Kombucha Fermentation System
ECE 445 Final Report - Spring 2026

Project No. 27
Rudy Beauchesne, John Puthiaparambil, Edwin Xiao

Professor: Yang Zhao
TA: Haocheng Bill Yang

Abstract

Kombucha fermentation is often inconsistent in home brewing because temperature, acidity, liquid level, and visual fermentation progress are usually monitored manually. This project developed a low-cost fermentation control system that uses an ESP32-based controller to collect temperature, pH, RGB color, and ultrasonic level measurements while commanding heating, pumping, and aeration hardware. A web dashboard displays live sensor readings, actuator states, alerts, and historical trends so the user can monitor the batch with less manual intervention. Demonstration testing showed that the system could integrate multiple sensing channels, control actuators, log fermentation data, and respond to off-target or unsafe conditions through alerts and lockouts. The final prototype demonstrates the feasibility of a sensor-based home fermentation platform, while also revealing practical limitations in pH calibration, RGB color sensing, heating pad attachment, and mechanical cleaning.

Contents

1 Introduction

1.1 Problem

Kombucha is a fermented tea produced by combining sweetened tea with a SCOBY, a symbiotic culture of bacteria and yeast, and allowing the mixture to ferment over several days. During fermentation, yeast and bacteria consume sugar and generate acids, ethanol, and a cellulose pellicle. These biological processes are connected, meaning that beverage quality and cellulose growth both depend on the same fermentation environment. As shown in Figure 1, successful fermentation requires reasonably stable process conditions rather than simple passive waiting.

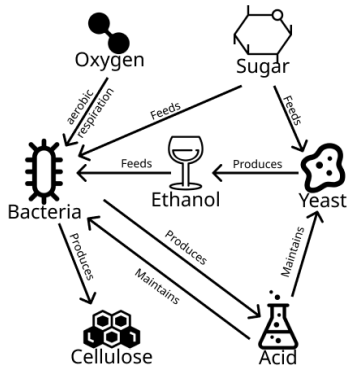

Figure 1: Kombucha fermentation interaction map

Despite the popularity of home kombucha brewing, most home setups provide little control over the variables that strongly influence fermentation behavior. Brewers often rely on a jar-based countertop process with minimal instrumentation, judging progress through taste, appearance, or rough timing estimates instead of continuous measurements. Temperature is usually left to fluctuate with room conditions, acidity is checked infrequently, and process timing is often based on intuition rather than data.

These limitations make home kombucha fermentation inconsistent. If the brew environment is too cool, fermentation may proceed slowly or stall. If the brew is too warm or left unchecked for too long, the final product may become overly acidic or inconsistent in taste. Variations in fermentation conditions can also affect pellicle growth, causing cellulose thickness and quality to vary between batches. These issues can lead to wasted ingredients, longer brewing times, discarded batches, and poor repeatability for users interested in beverage production or cellulose harvesting.

Existing solutions generally fall into two extremes: expensive commercial brewing systems that automate parts of the process, or low-cost do-it-yourself methods that still depend heavily on manual observation and intervention. Therefore, the goal of this project was to create a practical, low-cost fermentation control platform that could monitor key brewing variables, provide clear user feedback, and demonstrate corrective responses to off-target brewing conditions.

1.2 Solution

To improve the consistency and repeatability of home kombucha fermentation, we developed a low-cost closed-loop brewing system that monitors key process conditions and responds to off-target states. The system uses an ESP32-based controller on a custom printed circuit board (PCB) to collect data from temperature, pH, RGB color, and ultrasonic level sensors. These measurements are used to track fermentation conditions, support control decisions, and provide real-time feedback to the user.

The system is centered around a primary fermentation vessel connected to supporting external modules, as shown in Figure 2. The vessel contains temperature, pH, RGB color, and ultrasonic sensing elements used to track brewing conditions and visible changes throughout fermentation. The system also interfaces with a heating pad to raise the brew temperature when it falls below the desired range, an aeration pump to provide scheduled airflow, and peristaltic pumps that move liquid between the main vessel, a fresh-tea reservoir, and a waste reservoir.

A Wi-Fi-enabled dashboard displays real-time sensor readings, actuator states, alerts, and logged fermentation trends so the user can monitor system behavior without relying only on manual checks. Because fermentation occurs over multiple days, the system is designed for continuous wall-powered operation rather than battery-powered use. Figure 2 represents the system-level architecture used for the final prototype, although some physical mounting and integration details were adjusted during testing.

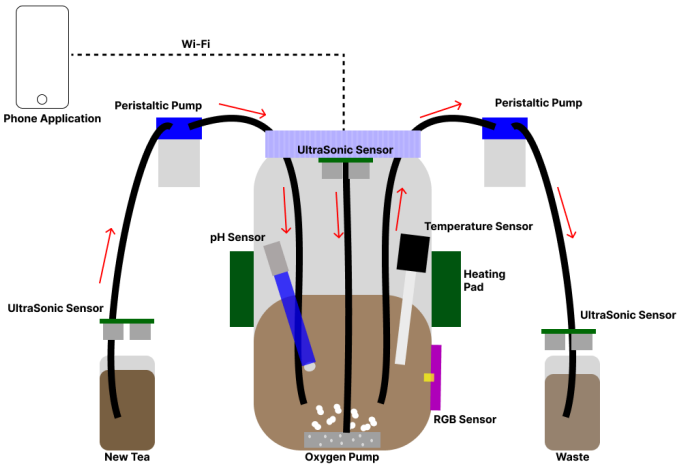

Figure 2: Visual aid of the proposed kombucha fermentation control system

1.3 High-level Requirements

Because full kombucha fermentation occurs over multiple days, project validation was performed through a short demonstration beginning from intentionally off-target conditions, including low temperature, shifted acidity, and a non-target color state. This approach allowed the system response to be evaluated within a single demonstration session rather than over a full multi-day fermentation cycle. The system was evaluated using the following high-level requirements:

- The system shall continuously monitor temperature, pH, color state (HSL), and liquid level, display these measurements on a Wi-Fi dashboard with timestamped logging, and provide user access to actuator status and control.
- For a demonstration volume of approximately 800 mL starting near room temperature, the system shall command heating and produce a measurable upward temperature trend toward a user-defined setpoint, demonstrated by at least +1.0°C change or entering the ±1.0°C band when starting within a few degrees of setpoint
- The system shall actively modify liquid acidity using pump-driven dosing and shall produce a measurable decrease in pH ≥1.5 change during demonstration testing.
- The system shall detect and report a measurable visual-state shift from an initially clear or lightly colored state to a visibly colored state using color-sensor-derived hue, saturation, and lightness (HSL) values during demonstration testing.

2 Design

2.1 Physical Design

Our physical design consists of a main fermentation jar, two auxiliary jars, external pumps, tubing, sensors, an acrylic support structure, and a separated electronics container. The main jar holds the kombucha sample, while the auxiliary jars hold fresh tea and waste liquid. Two peristaltic pumps are mounted beside the main jar on the acrylic stand: one pump adds fresh tea, and the other removes liquid from the main vessel. A separate oxygen pump provides airflow through tubing during scheduled aeration.

The temperature and pH probes are inserted into the main jar because they must directly contact the liquid. The RGB color sensor is placed underneath the jar to measure color changes through the bottom of the vessel. Ultrasonic sensors are mounted above the main jar, fresh-tea jar, and waste jar to measure liquid level without touching the liquid. A heating mat is wrapped around the outside of the main jar so the system can heat the brew without placing a heating element inside the liquid.

