Micro Manufacturing Beverage System

Parke Novak, Ryan Burns, Eric Velez, Lance Adler

Dept. of Electrical and Computer Engineering
University of Central Florida, Orlando, Florida, 32816, United States of America

***Abstract* — The object of this project is to build an automated beverage filler for small-scale manufacturing purposes. The system will also verify certain parameters regarding the filling process, and will confirm the weight of each receptacle via a load cell. This system will have a user-friendly graphical user interface to control and monitor all parameters.**

***Index Terms* — Beverage, Small-scale, Manufacturing, Pouring, Sensor, Concentrate**

I. Introduction

The beer brewing industry has rapidly transformed over the 30 years. In 1988 there were 124 breweries in the U.S., now there are more than 7000. To accommodate for this massive change in the industry, technology has been developed to allow breweries to brew and can product on a small scale. Recently, the shift toward small-batch production began carrying over to beverages as well, such as kombucha and soda. Some of the technology that has been developed surrounding the beer industry, namely the filling equipment, does not translate to the small-scale soda industry.

When filling a beer can, product is pulled from a tank and filled to a certain volume by the equipment, the only calculation that is done by the software is the amount of time the valve needs to be open to fill the can to a certain volume. Soda is made in a different manner. Soda contains two parts: carbonated water and concentrate (which contains all the ingredients in a concentrated form). When filling the soda into a can, the two parts must be portioned out correctly and filled to the correct volume.

The pouring must be done accurately, which makes valve choice an extremely important factor.
Many valves could not be considered for this project because of their susceptibility to clogs. Most valves are built to hande mediums with no impurities, such as water. The concentrate that must be handled during soda manufacturing contains undissolved solids and is also much more viscous than water. The only valves that were available on the market to handle such a medium cost upwards of $500 per valve, which is far too expensive for this project. Because of monetary constraints, we chose to build our own servo-controlled ball valves for this project. This type of valve is not susceptible to clogging, can achieve an accurate pour, and also costs far less than what is available on the market.

II. System Housing

A. Filling Unit Housing

The filling unit is a rectangular unit that is fixed to a table at the bottom and has a guide on each side (so that it may easily move up and down). The housing will have 6 fill heads coming from the body, which will have the ability to fill 3 receptacles at once. Each receptacle requires a carbonated water line and a concentrate line. When the unit is filling cans, it moves to the lowest position so that the fill heads are nearly touching the bottom of the inside of the can. Filling from this position reduces foam. Once the fill is completed, the unit moves upward so that the user may remove the cans. To move the unit up and down, a large linear actuator is fixed between the top of the housing and the table.

B. Liquid Dispensing

The filling unit will pull liquid from two different sources (carbonated water and concentrate). Both liquids will be held in food-grade kegs in a cabinet below the filling unit housing. The cabinet will be on wheels to allow for easy transportation for the demonstration. ¼ inch tubing will connect the kegs to the valves and fill heads.

The tubing used is food-grade vinyl. The cabinet will be made out of plywood.

III. Core Components

A. SAM3X88E ARM Cortex M3

The microcontroller used for this project is the SAM3X88E ARM Cortex M3. It has 512kb of flash memory, has a clock speed of 84Mhz, and must be supplied with a voltage source between 1.62V-1.95V. When the microcontroller is on it has a current draw of 700uA. There is 100kb of SRAM (broken up into a 64kb and 32kb banks) that can be used on the microcontroller. The one downside of this microcontroller is that it doesn't have EEPROM simulation so an SD card must be used to store user data, logs, and presets since if the project is powered off and if the data were stored in flash/sram it would be lost. There are over 100 i/o lines which allows for everything in this project to be controlled.

B. Valves

We chose to build our own servo-controlled ball valves for this project. Ball valves themselves are purely mechanical devices that require relatively low torque to turn off/on. Ball valves are exceptional candidates for this project, they can handle almost and kind of media and have a low cost. However, the major issue with ball valves is that they are strictly mechanical devices. To electrically control the valves, we connected servos to each valve. ¼ inch tubing will be used to connect one end of the valves to the fill heads and the other end to the kegs.

