18-549 Project Proposal

StreamFi

Team 10

John Bird

Selin Sirinterlikci

Evans Hauser

Nick Wilson

February 10, 2017

Table of Contents

1. Project Description

StreamFi is a liquid level sensor that is attached to the outside of any container holding liquid. StreamFi determines if the fill level is close to empty or full and notifies the sensor owner. The sensor and notifier will be self-contained and self-powered. Ideal applications of StreamFi are water containers at sports events or restaurants, kegs at stadiums or other large events such as music festivals, brake or wiper fluid in cars, and propane tanks. In addition to determining an empty condition, StreamFi could be used to check the full condition, for example the fill level of rain gauges or other reservoirs. Notification will be done through an application that allows the user to easily determine which sensor is nearing its critical condition. In addition, the sensor will notify the user when the sensor runs low on power. StreamFi could also reduce inventory costs for companies that store large quantities of liquid containers by allowing them to place orders only when exactly needed.

2. Design Requirements

Explicit Requirements:

- Shall determine full and empty condition
- Shall notify the user on the critical condition
- Shall be non-intrusive to container
- Shall be uniquely identifiable and specific to a user
- Shall be easily maintainable
- Shall send reliable data
- Shall store the data from the device
- Shall be durable and robust in case of physical wear
- Shall be liquid resistant

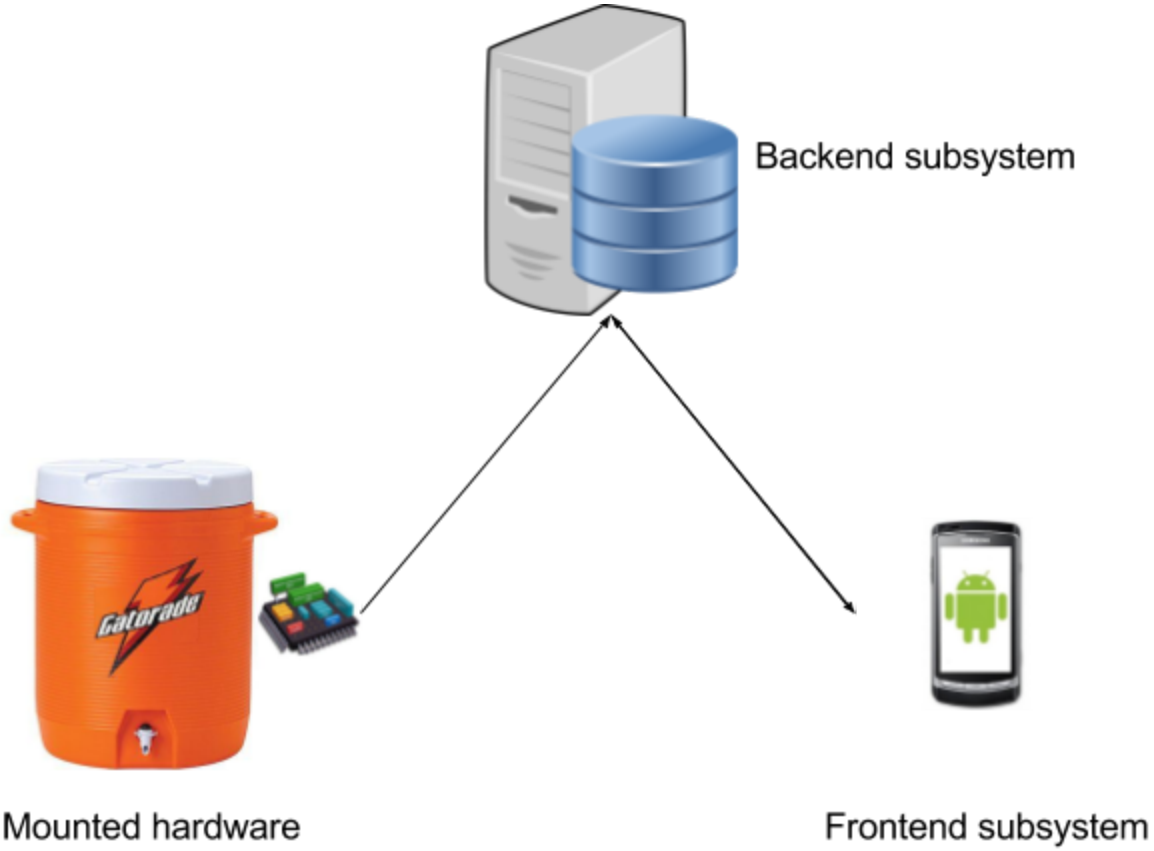
Implicit Requirements:

