2 Design

2.1 Block Diagram

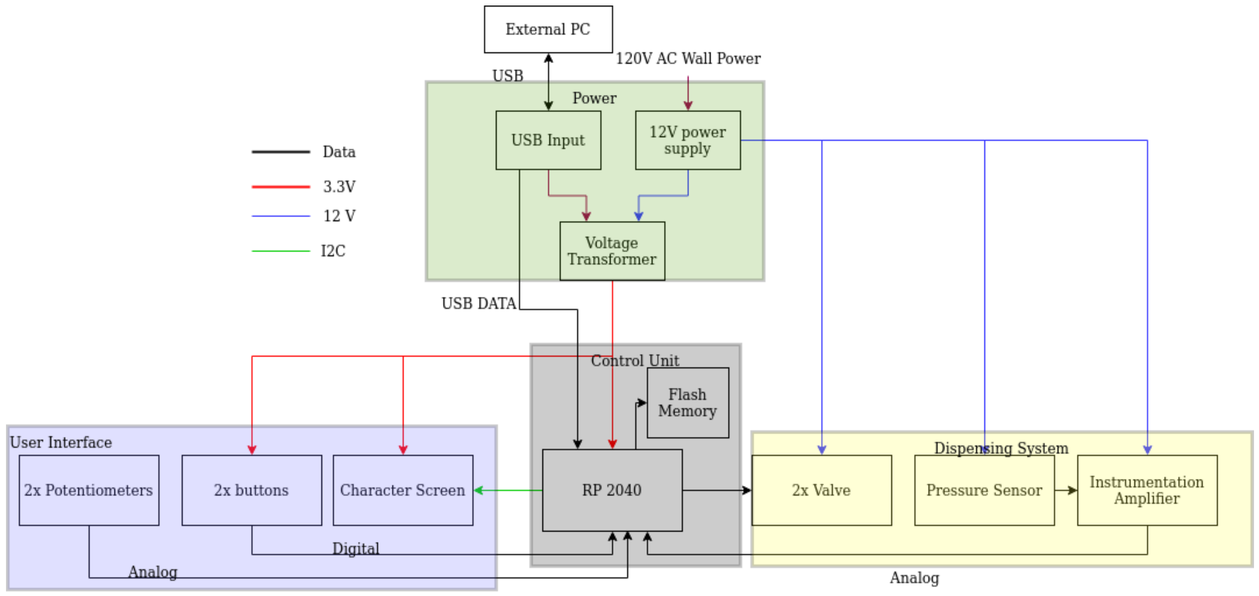

Figure 2: Block diagram for the vending machine

The block diagram has 4 main parts. The user interface interacts with the RP2040 by signaling button presses and potentiometer changes. The RP2040 then will display the number from the potentiometer onto the character screen of the user interface. The dispensing system valves are controlled by the RP2040. Additionally, the pressure sensor in the dispensing system will notify the RP2040 when to start and stop dispensing. The flash memory will be used to store program instructions and data about the containers. Finally, the power subsystem will supply all other subsystems with the proper power type.

