Weightb0ard Design Document

Team - Thomas Driscoll, August Gress, Kyle Patel
ECE 445 Project Proposal - Spring 2020
Group 8
TA: Dhruv Mathur

Table of Contents

1 Project Overview

1.1 Objective

In any environment that involves cooking or food preparation, knowing the amount of ingredients on hand is of the utmost importance. This can range from large-scale restaurants that have massive quantities of any given food to athletes engaging in meal prep, often down to the gram. In between these two extremes also exist the average consumer, who rely on unreliable memory and insatiable hunger when shopping instead of their objective needs.

Our solution is an internet connected, weight-sensitive kitchen cabinet/tray that pings a grocery list app. For items such as rice, sugar, flour, protein powder, creatine, etc., a scale could measure the amount at home. If it falls below an ingredient-appropriate threshold, a microcontroller will send an update to a user's phone. Simply checking the app once in the store, or while placing a large order, allows the user to purchase the correct amount of food. It will have 7 separately sized sensors that accurately measure ingredient amounts placed on top of it, which will be a proof-of-concept to show our idea's scalability.

1.2 Background

Kitchen preparation is a part of everyday life for many people. From creating meals for themselves or their families to working in a high pressure industrial kitchen that cranks out pounds and pounds of food an hour, there is a constant need to know which ingredients are on hand.

In the pursuit of not running out of ingredients; however, there comes a tendency to overbuy ingredients, generating food waste. Food waste is a global concern, and is the subject of many different studies and research articles on its effects on the environment and society [1]. In addition to overbuying, a business or individual can also forget a particular ingredient that they needed to make a recipe simply because they didn’t know they had ran out of it. One potential solution that has been brought to market is the Samsung Family Hub Smart Refrigerator, which has a 3 camera array on the inside of the fridge to view its contents from wherever you are [2]. One large issue with this however, is that it is difficult to

quantify how much is left of an ingredient as the camera is mounted where you only see the outside packaging of each ingredient. This is something our board proposes to solve by individually weighing each ingredient.

We saw a need to create a way to simplify the lives of everyone who relies on what ingredients they have on hand and prevent overbuying on ingredients that they didn't need. By creating this board, we aim to simplify the lives of those who depend on having the correct ingredients.

1.3 High Level Requirements

- Able to measure the weight of ingredients placed on pressure sensitive pads within ±10% of true weight.
- Board will send information about the ingredients on it to an internet connected database.
- Weight (in grams) is reported in a mobile application using the data stored in that internet connected database

