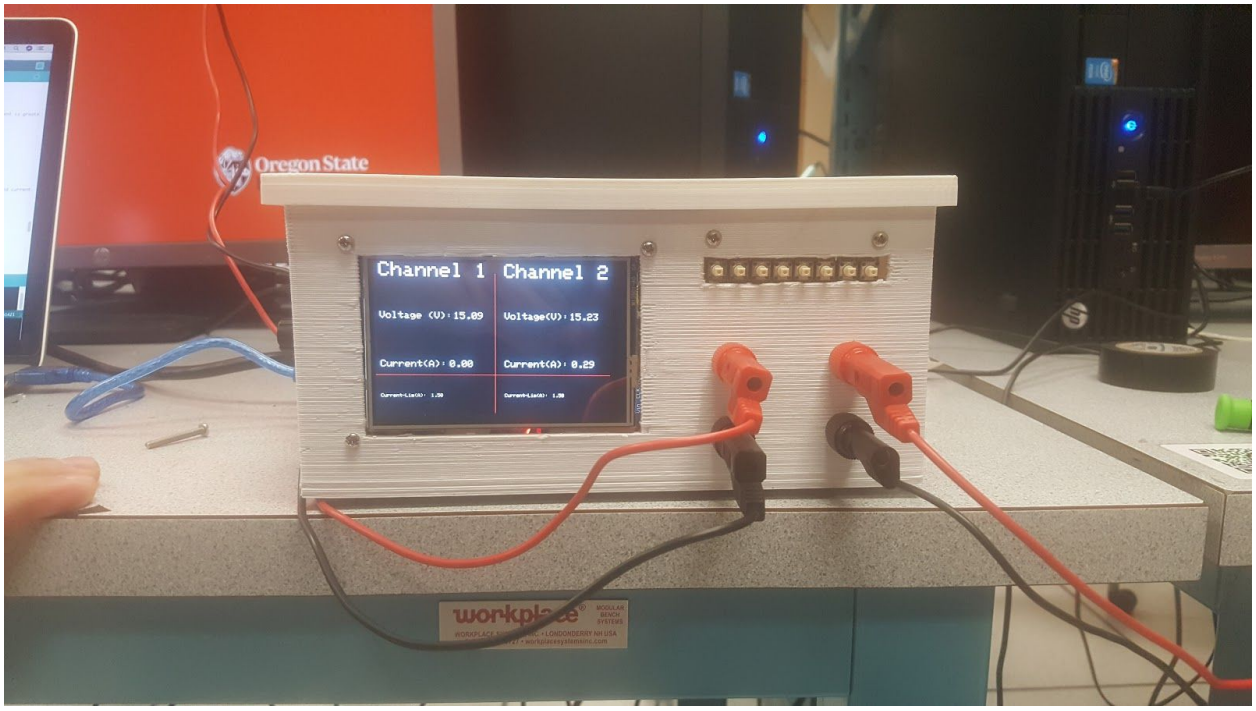

PC Controlled Power Supply

Group 6

William Ratnour, Victor Garcia Flores, Chase Pixley, and Zeyu Dai

Table of Contents

Block Diagram

Figure 1 below depicts the flow of the system when operating. This project consists of one sensing block that measures voltage and current across two channels from the step-down converters. We chose to use an Arduino Mega because it provided enough analog pins needed to run the system.

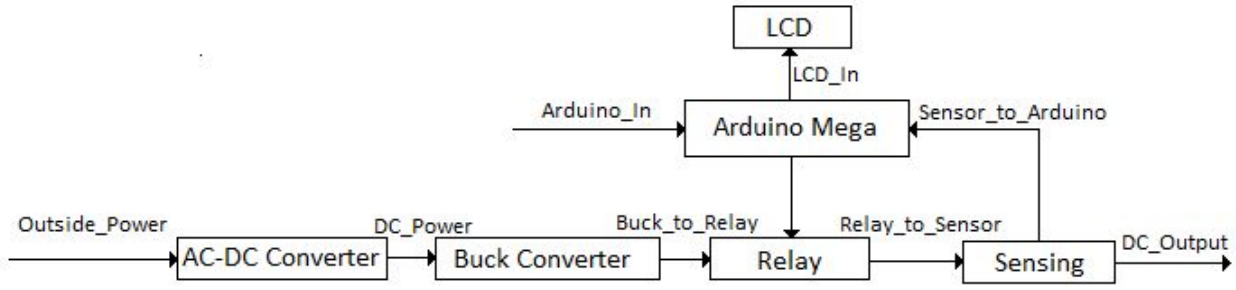

Figure 1: Block Diagram

Interface Properties

Interface	Properties
Outside_Power	• 120Vac • NEMA-15M Connector • I_{Peak}: 5A
DC_Power	• 16Vdc output • Supply up to 3.5A
LCD_In	• 5V • LCD will display voltage and current values • The value must within 5% or 0.1V of real values.
Sensor_to_Arduino	• Analog_input
Sensor_to_Relay	• Max current = 3.5A • Voltage = 16V
Arduino_In	• 7 - 12V • USB serial • Set current limit
Arduino_Out	• Digital Output
Output_Control	• Digital Output

Relay_Out	• Max current = 1.5A • Voltage = 16V
DC_Output	• Voltage Ripple < 0.25VP-P • Max Current = 1.5A • Output Voltage = 2-14V • Discharges To < 2V in under 5 Sec

Outermost Mechanical Drawing

Figure 2 is a side view of the enclosure used for the power supply. The 49 mm by 57 mm square was used to make room for the laptop socket; this was secured using screws. The purpose behind the 10 mm by 10 mm square is for the USB cable that connects to the Arduino. These two squares allow easy removal of all cables that go to the encasing. The two holes on the top right are where the analog potentiometers that connect to the step-down converters are placed.

