Proposal

Household Water Usage Monitoring System

Team 42
Daniel Baker (drbaker5), Advait Renduchintala (advaitr3), Jack Walberer (johnaw4)
drbaker5@illinois.edu, advaitr3@illinois.edu, johnaw4@illinois.edu

TA: Pusong Li

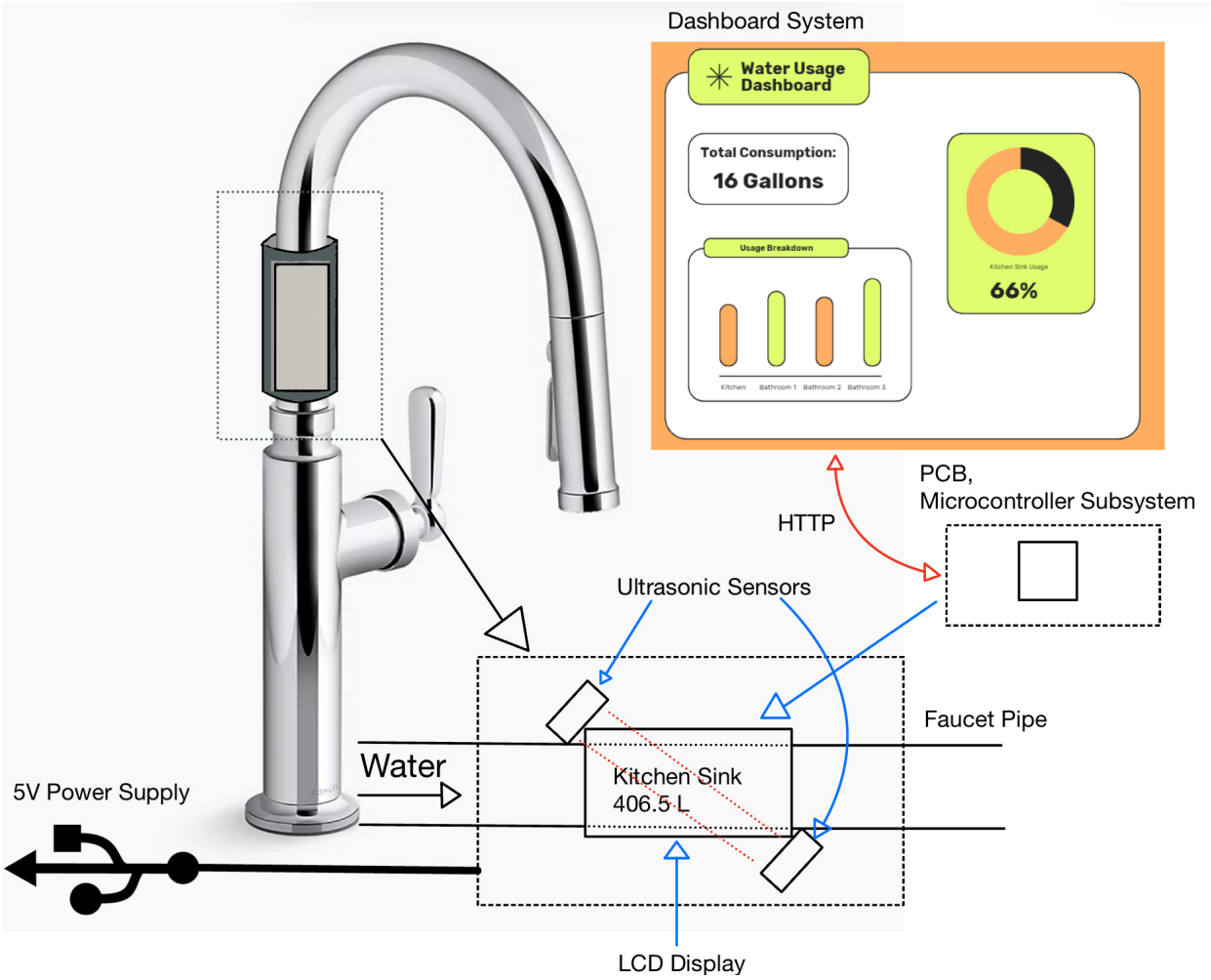
ECE445

09/19/2024

1 Introduction:

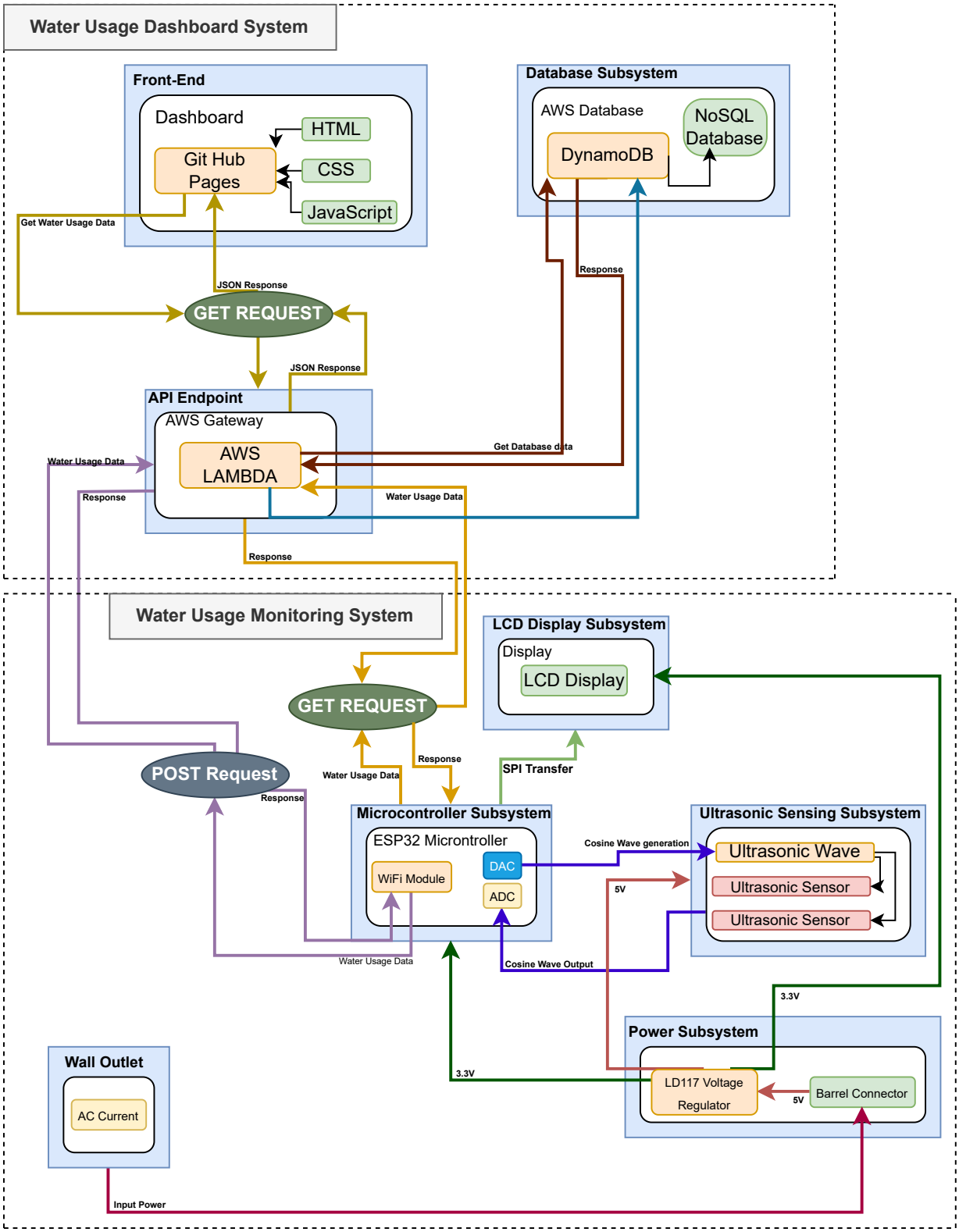
- **Problem:** Apartments charge utilities every month. Part of the utilities includes water cost, and in an apartment it is not always clear why the utility bill is high or where all the water is being used. Additionally over use of water is a problem more generally, and saving water conserves natural resources, promotes sustainability and conserves the environment. Our device can pinpoint where the water usage is coming from and how much water is being used.
- **Solution:** Our product is a device designed to attach to the end of faucets around the house and it will monitor the amount of water used by each faucet. Each time the faucet is turned on, the LCD screen will display a count of the measurement of water that had been dispensed since the sink was turned on. Multiple devices can be used in the same house, they will connect to a website dashboard which display stats about each connected device and their specific/aggregated water usage that month. For example, Kitchen sink, 50% of total usage, 100 gallons used.
- **Benefits and Features:**
  - Become more conscious of water usage in general, which increases sustainability.
  - An inexpensive way to monitor water usage, reducing utilities cost.
- **High-level requirements list:**
  - Ultrasonic Sensing and Microcontroller Subsystems can accurately measure and calculate flow rate and total water usage, determine flow with less than a 10% difference of the real amount of water dispensed. We will measure the real amount of water dispensed using a measurement cup.
  - The Front-End, API, Database, and Microcontroller Subsytems must provide conistent and available water usage data. The webpage must load (fetch water usage data from database) and display the dashboard within 1 second for 90% of requests assuming a standard internet connection (10 Mbps). The database and dashboard data must be consistent with the microcontroller data displayed on the LCD screen.
  - The Household Water Monitoring System must be resilient to network interruptions and power fluctuations, ensuring continued operation and data integrity across all subsystems. If network connectivity is lost, the Microcontroller Subsystem will continue collecting water usage data and store it in the Flash Memory. Once reconnected, the Microcontroller Subsystem must automatically synchronize the stored data with the Database Subsystem, no matter the time of day. This differs from the automatic daily synchronization of water usage data from the Monitoring System. Additionally, the Front-End Dashboard should display an alert indicating any data gaps caused by outages, ensuring users are aware of any temporary data inconsistency. This will be verified through stress testing, simulating both network and power interruptions.

