System Verification Documentation

Power Supply 004-01

Amanda Jung, Raymond Lee, David Luo

March 10, 2023

Table of Contents

1 Block Diagrams

1.1 Black Box Diagram

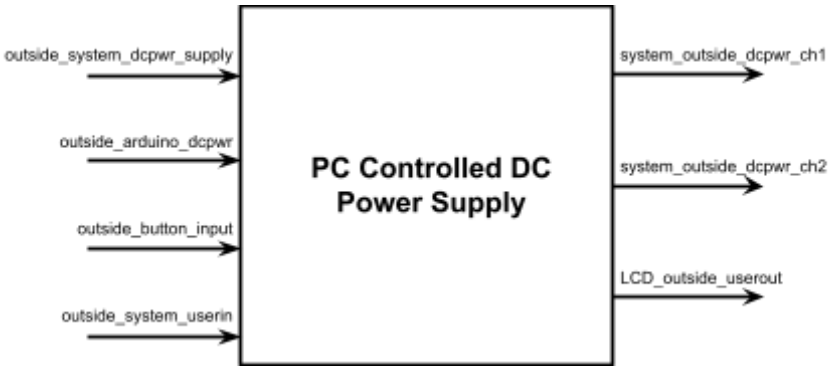


Figure 1: Black Box Diagram

The PC controlled DC power supply will take a DC input ranging from 17V to 28V to power the system. The system will have two output channels where the voltage for each channel can be adjusted independently either mechanically through buttons and switches or digitally through the PC that communicates with the microcontroller of the system. The user will be able to define voltages on the PC through SCPI protocol to communicate with the system. The range for the output voltage of each channel is 2V to 14V. The voltages and currents outputting on the system will be displayed to the user on an LCD as well as on the serial monitor. There are also safety measures in place that disable the system when current thresholds are reached for the input or the output. The system is also placed into an enclosure that prohibits external objects from entering and engaging with the internals of the system.

1.2 Top Level Diagram

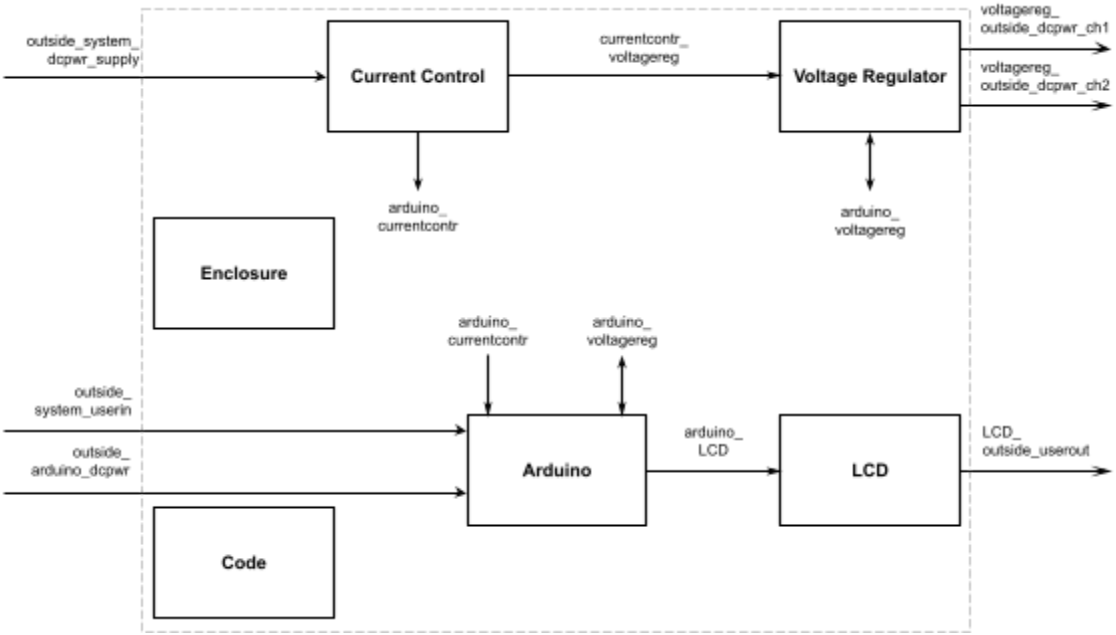


Figure 2: Top Level Diagram

2 Interface Definitions

Interface Name	Interface Type	Specifics
outside_arduino_userin	Code	• V_{max}: 5V • V_{min}: 0V • C language
outside_dcpwr_suppy (dc_power_input)	DC Power	• V_{max}: 28 V • V_{min}: 17 V • I_{peak}: 1 A • $I_{nominal}$: 0.2 A
outside_arduino_dcpwr (usb power)	Data and DC Power	• V_{max}: 5.25V • V_{min}: 4.4V • I_{peak}: 500 mA • $I_{nominal}$: 100 mA • USB-Type A to USB-Type B • Serial communication with Arduino
system_outside_dcpwr (channels 1 and 2)	DC Power	• V_{max}:14V • V_{min}: 2V • I_{peak}: 500 mA • $I_{nominal}$: 100 mA • Steady state DC signal output
outside_arduino_userin (button input inc/dec)	Digital Signal	• Increases output voltage by 0.25 V per press • Decreases output voltage by 0.25 V per press
LCD_outside_userout	Data	• 16x2 displayable characters • 4 bit data transmission • 12 digital pins
outside_user_input	User Digital Input	• V_{max}:14V • V_{min}: 2V • Arduino input from user input to PC to control the output voltage for either channel

Figure 3: Interface Definitions

3 Test Plan

3.1 System Requirements

1. **Customer Requirement: The power supply must have multiple channels.**
   Engineering Requirement: The system will supply at least two independent channels of voltage at least including the range of 2-14V.

2. **Customer Requirement: The power supply needs to be powerful.**
   Engineering Requirement: The system will supply up to 1.5A on each of its channels.

3. **Customer Requirement: The system must be safe.**
   Engineering Requirement: The system will only use US standard plugins for connecting to external devices, will not allow any object with a diameter greater than 1 mm to enter the enclosure, and will be disabled if more than 1A is drawn from the wall power or 1.5A from the output at any time.

4. **Customer Requirement: The system must be accurate.**
   Engineering Requirement: The voltages and currents displayed by the system (to user and reported via serial) must be within 5% or .1V of real values whichever is larger.

5. **Customer Requirement: The power supply needs to be programmable.**
   Engineering Requirement: The system will be configurable for all channels over a serial interface using SCPI protocol that can adjust voltages and current limits.

6. **Customer Requirement: The system must be easily read.**
   Engineering Requirement: The system will display the output voltage and current to the user in a clear manner on a LCD display.

7. **Customer Requirement: The power supply will function at different input DC voltages.**
   Engineering Requirement: The power supply will function with a constant DC input voltage between 17V to 28V.

