Barmaid

April 04, 2022

Table of Contents:

Introduction:

Motivation:

Well made cocktails are a highly valued commodity, but the necessary care and precision that they require deters many from making them. This issue is further emphasized when dealing with larger parties, since making certain elaborate cocktails is a tedious and time-consuming process. In such situations, hiring a bartender is an option, but it comes at a steep price; therefore, an automatic cocktail maker is developed to address such a problem.

Barmaid targets those who enjoy drinking and mixology, as it aims to turn the static process of cocktail-making into something more tangible and appealing in the form of integrated technologies. For some, Barmaid is simply a convenient device for those lazy nights, but for others, it can become the staple to host large-scale functions and relieve financial constraints.

Requirements and Objectives:

Barmaid must be able to provide the user with their selected drink accurately and consistently. The device must include at least four bottles of choice, be able to dispense the liquids at any quantity and allow for mixing. The accuracy of the pour must be within ∓10% (value is on the larger size because 10% of any generic 1 and a half ounce cocktail is negligible). Furthermore, the mechanical framework of the build must be able to support “tall drinks” of up to 9 ounces. As for the user interface, all drink requests must be processed on board with a simple and self-explanatory set-up using a screen and switches. The two components enable a reconfigurable interface, allowing the addition of a new cocktail recipe if necessary. Upon registering user inputs, Barmaid must properly orient the glass under the assigned bottles, and dispense the liquid at the programmed quantity.

Qualitative requirements and objectives include building the project to emulate a saloon theme. Barmaid is also expected to be visually appealing.

Design and Implementation:

Figure 1: Overview of Barmaid with labeled subsystems

Overview of Barmaid:

Users place a cup onto the conveyor platform then navigate through the user interface to select their drink option. Once the option is entered into the User Interface (UI) Box, the cup translates along the conveyor track until it is underneath the desired bottle or bottles that are necessary to make the user's drink. While the cup is under the bottle, a solenoid valve will be powered to allow the liquid to flow in. A weight sensor attached to the platform will be able to sense if the cup has been filled to the threshold amount and will power down the solenoid valve to close. If the cocktail is complete, the cup returns to the origin, otherwise, it will move onto the next bottle and this process will repeat.

Mechanical:

Frame:

Originally Barmaids' design utilized a T type Aluminum Extrusion frame, but due to the lack of flexibility, wood was chosen to build the frame. This design choice is justified with the following reasonings. Firstly, wood is easy to manipulate as it allows drill holes, screws in a structure, and etc, while providing strong support to our purposes. Second, Barmaid is built with a saloon theme in mind and wood will perfectly match this when stained appropriately.

Electronics and Valves:

To ensure Barmaid looks as authentic as possible, all of the valves are secured to the back of the frame and all of the electronics are secured to the bottom of the frame. This keeps the valves and electronics out of the way and out of sight of the user, which ensures they do not take away from the look of Barmaid. The valves are secured close to the bottle, this uses the smallest amount of tubing as possible, which helps keep costs down. It is inevitable when working with liquids that some will spill which is why electronics go to the bottom of Barmaid instead of the back. Electronics at the bottom can avoid and minimize the likelihood of water damage.

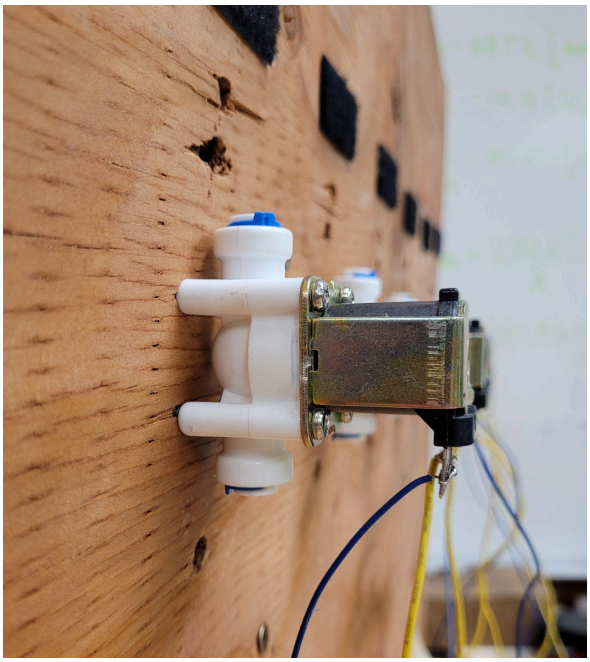


Figure 2: Solenoid Valve

Dispensing Mechanism and Fluid Dynamics:

The bottles are displayed at the top of the shelf upside down and visible to the user, this was done intentionally. By displaying the bottles, it adds to the overall bar look of Barmaid and sticks to the saloon theme. This design also takes advantage of gravity to dispense liquid from the bottles instead of using pumps. This dispensing mechanism is less financially constraining than peristaltic pumps, allowing more types of bottles for the users. However, one must understand the fluid dynamics of such an apparatus.

Figure 3: Bottle shelf and the two columns supporting it from underneath

The bottles are fitted with two tubes, one for the inlet, which allows liquid to flow into a valve and the other for an air intake tube. When initially testing the system with only one tube, fluid flow stopped when the liquid in the bottle reached some critical elevation. At this elevation, the pressure inside the bottle is less than the pressure exuded on the outlet tube connected to the valve. This occurs due to depressurization inside the bottle. Initially, the pressure inside the bottle is atmospheric pressure and the pressure of the liquid. When flow starts, the cavity of air inside depressurizes, causing the pressure inside the bottle to steadily decrease. This, coupled with the decreasing pressure of the liquid inside, eventually impedes the flow of the liquid when the entire pressure of the bottle falls below a certain threshold. Furthermore, because the valves could only accommodate tubing with a diameter of ¼ inch, it is too small for the air to get into the bottle to pressurize it. Adding a second tube to be an air intake tube was the cheapest and simplest solution to enable continuous flow.

Figure 4: Inlet and Intake Tubes

Weight Sensor:

To measure the quantity of the liquid being poured, a load cell is used. Load cells produce signals based on the mechanical force applied to them. Hence, the load cell is mounted in a way that allows bending similar to a cantilever beam, where load ‘w’ represents the force applied by the cup and the liquid (Figure 5). The simplified bending moment of this structure can be characterized by the bending moment diagram shown below. The load cell is attached to the platform of the conveyor system as that is where the users place their cups.

