Lab 1

Team 10

Rabi Alaya, Caiden Atienza, Nick Hageman, Nate Schaefer

October 4th

ECE:4880

Table of Contents

Design Documentation -- 2

System Level --- 2

Hardware Level --- 3

Software Level -- 4

Design Process and Experimentation -- 6

ESP32 vs Raspberry Pi Pico --- 6

OLED vs LCD-- 6

Rechargeable Power Bank vs AA Battery Pack --- 6

3D Printed Third Box vs Plastic Box -- 7

Test Report -- 10

Project Retrospective --- 18

Project Summary -- 18

Team Roles -- 18

Project Management -- 19

Appendix & References -- 20

Design Documentation

System-level View:

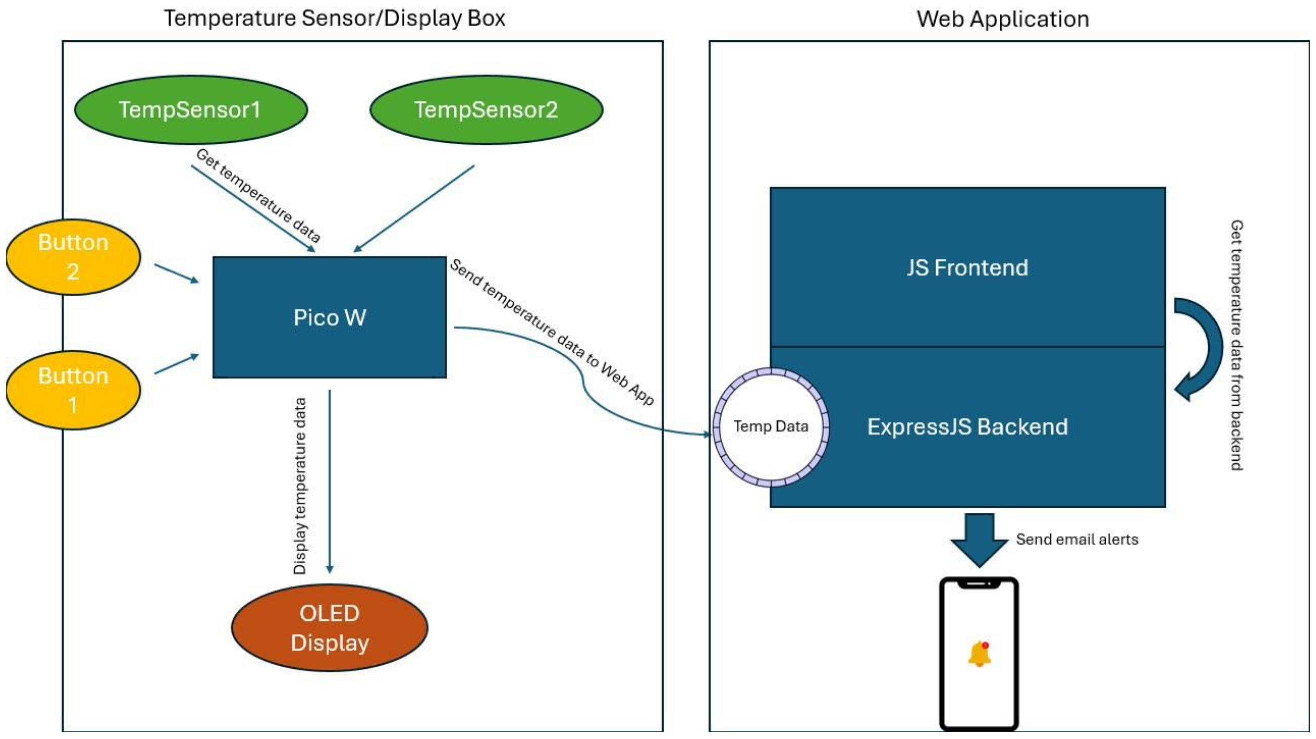

Figure 1. A block diagram showing the design of the product.

System-level Description:

The system consists of a Raspberry Pi Pico W that collects temperature data from two sensors. This data is displayed on an OLED screen and sent via HTTP POST requests to an NodeJS backend every second. The backend stores the last 300 data points.

The frontend fetches this temperature data every second via GET requests to the NodeJS backend. It then updates a real-time graph showing the last 300 data points. Users can toggle the graph between Celsius and Fahrenheit.

Both physical buttons connected to the Pico W and the virtual buttons on the frontend enable users to toggle each temperature sensor on or off. When a button is pressed, whether physically or via the web interface, a GET request is sent to the NodeJS server which updates a sensor states data structure that keeps track of the state (on/off) for each sensor. The Pico W sends GET requests every second to retrieve the current state of each sensor, ensuring that the OLED display accurately reflects whether each sensor is active.