3.2 System Test Plan

1. Requirement 1 (Video Link)
    a. Set CH1 to min/max voltage and check that it is less than or equal to 2V/14V
    b. Set CH2 to min/max voltage and check that it is less than or equal to 2V/14V
2. Requirement 2 (Video Link)
    a. Cannot pull more than 1.5A from the system
    b. Check if system limits the current or is disabled
3. Requirement 3 (demonstrated in video 2)
    a. Connect load such that more than 1A of current is being pulled for CH1 and CH2
    b. Check if system limits the current
4. Requirement 4 (Video Link)
    a. Vary voltage values between 2 - 14V in increments of 1V
    b. Measure output voltage and check that it is within 5% or 0.1V of expected value
5. Requirement 5 (Video Link)
    a. Use `[SOURce]:VOLTage[:LEVel]` command on CH1 and CH2
    b. Measure the output voltage of each channel and verify its accuracy
6. Requirement 6 (demonstrated in videos 1, 4, 5, 7)
    a. Vary the voltage for both channels while measuring output voltage
    b. Check that LCD displays and updates with correct voltage and current values
7. Requirement 7 (Video Link)
    a. Vary DC input to the system between 17 and 28V while measuring output
    b. Check that output voltage remains within 0.05V of expected value

4 Electrical Schematics

4.1 System Level

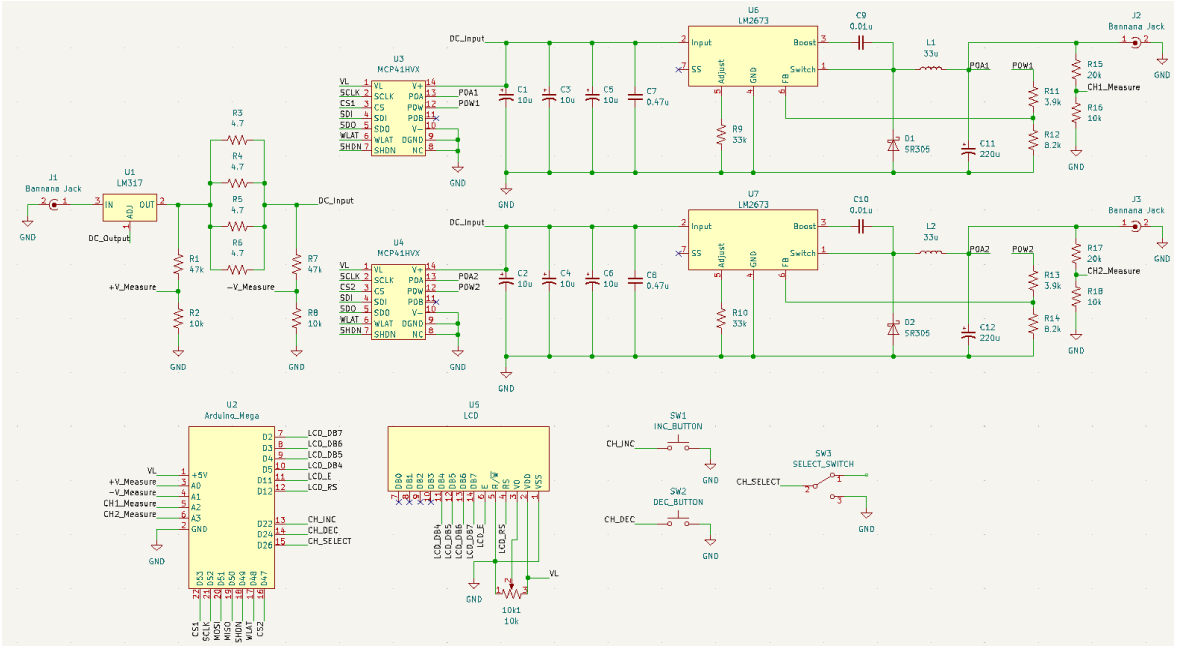


Figure 4: System Level Schematic

4.2 Current Control

This block is a current sensing block that has one input and two outputs. The input is from the DC power supply that is powering the system. The two outputs are a current limited DC power that outputs to the voltage regulator and a digital signal read over the sensing system where the Arduino can receive data and act accordingly to limit the current.