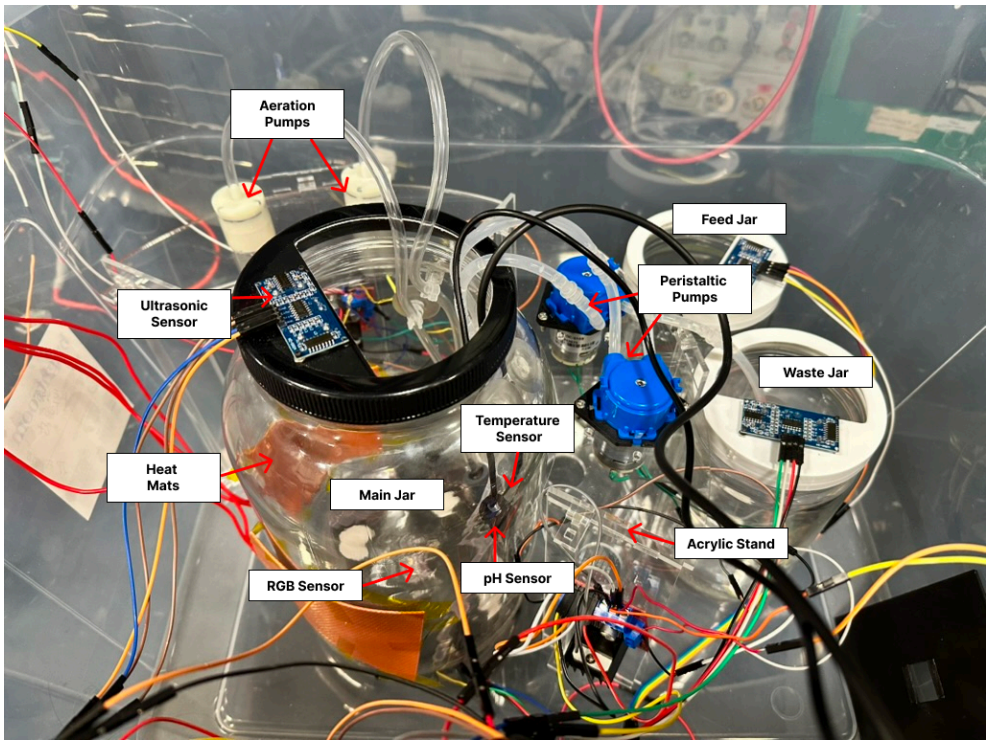

Figure 3: Early prototype of the kombucha fermentation control system

The electronics are placed in a separate container beside the fermentation setup rather than directly under or inside the wet area. This layout keeps the fluid-handling components accessible while separating the liquid path from the control electronics as much as possible. Figure 3 shows the physical prototype of the kombucha fermentation control system.

2.2 Block Diagram

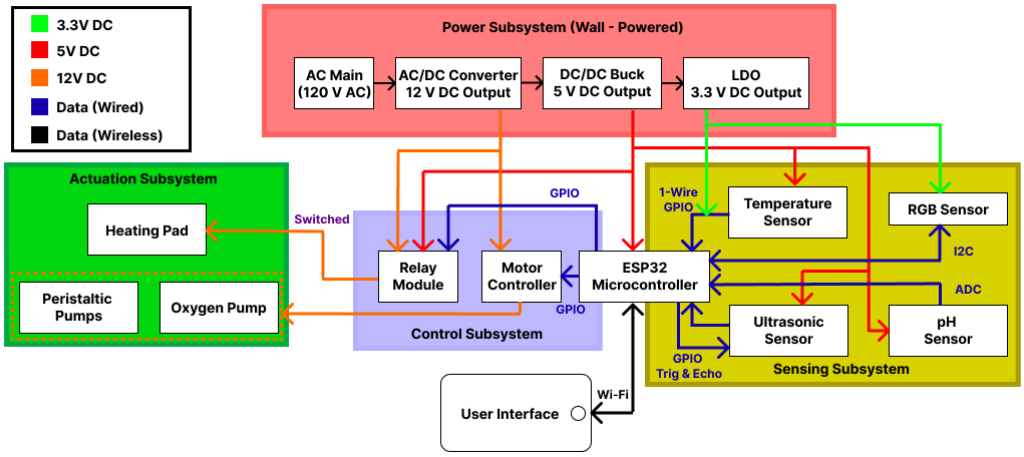

Figure 4: Kombucha Fermentation System Block Diagram

Figure 4 shows the overall block diagram of the kombucha fermentation control system. The system is divided into five main subsystems: power, sensing, control, actuation, and user interface. The power subsystem supplies the required voltage rails for the ESP32, sensors, pumps, and heating hardware. The sensing subsystem measures temperature, pH, color, and liquid level, then sends these measurements to the ESP32 control subsystem.

The control subsystem processes the sensor data and determines when to command the heating mat, peristaltic pumps, and oxygen pump. The user interface communicates with the ESP32 through Wi-Fi to display live measurements, actuator states, alerts, and logged data. This block structure was chosen because it separates measurement, decision-making, physical actuation, and user interaction into clear functional units, making the system easier to design, debug, and test.

2.3 Functional Overview & Block Diagram Requirements

2.3.1 Power Subsystem

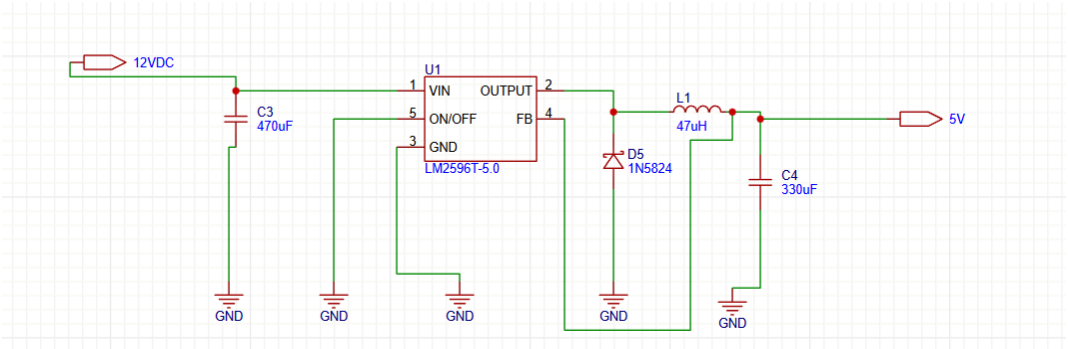

Figure 5: Power Subsystem Schematic

The power subsystem distributes electrical power to the ESP32, sensors, and actuation hardware. Since kombucha fermentation occurs over several days, wall power was chosen instead of battery power. The system uses an external AC-to-DC adapter to convert 120 V AC wall power into a regulated 12 V DC input. This keeps AC conversion outside the project enclosure and allows the internal system to operate only with low-voltage DC power.

The 12 V rail directly powers the higher-power actuation loads, including the peristaltic pumps, oxygen pump, and heating mat. This same 12 V input is stepped down using an LM2596-based buck converter to produce a 5 V rail for the ESP32 development board, sensor modules, and relay logic. A 5 V-to-3.3 V regulator then provides the 3.3 V logic rail used by the ESP32 and low-voltage sensor interfaces. All subsystems share a common DC ground so the ESP32 can reliably control the motor driver, relay module, and sensor inputs.

This power architecture was chosen because it provides the different voltage levels required by the system while keeping the design simple, low-cost, and safe for a wet fermentation environment. The 12 V rail supports the actuator loads, while the 5 V and 3.3 V rails support the control and sensing electronics.

2.3.2 Control Subsystem

The control subsystem is the central decision-making block of the kombucha fermentation control system. It receives temperature, pH, color, and liquid-level measurements from the sensing subsystem and uses these inputs to determine the appropriate system response. The controller commands the

heating mat, peristaltic pumps, and oxygen pump to help drive the system toward user-defined operating conditions during the demonstration.

The subsystem is built around an ESP32 microcontroller. The ESP32 was chosen because it provides enough GPIO pins for the sensors and actuators, includes analog input capability for the pH sensor, and supports Wi-Fi communication for the dashboard. Other options, such as an Arduino Uno or Raspberry Pi, could have been used, but the ESP32 provided the best balance of low cost, embedded control, wireless communication, and hardware interface support.

During operation, the ESP32 reads sensor data, checks for invalid readings or alert conditions, updates actuator commands, and sends status information to the user interface. The heating mat is controlled through a relay output, while the peristaltic pumps and oxygen pump are controlled through motor-driver or GPIO-based control signals. On startup or reset, the controller sets all actuators to the OFF state before beginning normal operation. This prevents unintended pumping, heating, or aeration while the system initializes.

The control subsystem connects the sensing and actuation portions of the project by converting measured fermentation conditions into physical responses. This allows the system to demonstrate closed-loop temperature control, liquid transfer, scheduled aeration, dashboard updates, and safety lockout behavior.

