CSE 123A - Winter 2026

Group 8

Problem statement

Living with roommates can sometimes mean your filtered water device is empty when you need water. This can be very frustrating and leave you needing to drink tap water while other roommates get to drink cold refrigerated and filtered water.

Need Statement

We want a way to know if the container is out of water when we need water.

- Avoid having no water left in addition to everybody using the water filter, not knowing there is no water left.
- Need a way to know the current water level of the pitcher so that it isn't empty without everyone knowing it's empty.
- We need to know if the container is empty, in order to not run out of clean fresh water.

Goal

- Keeping water level known at all times, while informing everybody in the household if the water level runs too low.
- Create a system to inform users if the container is empty.

If the container is empty, inform users.

Personas and Users

This is intended for shared households that rely on a shared filtration water container.

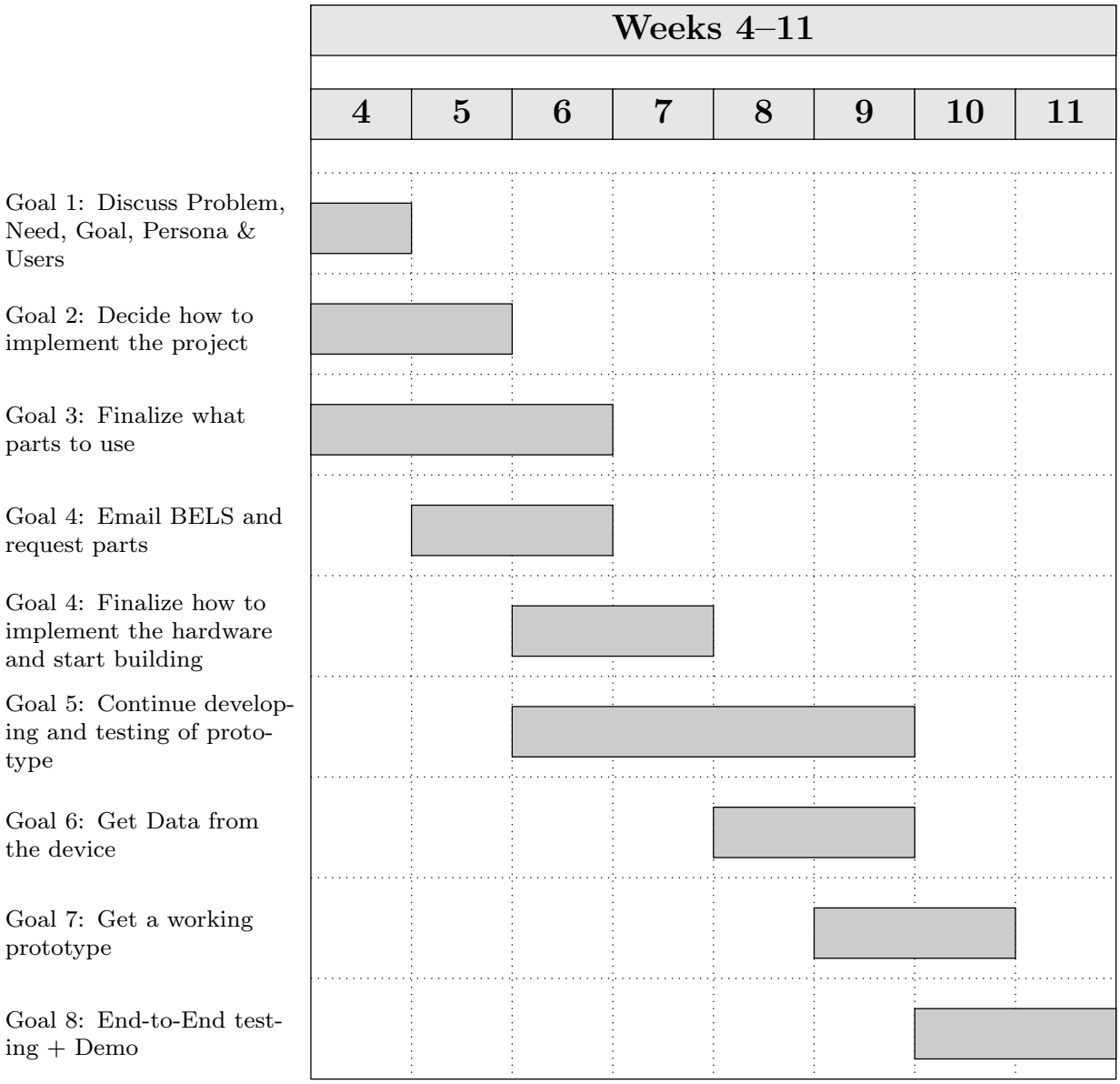
- College Roommates

  - John Doe, 20 years old, Student at UCSC. John lives in a dorm with two other roommates, and they all rely on a shared water filter for drinking water. Occasionally, John comes out to get water only to find that the water is empty. This creates frustration and delays as John has to choose between missing the bus to refill water or getting to class on time while being thirsty.

- Family
  - Jane Smith, 39 years old, working mother. Jane lives with her husband and two children. Jane gets home after a long, exhausting day of work and is feeling thirsty from the drive home. She goes to the shared water filter to find that it is completely empty. Jane feels frustrated with her stay-at-home husband and two children for not refilling the water filter. This causes her to lash out and yell at them until they all cry, creating a hostile environment in the household.

- Office
  - Joe Micheal, 30 years old, works with Excel spreadsheets. He has very limited time to refill his mug, and his boss, George, never refills the water filter. As a result, Joe's efficiency has substantially decreased at his desk.