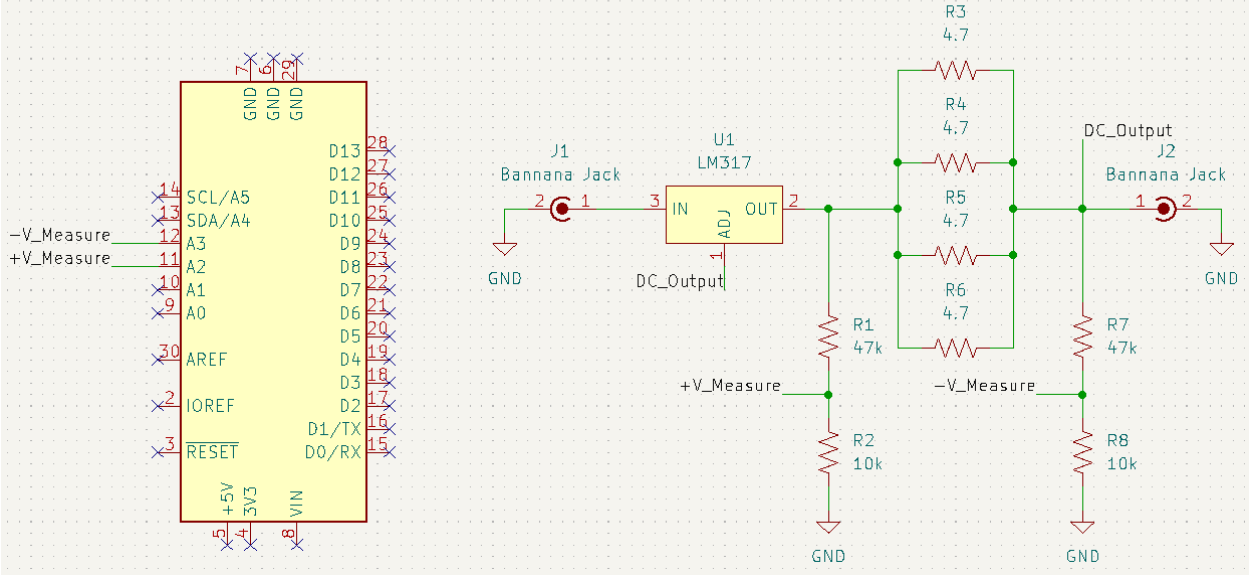


Figure 5: Current Control Circuit

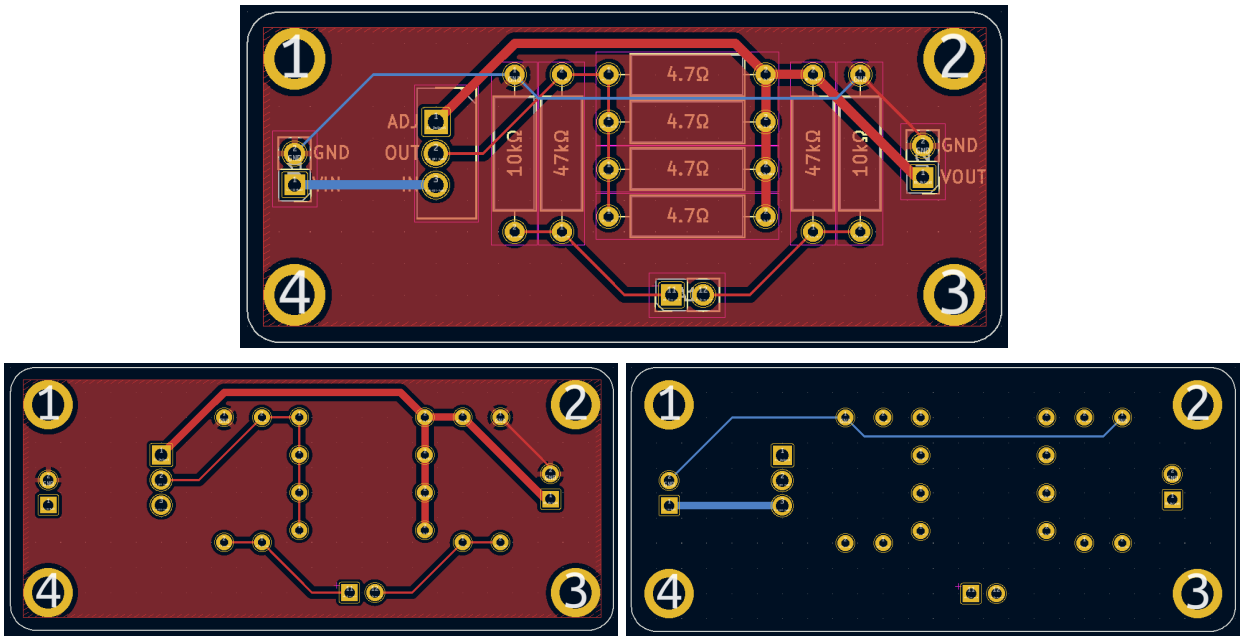


Figure 6: Current Control PCB

4.3 Voltage Regulator

The functionality of the voltage regulator is for the voltage to be adjusted to a certain output voltage regardless of the amount of input voltage as long as the input voltage is greater than 17V. This block uses a digital potentiometer and a switching regulator in order to communicate with Arduino. Communication with the arduino allows for user input to control the output voltage for both of the channels in the system.

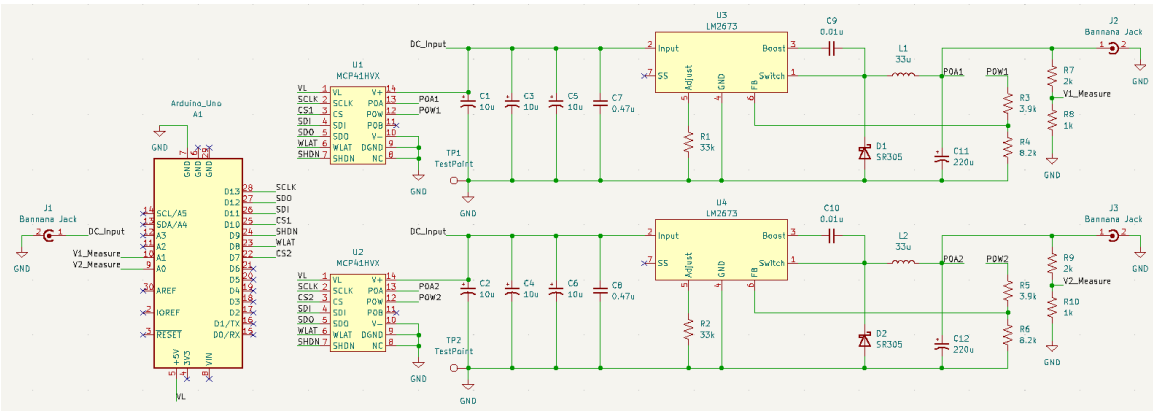
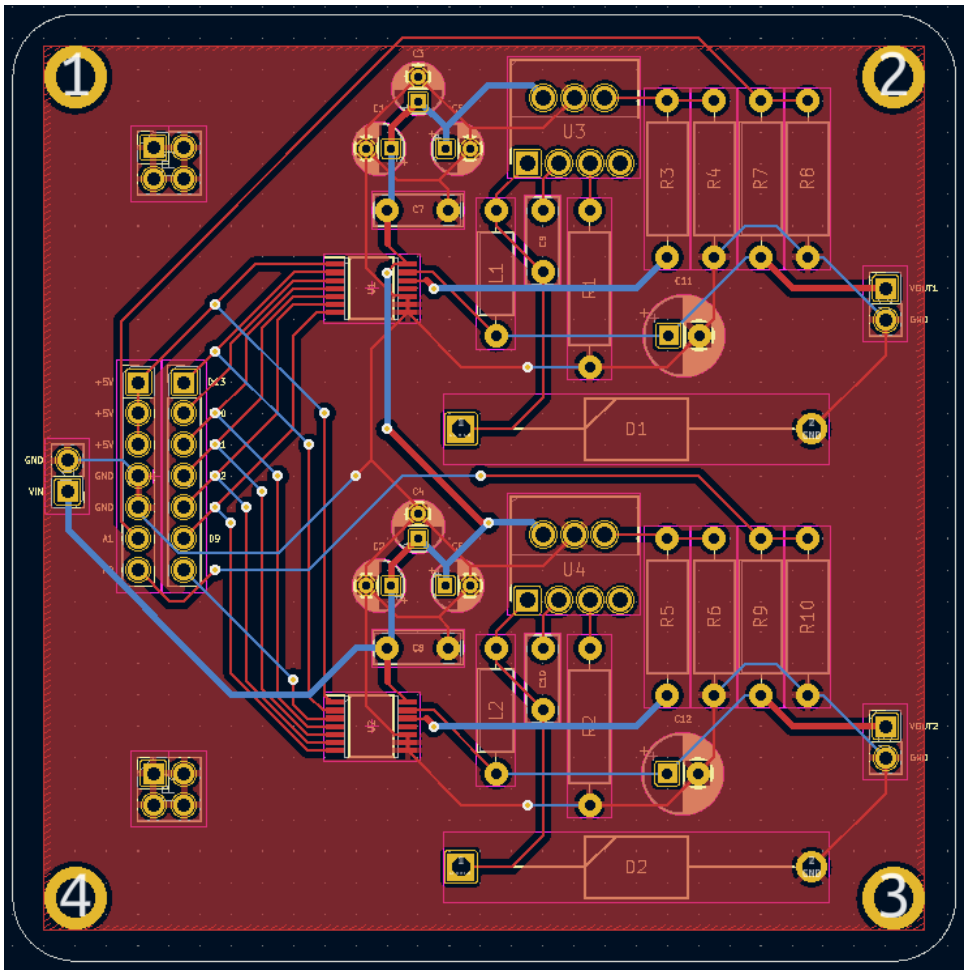


