Eclectronics Final Report

James Jagielski and Kenta Burpee

December 2022

1 Abstract

We were given loose requirements for the final project, which are, the project must involve designing, assembling, and testing a PCB that you have designed. We decided to build a source measure unit (SMU), which is a device that both sources and sinks voltage and current simultaneously on the same electrical leads. We built a SMU that only deals with positive voltages up to 12V and 65mA.

2 Circuit Schematic

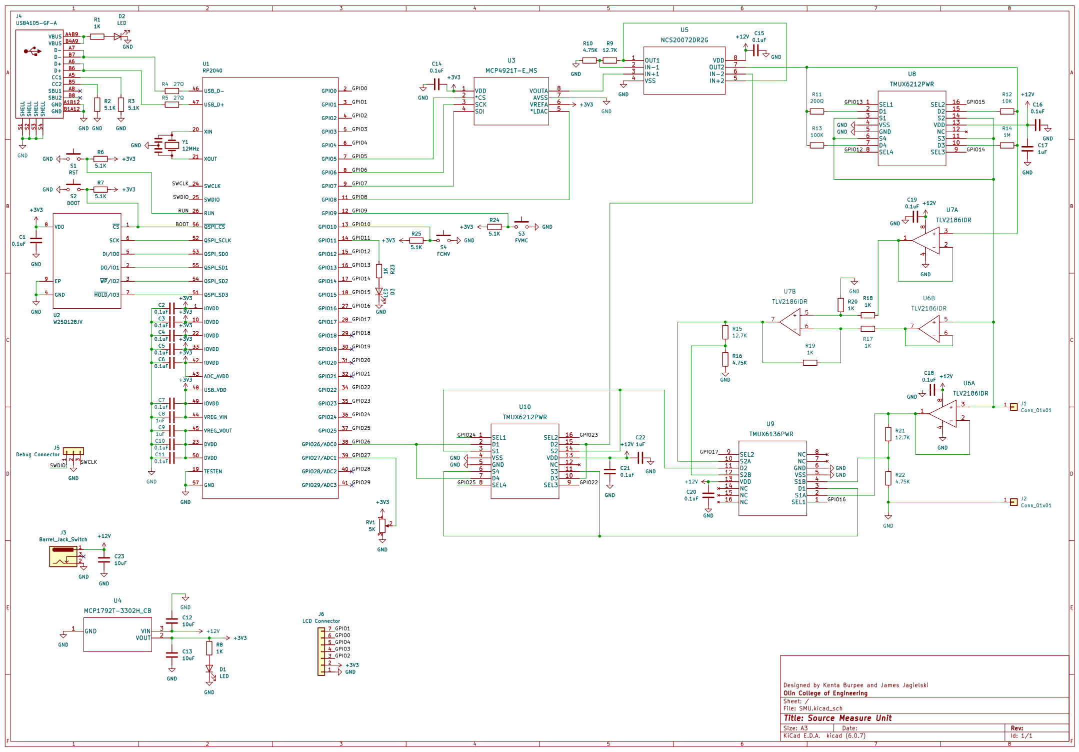

Figure 1: Schematic of entire SMU circuit.

