Espresso Overflow

Authors: Archana Navada, Stefan Orton-Urbina, Harper Weigle: Electrical and Computer Engineering, Carnegie Mellon University

***Abstract*—The taste of espresso is dependent on well known variables: pressure, heat, and surface area. Pressure and heat are fixed by a high quality espresso machine, surface area is tuned by a grinder. The taste can then be quantified by looking at the weight vs time of the pull. This project provides that information with a scale and iOS or Android application that records the relevant information and provides insight on how to improve.**

***Index Terms*—Bluetooth, Espresso, Weight scale, IoT, iOS, Android, Arduino, C++, Dart, Firebase, 3D-Printing, SPI, I^2C**

1 INTRODUCTION

Espresso is as much a science as any other process that comes before a niche but specialized tasting experience. In this process, the tuning variables are pressure, heat, and surface area. Ideally, pressure and heat can be held constant by a high quality espresso machine and consistent "tamping" of the ground coffee. The surface area is dependent on the fineness of the grind and can be tuned with an espresso grinder. However, the ability to finely tune variables in a system is only as helpful as the bench-marking of the output. While taste is subjective, the taste of espresso can be quantified by the weight of the espresso and how long it took to extract it from the coffee grounds. The beginning of the "pull" is very bitter and acidic while the end of the pull is more mellowed but less flavorful. The combination of these with even flow through the coffee grounds results in a sweet and generally deemed desirable shot of espresso. This project looks to provide a quantitative solution to the systems bench-marking on par with the specificity of the tuning variables.

2 DESIGN REQUIREMENTS

2.1 User Requirements

The usage should be simple but the simplicity should not inhibit the user from having full control over the process. For that reason the device must maintain functionality as a typical coffee scale in that it:

- Accurately measures and displays object weights with a resolution of .01g. This will happen on the LCD screen as well as on the weight display on the application side.
- Can functionally tare or set the zero point to the weight currently applied to the scale. This will happen on the push of a button and within 1s of the button press. The updated weight of 0g will be reflected on the LCD.
- The scale should be capable of measuring weights up to 200g and down to 0g without loss of resolution.
- The device must be able to be powered down to conserve power.
- The device must be able to work for repeated uses back to back for the use case that it is being used to serve multiple people espresso.

2.2 Physical Requirements

The scale must comply with the physical constraints of an espresso machine as it will measure the weight of the espresso while it is being poured. Given that a typical espresso machine has 4in of room from the portafilter to the drip tray and a typical 4oz glass standing around 2in must fit between the scale and the portafilter, the scale must be under 2in in height. To fit on a standard drip tray the device must have a footprint of less than 5in x 8in. Lastly, the device must be easily transferable from the espresso machine to a nearby table presumably with one hand. For the average human, a limit of 5lbs was set but the device is expected to be far under that threshold.

2.3 Software Requirements

The application looks to have a live feed of the weight on the scale. While it is not imperative that it happens in sync with the pull because the user will be operating the machine independent of the application, the data should be available as soon as the user goes to look for it. So the device will have to transmit the data as it is collected. Additionally, the duration of the pull can be as short as 10s and as long as 60s. For helpful data, the application is going to need at least 100 samples. Over the minimum duration, the software must be transmitting an updated value at 10Hz at the slowest.

2.4 Application Requirements

Seeing as the application will often be used at the earliest hours of the morning or at least when caffeine is desired by the user, the application will have to be user friendly and easily navigable. It should have no more than 3 steps to get from opening the app for the first time to being able

to retrieve the data informing the tuning variables for the process.

3 ARCHITECTURE OVERVIEW

The system can be described in two major components as illustrated in Fig. 1. The dark brown section described the hardware which will sit in the espresso machine, interact with the user during the process, and transmit data to the application end. The application end is detailed by the tan section. It will collect data from the hardware during the process and provide data to the user about the quality of the espresso as well as how it can be improved by adjusting the tuning variables.