Figure 7: Voltage Regulator Circuit



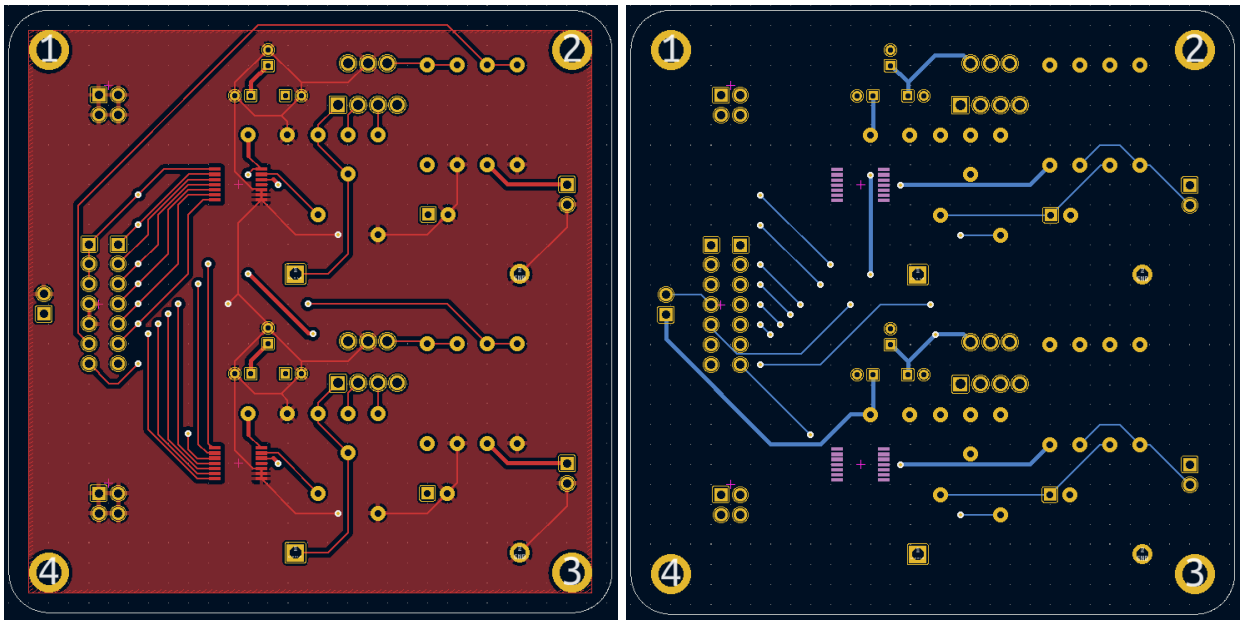


Figure 8: Voltage Regulator PCB

4.4 LCD Display

This block takes the voltages and currents measured by the Arduino to be displayed onto the LCD display. This allows for the user to easily read the measurements of the system. The LCD is a 16x2 character display module and uses four pins to be powered and receive data from the Arduino.