- Should send notification when container is empty or full, with a 15% margin of error
- Should communicate over WiFi with the end user's device
- Should send data at an interval that allows the backend to determine the status of the device
- Should communicate with a backend that determines which sensor is triggered and send the notification to the associated user
- Should be attached on the exterior of container
- Should be replaceable and detachable in under a minute

- Should take less than 10 minutes to calibrate and no more than two state changes
- Should detect temperature of container's contents
- Should provide timely notification of lower container levels to relevant users
- Should display the usage data from each device in an informative manner that includes multiple devices across user-defined intervals
- Should hold all data received from the device for the lifetime of the device
- Should provide a notification when running low on power or close to failure
- Should have implicit power saving, when not attached to a container
- Should be able to function after a five foot fall
- Should be encased in a protective outer shell
- Should be spill proof and capable of withstanding submersion in liquid

3. Functional Architecture

The StreamFi system will have three major subsystems: the mounted hardware, the backend subsystem, and the frontend subsystem. These subsystems are depicted in Figure 3.1.



The mounted hardware detects the level of the liquid in the container by frequency response analysis. To collect this status, the hardware module sends a wave through the liquid and receives a response. From the response, the hardware determines the response and sends data real time from the response of the liquid. This data is sent over a WiFi connection to the backend system. In the event that the hardware cannot process the data in real time, the frequency response will be sent to the backend for processing. Ensuring reliability, the sensor will send the data to the backend regularly to form a heartbeat system.

The backend server provides an hardware-accessible endpoint. The backend then processes the data and determines the fill level of the container. When the level reaches a user-defined threshold, the backend notifies the frontend. In addition to processing the frequency response, the backend server is responsible for storing the data from all hardware in the field for the hardware's lifetime. After processing, the backend places the sanitized data to a database that is shared with the frontend.

The frontend is the client facing side that consists of a mobile application or simple text and email notifications. These notifications originate from changes to the shared database. In addition to user notification, the front end displays the data with graphs that are customizable along user defined intervals.

4. Design Trade Studies

4.1 Sensor(s)

Initially, we plan to start by sensing liquid level by using the combination of a transducer and a piezoelectric sensor. Many smart water bottle products on the market at the moment are using accelerometers, such as the one seen in Figure 4.1. However, smart water bottle products that rely on accelerometers indicate that their drink detection is being done by a drinking motion. Since StreamFi is for use on larger beverage containers, an accelerometer is not a feasible option since users would not be tilting a water cooler or large container every time they get liquid from it.

Fig 4.1: ADXL335 Accelerometer

For the sensor, we have debated which method is best to detect liquid levels across different kinds of containers. We had discovered that another research group had been able to detect where users were touching an object depending on the resonant frequency when sound passed through the object, and we were hoping to apply a similar principle to this when determining water levels in a container. Our thought is that we can use a transducer to pass sound through the container and determine a “calibrated” value when the container is empty, and then approximate how full the container is depending on the variation from that calibrated value.

When examining the scope of our project and its intended use for large beverage containers, we determined that the most feasible way to sense the liquid levels in a large container would be through this method of using a transducer and piezoelectric sensor to detect resonant frequencies coming from the interior of the container.

4.2 Backend Server

There are two possible locations for the signal processing in the backend or on chip. On-chip processing decreases the amount of data sent out through the WiFi connection. Processing in the backend provides a more developer friendly environment that provides an easy avenue to upgrading the processing algorithms. In addition, outsourcing the processing to the cloud decreases the power consumption of the chip. Due to the power savings and data and developer compatibility, cloud processing is the clear choice for signal processing.

For our backend server platform, we have decided to go with Amazon Web Services (AWS). We feel that AWS is a reliable and secure method of storing information and performing any necessary signal processing calculations. AWS has a variety of services

that we could potentially use, but after examining the different services, it appears that we would be able to use DynamoDB for storing information and Lambda for any signal processing calculations after the raw data is uploaded from the hardware using WiFi connection. As compared to other alternatives, such as building a custom server or using Google Cloud Platform, AWS is a platform with a large variety of services that tailor to our needs pretty well. We have decided that AWS is the best suited for our needs, so that is the backend platform that we will be using for this project.

4.3 User Application

For the user application, we aim to provide a system that is able to notify the appropriate user(s) when certain container levels become low, or if there is another feature (such as warm liquid) that users should be notified about. When considering the skill sets of our team members, it was determined that developing this application for Android phones made more sense than for iOS devices, since the team was more familiar with Android development than iOS development.

