CSE 123A - Week 4 Status

Group 8

What we Did

We have purchased the power module for this device. For the prototype we will be using a 3.7V 950mAh lithium ion battery. It will be connected to the ESP32 using the battery contact pins and a soldered on connector. Once we have set the device up in a way that we don't need to talk to it using USB for testing, we can power it solely from this battery. It should have more than enough power to support the board and the connected sensor. We also began designing custom 3D-printed mounting plates for the Digital Load Cell Weight Sensor. The circular plates (110mm diameter) went through several design iterations to get the M4 screw hole positioning and sizing right, with test prints done to verify fitment against the actual load cell.

What we are Working On

We are currently calibrating the load cell sensor now that the mounting plates have been built and tested. This involves configuring the HX711 breakout board and tuning the readings to ensure accurate weight measurements. We are also working on an improved plate design that includes countersunk screw holes so the base sits flat on a surface, as the current version leans slightly due to the screw heads protruding from the bottom.

What's next

We will be working on implementing push notifications for users when the water level is low. This will involve integrating Firebase Cloud Messaging (FCM) into our web server and setting up the necessary infrastructure to send notifications to users. We will also be working on writing unit tests for receiving sample water level data and triggering notifications. Additionally, we will continue refining the hardware design and calibration of the load cell sensor to ensure accurate and reliable measurements.

Individual Contributions

Dev Chodavadiya:

- Designed custom circular mounting plates (110mm diameter) for the Load Cell Weight Sensor using OpenCAD

- Went through multiple design revisions to achieve proper M4 screw hole placement, aligning holes at 22mm and 32mm from the plate center to match the load cell mounting points
- Test printed plates to verify fitment and adjusted hole diameter sizing to ensure proper M4 screw compatibility

Next steps include designing and printing spacers to go between the load cell and the mounting plates, as well as adding a recessed countersink around the screw holes on the bottom of the plate so the screw heads sit flush and the base lays flat on a surface.

Logan Bossuwe:

- From Dev's constructed plates, we communicated plate designs and I built the weight prototype with the 3D printed designs. I used 2 screws for both the top and bottom plate to lock them wit the load cell. Additionally added 5 washers with the screws to put a small buffer that will later be a proper 3D printed design.
- I soldered the Load Cells, 4 wires to their respective connections on the HX711. The load cell itself is a wheatstoen bridge circuit, and the HX711 is the amplifier.
- Successfully communicated a connection between each respective component, ESP32, HX711, and load cell to produce raw data. At the moment it is noisy.
- Began making debugging prints for future edge case testing.

Next, I will calibrate the raw data into a useable data being Grams. I will start to build a main file that will connect all ESP32 code together for running. I will begin creating tests the functions and check the calibration through known weights.

Pieter Nauwelaerts:

- Purchased battery and required hardware to mount it to the ESP device.
- Updated working document and information regarding testing procedure and initial HTTP Code
- Created design drawings representing multiple stages of the design process
- Created design drawing representing the finalized and ideal product deployment.

Reid Graham:

- Created basic architecture diagram for code and communication between the ESP and Webapp
- Coded the ESP to host a soft access point for sending WiFi credentials
- Coded the ESP host an HTTP server and HTML page which can receive user input

- Helped to solder the sensor to the amplifier

My next tasks are to code the ESP to attempt to connect to the given WiFi credentials and exit the setup phase. If it cannot connect, it should stay in setup and ask the user to submit again through the HTML page. I also plan to create the basic HTTP send and receive code so the ESP is able to connect to our web server once it is on the WiFi.

Sam Lai:

- Cloned the separate web server repository and tested if I can deploy to Vercel successfully
- Tested the server locally and made myself familiar with the codebase
- Researched ways to implement notifications to users when the water level is low
- Found two methods: text message notifications and push notifications.

I am leaning towards push notifications since they are more user friendly and can be implemented using a service like Firebase Cloud Messaging (FCM). Text message notifications would require integrating with a service like Twilio, which have additional costs, while FCM has a free tier that should be sufficient for our needs. I will be working on integrating FCM into our web server and setting up the necessary infrastructure to send push notifications to users when the water level is low. I will also be working on writing some unit tests for receiving sample water level data and triggering notifications.

Shriyan Gote:

- Styled the web dashboard as a Brita water level checker with a pitcher graphic, animated water fill, and percentage display
- Added last-updated timestamp and a green status indicator when water is present in the pitcher
- Implemented demo mode with mock sensor data so the UI can be tested without live hardware
- Fixed the water level animation so it shows the current level immediately on page load and only animates when the value changes from new data
- Set up deployment workflow to push to both the UCSC (GitLab) and GitHub repositories

Next steps include keeping the web server in sync with the team repo and continuing to refine the dashboard as sensor data becomes available.
Additionally, I will be working with Sam to implement additional features for the server and web app.