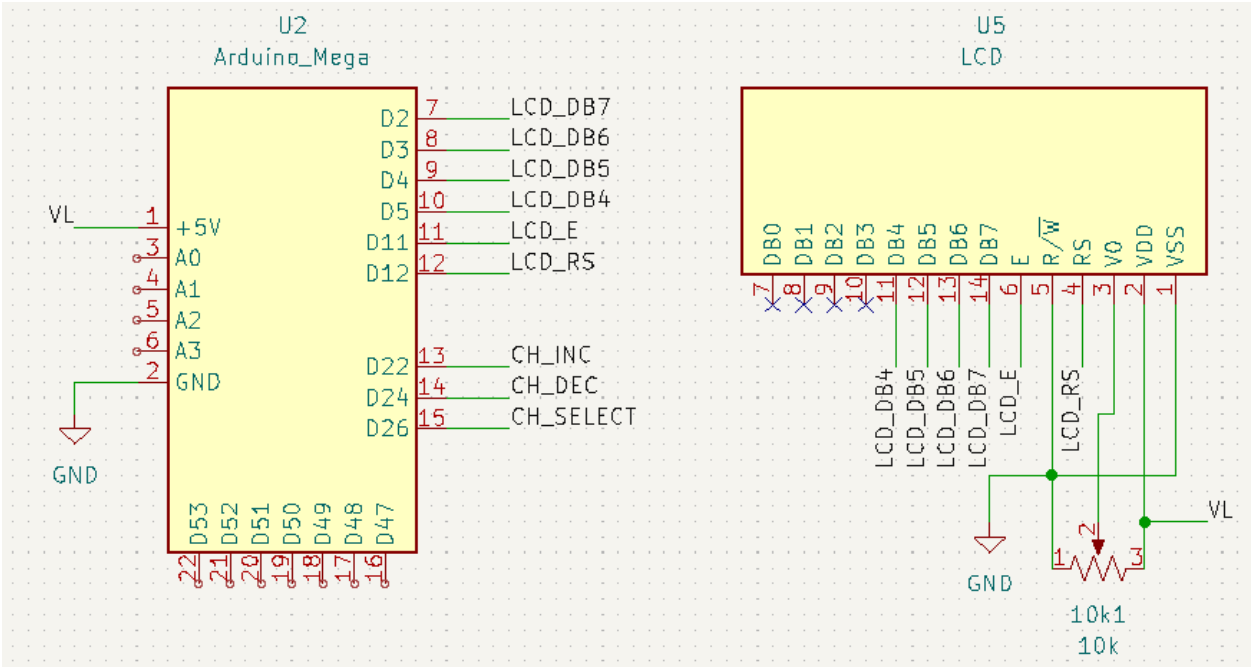


Figure 9: LCD Schematic

5 Circuit Simulations

5.1 Current Control

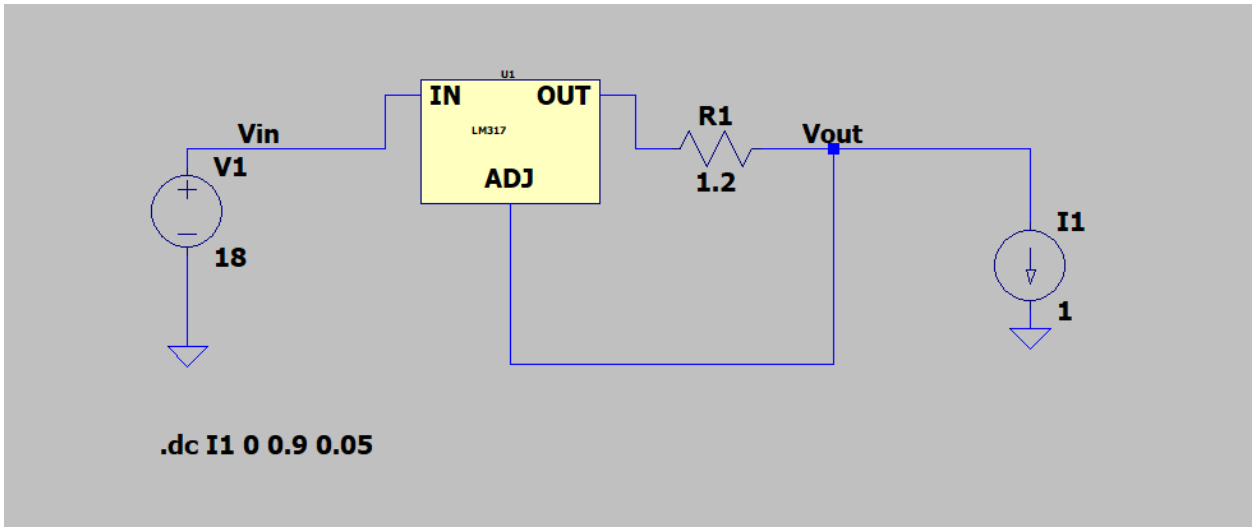


Figure 10: Current Control LTSpice

For the current limit to be set as 1A, according to the equation given by the datasheet, where I_{limit}= R1/1.2, R1 would need to be 1.2Ω. This would mean that when there is 1A going across the R1 resistor, there will be a maximum of a 1.2V voltage drop. To get this measurement onto an Arduino, since the maximum input voltage will be 28V, a voltage divider with a ratio of 6:1 was used to have the maximum voltage reading for one node of the resistor be under 5V. Simulations of the circuit, for the current limiting functionality are shown below.

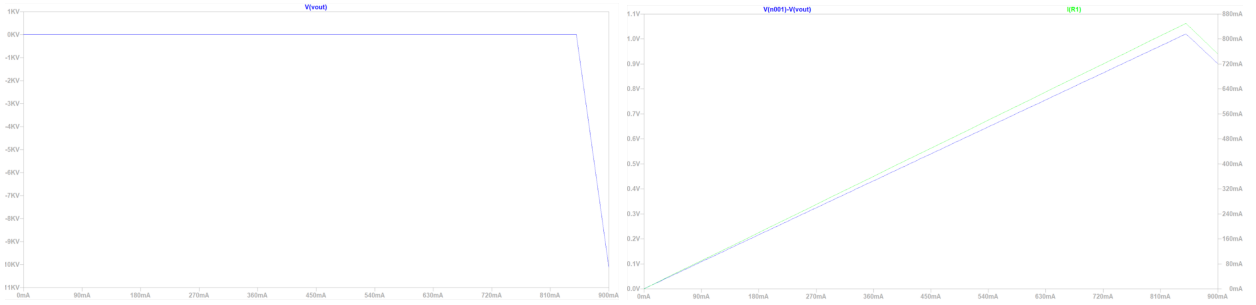


Figure 11: Current Control Simulation

5.2 Voltage Regulator

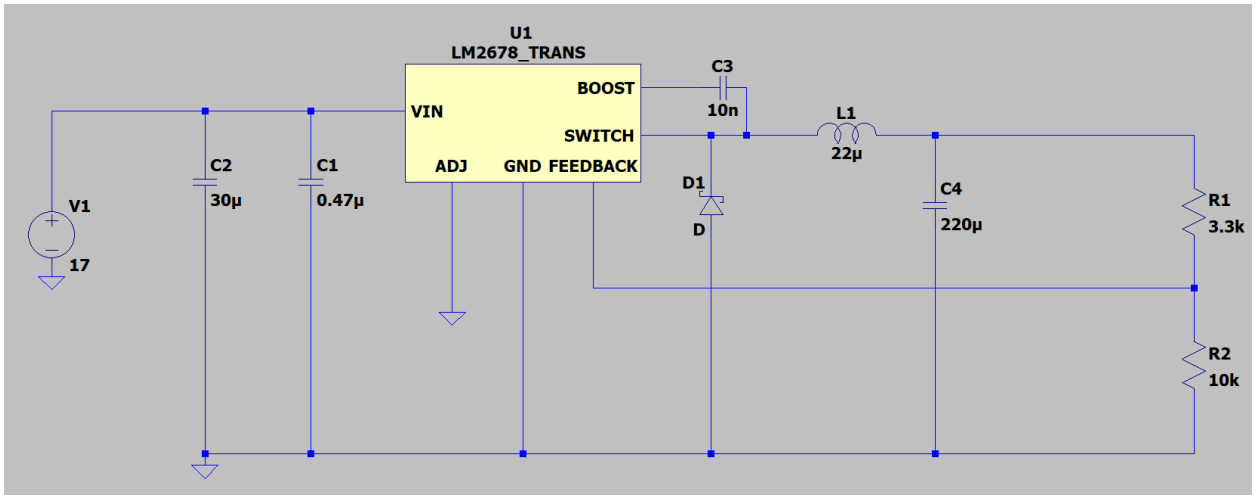


Figure 12: Voltage Regulator LTSpice

For the voltage regulator, a fixed 4V circuit was simulated for stability and heat dissipation. A voltage sweep from 10V to 28V was performed and it can be seen that voltage remains fixed after a 17V input voltage. The maximum power dissipation for the LM2678 is 24mW when input voltage is 28V which is well within below the 1W maximum listed in the datasheet.

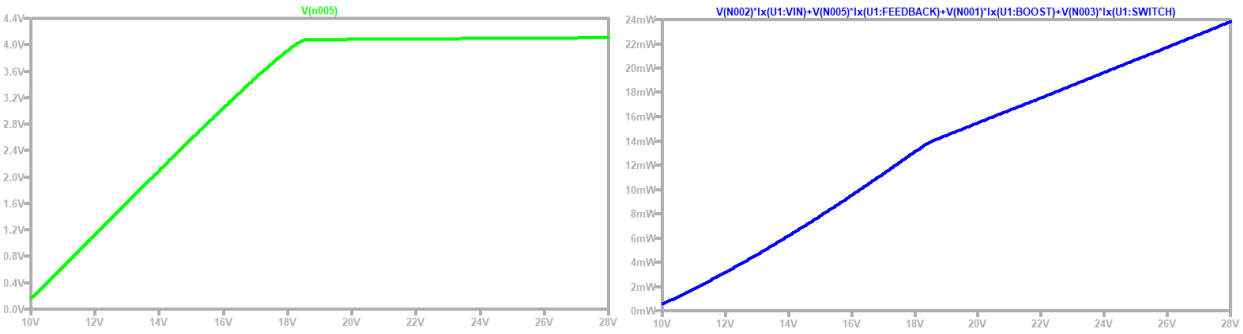


Figure 13: Voltage Regulator Simulation

6 Arduino Mega Pinout Connections

The Arduino Mega was used for the microcontroller of this system for the number of I/O pins it provides. Libraries used included the LiquidCrystal library for the LCD display and the MCP41HVX1 library sourced from GitHub for the digital potentiometer.