2.3.3 Sensing Subsystem

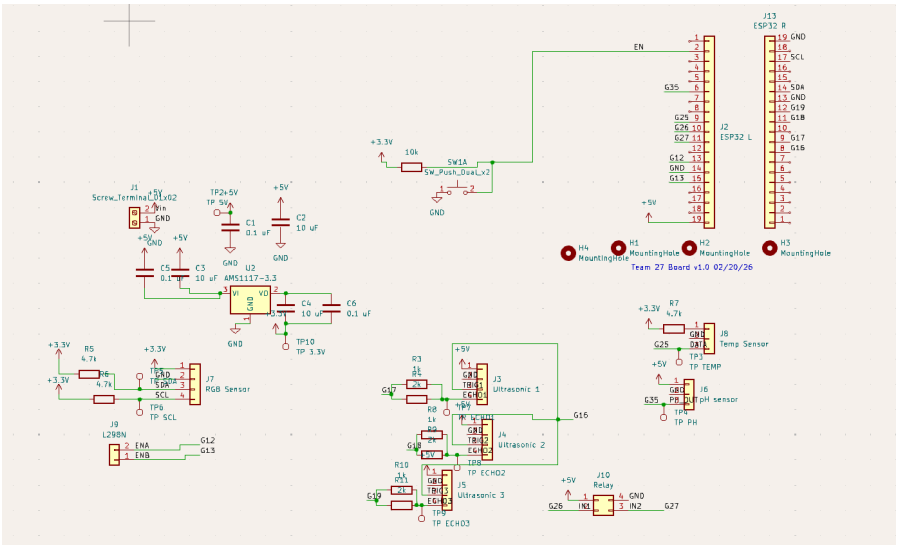

Figure 6: Sensing Subsystem Schematic

The sensing subsystem measures the brewing conditions needed to monitor fermentation and support automated control. The system uses ultrasonic sensors for liquid-level monitoring, a DS18B20 temperature sensor for brew temperature, a TCS34725 RGB color sensor for color-state tracking, and a pH probe with an analog sensing board for acidity measurement.

Several sensing methods were considered. Liquid level could have been measured using float switches, load cells, or capacitive sensors, but ultrasonic sensors were chosen because they measure level without touching the liquid. Temperature could have been measured using a thermistor, but the DS18B20 was chosen because it provides a simple digital 1-Wire interface and can be placed directly in the brew. Color could have been monitored using a camera, but the RGB sensor was simpler, cheaper, and easier to integrate with the ESP32. Acidity could have been checked manually with pH strips, but an electronic pH probe allowed pH trends to be logged over time.

The ultrasonic sensors are mounted above the main jar, fresh-tea reservoir, and waste reservoir. The main ultrasonic sensor monitors the brew level, while the side sensors help detect low fresh-tea level and waste overflow conditions. To improve measurement stability, the firmware takes five ultrasonic readings and uses the median value, reducing the effect of occasional noisy distance measurements. The RGB sensor is placed beneath the main jar to measure visible color changes through the liquid. The temperature and pH probes are inserted into the liquid because they require direct contact with the brew.

The sensors interface with the ESP32 using several communication methods. The TCS34725 communicates over I^2C, the DS18B20 uses a 1-Wire GPIO connection, the ultrasonic sensors use trigger and echo GPIO signals, and the pH board outputs an analog voltage to an ESP32 ADC input. Because the ESP32 uses 3.3 V logic, the ultrasonic echo signals are reduced from 5 V to approximately 3.3 V using resistor dividers, and the pH output is kept within the ESP32's 0–3.3 V ADC range.

The voltage divider relationship is

$$Vout = Vin \frac{R2}{R1 + R2}$$

Select R1=1kΩ and R2=2kΩ. Then the nominal scaling is:

$$\frac{R2}{R1 + R2} = \frac{2}{1 + 2} = \frac{2}{3} \approx 0.667$$

So at the worst-case input Vin =5V:

$$Vout \approx 5(0.667) = 3.33\,V$$

2.3.4 Actuation Subsystem

The actuation subsystem physically changes the brewing environment in response to commands from the controller. The system uses a heating mat to raise the temperature of the main fermentation jar, peristaltic pumps to add or remove liquid, and an oxygen pump to provide scheduled aeration during operation.

Several actuation options were considered. For heating, wall-powered heating peripherals or immersion-style heaters could have been used, but these would introduce more electrical safety concerns near the liquid. Instead, the final design uses a 12 V DC heating mat wrapped around the outside of the jar. This keeps the heating element outside the brew while still allowing controlled thermal input. For liquid transfer, gravity-fed valves or solenoid valves could have been used, but peristaltic pumps were chosen because they provide controlled flow and keep the liquid contained inside the tubing. For aeration, passive airflow could have been used, but an oxygen pump gives the controller direct timing control.

The final actuation design uses 12 V DC loads for the heating mat, peristaltic pumps, and oxygen pump. Using 12 V peripherals made the system simpler and more efficient because the same 12 V rail could power the main actuation hardware without requiring separate wall-powered devices for each function. The ESP32 does not drive these loads directly; instead, it sends low-voltage control signals to a relay module and motor driver. The relay switches the heating mat, while the motor driver controls the pumps.

This design was chosen because it keeps the actuator power architecture simple, separates higher-current loads from the ESP32 GPIO pins, and avoids placing AC-powered hardware directly near the fermentation liquid. The subsystem provides four physical outputs: heating applied to the main jar, oxygen delivery to the brew, fresh tea input, and liquid removal from the main vessel.

2.3.5 User Interface

The User Interface Subsystem provides the primary operator interface for the kombucha fermentation controller. The dashboard displays live sensor measurements, actuator states, system alerts, configuration values, manual controls, command responses, and historical graphs. This interface was chosen because kombucha fermentation occurs over multiple days, so the user needs both real-time monitoring and access to longer-term trends.

Several interface options were considered, including serial monitor output, an onboard LCD, a local ESP32-hosted webpage, and a cloud-connected dashboard. A dashboard with an MQTT-based data pipeline was chosen because it allows live system visibility, remote monitoring, historical data storage, and alert handling without requiring the user to stay physically near the device.

In the final design, the ESP32 publishes telemetry, status messages, alerts, and command acknowledgments over Wi-Fi using MQTT. These MQTT messages include temperature, pH, RGB/HSL color values, ultrasonic level data, actuator states, connection status, and fault conditions. A Python ingest script subscribes to the MQTT topics, processes the incoming messages, stores telemetry and event data in a SQL database, and forwards important alerts to Discord so the user can be notified when unsafe or abnormal conditions occur.

The dashboard uses this data pipeline to show live sensor readings, active alerts, actuator states, configuration settings, manual control buttons, and historical plots. The history page allows the user to view trends such as temperature, pH, liquid level, and color-state changes over time. The dashboard also supports batch tracking so new fermentation runs can be separated from previous data.

Manual controls are included for testing and override, but unsafe actuator actions can be blocked by alert or lockout conditions. For example, the interface can prevent pump commands during a waste-overflow condition or heater commands during a temperature sensor fault. This makes the user interface both a monitoring tool and a control interface while preserving the safety behavior handled by the controller.

3 Verification

3.1 Power Subsystem Verification

The full power subsystem requirements and verification table is provided in **Appendix A, Table A.1**. In the main verification test, the power subsystem was evaluated by measuring the system voltage rails with a digital multimeter while the ESP32, sensors, relay module, motor driver, and actuator hardware were connected. The external AC-to-DC adapter supplied the 12 V rail used by the peristaltic pumps, oxygen pump, and heating mat. This rail was then stepped down by the LM2596 buck converter to produce the 5 V rail for the ESP32 development board, sensor modules, and relay logic. The 3.3 V logic rail was also measured to confirm that the ESP32 and low-voltage sensor interfaces were receiving the correct operating voltage.

When measured with a DMM, the output voltages remained stable within approximately ±0.01 V during normal operation. The 12 V rail stayed near its expected value, the 5 V rail remained within the required range for the ESP32 and sensor modules, and the 3.3 V rail remained within the safe operating range for low-voltage logic. During combined sensing and actuator operation, the ESP32 did not brown out or reset, indicating that the power subsystem provided stable power for the full system.

3.2 Control Subsystem Verification

The full control subsystem requirements and verification table is provided in **Appendix A, Table A.2**, and the control firmware state diagram is shown in **Appendix C, Figure C.4**. The control subsystem was verified by confirming that the ESP32 could read sensor values, process system conditions, publish status data, and command the correct actuators. During testing, the ESP32 successfully received inputs from the sensing subsystem, including temperature, pH, RGB/HSL color values, and ultrasonic level readings. These values were then used to update system state, actuator commands, dashboard telemetry, and alert conditions.

Manual dashboard commands were used to confirm that the controller could switch the heating mat, peristaltic pumps, and oxygen pump on and off through the relay and motor-driver interfaces. The controller also correctly initialized all actuator outputs to the OFF state during startup, preventing unintended heating, pumping, or aeration when the system first powered on.

