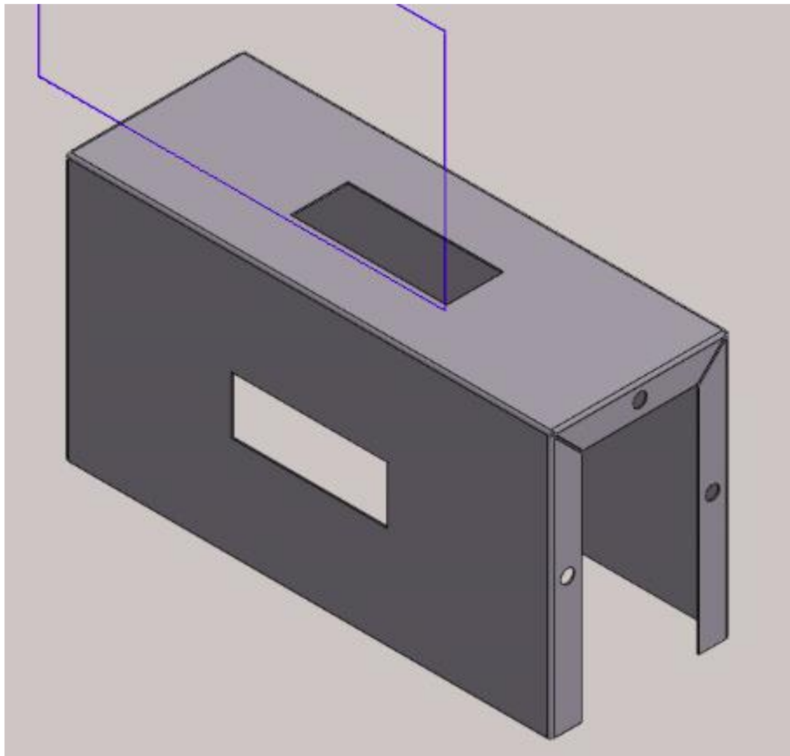


Figure 23: Enclosure Top

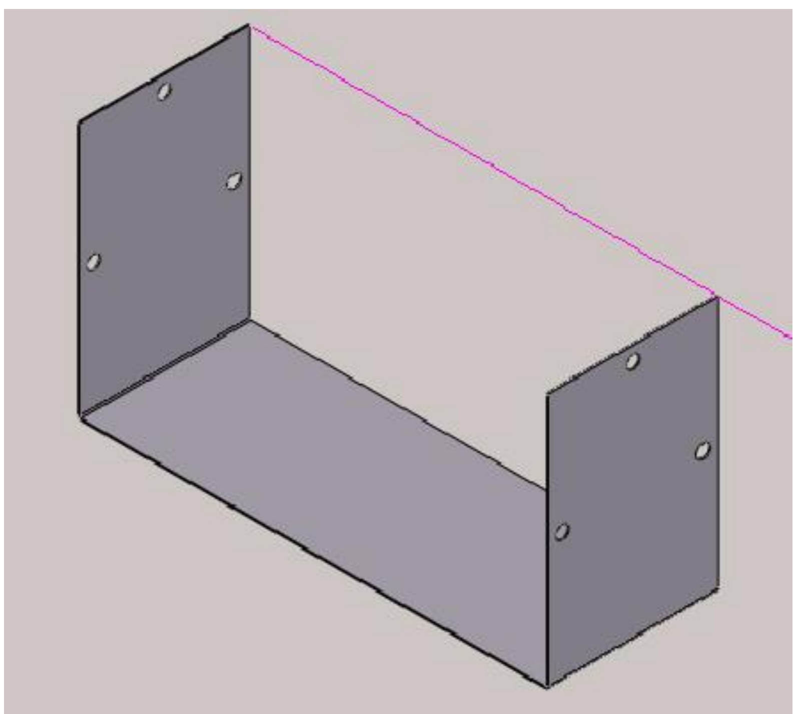
