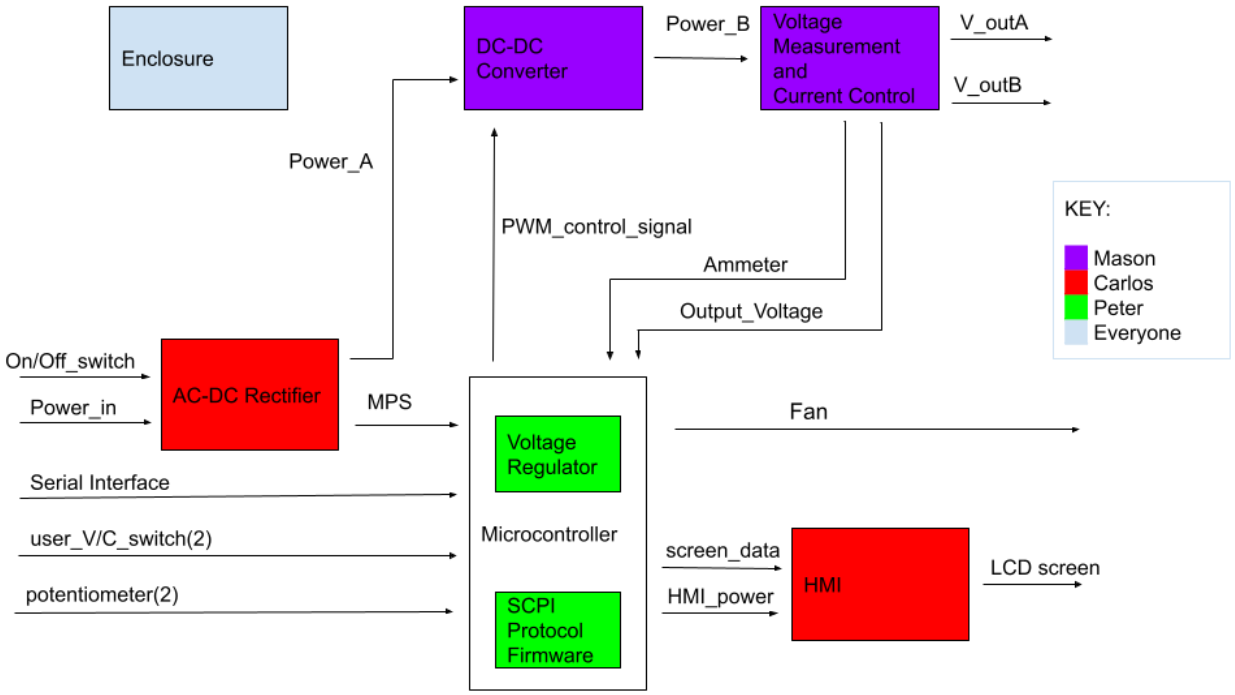
System Verification Documentation

Power 4

Mason Obery, Carlos Beleno, and Peter Thompson
5/27/22

Top-level Block Diagram:



Interface Definition:

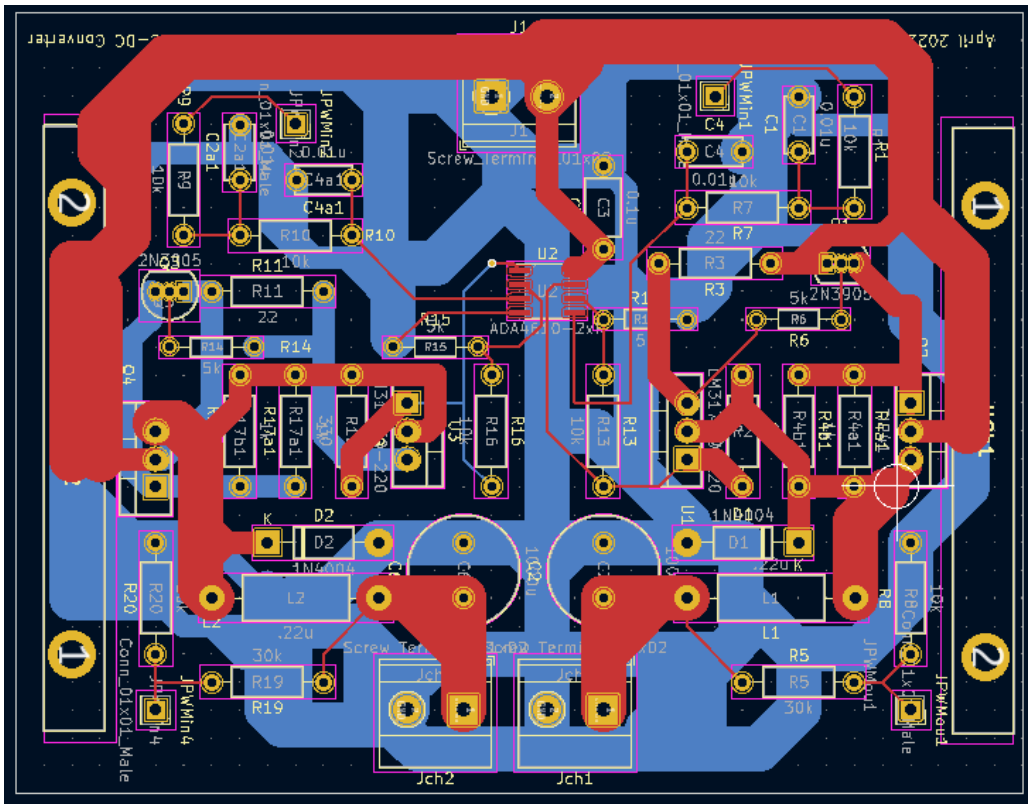
Interface Name	Type	Specifications
Power_in	acpwr	• $V_{nominal}$: 120VAC • Other: NEMA 5-15R • I_{peak}: 1A
Serial Interface	dsig	• Protocol: UART 8N1 • Baud rate: 9600 • Logic-Level: 5V • Data: ASCII Encoding of SCPI commands
user_V/C_switch(2)	usrin	• $V_{nominal}$: 5VDC • I_{peak}: 1mA
On/Off_switch	usrin	• $V_{nominal}$: 120VAC • I_{peak}: 1A
potentiometer(2)	usrin	• $V_{nominal}$: 5Vdc

		• $I_{peak: 1.5 A}$
Power_A	dcpwr	• C_l: 24VDC • I_{peak}: 3.75A
Power_B	dcpwr	• $V1_{max}$: 14V • $V1_{ABSmax}$: 20V • $V1_{min}$: 2V • $V1_{ABSmin}$: 0V • $I1_{max}$: 1.5A • $V2_{max}$: 14V • $V2_{ABSmax}$: 20V • $V2_{min}$: 2V • $V2_{ABSmin}$: 0V • $I2_{max}$: 1.5A
MPS	dcpwr	• V_{max}: 12V • V_{min}: 7V • I_{max}: 19mA
PWM_contol_signal	pwmsig	• V_{max}: 5V • I_{max}: 40mA • 4.3% -> 85.9% Duty Cycle • ~65000 Hz
Ammeter	vsig	• V_{min}: 4.5 V • V_{max}:5.5 V
Output_voltage	dcpwr	• V_{max}: 4.6V • I_{max}< 1mA
HMI_power	dcpwr	• V_{max}: 5V • I_{max}: .6mA • I_{nom}: .35mA
screen_data	dsig	• Logic level: 5V • 6pin logic
V_outA	dcpwr	• V_{max}: 14V • I_{max}: 1.5A
V_outB	dcpwr	• V_{max}: 14V • I_{max}: 1.5A
fan	dcpwr	• $V_{nominal}$:5 V • $I_{nominal}$: 0.2A
LCD_screen	usrout	• light

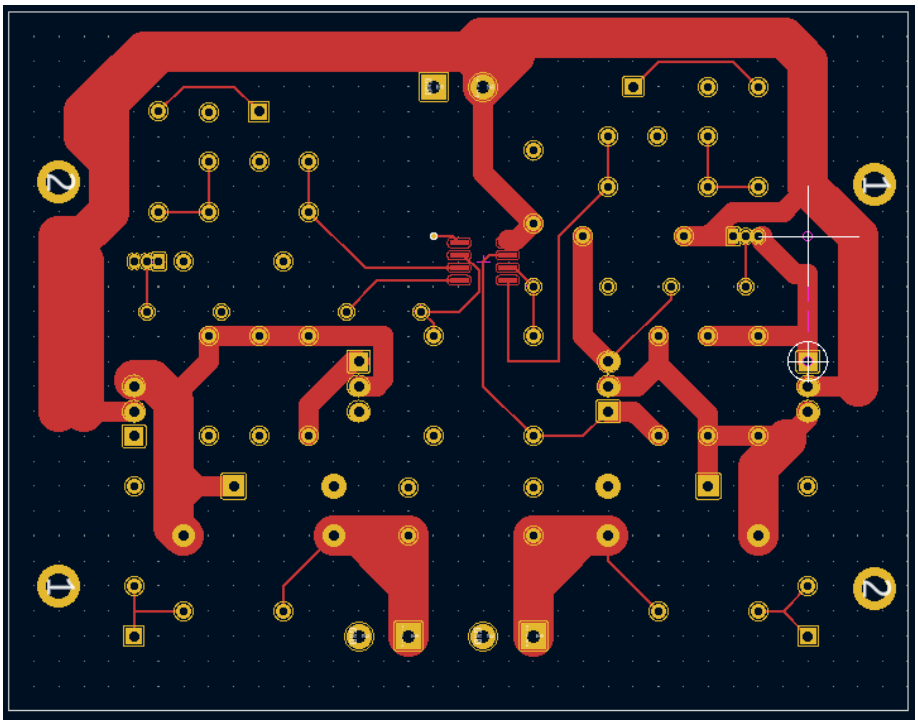
Wiring Diagram:

Attached, see “Schematic.pdf”.