6.1 Connections Table

Pin Label	Connection
5V	Digipot VL / LCD V0 and VDD
A0	+V Measure
A1	-V Measure
A2	CH1 Measure
A3	CH2 Measure
D2	LCD DB7
D3	LCD DB6
D4	LCD DB5
D5	LCD DB4
D11	LCD E
D12	LCD RS
D22	Increment Push Button
D24	Decrement Push Button
D26	Channel Select Switch
D47	Digipot CS2
D48	Digipot WLAT
D49	Digipot SHDN
D50	Digipot MISO
D51	Digipot MOSI
D52	Digipot SCLK
D53	Digipot CS1

Figure 14: Pin Connections Table

6.2 Wiring Diagram

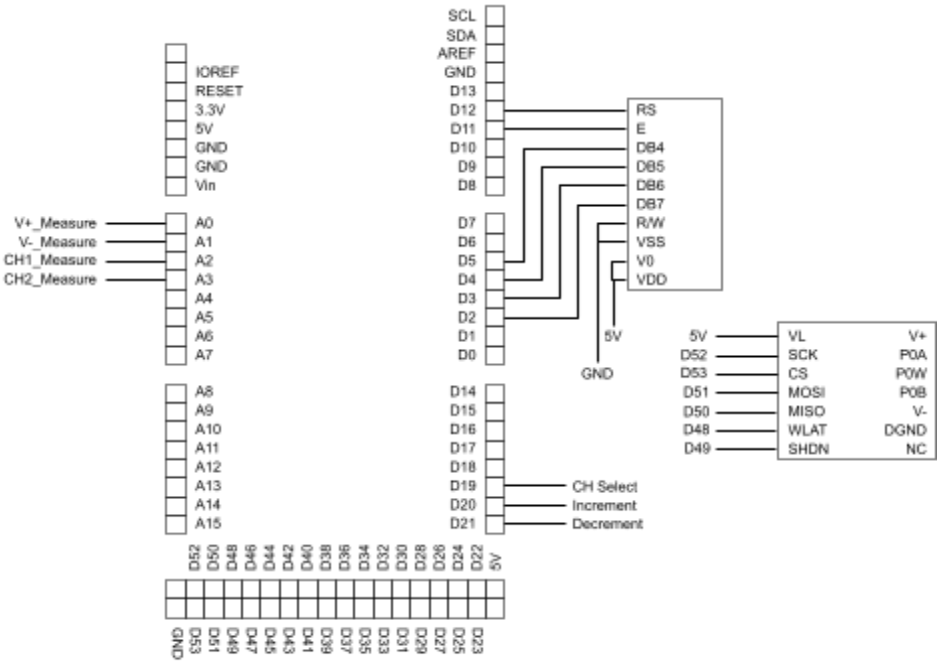
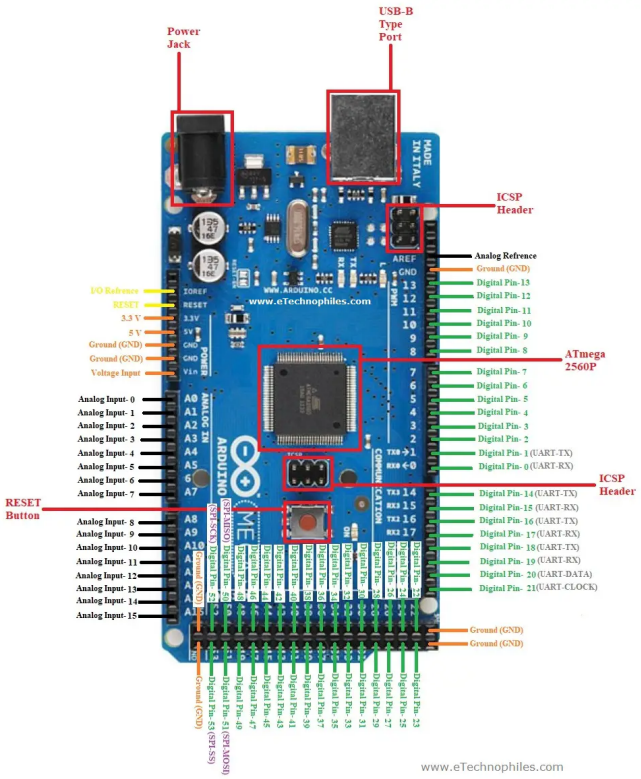


Figure 15: Arduino Mega Wiring Diagram



7 Code

7.1 Setup

```
#include "LiquidCrystal.h"
#include "MCP41HVX1.h"

#define DB7         2
#define DB6         3
#define DB5         4
#define DB4         5
#define EN          11
#define RS          12

#define INC_Button  22
#define DEC_Button  24
#define CH_Switch   26

#define WLAT        48
#define SHDN        49
#define CS1         53
#define CS2         47

LiquidCrystal lcd(RS, EN, DB4, DB5, DB6, DB7);
MCP41HVX1 Digipot1(CS1, SHDN, WLAT);
MCP41HVX1 Digipot2(CS2, SHDN, WLAT);

int CH1_Wiper = 0;
int CH2_Wiper = 0;
int CH_Select = 1;
int CH_SCPI;
String input;
char* token;
int isSCPI;
int isInt;

int CH1_Read;
int CH2_Read;
float CH1_Volt;
float CH2_Volt;
float current;

int INC_Curr;
int INC_Prev;
int DEC_Curr;
int DEC_Prev;

void setup(){
  Serial.begin(9600);
  lcd.begin(16, 2);

  //Initialize digital potentiometer 1
  Serial.print("CH1 Start = ");
  Serial.println(Digipot1.WiperGetPosition());
  Serial.print("CH1 Set = ");
  Serial.println(Digipot1.WiperSetPosition(127));
  //Initialize digital potentiometer 2
  Serial.print("CH2 Start = ");
```

```
  Serial.println(Digipot2.WiperGetPosition());
  Serial.print("CH2 Set = ");
  Serial.println(Digipot2.WiperSetPosition(127));

  //initialize inc and dec buttons
  pinMode(INC_Button, INPUT_PULLUP);
  INC_Prev = digitalRead(INC_Button);
  pinMode(DEC_Button, INPUT_PULLUP);
  DEC_Prev = digitalRead(DEC_Button);

  //initialize CH selection switch
  pinMode(CH_Switch, INPUT);
  pinMode(28, INPUT_PULLUP);
}
```

Figure 16: Setup Code

The two libraries used for the code are “LiquidCrystal.h” for the display and “MCP41HVX1.h” for the digital potentiometer. Pin definitions for the LCD, front panel buttons, and digital potentiometer are all listed respectively. After assigning the LCD and digital potentiometer with their respective pins, all variables are declared. In the setup function, serial communication is initialized with baud rate of 9600, LCD is initialized, digital potentiometer wiper positions are set to 127 so that channels are outputting the minimum value, and pins for the front panel buttons/switch are configured.