1.3 Visual Aid

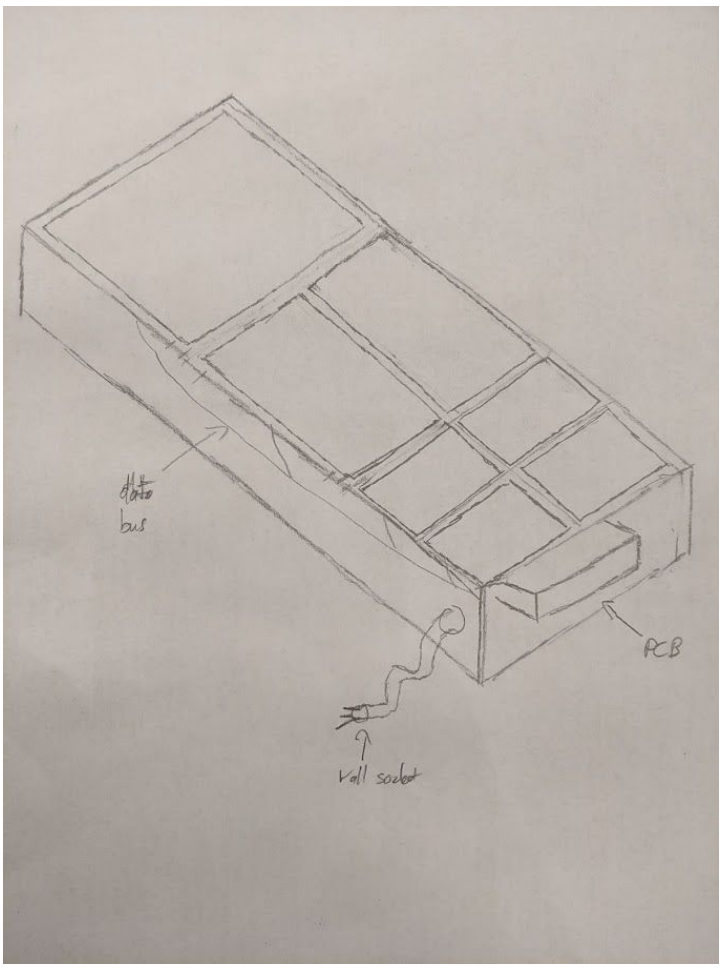

Figure 1: Pictorial representation of the Weightb0ard

2 Design

2.1 Block Diagram

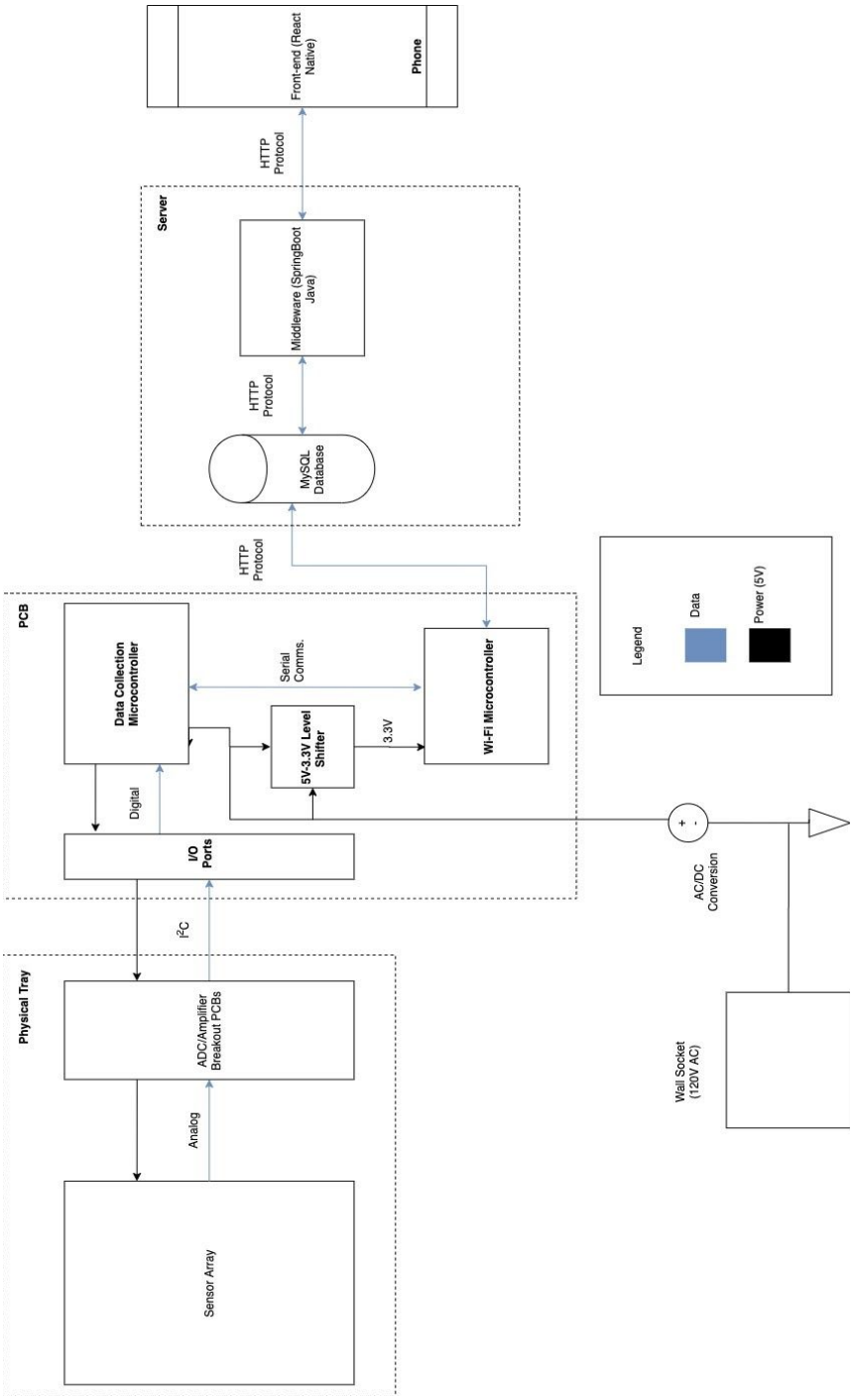

Figure 2: Block Diagram

2.1.1 Physical Tray:

The tray will support 7 bar-type sensors that will be processing the real-world input of each ingredient. They will be mounted in a cantilevered fashion with attached weighing platforms of varying sizes above the tray's center. 4 sensors are rated to measure a maximum of 100g of weight, 2 are rated with a maximum of 500g, and 1 sensor is rated with a maximum of 5kg. Each sensor will have an associated board on the tray, which will send weight data to the data collection microcontroller's GPIO pins via an I^2C-like interface. The microcontroller will send a common clock signal to each breakout board. The boards receive power from GPIO pins and provide amplified voltages as excitation inputs for each sensor. Bar-type sensors are considerably cheaper than transducer type load cells and are still quite accurate.

A sensor contains 4 strain gauges in a wheatstone bridge configuration [3]. The differences in strain between the two sets of gauges (SG1 and SG2 vs. SG3 and SG4) when one end of the load cell is bent produces an output voltage, which is amplified and converted to 24-bit digital values on the aforementioned breakout boards. A load cell has 4 wires as seen in the diagram below. The EXCITATION+ input corresponds to the input voltage from the breakout board's amplifier and EXCITATION- is grounded. The OUTPUT+ and OUTPUT- wires are inputs for the breakout board's A/D converter.

Each breakout board will output data at a rate of 10Hz (which it does without additional input). There is the choice to gather sensor data at 80Hz, but sampling at that rate increases the amount of digital conversion noise in weight readings [4]. Having precise readings is more in line with our high-level requirements despite the higher but still subsecond settling time at 10Hz compared to 80Hz.

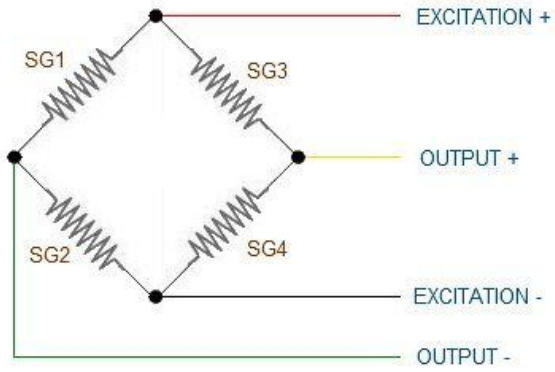

Figure 3 : Wiring configuration of load cell [3]