With this application, we plan to have the user be notified whenever the container is reaching a critically low level, which would allow them to have enough time to replace the contents in the container.

5. System description / depiction

As described in the Functional Architecture section, the StreamFi system will have three major subsystems: the mounted hardware, the backend subsystem, and the frontend subsystem. This section describes these subsystems in detail.

5.1 Mounted Hardware

This subsystem will collect data from the container and communicate it to the backend subsystem. It consists of several components: power, mounted communication, a transducer, a vibration sensor, and a mounting adhesive, as depicted in Figure 5.1.

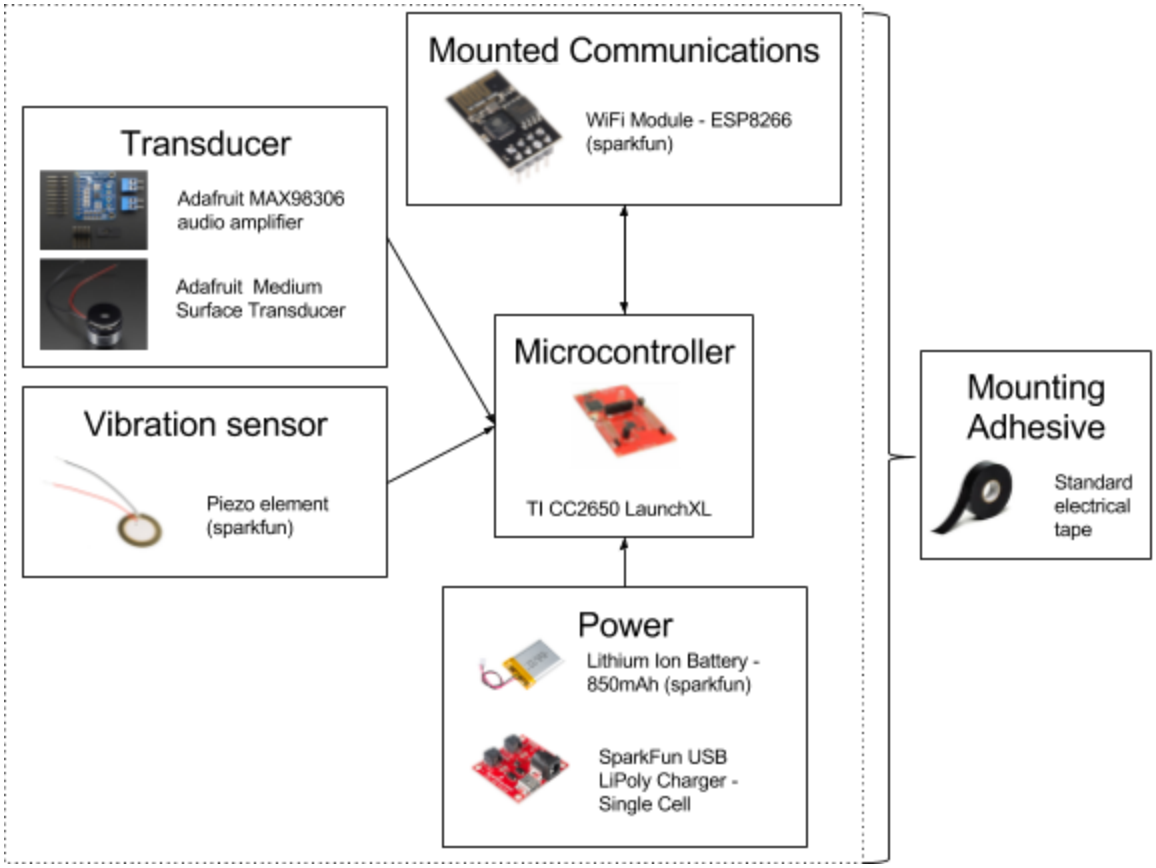


Fig 5.1: Mounted Subsystem Breakdown

5.1.1 Power

Because the system is to be mounted on the side of a container and attached, via the mounting adhesive subsystem, the system must have low weight. Too much weight could overwhelm the mounting adhesive, causing the entire system to fall and fail. Thus, one of the primary considerations for the power component is for it to be small and light. The other primary consideration for the power subsystem is for it to last for long periods of time, to minimize the inconvenience of recharging the subsystem. In particular, the device must be powered for at least a full day at a time. The power system should be rechargeable.

5.1.2 Mounted Communication

The entire mounted subsystem is to communicate data via WiFi technology to the backend and frontend subsystems. This communication will be performed by the mounted communication component. This component will be a simple WiFi module that is connected to the Mounted Subsystem.