The backend enables users to configure temperature thresholds through the frontend. When updated, the frontend sends an HTTP POST request to the backend, which continuously monitors sensor data for any deviations from these thresholds. If the temperature crosses the specified limits, the backend automatically sends an email alert to notify the user via their phone. Both the email notifications and the thresholds are fully customizable through the frontend.

Overall, the system integrates hardware and software components to provide users with real-time temperature monitoring, controlled sensor management, and automated alerts.

Hardware Description:

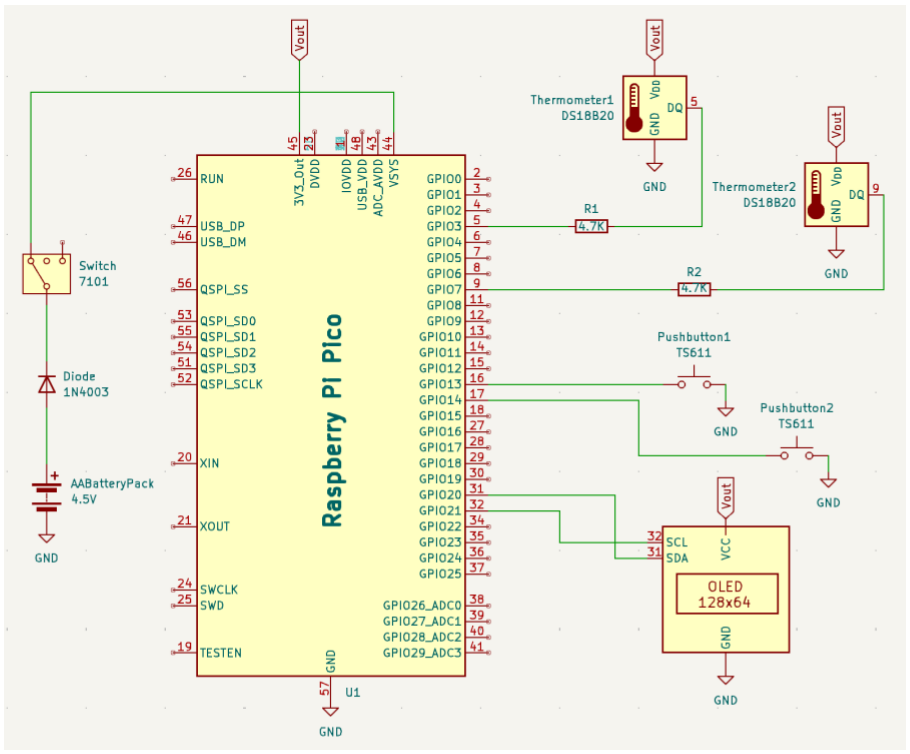

Figure 2. Circuit schematic showing the corresponding components used and their connections to microcontroller.

The Raspberry Pi Pico is powered by a 4.5V AA battery pack, providing a stable DC power source to the microcontroller. The power supply is regulated using a 7101 three-position toggle switch, placed in series with the battery pack, allowing the user to control system power between ON, OFF, and an intermediate state for low-power modes. A 1N4003 rectifier diode is integrated into the circuit to prevent reverse current flow, ensuring that if the microcontroller is also connected to an external power source, there is no risk of back-charging the batteries. This diode has a forward voltage drop of approximately 0.7V, which is factored into the overall voltage supply to the microcontroller.

For temperature sensing, two DS18B20 digital thermometers are interfaced with GPIO pins 3 and 7. The DS18B20 sensors communicate using a One-Wire protocol, which allows multiple devices to be connected to a single data line. However, we opted to have each sensor assigned to their own data line. To ensure proper communication and data integrity, each sensor is accompanied by a 4.7kΩ pull-up resistor (R1 and R2) on the data lines, connected between the data pin and the 3.3V rail. These pull-up resistors are critical for keeping the data line in a defined high state when no data is being transmitted, preventing floating states.

Two TS611 pushbuttons are connected to GPIO pins 13 and 14, serving as user input controls. These buttons are configured using internal pull-up resistors in the microcontroller, ensuring that the pins are held in a high state when the button is unpressed. When a button is pressed, the corresponding pin is pulled low allowing the microcontroller to detect the state change and trigger the appropriate software event.

The system includes a 128x64 OLED display module connected via I2C communication to GPIO pins 20 (SDA) and 21 (SCL). This display serves as the primary output for real-time temperature readings from the DS18B20 sensors. The I2C protocol is ideal for this application due to its simplicity and ability to communicate with multiple devices over just two wires. The display operates at 3.3V, matching the output of the Pi's voltage output pin.