C. Servo Motors

Servos have many gears which allow them to generate a high torque output. They typically rotate slowly, and usually have an axis of rotation limited to 180 degrees or less. Due to the many gears, they can make precise movements. Servos also provide positioning feedback to the software, which make them ideal for this project. Since the torque and positioning required for this project are not substantial, servo motors will be able to do the job.

The servos will be controlled using the NXP PCA9685 chip. This chip simplifies the control of the servos and localizes all components to one area of the PCB. The PCA9685 communicates with the MCU via I2C protocol, and controls the servos with various output pins. Using this chip diminishes the number of pins used on the MCU and also simplifies the code.

D. Linear Actuator

The linear actuator in this project is used to raise and lower the filling station. This is required in order to fill the beverage from bottom to top – while being submerged in the liquid. The main purpose of this is to reduce foam and also to conserve carbon dioxide. If the beverage was not filled in this way, the liquid would make more contact with the surrounding air. This leads to a higher air content in the fluid (which leads to foaming) and more carbon dioxide loss.

The station must be able to go between the UP and DOWN positions quickly. The PA-15 linear actuator is used in this project for its high speed and high output force. The PA-15 inputs 12V power and moves at 3.15" per second. The high speed of this actuator allows the user to efficiently produce product.

E. TFT Touchscreen Display

Touchscreens are a major part of the modern world, especially with relation to User Interfaces and Interactivity. As such, it was deemed important that the project make use of a touchscreen system, both to limit the number of input buttons on the board and to modernize the machine.

The touchscreen used in this project is the 7" 40-pin TFT Display, which allows for a 800x480 pixel picture. The touchscreen, however, is useless on its own, as it needs to be connected to a driver board for proper use. The RA8875 Driver Board for 40-pin TFT Touch Displays was utilized to correct the issue.

One of the major issues the team found itself in was that the board itself was resistive rather than capacitive; this lead to certain issues with touch recognition, specifically where the touch occurred. This is explored later in the software section of this report.

F. SD Card Reader

The SD Card Reader is needed for storing logs and presets; in the event of a power outage, this memory would be lost, as the SAM3X88E ARM Cortex M3 does not have EEPROM. As the logs and presets are important information that would be desirable to be maintained through power outage, an SD Card Reader was considered to be the safest option.

The SD Card Reader chosen was the SunFounder SD Card Reader for Arduino systems. The SD card chosen has a limit of 16 GB of memory; the reason for this was two-fold: the extra space would allow testing to occur without any potential problems regarding memory limitations and/or memory leaks making the card unsalvageable, and the fact that it was the smallest SD card that could be found without special ordering.

IV. Sensors

A. Load Cell

This project utilizes a load cell to measure the weight of the receptacle before and after a fill is completed. This is done to ensure that the fill was done correctly, and the data is compared with the data that is on file in the software. The load cell used in this project utilizes a half-bridge wheatstone bridge for reliability.

B. Temperature Sensor

Two temperature sensors are used in this project to monitor water temperature. Both the carbonated water and concentrate temperature must remain between 0 degrees and 5 degrees celsius during operation. If the fluids are too warm during filling, too much carbon dioxide will escape resulting in a flat beverage. Two DS18B20 waterproof sensors were used for this project. The sensors communicate with the MCU via a digital I/O pin and utilize the maxim single wire protocol for communication.

V. Power

A. 12V power

The 12V power is supplied using a common 12V wall supply. The supply that we chose is BMOUO 12V 30A supply. The supply inputs 120VAC and will supply the entire project, and will connect directly to the 12V PA-15 linear actuator.

B. 5V Power

The TPS56637RPAR from TI is used to step down from 12V to 5V. The buck converter can supply up to 6A. The converter will supply the servos and the display.

C. 3.3V Power

The TPS62125DSGR from TI is used to step down from 12V to 3.3V. The buck converter can supply up to 300mA. The converter will supply all ICs on the board and also power all data pins.

VI. Software Components/Details

A. Standard Tools

The code used predominantly for this project was completed in C++. Much of the design was based on the Arduino Due, which provided an excellent starting point for testing and coding design. Much of the work was completed on the Arduino IDE, as it allowed for relative ease with coding and testing on an Arduino design board.

B. User Interface

The project makes extensive use of a user interface for determining what the user wishes to do. The flowchart in figure 1 demonstrates how the user interface pages interact with each other.