2.2 Power

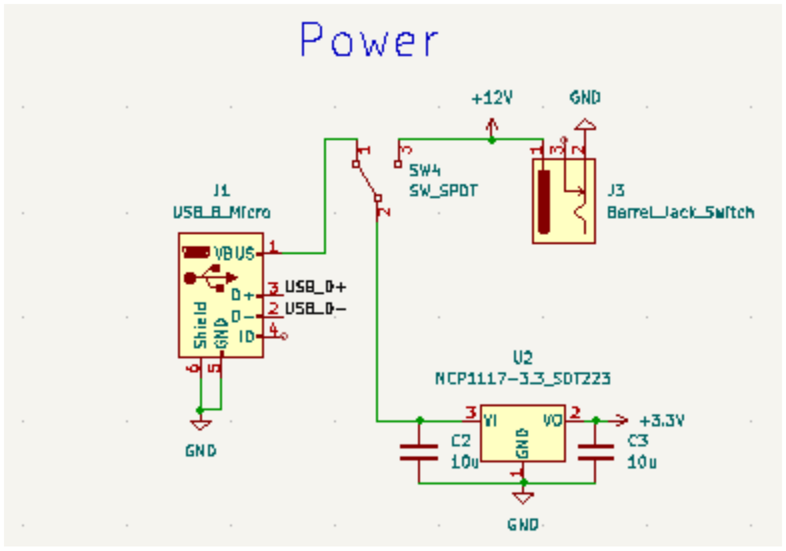

Figure 3: Power Subsystem

The power system consists of several parts: a barrel jack for the 12V line, a 3.3V voltage regulator, two 10uF capacitors, and a switch to be able to switch from the 5V USB power line to the 12V line from the barrel jack. The USB power input is necessary to initially flash the device, but the 12V input is intended as the main power input during normal usage. The high voltage components(e.g. solenoid valves) are only powered from the 12V line, but the digital components can be powered from either the USB or the 12V power supply filtered through the voltage regulator.

2.3 Dispensing System

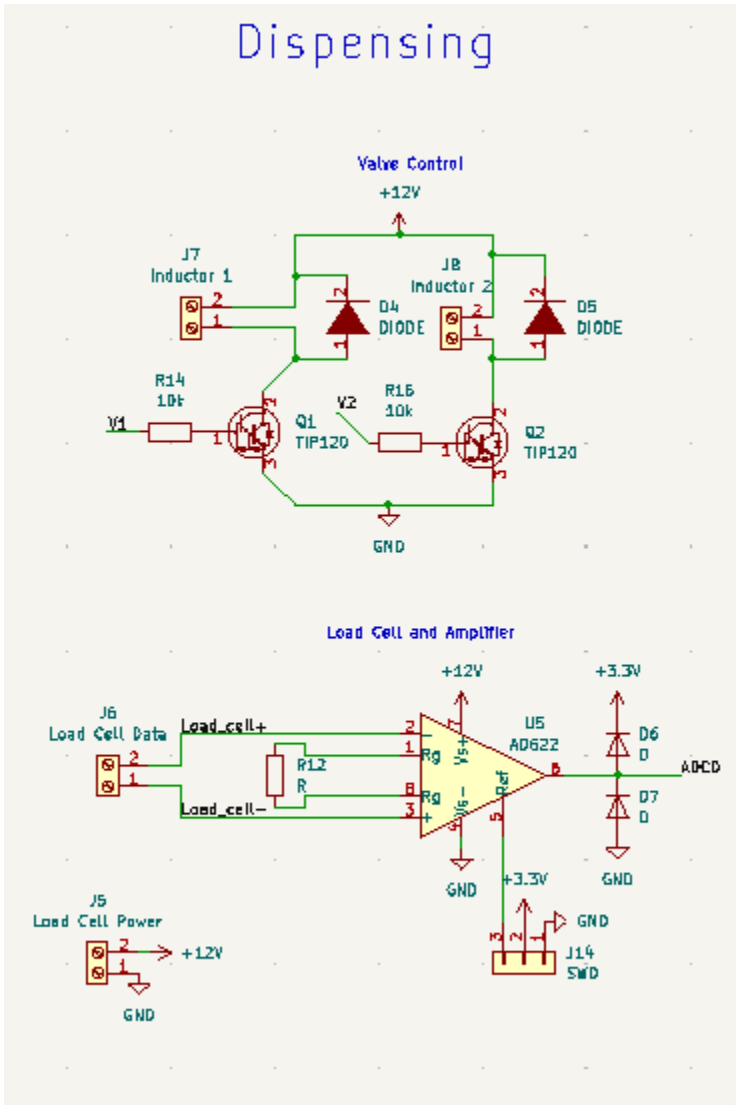

Figure 4: Dispensing Schematic

The dispensing system consists of two halves: solenoid valves and a load cell. The load cell is composed of a strain gauge (essentially a resistor whose value changes as it is bent) attached across a Wheatstone bridge. This generates a small differential voltage that is then fed through an instrumentation amplifier. The solenoid valves are simple 12V 400mA valves that are by default closed, each of which is connected to a Darlington BJT transistor and a flyback diode.