Software Description:

- Raspberry Pi Pico W
  1. Initialize Components:
     a. Set up the temperature sensors, buttons, OLED display, and Wi-Fi connection.
  2. Main Loop:
     a. Read temperature values from both sensors.
     b. Display the temperature on the OLED screen for each sensor that is currently active (on).
     c. Send the temperature data to the server.
     d. Request the current sensor statuses (on or off) from the server to know whether to display data for each sensor.
     e. If a physical button is pressed, send a request to the server to toggle the corresponding sensor on or off.
     f. Wait for one second before repeating the process.
- Backend
  1. Receive and store temperature data
     a. Keep a record of the most recent 300 temperature readings for each sensor, both in Celsius and Fahrenheit.
     b. Track whether each sensor is currently active (on or off).

2. Send Temperature Data to Frontend
    a. Provide the frontend with the latest temperature data every second so that the frontend can display the data in real-time and graph it.
3. Send Sensor Status Data to Raspberry Pi Pico W
    a. Provide the Pico W with sensor status (on/off).
4. Send Email Alerts
    a. Continuously check if the temperature readings exceed the user-defined thresholds.
    b. If any threshold is crossed, send an email alert to notify the user.

- Frontend

1. Get temperature updates
    a. Get new temperature data from the backend every second
        i. Update the graph with the new temperature values.
        ii. Display the most recent temperature reading for each sensor on the page.
        iii. Check the current sensor statuses (on or off) and update the display accordingly.
2. Toggle Temperature Units
    a. Allow the user to toggle between Celsius and Fahrenheit by pressing a button. When toggled:
        i. Convert the temperature data accordingly.
        ii. Update the graph and the temperature readings on the page to reflect the selected unit.
3. Toggle Sensor States
    a. Provide buttons for the user to toggle each sensor on or off.
    b. When the user presses a button, send a request to the server to change the sensor's state and immediately update the display to reflect the new status.
4. Set Temperature Thresholds for Email Alerts
    a. Allow the user to input minimum and maximum temperature thresholds.
    b. When the user updates these thresholds, send the new values to the server so that email alerts can be sent if the temperature goes outside of the specified range.

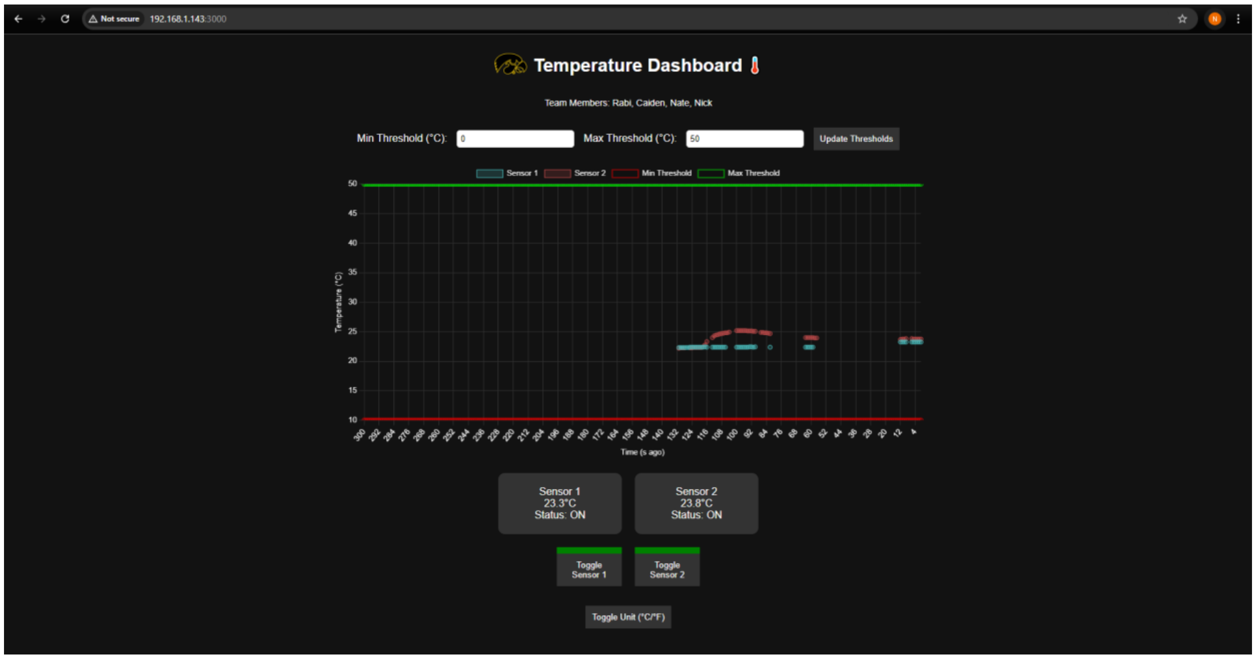

Figure 3. Web server's display for the user to watch current temperature values.