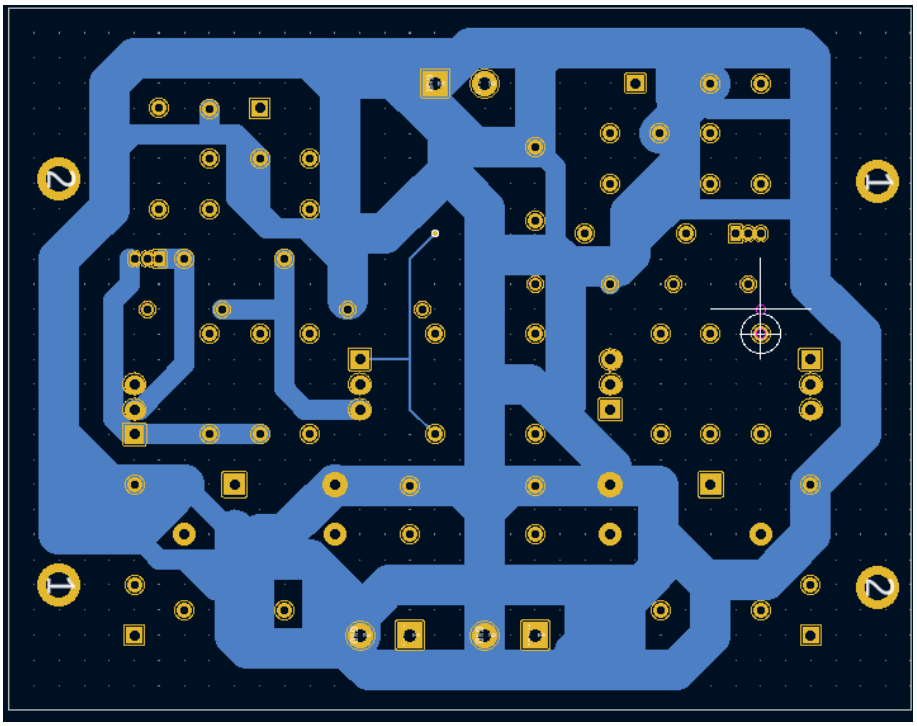
PCB Layout:



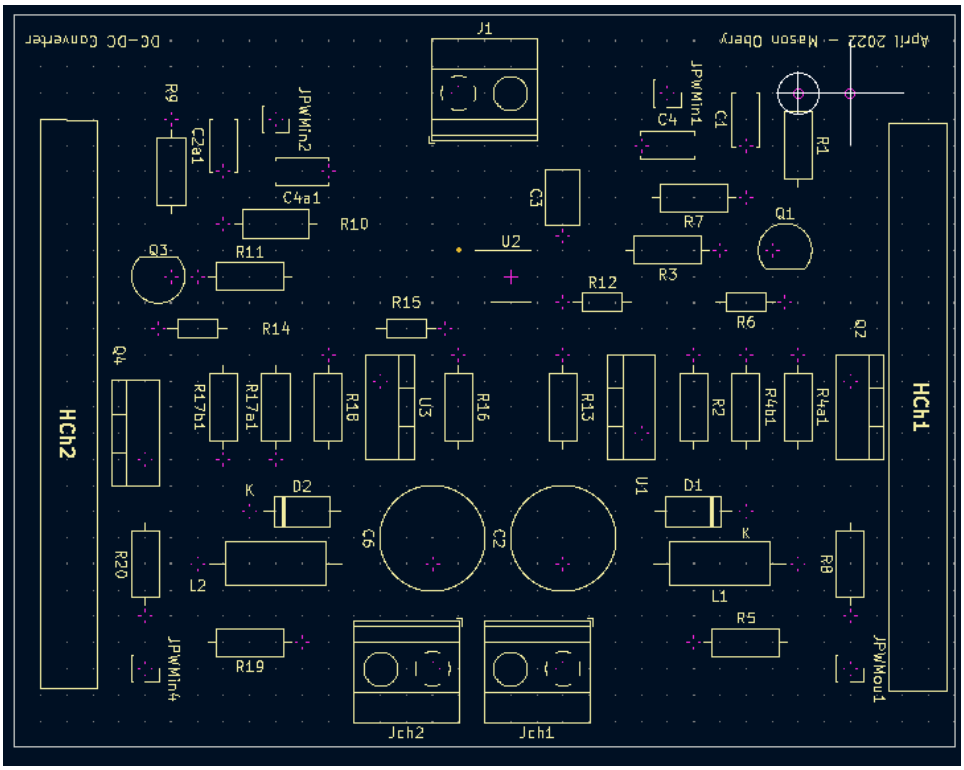
PCB Top Layer



PCB Bottom Layer



PCB Top Silkscreen

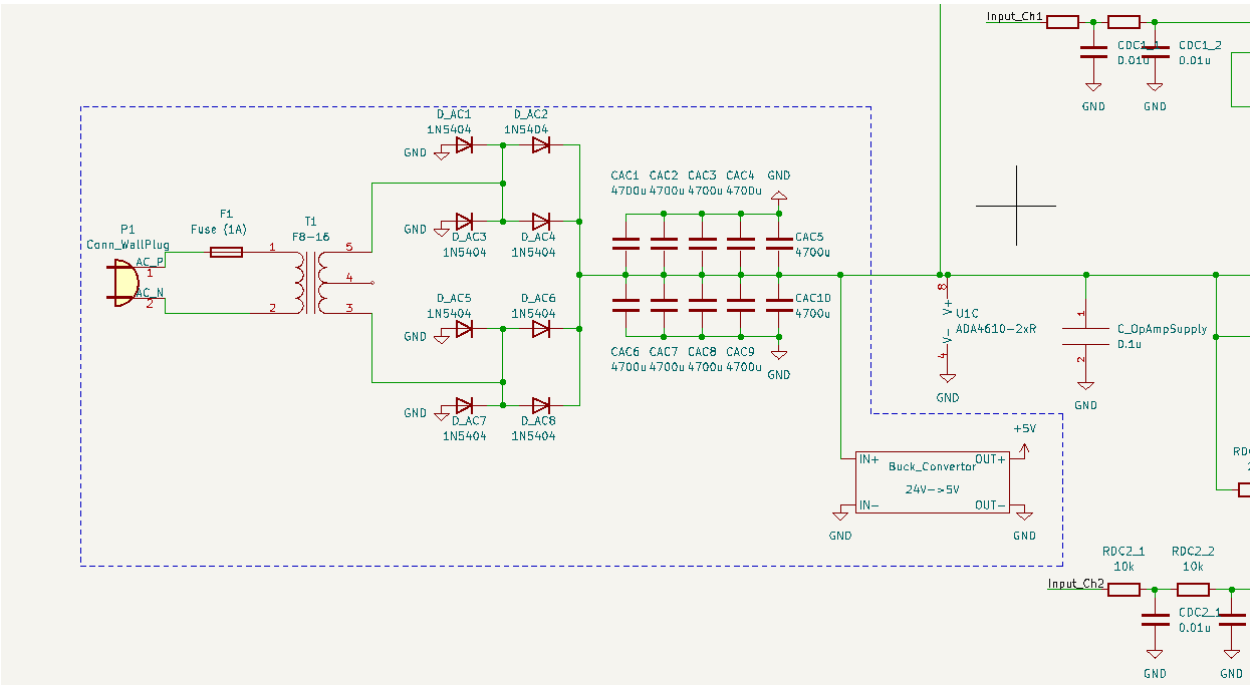


Block Explanations:

AC/DC Rectifier

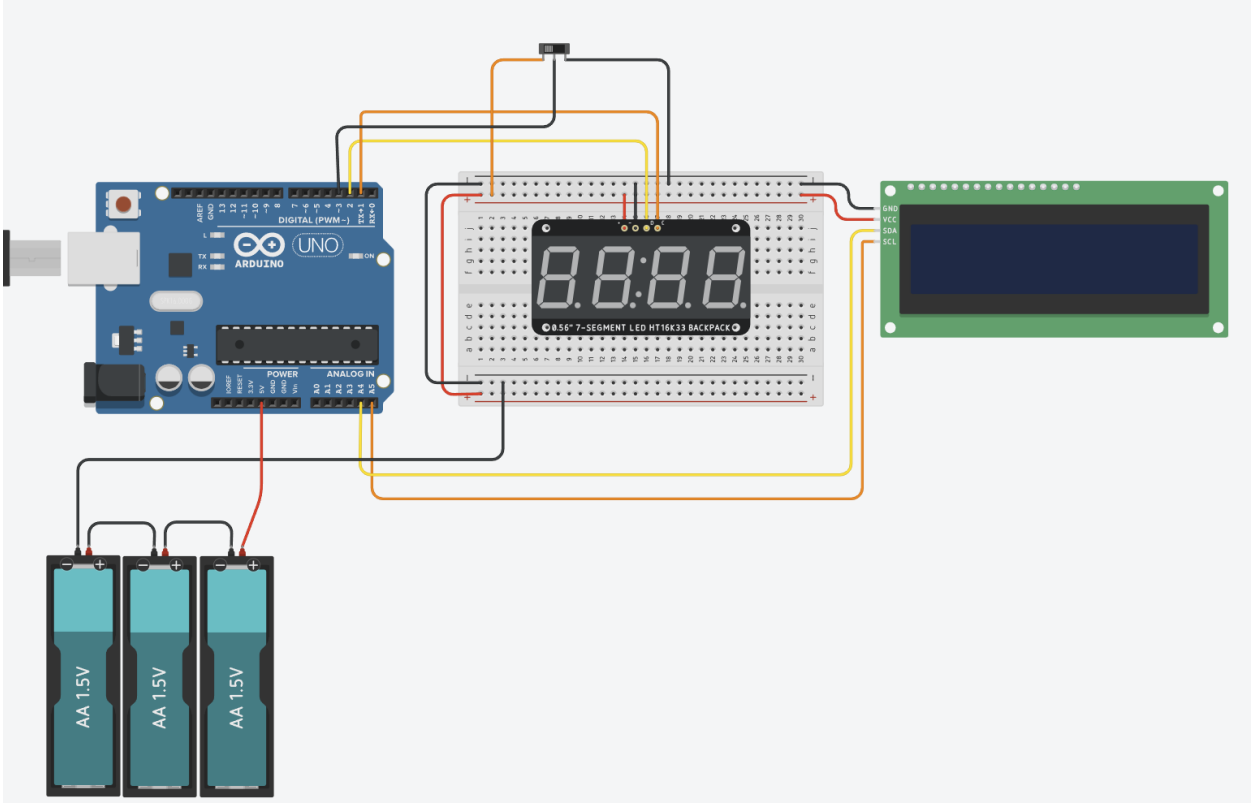
Operation of the Voltage Rectifier requires a IEC320 C14 inlet, a 120 VAC A Amp fuse with size 5x20mm (or 240 VAC), a buck converter, eight 1N5404 diodes, six (6) 18 AWG female spade crimps, two (2) 18 AWG male spade crimps, at least 24 inches of 14 AWG electrical wire, ten 4700 uF 25V-rated capacitors, a perforated board at least 9x7 cm in size, a 18 AWG AC power cord (NEMA 5-15 to C13 10 Amp), and a step-down transformer (F8-16). The below circuit diagram shows this block surrounded by the dotted line.