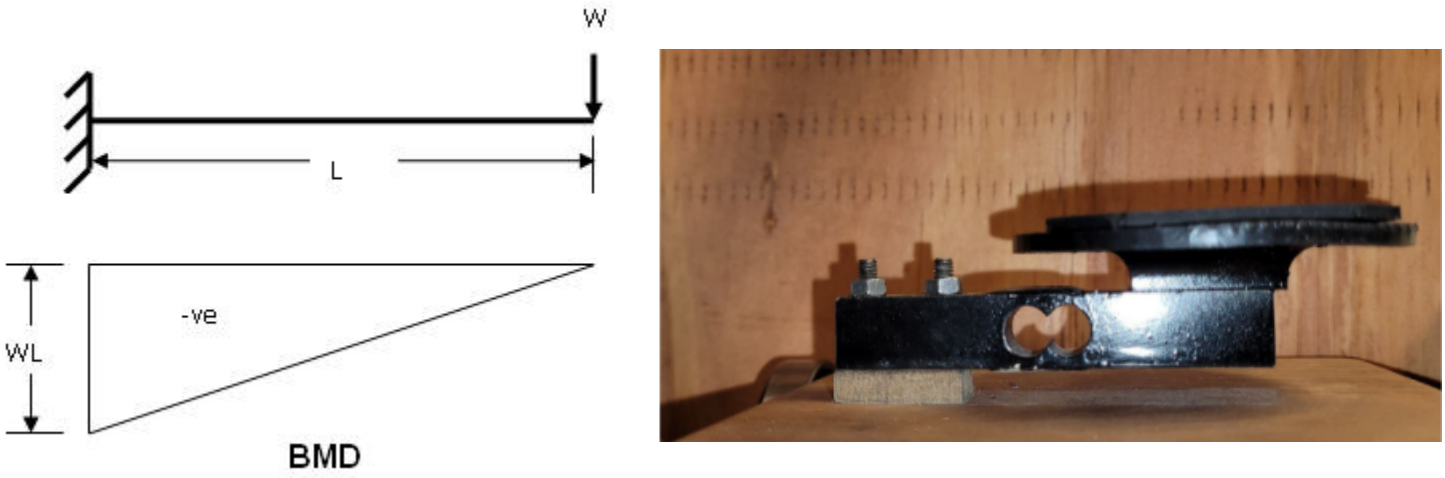


Figure 5: Bending Moment Diagram and Mounted Load Cell

Additionally, a piece of vulcanized rubber is mounted on top of the circular extrusion that the cup is placed on (Figure 5). This prevents the bottom of the cup from sliding since the coefficient of friction between the two surfaces is increased. Originally, the interaction of the bottom of the cup with the polylactic acid (PLA) surface caused the cup to slide from side to side, but by introducing the rubber, this no longer happens.

Conveyor:

To make Barmaid more visually appealing, a conveyor belt system to translate a moving platform from side to side is used. This not only adds to the attraction to the project, but it reduces the need for additional tubing. The current configuration for the tubing places them underneath their drink station (Figure 1). Hypothetically, if these tubes were to route to the same place, several issues would arise. Firstly, if this project is ever implemented to a larger scale, routing would become a tedious process. Secondly, the additional losses from friction in the tubing would make it so that flow would stop before the bottles completely emptied.

Our conveyor uses a motor in conjunction with ball bearings. The motor runs along a rubber belt with a series of periodic indentations, known as teeth. As the motor turns, it catches on these teeth and moves the platform. A rectangular wooden piece is manufactured with a small groove as shown in the figure below for the ball bearings to rest in, which runs parallel to the rubber belt to ensure movement in a single axis.

Figure 6: Belt on the left, Groove on the right

Endstop:

Stepper motor uses an open-loop control system, which means if one takes the platform off of the frame, it can decrease the repeatability of the motor positioning. As a result, an endstop is developed and installed as an easy way for Barmaid to initialize/home the platform, similar to how a 3D printer first resets its positional value by homing to one of the corners on the startup. In the case of Barmaid, users simply slide the platform until it closes the endstop, which is indicated by the red LED, explained further in the electrical section.

The endstop box is designed to hold the endstop in a position that will allow the switch to be triggered when the platform drives back to the origin of the conveyor. During the design of the box, some constraints and requirements are put into place:

1. The switch box must hold the endstop firmly in place.
2. The switch box cannot obstruct the platform's movement.
3. The switch box needs to have a hole to allow the wires of the endstop to come out.

The endstop box is modeled in Solidworks and 3D-printed using PLA printer filament. The shape of the design utilizes curved geometry in order to reduce stress points of corners and make the model more visually appealing. It is also fitted within a rectangular slot that matches the profile of the switch. The slot relies on friction to hold the switch firmly in place. This is achieved by purposely printing the slot with a smaller dimension and then sanding the slot to guarantee a tight fit. Two #6 ¾” pan head screws are used to secured both sides of the switch box to the frame of the barmaid, and a circular access port is made to route the endstop switch wires out the side of the switch box on the side that would not be visible to the user, wich made for a cleaner look.

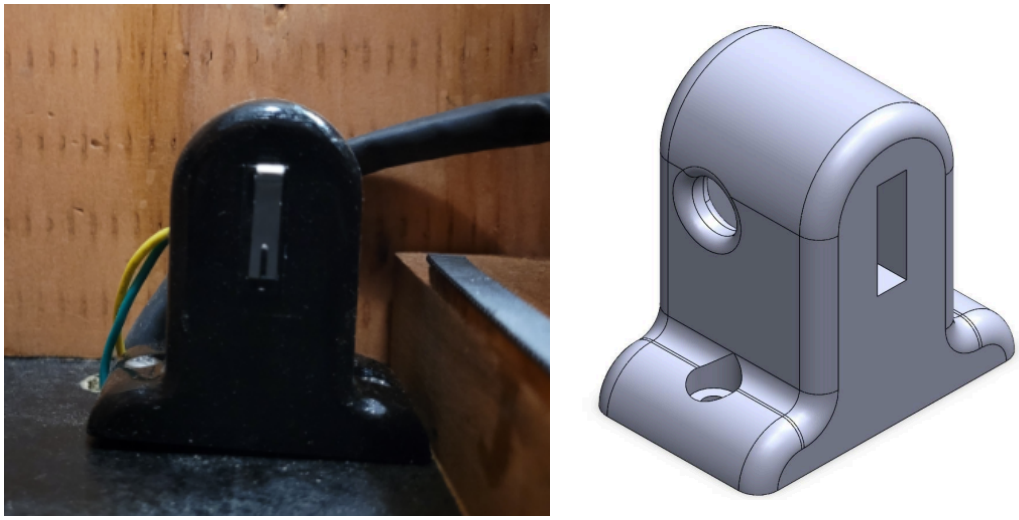


Figure 7: Installed and 3D Modeled Endstop

User Interface (UI) Box:

The purpose of the UI box is to house the UI components. It is modeled in Solidworks and 3D-printed. The UI box is made of two parts: the front plate and back plate. The backplate is attached to the frame of the barmaid using four 1½" wood screws. The backplate is designed with a slope that allows for the front plate of the UI box to be inserted into the back plate at an angle. This angle allows users to easily access the UI components. The front plate of the UI Box has cutouts to mount four push buttons and the Liquid Crystal Display (LCD).