2.1.2 Physical Tray Schematic

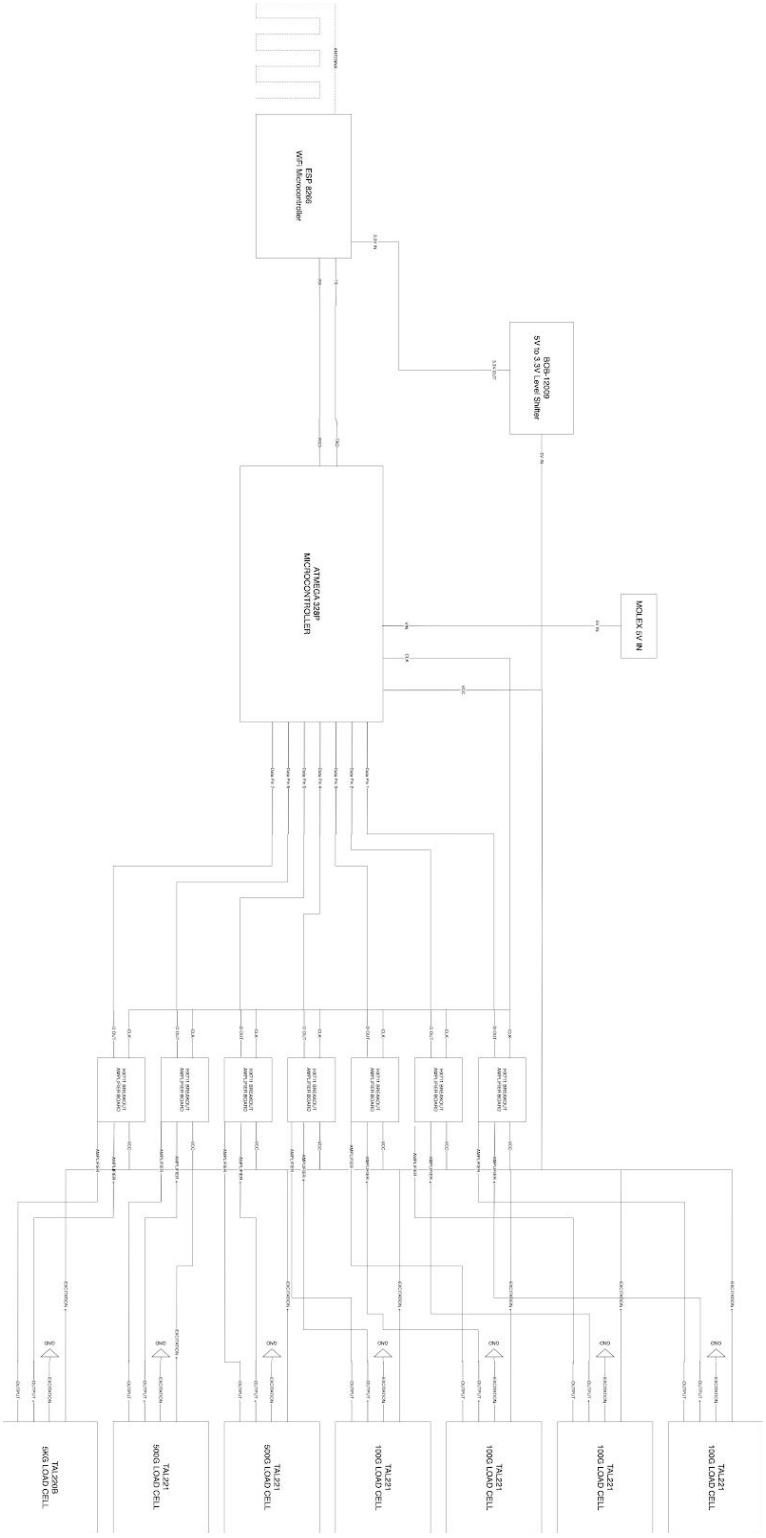

Figure 4: Physical Tray Schematic

Below we have included some close-up figures with explanations of the schematic. Figure: Schematic-A shows the microcontroller and how it interfaces with the ESP8266 WiFi Microcontroller. We also show the level shift from the 5V V_{cc} line to the 3.3V IN on the ESP8266 to power the ESP8266. We also show the TXD->TX and RXD->RX connections between the ESP8266 and the ATmega328P. These are used to facilitate send (TX) and receive (RX) communication between the two microcontrollers.