The making of the user interface was completed using the TFT library as well as the RA8875 library. The TFT library is a standard Arduino library for help in designing and creating images and events on a touchscreen interface. The RA8875 library is simply the library for the particular touchscreen the project uses. The TFT library assisted in creating pages and screens for interaction, but was not directly designed for multi page setups; this had to be coded directly for the project itself.

The user interface will make use of the RA8875 Touch Screen mentioned above. Buttons were designed to move between the pages. Several sensor and input checks were designed such that the user may be prevented from moving between pages, specifically if there is an invalid

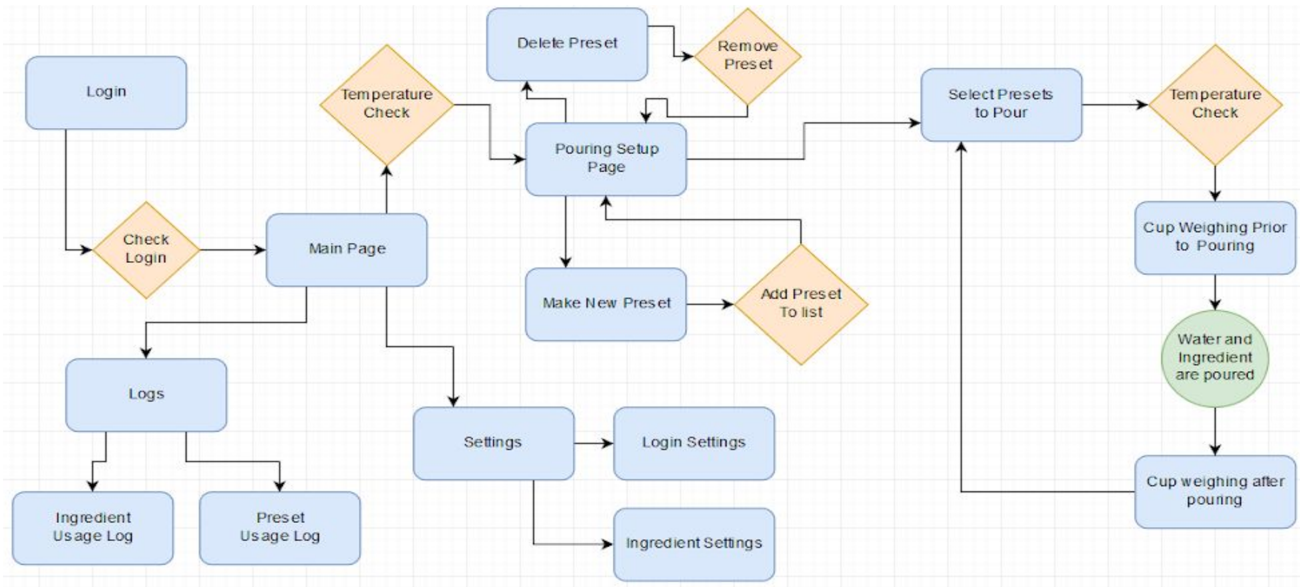

Figure 1: User Interface Flow Chart

input. This is to prevent possible malicious and/or unsafe use.

To begin, the user must log-in by input a Username and Password; this is checked against pre-existing usernames and passwords. If valid, the user is allowed to enter into the main page. From the Main page, the user may go to the Logs page, which holds the logs for usage, the Settings Page, which allows the user to change the settings of the brewing machine, and the Pouring Setup page (after passing a temperature check).

Entering the Logs page allows the user to see how much of the ingredient and water was used in production, as well as how many times each preset was used. This allows the user/manager of the machine to determine if improper usage of the machine has occurred, as well as a way to check inventory.

Entering the Settings Page allows the user to change the Log-in settings for this user, as well as change the ingredient settings (namely, the name, density, and safe temperature of the ingredient). This allows the user to ensure that security is kept and that the machine is working within appropriately.

Entering the Pouring Setup Page requires the user to pass a "Safe Temperature" check. This ensures that the system is valid for brewing, as incorrect temperatures can mean unsafe ingredients for brewing. Once on the Pouring Setup Page, the user is capable of Making or Deleting a Preset, which will update the Preset List as needed. The current limit on Presets is 12; It is assumed that most users will be able to determine whether a preset is valid long before reaching that limit, and delete the unneeded presets. Alternatively, the user may move onto the Preset Selection page.