- **Visual Aid:**


Dashboard System
Water Usage Dashboard
Total Consumption:
16 Gallons
Usage Breakdown
Kitchen
Bathroom 1
Bathroom 2
Bathroom 3
Kitchen Sink Usage
66%
PCB,
Microcontroller Subsystem
HTTP
Ultrasonic Sensors
Faucet Pipe
Water
Kitchen Sink
406.5 L
5V Power Supply
LCD Display

2 Design:

- **Block Diagram:**



- **Subsystem Overview:**

  - **Dashboard System - Frontend:** The user will interact with the water usage data on the online dashboard created through Github Pages, HTML, CSS, and JavaScript. The usage data is gathered during communication with the Monitoring System through API calls.
  - **Dashboard System - API Endpoint:** AWS Lamda will be middleman between the frontend, database, and monitoring system. When an API endpoint is hit, AWS Lamda will relay the updated data to the frontend dashboard, database, or monitoring system.
  - **Dashboard System - Database:** Through AWS DynamoDB, this will simply store records of each water monitoring system, including its ID, user given name, and water used. The frontend will request information from the database to update water usage on its dashboard. DynamoDB will send HTTP POST requests to respond to HTTP POST requests from AWS Lamda.
  - **Monitoring System - Microcontroller Subsystem:** All computation of water usage will be done on the ESP32 microcontroller. ESP32 will be connected to the Ultrasonic Sensing System which will transmit and receive its cosine wave forms through its DACs and ADCs for measuring flow rate in the pipe. ESP32 will also be connected to the LCD screen, displaying the device name and water usage through SPI. It has an internal 2.4 GHz WiFi system for communication with AWS Lamda.
  - **Monitoring System - Power Subsystem:** The Power Subsystem is designed to provide reliable and efficient power distribution to all components within the a Monitoring Device of household water monitoring system. The Power Subsystem receives a 5V input from a USB power source to be used by the LCD screen and MCP6001 Op-Amp. It will also utilize the LD1117-3.3V linear voltage regulator to additonally provide a 3.3V supply for the ESP32 Microcontroller. The Power Subsystem will include a jumper-selectable voltage for signal input/output for the ultrasonic sensing system. While the datasheet uses 3.3V for its example signal, we know that a 5V signal amplified by the MCP6001 Op-Amp could be more effective when deriving transmission time, ultimately leading to a more accurate flow measurement. Thus, we will incorporate jumpers on each of the signal paths to the ultrasonic sensors, shorting the one that we select.
  - **Monitoring System - Ultrasonic Sensing Subsystem:** Using two H2KMPYA1000600 transducers, we will take the given cosine amplified waves and transmit the signal. Each will also receive signals from each other and relay them to the ESP32's ADC. The time difference between transmission and reception will help us determine the flow rate in the pipe.
  - **Monitoring System - LCD Display Subsystem:** The LCD Subsystem in the water monitoring device is responsible for displaying water usage data. This subsystem uses a simple LCD screen that allows users to see how much water has been used since the faucet was activated. It reads data through the SPI (Serial Peripheral Interface) protocol, which is used for communication between the ESP32 microcontroller and the LCD.