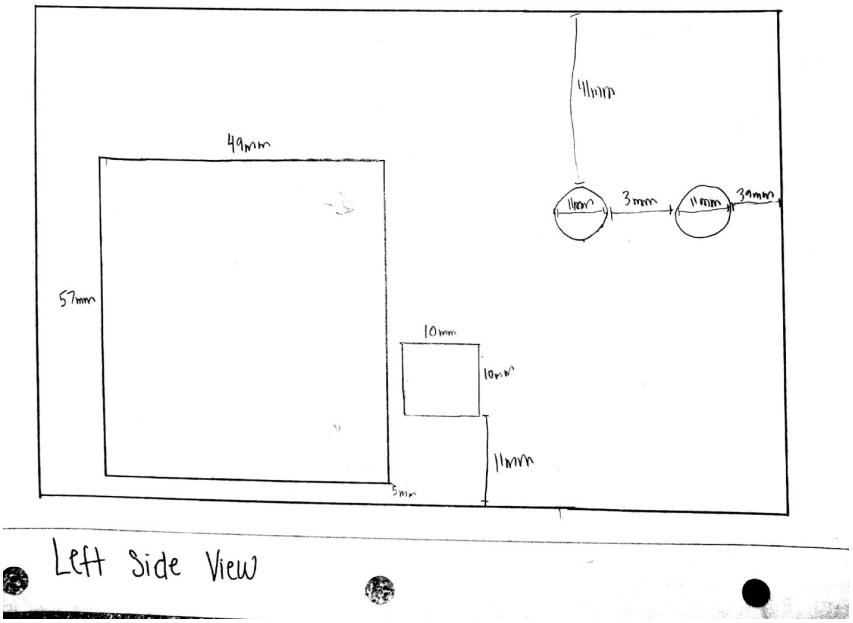

Figure 2: Side View of Enclosure

Figure 3 depicts the front view of the encasing. The 87 mm by 58 mm square is where the Adafruit LCD display is mounted, the 58 mm by 10 mm square is meant for the button board, and the holes are for the female sockets that connet to the load.

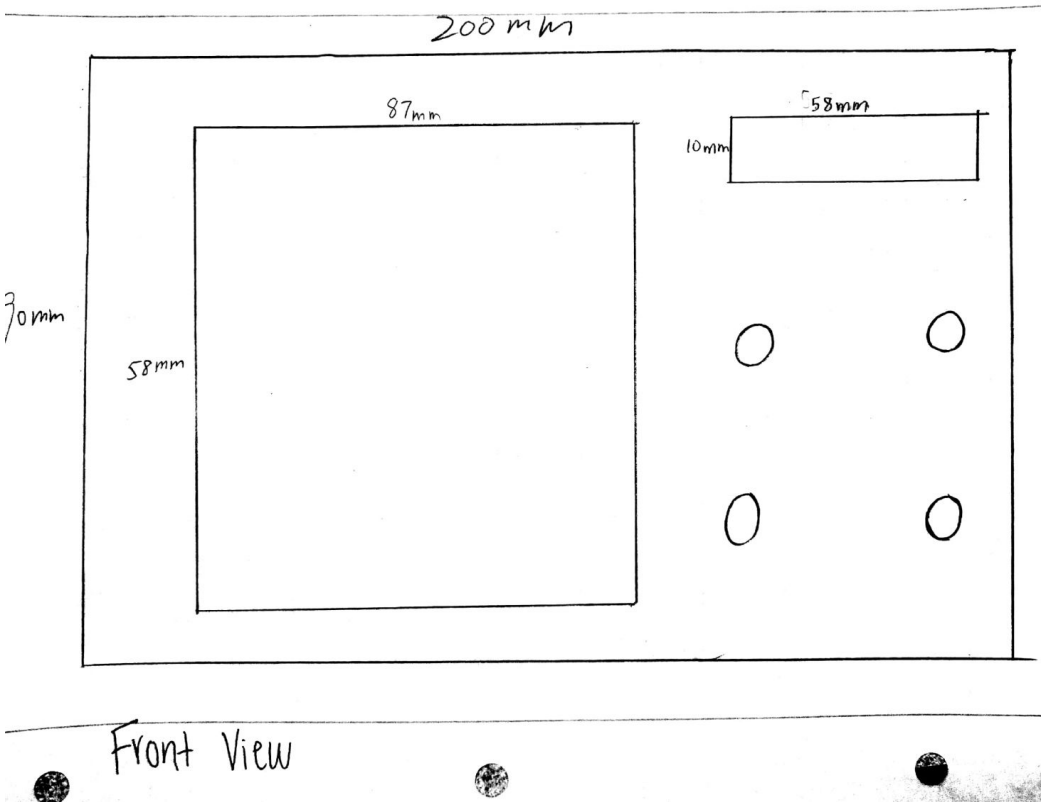

Figure 3: Front View of Encasing

Innermost Mechanical Drawing

Figure 4, the image depicted below, demonstrates the layout for the components inside the encasing. As one can see, the order that the components are laid out follows the order of the process that leads to the output barrels. For example, the AC to DC conversion is placed at the very back, after which the PCB and Arduino were placed, followed by the side encasings that hold the relays and the step-down (“buck”) converters ath the front.