Design Process and Experimentation

ESP32 vs. Raspberry Pi Pico

During our planning process of the project, we made the decision to incorporate a Raspberry Pi Pico into our system. Although the Arduino would have made the hardware simpler, we had trouble preliminarily testing its connection with the Wi-Fi and attempting to send data to the web server. When we pivoted to the Raspberry Pi Pico, the process of exchanging data between the web server and third box was much easier to implement.

OLED vs. LCD

While developing our project, we intended to use an LCD display for the third box display. The LCD screen seemed to bring us trouble during the development process. Something seemed to keep going wrong with the initialization process for the LCD, preventing it from working properly. We pivoted to using an OLED display, which was much easier to set up. The OLED presented no problems throughout the rest of the development process.

Rechargeable Power Bank vs. AA Battery Pack

Choosing to power the Raspberry Pi Pico with a pack of 3 AA batteries instead of a USB power bank was a practical decision for several reasons. First, the cost of AA batteries is lower compared to a USB power bank, making it a more budget-friendly option, especially for smaller projects. Additionally, AA batteries provide a direct and easily controlled power source to the input voltage of the Pico. This allowed for greater flexibility in managing power, including the ability to cut power immediately when needed versus having to wire the switch to the 3V3 pin on the Raspberry Pi. The Raspberry Pi Pico's power input was easy since it has a built-in voltage regulator and could be powered from 1.8 to 5.5 volts.

3D Printed Box vs. Plastic Box

While planning the enclosure for our system, we initially considered using a plastic Tupperware container, thinking it would be the simplest and quickest option. However, when we attempted to cut holes for the temperature sensors and OLED display, we quickly encountered difficulties. The cutting process was challenging, and the result looked unpolished. One team member suggested 3D printing a custom box, which provided precise openings for the sensors and display. This made the process much smoother and significantly improved the overall presentation of the project. In addition, one concern with using Tupperware was the structural integrity and how the implementation of the push buttons was going to hold up. By choosing to 3D print a box, we were able to design mounting holes for each of the components. The one component that benefited significantly was the push buttons. The push buttons were press fitted into the mounting holes and had center ledge to sit on. With this design, the buttons would stay securely mounted and could still be wired to the raspberry pi through the designated pin channels in the lid.

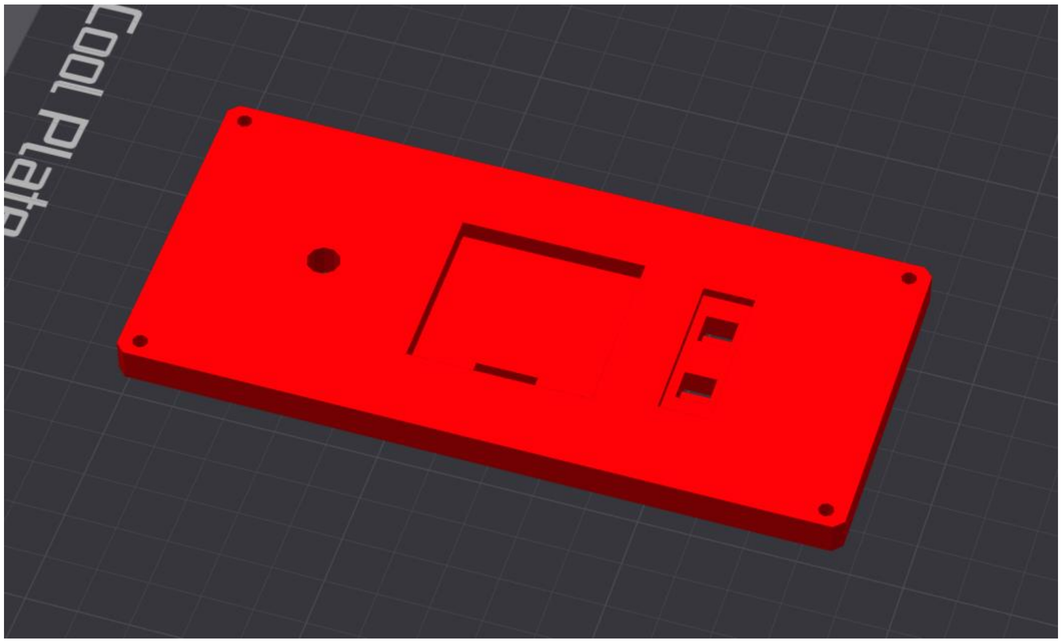

Figure 4. Cover of the third box represented in Bambu Lab software, before being 3D printed.