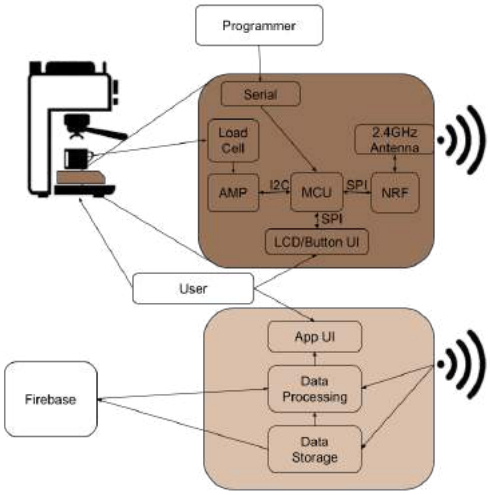


Figure 1: Architecture Overview Block Diagram

3.1 Hardware

The MCU described in Fig. 1 is an ATmega328P: the MCU included on an Arduino UNO. It will communicate with a NAU7802 load cell amplifier through I^2C communication protocol. The load cell amplifier will in turn communicate with a load cell that was removed from a NEXTSHINE kitchen scale and placed into the physical enclosure. The MCU will also communicate with the LCD display through I^2C communication protocol and the button interface through dedicated GPIO pins on the MCU. On the SPI bus is the NRF24L01 2.4GHz transmitter. The MCU will transmit weight data from the load cell to the transmitter so that it can be received by the application.

Figure 2: Full physical device in use

3.2 Application

The application is uses Firebase as a web server and keeps the data of previous espresso pulls as well as user uploads such as bean brand, pictures of the resulting espresso, and user ratings as well as the applications recommendations on improvements. It will communicate with the ATmega by reading the manufacture data on the paired Bluetooth device. The pairing will occur using the built in Bluetooth functionality on a modern Android or iPhone. This manufacturer data will contain the scale readings and update the graph in the application user interface accordingly. The application also provides users with the ability to upload custom logs.

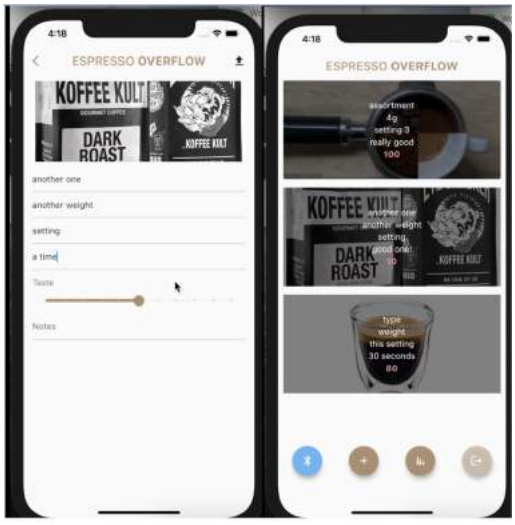


Figure 3: Sample of Application User Interface

4 DESIGN TRADE STUDIES

4.1 Hardware

4.1.1 MCU

The MCU had to be capable of communicating with the LCD and NRF24L01 meaning it had to be equipped with a SPI bus. Additionally, it required a I^2C bus for communication with the load cell amplifier and a few GPIO ports for use with the button interface. While lengthy, this list of requirements does not narrow down the list of MCUs very much. Originally, the MCU was going to be a STM32F423CHU6 mounted on a custom PCB. However, due to supply shortages in China, the MCU was unable to be acquired in time and the risk mitigation MCU (ATmega328P) was used instead. The trade off that lead to the choice of the STM32F423CHU6 rather than the ATmega328P was the compactness that the custom PCB offered. Such compactness would allow the device to better address the size requirements detailed in Sec. 2.2. Pivoting to the risk mitigation plan forced more pedantic space allocation of the individual evaluation boards for each component but the design requirements were still met. The ATmega328P was selected as a backup MCU because of its versatility in that it was able to function with each peripheral component selected and did not compromise the vertical constraints detailed in Sec. 2.2. It also is equipped with a 16MHz internal clock which is divided by 4 for the SPI clock which will be used to transmit data. This provides a data rate with plenty of room to meet the 100 samples per second data rate described in Sec. 2.3.

4.1.2 Load Cell

The load cell specifications are strict given the size constraints detailed in Sec. 2.2 and the weight resolution required to meet the design requirements detailed in Sec. 2.1 all the while staying within the price range. It was determined that buying a load cell itself capable of measuring 0g-200g at a resolution of .01g would quickly take the project out of the scope of the budget. As an alternative, a NEXT-SHINE kitchen scale was purchased and disassembled to retrieve the 500g load cell with a resolution of .01g. Because of the load cell amplifier (described in Sec. 4.1.3), the load cell could be easily cut, striped, and plugged into the terminal blocks to work for the projects purposes.