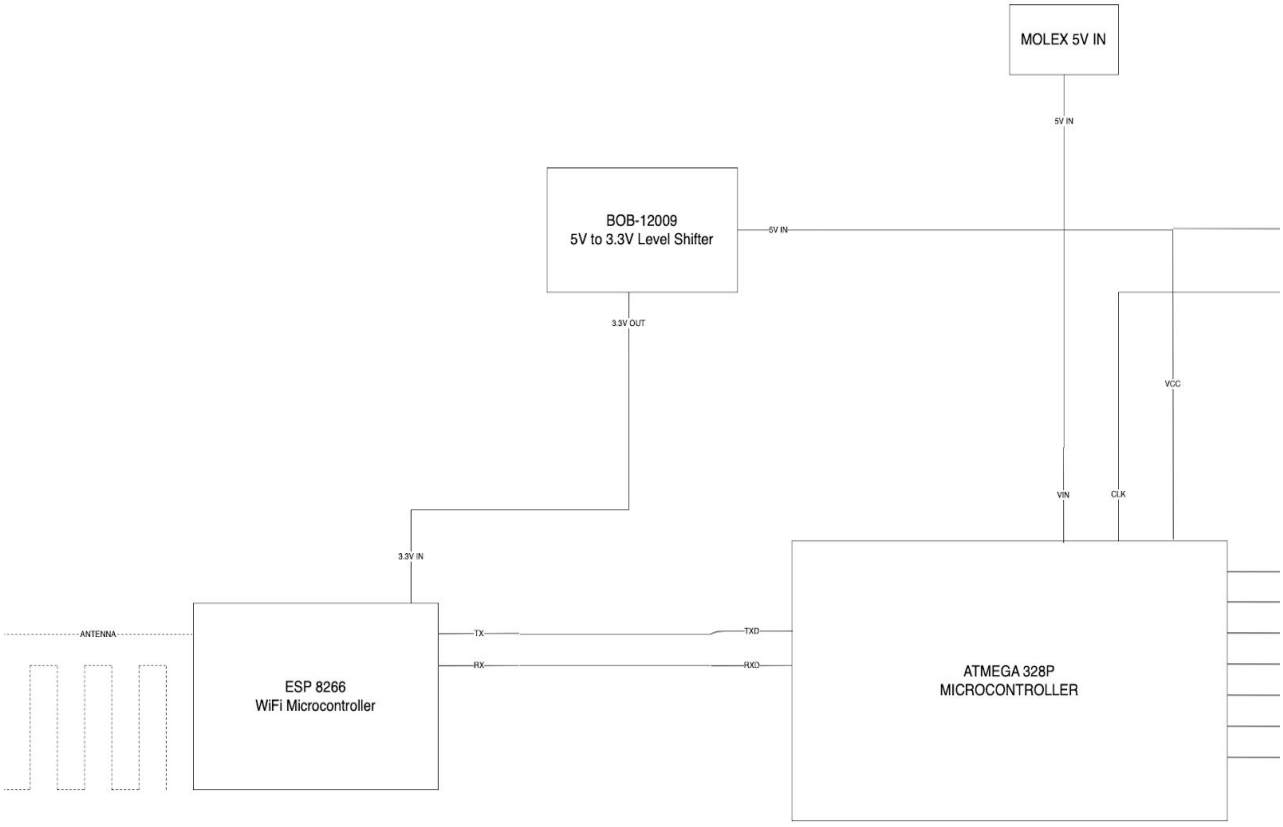

Figure 5: Schematic-A

Below in Figure: Schematic-B we consider the portion of the schematic that deals with communication between the ATmega328P microcontroller on the left side through Data Pin 1-7, and the HX711 Amplifier Board and the TAL221/220B Load Cells. We connect the TAL221/220B load cells through the amplifier board to boost the signal to where the ATmega328P can read the values from the cells accurately. We must also supply a clock signal to the amplifier boards to enable synchronous communication between itself and the ATmega328P.