3 Circuit Layout

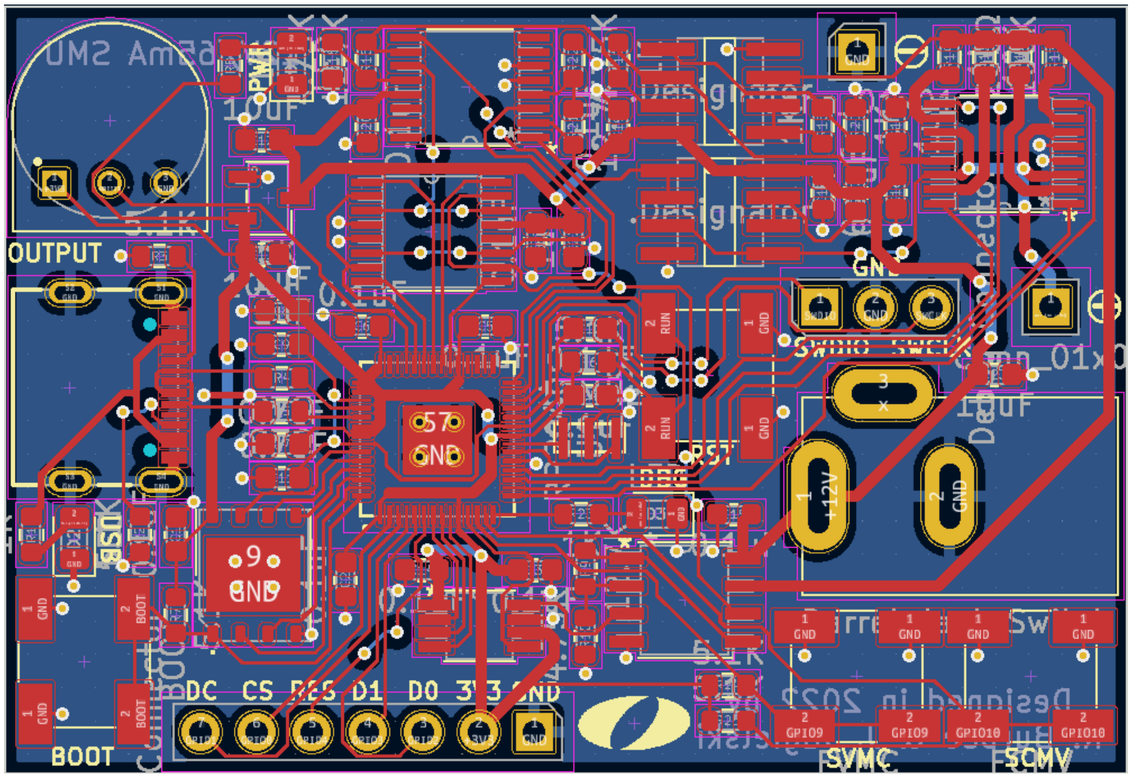
Figure 2: Layout of entire SMU circuit.

4 Ciruit Design

We had a lot of design decisions for this project because it was so open ended. We knew we had to interface with a microcontroller to control the circuitry and read in values from the device under test. We decided on using the RP2040 microcontroller because we had previously used the RP2040 for our last project and had some experience with it. The RP2040 also had all of the features we needed for our project. The RP2040 has Analog-to-Digital converter (ADC) pins, which we needed to read in the values of the device under test. We also wanted a Digital-to-Analog (DAC) on the microcontroller, but the RP2040 doesn't have DAC pins we used a peripheral component, which communicates over SPI. We didn't need to consider the speed of the microcontroller because our applications are not dependent on speed. The circuit design of the peripheral components on the RP2040 were inspired by the documentation supplied by the manufacturer of the chip[1].