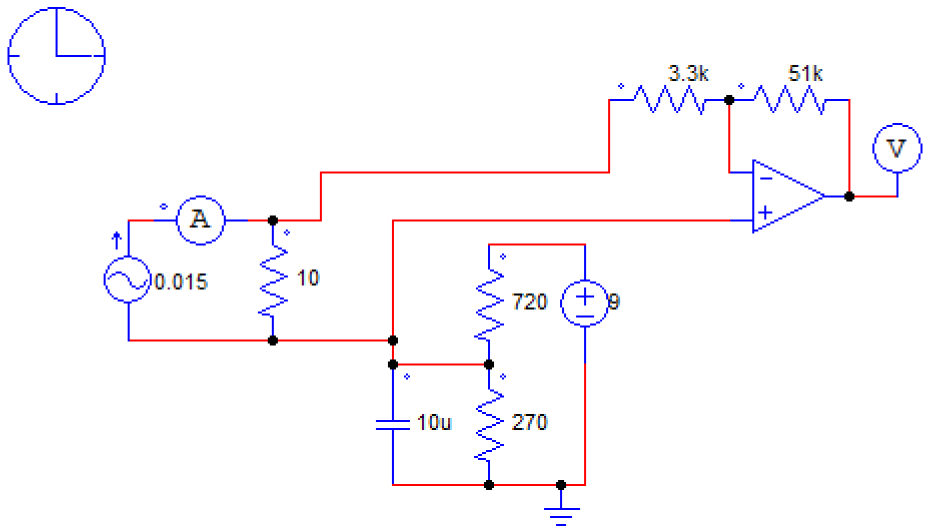
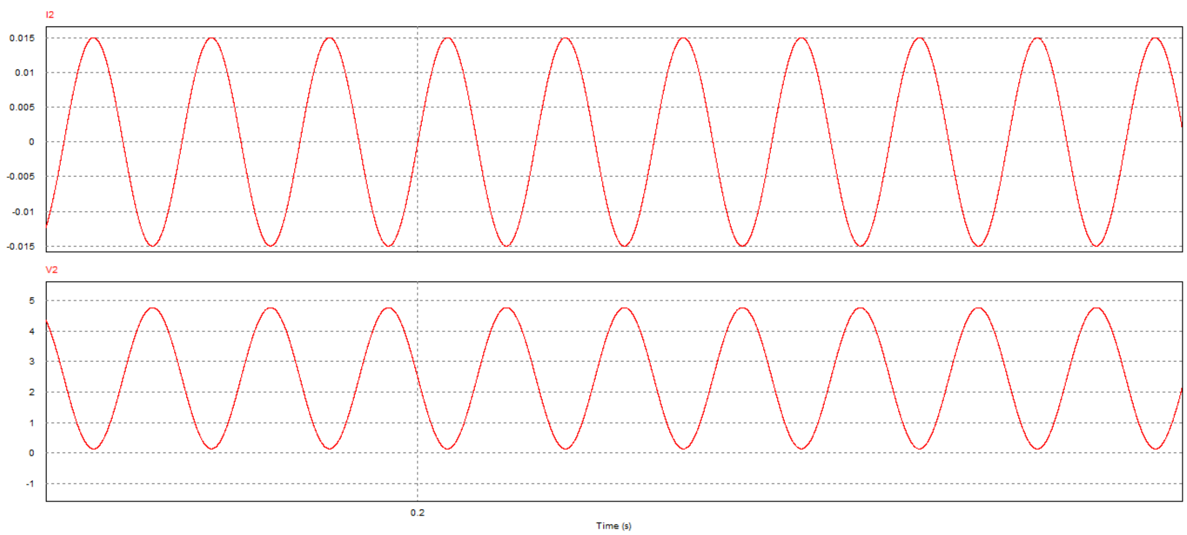