Once on the Preset Selection Page, the user may select 3 different presets (or the same preset 3 times). Upon selecting the presets, the UI will check the temperature sensor again to determine if the ingredient is still in the safe zone; this is again to ensure that the brewing process is safe. After this, it will go to the Weighing and Pouring pages.

Immediately prior to brewing, the user will have to weigh the 3 cans while they are empty. This is for the weight checking later on (discussed more in the Math and Equations Used section). The cans are then placed on the pouring station, and the system will pour the appropriate mix of ingredient and water. After this, the user will be asked to weigh the cans (now full) again. The weight sensors will determine whether the weight is within the appropriate tolerance for the beverage. Should the beverage be valid, a green circle will be displayed; should the beverage be invalid, the user will receive either a red minus or a red plus, detailing whether the beverage is under- or overweight. Should the user find multiple reds of a similar sign, it is indicative that the system is outputting the beverage incorrectly; if there are three red minus signs, then the system is likely working with an incorrect density value for the liquid.

C. Touchscreen Calibration

Due to electrical faults from components being too close together, or mechanical misalignment caused during manufacturing, a touchscreen's input is often faulty and needs calibration. An example of the difference between the reported input and the true, intended input is shown in the figure below.

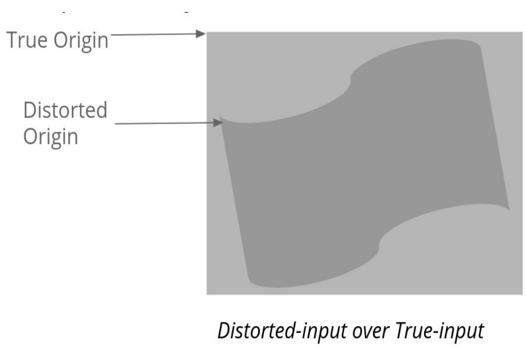

Figure 2: Touchscreen Distortion

So the aim of calibration is given some distorted input, transform it to the true origin. The algorithm we went with for calibration is the Classical Three Point Algorithm.

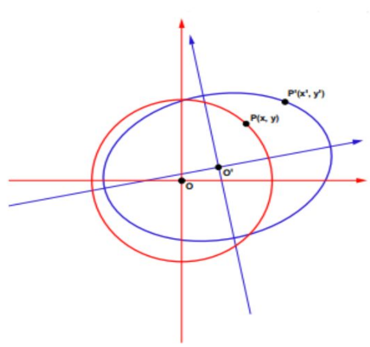

Figure 3: Input Map

The aim is to approximate the shape of the distortion map by a circle which is assumed to have been rotated, scaled, and shifted from the true origin.

Let there be three known points P1, P2, and P3. Displaying their positions on the touchscreen and and allowing the user to touch those points generates distorted inputs P1', P2' and P3'. We can then generate values for the following equations

$$\begin{cases} x = KX_1x' + KX_2y' + KX_3 \\ y = KY_1x' + KY_2y' + KY_3 \end{cases} \quad (1)$$

Where x and y are the true values, x' and y' are the read values and the coefficients are the calibration constants and are six unknowns, K being a value we provide. Using the data of the three points, we generate six equations with six unknowns and solve for the coefficients. Now given any future inputs we can use this equation to approximate where the users intended click is located.

D. Virtual Keyboard

Since we have a touchscreen and we require input from the user, we've implemented a virtual keyboard. We use the TFT library for our draw functions which includes the rounded rectangle shape. Next we create a series of constants to scale our keyboard into different sizes and allow for key detection.

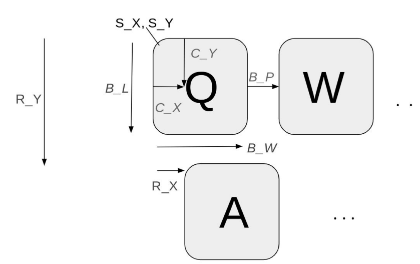

Figure 4: Button Layout

Then given some input we can detect the index of key that was pressed.

$$row = \frac{y - S_y}{R_y} \qquad column = \frac{x - S_x - row \cdot R_x}{B_w + B_s}$$

$$index = \sum_{i=0}^{row-1} (CharactersInRow_i) + column \quad (2)$$

Note that the order of characters on the keyboard is stored in a character array so the index is all we need to know which character was pressed.