The control logic was also tested under alert and lockout conditions. As shown in the firmware state diagram in **Appendix C, Figure C.4**, the controller moves through sensor reading, connection checking, alert checking, actuator control, publishing, and lockout states. When unsafe or invalid states

were detected, such as a temperature sensor fault or waste-overflow condition, the controller generated alerts and blocked actuator commands that could worsen the condition. This verified that the ESP32 was able to coordinate sensing, actuation, dashboard communication, and safety lockout behavior during system operation.

3.3 Sensing Subsystem Verification

The full sensing subsystem requirements and verification table is provided in **Appendix A, Table A.3**. The sensing subsystem was verified by collecting live data from the ultrasonic level sensors, temperature sensor, pH sensor, and RGB color sensor during system operation. These tests confirmed that the subsystem could measure the main fermentation variables needed for monitoring, dashboard display, and control decisions. The collected sensing results are shown in **Appendix B, Figures B.1–B.4**.

The ultrasonic sensors were verified by tracking the measured distances for the main jar, feed reservoir, and waste reservoir over a 5-minute test period, as shown in **Appendix B, Figure B.1**. The main jar reading moved toward the configured lower-limit threshold, while the feed and waste readings changed as liquid was transferred between containers. This showed that the ultrasonic sensors responded to changing liquid levels and could support level-based conditions such as low main-jar level, feed availability, and waste accumulation.

The temperature sensor was verified by recording brew temperature over the same test period, as shown in **Appendix B, Figure B.3**. The measured temperature increased from approximately 20.0 °C to above the 22.0 °C temperature limit, showing that the sensor could detect a clear upward thermal trend during heating. This verified that the temperature sensor provided useful feedback for both controller decisions and dashboard monitoring.

The pH sensor was verified by recording measured pH during an acidity-response test, as shown in **Appendix B, Figure B.4**. The pH readings showed a consistent trend over time and followed the best-fit linear approximation shown in the figure. This demonstrated that the pH sensing circuit could detect measurable acidity changes during testing. However, calibration stability remained a practical limitation, so pH was treated primarily as a monitoring and trend measurement rather than a fast closed-loop control input.

The RGB color sensor was verified using the color timeline gradient shown in **Appendix B, Figure B.2**. The gradient shows a visible transition from a lighter liquid state to a darker tea-colored state over the test period. This confirmed that the RGB sensor could detect broad color-state changes and

provide visual feedback through the dashboard. However, color sensing was sensitive to liquid opacity, lighting, and sensor placement, so the color data was most useful for identifying broad visual changes rather than precise chemical state.

Overall, the sensing subsystem successfully provided live temperature, pH, liquid-level, and color-state measurements for the kombucha controller. The strongest sensing results were temperature and ultrasonic level tracking, while pH and RGB sensing were useful but more sensitive to calibration and physical setup conditions.

3.4 Actuation Subsystem Verification

The full actuation subsystem requirements and verification table is provided in **Appendix A, Table A.4**. The actuation subsystem was verified by confirming that each actuator physically responded to controller and dashboard commands. The tested actuators included the heating mat, feed pump, waste pump, and oxygen pump. Each actuator was commanded individually so its response could be observed without interference from the rest of the system.

The heating mat was verified by commanding it ON when the measured liquid temperature was below the configured temperature limit. During testing, the heating mat activated successfully and produced an upward temperature trend, as shown in **Appendix B, Figure B.3**. The temperature increased from approximately 20.0 °C to above the 22.0 °C limit, confirming that the heater provided measurable thermal input to the main jar.

The peristaltic pumps were verified by commanding liquid transfer between the main jar, feed reservoir, and waste reservoir. The pumps moved liquid through the tubing in the expected direction, and the ultrasonic level readings changed during the pumping sequence, as shown in **Appendix B, Figure B.1**. This confirmed that the pump hardware was able to add and remove liquid from the main vessel during operation.

The oxygen pump was verified by commanding scheduled aeration. The pump turned on and off according to the programmed timing behavior, showing that the system could provide controlled airflow without constant user input.

Overall, the actuation subsystem successfully demonstrated heating, liquid transfer, and scheduled aeration. The main limitation was mechanical rather than electrical: the heating mat did not always maintain strong contact with the curved jar surface, which reduced heating efficiency during some tests.

3.5 User Interface Verification

The full user interface and data pipeline requirements and verification table is provided in **Appendix A, Table A.5**. The user interface and data pipeline were verified by confirming that sensor readings, actuator states, configuration values, commands, alerts, and historical data were correctly passed from the ESP32 to the dashboard. During operation, the ESP32 published telemetry and status messages over MQTT, including temperature, pH, ultrasonic level readings, RGB/HSL color values, actuator states, and system fault conditions. The MQTT-based data pipeline is shown in **Appendix C, Figure C.1**.

A Python ingest script subscribed to the MQTT topics and processed incoming telemetry, alert, status, and command acknowledgment messages. The script stored sensor and event data in the SQL database so the dashboard could display both live values and historical trends. Important alert messages were also forwarded to Discord, allowing the user to receive notifications when abnormal or unsafe conditions occurred, as shown in **Appendix C, Figure C.2**.

The dashboard was verified by comparing the live displayed values with the telemetry published by the ESP32. The dashboard successfully displayed live sensor readings, actuator states, connection status, configuration values, manual control buttons, active alerts, and command responses. The history page also displayed stored sensor data over time, including temperature, pH, liquid level, and color-state trends, as shown in **Appendix C, Figure C.3**.

Manual controls were tested by sending actuator commands from the dashboard and observing the resulting command acknowledgments and actuator state changes. Configuration updates were tested by changing threshold values from the dashboard and confirming that the updated settings were reflected in system behavior. Alert display was verified by triggering fault or unsafe conditions and confirming that the dashboard showed the active alert while the ingest script sent the corresponding Discord notification.

Overall, the user interface and data pipeline successfully provided real-time monitoring, manual control, configuration updates, alert reporting, Discord notifications, and historical data visualization. The main limitation was batch organization in the database, since some telemetry points were not always assigned to the intended batch during testing. This limited the reliability of comparing separate fermentation runs and should be improved in future revisions.

4 Cost and Schedule

4.1 Cost Analysis

The total cost for parts, as listed below, is approximately $169.29 before shipping and tax. For labor, we estimate an average compensation of $17.50/hr based on a typical UIUC ECE graduate assistant monthly pay of approximately $2800/month, assuming a 40-hour work week. Using the course labor formula, this gives $17.50/hr × 2.5 × 50 = $2187.50 per team member for the duration of the project. With three team members, the total estimated labor cost is $2187.50 × 3 = $6562.50. Therefore, the combined estimated project cost is approximately $6731.79. The full itemized parts list is provided in **Appendix D, Table D.1**.

4.2 Schedule

The project schedule is provided in **Appendix D, Table D.2**. This schedule summarizes the major project milestones, including ordering parts, subsystem design, PCB development, firmware implementation, dashboard development, integration testing, final demonstration, and final report preparation.

5 Conclusion

The Kombucha Fermentation Control System demonstrated the integration of sensing, actuation, and wireless monitoring into a single ESP32-based embedded platform for small-scale fermentation monitoring and control. The final prototype monitored temperature, pH, liquid level, and color-state changes while controlling heating, liquid transfer, and oxygen circulation through a Wi-Fi dashboard interface. Demonstration testing verified the system's primary sensing, actuation, dashboard, and alert functions while also highlighting real-world engineering limitations and opportunities for future improvement.

5.1 Accomplishments

During testing, the heating subsystem demonstrated a measurable temperature response using external heating pads controlled through relay switching by the ESP32. Starting from room-temperature conditions, the system increased the liquid temperature from approximately 20 °C to above the configured 22 °C threshold during the demonstration. This verified that the controller could detect a low-temperature condition, activate the heater, and produce an upward temperature trend. Some thermal lag was observed due to heat loss through the glass fermentation vessel and imperfect contact between the heating pads and the jar surface.

The pH subsystem successfully detected measurable acidity trends and logged pH data to the dashboard during testing. While the pH sensor was useful for observing fermentation-state changes and large acidity shifts, calibration drift and probe placement limited its precision. For this reason, pH was treated primarily as a monitoring and trend measurement rather than a precise closed-loop control input.

The RGB sensing subsystem identified transitions from lightly colored liquid states to darker fermentation conditions using RGB measurements converted into HSL values. The ultrasonic sensing subsystem also provided stable liquid-level measurements across the main jar, fresh-tea reservoir, and waste reservoir. These sensing results were sufficient for demonstration-scale monitoring and level-based alert behavior.