On the front surface, a set of bush buttons are used. To facilitate user interaction, only four buttons, labeled "Up, Down, Select, and Cancel," are attached to the UI box. This simplistic arrangement of buttons makes the user interface extremely easy to navigate.

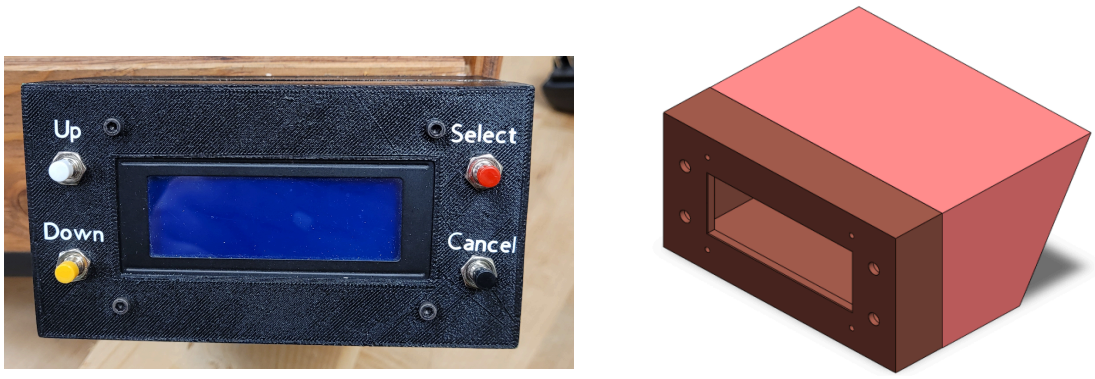


Figure 8: Installed and 3D Modeled UI Box

Electrical:

Valves:

Valves play an important role in dispensing the liquid from bottles. They are controlled by a microcontroller, in this project, Arduino Uno is used. Due to the mechanical design of Barmaid, fluids in the bottles are always trying to flow, hence, valves are chosen as a way to control it. One advantage of using solenoid valves is that they operate with a simple apparatus; they open up when a current passes and energizes an internal coil, generating a magnetic field (MF) and lifting the magnet inside. In order to generate a MF strong enough to lift the magnet, the valve is rated for 12V, which is a higher voltage level than what Arduino can operate at. This voltage necessitates a relay module which essentially acts as a switch controlled by a microcontroller. Interestingly, the relay and the valve have similar electromechanical structure in a sense that a relay closes a switch with a MF generated by the MCU's power.

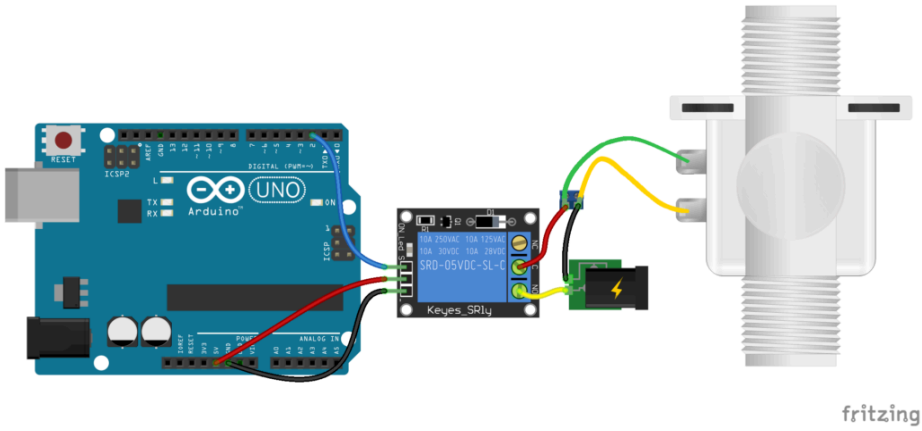


Figure 9: Circuit Diagram of Relay and Solenoid Valve [4]

Unlike the relay, valves generate a MF that is potentially dangerous, as it forms an arc of electricity between the switch terminals when opened. This is due to the nature of the solenoid to resist a change in current. When the switch is opened, the magnetic field will try to collapse rapidly in the form of a spark between the open terminals. To prevent this, one must implement a diode, also known as a flyback diode, in parallel with the solenoid, this is to allow the MF to slowly collapse through it.

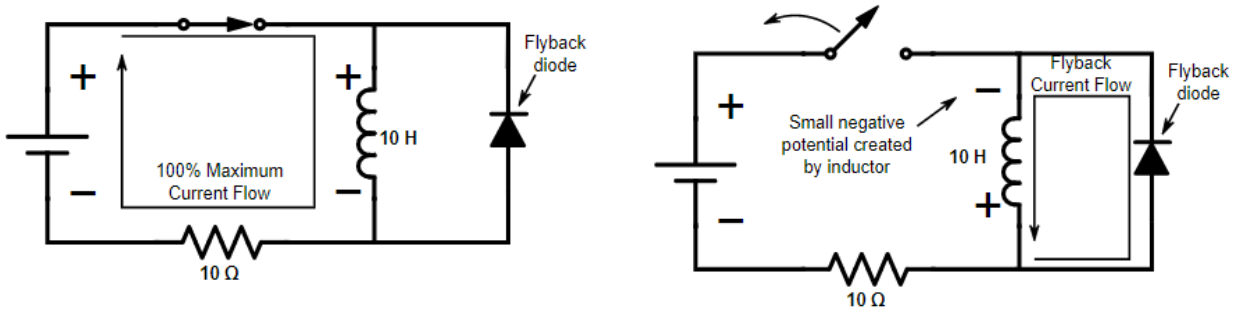


Figure 10: Flyback Diode Circuit

Weight Sensor:

This subsystem works in conjunction with valves by employing a load cell to accurately weigh the liquid entering the cup and thereby measuring its volume. Load cells are basically resistors that vary depending on the bend it experiences in the structure. By setting up a voltage divider using a wheatstone bridge circuit, a microcontroller can read the voltage levels. Although load cells produce small signals, it can be amplified through the HX711 Analog to Digital Converter (ADC) Module which allows Arduino to interpret the data more clearly. In the software section, this report will explore in depth how to set this electronic up and how it controls the valves, but it essentially tells the valves when to close and eliminates the risk of over/undershooting.