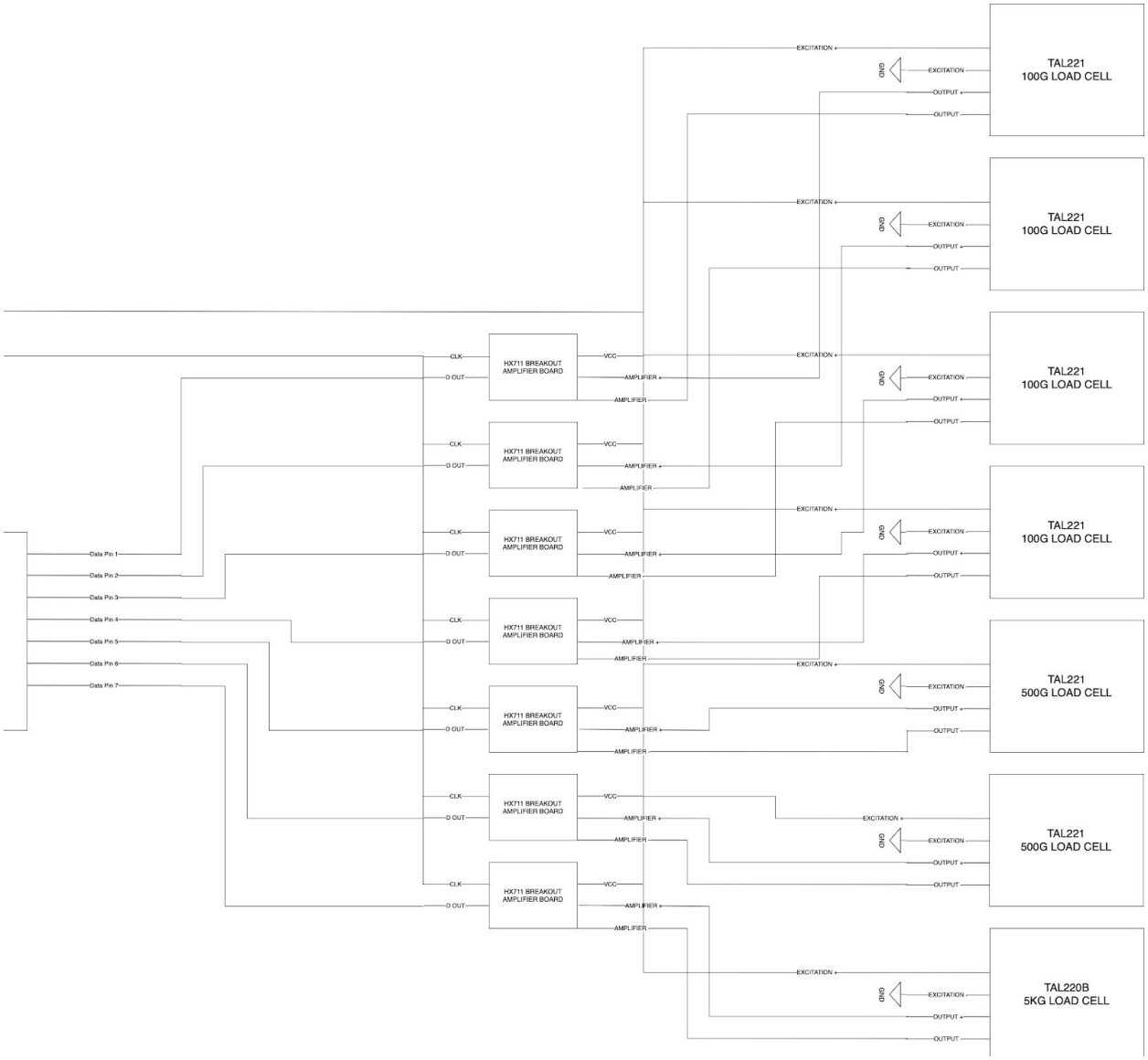

Figure 6: Schematic-B

Module	Requirements	Verification
Physical Tray: 4x TAL221 100g Load Sensors, 2x TAL221 500g Load Sensors, 1x TAL220B 5kg Load Sensor	1. Each sensor must be mounted in a structurally stable manner such that they can hold plastic containers, a mounted platform, and 100g/500g/5kg of an ingredient. 2. Sensors must provide stable readings within 90% accuracy.	1. a. Mount a sensor with a platform onto the tray base such that the platform is parallel to the base. b. Place maximum weight rated for each sensor on their respective platforms (reword) c. Remove weight, inspect and make platform level upon removal. 2. a. i. Place 50g of dry ingredients on platforms supported by TAL221 100g sensors. ii. Place 250g of dry ingredients on platforms supported by TAL221 500g sensors. iii. Place 1kg of dry ingredients on the platform supported by the TAL220B 5kg sensor. b. Confirm that weight readings from each sensor are within 90% accuracy by printing them to the microprocessor's serial monitor.
Physical Tray: 7x HX711 Breakout Boards for Load Sensors	1. Breakout boards must be able to interface with I/O ports on the data collection microcontroller.	1. a. Connect load sensors wires to breakout boards and connect DAT (data) and SCK (slave clock) wires to GPIOs of microprocessor. b. Connect V_{CC} and V_{DD} outputs on microprocessor to corresponding pins on breakout board. c. Confirm that a floating point value outputs to the microprocessor's serial monitor for each breakout board.

2.1.3 PCB:

Our PCB will consist of 3 main entities: I/O ports, the microcontroller (ATmega328P), and the Wi-Fi adapter (ESP8266). The seven I/O ports, one for each sensor, will connect the data path of the sensors to the microcontroller. One I/O port will be used for a common input clock for all sensor breakout boards. The microcontroller will handle processing the weights in real time, sensor calibration,

and any other circuit controls necessary. If the microcontroller detects a significant weight change, it will send a HTTP message via the Wi-Fi adapter (which is sending and receiving at 2.4 GHz) to our remote database. This ensures that the board and the user can communicate via any common Internet connection.

Module	**Requirements**	**Verification**
PCB: Microprocessor (ATmega328P)	1. Must have V_{cc} output from microcontroller be 5V ± 0.3V. 2. Must be able to send digital sensor readings to Wi-Fi Microchip via SPI (serial port).	1. a. Use a DMM to probe the output of V_{cc} and confirm that V_{cc} is within acceptable range. 2. a. Send test data sequence of 7 arbitrary floating point values to the Wi-Fi microchip from the microprocessor, ensure data is not changed.
PCB: Wi-Fi Microchip (ESP8266)	1. Latency of transmission of sensor data array to MySQL DB must be no greater than 30 seconds. 2. Must store API Key (approx. 100 characters) and volatile weight values (7 floats) in microchip's flash memory.	1. a. Place an ingredient of known weight on a load sensor. b. Confirm stable reading value in serial terminal, begin timing. c. Confirm that the value read in a terminal appears in the MySQL DB within 30 seconds. 2. a. Print API Key and weight change values to serial monitor specifically for ESP8266.

2.1.4 Linux Server:

The server will be a droplet operating a Linux environment that runs the MySQL database and the SpringBoot Java API. The droplet will be hosted on DigitalOcean with a 1 GB RAM and 25 GB of SSD memory. The choice for DigitalOcean is based on the designer's belief in open-source software and the low-cost. Further, the web hosting service is easily scalable and, if we were to take our prototype to production, our server needs would be easily met.

To make the server secure and follow HTTPS protocol, we decided it would be necessary to use and purchase a domain address. The domain name itself is trivial as it will be used solely for machine-to-machine interaction. After checking availability and looking ahead to the future, we purchased thomas-driscoll.com. The domain and the droplet will be connected via a SSL certificate.

Module	Requirement	Verification
Server	1. Domain name thomas-driscoll.com is pingable from any remote address 2. An admin can SSH into a command line	1. From any remote address, the command 'ping thomas-driscoll.com' should return an acknowledgement 2. From command line (or Git bash), a user with appropriate credentials can SSH into terminal. Simple command of 'ls' is prove enough of access

2.1.5 MySQL Database:

The MySQL database will be used to store the weights of the tray as they change and store any user data. As the weight changes on the board, the database will be updated to reflect the new associated weight for each sensor. When the user makes a request for the current weight, the database will interact with the middleware (written in SpringBoot Java) to send the weights and other pertinent information to the user.

We decided that our database only required three tables: User, Weights, and User_Definitions. The User table allows for multiple users to exist in our system, thus ensuring scalability. The username and API_key (stored on the board and a unique identifier for the board) uniquely each user/board pairing. This insures scalability as multiple users and can exist and each user can have multiple devices with a unique set-up for each. The Weights table holds weights received from the PCB (uniquely identified by the API_key) with an associated timestamp on when it was received. Finally, the User_Definitions table holds the threshold weights the user chooses to set for each pad. This was included as future product owners may want to measure user preferences for better quality control in future products.

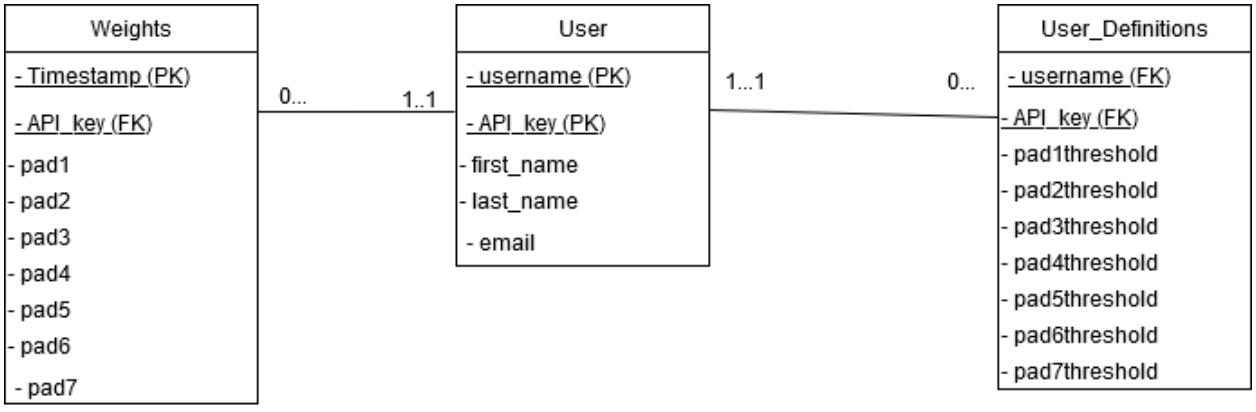

Figure 7: UML Diagram showing relationships between MySQL tables

Module	Requirements	Verification
Server: MySQL DB	1. Database supports a table for each sensor 2. Database only accepts input from a recognized user 3. Database inserts data correctly from .json file with timestamp	1. From command line, run the command to run a SQL shell. In that shell, the command 'SHOW TABLES' should return all relevant tables 2. Develop and run two unit tests. One for a user with proper authentication (i.e. the API key associated with the board) and one without. The verified user case should return an acknowledgement and the unverified user should return a failure. 3. Develop and run a unit test with some data (doesn't have to change). This should send the data (with a timestamp as its key) into a test table. This will verify that the data has been inserted at a particular time successfully