Circuit Diagram



Human Machine Interface (HMI)

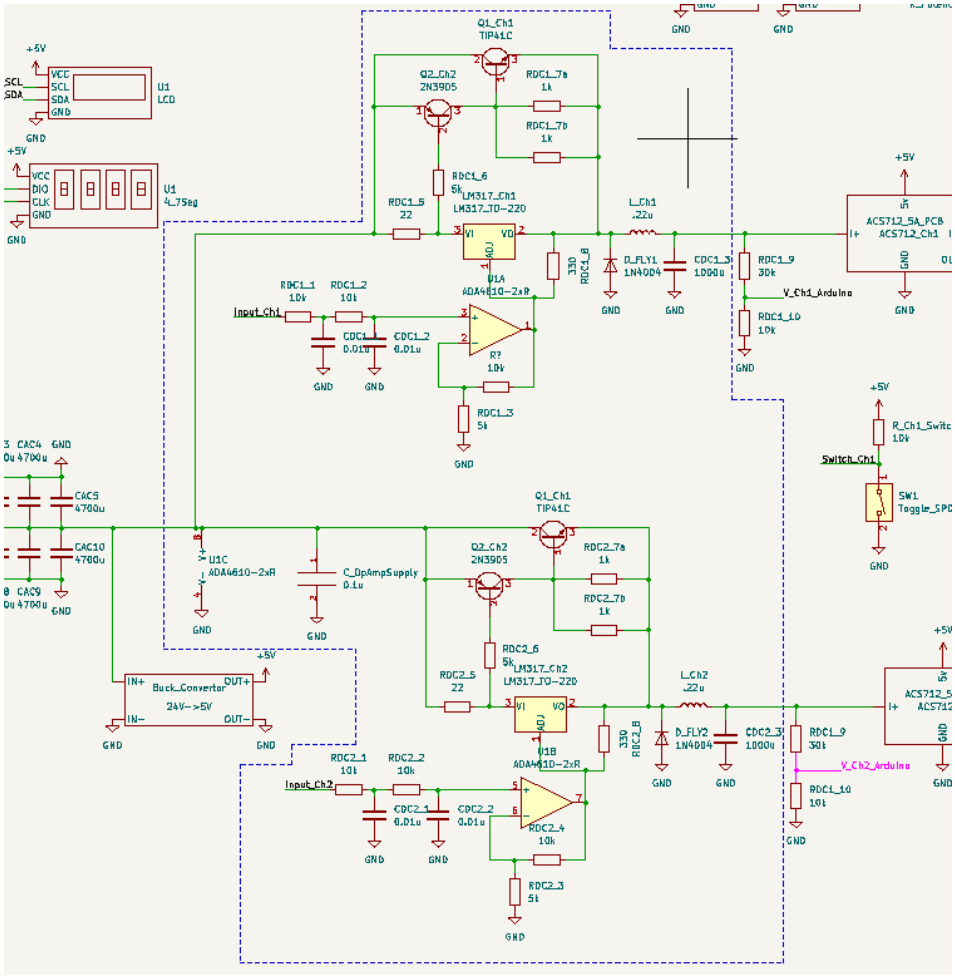
Operation of the Human-Machine Interface (HMI) requires a microcontroller, wiring, an LCD screen, a switch, a resistor (or potentiometer), a perforated board (or bread board), and a 5 V DC power source. The LCD screen used for this HMI iteration comes with an integrated I2C compatible interface. This means that the microcontroller and LCD screen communicate through I2C. Operation of the HMI is divided in three phases: Power on, Displaying, Power Off. During the Power On phase, the LCD screen does not display any usable data, while still providing backlight. During the Display phase, the LCD screen communicates through I2C with the microcontroller to display a voltage. The voltage given by the MC is captured by an analog pin in the MC and is measured from the analog pin to ground. The displayed voltage measured by the MC is updated every 200 ms. During the Power Off phase, the LCD screen is completely off and does not display backlight. This block also includes a four digit seven-segment display which shows which mode the power supply is currently using.



DC-DC Convertor

There are two separate circuits that receive a PWM signal, convert it to analog and amplify it, and use it to control a LM317. Each of the LM317 Voltage Regulators is connected to a Sziklai pair to enable higher current supply at each output. Each output is attached to a large capacitor and a small inductor in series to help smooth the output. Heat dissipation is a major concern, which is why the heat sinks are of such a size. The heatsinks will also be cooled by a fan, which is part of the enclosure. A voltage divider is placed at the output so that the Arduino is able to measure voltage. Two 1.5A fuses may be affixed to the ends of the board to cap current when the Arduino is unable to react adequately. A short circuit condition is needed as the circuit cannot disconnect itself from power or supply less than 1V. This block is shown below circled by a dotted line.

Circuit Diagram

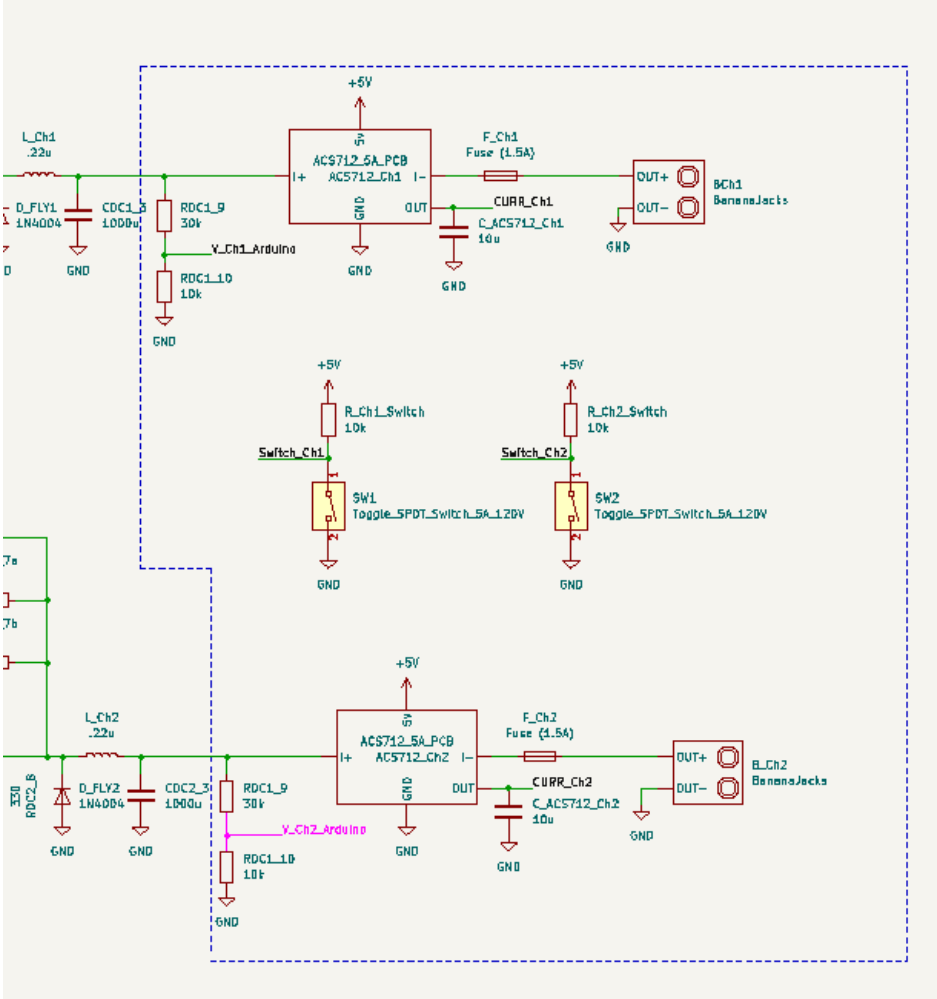


Current and Voltage Measurement

The two ACS modules are attached to the A1 and A2 analog input pins on the Arduino Nano. The Arduino Nano struggled initially with highly variable pin readings, and the ACS712 module has a 5A range. This means that just a small fluctuation can result in significantly different measured currents. To remedy this, two capacitors smooth the voltage entering the device. Because the device may not be running on a perfectly steady current from the wall, the code accounts for power supply fluctuations when calculating the current. A running average is also calculated to further smooth the data. Current is calculated from a line of best fit.

Voltage is measured using a voltage divider, which divides the voltage by four so that it can be measured by the Arduino Nano. Although the output should never exceed 14V and the maximum the Arduino can read is 25, the need for a safety margin is essential.

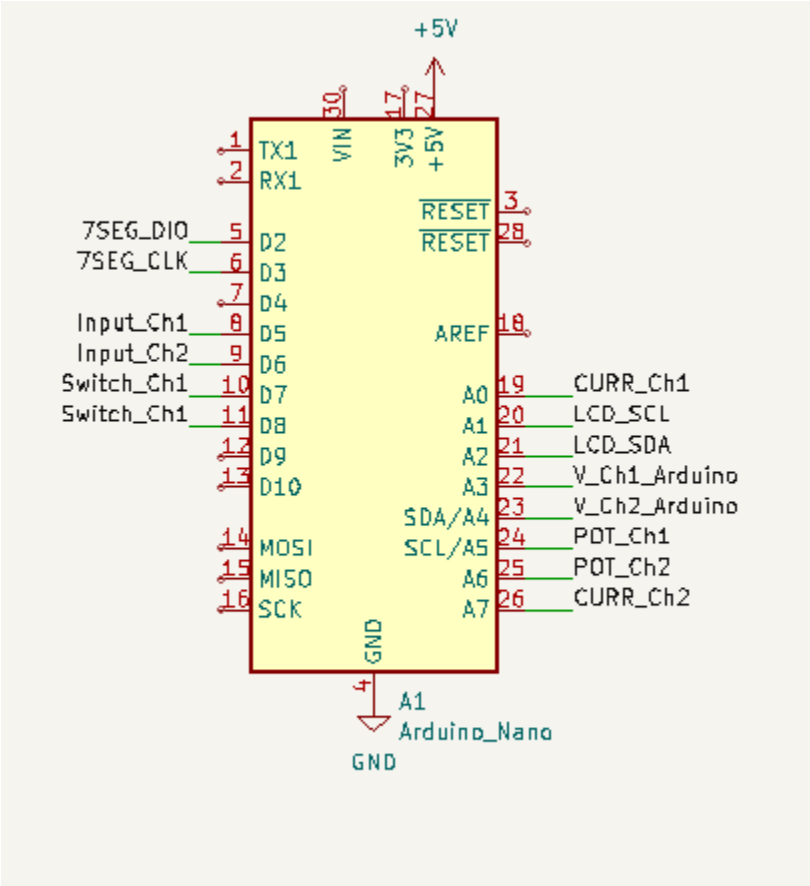
Circuit Diagram



Microcontroller

For our microcontroller we used an Arduino nano. It serves to control the voltage output with a PWM signal and show the outputs on an LCD screen. It can use both the system interface and a serial connection to control the power supply. We are using 10 of its pins as inputs and two as outputs as described in the pinout table.