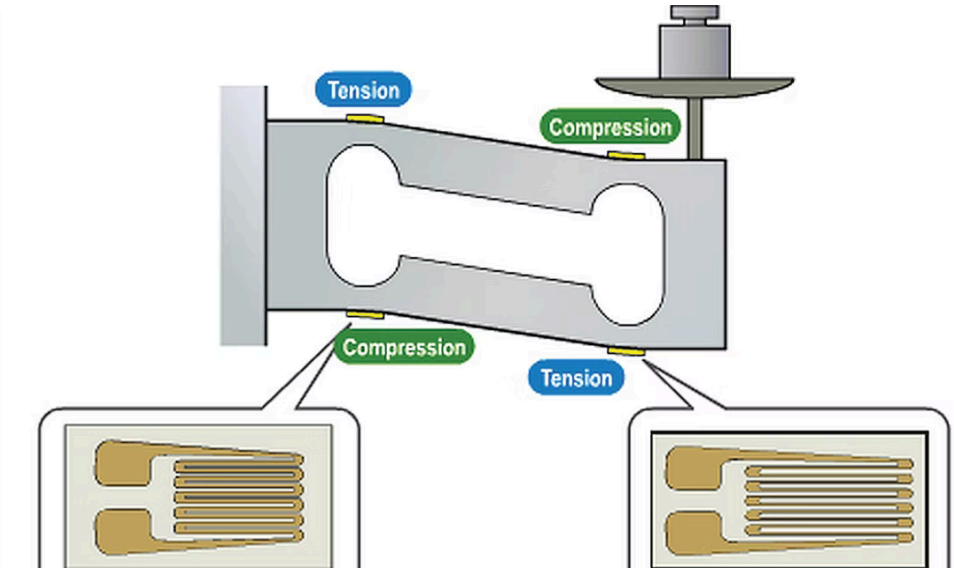


Fig 11: Mechanic of the load cell

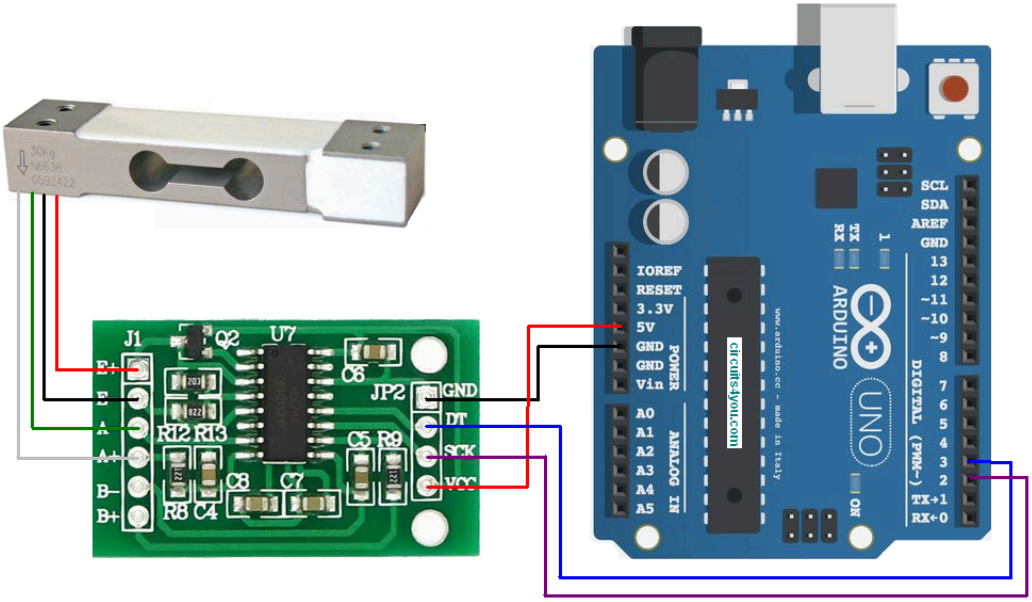


Figure 12: Circuit of the load cell, HX711, and Arduino. [2]

Conveyor:

Motor:

For the main moving force of the platform, a motor is used, specifically NEMA-17 stepper motor. Stepper motors provide accurate positioning and high levels of torque thanks to the gearbox which is suitable for Barmaid's application. To drive this motor, an A4988 stepper motor driver is selected with the wirings shown in figure 13. Notice the capacitor in the circuit placed in parallel with the power supply; this capacitor is also known as a bypass capacitor, and it is used to smooth out the voltage supply by sending a voltage spike to ground (since capacitor passes AC and blocks DC). Additionally, the capacitor also acts as a bulk capacitor to provide power when the motor attempts to draw more current during switchings in the driver.

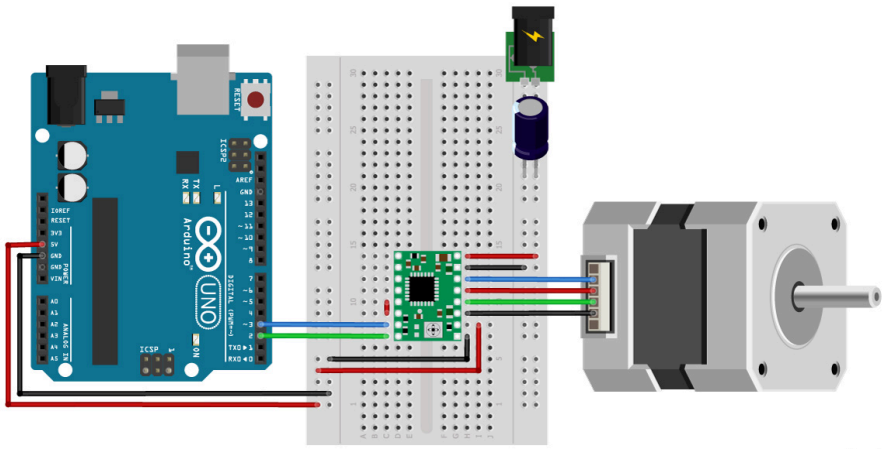
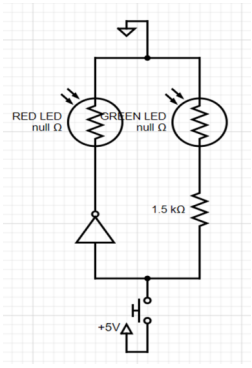


Figure 13: Breadboard Schematic [3]

Endstop:

Endstop provides an opportunity to make Barmaid more visually appealing by installing LEDs and a simple NOT gate, a digital circuit built with a NPN transistor as shown in the diagram below. Notice in the circuit below one of the LEDs is always on, refer to the truth table for the states of the LEDs and platform.



Endstop input	Green LED	Red LED	Platform state
LOW	HIGH	LOW	Driving
HIGH	LOW	HIGH	Not driving

Figure 14: Endstop and NOT gate circuit. Truth table for the circuit.