One of the most significant accomplishments of the project was the successful integration of multiple sensing, actuation, and communication subsystems into a unified platform. The system simultaneously operated relays, pumps, sensors, Wi-Fi communication, MQTT telemetry, dashboard updates, and alert handling. Overall, the project demonstrated the feasibility of a low-cost, modular fermentation monitoring system for home use.

5.2 Uncertainties

Several limitations and uncertainties were identified during testing and system integration. Calibration drift and electrical noise were observed in the pH subsystem during extended operation, requiring periodic recalibration and software averaging to maintain stable readings. While the pH sensor was sufficient for detecting large acidity changes, it was not intended for laboratory-grade precision measurements.

The RGB sensing subsystem was also affected by ambient lighting conditions and sensor positioning. Small changes in external lighting or container placement introduced variation in HSL measurements, reducing repeatability. Similarly, the ultrasonic level sensor occasionally experienced fluctuations caused by liquid surface motion and condensation near the sensor surface.

Full-system integration introduced practical engineering challenges as well. Simultaneous operation of relays, pumps, sensors, and Wi-Fi communication occasionally caused voltage instability and transient electrical noise on shared power rails. Additional filtering capacitors, improved grounding practices, and software timing adjustments were implemented to reduce instability and prevent ESP32 resets.

Mechanical integration also required multiple revisions throughout development. Sensor placement, tubing routing, and wiring organization needed to be adjusted several times to reduce interference between liquid splashing, optical sensing, and actuator operation. These uncertainties highlighted the complexity of combining electrical, mechanical, and software subsystems into a single embedded design.

5.3 Ethical Considerations

This project was designed with safety, accessibility, and responsible engineering practices in mind, following the IEEE Code of Ethics and ACM Code of Ethics. Because the system interacts with a food product, user safety was a major concern. Although the system monitors pH, temperature, liquid level, and fermentation trends, it is not a replacement for proper sanitation, manual inspection, or commercial-grade food safety monitoring. The system does not detect mold, contamination, or unsafe microbial growth, so users must still follow safe fermentation practices.

The project also considered privacy and accessibility. The dashboard stores only fermentation-related data, such as sensor readings, actuator states, and alerts, and does not require unnecessary personal information. The system was designed as a low-cost, modular alternative to commercial fermentation systems, making monitored fermentation more accessible to students and home users.

Electrical and physical safety were addressed by using an external AC-to-DC adapter, keeping the internal system low-voltage DC, separating electronics from wet fermentation components, and placing the electronics in a separate container. Food-contact materials were selected where possible, and heater limits, sensor-fault alerts, and actuator lockouts were implemented to reduce risks from overheating, overflow, or unintended pump operation.

5.4 Future Work

Several future enhancements could increase the reliability, usability, and scalability of the system. One potential enhancement would be the addition of an insulated fermentation chamber to improve thermal efficiency and reduce environmental heat loss. Improvements to food-safe tubing integration and enclosure design could also simplify cleaning and long-term maintenance.

Future revisions could incorporate automatic fermentation-state prediction using historical sensor trends, cloud-based data storage, and machine-learning-based fermentation estimation algorithms. Battery backup functionality could also improve system reliability during power interruptions.

The RGB sensing subsystem could benefit from a closed optical sensing chamber or dedicated illumination source to improve measurement repeatability under varying lighting conditions. Additional filtering and isolation techniques could also further reduce electrical noise and improve sensor stability during simultaneous subsystem operation.

Overall, the project demonstrated a strong proof-of-concept for a low-cost embedded fermentation monitoring and control platform while identifying several meaningful opportunities for future refinement and expansion.

6 References

[1] Association for Computing Machinery, "ACM Code of Ethics and Professional Conduct," Acm.org, 2018. https://www.acm.org/code-of-ethics (accessed Feb. 27, 2026).

[2] Centers for Disease Control and Prevention, "Food Safety," Cdc.gov, 2024. https://www.cdc.gov/foodsafety (accessed Feb. 27, 2026).

[3] D. Ozkan, M. Dade-Robertson, R. Morrow, and M. Zhang, "Designing a living material through bio-digital-fabrication - guiding the growth of fungi through a robotic system," Proceedings of the 39th International Conference on Education and Research in Computer Aided Architectural Design in Europe (eCAADe), 2021. doi:10.52842/conf.ecaade.2021.1.077

[4] "dfrobot.com," Dfrobot.com, 2026. https://www.dfrobot.com/product-1110.html?srsltid=AfmBOorAB3ONc8KylDlCaaHeq7VekZsWP8UKDdHVbnwwj0zin2uspfUq (accessed

[5] "EMQX Cloud Overview | EMQX Cloud Docs," Emqx.com, 2017. https://docs.emqx.com/en/cloud/latest/?utm_source=chatgpt.com (accessed Apr. 02, 2026).

[6] "ESP32 Relay Module - Control AC Appliances (Web Server) | Random Nerd Tutorials," Dec. 18, 2019. https://randomnerdtutorials.com/esp32-relay-module-ac-web-server/

[7] Espressif Systems, *ESP32-WROOM-32 Technical Reference Manual*, Espressif Systems, Shanghai, China, 2024. [Online]. Available: https://www.espressif.com (accessed Feb. 27, 2026).

[8] F. Bell, D. Chow, H. Choi, and M. Alistar, "Scoby breastplate: Slowly growing a microbial interface," Proceedings of the Seventeenth International Conference on Tangible, Embedded, and Embodied Interaction, 2023. doi:10.1145/3569009.3572805

[9] Federal Communications Commission, "FCC Part 15 – Radio Frequency Devices," Fcc.gov, 2024. https://www.ecfr.gov/current/title-47/chapter-I/subchapter-A/part-15 (accessed Feb. 27, 2026).

[10] "Guide to Homemade Kombucha - 1st Fermentation | You Brew Kombucha," youbrewkombucha. https://www.youbrewkombucha.com/guide-to-first-fermentation

[11] "Hybrid Atelier - PROJECTS," *Uta.edu*, 2024. https://hybridatelier.uta.edu/projects/65-biomaterial-fabrication (accessed Feb. 13, 2026).

[12] IEEE, "IEEE Code of Ethics," IEEE.org, 2025. https://www.ieee.org/about/corporate/governance/p7-8.html (accessed Feb. 27, 2026).

[13] M. Nurikasari, Y. Puspitasari, and R. P. Y. Siwi, "Characterization and Analysis Kombucha Tea Antioxidant Activity Based On Long Fermentation as a Beverage Functional," Journal of Global Research in Public Health, vol. 2, no. 2, pp. 90–96, 2017, doi: https://doi.org/10.30994/jgrph.v2i2.109.

[14] Texas Instruments, *LM2596 SIMPLE SWITCHER® Power Converter 150-kHz 3-A Step-Down Voltage Regulator*, Datasheet, Rev. G, Texas Instruments, Dallas, TX, USA. [Online]. Available: https://www.ti.com/lit/ds/symlink/lm2596.pdf

[15] "Ultrasonic Ranging Module HC -SR04." Available: https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf? (accessed Feb. 27, 2026).

[16] United States Environmental Protection Agency, "Electronics Donation and Recycling," Epa.gov, 2024. https://www.epa.gov/recycle/electronics-donation-and-recycling (accessed Feb. 27, 2026).

[17] Y. Ohno, "CIE Fundamentals for Color Measurements," NIST, vol. No. 16, Sep. 2000, Available: https://www.nist.gov/publications/cie-fundamentals-color-measurements-0

Appendix A: Requirements and Verification Tables

Table A.1: Power Subsystem - Requirements & Verification

Requirements	Verification
The Power Subsystem shall accept 120 VAC ±10% at 60 Hz through an external AC/DC adapter and provide regulated low-voltage DC power to the system.	• Connect the AC/DC adapter to a standard wall outlet. • Use a DMM to verify the adapter output under normal operating conditions. • Confirm that the adapter provides the required DC input for the system.
The Power Subsystem shall provide a 12.0 V DC output within ±10% while supplying continuous current to the motor driver, oxygen pumps, heating mats, and downstream converter input during normal operation.	• Energize the external AC/DC adapter and operate the 12 V loads under representative conditions, including both peristaltic pumps, the oxygen pumps, and the heating mats as applicable. • Use a DMM to measure the 12 V rail under load. • Record the voltage and current and confirm that the measured voltage remains between 10.8 V and 13.2 V.
The Power Subsystem shall provide a 5.0 V DC output within ±5% while supplying at least 1.0 A continuous current to the ESP32, sensors, and relay logic.	• Power the ESP32, sensor modules, and relay logic from the 5 V rail during normal operation. • Measure the 5 V rail using a DMM and record the result. • Confirm that the measured voltage remains between 4.75 V and 5.25 V.
The Power Subsystem shall maintain stable operation of the ESP32 and sensing subsystem during simultaneous operation of both peristaltic pumps and relay switching, without brownout reset or loss of valid sensor data over a 5-minute interval.	• Run the full low-voltage subsystem with both peristaltic pumps active while toggling the relay-controlled loads. • Monitor the ESP32 output and dashboard readings over a 5-minute interval. • Confirm that no reset occurs and that valid sensor data is maintained throughout the test.
The Power Subsystem shall provide a 3.3 V DC output within ±5% for low-voltage sensor or logic interfaces that require 3.3 V operation.	• Power the system under normal operating conditions and measure the 3.3 V rail or ESP32-provided 3.3 V output using a DMM.• Record the measured voltage and confirm that it remains between 3.135 V and 3.465 V during operation.