5.1.3 Transducer

The transducer will be responsible for applying sweeping frequencies across the container. These vibrations can be read by the vibration sensor subsystem. The transducer component of the mounted subsystem will consist of both a transducer and an audio amp, to make sure that the sound is strong enough to be read by the vibration sensor.

5.1.4 Vibration Sensor

The vibration sensor will read back the sounds played by the transducer subsystem, seeing how the container plays them. As shown in other projects, an object will play these vibrations differently when it is touched or not touched; our team believes this principle can be applied the containers with fluids inside of them. The containers will play the vibrations differently when they are full, when they are empty, and when they are at different levels in between. The vibration sensor will sense these vibrations, and transmit them to the controller of the system to be processed, and translated into meaning.

5.1.5 Mounting Adhesive

The mounted subsystem will need to be attached to the container, so that both the transducer and vibration sensor can have direct contact with the container's surface. The mounting adhesive will connect the mounted subsystem to the container in such a way that the transducer and vibration sensors are flush with the container. Our team is currently considering several different alternatives for this component, including a strap that wraps around the container, velcro, or sticky patches.

5.2 Backend subsystem

The backend subsystem will be responsible for receiving and storing data received from the mounted subsystem. In addition to maintenance of the data, the backend calculates the fill level of the container from the data received. The backend subsystem is also responsible for making this data retrievable by the frontend subsystem, and pushing messages to the frontend when the container is nearing empty.

The data storage will be done in S3 and processing will use AWS Lambda. S3 is a scalable object storage and AWS Lambda is a service that runs code on specified events, such as a file added to S3. Each device will push its data to the S3 bucket through its provided http endpoint periodically and when the data arrives, the AWS Lambda function will run to calculate the fill level form the frequency response and push the data to another S3 bucket or DynamoDB instance that is accessible to the frontend.

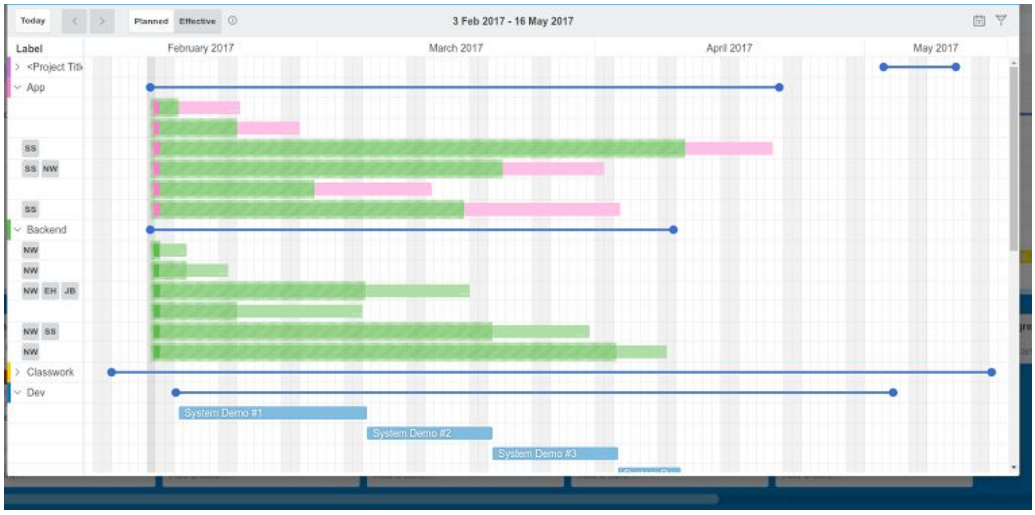
5.3 Frontend subsystem

The frontend subsystem will take the form of an android application that displays data from the backend, and creates push notifications when the container is nearing empty. The frontend subsystem is responsible for visualizing data it retrieves from the backend in a meaningful way, making this data easy to understand for the user, and creating push notifications from pushed messages that it will receive from the backend subsystem.