UI Box:

To interact with Barmaid's system, the user communicates with a liquid crystal display (LCD) and a set of push buttons mounted on the UI box. The display is connected to a module which decreases the number of pins it occupies on the Arduino Uno by using the I2C communication protocol. This module only requires two pins on the Arduino, SDA (serial data) and SCL (serial clock), to display, instead of the default 8 data pins on the LCD. The buttons are wired with pull down resistors so that they can assert the default readings of the buttons as LOW on the microcontroller. This will prevent the Arduino from misreading the state of the button.

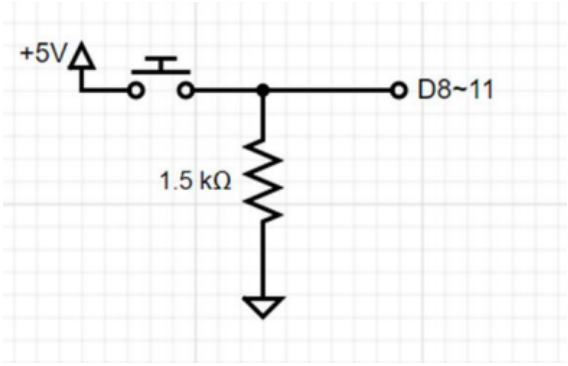


Figure 15: Pull-down resistor circuit

Software:

Weight Sensor:

In order to integrate the load cell into Barmaid, one needs the HX711_ADC.h library to use the HX711 module and EEPROM.h library. The load cell must be calibrated initially before it is ready to use, which is done by running the calibration code. The user must first set the tare offset, meaning resetting the sensor to read a value of 0. Next, an object with a known mass is placed and its weight value is sent into the serial monitor of Arduino. Lastly, the code will compute the calibration value and store it in the Electrically Erasable Programmable Read Only Memory (EEPROM), so one can refer to that value in a different program without the need to recalibrate the load cell.

It is important to notice that the weight is only measured when the Arduino is told to, meaning during delays, interrupt service routine (ISR), loops, and etc, the load cell is not doing anything useful. This compromise is overcome by running all electronics sequentially so nothing is turned on at the same time. The aforementioned software architecture is reflected in further sections (Figure 21).

Conveyor:

For the motor, several libraries to control the motor were explored, but only one fits our goals, which is the AccelStepper library. What makes this library so desirable is the fact that the motor knows exactly where to move by calling certain functions.

Here is a basic flowchart of how the code works:

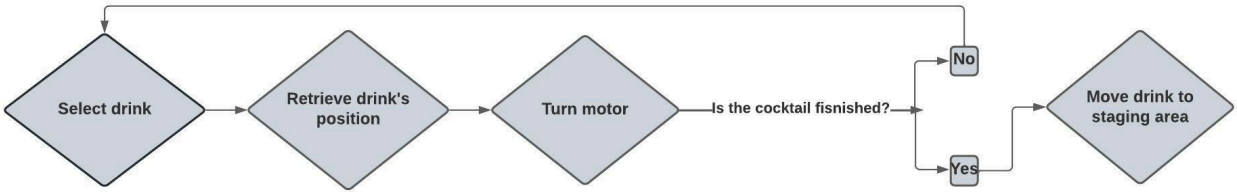


Figure 16: Motor Logic Flowchart

In order to understand how the motor turns to reach the desired position, it is important to understand how the motor is configured. In brief, the motor has 200 steps per revolution, meaning that if we told the motor to move to position 200, it would make one full revolution in a clockwise direction (-200 for a full revolution counter-clockwise). Using this information, we can calibrate the positions of each station by stepping the motor by x steps to center the platform with its respective valve. Once this is done, this x value becomes the positional value for the station. Each station will be assigned one of these values, so for example, if vodka was assigned a position of 200, once the user specifies that they want a drink with vodka, the motor will step 200 times to reach the vodka station. This process will continue until the drink is finished. The platform will then return to the origin, or a staging area (yet another hard-coded positional value for the motor), where it will remain.

User Interface:

The display is controlled using the Liquid Crystal I2C library. This library has several functions that allow it to easily print messages to the screen. To interface with this display, the push buttons next to the LCD are continuously checked, and when an input is detected, the screen changes accordingly.

When the system is booted up, the Menu is initially displayed, as shown in the figure below:

Figure 17: Menu Display

The user then interacts with the various push buttons to either scroll the screen or select an option.

The buttons on the left side toggle the menu up and down. The way this is done is by incrementing and decrementing a count variable that corresponds to the various options. This variable is initially set to 1, which prints out the default menu as shown in figure 17, with a cursor on option 1. If the Down button is pressed, the count will be incremented by one, and a new message will be printed on the screen with the cursor set to the current count. The Up button works the same, but in the opposite sense, as it decrements the count when pressed. To account for the boundary cases, the code sets up a blockade if the user is at either count 1 or count 4 (since there are only four options configured), which ensures that the screen cannot be scrolled any further than is allowed.

As for the right side, if the cancel button is pressed while the menu is displayed, nothing will happen. This is done to ensure that no extraneous button inputs are being checked for. When the user presses the select button, the following message will be displayed

Figure 18: Confirmation Message

The number seen in the display is linked to the aforementioned count variable, so if the current count is at 3, the screen will display “Select Option 3?”. From here, the program will only detect inputs from the Select and Cancel button. If cancel is pressed, the display will return to the menu (Figure 18), and the buttons will all be checked again. However, if the Select button is pressed, the LCD will flash a preparation message and the drink will be made.

Figure 19: Preparation Message

Once the drink is finished, a completion message will be displayed for a total of three seconds, and the code will re-enter the loop and display the menu once again.

Figure 20: Completion Message

Logic and Integration:

The way the program integrates all the subsystems is by calling on the necessary functions from the various subsystems during drink preparation. The way this is accomplished is by nestling these functions within the UI code. More precisely, the functions are called while the message “Preparing Drink” is displayed (Figure 19). The following page contains a flowchart of the logic.