Table A.2: Control Subsystem - Requirements & Verification Pt. 1

Requirements	Verification
When the Control Subsystem detects that the measured liquid temperature is outside the user-defined temperature range, it shall command the heating pad appropriately to drive the system back toward the acceptable range.	• Begin operation from a low-temperature condition and observe controller output and relay behavior. Confirm that the heating pad is enabled when the temperature is below range and disabled once the measured temperature returns to the acceptable band.
When the Control Subsystem detects that the liquid level in the main fermentation jar is outside the allowed range, it shall command the peristaltic pump subsystem to add or remove tea as needed to drive the level back toward the acceptable range.	• Create both low-level and high-level conditions in the main jar and observe controller output and pump behavior. Confirm that the controller commands tea input when the level is too low and commands liquid removal when the level is too high.
When the Control Subsystem detects that the measured color state is below the user-defined target band, it shall issue an alert indicating that liquid refresh is recommended, allowing the user to decide whether to initiate corrective action.	• Begin operation from an off-target low-saturation color condition and observe controller output and pump behavior. Confirm that the controller initiates the intended pump sequence and that the measured color state trends toward the target band.
The Control Subsystem shall monitor pH as an indicator of fermentation progression and batch condition and shall make this information available for logging and user review without requiring direct pH-based actuation during normal operation.	• Operate the system while collecting pH data and confirm that the controller receives and processes pH measurements during operation without requiring direct pH-triggered actuator commands.
The Control Subsystem shall	• Run the system with the oxygen schedule enabled and observe

enable oxygen delivery according to a programmed timing schedule during operation.	the oxygen pump control output over the programmed interval. Confirm that the pump is activated and deactivated according to the intended timing sequence.
When the system boots or resets, the Control Subsystem shall default all actuator outputs to the OFF state until valid control operation begins.	• Power cycle or reset the controller and observe the actuator outputs immediately after startup. Confirm that all actuators remain off until the controller intentionally begins normal operation.

Table A.3: Sensing Subsystem - Requirements & Verification Pt. 1

Requirements	**Verification**
Taking the ultrasonic sensors as input, the sensing subsystem must determine the liquid levels in the main fermentation jar and auxiliary side jars with repeatability of ±1 cm.	• Ensure the main fermentation jar and both auxiliary jars are each filled to fixed stationary levels and record the measured liquid levels from the ESP32 via serial debugging. Repeat these measurements several times and confirm the readings remain within ±1 cm for each sensor. • Next, increase the liquid level in each jar and record the updated readings. Confirm that each sensed level changes in the correct direction and reflects the higher fill condition. • Then, decrease the liquid level in each jar and record the updated readings. Confirm that each sensed level changes in the correct direction and reflects the lower fill condition.
Taking the temperature sensor as input, the sensing subsystem must determine the brew temperature at a rate of at least 1 sample per second and with sufficient stability to verify whether the system is within ±1.0 °C of the target range.	• Ensure the temperature sensor is placed in a stable liquid bath and record the reported temperature from the ESP32 via serial debugging for at least 1 minute. Confirm that the subsystem updates temperature readings at a rate of at least 1 sample per second. • Next, place the sensor in a colder liquid condition and record the measured temperature. Then place the sensor in a warmer liquid condition and record the measured temperature. Confirm that the sensed temperature changes in the correct direction for both cases. • Finally, hold the liquid near a fixed temperature and confirm that the readings remain stable enough to verify whether the system is within ±1.0 °C of the desired target range.

Taking the RGB color sensor as input, the sensing subsystem must determine the color state of the brew and provide values sufficient to compute HSL measurements	• Ensure the color sensor is positioned beneath a stationary liquid sample of known appearance and record the RGB values as read from the ESP32 via serial debugging for at least 1 minute. Confirm that the subsystem updates readings at a rate of at least 1 sample per second. • Next, replace the sample with a lighter or less saturated liquid and record the new RGB and computed HSL values. Confirm that the reported values change from the original reading. • Then, replace the sample with a darker or more tea-colored liquid and record the updated RGB and HSL values. Confirm that the subsystem detects a measurable trend toward the desired color band.
Taking the pH sensing board and probe as input, the sensing subsystem must determine the pH of the brew with repeatability of ±0.3 pH over a 5-minute interval and must be capable of tracking a pH change of at least 1.5 pH units.	• Calibrate the pH probe using appropriate buffer solutions according to the sensor procedure. • Then, place the probe in a stable, well-mixed solution and record the measured pH over a 5-minute interval via serial debugging. Confirm that the readings remain within ±0.3 pH over that interval. • Next, place the probe in a second solution whose pH differs significantly from the first, or introduce an acidic solution into the sample. Record the updated pH values and confirm that the subsystem detects a change of at least 1.5 pH units.
Taking all sensing inputs together, the sensing subsystem must provide simultaneous valid measurements of liquid level, temperature, color state, and pH without loss of sensor data over a 5-minute interval.	• Run the full sensing subsystem with the main-jar ultrasonic sensor, side-jar ultrasonic sensors, temperature sensor, RGB sensor, and pH sensor all active at the same time. • Record all sensor outputs from the ESP32 via serial debugging or dashboard logging over a 5-minute interval. • Confirm that each sensing channel continues to report valid data throughout the test and that no sensor stream is lost during simultaneous operation.
Taking all sensing outputs as input, the sensing subsystem must ensure that any signal delivered to the ESP32 remains within the allowable 0–3.3 V input range during normal operation.	• Power the sensing subsystem under normal operating conditions with the pH sensing board, ultrasonic sensors, temperature sensor, and RGB sensor connected. • Measure the conditioned pH output and each ultrasonic ECHO line at the ESP32 input pins using a DMM or oscilloscope. • Record the measured voltages and confirm that no sensing signal applied to the ESP32 exceeds 3.3 V during operation.

Table A.4: Actuation Subsystem - Requirements & Verification

Requirements	Verification
Taking controller commands as input, the Actuation Subsystem must switch the heating mats and oxygen pumps on and off through the 12 V actuation interface during normal operation.	• Connect the heating mats and oxygen pumps to the 12 V actuation interface. • Command each load on and off from the controller and confirm correct activation and deactivation. • Inspect the final assembly and confirm that the switched AC wiring is enclosed and strain-relieved during normal operation.
Taking controller commands as input, the Actuation Subsystem must allow the heating pad to provide measurable heating to the fermentation jar.	• Begin with the fermentation jar below the desired operating range. • Command the heating pad on and record the measured liquid temperature over time. • Confirm either a sustained temperature rise of at least 0.15 °C/min for 10 minutes or that the liquid reaches within ±1.0 °C of the selected setpoint during testing.
Taking controller commands as input, the Actuation Subsystem must allow the oxygen pump to switch on and off according to the programmed aeration schedule.	• Connect the oxygen pumps to the 12 V actuation interface. • Run the system with the aeration schedule enabled and observe pump activation over the test interval. • Confirm that the pump turns on and off according to the commanded schedule.
Taking controller motor-driver commands as input, the Actuation Subsystem must operate the two 12 V peristaltic pumps and provide sufficient fluid transfer for system operation.	• Connect both peristaltic pumps to the motor driver and command each pump during normal operation. • Measure the volume transferred over a known time interval and compute the corresponding flow rate. • Confirm that the pumps respond correctly to controller commands and that the measured net flow rate meets the required operating value.