[1]https://datasheets.raspberrypi.com/rp2040/hardware-design-with-rp2040.pdf

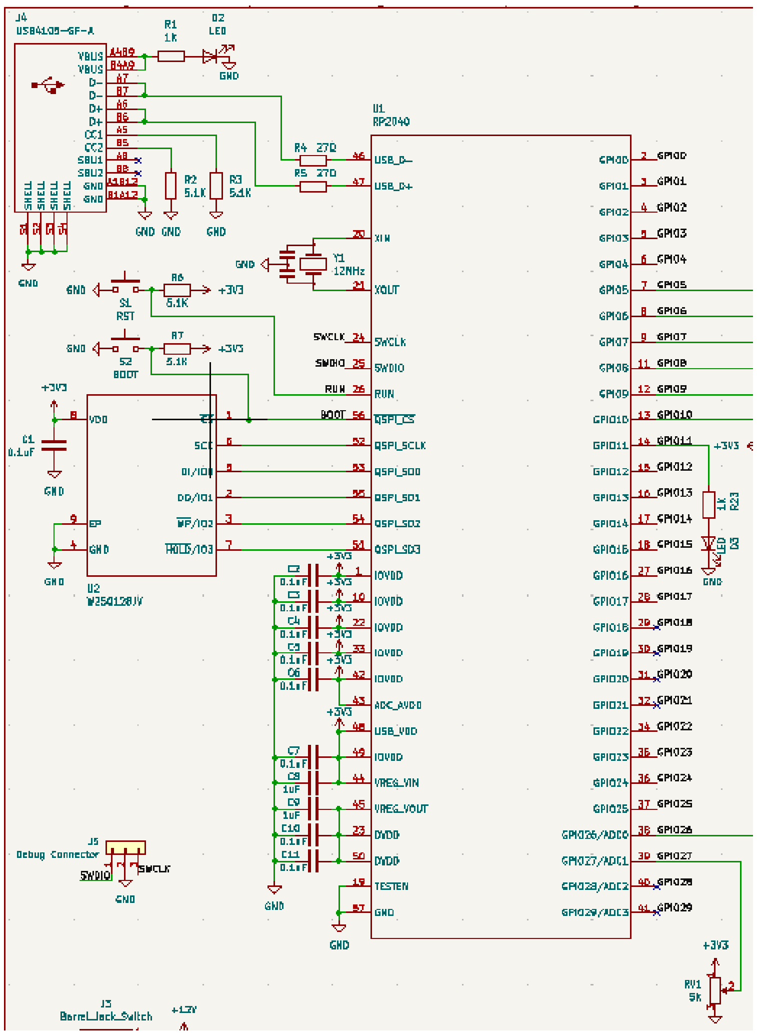
Figure 3: Schematic of microcontroller used in our project.

We included a memory unit for the chip and a crystal oscillator. The rest of circuit is the feedback loops, which create the interface with the device under test (DUT).