- **Subsystem Requirements:**

  - **Dashboard System - Front-End:** Must be able to initiate data requests through AWS Lamda's AWS API Gateway. Upon receiving a response for this request, it must update its UI with the latest water usage data.
    * Unit Test: Given POST request in JSON format from AWS Lamda below
      { "id": "432doissrv4-ghw8405mxjvurxj-4jvjw84kcos",
      "usage": 453.23 }
      The Dashboard System must update the device with id "432doissrv4-ghw8405mxjvurxj-4jvjw84kcos" to have water usage to be 453.23 mL on its front end to ensure accurate data reporting.
    * Unit Test: Given user changes a device name, by clicking "Save Changes" button after inputting the new username, the Dashboard System must immediately trigger using an eventListener to have AWS Lamda send the following POST Request in JSON format to the Database Subsystem using its URL configured in AWS Lamda:
      { { "id": "432doissrv4-ghw8405mxjvurxj-4jvjw84kcos",
      "username": "Kitchen Sink" }

- **Dashboard System - API Endpoint:** AWS Lamda must be able to send/receive HTTP GET/POST requests to/from the Microcontroller, Front-End, and Database Subsystem. This will be done through configuration of request routing for AWS Lamda on aws.amazon.com. The unit testing of correct configuration is done on all systems that interact with AWS Lamda, such as Front-End, Database, and Microcontroller Subsystems.
  - **Dashboard System - Database:** AWS DynamoDB must be able to send/receive HTTP GET/POST requests and respond with requested data or update the database accordingly.
  - Unit Test: Given POST request in JSON format from AWS Lamda below
    { "id": "432doissrv4-ghw8405mxjvurxj-4jvjw84kcos",
    "username": "Kitchen Sink" }
    The Dashboard System must update the row with id "432doissrv4-ghw8405mxjvurxj-4jvjw84kcos" to have the column "username" to be "Kitchen Sink to ensure Database, Monitoring System, and Front-End agreement.
  - Unit Test: Upon a GET request from AWS Lamda, the Database System should trigger a POST request with the following data to update the Front-End or Microcontroller Systems
    {"id": "432doissrv4-ghw8405mxjvurxj-4jvjw84kcos",
    "username": "Kitchen Sink",
    "usage": 453.23 }

- **Monitoring System - Microcontroller Subsystem:** ESP32 must be able to provide and receive 3.3V Amplitude Cosine Waveforms from its DAC and ADC. Must use these signals to calculate ΔT between send/receive, and use this ΔT to calculate the water flow rate in the pipe. 2.4GHz WiFi system must send/receive HTTP GET/POST requests. Must communicate through SPI with the LCD screen.
  - Unit Test: Given POST request in JSON format from AWS Lamda below
    { "id": "432doissrv4-ghw8405mxjvurxj-4jvjw84kcos",
    "username": "Kitchen Sink" }
    The ESP32 must update its flash memory such that the device with id of "432doissrv4-ghw8405mxjvurxj-4jvjw84kcos" a corresponding value of "Kitchen Sink" for its username. We will use a dictionary data structure in C/C++ to accomplish this. The SPI communication with the LCD should reflect this change.
  - Unit Test: Upon periodic update cycle, the Microcontroller System should send a POST Request to the Database System with the following data
    {"id": "432doissrv4-ghw8405mxjvurxj-4jvjw84kcos",
    "usage": 453.23 }