Figure 24: Enclosure Bottom

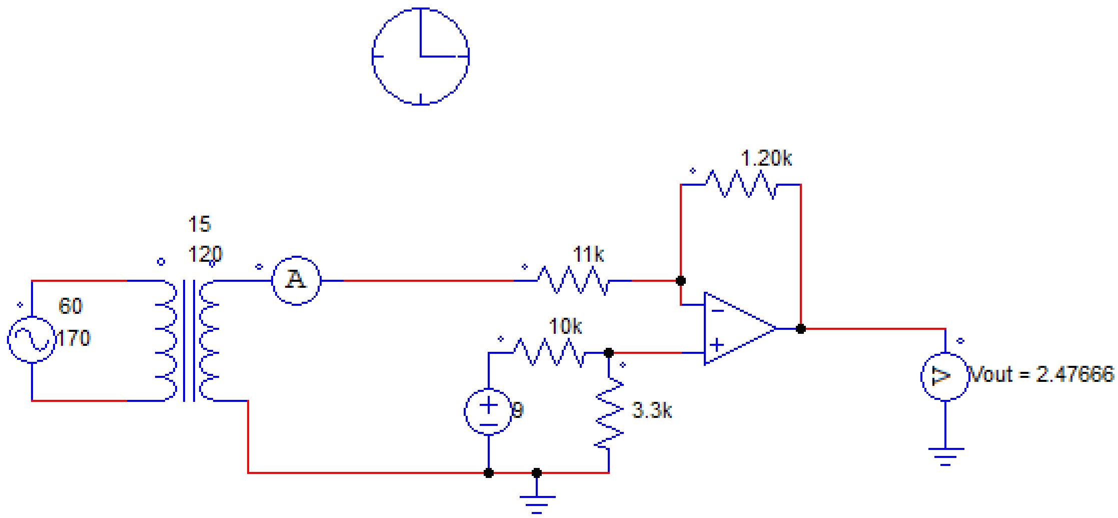
Appendix J: Hardware Simulations

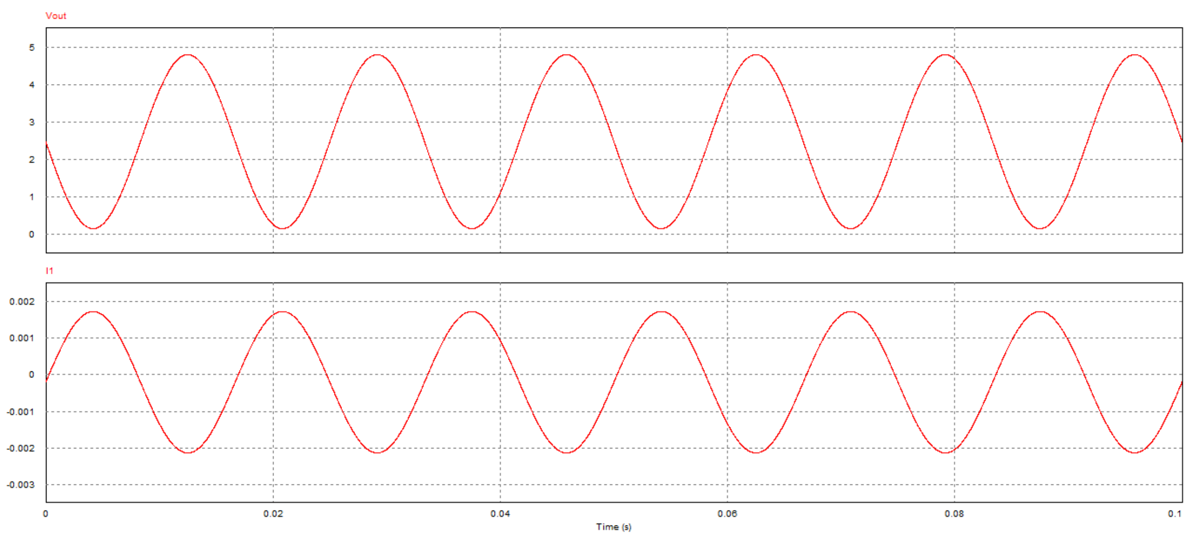
Current Measurement Simulation:



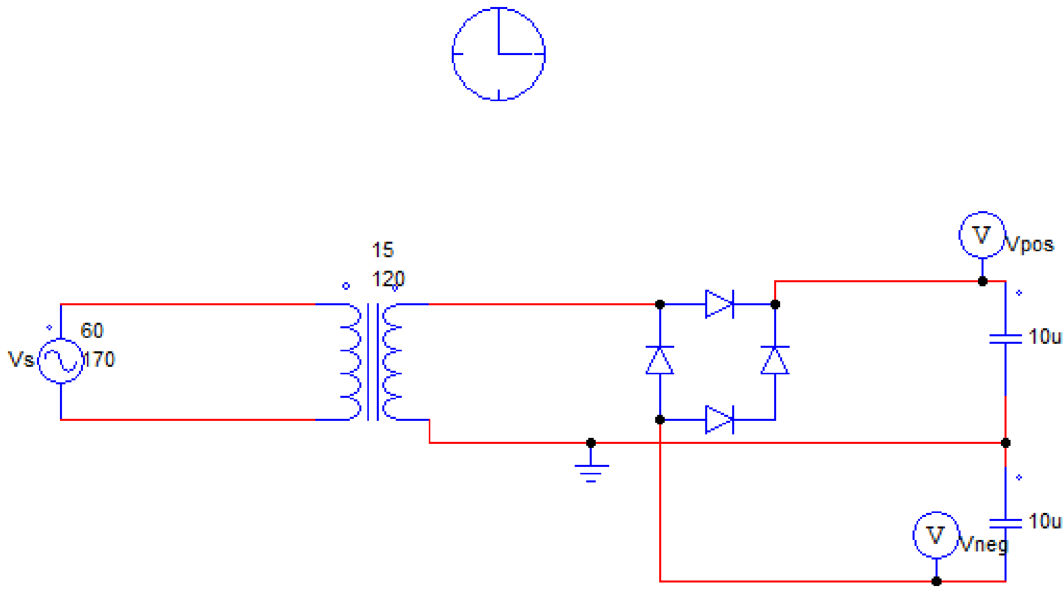


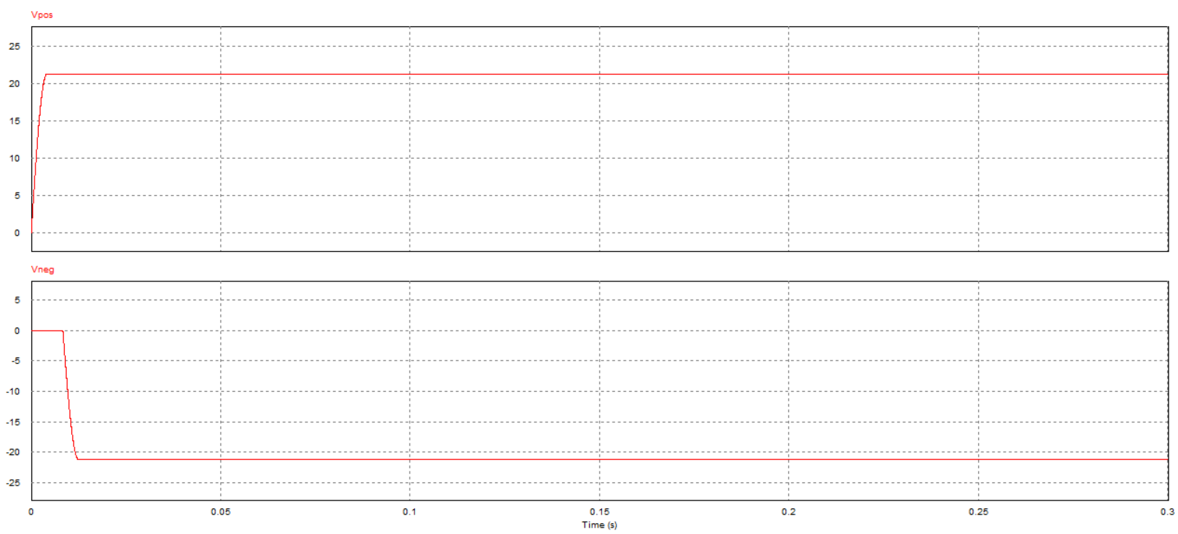
Voltage Measurement Simulation:





Power Supply Simulation:





Final report

The final report should be stand alone, self-consistent (i.e., not rely too much on past intermediate deliverables). In general, the reader should find:

- the **problem**: what it is that you were trying to solve, build, achieve, demonstrate (this may have evolved since you did the proposal in October)
- the proposed **solution**: your most recent design, with some justifications of the choices made, for example linking your design choices to the requirements or to the constraints
- your **validation** of the solution: in most cases, this will take the form of a prototype, and testing done on this prototype. Certain projects may have other form of validation (user survey, for example)
- other pertinent information such as budget (your team expenses); all requests for reimbursements must have been filed.
- Your final report may have other sections as appropriate for your project kind or domain (see your instructor).

This final report must be submitted both in Connect (to have a stamp mark), and in print by dropping a copy in the mailbox in McLeod.

The report will be graded primarily by your instructor and your TA for technical content, but also for form, style, language by a communication TA and instructor. A marking outline for the final report can be found on Connect.