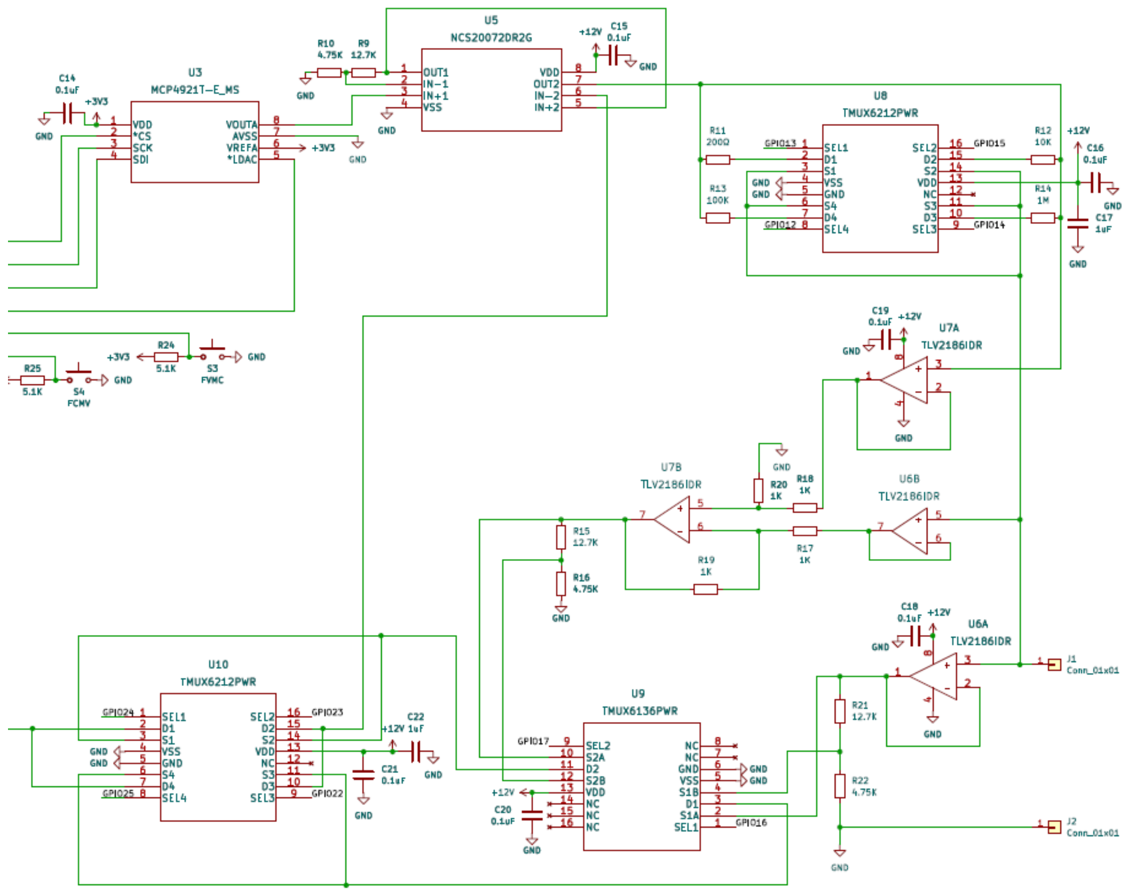

Figure 4: Schematic of the source and measuring feedback loops

For our connectors (J1 and J2), we simple used header pins to allow contact with alligator clips. One of the leads connects to the ground reference of our circuit and the other connector is where we read the voltage (in reference to ground). We use an op-amp buffer to minimize the current draw into our circuit, so our circuit has minimal effect on the device under test. We used a voltage divider to step down the voltage, so the voltage would be in the range of the ADC on the RP2040. If we had not done this the voltage would rail out to 3.3V and might damage the microcontroller. We used the analog switches in the TMUX6212PWR chip to switch between the different modes. We can break down the circuit into its two modes: source voltage while measuring current or source current while measuring voltage.

Starting with the mode sourcing voltage while measuring current, we source the amount of voltage from the RP2040, which connects to a DAC (U3 in Fig.4). We then output the DAC to an op-amp (U5 in Fig.4), which will scale the voltage from the 3.3V RP2040 range to the 12V scale in our circuit. The voltage from the op-amp then outputs directly to the DUT. We included analog switches connecting to resistors to measure the current going into the circuit (U8 in Fig.4). We really measure the voltage drop across the resistor and the multiplexer and calculate the current with the microcontroller. Due to the resistor, we decided to include a feedback loop with the voltage drop across the two leads for the DUT. This allows us to compensate for the voltage drop across the current sense resistor. Thus, we calculate the difference between our ideal output and the real output and adjust for that change with the circuit. The current in this configuration will be measuring into the ground connector pin.

Our next mode is sourcing current while measuring voltage. For this mode we use the resistor between the source op-amp and the positive connector pin to control the range of the current and for the fine tuning we change the voltage across the resistor. We use Ohms law to achieve calculate the current output. We have the analog switches to change the range of the current. For instance, the 1 megohm resistor has a max of $12\mu A$, while the 200Ω can reach the mA range. We again have a feedback loop for the voltage across the resistor to adjust to the ideal current. This feeds back into the sourcing op-amp. We then measure the voltage across the two leads by stepping down the voltage into the 3.3V range and reading the voltage with the RP2040 ADC.

5 Firmware

We wrote the firmware for the SMU using the Arduino Pico Project, which allows interfacing with the RP2040 through the Arduino IDE. Other than initial set up, there are three main parts to the firmware: switching between SMU modes, setting the DAC output, and reading the ADC.