2.1.6 SpringBoot Java API:

The SpringBoot Java API will handle incoming requests to post or get data from the database, authenticating the requests to ensure data security. The middleware will operate by sending and receiving GET/POST operations following standard HTTP methods. There are four functions we have determined necessary for the operation of our back-end.

The first is an authentication function that checks if the request is from a valid source and formatted correctly. If it is not, the API returns the appropriate error as indicated in the flowchart below. Otherwise, the function parses whether the request is from the user or from the PCB. If the request is from the PCB, it calls the POST function handler /weights. This inserts the new weight info into the database and returns a HTTP 200 code, indicating a successful operation.

If the request is from the user, there are two possible functions: /all or /thresholds. The /all function occurs in response to a GET request and returns the latest weights in the Weight table as well as the last threshold weights provided by the user. The /thresholds function handles POST requests from the user and inserts or updates the user generated threshold weights. It returns a HTTP 200 code, indicating a successful operation.

Figure 8: Flow Diagram for SpringBoot Java API

Module	Requirements	Verifications
Server: SpringBoot Java Middleware	1. Accepts GET request for latest weight measurement and returns an HTTP 200 header with JSON data to user-facing application 2. Accepts POST request for latest weight measurement and returns an HTTP 200 ACK to Wi-Fi chip on tray 3. 4. Checks that request is from a valid point (i.e. API key or user) 5. >80% unit test coverage for every HTTP method	0. To make sure each function works independently of any internet access, write a mock database with hard-coded data. 1. Write unit tests for GET request that returns mock data in JSON format with 200 header. Return 400 error otherwise. 2. Write unit tests for POST request that returns HTTP 200 ACK. Return 400 error otherwise. 3. Write unit test for POST request that returns HTTP 200 ACK. Return 400 error otherwise. 4. Write unit test to mock API_keys. Write another unit test to test with a valid user. Return 200 ACK on either case, 400 error otherwise. 5. We define 80% unit tests as covering all possible paths where a function could go that we can think of. We say 80% as it is impossible to consider every possible case.

2.1.7 User Phone:

The user phone block indicates an app allowing users to remotely view the weights on their Weightb0ard at any given time. In order to interact with the board, each user will load an app onto their phone. Upon opening the app, the user's phone will automatically make a request to the Linux server, where our middleware will direct the appropriate operations to perform. Our plan is to write this front-end in React Native to allow cross-platform support and fast development time.

Figure 9: Minimal mock-up for the front-end

Module	**Requirements**	**Verification**
Phone: React Native Front-End	1. Users should be able to view weight information from all sensors in readable format 2. Application should successfully show user if any readings are below user-defined thresholds for each ingredient by changing label to red 3. Application allows customizable labels for each sensor	0. Write a mock API response(s) that will be used for each verification test 1. Write unit test to mock API response and JavaScript weight variables received it 2. Write unit test with mock API response for weights below user-threshold. Successfully demonstrate threshold variables CSS style changes to red

	4. >80% unit test coverage for every React component	3. Write unit test for a mock user that returns customized labels. 4. We define 80% unit tests as covering all possible paths where a function could go that we can think of. We say 80% as it is impossible to consider every possible case.

2.1.7 Other Design Considerations

We considered adding LEDs to the board as a way to tell at a glance how much you have of an ingredient, or as a way to know when to get more. We decided against this however, as when in the physical presence of the board it is easy to pick up the ingredient off of it and judge whether or not to get more. This also does not contradict our critique of the Samsung Smart Fridge, as cameras are still unable to look into non-clear containers to judge remaining ingredients. Further, if precise measurements are required, the user would have the app handy anyway.

2.2 Physical Design

Fig. 2 is what our physical design is going to look like for this project from the top down view. Our choices are such that each progressively larger sized sensor has a larger footprint on the board for measuring ingredients. This choice was made so that our board had space to accommodate the larger, heavier ingredients on the large pressure sensors, and the smaller, less massive ones weren't given an unnecessarily large space on the board. The 4 smallest sensors will be the same size, as will the 2 medium-sized sensors. Each load cell (weight sensor) will be mounted from the edge of the pad, with a platform centered at the other end of the pad, such that the rubber center of the cell can bend to produce sensor outputs.

Fig. 3 is what our physical design is going to look like for this project from the side view perspective. We also wanted to design the board such that there was a moderately sized cavity underneath to house the PCB and wires that came off of the pressure sensors.