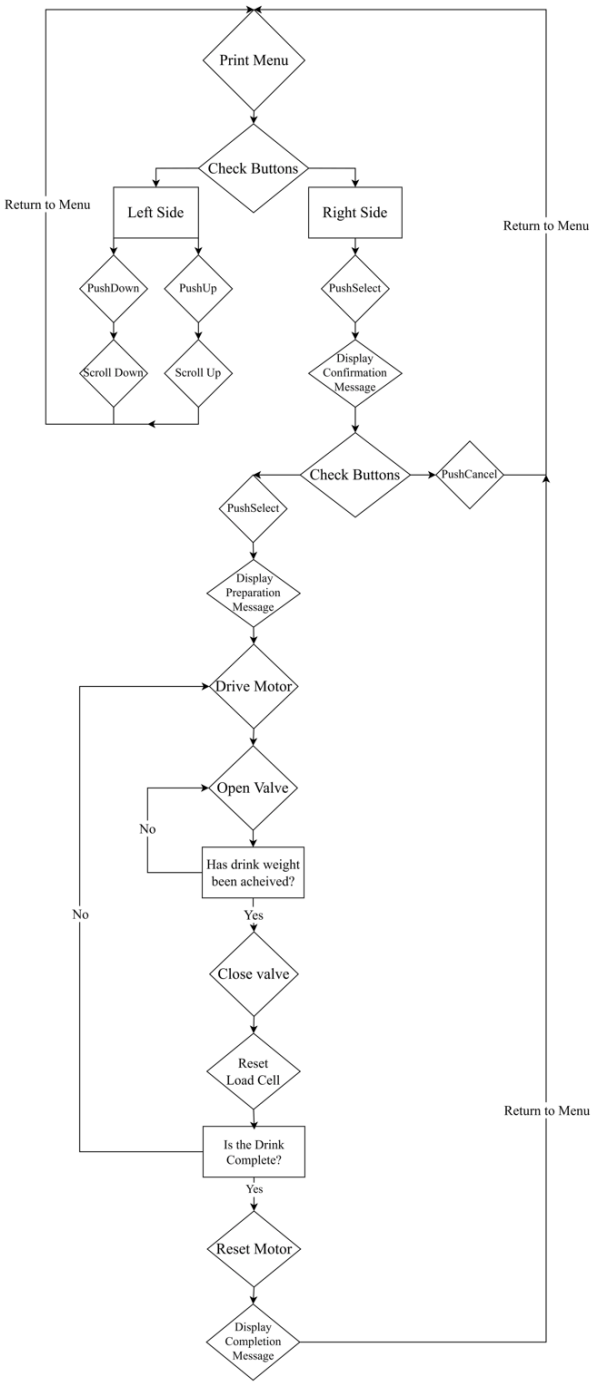


Figure 21: Barmaid Flowchart

As shown, once the drink option is selected and confirmed, a pre-programmed sequence will be executed. In short, the platform will move to the desired drink station, and it will pour a specified amount of liquid by opening and closing the required valve; the operation of which is controlled by the load cell. This process continues until the drink is complete, and the platform returns to the staging area.

Verification and Validation

Verification:

System Requirements:

1) Drink must be within 10% of the desired volume
2) Longest drink must be complete within 30 seconds
3) User Interface must be easy to navigate

To test the first requirement, the load cell is configured and takes a sample measurement of liquid. This number is recorded and compared to the measurement from a weight scale. This process is repeated five times, and the load cell measurements were found to be within +/- 3% of the actual weight of the liquid. This variance shows that the specified drink can be consistently made to a high degree of accuracy. This process also removes the monotony of measuring exact quantities by hand, as Barmaid takes care of it for the user.

As for the second requirement, the longest drink currently configured takes a 10 gram sample from the fifth bottle and a 10 gram sample from the third bottle. In total, this process lasts under 25 seconds. This short time frame further shows that Barmaid is comparable to a hand-made drink made by a mixologist, as both the quantity of liquid and time for the drink to be prepared have been optimized.

The user interface is also straightforward to use. The team experimented with multiple button combinations and found that the fewer buttons there are, the easier it is to understand what is going on. To address this, a total of four buttons were used.

Validation:

By achieving the aforementioned system requirements, Barmaid becomes a product that can benefit the stakeholders. A test to demonstrate the autonomy of Barmaid is conducted by programming it to pour multiple drinks into a cup with a push of a button. As a result, users are able to autonomously make cocktails with buttons and without the need to supervise the entire process. The automated dispensing mechanism displayed during this test implies that stakeholders such as the owner of bars and restaurants are able to alleviate their busy work hours by letting Barmaid make drinks. Furthermore, the process of serving drinks can be entirely automated and may even get rid of the need to hire anyone; as a result, the business may see an increase in their revenue stream.

On a more qualitative side, Barmaid is designed and decorated with a saloon theme in an attempt to look appealing to the users. The testing method we employed to validate this is by surveying people who were interested in Barmaid at DAID and recording their comments. Many people enjoyed the theme Barmaid envisioned initially and it is safe to say that the visually appealing saloon concept is validated.

Conclusions and Future Work:

Barmaid is developed with the intention to deliver the joy and the feelings of being in a real bar, where the user only has to request the type of drink and Barmaid makes it. Barmaid has reached her final testing and is proudly fulfilling her mission of making cocktails.

Throughout this project, we have encountered numerous problems and overcame all of the ones that are within our control; however, there is still some future work that can improve Barmaid:

1. One can improve the dispensing mechanism by using peristaltic pumps to accurately and consistently pour drinks
2. One can make the process of bottle replacement better by installing universal caps to accommodate for wide range of drinks
3. One can make Barmaid bluetooth compatible to control remotely or app compatible
4. One can add more bottles to increase the variety
5. One can add a station for dispensing ice
6. One can turn Barmaid into a vending-machine-like by accepting payments for a drink

References:

[1]J. Fassl, “Advanced weight cell tech boosts efficiency in packaging uses,” *ProFood World*, 12-Apr-2021. [Online]. Available: https://www.profoodworld.com/automation/process-instrumentation/article/21378156/advanced-weigh-cell-tech-boosts-efficiency-in-packaging-uses. [Accessed: 21-Feb-2022].

[2]Bramm, “Use load cell setup without tare on each use,” *Arduino Forum*, 07-Jan-2021. [Online]. Available: https://forum.arduino.cc/t/use-load-cell-setup-without-tare-on-each-use/690267. [Accessed: 21-Feb-2022].

[3]“A4988 Stepper Motor Driver with Arduino Tutorial (4 examples),” *Makerguides.com*, 28-Sep-2021. [Online]. Available: https://www.makerguides.com/a4988-stepper-motor-driver-arduino-tutorial/. [Accessed: 21-Feb-2022].

[4]“structural engineering - How to plot bending moment diagram from shear force diagram,” *Engineering Stack Exchange*. https://engineering.stackexchange.com/questions/38104/how-to-plot-bending-moment-diagram-from-shear-force-diagram

[5]admin, “Solenoid valve control using arduino,” *Mechatrofice*, Dec. 05, 2017. https://mechatrofice.com/arduino/solenoid-valve-control

[6]“bi-directional flyback diode for relay spike protection,” *Electrical Engineering Stack Exchange*. https://electronics.stackexchange.com/questions/204332/bi-directional-flyback-diode-for-relay-spike-protection (accessed Apr. 07, 2022).

[7]“Getting Started with Load Cells - learn.sparkfun.com,” *learn.sparkfun.com*. https://learn.sparkfun.com/tutorials/getting-started-with-load-cells/all

[8]A. B. de Bakker, “A4988 Stepper Motor Driver with Arduino Tutorial (4 Examples),” *Makerguides.com*, Feb. 11, 2019. https://www.makerguides.com/a4988-stepper-motor-driver-arduino-tutorial/ (accessed Apr. 07, 2022).