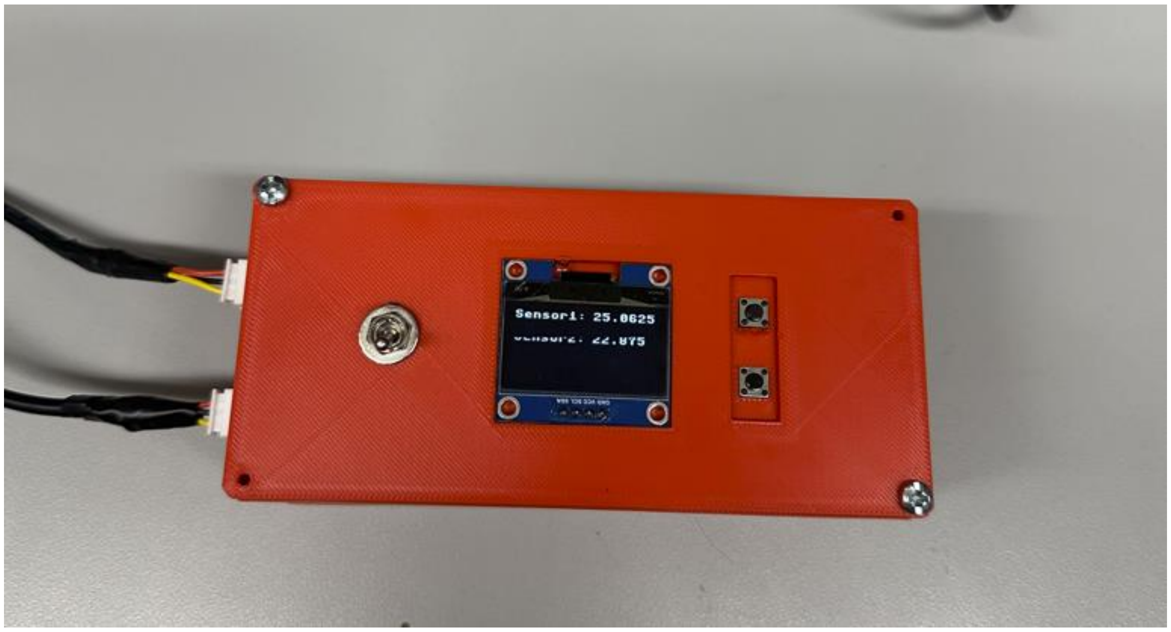

Figure 5. Top of third box after implementation. Includes switch, OLED, and buttons. Sensor2 value is distorted from iPhone camera.