7.2 Serial Control

```
  //read and trim from serial input
  input = Serial.readString();
  input.trim();

  //copy input into c string
  char temp[input.length() + 1];
  strcpy(temp, input.c_str());

  //change to corresponding CH if input was string
  if(input.toInt() == 0){
    if(input == "CH1"){
      CH_Select = 1;
    }else if(input == "CH2"){
      CH_Select = 2;
    }else{
      while(1){
        //check if input contains exactly 2 colons
        isSCPI = 0;
        for(int i = 0; i < strlen(temp); i++){
          if(temp[i] == ':'){
            isSCPI++;
          }
        }

        //break if there aren't exactly 2 colons in input
        if(isSCPI != 2){
          break;
        }
```

```
      //get and assign first token for source
      token = strtok(temp, ":");
      if(strcmp(token, "CH1") == 0){
        CH_SCPI = 1;
      }else if(strcmp(token, "CH2") == 0){
        CH_SCPI = 2;
      }else{
        break;
      }

      //check if second token is valid
      token = strtok(NULL, ":");
      if(strcmp(token, "VOLTAGE") != 0 && strcmp(token, "VOLT") != 0){
        break;
      }

      //check if third token is an int value
      token = strtok(NULL, ":");
      for(int i = 0; i < strlen(token); i++){
        if(isdigit(token[i])){
          isInt = 1;
        }else{
          isInt = 0;
          break;
        }
      }

      if(isInt == 1){
        //check if value is within possible voltage range
        if(atof(token) >= 2 && atof(token) <= 14){
          if(CH_SCPI == 1){
            CH1_Wiper = (17811 / 125) - (9 * atof(token)) + 2;
            Digipot1.WiperSetPosition(CH1_Wiper);
            //calbiration for higher voltages
            if(atof(token) >= 10){
              Digipot1.WiperSetPosition(CH1_Wiper - 2);
            }
          }else if(CH_SCPI == 2){
            CH2_Wiper = (17811 / 125) - (9 * atof(token)) + 2;
            Digipot2.WiperSetPosition(CH2_Wiper);
            //calibration for high voltages
            if(atof(token) >= 10){
              Digipot2.WiperSetPosition(CH2_Wiper - 2);
            }
          }
        }
      }
      break;
    }
  }
//change pot value for corresponding CH if input was int between 1 and 127
}else{
  if(input.toInt() != 0 && CH_Select == 1){
    if(input.toInt() <= 127 && input.toInt() >= 0){
      CH1_Wiper = Digipot1.WiperSetPosition(input.toInt());
    }
    Serial.print("CH1 Position = ");
```

```
      Serial.println(CH1_Wiper);

    }else if(input.toInt() != 0 && CH_Select == 2){
       if(input.toInt() <= 127 && input.toInt() >= 0){
        CH2_Wiper = Digipot2.WiperSetPosition(input.toInt());
      }
      Serial.print("CH2 Position = ");
      Serial.println(CH2_Wiper);
    }
  }
```

Figure 17: Serial Control Code

In order to control the system through the serial terminal, a string user input is taken in. The string is then trimmed to remove all white space at the end. The string is then converted into a c string variable so that it can be operated on by certain functions.

First, the c string is then checked to see if the input was an integer value. If the input was an integer, then the entered value will change the wiper position of the corresponding channel being interfaced with. The channel can be changed by inputting “CH1” for channel 1 and “CH2” for channel 2.

Lastly, if the input was a SCPI command following the format “`[SOURce]:VOLTage[:LEVel]`”, then the voltage value will be set at the corresponding “level” for the corresponding “source”. To check if the input follows this format, the number of colons in the inputted string is set by looping through every character in the c string. If there are exactly two colons then the code will tokenize the input string. If the first token is equal to “CH1” or “CH2” then the corresponding channel will be stored into a CH_SCPI value. The next token simply checks if it is equal to the long and short form values “VOLTAGE” or “VOLT” respectively. The last token is then checked if it is an integer between the minimum and maximum voltage values (2 to 14V). If this condition is met then the corresponding wiper value will be calculated and set to output the desired voltage.

7.3 Front Panel Control

```
  //read state for inc and dec buttons
  INC_Curr = digitalRead(INC_Button);
  DEC_Curr = digitalRead(DEC_Button);

  //decrease voltage by about 0.2 for corresponding CH if button pressed
  if(INC_Prev != INC_Curr){
    if(digitalRead(CH_Switch) == LOW && Digipot1.WiperGetPosition() <= 125){
      CH1_Wiper = Digipot1.WiperSetPosition(Digipot1.WiperGetPosition() + 2);
      Serial.println("INC CH1");
    }else if(digitalRead(CH_Switch) == HIGH && Digipot2.WiperGetPosition() <=
125){
      CH2_Wiper = Digipot2.WiperSetPosition(Digipot2.WiperGetPosition() + 2);
      Serial.println("INC CH2");
    }
  }
```

```
  //increase voltage by about 0.2 for corresponding CH if button pressed
  if(DEC_Prev != DEC_Curr){
    if(digitalRead(CH_Switch) == LOW && Digipot1.WiperGetPosition() >= 2){
      CH1_Wiper = Digipot1.WiperSetPosition(Digipot1.WiperGetPosition() - 2);
      Serial.println("DEC CH1");
    }else if(digitalRead(CH_Switch) == HIGH && Digipot2.WiperGetPosition() >=
2){
      CH2_Wiper = Digipot2.WiperSetPosition(Digipot2.WiperGetPosition() - 2);
      Serial.println("DEC CH2");
    }
  }
```

Figure 18: Front Panel Control Code

For the front panel control, the increment and decrement buttons states are continuously checked. If the state is changed from its previous state, then the corresponding channel wiper position will either be incremented or decremented by 2 which is about 0.2V. In order to know which channel is being operated on, the state of an SPDT switch connected to 5V and GND is continuously read. If the switch pulls the digital read high and low then channel 1 will be operated on and if the digital read is pulled high then channel 2 will be operated on.

7.4 Voltage/Current Display

```
  //Find voltage of CH1
  CH1_Read = analogRead(A2);
  CH1_Volt = CH1_Read * (5 / 1023.0) * 3.28;
  //Find voltage of CH2
  CH2_Read = analogRead(A3);
  CH2_Volt = CH2_Read * (5 / 1023.0) * 3.28;

  //Find current through system
  current = (analogRead(A0) - analogRead(A1)) * (5 / 1023.0) * 5.7;

  //CH1 Voltage
  lcd.setCursor(0, 0);
  lcd.print("C1:");
  lcd.print(CH1_Volt);
  Serial.print("CH1:");
  Serial.println(CH1_Volt);
  lcd.print("V ");

  //CH2 Voltage
  lcd.setCursor(0, 1);
  lcd.print("C2:");
  lcd.print(CH2_Volt);
  lcd.print("V ");
  Serial.print("CH2:");
  Serial.println(CH2_Volt);

  //System Current
  lcd.setCursor(11, 0);
  lcd.print("CURR");
  lcd.setCursor(11, 1);
  lcd.print(current);
  lcd.print("A ");
```

Figure 19: Voltage/Current Display Code

The voltage of each channel is calculated using analog read at a voltage divided point from the output. This value is then used to calculate the voltage at the output. Through testing various different voltage values, the 3.28 multiplier is an adjusted value that calibrates and accounts for the error in resistor values. Current through the system is calculated in the same way by measuring voltage at two ends of a resistor. These values are then printed on the LCD which constantly updates the values about every 50ms.