Timeline



Decision Table

Items to decide on:

- Microcontroller
- Sensor
  - Ultrasonic: Pointed down from the lid onto a floating object
  - Load Cell: Place the container on the load cell to measure the weight and calculate the water level

  - Laser: Shoots a laser into the water from the lid, and the distance is sent back to the sensor
  - Camera: Setup to watch the water filter and measure the water level based on computer vision
  - Non-Contact Sensor: Patch that sits on the side, when water level goes below sensor it can alert the device

- Server
  - HTTP
  - AWS
  - Vercel

- Challenges
  - Signal in the fridge
  - Size limit
  - Contact Vs No Contact
  - Wifi connectivity
  - Power source (battery, USB, etc.)

Criteria (Weight)	**Capacitive**	**Weight**	**Laser**	**Ultrasonic**
Cost (0.30)	5	5	3	5
Power (0.20)	5	4	3	4
Complexity (0.25)	3	4	2	3
Measurement Data (0.25)	3	5	3	3
Weighted Total	**4**	**4.55**	**2.70**	**3.85**

Table 1: Decision matrix comparing feasible sensing approaches (1 = poor, 5 = excellent)

Based on the decision matrix, the weight sensor is the best sensing method because it has the highest weighted total score (4.55). It performs especially well in measurement quality while still keeping cost, power, and overall complexity at a strong level compared to the other options.

Testing Plan

There are multiple sections of our design that require testing individually, then together as a whole. The following list represents the pathway we will take when testing our prototype segments.

- ESP Data Reporting

- ESP Web Posting
- Vercel Receiving and Displaying Data
- Notification System
- Phone App Integration
- Prototype Setup Out-Of-The-Box

ESP Data Reporting

The pressure sensors we use to build the base will output data to our ESP device. We want to be sure the ESP is capable of receiving this data and displaying it to the debug terminal, which will be IDF MONITOR. Once we can verify posting to the terminal we will move to the next stage of the data reporting testing.

With data being displayed properly, we can now use this to interpret what an empty versus full pitcher could be expected to weigh. The first test will include to test a known weight to make sure we get accuracate weight readings. Next we will ensure the that when there is no wight on the plate, the data will be read as 0 g. Once this correct, we will move on to the more complex calibration with Pitcher + water.

We will use different types of containers given the variety of containers possible to be used by customers. We will test at different set water levels, and verify that the expected weight represents the following formula.

$$W_{total} = W_{container} + V * \frac{g}{mL} \tag{1}$$

Water weighs 1 gram per milliliter so if we keep track of the volume we put into the container we know exactly what to expect for output weight if we know the base weight of the container alone.

ESP Web Posting

To test this functionality the simplest way is to have the ESP send some sort of HTTP POST pointed at one of our local machines. We can receive and print this data and then move forward to testing sensor data output.

With a working sensor we can trigger a POST request when an event happens. In this test case it could be picking up the container from the base which triggers a POST request with some set information. Upon successful receipt of this we can confirm that the HTTP method is working on a small scale and move to the next segment of testing.

Vercel Receiving and Displaying Data

Once we have confirmed the ESP can send POST requests to a local machine, the next stage is to implement that using Vercel.

Notification System

With data being sent out of the ESP, the next stage is to test if we can send a notification on some set event. For a test case this could be a container being placed on the device. The data sent as part of this notification could be a value representing the weight of the container, or an estimate of how much liquid is inside.

If we know how much liquid is inside the container when we place it on the base, it is easy to verify if the value we receive in the notification is the correct volume. Using different volumes we can create a table to show how close the reported volume was, and tweak our math to get it closer to accurate with different containers.

When we pass the fake data tests, we will compare the debug terminal and the notification water levels to ensure that it is correct.

Phone App Integration

Prototype Setup Out-Of-The-Box

In order to set up the prototype, the web app must connect to the ESP through its hosted soft AP and scan a QR code to enter the WiFi credentials. To test this, we will try to connect to the ESP on boot multiple times using different devices, such as iOS vs Android, and see if it is still able to connect through soft AP. We will also test different types of WiFi networks, such as mobile hotspots, traditional password-protected networks, and networks with outside authorization such as university WiFi.

The user must also calibrate the device by measuring the empty and full weights of the filter on the weight sensor. On the webapp, they will be given instructions step-by-step to do this. In order to test this process, we will give the device to a new user and see if they are able to clearly follow the instructions and calibrate the device correctly. We will also test robustness by intentionally disobeying the instructions, such as by weighing it empty both times, to see our device handle it.

Web Code

The goal of this section is to define the web code we will be using to display information from the microcontroller. This will eventually migrate from a web implementation to a phone app, but in the initial stage it exists as a web display.

Initial Design

In its most simple form it must be able to display static data on a web page. We have created a .html file with this basic information, shown in the following figure.

From here there are a few changes that can be made. These changes follow the line of next steps to test this code and our designs.

- Dynamically updating data with a controller (Proof of concept testing w/ simple text)
- Sending and interpreting sensor data

Figure 1: Output of Initial HTML Code

- Graphic interface changes
- Sending text notifications to a phone

In order to test these things we will create a local server interface using one of our computers that hosts this data. Then we will use a microcontroller to send HTTP PUT requests. Finally, using the received data to change what gets displayed.

End Point of this Design

The plan is to host this using Vercel. The microcontroller is able to send data to Vercel that can be interpreted and displayed properly using Javascript. We are in the process of setting up Vercel to get this system working as intended. Once it is set up it is easy to migrate this initial testing code over to that platform and continue development from there.