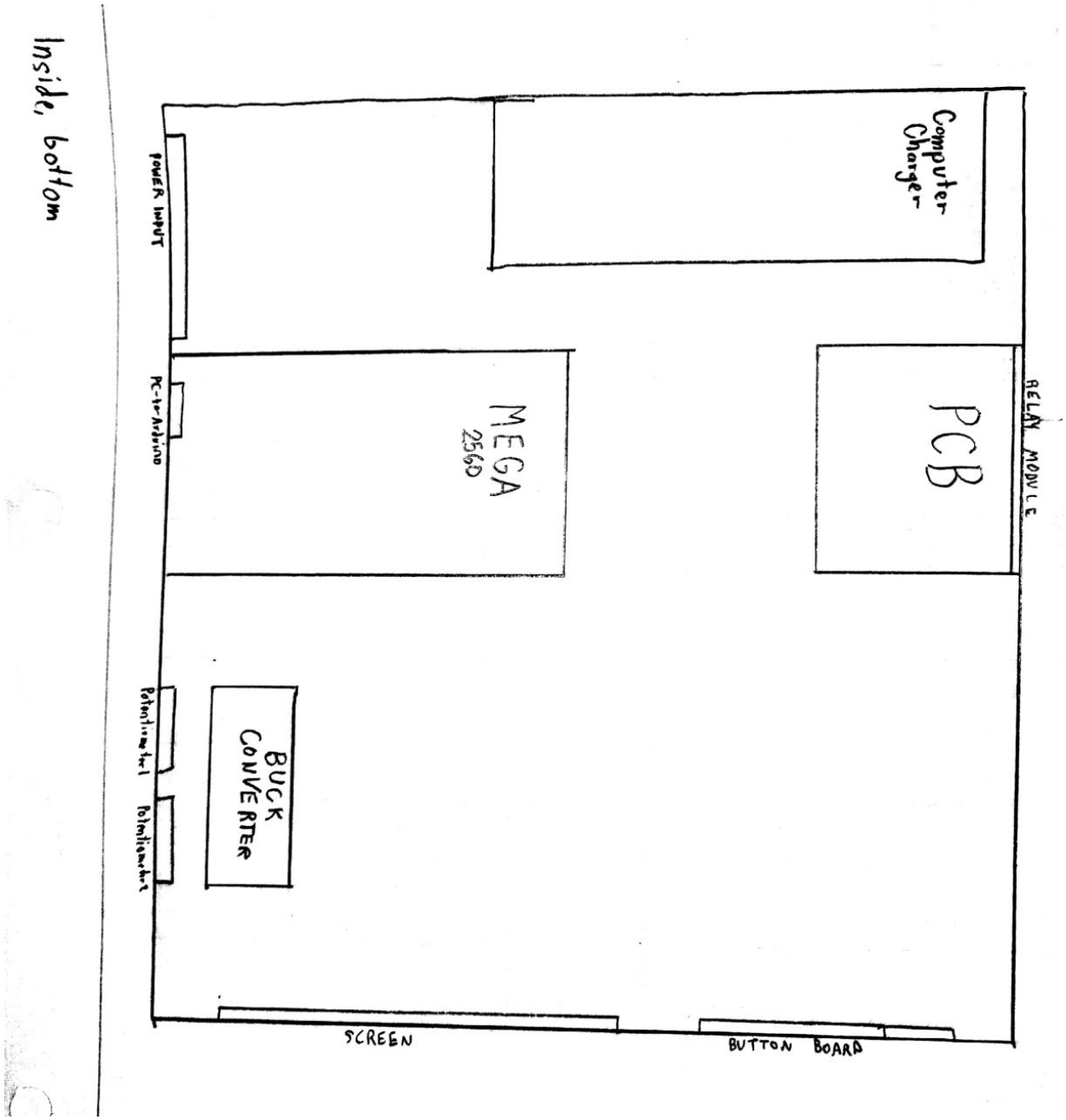

Figure 4: Layout of Current Components Inside Encasing

System Schematic

The system schematic for the whole system is depicted below, as Figure 5:

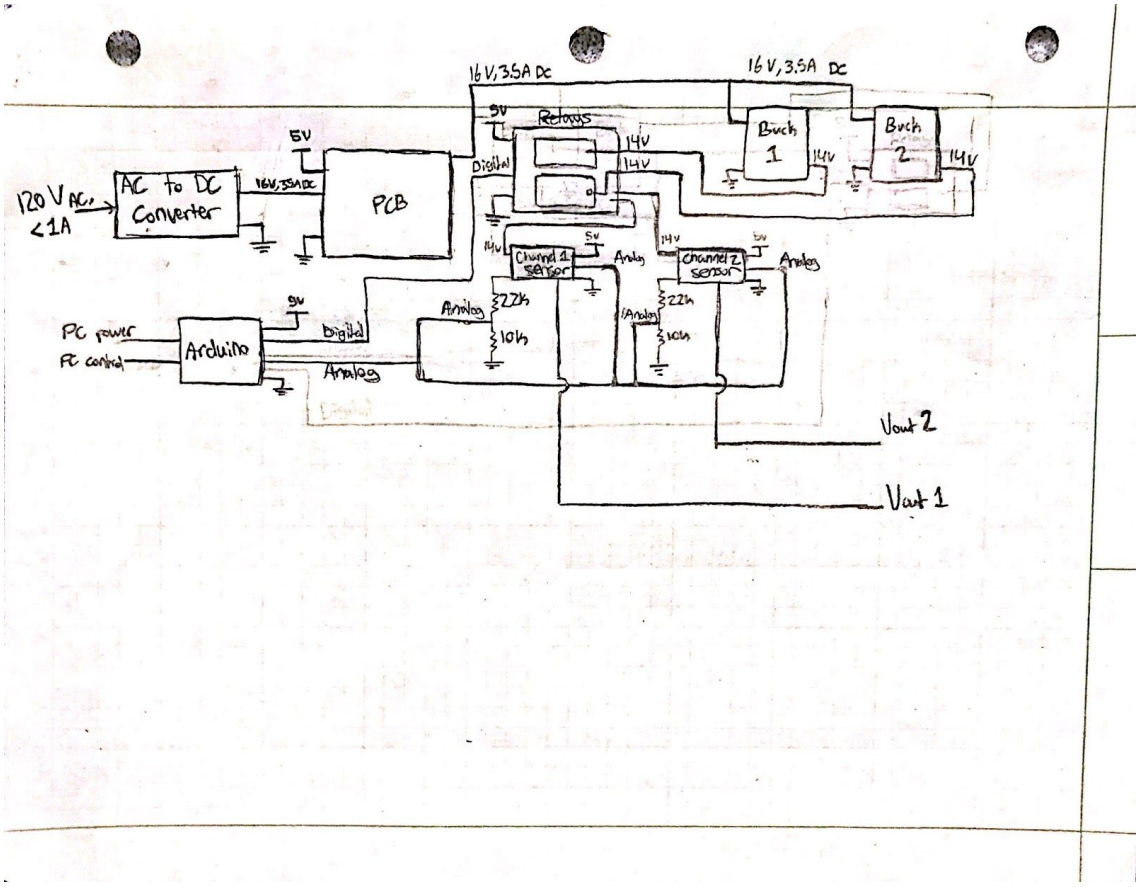

Figure 5: System Schematic

PCB Design

The designed PCB, shown below as Figure 6,

Add Headings (Format > Paragraph styles) and they will appear in your table of contents.

was created to fulfill various purposes: to serve as a common ground source, a 5 V source, a rectified DC voltage connector, and to perform voltage and current sensing. The longest rail, RAIL1, is where a female header rail was placed and RAIL2 is where the 5 V distribution rail was placed. An 8 pin rail was used because there were various components that needed 5 V supplied such as the relays, LCD display, button board, and sensors. Other female rails were used so that the Hall Effect sensors (ACS 712) could mount directly onto the PCB. As one can see, room for resistors was made; these are used to construct a voltage divider that measures voltage.

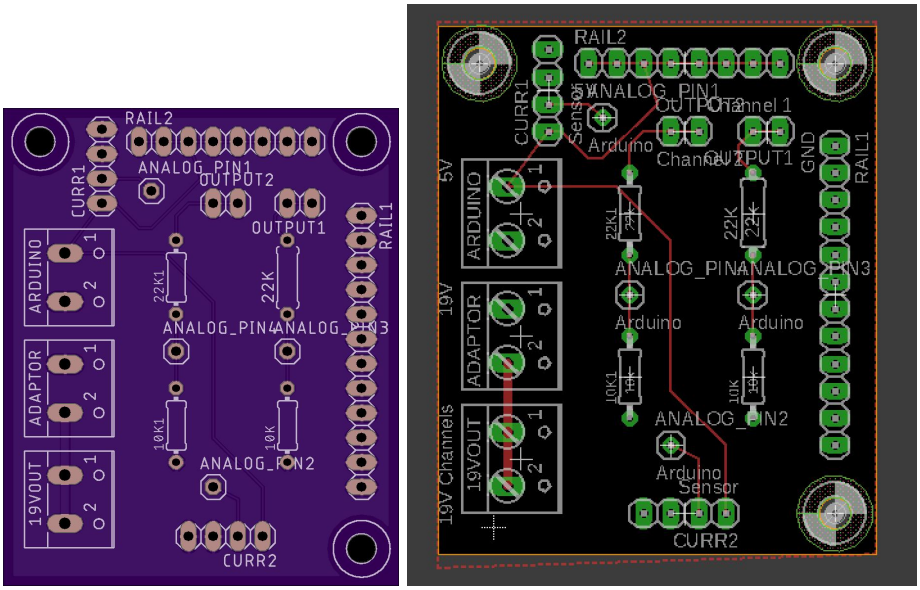

Figure 6: Designed PCB

Video Verification

Base Engineering Requirement Verification

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.
   https://drive.google.com/drive/u/0/folders/1UvIl32TvtCo7hoDOlJhqZo6gk7PbMDE_

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.
   https://drive.google.com/drive/u/0/folders/1UvIl32TvtCo7hoDOlJhqZo6gk7PbMDE_

   *Note that this section has two videos

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 1mm to enter the enclosure, and will be disable if more than 1A is drawn from the wall power or 1.5A from the output at anytime.
   https://drive.google.com/drive/u/0/folders/1UvIl32TvtCo7hoDOlJhqZo6gk7PbMDE_

4. **Customer Requirement:** The system must be reliable
   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.
   https://drive.google.com/drive/u/0/folders/1UvIl32TvtCo7hoDOlJhqZo6gk7PbMDE_

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.
   https://drive.google.com/drive/u/0/folders/1UvIl32TvtCo7hoDOlJhqZo6gk7PbMDE_

Unique Engineering Requirement Verification

1. **Customer Requirement:** The system should be compact.

   Engineering Requirement: The box in which the system is contained should be at most 12” x 8.5” x 5”.

   https://drive.google.com/drive/u/0/folders/1UvIl32TvtCo7hoDOlJhqZo6gk7PbMDE_

2. **Customer Requirement:** The channels should share most of the same system.