Circuit Diagram



Pinout table

Pin Label	connection
A0	Read the current through node 1
A1	Connects to the LCD_SCL pin
A2	Connects to the LCD_SDA pin
A3	Reads the voltage of node 1

A4	Reads the voltage of node 2
A5	Reads the value of the first potentiometer
A6	Reads the value of the second potentiometer
A7	Reads the current through node 2
D5	PWM output for node 1
D6	PWM output for node 2
D7	Digital switch pin for node 1
D8	Digital switch pin for node 2

Code

All of this code was written on Arduino, and is run on the Arduino Nano microcontroller, etc. The code reads the voltage and currents at the nodes, it reads the serial inputs, and it reads the values of the potentiometers and switches. Using these inputs it will set the value of the PWM signal to meet the desired value given by the serial input or the potentiometers. It will then use the current and voltage at the nodes to adjust to the PWM to meet the desired voltage. It uses string parsing to convert the serial commands to values which are put through a switch statement. The LCD screen takes the measured values at the nodes and shows them to the user.

```
 /* Variable Voltage Control (version 1)
   * The point of this code is to use a PWM to set a voltage output
   * that will be amplified to create the needed voltage output
   * The arduino will detect the output voltage and current to adjust
   * its value. This also uses SCPI protocol to allow a serial connection
   * to control the outputs.
  */


#include <Wire.h>
#include <LiquidCrystal_I2C.h>

// set the LCD address to 0x27 for a 16 chars and 2 line display
LiquidCrystal_I2C lcd(0x27, 16, 2);

int voltage_out_1 = 0;  //the set vout value
int voltage_out_2 = 0;
float current_out_1 = 0;  //the set cout value
float current_out_2 = 0;
bool is_on_1 = false; //is node 1 on
bool is_on_2 = false; //is node 2 on
bool is_volt1 = true; //states the mode of operation of out 1
bool is_volt2 = true; //states the mode of operation of out 2
float measured_voltage_1 = 0; //the measured voltage output
float measured_voltage_2 = 0;
float measured_current_1 = 0; //the measured current output
```

```
float measured_current_2 = 0;
bool serial_only = false;

void setup() {
  //LCD setup
  lcd.init();  //initialize the lcd
  lcd.backlight();  //open the backlight
  Serial.begin(9600);
  lcd.begin(16,2);

  int PIN_READ_VOLT1 = A3; //pin number to read out voltage 1
  int PIN_READ_VOLT2 = A4; //pin number to read out voltage 2
  int PIN_READ_CURNT1 = A0; //pin number to read ammeter 1
  int PIN_READ_CURNT2 = A7; //pin number to read ammeter 2
  int PIN_POT1 = A5; //pin number to read pot 1
  int PIN_POT2 = A6; //pin number to read pot 2
  int PIN_SWITCH1 = PD7; //pin number to read switch 1
  int PIN_SWITCH2 = PB0; //pin number to read switch 2
  int PIN_PWM1 = PD5; //pin number to output pwm 1
  int PIN_PWM2 = PD6; //pin number to output pwm 2
  int Vpin = A1; //lcd pin 1
  int Vpin2 = A2; //lcd pin 2


  int PWM_CAP = 240; //caps the PWM signal
  int set_pwm1 = 0; //holds the pwm value for output 1
  int set_pwm2 = 0; //holds the pwm value for output 2
  unsigned long last_change = 0; //keeps track of time to allow the buffer to fill
  unsigned long buffer_wait = 0; //keeps track of time since the last change of out outputs
  unsigned long lcd_change = 0; //keeps track of when to wipe the LCD screen
  int pot_voltage1 = 0; //the return voltage of the first potentiometer
  int pot_voltage2 = 0; //the return voltage of the second potentiometer
  bool high_current1 = false; //flags if the current was above 1.5 A recently in out 1
  bool high_current2 = false; //flags if the current was above 1.5 A recently in out 2
  String command; //placeholder for the commands
  String output; // placeholder for the serial response

  // pin mode set
  pinMode(PIN_READ_VOLT1, INPUT);
  pinMode(PIN_READ_VOLT2, INPUT);
  pinMode(PIN_READ_CURNT1, INPUT);
  pinMode(PIN_READ_CURNT2, INPUT);
  pinMode(PIN_POT1, INPUT);
  pinMode(PIN_POT2, INPUT);
  pinMode(PIN_SWITCH1, INPUT);
  pinMode(PIN_SWITCH2, INPUT);
  pinMode(PIN_PWM1, OUTPUT);
  pinMode(PIN_PWM2, OUTPUT);

  // sets the PWM frequency of 62500.00 Hz
  TCCR0B = TCCR0B & B11111000 | B00000001;

while(true){
  measured_current_1 = get_current(digitalRead(PIN_READ_CURNT1));
  measured_current_2 = get_current(digitalRead(PIN_READ_CURNT2));
  measured_voltage_1 = get_voltage(digitalRead(PIN_READ_VOLT1));
  measured_voltage_2 = get_voltage(digitalRead(PIN_READ_VOLT2));

  if(Serial.available() && buffer_wait > millis() + 100){
            buffer_wait = millis();
  }
```

```
if(Serial.available() && buffer_wait > millis() + 100){
        command = get_command();
        output = read_command(command);
        Serial.print(output);
}

if(digitalRead(PIN_SWITCH1)){
        is_volt1 = true;
}
if(digitalRead(PIN_SWITCH2)){
        is_volt1 = true;
}
//if the system is not controlled by serial it will get its inputs from the interface
if(!serial_only){
        if(is_volt1){
        voltage_out_1 = get_pot_voltage(digitalRead(PIN_POT1));
        }
        else{
        current_out_1 = get_pot_voltage(digitalRead(PIN_POT1));
        }
        if(is_volt2){
        voltage_out_2 = get_pot_voltage(digitalRead(PIN_POT2));
        }
        else{
        current_out_2 = get_pot_voltage(digitalRead(PIN_POT1));
        }
}

if (last_change + 300 > millis()){
        if(is_volt1){
        set_pwm1 = adjust_voltage(voltage_out_1, measured_voltage_1, set_pwm1);
        }
        else{
        set_pwm1 = adjust_current(set_pwm1, measured_current_1, current_out_1, PWM_CAP);
        }
        if(is_volt2){
        set_pwm2 = adjust_voltage(voltage_out_2, measured_voltage_2, set_pwm2);
        }
        else{
        set_pwm2 = adjust_current(set_pwm2, measured_current_2, current_out_2, PWM_CAP);
        }
        last_change = millis();
}

if(measured_current_1 > 1.5){
        is_on_1 = false;
}
if(measured_current_2 > 1.5){
        is_on_2 = false;
}

if(is_on_1 = false){
        set_pwm1 = 0;
}
if(is_on_2 = false){
        set_pwm2 = 0;
}

analogWrite(PIN_PWM1,set_pwm1);
analogWrite(PIN_PWM2,set_pwm2);
```

```
  //LCD Screen printing
  lcd.setCursor(0, 0); // set the cursor to column 3, line 0
  lcd.print("Ch1:");  // Print a message to the LCD
  lcd.print(measured_voltage_1);

  lcd.setCursor(0, 1); // set the cursor to column 2, line 1
  lcd.print("Ch2:");  // Print a message to the LCD.
  lcd.print(measured_voltage_2);
  //wipes the lcd screen
  if(lcd_change + 200 < millis()){
            lcd.clear();
            lcd_change = millis();
  }
}

}
//end main

//control functions

  /*  adjust_voltage takes the output voltage and compares it to the desired voltage and returns a value to make it closer
   *  desired_value is the voltage that the function works towards
   *  vout is the voltage at the node
   *  pwm is the latest pwm signal
   *  returns a new PWM
   */
  int adjust_voltage(int desired_value, float vout, int pwm){
            float good_range_top = desired_value + desired_value * 0.05;
            float good_range_bot = desired_value - desired_value * 0.05;
            float quotient;
            if(desired_value < 2){
            return 0;
            }
            if(vout > good_range_bot && vout < good_range_top){
            return 0;
            }
            else{
            quotient = desired_value / vout;
            return pwm * quotient;
            }
  }

  /* adjust_current will calculate how to adjust the voltage to change the current to its desired value
   * last_PWM is the current PWM value that is being used
   * voltage_out is the voltage on the out node
   * current is the current through the system
   * desired_current is the current value in amps that the function will achieve
   * PWM_CAP is the maximum allowed PWM for the DC-DC voltage converter
   * returns a PWM value for the desired current
   */
  int adjust_current(int last_PWM, float current, float desired_current, int PWM_CAP){
            float quotient;
            int pwm = 0;
            if(desired_current < 2){
            return 0;
            }
            if(last_PWM == 0){
            return 17;   // This is subject to change
            }
            float too_high = desired_current + 0.05 * desired_current;
```

```
            float too_low = desired_current - 0.05 * desired_current;
            if( current < too_high && current > too_low){
            return last_PWM;
            }
            else {
            quotient = desired_current / current;
            pwm = (int)last_PWM * quotient;
            if (pwm <= PWM_CAP){
            return pwm;
            }
            else{
            return PWM_CAP;
            }
            }
}


 /* calculates the value of the potentiometer
  * voltage is the voltage delivered to the arduino from the potentiometer circuit
  * return the integer value of 0, and 2 to 14 of the desired voltage
 */
 int get_pot_voltage(int voltage){

            if(voltage < 144){
            return 0;
            }
            else{
            return voltage / 72;
            }
}

 /* calculates the voltage from the ammeter to describe the current in amps
  * ampmeter_value is the voltage from the ammeter to be converted
  * returns the value of the current in amps
 */
 float get_current(int ampmeter_value){
            float current = ((4.8/1023)*(ampmeter_value - 510) /0.169);
            return current;
}

 float get_voltage(int voltage_value){
            float voltage = 19.2 * voltage_value/ 1024;
            return voltage;
}


//serial functions

/* get_command will take read from the serial buffer and
 * return the String sent to the buffer, it ends with a semicolon
 *
 * returns the input string without the semicolon or error if no semicolon is found
*/

String get_command(){
  char inByte;
  String cmd;
  while (Serial.available() > 0) {
            inByte = Serial.read();
            if(inByte == ';'){
            Serial.println(" received");
```

```
            return cmd;
            }
            else {
            cmd += inByte;
            }
 }
 return "  error";
}



/* read_command takes in a command line and parses it to be used by run_command
 *
 * cmd is the input command string including spaces and colons
 * returns a string saying error or run_command's return string
 */


String read_command(String cmd){
  char ch;
  String temp = "";
  bool query = false;
  float value = -1;
  int cmd_val = 0;
  for(int i = 0; i < cmd.length(); i++){
            ch = cmd[i];
            if(ch == ':' || ch == ' ' || ch == '?'){
            //saves the command
            if(temp.length() > 4){
            temp = temp.substring(0,4);
            }
            //adds the values to a int
            cmd_val += cmd_val * 10;
            cmd_val += code_map(temp);
            //checks for spaces leading numerical values values
            if(ch == ' '){
            temp = cmd.substring(i+1);
            value = temp.toFloat();
            return run_command(cmd_val, value, query);
            }
            else if(ch == '?'){
            query = true;
            }
            }
            else{
            temp.concat(ch);
            }
 }
  // adds final command value
  cmd_val += cmd_val * 10;
  cmd_val += code_map(temp);
  return run_command(cmd_val, value, query);
}

/* run_command will take a command and execute it, if it is
 * not a known command it will return an error, if it is valid, it
 * will acknowledge its success
 *
 * cmd1 the first part of the command
 * cmd2 the second part of the command
```

```
 * cmd3 the third part of the command
 * cmd4 the fourth part of the command
 * value is the input for floats
 * query is a bool that denotes if it expects a value response
 *
 * returns error strings or acknowledges success
*/

String run_command(int cmd, float value, bool query){

 //returns values
 if(query){
        switch(cmd){
        //returns the intended voltage of node 1
        case 31:
        return "The intended voltage of node 1 is: " + (String)voltage_out_1;
        break;
        //returns the intended voltage of node 2
        case 32:
        return "The intended voltage of node 2 is: " + (String)voltage_out_2;
        break;
        //returns the intended current of node 1
        case 41:
        return "The intended current of node 1 is: " + (String)current_out_1;
        break;
        //returns the intended current of node 2
        case 42:
        return "The intended current of node 2 is: " + (String)current_out_2;
        break;
        //measure the voltage of node 1
        case 731:
        return "The voltage at node 1 is: " + (String)measured_voltage_1;
        break;
        //measure the voltage of node 2
        case 732:
        return "The voltage at node 2 is: " + (String)measured_voltage_2;
        break;
        //measure the current of node 1
        case 741:
        return "The current at node 1 is: " + (String)measured_current_1;
        break;
        //measure the current of node 2
        case 742:
        return "The current at node 2 is: " + (String)measured_current_2;
        break;
        //defaults to an error
        default:
        return "command error";
        }
 }
 //sets values
 else{
        switch(cmd){
        //set the voltage of node 1
        case 31:
        if(value >= 0 && value <= 14){
        voltage_out_1 = (int)value;
        return "set node 1 current to" + (String)value;
        }
        else{
        return "bad value error";
```

```
}
break;
//set the voltage of node 2
case 32:
if(value >= 0 && value <= 14){
voltage_out_2 = (int)value;
return "set node 2 voltage to" + (String)value;
}
else{
return "bad value error";
}
break;
//set the current of node 1
case 41:
if(value >= 0 && value <= 1.5){
current_out_2 = value;
return "set node 1 current to" + (String)value;
}
else{
return "bad value error";
}
break;
//set the current of node 2
case 42:
if(value >= 0 && value <= 1.5){
current_out_2 = value;
return "set node 2 current to" + (String)value;
}
else{
return "bad value error";
}
break;
//sets the arduino to be controlled by Serial
case 95:
serial_only = true;
return "serial only";
break;
case 96:
serial_only = false;
return "interface only";
break;
//change the output type of node 1 to voltage
case 831:
is_volt1 = true;
return "node 1 is a voltage source";
break;
//change the output type of node 2 to voltage
case 832:
is_volt2 = true;
return "node 2 is a voltage source";
break;
//change the output type of node 1 to current
case 841:
is_volt1 = false;
return "node 1 is a current source";
break;
//change the output type of node 2 to current
case 842:
is_volt2 = false;
return "node 2 is a current  source";
break;
```

```
            //turn node 1 on
            case 851:
            is_on_1 = true;
            return "node 1 on";
            break;
            //turn node 2 on
            case 852:
            is_on_2 = true;
            return "node 2 on";
            break;
            //turn node 1 off
            case 861:
            is_on_1 = false;
            return "node 1 off";
            break;
            //turn node 2 off
            case 862:
            is_on_2 = false;
            return "node 2 off";
            break;
            //bad command error
            default:
            return "command error";
            }
  }
}


/*  code map takes a four letter command and assigns a number to it
 *  cmd is the command to be assigned
 *  return a integer value between 1 and 9
*/


int code_map(String cmd){
  String terms[9] = {"ONE", "TWO", "VOLT", "CURR", "STAR", "STOP", "MEAS", "OUTP", "CONT"};
  for (int i = 0; i < 9; i++){
            if (cmd == terms[i]){
            return i + 1;
            }
  return -1;
  }
}
```