- **Monitoring System - Power Subsystem:** Must provide stable 5V and 3.3V Power sources through the barrel connector and LD1117-3v3 linear voltage regulator. We will verify this by measuring the input 5V voltage and the output voltage of the LD1117-3v3 linear voltage regulator using a multimeter.
- **Monitoring System - Ultrasonic Sensing Subsystem:** Must transmit signals from ESP32's DAC to the other Ultrasonic Sensor and vice versa. MCP6001 must amplify the cosine waveforms to/from 3.3V and 5V. We will verify the the trasnimission and reception of the ultrasonic sensors by measuring the output of one of the sensors using an oscilloscope.
- **Monitoring System - LCD Display Subsystem:** The LCD must correctly display water usage values in a clear, readable format with no data corruption for at least 99% of updates, which will be manually verified through 100 test cases.

- **Tolerance Analysis:** Tolerance Analysis

  Our project needs to measure amount of water coming out of a faucet. The main reason this measurement would be off is timing. Two signals are sent across the ultrasonic sensors from the DACs to the ADCs. The timing of these signals will give us the information we need to calculate flow rate of the water. When a signal is sent from the DAC and received at the ADC there will be some internal delay. The ESP32 runs at 80MHz. Using some calculations below, here is the maximum amount of clock cycles that the timer can be off by to ensure a measurement that is within 20 percent of the true value.

  The approximate time it takes for a sine wave to cross the faucet is found by taking the speed of sound through various mediums times the distance of the medium. The faucet this product will be build for has .25in aluminum

as a pipe encircling a .5in diameter spout for water. Since the sensors are at 45 degrees, the distance of travel through the faucet is root 2 times these values times the velocity of sound in the medium. For the calculations the distances were changed meters. 6320 m/s is the speed of sound in aluminum, and 1482 m/s is the speed of sound in water. We need the flow rate of water up stream. Flow velocity can be found by

$$Q = vA$$

where Q is flow rate, v is velocity and A is cross sectional area. Based on the average faucet producing 2 gallons per minute (GpM) and the cross sectional area of the water spout to be pi*(.25in*.25in). In SI units this is-

$$1.26 * 10^{-4} m^3/s = v * 1.266 * 10^{-4} m^2$$

Then, v = 0.995 m/s. So the velocity of the signal in water up stream is 1482 - .995 or about 1481 m/s.

$$(\sqrt{2} * .00635m * (1/6320m/s)) + (\sqrt{2} * .0127m * (1/1481m/s)) = t1$$

To get the time down stream we can just add the v flow to the velocity in water. So we get 1483 m/s.

$$(\sqrt{2} * .00635m * (1/6320m/s)) + (\sqrt{2} * .0127m * (1/1483m/s)) = t2$$

Take the difference of these terms.

$$t1 = 1.35482 * 10^{-5}$$

$$t2 = 1.35319 * 10^{-5}$$

The time difference is on the scale of $10^-8, and that is the same time scale of our clock cycle, which is 80MHz, so we will be adding a 3 clock cycles, the error would be very high. Because in this case, the time difference is 1.6*10^-8 seconds, and the clock has a rising e 10^-8 seconds. So (1.6 - 1.25)/((1.6 + 1.25) * .5) = 25$

This error is simple too high, so we will add an external device that can have a faster clock cycle allowing us to gather accurate data.

3 Ethics and Safety:

There are a couple of possible ethics and safety concerns in our project. Our first concern has to do with avoiding harm for the user. Our device is an electric circuit board, with voltage running out of the wall, close to running water. We need to make sure the casing for out PCB and LCD screen and other modules don't let water in then because that could electrocute the user and break the circuit board.

Another possible concern has to do with respecting privacy and confidentiality, these device take data about the water usage of the household they are monitoring and store them on a website. We need to make sure that the data stored on the website does not get out so that the privacy of our users is protected.

References:

ACM Code of Ethics and Professional Conduct. Code of Ethics. (n.d.). https://www.acm.org/code-of-ethics

IEEE - IEEE Code of Ethics. (n.d.). https://www.ieee.org/about/corporate/governance/p7-8.html