   Engineering Requirement: The two channels should have the same transformer, rectifier, filter, regulator, and should split at the control unit.

   https://drive.google.com/drive/u/0/folders/1UvIl32TvtCo7hoDOlJhqZo6gk7PbMDE_

System C++ Code

```
#include <SPI.h>
#include "Adafruit_GFX.h"
#include "Adafruit_HX8357.h"

// ------------------ Button Board --------------------
int VUpPin = 0;             //The pin the VUp button is connected to
int VDownPin = 1;           //The pin the VDown button is connected to
int AUpPin = 2;             //The pin the AUp button is connected to
int ADownPin = 3;           //The pin the ADown button is connected to
int outputPin = 4;          //The pin the change channel button is connected to

int VUp;                    //The state of the VUp button
int VDown;                  //The state of the VDown button
int AUp;                    //The state of the AUp button
int ADown;                  //The state of the ADown button
int pressTime = 0;          //How long this button has been held down for
int oput;                   //The state of the change channel button
int whichOutput;            //Which channel (1/2) is being controlled

float amp1 = 1.50;          //Starting current is 1.5A
float amp2 = 1.50;
float volt1 = 8.50;         //Starting voltage is 8.5V
float volt2 = 8.50;
float interval = 0.01;      //How much the current and voltage change by each cycle
// ----------------------------------------------

// ------------------ Relays --------------------
int relayPin1 = 7;
int relayPin2 = 8;
// ----------------------------------------------

// ------------------ Computer Control --------------------
String computerString;//entire string
float dataVal; //Converted float value
// ----------------------------------------------


// ------------------ Voltmeter --------------------
#define NUMBER_SAMPLES 10

int sum = 0;// sum of samples taken
unsigned char sample_count = 0;// current sample number
float voltage = 0;

float voltage1 = 0.0;// calculated voltage channel 1
```

```
int PinChannel1 = A0;//Pin for the first channel

float voltage2 = 0.0;// calculated voltage channel 2
int PinChannel2 = A1;//Pin for the second channel

// -------------------------------------------------

// ------------------ Ammeter --------------------
float average_voltage = 0;
float Current = 0;
int Current_Pin1 = A2; //Channel 1 Pin
int Current_Pin2 = A3; //Channel 2 Pin
float Voltage = 0;
float Current1; //Channel 1 Current
float Current2; //Channel 2 Current
// -------------------------------------------------


// These are 'flexible' lines that can be changed
#define TFT_CS 10
#define TFT_DC 9
#define TFT_RST 8 // RST can be set to -1 if you tie it to Arduino's reset

// Use hardware SPI (on Uno, #13, #12, #11) and the above for CS/DC
Adafruit_HX8357 tft = Adafruit_HX8357(TFT_CS, TFT_DC, TFT_RST);

// SoftSPI - note that on some processors this might be *faster* than hardware SPI!
//Adafruit_HX8357 tft = Adafruit_HX8357(TFT_CS, TFT_DC, MOSI, SCK, TFT_RST, MISO);


// Assign human-readable names to some common 16-bit color values:
#define  BLACK   0x0000
#define BLUE    0x001F
#define RED     0xF800
#define GREEN   0x07E0
#define CYAN    0x07FF
#define MAGENTA 0xF81F
#define YELLOW  0xFFE0
#define WHITE   0xFFFF

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  Serial.println("HX8357D Test!");

  tft.begin(HX8357D);

  //Initialize Screen arrangement
```

```
  tft.setRotation(1);
  //initiate screen
  tft.fillScreen(BLACK);
  tft.drawLine(240,20,240,310,RED);//Draws straight red line thorugh the middle of the screen. This is
used to split it in half to distinguish each channel
  tft.drawLine(0,230,480,230,RED); //Draws line accross screen(diagonal)

//----------------- Channel 1 -----------------
    //Header
  tft.setCursor(10,0);// Move the cursor to the top left of the screen to label
  tft.setTextSize(4); //Make the header text big to make it distiguishable
  tft.setFont();
  tft.setTextColor(WHITE);
  tft.print("Channel 1");

    //Voltage
  tft.setCursor(10,100);
  tft.setTextSize(2);
  tft.print("Voltage (V): ");

    //Current
  tft.setCursor(10,200);
  tft.setTextSize(2);
  tft.print("Current(A): ");

  //Current LIMIT
  tft.setCursor(10,270);
  tft.setTextSize(1.85);
  tft.print("Current-Lim(A): ");


//----------------- Channel 2 -----------------
    //Header
  tft.setCursor(260,0);
  tft.setTextSize(4);
  tft.print("Channel 2");

    //Voltage
  tft.setCursor(260,100);
  tft.setTextSize(2);
  tft.print("Voltage(V): ");


    //Current
  tft.setCursor(260,200);
  tft.setTextSize(2);
  tft.print("Current(A): ");
```

```
  //Current LIMIT
  tft.setCursor(260,270);
  tft.setTextSize(1.85);
  tft.print("Current-Lim(A): ");

//---------------- Relays ----------------
  pinMode(relayPin1,OUTPUT);
  pinMode(relayPin2,OUTPUT);
  digitalWrite(relayPin1,LOW);
  digitalWrite(relayPin2,LOW);

  pinMode(VUpPin, INPUT);
  pinMode(VDownPin, INPUT);
  pinMode(AUpPin, INPUT);
  pinMode(ADownPin, INPUT);
  pinMode(outputPin, INPUT);

  digitalWrite(VUpPin, HIGH);
  digitalWrite(VDownPin, HIGH);
  digitalWrite(AUpPin, HIGH);
  digitalWrite(ADownPin, HIGH);
  digitalWrite(outputPin, HIGH);

  whichOutput = 1; //Set the system to initially control Channel 1
}

void loop() {
  computer_control();//Check for serial command

  switchOutput();
  changeVoltage();

  voltage1 = voltmeter1(PinChannel1); //Perform operation that calculates voltage at the specified node for
channel 1
  voltage2 = voltmeter2(PinChannel2);//Perform the same above operation that calculates voltage at the
specified node for channel 2