Bill of Materials:

Element	Count	Datasheet	Origin	Total Cost
PCB	3 Ordered, 1 Used		OSHpark	$57.75
PV-T21-38E TO-220 Heatsinks	2	https://www.ohmite.com/assets/docs/sink_p.pdf	Mouser	$14.06
TIP41 (TO-220 Package)	2	https://resi.store/products/792/tip41.	Resistore	$0.80

		pdf		
2N3905 (TO-90 Package)	2	https://my.centralsemi.com/datasheets/2N3905.PDF	Mouser	$0.88
LM314 Voltage Regulator (TO-220 Package)	2	https://www.ti.com/lit/ds/symlink/lm317.pdf	Resistore	$1.00
22Ω resistor	2	https://resi.store/products/141/resistor.pdf	Resistore	$0.10
330Ω resistor	2	https://resi.store/products/141/resistor.pdf	Resistore	$0.10
1kΩ resistor	4	https://resi.store/products/141/resistor.pdf	Resistore	$0.20
10kΩ resistor	8	https://resi.store/products/141/resistor.pdf	Resistore	$0.40
30kΩ resistor	2	https://resi.store/products/141/resistor.pdf	Resistore	$0.10
5kΩ resistor	4	https://www.mouser.com/datasheet/2/447/Yageo_08272021_CFR_Series-2530393.pdf	Mouser	$0.40
Block Terminal	3	https://www.mouser.com/datasheet/2/670/tb007_508-2306759.pdf	Mouser	$1.98
0.22uH Axial Inductor	2	https://product.tdk.com/system/files/dam/doc/product/inductor/inductor/lead/data_sheet/30/ds/b78108_148e.pdf	Mouser	$0.82
ADA 4098-2BR2	1	https://www.analog.com/media/en/technical-documentation/data-sheets/ada4098-1_ada40	DigiKey	$6.56