Table A.5: User Interface Subsystem - Requirements & Verification

Requirements	Verification
Taking system data as input, the User Interface Subsystem must display temperature, pH, color-state, liquid-level, and	• Run the full system under normal operating conditions and observe the web dashboard. Confirm that the major sensor values and actuator states are visible during operation.

actuator-state information during operation.	
Taking system data as input, the User Interface Subsystem shall maintain timestamped logs and short-term history for at least 10 minutes while active.	• Operate the system for at least 10 minutes while the dashboard is active. • Confirm that the log contains timestamped entries for the major sensor channels and actuator states throughout the test interval.
Taking user input as input, the User Interface Subsystem must allow the user to enter operating parameters and manually command actuators for testing and supervision.	• Enter updated operating parameters through the dashboard and confirm that the new values are accepted by the system. Then manually command each actuator from the interface and confirm the expected system response.
Taking system status as input, the User Interface Subsystem shall display system alerts/status messages for invalid sensor readings, connectivity/telemetry issues, and operator actions; reservoir-specific level alerts are planned enhancements.	• Create at least one alert condition, such as lowering the fresh-tea jar level, raising the waste-jar level, or disconnecting a sensor input. Confirm that the appropriate warning or status message appears on the dashboard.

Appendix B: Verification Data and Graphs

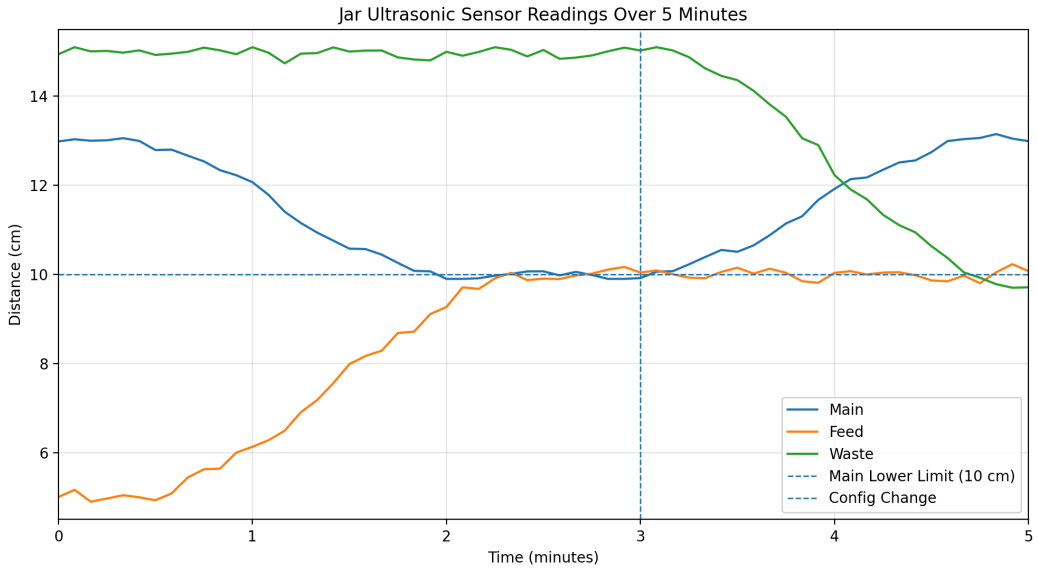

Figure B.1: Ultrasonic sensor distance readings over 5 minutes

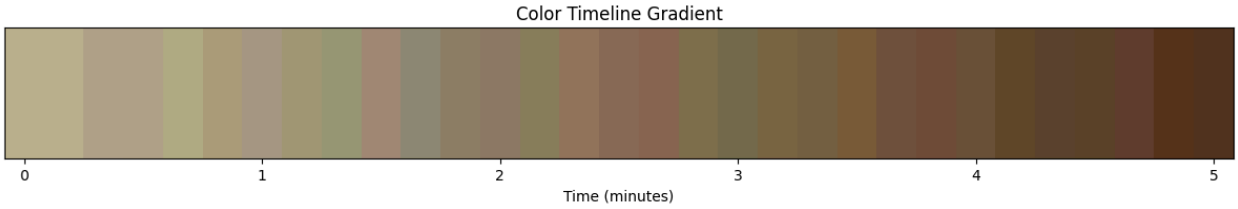

Figure B.2: RGB color timeline gradient over 5 minutes

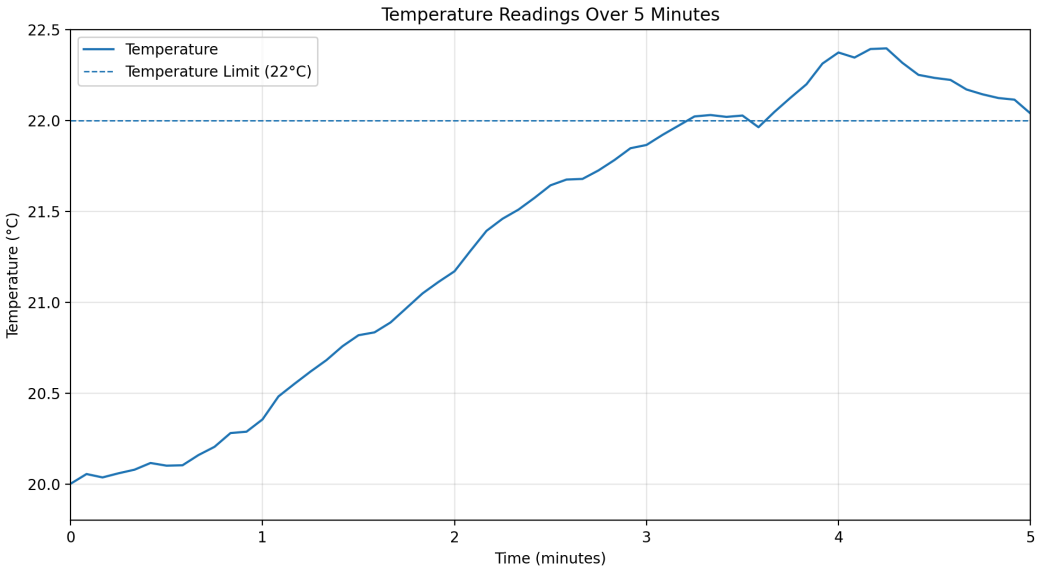

Figure B.3: Temperature readings over 5 minutes

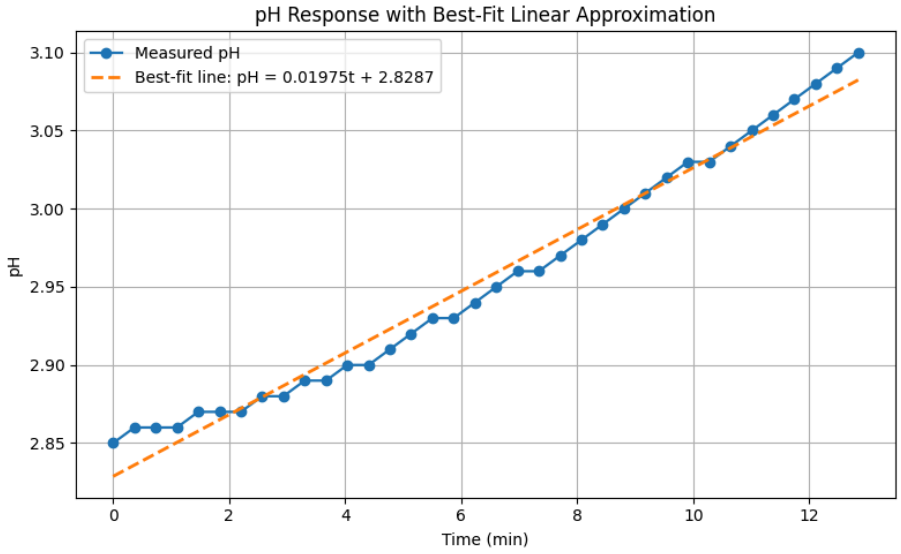

Figure B.4: pH response with best-fit linear approximation

Appendix C: User Interface / Data Pipeline Screenshots

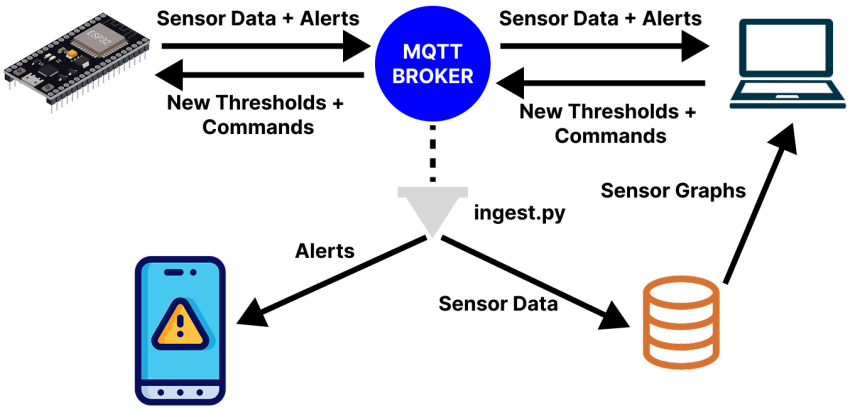

Figure C.1: MQTT-based data pipeline for telemetry, alerts, and commands

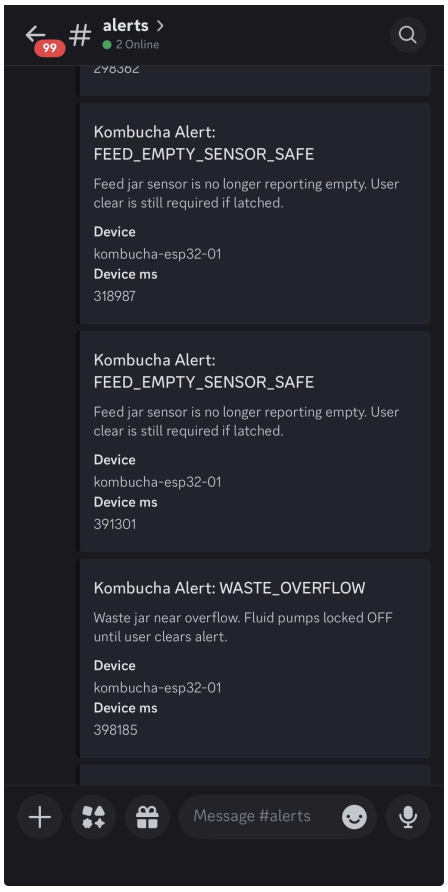