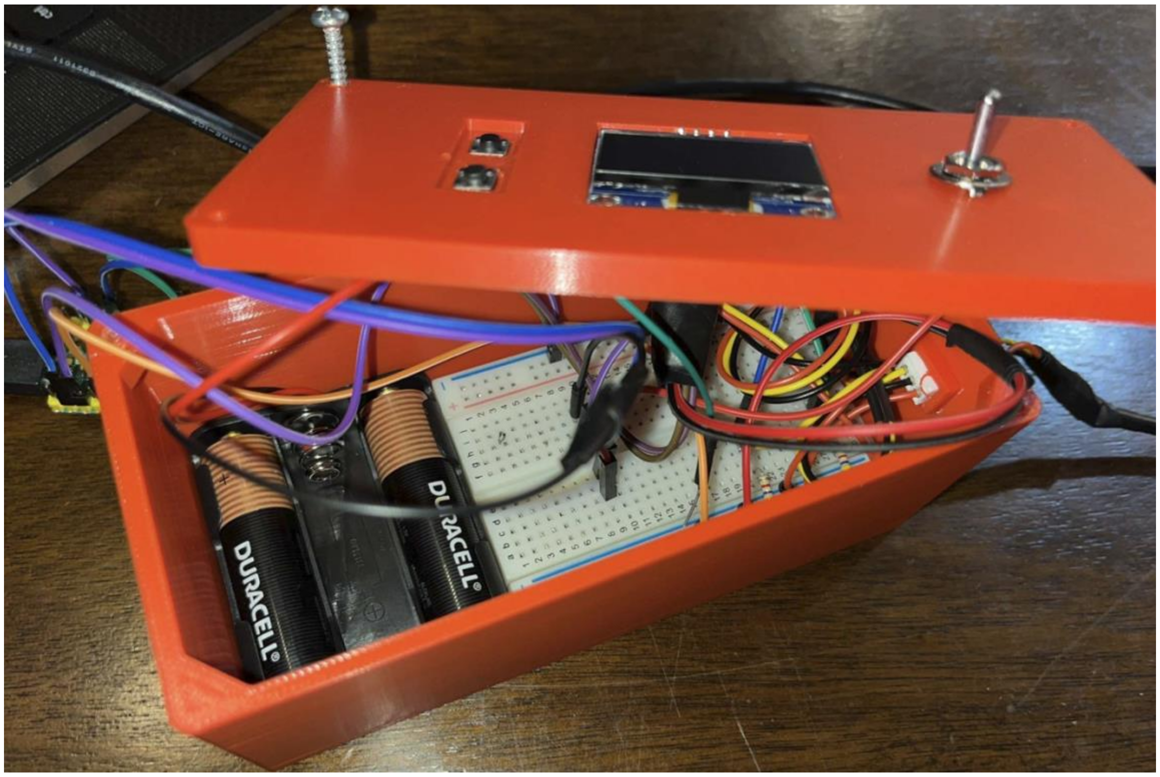

Figure 6. View of third box with insides exposed.

Test Report

Requirements	Test Procedure	Test Result	Comments	Signature	Test Date
2a	Turn the third box on and check if the LCD display is on and buttons/switch work. Turn the box upside down and back to normal. Check that the OLED display, buttons, and switch still work.	Pass		Caiden A.	10/2
2a	Turn the third box on and check if the LCD display is on and buttons/switch work. Drop the box from bench height. Pick it up and check that the OLED display, buttons, and switch still work.	Pass		Caiden A.	10/1
2b	Turn the third box on. Verify from the OLED that both temperature sensors are reading in data. For each sensor, unplug it and check that the OLED displays a message that the corresponding sensor is disconnected. Plug each sensor back in and verify that the temperature is being read from that sensor.	Pass		Caiden A.	10/3
2c	Turn the box on. Verify that the box is reading in temperatures from both sensors. Drop the box from bench height. If the	Pass		Caiden A.	10/3

	connectors aren’t disconnected, verify that the box is still reading temperature. If the connectors are disconnected, connect them and verify that the box is still reading temperature.				
2d	Turn the third box on. Verify that it is reading temperature from both sensors. Unplug one of the sensors and verify that the OLED display is still on. Grab the other sensor and check that the temperature is rising. Plug the sensor back in and do the same procedure with the other sensor.	Pass		Caiden A.	10/2
3	Turn on the third box. Verify that the OLED is displaying temperatures. Turn the switch off. Verify that the OLED does not display temperatures.	Pass		Caiden A.	9/27
3	Turn on the third box and connect it to the web server. Verify that the web server is displaying the current temperatures. Turn the switch off and verify that the web server stops displaying the current temperatures.	Pass		Nate S.	10/1
4a	Turn the third box on and verify that the temperatures are being displayed on the OLED. Press a button to turn a sensor off and press it again to turn it	Pass		Caiden A.	10/1

	on. Check that the delay of the temperature appearing is not noticeable. Repeat the same process for the other sensor.				
4b	Turn on the third box. Verify that the OLED display is clear and easy to read. Verify that temperature values are displayed correctly.	Pass		Caiden A.	9/23
4c	Turn the third box on. Press one button and verify that the OLED displays that the corresponding sensor is off and that the other sensor is on. Press the other button and verify that both sensors are off from the OLED. Press the first button and verify that the corresponding sensor turns on from the OLED reading. Turn the other sensor on by pressing the other button and verify that both sensors are reading temperature values.	Pass		Caiden A.	10/2
4d	Turn on the third box. Verify that both temperature values are being read in from the OLED. Unplug one sensor and verify that the OLED displays that the corresponding sensor is errored out. Repeat the same process with the other sensor.	Pass		Caiden A.	10/2