  Current1 = current1(Current_Pin1); //Call function to determine current going across the sensors on
channel 1
  Current2 = current2(Current_Pin2); //Call function to determine current going across the sensors on
channel 2

  relay1(amp1, Current1); //Make sure channel 1 doesn't pass what the user wants
  relay2(amp2, Current2);  //Make sure channel 2 doesn't pass what the user wants

  /*Serial.print("\n\n Voltage Channel 1 (V) = "); // shows the voltage measured
  Serial.print(voltage1); // the â€˜2â€™ after voltage allows you to display 2 digits after decimal point
```

```
  Serial.print("\t Voltage Channel 2 (V) = "); // shows the voltage measured
  Serial.print(voltage2); // the â€˜2â€™ after voltage allows you to display 2 digits after decimal point

  Serial.print("\n Current Channel 1 (A) = "); // shows the voltage measured
  Serial.print(Current1); // the â€˜2â€™ after voltage allows you to display 2 digits after decimal point

  Serial.print("\t Current Channel 2 (A) = "); // shows the voltage measured
  Serial.print(Current2); // the â€˜2â€™ after voltage allows you to display 2 digits after decimal point*/
/* Serial.print("\n Amp1 (A) = ");
  Serial.print(amp1);
  Serial.print("\n Amp2 (A) = ");
  Serial.print(amp2);*/

  main_menu(voltage1,Current1,voltage2,Current2,amp2,amp1); //update the menu

}

/*Function Name: main_menu(float volt1, float amps1, float volt2, float amps2, float limit1, float limit2)
 *Inputs: float volt1- The passed value for voltage sensing in channel 1; float amp1- The passed value for current sensor in channel 1; float volt2- Measured voltage across channel 2; float amp2- Measured current across channel 2; Limit 1 & 2- current that the user wants the current to cut off at.
 *Outputs: None. There is no return type on this function.
 *Description: The purpose of this function is to clear previous sensed values on the screen by overwriting it with a black patch. After doing that, we set the cursor at corresponding locations on the screen and print the values passed into the function.
*/
void main_menu(float volt1, float amps1, float volt2, float amps2, float limit1, float limit2){

//----------------- Channel 1 -----------------
    //Voltage
  tft.fillRect(157, 100, 70,80, BLACK); //Fill the previous value on the screen black. What this does is generate a black rectangle above the previous value to avoid printed values from overlapping.
  tft.setCursor(157,100); //Set cursor to the location we want to print the voltage for channel 1 at.
  tft.setTextSize(2); //Set the size to a desired value that's smaller than the headings.
  tft.setFont();
  tft.setTextColor(WHITE); //Set the text to white
  tft.print(volt1); //Print the voltage value at the cursor location

    //Current
  tft.fillRect(148, 200, 70,20, BLACK);//Fill the previous value on the screen black.
  tft.setCursor(148,200); //Set the cursor where we want to display the current for channel 1
  tft.print(amps1);

  //Current LIMIT
  tft.fillRect(360, 270, 70,20, BLACK);//Fill the previous value on the screen black.
  tft.setCursor(360,270); //Set the cursor where we want to display the current for channel 1
  tft.setTextSize(1);
  tft.print(limit1);
```

```
//----------------- Channel 2 -----------------
    //Voltage
  tft.fillRect(395, 100, 70,80, BLACK);//Fill the previous value on the screen black.
  tft.setCursor(395,100);//Set the cursor where we want to display the voltage for channel 2
  tft.setTextSize(2);
  tft.print(volt2);

    //Current
  tft.fillRect(400, 200, 70,20, BLACK);
  tft.setCursor(400,200); //Set the cursor where we want to display the current for channel 2
  tft.print(amps2);

  //Current LIMIT
  tft.fillRect(110, 270, 70,20, BLACK);//Fill the previous value on the screen black.
  tft.setCursor(110,270); //Set the cursor where we want to display the current for channel 1
  tft.setTextSize(1);
  tft.print(limit2);

}

/*Function Name: voltmeter1(int pin)
 *Inputs: int pin - This function uses the analog pin we are connecting to our voltage divider as an input. The purpose of this is to be able to re-use the function.
 *Outputs: float voltage- The ouput for this function is the calculated/sampled voltage from the voltage divider for channel 1 after having performed voltage divider arithmetic.
 *Description: This function takes ten samples at the specified input pin to find the average voltage accross the voltage divider. To do so, we took the analog readings, divided by the number of samples, and divided/multiplied by values to convert to a voltage. After having done that, we multiply it by a different constant obtained from thhe resistors.
*/
float voltmeter1(int pin)
{

    sample_count = 0; // Re-initialize value
    sum = 0; // Re-initialize value
    voltage1 = 0; // Re-initialize value

    // take a number of analog samples and add them up
    while (sample_count < NUMBER_SAMPLES) {
        sum += analogRead(pin);
        sample_count++;
    }

    // calculate the voltage from the analog pin from the Arduino. Note how I used 5.11 as the reference voltage instead of 5 V. This is a calibrated value for accurate readings
    voltage1 = ((float)sum / (float)NUMBER_SAMPLES * 5.08) / 1024.0; //1) Divide the sum by the number of samples taken. 2) Convert to a voltage using a reference voltage. In this case I used 5.08 because
```

```
that's the value a voltmeter would read when comparing values.

    voltage1 = voltage1 * 2.973;//Since a voltage divider was used, perform the arithmetic to derive 3.2: vin(r1/(r1+r2))
    //voltage1 = voltage1 * 3.1921922;//Since a voltage divider was used, perform the arithmetic to derive 3.2: vin(r1/(r1+r2))