6. Project Management

6.1 Schedule

- We will use Trello to manage our project. Using Elegantt (https://elegantt.com/) we generated a Gannt Chart from our boards. The board (and chart) can be viewed at https://trello.com/b/FWAuMq2b/capstone



Example shot of Gantt Chart

6.2 Team Member Responsibilities

- Nick
  - App development and front end
  - Signal processing
  - Attachment strategies + form factor
  - Website updates + frontend
- Selin
  - Backend integration
  - Signal processing
  - Website setup + backend

    - Manufacturing
- Evans
    - Assembly
    - Firmware
    - Signal processing
- John
    - PCB design
    - Firmware
    - Signal processing

6.3 Budget and Parts List

- Microcontroller
    - TI CC2650
        - $9.50
- WiFi Module
    - ESP8266
        - $6.95
- Transducer
    - COM-10917
        - $9.95
    - Needs an audio amp
        - MAX98306
            - $8.95
- Vibration Sensor (piezoelectric)
    - SEN-10293
        - $1.95
- Battery / Charging Circuit
    - Charging Circuit
        - PRT-10217
        - $7.95
    - Battery
        - PRT-13854
        - $9.95

The cost of major components totals in at **$55.20**. This leaves room for a few board iterations if we choose to alter the design and optimize for power consumption. It additionally leaves room to add more sensors if we choose to expand our sensing capabilities. Besides components, some funds will need to be allocated for board housing, as well as some means of fastening the device to various containers. Placing

$100 aside for both of these aspects still leaves us with a portion of our budget set aside for unexpected challenges.

6.4 Risk Management

We have identified three major areas of risk associated with our project.

6.4.1 Design

One challenge we will face is the device's ability to fit different form factors. Ideally, the sensing unit should be adaptable to most shapes and sizes of containers. A flexible piezoelectric element and rear mounted transducer will allow for direct mounting of the sensing device to any vessel of sufficient size. Obviously keeping device size down will be important for retrofitting to smaller containers.

Figuring out a reusable way to attach the piezoelectric sensor will be another challenge. For early stages we plan to tape the sensor on, however finding a sleeker solution will be important later in development.

6.4.2 Schedule

Maintaining a schedule will be imperative to success, because many of the application / backend features require data from the sensing units. We need to ensure careful work goes into hardware and firmware development to ensure we have available data to manipulate. While boards / firmware are moving along the device pipeline, sample datasets can be used as stubs for developing the Android Application / data server.

6.4.3 Resources

Budget and personnel are two other factors to take into consideration for this project. With a wide range of skills, our team was able to divide the project into (hopefully) modular parts. This allows us to develop parts of the project in parallel throughout the semester. Save for a few early dependencies, few problems should halt the development flow. As mentioned earlier, some data or functionality can be emulated with stubs / development boards while we wait for hangups to be sorted out.

The risk of running over budget can be mitigated by being mindful of our performance / price requirements. Using the cheapest components that will not compromise accuracy and performance can help to keep our hardware affordable.

7. Related Work

When researching smart sensors on beverage containers, we found that StreamFi had similar goals and functionalities of three products on the market:

7.1 Vessyl

Vessyl is a product that is currently in development, with orders available at $199. Its goal is to track beverage consumption by determining which beverage is in the bottle. In addition to calculating usage statistics, it is able to keep track of how many calories someone is consuming through beverages. It informs them via a mobile app about how they are performing in terms of their health and fitness goals, as it connects to popular wearable fitness devices, such as Fitbit. This product differs from our product in that Vessyl tracks an individual user's beverage consumption through an individual bottle, whereas we hope to track usage of large beverage containers and allow our sensor to adapt to multiple different containers. Despite these differences, the product is actually fairly similar in that it provides the user notifications about multiple different beverages that they are drinking, which is something we hope to do with our product.

7.2 H2OPal

H2OPal ($99) is a product that attaches to the bottom of a water bottle and tracks usage statistics. Similar to Vessyl, it automatically tracks your intake (this time of just water) and allows the user to keep track of their intake through a mobile application. While H2OPal focuses primarily on water intake for individual users which is different than what we are doing, H2OPal is similar in the form factor of the product. We aim to have something easy to use and detachable.

7.3 KegBot

KegBot ($129/tap) tracks consumption of a beer keg across multiple users, determining how much each user is consuming by having them sign in using the Android NFC functionality. This is pretty similar in overall scope of the project, since it provides real-time usage statistics on a tablet interface next to the tap and also through user statistics online. One major similarity between KegBot and our project is that we also hope to measure consumption of a larger container across multiple users, and to notify whomever is responsible for re-filling the container before it is empty.

Overall, our product can be viewed as a combination between the real-time notification of each of these products, combining the ability to measure multiple liquids similar to Vessyl, a form factor that is removable from the container such as H2OPal, and a product that tracks usage across multiple users such as KegBot.

8. References

- Vessyl: https://www.myvessyl.com/vessyl/
- H2OPal: https://www.h2opal.com/products/h2o-pal
- KegBot: https://kegbot.org/