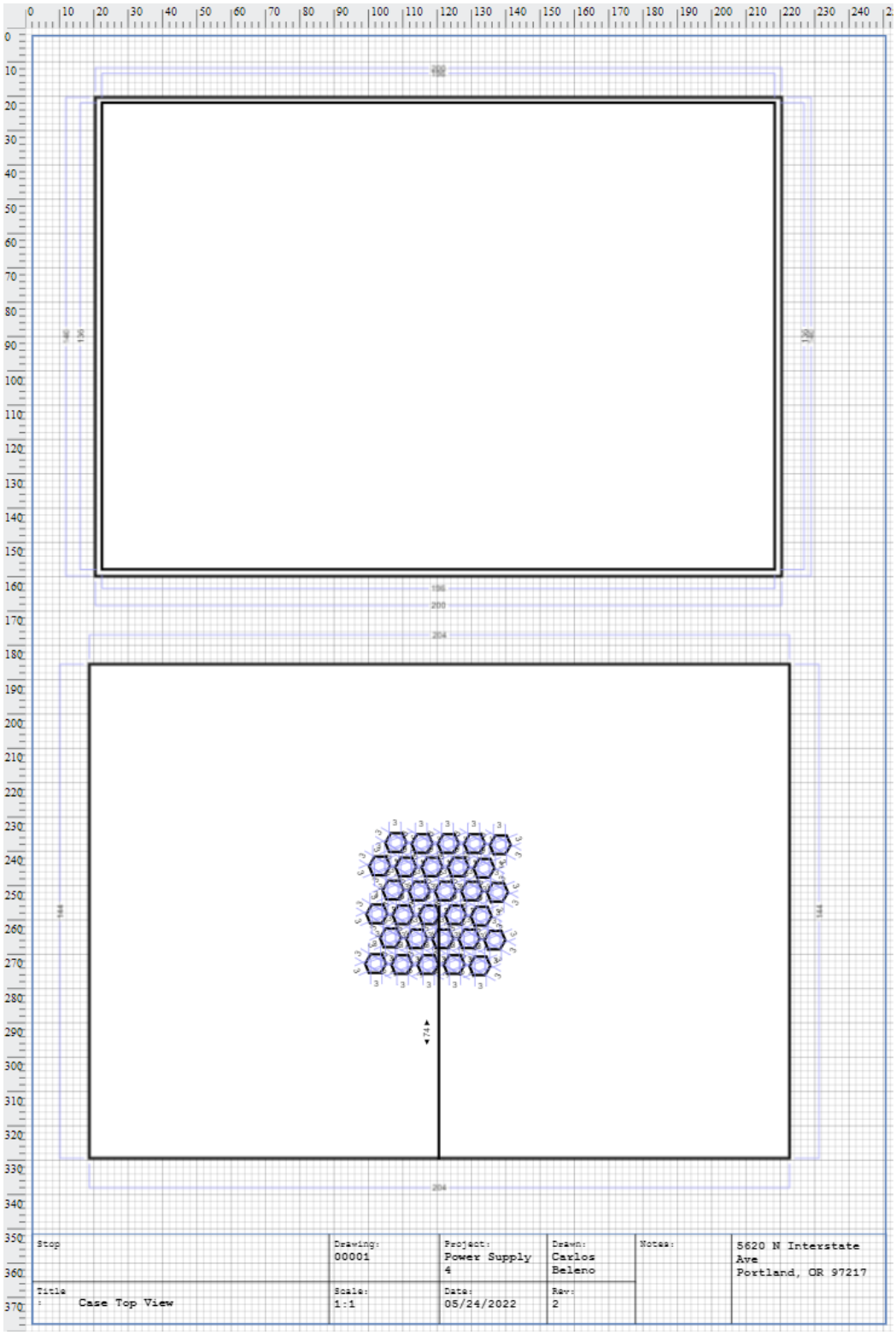
		98-2.pdf		
10nF Ceramic Capacitor	4	https://resi.store/products/270/ceramic.pdf	Resistore	$1.40
0.1 uF Film Capacitor	1	https://resi.store/products/496/filmcap.pdf	Resistore	$0.70
1000uF Electrolytic Capacitor	2	https://resi.store/products/291/electrolytic.pdf	Resistore	$1.00
1.5A Glass Fuse	10 Ordered, 2 Used	https://www.mouser.com/datasheet/2/643/ds_cp_5mf_5mfp_series-1313109.pdf	Mouser	$2.70
Fuse Holder	2	https://eecs.oregonstate.edu/education/inventory_datasheets/P153-1418756154.pdf	TekBots	$0.00
Arctic Silver 5 Thermal compound	3.5g	SDS: http://www.arcticsilver.com/PDF/AS5_SDS.pdf Website: http://www.arcticsilver.com/as5.htm	Bellevue Computer	$8.99
1N5404-B Rectifier Diode	8	https://www.mouser.com/datasheet/2/345/1n5400g_1n5408g-1622519.pdf	Mouser	$1.52
10uF Electrolytic Capacitor	2	https://resi.store/products/281/electrolytic.pdf	Resistore	$0.90
Female-to-female Jumper Wires	11	https://resi.store/products/1020/awg.pdf	Resistore	$0.00
14 Ga Wire		https://www.homedepot.com/p/Southwire-50-ft-14-Red-Stranded-CU-THHN-Wire-22957583/204834022	Home Depot	

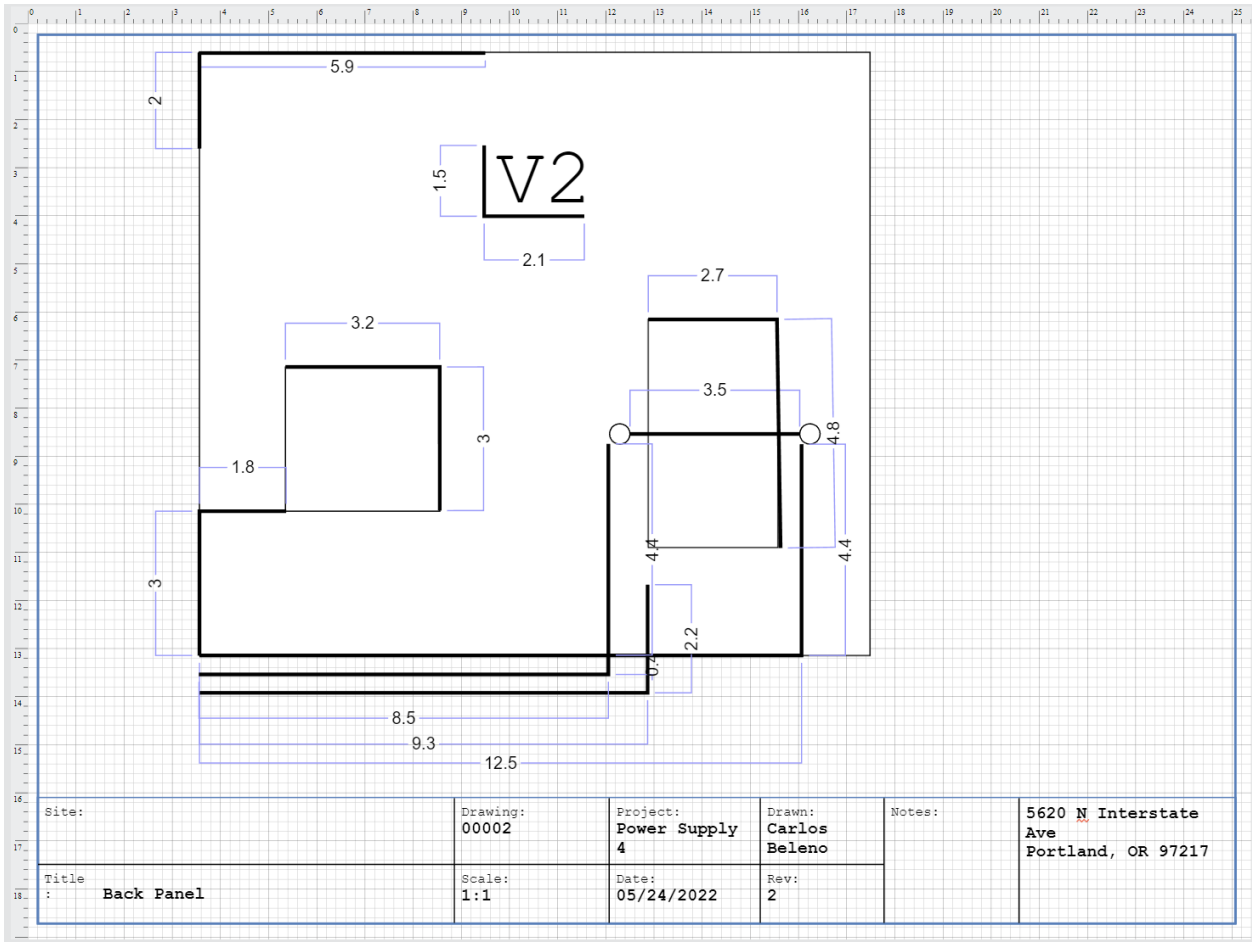
1A Glass Fuse	3	https://resi.store/products/554/fuse.pdf	Resistore	$1.50
3D Printer Reel	1	https://www.amazon.com/dp/B01EKEMDA6?ref=nb_sb_ss_w_as-reorder-t1_ypp_rep_k0_1_4&amp&crid=4K92T3WXAQTL&amp&sprefix=esun	Amazon	22.99
50mm 5V fans	1 4-pack	https://www.amazon.com/dp/B08CMVKHDX?ref=ppx_yo2ov_dt_b_product_details&th=1	Amazon	$10.99
Power Cable	1	https://www.amazon.com/dp/B001CATESG?psc=1&ref=ppx_yo2ov_dt_b_product_details	Amazon	$4.20
IEC320 Inlet	2	https://www.amazon.com/dp/B08P8ZJNZ6?psc=1&ref=ppx_yo2ov_dt_b_product_details	Amazon	8.29
Perfboard Kit	1	https://www.amazon.com/dp/B088GP981V?psc=1&ref=ppx_yo2ov_dt_b_product_details	Amazon	$14.59
Standoff Kit	1	https://www.amazon.com/dp/B08FZKLYSB?psc=1&ref=ppx_yo2ov_dt_b_product_details	Amazon	$11.99
8/32 Machine Nut	2 (Sold by the piece)	https://www.acehardware.com/departments/hardware/screws-and-anchors/machine-screws/59329	Ace Hardware	$0.20
6/32 Machine Screw	13 (Sold by the piece)	https://www.acehardware.com/departments/hardware/screws-and-anchor	Ace Hardware	$1.30