Figure C.2: Discord alert notifications

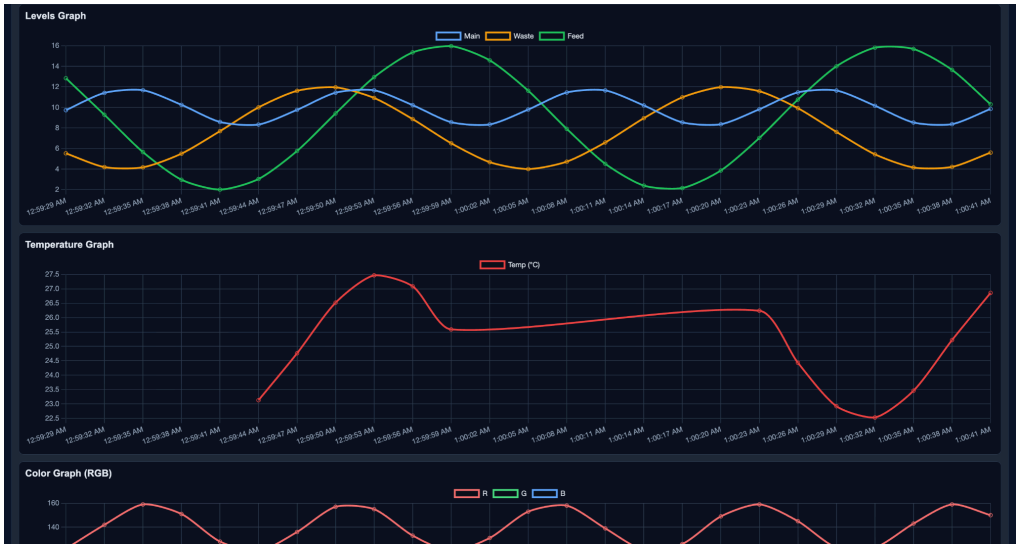

Figure C.3: Dashboard history graphs

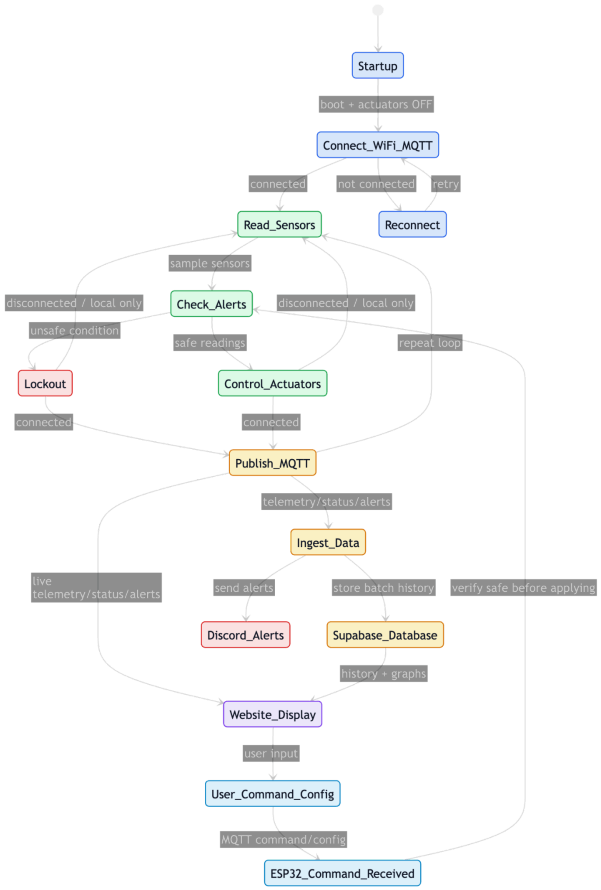

Figure C.4: Firmware and data pipeline state diagram

Appendix D: Cost and Schedule Tables

Table D.1: Itemized list of components and costs

Description	Manufacturer	Quantity	Extended Price	Source
HC-SR04 Ultrasonic Module Distance Sensor	ELEGOO	3	$8.99	Link
2 Channel DC 5V Relay Module	SunFounder	1	$6.79	Link
Silicone Heater Mat 3.15 x 3.94 inch	IS	4	$22.99	Link
DC 12V Mini Air Pump Motor	SAIM	2	$9.94	Link
12V Peristaltic Pump	Shutao	2	$12.99	Link
DS18B20 Temperature Sensor	DIYables	1	$6.99	Link
TCS-34725 RGB Light Color Sensor	HiLetgo	1	$6.99	Link
Supply 1 Gallon Glass Jar Wide Mouth	North Mountain	1	$12.99	Link
Pure Silicone Tubing, 3mm ID x 5mm OD	Quickun	1	$7.99	Link
ESP32-DevKitC-32UE	Espressif Systems	1	$8.08	Link
L298N Motor Drive Controller Board Module	Qunqi	1	$6.99	Link
Gravity: Analog pH Sensor - SEN0161	DFRobot	1	$29.50	Link
Supply 32oz Glass Jar	ULINE	2	$7.25	Link
PCBWay Audit Cost (2 different PCBs)	PCBWay	20 (10 per PCB)	$10	Link
Capacitors (3x 0.1 uF, 3x 10 uF)	Digikey	6	$3.03	Link Link
Connectors (2 01x19, 4 01x04, 2 01x03, 1 01x02, 1 02x02)	Digikey	10	$4.91	Link Link Link Link Link
Resistors (1x 10k, 3x 1k, 3x 2k, 3x 4.7 k)	Digikey	10	$1.04	Link Link Link Link

2 Position Wire to Board Terminal Block	Digikey	1	$0.79	Link
Tactile Switch SPST-NO Top Actuated Surface Mount	Digikey	1	$0.43	Link
AMS1117-3.3 V, Low Voltage Dropout Regulator	Digikey	1	$0.29	Link
Input Capacitor 470uF	DMBJ	1	$0.0725	C1927 0690
Output Capacitor 330uF	KNSCHA	1	$0.0218	C5155 335
Schlottsky Diode	FUXINSEMI	1	$0.0143	C7503 125
Inductor 47uH	SHOU HAN	1	$0.0348	C6364 681
Buck Regulator	HTC	1	$0.1715	C1260 46

Table D.2: Schedule for project progression

Week	Task	Worker
1/19	Project pitch	Everyone
1/26	Project approval	Everyone
2/2	Project design	Everyone
2/2	Submit proposal	Everyone
2/16	Order sensors + testing them	John
2/23	PCB revision	Rudy + Edwin
3/2	PCB Assembly	Everyone
3/9	PCB Revision	Rudy + Edwin
3/16	Spring Break	N/A
3/23	Integrate parts	Everyone

3/30	Setup the user interface	John
4/6	Test the system	Everyone
4/13	Refine any errors	Everyone
4/20	Mock demo	Everyone
4/27	Final demo	Everyone
5/4	Final report	Everyone