Switching between SMU modes involves setting a number of GPIO pins high and low depending on the mode. for instance, the function below sets the SMU to the source voltage, measure current mode. In order to do so, it sets GPIO pins so that the voltage across the DUT is used as negative feedback to the force amplifier and the voltage across the 200 ohm resistor is read by the ADC to calculate the current across the DUT.

```
void set_svmc() {
  smu_mode = 0;
  pot_value = 0;
  digitalWrite(RSENSE_TO_FEEDBACK, LOW);
  digitalWrite(DUT_TO_ADC, LOW);
  digitalWrite(RSENSE_VOLTAGE_SELECT, LOW);
  digitalWrite(DUT_VOLTAGE_SELECT, HIGH);
  digitalWrite(TEN_K_OHM, LOW);
  digitalWrite(ONE_HUNDRED_K_OHM, LOW);
  digitalWrite(ONE_MILLION_OHM, LOW);
  digitalWrite(TWO_HUNDRED_OHM, HIGH);
  digitalWrite(DUT_TO_FEEDBACK, HIGH);
  digitalWrite(RSENSE_TO_ADC, HIGH);
  Serial.println("Mode Changed: Source Voltage, Measure Current");
 }
```

Setting the DAC output requires SPI communication between the RP2040 and the DAC. The DAC takes twelve bits of information to set its output, as well as four bits of settings for a total of two bytes in order to set its output using SPI. Below is the function used set a new DAC output, which is called when the ADC reads a new voltage from the potentiometer.

First, the debug LED is set high to indicate to the user that the DAC output is being changed. Next, a SPI transaction is initiated, and the chip select pin for the DAC is set low. Then, the sixteen bits of data are split into two separate bytes and transferred to the DAC one byte at a time. Finally, the LDAC pin on the DAC is set low then high again to latch the data and set the analog output of the DAC, and the SPI transaction is finished.

```
void set_dac_output(uint16_t input) {
  digitalWrite(LED, HIGH);
  SPI.beginTransaction(SPISettings(20000000, MSBFIRST, SPI_MODE0));
  digitalWrite(CS, LOW);
  byte0 = (DAC_SETTINGS | (input >> 8));
  byte1 = (input & 0xFF);
  SPI.transfer(byte0);
  SPI.transfer(byte1);
  digitalWrite(CS, HIGH);
  delay(5);
  digitalWrite(LDAC, LOW);
  delay(5);
  digitalWrite(LDAC, HIGH);
  SPI.endTransaction();
  delay(50);
  digitalWrite(LED, LOW);
}
```

There are two pins we read analog values from using the ADC. The first is the potentiometer, which is used to determine the output of the DAC. This digital number is then converted to an analog voltage or current sepending on the mode and printed to Serial to communicate the value of the sourced voltage or current to the user. The second pin is used to measure the voltage across the DUT or the internal sensing resistor. This value is also converted to an analog voltage or current and is printed to Serial as the measured value.

6 Results

We were able to get the PCB working with code to switch between the different modes with button presses. There were some inaccuracies with our measurements. When we were measuring a resistance based on our measurements, we were about 10% off the actual resistance.