8 Enclosure

The enclosure was designed to fit the components being used for the DC Power Supply System and 3D printed. The components included the two PCB modules for the current control and the voltage regulator and the Arduino Mega. The enclosure has cut outs for the external inputs and outputs for the system. There are six banana plug cutouts–two for each output channel for a total of four and two for the DC power that is powering the system. There are also notches for the button and switch interface for adjusting the voltages mechanically rather than with the PC. Finally, there is another cutout for the USB power supplying the Arduino Mega. Each portion of the enclosure, as can be seen below, was modeled and printed separately, in order to prevent massive reprints in case the 3D print would mess up while printing.

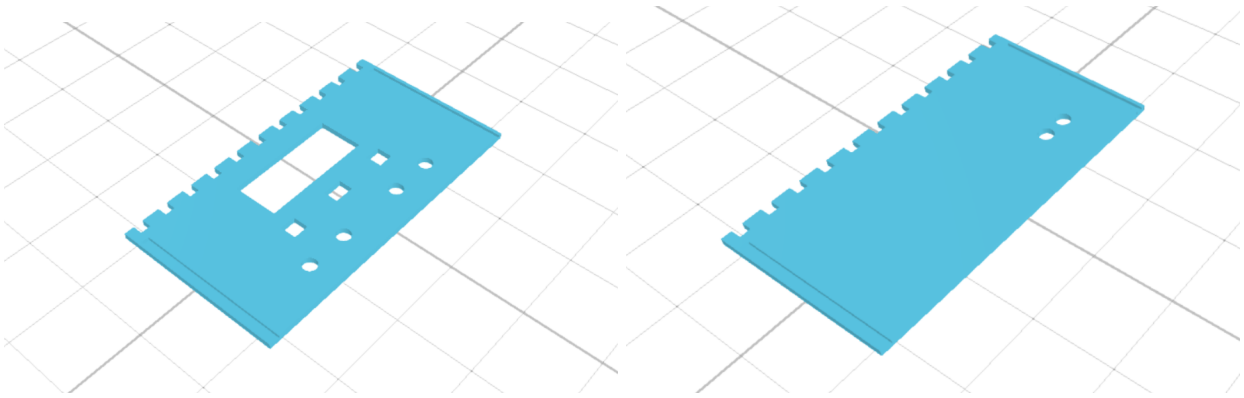


Figure 20: Enclosure Front/Back Wall

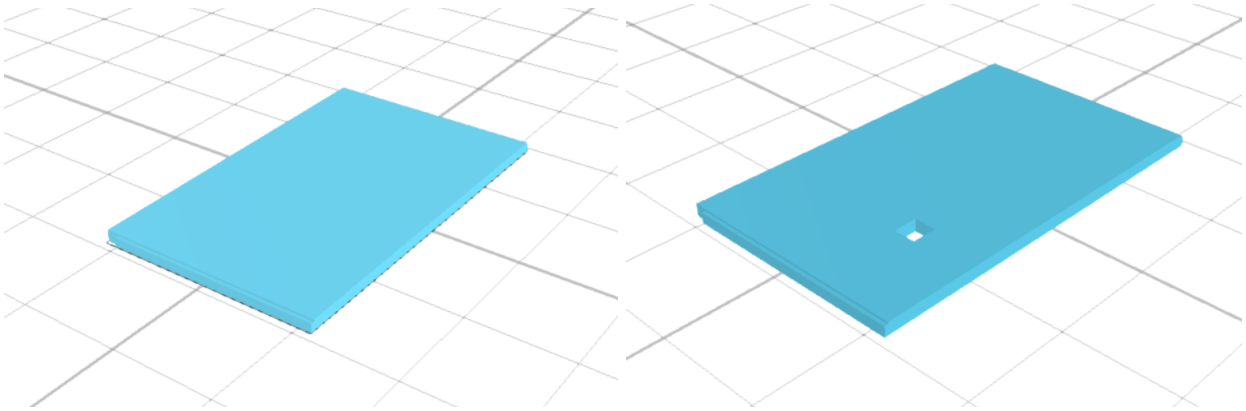


Figure 21: Enclosure Side Walls

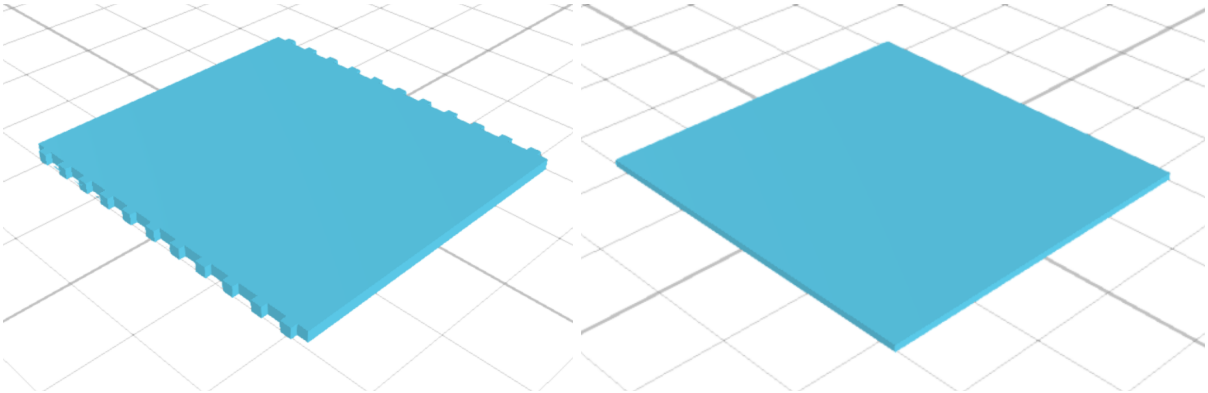


Figure 22: Enclosure Top/Bottom

Front Wall of Enclosure (220 x 120 x 4 mm)
Back Wall of Enclosure (220 x 120 x 4 mm)
Side Walls of Enclosure (192 x 113 x 6 mm)
Top of Enclosure (220 x 208 x 7 mm)
Bottom of Enclosure (221 x 202 x 4 mm)

9 Bill of Materials

https://docs.google.com/spreadsheets/d/1T_7y_Y9acUjhFWEUXpAy8GyFIrlvGijIzpG1UVsA_ew/edit#gid=0

10 Time Report

https://docs.google.com/spreadsheets/d/1OE9pyGqfvoHeF8KxJliZF-Ln-f5ST6dKCXfUV1o3j6o/edit#gid=1115838130

11 References

Current Control

LM317 3-Terminal Adjustable Regulator

- Datasheet

Voltage Regulator

LM2673 Simple Switcher 3-A Step-Down Voltage Regulator

- Datasheet

MCP41HVX1

- Datasheet
- Arduino Library: https://github.com/gregsrabian/MCP41HVX1

LCD

- Datasheet

SCPI Parser

- Arduino Library: https://github.com/Vrekrer/Vrekrer_scpi_parser