5ai	Turn on the third box and connect it to the web server. Verify that both temperature values are being displayed. Unplug one of the sensors and verify that a message saying “unplugged sensor” is being displayed on the web server. Repeat with the other sensor.	Pass		Nate S.	10/2
5aii	Turn on the third box and connect it to the web server. Verify that both temperature values are being displayed. Turn the switch off and verify that the web server displays a “no data available” message.	Pass		Nate S.	10/2
5b	Turn on the third box and connect it to the web server. Verify that both temperature values are being displayed. Press the button for one of the sensors from the web server. Verify that the corresponding sensor stops displaying its readings for the web server and OLED within 1 second. Press the web server button again and verify that the sensor readings are displayed as usual. Repeat the same process with the other sensor.	Pass		Nick H.	10/1
5ci	Turn on the third box and connect it to the	Pass		Nate S.	9/25

	web server. Verify that both temperature values are being displayed on the web server. Click the button to switch between degrees C and degrees F. Verify that the values change correctly, and that the orientation of the graph matches the requirement.				
5cii	Turn on the third box and connect it to the web server. Verify that both temperature values are being displayed on the web server. Verify that the graph shifts every second and if a new value is recorded, it is added on the right side. Check that older values scroll off the left side of the graph.	Pass		Nate S.	9/26
5ciii	Turn on the third box and connect it to the web server. Verify that both temperature values are being displayed on the web server. Verify that the temperature values recorded over 300 seconds scroll off the graph. Check that the X axis of the graph is labeled in “seconds ago from the current time”.	Pass		Nate S.	9/26
5iv	Turn on the third box and connect it to the web server. Verify that both temperature	Pass		Nick H.	9/23

	values are being displayed on the web server. Turn the switch off for a few seconds. Verify that the graph continues to scroll without displaying new data. Turn the switch back on. Check that the graph starts displaying the current data.				
5v	Turn on the third box and connect it to the web server. Verify that both temperature values are being displayed on the web server. Turn the switch off for a few seconds and then turn it back on. After a few seconds, verify that a noticeable gap in the graph is shown from the time that the switch was off.	Pass		Nick H.	9/25
6	Turn on the third box and connect it to the web server. Start a timer when the box is turned on. Verify that both temperature values are being displayed on the web server before the timer hits 10 seconds.	Pass		Nick H.	9/25
7	Turn on the third box and connect it to the web server. Verify that both temperature values are being displayed on the web server. Set the upper threshold to 5 degrees above room temperature and	Fail	Email is sent to hardcoded email; email cannot be changed by user.	Nate S.	10/4

	configure an email. Hold a temperature sensor in your hand and check that the reading goes above the threshold and an email is sent to the specified email. Repeat the same process with a lower threshold of 5 degrees below room temperature and putting the sensor in water.				
8b	Turn on the third box and connect it to the web server. Verify that both temperature values are being displayed on the web server. Hold one of the sensors in your hand and verify that the corresponding temperature rises. Repeat the process with the other sensor.	Pass		Nick H.	9/28
8c	Turn the third box on in the lab room. Verify that the temperature values are being displayed. Verify that the temperature recorded is within 4 degrees of 22 degrees Celsius.	Pass		Rabi A.	9/27
8d	Turn the third box on. Verify that the temperature values are being displayed. Place the sensors in a water-ice mixture and verify that the temperature recorded is within 2 degrees of 0 degrees Celsius.	Pass		Rabi A.	9/27

Project Retrospective

Project Summary

We succeeded in creating a third box that communicates with a web server to provide the user with temperature values no later than 300 seconds ago. Using a web server based off NodeJS, with JavaScript, and in the Raspberry Pi Pico: python code to read the sensors and send HTTP requests between the Pi and the server.

Our planning phase ended up needing to be changed quite a bit. Using our agile methodology, we were able to pivot quickly to save time implementing a different concept for a solution. Some of the ways we had to pivot in our project were when we decided to use a Raspberry Pi Pico instead of an Arduino, when we chose to use an OLED instead of an LCD for the display, and when we made the decision to use our laptop's local IP, rather than a service like Ngrok.

The communication between our Pi and web server was the part of our lab that took the longest for us. We had a lot of trouble trying to communicate between eduroam and UI-DeviceNet. We chose to use a service like Ngrok that would give us an endpoint that our Pi was able to communicate with. This worked for a while until we realized the limitations of it. There was a limit to HTTP requests, about 120 per minute, which was not enough for us. Our solution came when Noah announced the custom router for the class. We were then able to communicate with the laptop's local IP using this router.

In hindsight, we would have put more effort into our planning stage, so that we didn't need to be affected by pivoting in our project multiple times. Another thing would be to read the requirements a bit more closely. We ended up losing points on the checkoff from not including a text box to allow the user to specify an email. This issue could have been avoided with more careful planning and clear communication in the early phase of the project.

Team Roles

- **Rabi**: Set up the Raspberry Pi Pico W with MicroPython. Wrote code to read in temperature, connect to Wi-Fi, and send temperature data to the server.
- **Caiden:** Oversaw the hardware aspects and was responsible for integrating the hardware into the third box. His efforts included powering the raspberry pi, soldering the wires, and designing/3D printing the third box.
- **Nick**: Developed the frontend and backend for the server. Handled the graphing for temperature data, implemented the Celsius to Fahrenheit toggle button, and managed the email alerts system.
- **Nate**: Handled sprint planning and created the Gantt chart. Wrote all the test cases. Implemented code for sensor statuses on web server.