E. SD Card/Memory Usage

We have an object of the user and all of their presets in main memory and seek long term storage in the SD card. We construct a function capable of storing the user and preset objects as shown in Figure 5.

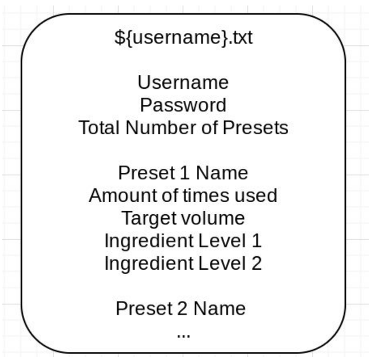

Figure 5: User File Layout

Once the data is stored, we can retrieve it line by line to instantiate a new object for use on the next bootup.

F. Valve/Motor Control

Valve and motor control was designed based on a singular observation: there are only 4 phases for each valve on the project. The phases are as follows: a) the valve is closed, b) the valve is turning open, c) the valve is completely open, and d) the valve is closing. Because of this, it was determined that motors will have a specific "fully open" setting and a "fully closed" setting.

The Arduino PWM Servo Driver library allowed the project to safely determine a specific max value and a specific min value. These values then allow for the project to simply make a "turn" order to the servos, then have the users wait until it has reached the final position. To wit, this means that the deciding factor for motor control is effectively time. The project makes use of delays for this to occur; this is both because nothing else should be happening while the motors are turning, and because it allows for precise timing of the motor control to occur.

To control the time, we use a combination of the time and amount formulas found in the "Math and Equations Used" section below. In essence, the time is found by taking the amount of ingredient/water poured into the cup, subtract out the opening and closing values on pouring (this is a roughly constant amount), and then determine the time need based on flow rate of the ingredient/water. From here, the time is input into the system, which determines the appropriate time to send a "close valve" signal.

On the coding side, time is designed around delays. It was determined that .1 second delays would occur while waiting for alternative instructions; this was used because it allowed easy testing to occur, as well as allowed for an adjustable flow rate while testing occurred. Delays were used because the PWM Servo Driver Library is designed to only send "turn" instructions then have the motors hold their positions until an alternative "turn" instruction is sent. In this way, a delay does not affect the movement of the servos; rather, it is how the servos are capable of moving.

VII. Math and Equations Used

The project made use of the following formulas for ensuring operational efficacy.

A. Weight Formula

To calculate the weight of a can when filled with the proper beverage, the weight of the can while empty (W_{empty}), the density (d) and the amount (A) of both water and ingredient are needed. While the majority weight may come from the liquid, it is necessary for the empty can weight to be calculated to ensure that incorrect weight is the result of a failed liquid pouring rather than the can being slightly heavier than anticipated. The equation for finding the full weight of a can is the following:

$$W_{Full} = W_{empty} + (d_{Water} * A_{Water}) + (d_{Ingred} * A_{Ingred}) \quad (3)$$

B. Amount Formula

The primary means of controlling the motors in this project is by determining the length of time for the motors to be active. Calculating the time needed for the motors to operate needs the time to turn on the motors ($t_{off->on}$), the time it remains open (t_{open}), and the time to turn off the motors ($t_{on->off}$) are added together. The equation for total time is the following:

$$t_{total} = t_{off->on} + t_{open} + t_{on->off} \quad (4)$$

Because the time to go from off->on and on->off on the motors is exactly 0.5 seconds each, the only real variable in the equation is the time the motors remain open.

C. Amount formula

As stated above, the primary variable used for determining motor operation time is the time the motor remains open. To calculate this, the project makes use of an amount formula; this needs the amount when the motors are opening (A_{start}) and closing (A_{Finish}) added together (these are roughly constant), then it requires that this sum be added to the multiplication of the flow rate for water and ingredient (Fr) by the time the motors remain open (t_{open}). The total amount formula is as follows:

$$A_{total} = A_{start} + (Fr * t_{open}) + A_{Finish} \quad (5)$$

All parts of this formula are constant (or as constant as possible outside of theory) with the exception of the time the motors remain open, which allows the project to determine how long the motors remain open.