Appendix:

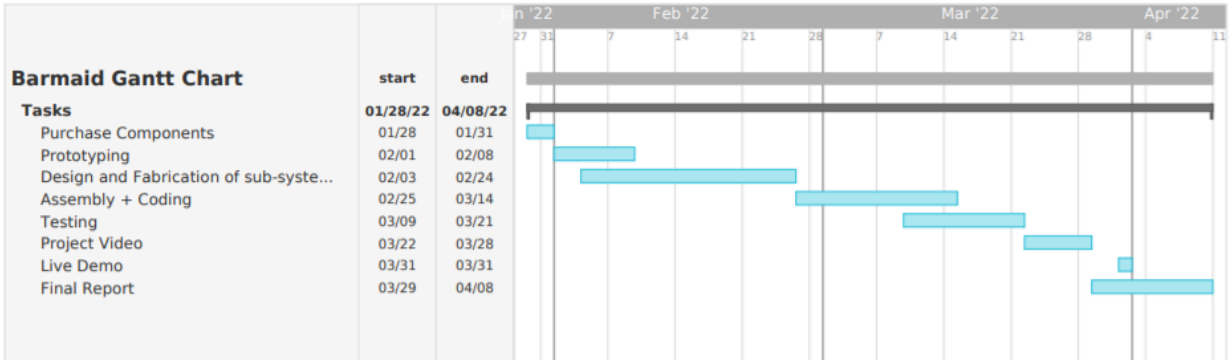


Figure A: Gantt Chart timeline

A	B	C	D	E	F	G
Component Name	Price	Sub-system	Total Sub-system Cost		Total Spent	Remaining budget (including misc)
Load cell + module	$13.99	Weight Measuring	$14.69		$207.68	-$7.68
Stepper motor	$15.99	Conveyor belt				
timing belt + pulleys	$15.99	Conveyor belt	$35.68			
motor driver	$17.99	Conveyor belt				
2pcs 4 Channel Relay Module	$10.49	Valves	$46.16			
Solenoid valves (6 units) (HALF I	$35.67	Valves				
DC 12v adapter	$14	Power	$13.95			
Push Buttons	$10.37	UI	$10.37			
Plastic Tubing	$9.83	General	$9.83			
Extra valve	$12	Valves	$12.00			
Extra Load Cell +Motor Driver	$65.00		$65.00			
inconvenience Fee	10					