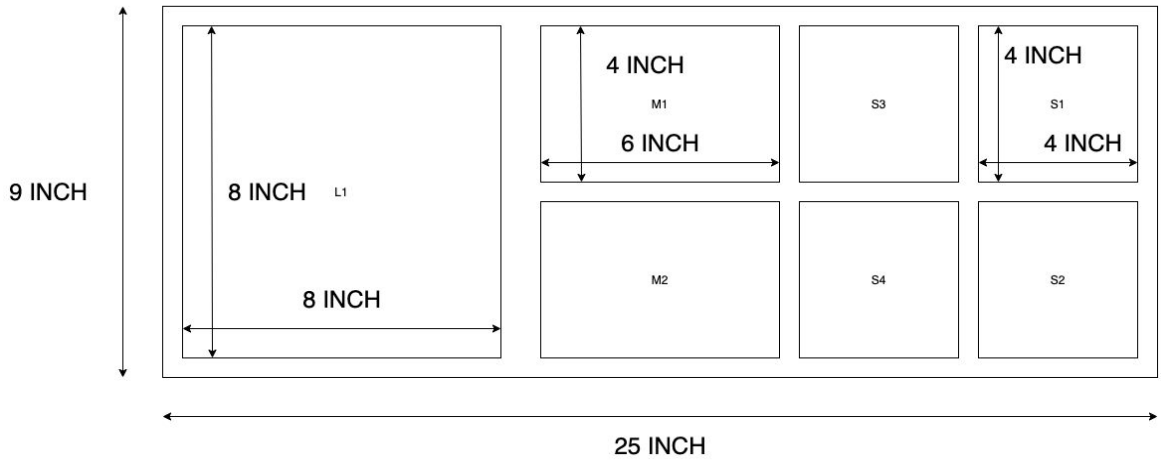

Figure 10: Physical Diagram (Top Down View)

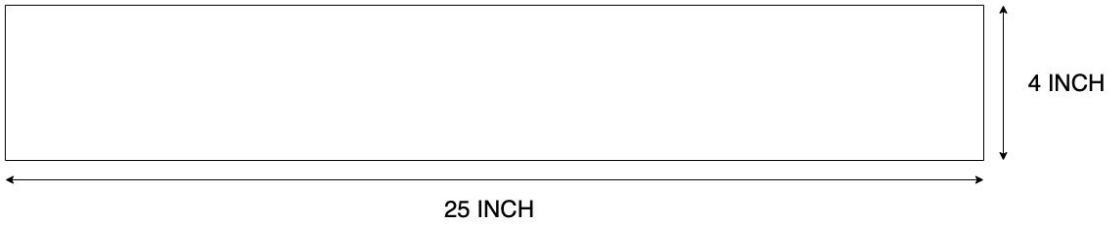

Figure 11: Physical Diagram (Side View)

2.3 Tolerance Analysis

The critical block that poses the most challenging requirement is the physical tray's sensor array. The type of sensor we're using is susceptible to small amounts of creep, an inevitable issue on the molecular level [6] where stable readings change slowly over time, specifically at a rate of ±0.05% every 3 minutes for the TAL221 load cell [7] (the S and M cells), and ±0.1% every 3 minutes for the TAL220B [8] [load cell (the L cell). In order to provide readings within an error of 10%, we do not want to continuously have the sensor powered on for longer than 10 / 0.05 * 3 = 600 minutes for the TAL221, and 300 minutes from the TAL220B. A possible solution is to use a low-power mode for the HX711 breakout boards for when weights aren't modified for long periods of time between readings.

3 Cost and Schedule

3.1.1 Manpower Cost

The average salary of a 2017-2018 ECE Illinois Computer Engineering Grad (as our group is comprised of) was $92,430 [9]. Working 52, 40 hour weeks (for a total of 2,080 hours a year), this breaks down to $44.43 per hour. We will use this as our fixed hourly cost for our project. We assume working 10 hours per week, and for the remaining ~10 weeks for this class. This formula also neglects any time working with product marketing or external partnerships. We therefore calculate our manpower cost for this project as follows:

$$2.5 \times Number\ of\ Group\ Members \times Fixed\ Hourly\ Cost \times \#\ of\ Hours\ per\ Week \times \#\ of\ Weeks = Cost$$

$$2.5 \times 3 \times \$44.43 \times 10 \times 10 = \$33,322.50$$

As shown above, we calculate the manpower cost to be $33,322.50 for this prototype.

3.1.2 Part Cost

Below we have a list of all of the parts required to make our board, broken down into bulk and prototype pricing.

Part	Cost (bulk)	Cost (prototype)
5x TAL221 100g Load Sensors	$2.80 * 5 = $14	$8.95 * 5 = $44.75
3x TAL221 500g Load Sensors	$2.50 * 3 = $7.50	$9.95 * 3 = $29.85
2x TAL220B 5kg Load Sensor	$1.80 * 2 = $3.60	$10.95 * 2 = $21.90
1x ESP-8266 Wi-Fi Microchip	$1.40 * 1 = $1.40	$6.95 * 1 = $6.95
8x HX711 Breakout Boards	$1.47 * 8 = $11.76	$9.95 * 8 = $79.60
1x ATMega328P Microprocessor	$3.16 * 1 = $3.16	$4.30 * 1 = $4.30
1x TOL-15664 5V 2A AC/DC Wall Adapter	$3.00 * 1 = $3.00	$10.95 * 1 = $10.95
1x Domain Name for Web Hosting	$15.66/2 years	$31.32/2 years
1x Remote Server Droplet	$60/year	$60/year
1x BOB-12009 5v to 3.3v Level Shifter	$0.65 * 1 = $0.65	$2.95 * 1 = $2.95
Misc. Construction Costs (Wood, surface finishing, screws, etc.)	$10.00	$20.00

Final Part Cost	$130.73	$312.57

As shown above, we calculate the bulk cost for our board's parts to be $70.73 and the prototype cost to be $252.57.

3.1.3 Shop Cost

We estimate the shop hourly rate to be $30 per hour. We have a relatively simple design, requiring only a frame to affix the sensors to along with a storage space on the underside for cabling. Due to this, we estimate our total shop hour requirement to be 20 hours. Due to this, we say the shop labor cost to be:

$$Total\ Hours\ \times Fixed\ Hourly\ Cost\ = 20\ x\ \$30\ =\ \$600$$

3.1.4 Total Cost

We calculate the total cost including manpower, shop labor, and parts to be $33993.20 accounting for bulk parts and $34175.10 for the prototype.