Project Management

We used an agile based approach for our project. We did this because we wanted to be able to change our requirements if we learned that something we planned to implement would be a lost cause, which ultimately ended up happening multiple times. Some examples of this are when we had issues connecting to the Wi-Fi with the Arduino, or when the LCD stopped functioning as we wanted it to. When we pivoted from the Arduino to using the Raspberry Pi Pico, we started having issues with the LCD, eventually finding that an OLED display may work better for us. Both issues were easily fixed after we were able to pivot quickly due to our agile methodology. Using it, we were able to modify our requirements accordingly so as not to waste time.

As for our management, we planned two sprints that each took two weeks. Our first sprint, we met around once a week on Tuesdays to discuss our work and continue working on the project together. The intention of our first sprint was to write out a game plan for what our product would work and start its implementation. Our intention was to have the Raspberry Pi Pico display temperatures on the third box display, communicating with the web server and the web server displaying that data on the graph. We achieved this goal but had to cut some corners in order to do so. The second sprint was where we refined our logic to satisfy the requirements completely and configured our project to work correctly; we also tested our product during this time. We met multiple times a week for a few hours each time to do this during our second sprint. As for a methodology, we prefer using agile rather than waterfall, as it allowed us to dynamically pivot towards new implementations for our project.

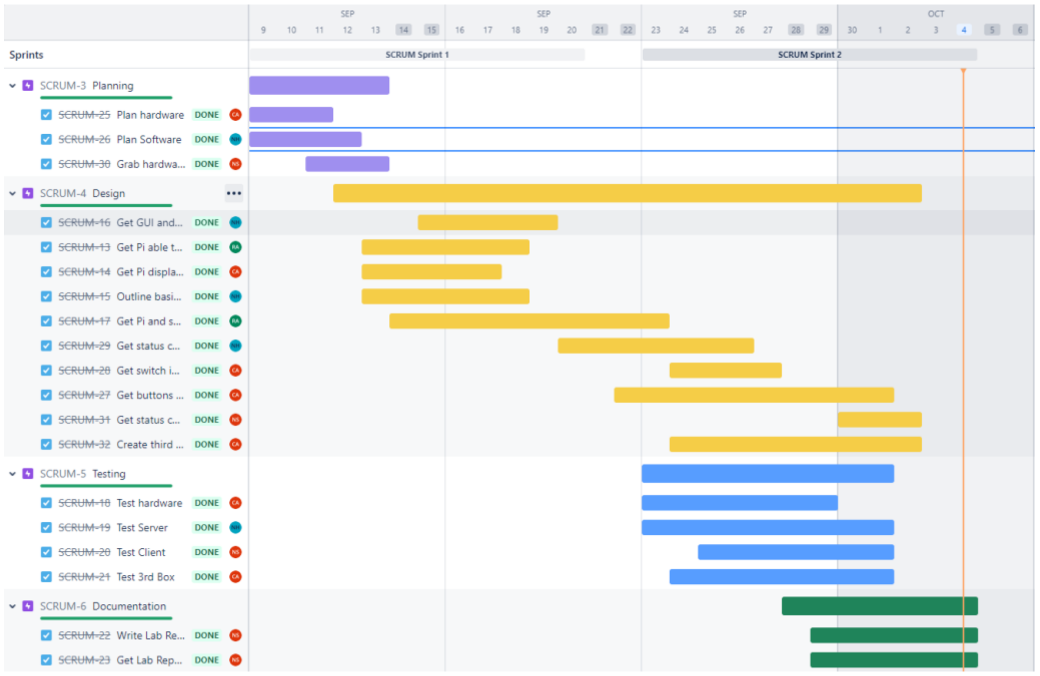

Figure 6. Gantt Chart for Lab 1 Timeline

Appendix & References

Raspberry Pi Pico docs: https://www.raspberrypi.com/documentation/microcontrollers/pico-series.html

Raspberry Pi Pico Datasheet: https://datasheets.raspberrypi.com/pico/pico-datasheet.pdf

NodeJS docs: https://nodejs.org/docs/latest/api/

Nodemailer for emailing: https://nodemailer.com/

Jira: https://uiowa-senior-design.atlassian.net/jira/software/projects/SCRUM/boards/1/timeline

PTC Creo 10.0: https://www.ptc.com/

Bambu Lab, Bambu Studio, Version 2.0: https://bambulab.com/en-us