Originally, we planned to feed the load cell values through the AD622 instrumentation amplifier which would allow the value to be read by an ADC port of the microcontroller. We added in an analog to digital converter (ADC) protection circuit at the output of the amplifier to ensure we didn't overload the ADC on the microcontroller as the instrumentation amplifier required 12V while the microcontroller could only handle 3.3 V. After unit testing this section of the design on the breadboard, it was clear that something was wrong and it wasn't going to work. Instead, we came up with another way to get the load cell values over to the microcontroller.

Our new method involved using an HX711 amplifier which had the 24-bit ADC built in. We used a MicroPython HX711 library to tare the scale and analyze the values coming from it [2]. This allowed the 2^24 or roughly sixteen million unique values that the ADC was capable of differentiating. A further advantage was that we could power this amplifier from the 3.3V line. However, this amplifier complicated our design because it did not conform to a standard data transfer protocol. We were successfully able to read values from the load cell this way, however, it was too late to make a spot for the HX711 on the printed circuit board.

2.4 User Interface

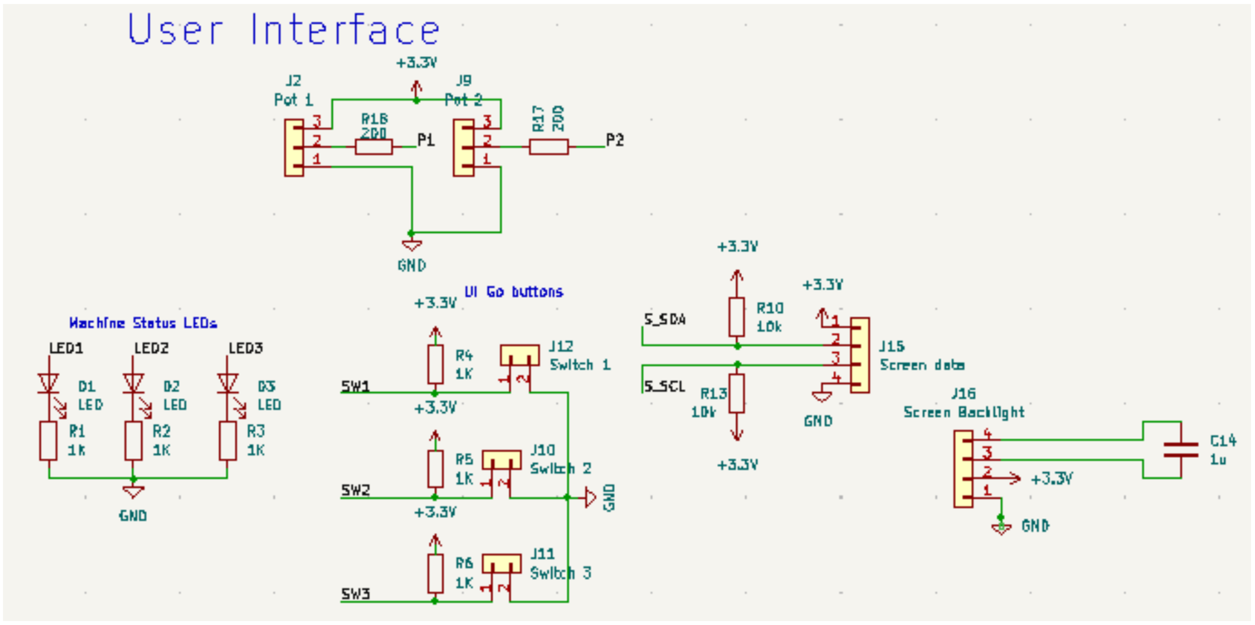

Figure 5: User Interface Schematic

The user interface consists of two potentiometers, three push-buttons, three LEDs and a screen. Two buttons are located underneath the bottles which lets the user know which button to press to dispense an item. The potentiometers are each located underneath the buttons to signify that turning it controls the amount to be dispensed for that particular bottle. As the potentiometers are turned the value gets sent to the microcontroller which sends the proper value to the screen to update the amount to be dispensed in grams. There is a red button in the middle of the user interface which resets the machine after a restocking takes place or