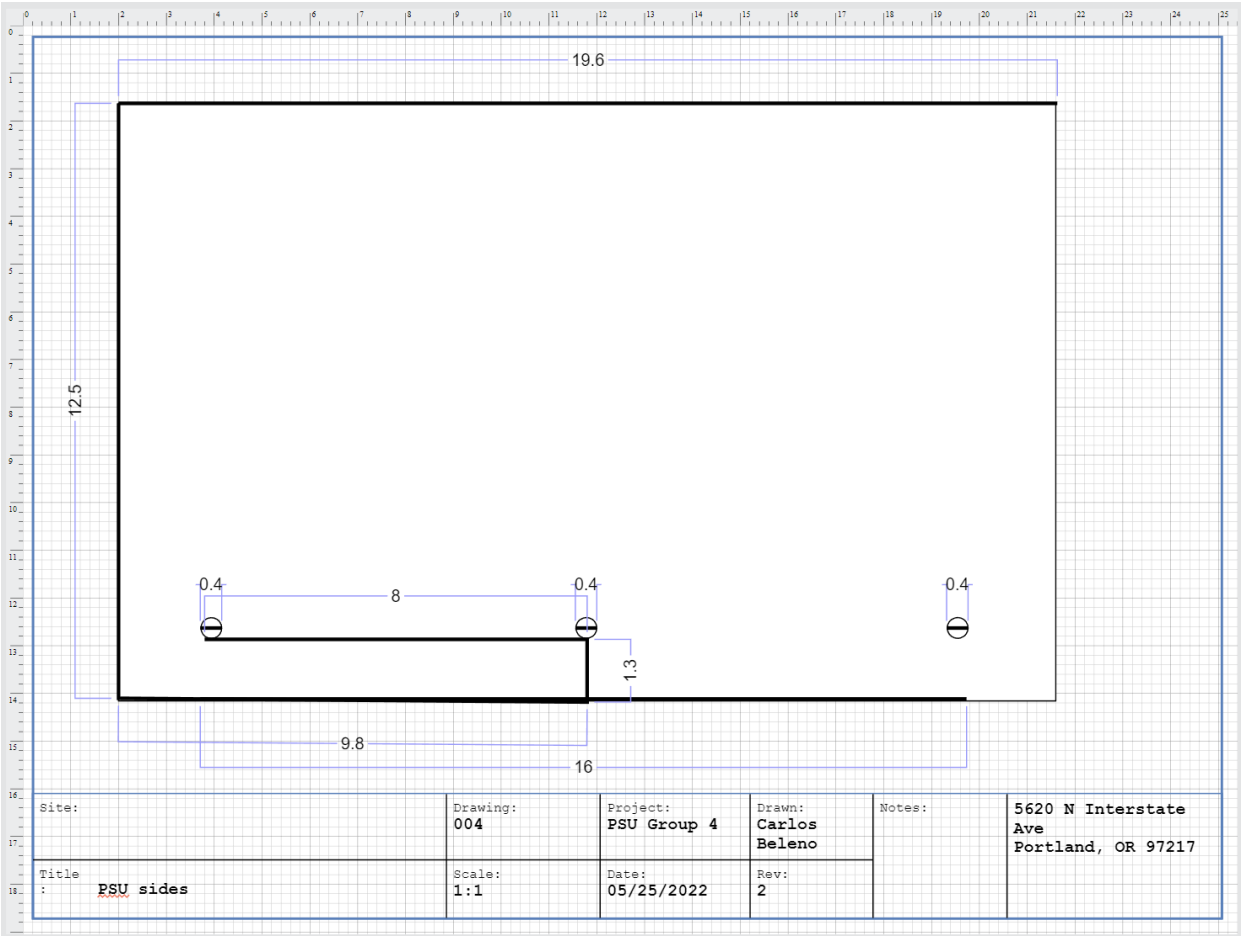
		s/machine-screws/59253		
6/32 Machine Nut	12 (Sold by the piece)	https://www.acehardware.com/departments/hardware/nuts-and-bolts/nuts/59556	Ace Hardware	$1.20
8/32 Machine Nut	2 (Sold by the piece)	https://www.acehardware.com/departments/hardware/nuts-and-bolts/nuts/56366	Ace Hardware	$0.20
#4 x 1/2 in. Phillips Flat Head Brass Wood Screw	16	https://www.homedepot.com/p/4-x-1-2-in-Phillips-Flat-Head-Brass-Wood-Screw-6-Pack-809501/204275698	Home Depot	$3.00
White Pine	1		Woodshop scraps	$0.00
F8-16 Transformer	1	https://catalog.triadmagnetics.com/Asset/F8-16.pdf	Digi-Key	$26.32
LCD	2	https://www.openhacks.com/uploadsproductos/eone-1602a1.pdf	Amazon	10.99
10k Ohm Linear Potentiometer	2	https://resi.store/products/822/linearpot.pdf	Resistore	$2.00
Toggle SPDT Switch 5A 120V	2	https://resi.store/products/1003/tactile.pdf	Resistore	$6.00
4 digit 7 Segment display	1	https://www.mcielectronics.cl/website_MCI/static/documents/Datasheet_TM1637.pdf	Amazon	1.99
Electrical Tape (66ft)	1	https://www.homedepot.com/p/3M-Scotch-0-75-in-x-66-ft-x-7-mil-35-Electrical-Tape-White-10828-DL-2W/100555712	Home Depot	$5.97

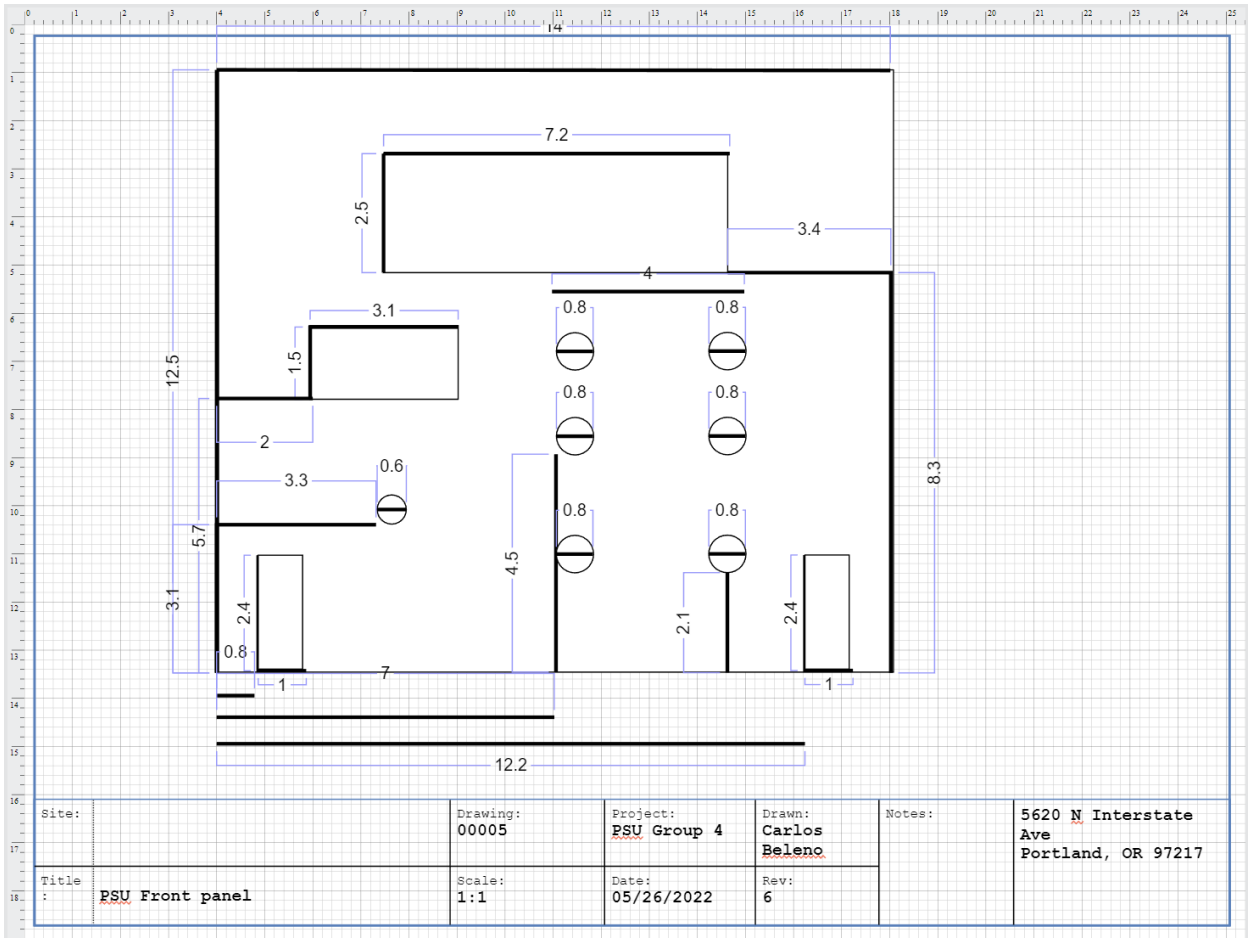
Mechanical Drawings:

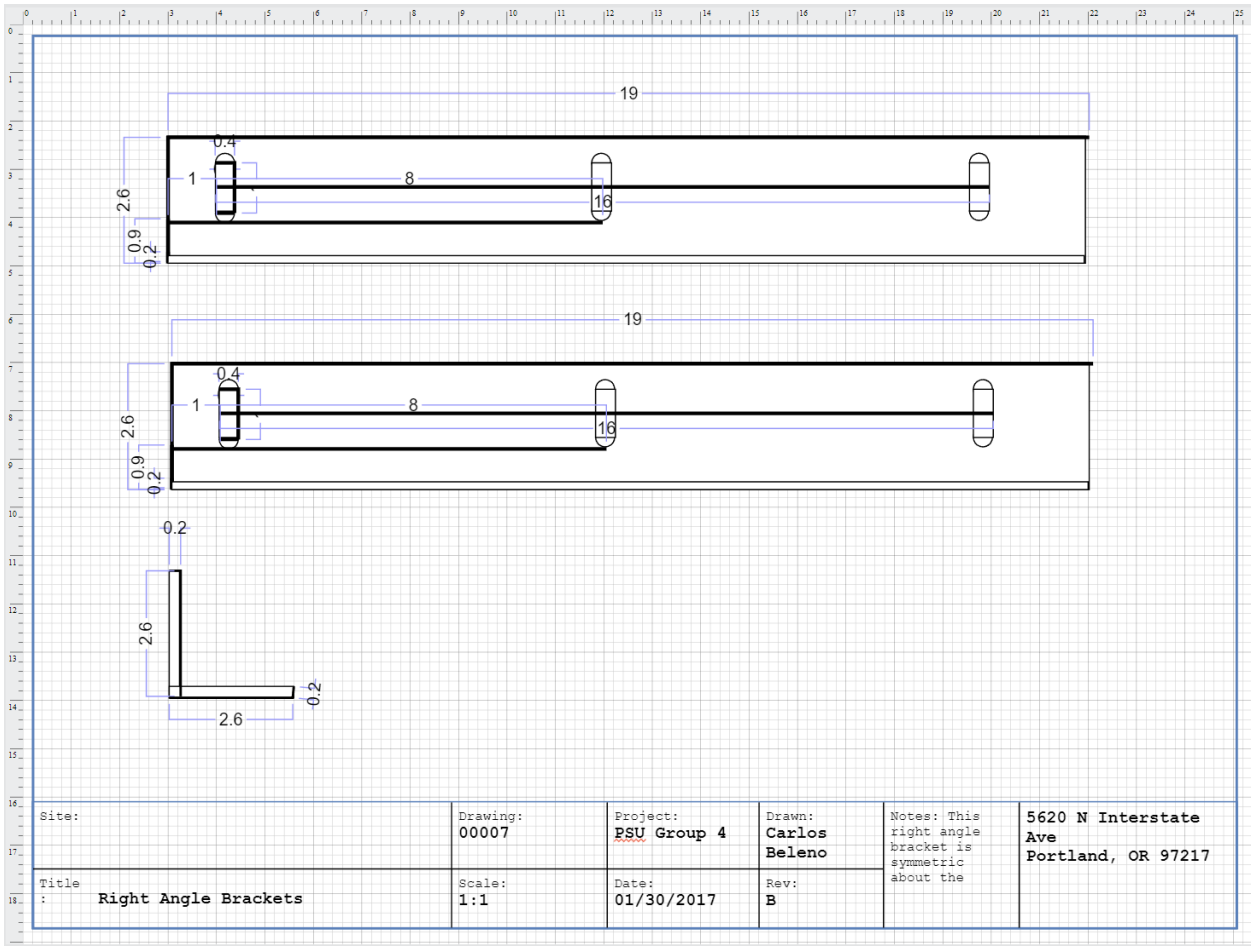
All drawings are given in cm unless otherwise specified.