Figure B: Budget

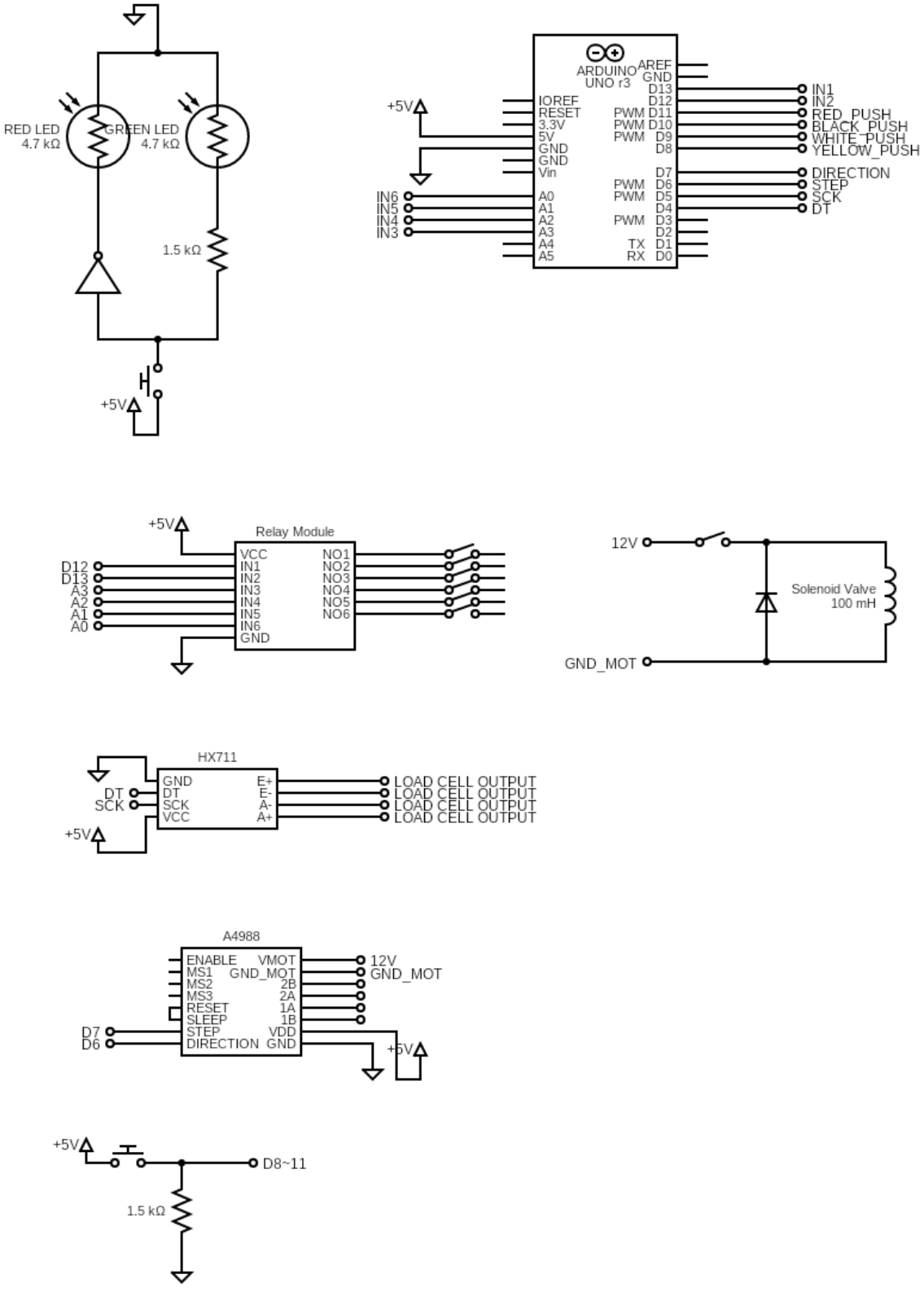


Figure C: Overall Electrical Circuit Diagram

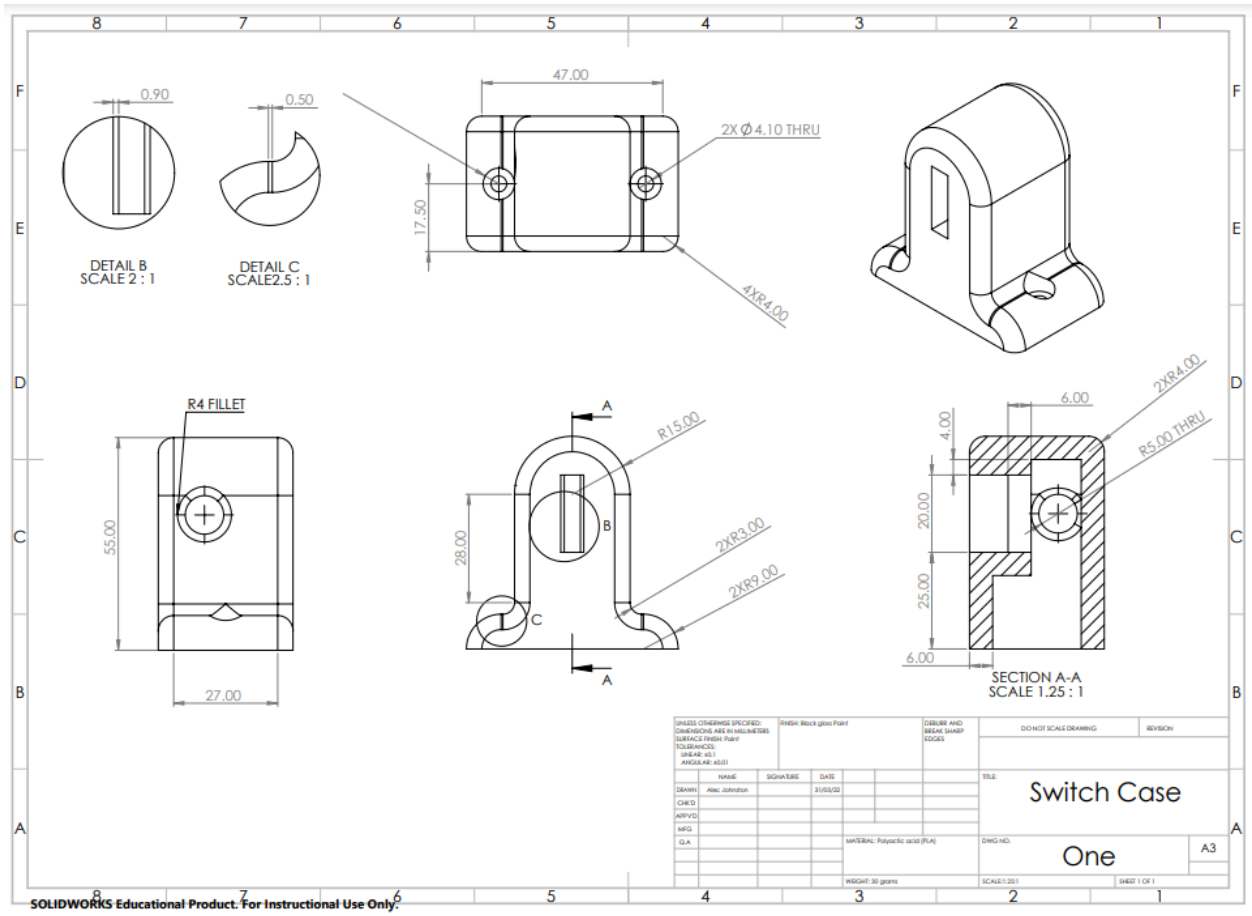


Figure D: Switch Case Drawing

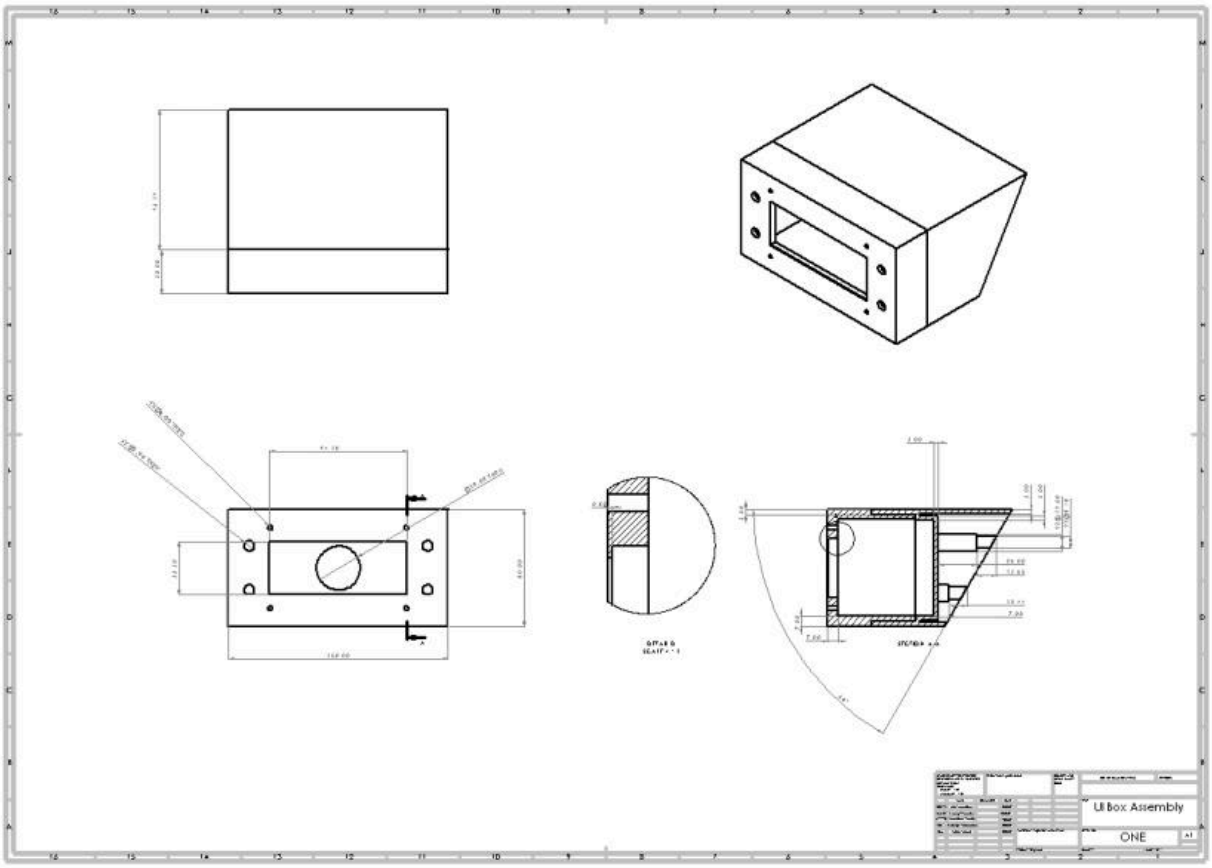


Figure E: UI Box Drawing