    return voltage1; //return the measured value
}

/*Function Name: voltmeter2(int pin)
 *Inputs: int pin - This function uses the analog pin we are connecting to our voltage divider for channel 2 as an input. The purpose of this is to be able to re-use the function.
 *Outputs: float voltage- The ouput for this function is the calculated/sampled voltage from the voltage divider after having performed voltage divider arithmetic.
 *Description: This function takes ten samples at the specified input pin to find the average voltage accross the voltage divider. To do so, we took the analog readings, divided by the number of samples, and divided/multiplied by values to convert to a voltage. After having done that, we multiply it by a different constant obtained from thhe resistors.
*/
float voltmeter2(int pin)
{

    sample_count = 0; // Re-initialize value
    sum = 0; // Re-initialize value
    voltage2 = 0; // Re-initialize value

    // take a number of analog samples and add them up
    while (sample_count < NUMBER_SAMPLES) {
        sum += analogRead(pin);
        sample_count++;
    }

    // calculate the voltage from the analog pin from the Arduino. Note how I used 5.11 as the reference voltage instead of 5 V. This is a calibrated value for accurate readings
    voltage2 = ((float)sum / (float)NUMBER_SAMPLES * 5.08) / 1024.0; //1) Divide the sum by the number of samples taken. 2) Convert to a voltage using a reference voltage. In this case I used 5.08 because that's the value a voltmeter would read when comparing values.

    //voltage2 = voltage2 * 3.226804;//Since a voltage divider was used, perform the arithmetic to derive 3.2: vin(r1/(r1+r2))
    voltage2 = voltage2 * 3.0;//Since a voltage divider was used, perform the arithmetic to derive 3.0: vin(r1/(r1+r2)). This also takes into account the tolerances.

    return voltage2; //return the measured value
}

/*Function Name: current1(int pin)
```

```
 *Inputs: int pin- This function uses the pin we are connecting our ACS 712 5A sensor to.
 *Outputs: float Current- The ouput for this function is the calculated/sampled current from the sensor
 *Description: This function takes one thousand samples at the specified input pin to find the average
voltage accross the component. After finding the averaged value, one converts it to a current by
subtracting it's resting point and dividing it by the volts per amp relationship. Note that this function is
calibrated for channel 1;
*/
float current1(int pin){
  Voltage = 0;
  average_voltage = 0;
  Current = 0;

  // Voltage is Sensed 500 Times for precision
  for(int i = 0; i < 500; i++) {
    Voltage += (float)analogRead(pin)*(5.15/1024.);
    delay(1);
  }

  average_voltage = Voltage/500.;// Find the average voltage after 500 samples

  Current1 = (average_voltage -2.56)/0.140; // Convert to the current we seek
  return Current1;
}

/*Function Name: current2(int pin)
 *Inputs: int pin- This function uses the pin we are connecting our ACS 712 5A sensor to.
 *Outputs: float Current- The ouput for this function is the calculated/sampled current from the sensor
 *Description: This function takes one thousand samples at the specified input pin to find the average
voltage accross the component. After finding the averaged value, one converts it to a current by
subtracting it's resting point and dividing it by the volts per amp relationship. Note that this function is
calibrated for channel 2;
*/
float current2(int pin2){
  Voltage = 0;
  average_voltage = 0;
  Current = 0;

  // Voltage is Sensed 500 Times for precision
  for(int i = 0; i < 500; i++) {
    Voltage += (float)analogRead(pin2)*(5.12/1024.);
    delay(1);
  }

  average_voltage = Voltage/500.;// Find the average voltage after 500 samples

  Current2 = (average_voltage -2.55)/0.145; // Convert to the current we seek
  return Current2;
}
```

```
/*Function Name: computer_control()
 *Inputs: N/A
 *Outputs: N/A
 *Description: This function takes serial input and decides if the command is controlling voltage or current accross channel 1 or 2.
*/
void computer_control(){
  while(Serial.available()) {
      computerString = Serial.readString();// read the incoming data as string
      dataVal = dataParser(computerString);
      Serial.println(dataVal);
//--------------- Voltage ---------------
      if(computerString.charAt(1) == 'v'){ //Check if controlling voltage
        if(computerString.charAt(2) == '1'){//Check if we control channel 1 voltage
          Serial.println("Voltage Channel 1 control");
        }
        else if(computerString.charAt(2) == '2'){//Check if we control channel 2 voltage
          Serial.println("Voltage Channel 2 control");
        }
      }
//--------------- Current ---------------
      else if(computerString.charAt(1) == 'c'){ //Check if controlling current
        if(computerString.charAt(2) == '1'){//Check if we control channel 1 current
          Serial.println("Current Channel 1 control");
          amp1 = dataVal;
        }
        else if(computerString.charAt(2) == '2'){//Check if we control channel 2 voltage
          Serial.println("Current Channel 2 control");
          amp2 = dataVal;
        }
      }
  }
}

/*Function Name: dataParser(String command)
 *Inputs: String command- This is the stringed command read from the serial input queue.
 *Outputs: float dataVal
 *Description: this takes in a string from the serial input and pulls the floating values associated with the string.
*/
float dataParser(String command){
  String dataString = computerString.substring(4,computerString.indexOf(";")); //Get the that contains the floating value