VIII. Design and PCB

The PCB for this project looks likes like the following:

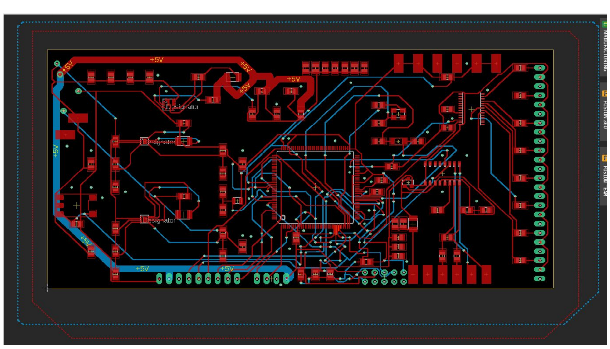

Figure 6: PCB

The pcb has 3 buck converters:

- 12v to 5v
- 12v to 3.3v
- 12v to 1.8v

The pcb inputs power from the power supply using a 12v and gnd pads which then go through the buck converts to power everything for the project. The following is the hardware diagram for the project:

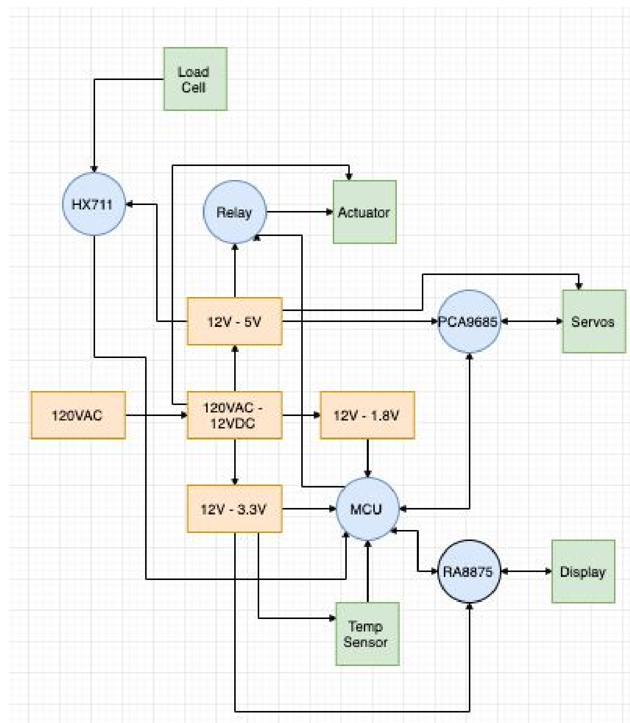

Figure 7: Hardware Diagram

The microcontroller, the crystal and all the filtering for the input of the microcontroller are found in the middle of the board. Headers for the tft touchscreen driver board, relay, and jtag are found on the bottom of the board. The jtag will allow the atmel ice to program the microcontroller.

On the right side of the pcb the servo board (PCA9685 chip) and load cell (HX711) are implemented onto the board. The header for the servo board can support up to 8 servos. There are also pads for the load cell

IX. Conclusion

These systems and components comprise the Micro Manufacturing Beverage System. This system will allow the small-batch soda manufacturing community to produce product automatically at a low price point. Implementation of the system will allow small-scale manufacturers to move away from manual production lines and join the neighboring industries in automation.

Acknowledgement

The authors wish to acknowledge the assistance and support of Dr. Samuel Richie and Dr. Lei Wei over the course of Senior Design I/II.

References

- Analog Devices, MMSE-Based Multipoint Calibration Algorithm for Touch Screen Applications by Ning Jia

The Team

Ryan Burns, a senior Computer Engineering Student from University of Central Florida with a focus on Object-Oriented Software Design and Discrete Mathematics. He intends to pursue a career in Software Engineering upon graduation.

Eric Velez, is an Electrical Engineering major that is pursuing a career in Electrical Engineering more specifically in dsp audio.

Lance Adler, a senior Student from the Electrical Engineering curriculum at the University of Central Florida. Lance looks forward to pursuing his career in control systems.

Parke Novak, a senior Student graduating from the Computer Engineering curriculum at the University of Central Florida. Pursuing a career in the computer engineering profession, specifically in User Interfaces, Robotic Vision, and Integrating Hardware/Software Ideas.