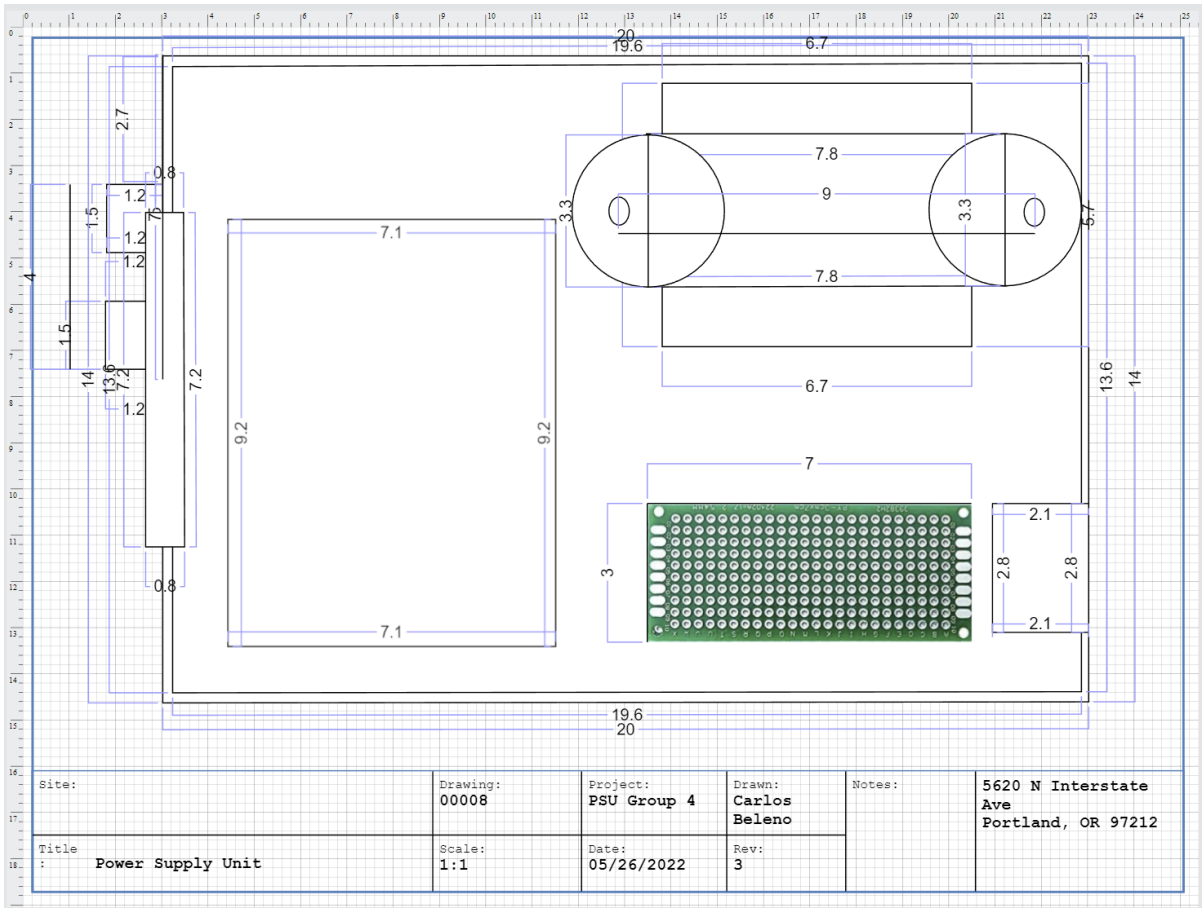

V2
5.9
2
1.5
2.1
2.7
3.2
3.5
4.8
1.8
3
4.4
4.4
3
2.2
0.4
8.5
9.3
12.5
Site:
Drawing:
00002
Project:
Power Supply 4
Drawn:
Carlos Beleno
Notes:
5620 N Interstate Ave
Portland, OR 97217
Title
:
Back Panel
Scale:
1:1
Date:
05/24/2022
Rev:
2

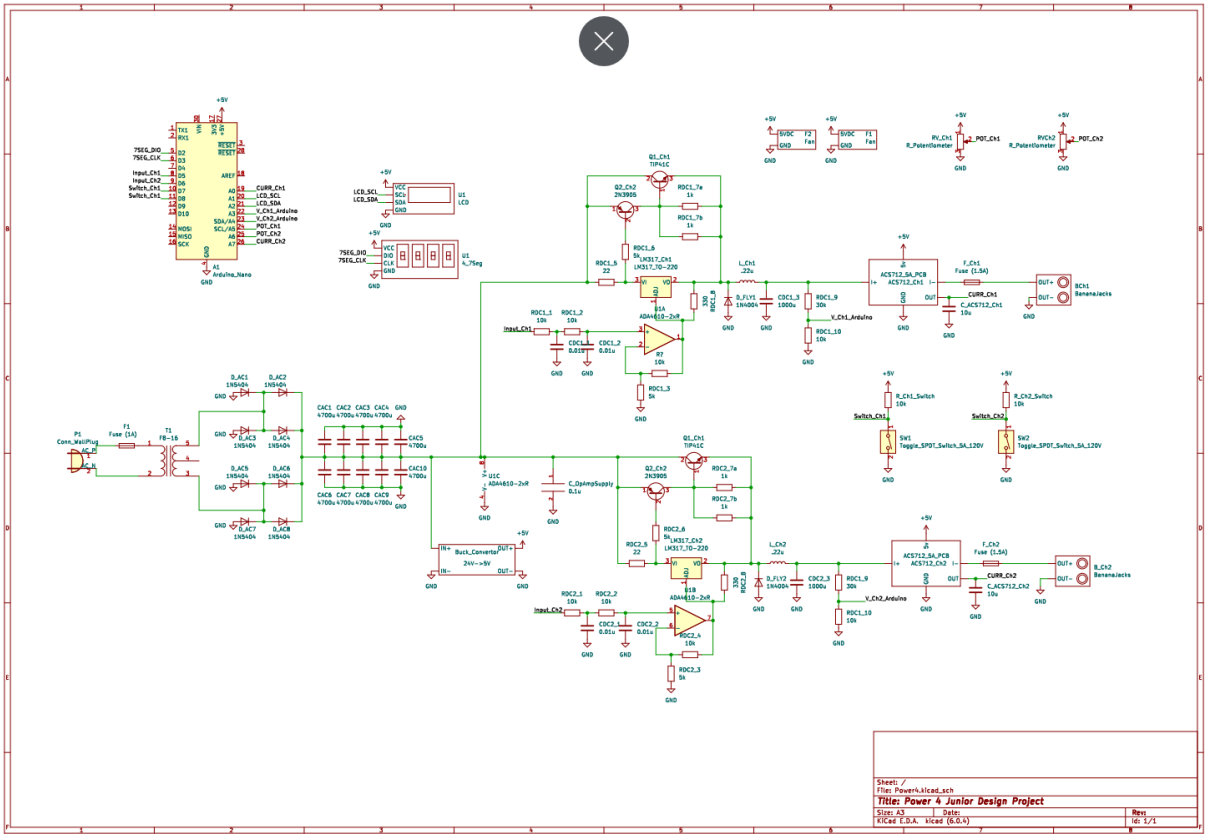

19.6
12.5
0.4
8
0.4
0.4
1.3
9.8
16
Site:
Drawing:
004
Project:
PSU Group 4
Drawn:
Carlos Beleno
Notes:
5620 N Interstate Ave
Portland, OR 97217
Title:
PSU sides
Scale:
1:1
Date:
05/25/2022
Rev:
2


14
7.2
2.5
3.4
4
3.1
1.5
12.5
2
3.3
0.6
5.7
3.1
2.4
0.8
1
7
4.5
0.8
0.8
0.8
0.8
0.8
0.8
2.1
2.4
1
8.3
12.2
Site:
Drawing:
00005
Project:
PSU Group 4
Drawn:
Carlos Beleno
Notes:
5620 N Interstate Ave
Portland, OR 97217
Title :
PSU Front panel
Scale:
1:1
Date:
05/26/2022
Rev:
6



Site:	Drawing: 00007	Project: PSU Group 4	Drawn: Carlos Beleno	Notes: This right angle bracket is symmetric about the	5620 N Interstate Ave Portland, OR 97217
Title: Right Angle Brackets	Scale: 1:1	Date: 01/30/2017	Rev: B		


20
19.6
6.7
2.7
0.8
1.2
1.5
7
1.2
1.2
4
1.5
14
13.6
7.2
1.2
7.2
7.1
9.2
9.2
7.1
7.8
9
7.8
3.3
3.3
5.7
6.7
13.6
14
7
3
2.1
2.8
2.8
2.1
0.8
19.6
20
Site:
Drawing:
00008
Project:
PSU Group 4
Drawn:
Carlos
Beleno
Notes:
5620 N Interstate
Ave
Portland, OR 97212
Title
:
Power Supply Unit
Scale:
1:1
Date:
05/26/2022
Rev:
3


Sheet: /
File: Power4.kicad_sch
Title: Power 4 Junior Design Project
Size: A3
Date:
KiCad E.D.A. kicad (6.0.4)
Rev:
Id: 1/1

Time Report:

	Hours Contributed		
Week	Mason	Carlos	Peter
1	0	2	1
2	6	5	6
3	3	4	5
4	6	10	10
5	16	12	12
6	7	4	4
7	10	16	9
8	19	16	8
9	18	15	8