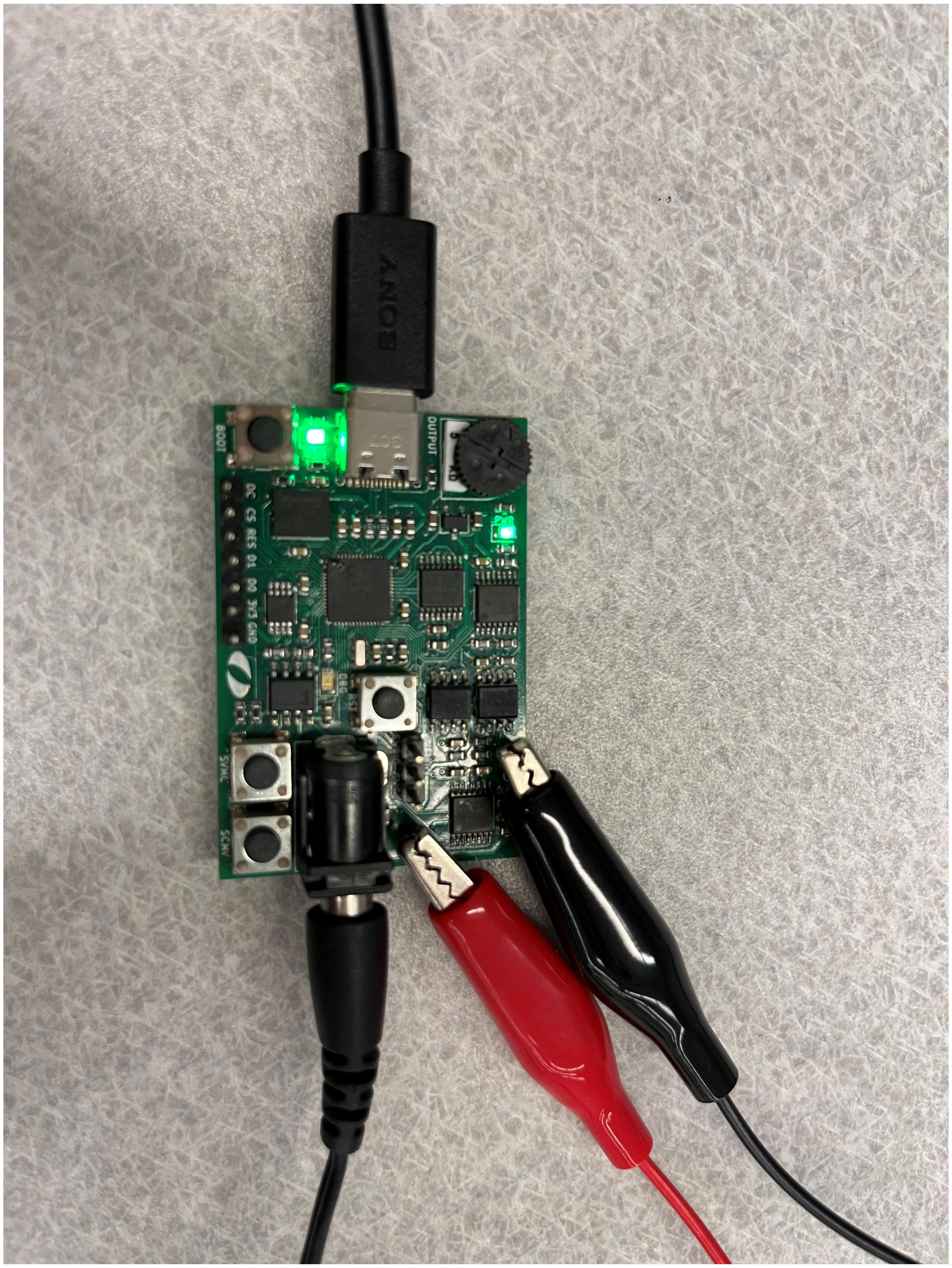

Figure 5: The built PCB working. We were reading in serial data from the RP2040 across the two alligator clips with a resistor as the DUT.

The GitHub repository containing all of our documents: https://github.com/kburp/source-measure-unit.

The bill of materials: https://olincollege-my.sharepoint.com/:x:/g/personal/jjagielski_olin_edu/Eego2vmsqwVHoNpAVt0sM7UBYRnT2plQWsWwdy28PpcxZQ?e=19auvo

The demonstration video: https://youtu.be/EKSfAqJ0a-o

7 Reflection

We ran into difficulties during the circuit design phase of this project. We were in between making an only positive voltage SMU or a positive and negative voltage SMU. We decided to go with the positive SMU because we would've only been able to make one board with the positive and negative voltages with our budget. We did make the design for the positive and negative voltages.

The circuit design of this project was the toughest part. We did a lot of research into how SMUs in industry work and we had to really know the system. The circuit design was also influenced by the components we could get. Our project went smoothly after the circuit design. Our goal was to design a very condense board layout and make it as small as possible, which we think we have achieved. The coding portion of the project was also quite smooth because when we were waiting for the boards to be manufactured we were able to use a PCB with a RP2040 to communicate over SPI with a similar DAC to ours. This was the most difficult part to the code because the rest of it was switching the analog switches with high and low signals from GPIO pins.