    //Convert to float
    dataVal = dataString.toFloat();
    return dataVal;
}
```

```
/*Function Name: relay1
 * Input: digital signal from Arduino board
 * Output: the status of channel one
 * Description: this function judges if the current through the whole circuit is greater than the limit current.
 */
void relay1(float max_i, float exact_i ){
  if(exact_i >= 1.5 || exact_i > max_i){ //If the exact, measured current is greater than or equal to 1.5, cut the current to the system. //If the exact, measured current is greater than or equal to 1.5, cut the current to the system. Also make sure that the measured current isn't greater than what the user asked.
    digitalWrite(relayPin1,HIGH);
  }
  else if (exact_i >= 0.01){ //Otherwise shut the system off
    digitalWrite(relayPin1,LOW);
  }
}
/*Function Name: relay2
 * Input: digital signal from Arduino board
 * Output: the status of channel two
 * Description: this function judges if the current through the whole circuit is greater than the limit current.
 */
void relay2(float max_i, float exact_i ){
  if((exact_i >= 1.5 || exact_i > max_i)){ //If the exact, measured current is greater than or equal to 1.5, cut the current to the system. Also make sure that the measured current isn't greater than what the user asked.
    digitalWrite(relayPin2,HIGH);
  }
  else if ( exact_i >= 0.03 ){
    digitalWrite(relayPin2,LOW); //Otherwise shute the system off
  }
}

/*Function Name: changeVoltage
 * Input: digital signal from 8 button board
 * Output: the incremented or decremented values of volt1, volt2, amp1, or amp2
 * Description: this function increments or decremends voltage and current values dependent on the button board
 */
void changeVoltage(){
  VUp = digitalRead(VUpPin);  //Check if the VUp Button is being pressed

  VDown = digitalRead(VDownPin); //Check if the VDown Button is being pressed

  AUp = digitalRead(AUpPin);  //Check if the AUp Button is being pressed
  ADown = digitalRead(ADownPin); //Check if the ADown Button is being pressed


  if(VUp == 0 || VDown == 0 || AUp == 0|| ADown == 0) { //If at least one button is being pressed
    pressTime += 1;      //Increase pressTime
```

```
 }
 else if(VUp == 1 && VDown == 1 && AUp == 1 && ADown == 1) {   //If no buttons are being pressed
  pressTime = 0;  //Reset pressTime
  interval = 0.01;  //Reset interval
 }

 if(pressTime >= 6) interval = 0.1;  //If pressing for at least 3 seconds, make the interval larger
 else interval = 0.01;  //Reset if not

 if(whichOutput == 1){   //If Channel 1 is being controlled
  if(VUp != VDown)  //As long as both Voltage Control buttons aren't pressed at the same time
  {
   if(VUp == 0 && volt1 < 14){   //If volt1 is below the maximum
    volt1 + interval > 14; //Increase requested voltage
   }
   else if(VDown == 0 && volt1 > 2.00) volt1 = volt1 - interval;  //If VDown is pressed and the voltage is
above the minimum, decrease requested voltage
  }
  if(AUp != ADown)
  {      //If either both AUp and ADown are being pressed, or neither are, do nothing
   if(AUp == 0 && amp1 < 1.50) amp1 = amp1 + interval;   //If just AUp is pressed and the current is
below the maximum, increase requested current
   else if(ADown == 0 && amp1 > 0.05) amp1 = amp1 - interval;   //If ADown is pressed and the current
is above the minimum, decrease requested current
  }
 }

  if(whichOutput == 2){  //If Channel 2 is being controlled
  if(VUp != VDown)  //As long as one voltage button is pressed, and only one
  {
   if(VUp == 0 && volt2 < 14)  volt2 = volt2 + interval; //If just VUp is pressed and the voltage is below
the maximum, increase requested voltage
   else if(VDown == 0 && volt2 > 2.00) volt2 = volt2 - interval; //If VDown is pressed and the voltage is
above the minimum, decrease requested voltage
  }
  if(AUp != ADown)  //As long as one current button is pressed, and only one
  {
   if(AUp == 0 && amp2 < 1.50) amp2 = amp2 + interval; //If just AUp is pressed and the current is below
the maximum, increase requested current
   else if(ADown == 0 && amp2 > 0.05) amp2 = amp2 - interval;  //If ADown is pressed and the current is
above the minimum, decrease requested current
  }
 }

 if(amp1 < 0.2) amp1 = 0.2;  //If amp1 is outside the required scope, bring it back in
 if(amp1 > 1.5) amp1 = 1.5;
 if(amp2 < 0.2) amp2 = 0.2;  //If amp2 is outside the required scope, bring it back in
 if(amp2 > 1.5) amp2 = 1.5;
```

```
  /* Button Board Troubleshooting Code
  Serial.print("/n");
  Serial.print(oput);
  Serial.print(ADown);
  Serial.print("     ");
  Serial.print("Current Up: ");
  Serial.print(AUp);
  Serial.print("     ");
  Serial.print("Change Output ");
  Serial.print(oput);
  Serial.print("     ");*/
}

/*Function Name: switchOutput
 * Input: digital signal from 8 button board
 * Output: the channel to be controlled over the next loop through the program
 * Description: this function chooses whether or not to switch to controlling the currently inactive channel
 */
void switchOutput(){
  oput = digitalRead(outputPin);  //Check value of outputPin

  if(oput == 0 && whichOutput == 1)   //If channel is changed from 1 to 2
  {
    whichOutput = 2;  //Change control to Output 2
    Serial.print("\n Now controlling Output 2");  //Send alert that output control has been switched
    Serial.print("     ");
  }
  else if(oput == 0 && whichOutput == 2) //If channel change button is pressed to change from 2 to 1
  {
    whichOutput = 1;  //Change control to Output 1
    Serial.print( "\n Now controlling Output 1"); //Send alert that output control has been switched
    Serial.print("     ");
  }
}
```