overflow gets cleaned up. This button sends the machine back to the beginning where it’s waiting for a user to place an order. There are three LEDs on the side of the machine which indicate to the user if the machine is ready for an order (green), the machine is processing an order (orange), or if the machine needs maintenance (red).

Originally, we planned to use the NHD-C0220BIZ screen from Newhaven Display as we thought it would interface well with the I2C while consuming little power. Unfortunately, that was not the case. After carefully building out the circuit specified in the datasheet and turning on power, the screen would heat up immediately and no text could be displayed. We decided to switch out the screen for an OLED LCD Display Board Module I2C IIC SSD1306. This screen was easier to use as there were only 4 pins instead of 12 and the screen worked well with the Machine, SSD1306, and OLED libraries in MicroPython[3]. The screen communicates the amount to be dispensed to the user, live-time updates of the amount dispensed, the final amount dispensed, and if the machine needs maintenance.

2.5 Control Unit

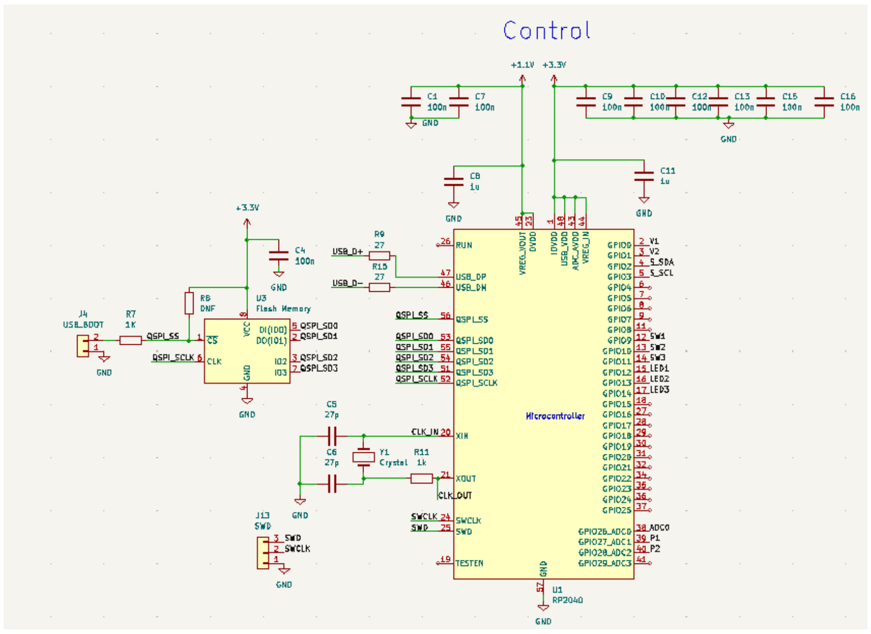

Figure 6: Control Unit Schematic

We used an RP2040 in combination with flash memory on the PCB. Our design is inspired by the one in the RP2040 Hardware design doc [4]. In order to load code to the RP2040, a USB connection was made to the computer, the code was downloaded and stored on the flash memory and then the program would continuously run. We chose the RP2040 as it was quite powerful for the price, which made it useful for scalability of the dispensing machines, and it is programmable using a variety of IDEs.
For testing purposes, we used a Raspberry Pi Pico on a breadboard which was cheap ($4), easily replaceable, and allowed us to test each subsystem before placing it on the PCB. Additionally, since the code relied on GPIO pins instead of physical pin numbers, we were able to interchange the code on the Pico for the RP2040 seamlessly.