3.2 Schedule of Work

Week of	Kyle	August	Thomas
2/23	Finish Design Doc	Finish Design Doc, Design PCB after approved	Finish Design Doc
3/1	Complete Soldering Assignment	Engage with ECE Shop, Complete soldering assignment	Set up server and connect to domain name (must be pingable)
3/8	Finalize PCB design for Early Bird order	Place order for PCBWay	Set up SpringBoot Java API and test connection using Arduino Uno
3/15	Program ATMEGA and WIFI chips	Program ATMEGA and WIFI chips on personal testing board	Write unit tests for API
3/22	Work on individual progress report, test PCBs	Work on individual progress report, test completed PCBs	Work on individual progress report, continue front-end

			dev
3/29	Work on integration between board and mobile application	Work on integration between board and mobile application	Work with August on board/mobile app integration
4/5	Continue bug-fixing integration locally	Continue bug-fixing integration locally	Write unit tests for front-end dev
4/12	Prepare for Mock Demo	Prepare for Mock Demo	Prepare for Mock Demo
4/19	Do Mock Demo, final tweaks for boards	Do Mock Demo, any final tweaks for board	Do Mock Demo, any final tweaks for server/API/front-end
4/26	Work on final design paperwork	Work on final design paperwork	Work on final design paperwork
5/3	Turn in final design paperwork and do final presentation	Turn in final design paperwork and do final presentation	Turn in final design paperwork and do final presentation

4 Ethics & Safety

The ethical or safety issues with our project pertain to the physical tray itself, and the microcontroller and Wi-Fi chips.

Citing the IEEE Code of Ethics #9 [10] - to avoid injuring others, their property, reputation, or employment by false or malicious action, we will work to ensure that the construction of our tray is structurally sound such that a user will not be concerned with electrical hazards such as exposed wires or static shock, or any harm from burning ICs or plastic. We will also make considerations to prevent damage to the tray's main circuitry by contact with user ingredients. (These considerations would take the form of a protective layer on the top of the board that prevents any spillage into the sensitive electronics underneath)

Citing the ACM Code of Ethics 2.9 [11], the greatest source of ethical and security concerns is the database itself. We will be allowing multiple users the ability to store or request potentially sensitive information, specifically an email address. This could lead to bad actors stealing this data and targeting users [12]. To avoid these concerns, we will be hosting our server on DigitalOceans, which comes with its

own security measures to prevent bad actors. Further, we will have our own authentication measures to do our best to prevent hacking.

An additional source of safety concern is the user-facing application, specifically in regards to the ACM Code of Ethics 2.9 [11]. While we expect the user of our prototype to load the application from source code provided by the designers, bad actors could potentially hijack API calls in the app itself to download malware onto a user's phone [13]. These concerns, while possible, are an extremely low risk as our application will not be downloaded outside of the authors knowledge for the duration of the project. Further, we will adopt a one-origin policy to authenticate the requests, since our HTTP protocols will only be handled by our one server.

References

[1] Schanes, K., Dobernig, K. and Gözet, B. (2018). *Food waste matters - A systematic review of household food waste practices and their policy implications*. [online] Science Direct. Available at: https://www.sciencedirect.com/science/article/pii/S0959652618303366 [Accessed 12 Feb. 2020].

[2] "Samsung Smart Refrigerator: Family Hub Touchscreen Fridge Overview," Samsung Electronics America. [Online]. Available: https://www.samsung.com/us/explore/family-hub-refrigerator/overview/. [Accessed: 28-Feb-2020].

[3] "Wheatstone Bridge," *Sparkfun*. [Online]. Available: https://cdn.sparkfun.com/assets/learn_tutorials/3/8/3/Wheatstone-Bridge-02.jpg. [Accessed: 25-Feb-2020].

[4] Avia Semiconductor Ltd. "24-Bit Analog-to-Digital Converter (ADC) for Weigh Scales" HX711 datasheet, 2012, v2.0. Available: https://cdn.sparkfun.com/assets/b/f/5/a/e/hx711F_EN.pdf [Accessed: 13-Feb-2020]

[5] N. Seidle and A. Wende, "SparkFun_HX711_Load_Cell.pdf." Boulder, CO, 15-May-2019. [Accessed: 02-Mar-2020].

[6] S. Fiere, "What is creep and how to deal with it?," *Zemic Europe*, 09-Oct-2018. [Accessed: 02-Mar-2020].

[7] HT Sensor Technology Co. Ltd. "TAL221 Miniature Load Cell" TAL221 datasheet, (n.d.) Available: https://cdn.sparkfun.com/assets/9/9/a/f/3/TAL221.pdf [Accessed: 13-Feb-2020]

[8] HT Sensor Technology Co. Ltd. "TAL220B Miniature Load Cell" TAL220B datasheet, (n.d.) Available: https://cdn.sparkfun.com/assets/e/5/f/5/6/TAL220B.pdf [Accessed: 13-Feb-2020]

[9] "All Campus Undergrad 2017-2018 Report," *Engineering Career Services*, 31-Dec-2018. [Online]. Available: https://ecs.engineering.illinois.edu/files/2019/03/IlliniSuccess_AnnualReport_2017-2018_FINAL.pdf. [Accessed: 25-Feb-2020].

[10] ieee.org, "IEEE IEEE Code of Ethics", 2016. [Online]. Available: http://www.ieee.org/about/corporate/governance/p7-8.html. [Accessed: 13-Feb-2020].

[11] Acm.org. (2020). *The Code affirms an obligation of computing professionals to use their skills for the benefit of society.*. [online] Available at: https://www.acm.org/code-of-ethics [Accessed 13 Feb. 2020].

[12] Digital Guardian. (2020). *The History of Data Breaches*. [online] Available at: https://digitalguardian.com/blog/history-data-breaches [Accessed 14 Feb. 2020].

[13] Chicago Tribune. (2020). *Chicago Tribune - Widespread Android Virus Could Hide in Popular App Updates*. [online] Available at: https://www.chicagotribune.com/news/ct-xpm-2012-04-16-sns-201204161154usnewsusnwr20120413androidapr16-story.html [Accessed 13 Feb. 2020].