4.1.3 Load Cell Amplifier

Firstly, the load cell amplifier was included in the design to compartmentalize the load cell voltage output away from the MCU. This reduces the need to occupy the internal ADC in the MCU in case it was needed for additional features but it also isolates the load cell output from digital lines that could create unwanted noise on the low voltage line. The NAU7802 was selected for it's I^2C compatibility and it's internal wheatstone bridge removing the need for external passive components both increasing the devices footprint but also it's susceptibility to failure.

4.1.4 2.4GHz Transmitter

The transmitter was added to the design to remove the need for wired communication between the device and a mobile phone. The NRF24L01 was selected for its low price point and its open source nature. The device could easily be appended to the SPI bus and was known to transmit the frequency needed to be detected by modern mobile phones. The project could have included a dedicated proprietary BTE transmitter, however, this peripheral would have consumed a substantial portion of the budget for little benefit over the NRF24L01 outside of ease of setup for the engineers. The user should notice no difference between the two use cases because Firebase (described in Sec. 3.2) does not include native bluetooth functionality. This requires the user to pair with the bluetooth device outside of the Espresso Overflow app to begin with.

4.1.5 LCD

The LCD serves a critical purpose on the device as it has to relay information to the user that will ultimately guide the user interaction. The LCD chosen was the SparkFun LCD-16396. It connects with other selected components well because of its I^2C connection and has software infrastructure already written to allow for seamless intergration with the other components.

4.1.6 Enclosure

Due to our tight sizing constraints described in Sec. 2.2, the enclosure created had to fit the chosen components very closely which ruled out any pre-fabricated enclosures. Additionally, the access to raw materials was limited but 3D-printer access was virtually unlimited. As a result, a 3D-printed enclosure was selected to allow for maximum accessibility and minimum footprint. This choice also lowered the cost of the enclosure because all that would have to be paid for is the printer filament.

4.2 Software

4.2.1 Application

The application was written in Dart with a Firebase backend. The primary driving factor behind this design choice was the need to function on both Android and iOS. This development toolchain allows for a much smaller and more compact exportable application that makes the device usage quick for all users without the need for any dependencies.

4.2.2 Embedded Software

The software for the MCU was written in C++ and compiled and downloaded using the Arduino IDE through

Serial. The primary benefit to the Arduino IDE is the software infrastructure for each of the peripheral components which was necessary given the time constraints after the project pivot. Furthermore, the Arduino IDE allowed the group to forgo a writing or finding a bootloader for the ATmega328P.

5 SYSTEM DESCRIPTION

The system as a whole is controlled by two main hardware systems, the MCU (Sec. 5.1 and the users mobile device. The MCU communicates with the LCD (Sec. 5.5 and the load cell amplifier (Sec. 5.3) through its I^2C bus and the 2.4GHz transmitter (Sec. 5.4 via its SPI bus. The button interface (Sec. 5.6 communicates with the MCU through its GPIO pins. The mobile phone houses the application software and queries the Firebase database for all information transfer. The scale information is received the the bluetooth manufacturer data transmitted by the 2.4GHz transmitter on the scale.

5.1 MCU

The ATmega328P[2], in the overall system, is tasked with I^2C communication to control the LCD display and the load cell amplifier as well as SPI communication to control the 2.4GHz transmitter. The I^2C line, by the nature of I^2C clock stretching, has no defined clock speed but the average I^2C clock for an Arduino and SparkFun peripherals during standard operation is around 100KHz. Additionally, the 16MHz on board clock is divided by 4 for the ATmega328P's internal SPI clock resulting in a 4MHz clock. As shown in Lst. 1, the MCU writes 9 bytes and reads 3 bytes over I^2C, writes 6 bytes over SPI, and writes 7 bytes over Serial to the debugger. The ATmega328P's serial was initialized to a baud rate of 9600. The I^2C reads and 1 write occurs in the getReading() function. Understanding that the weight is a 32bit integer that must be transmitted over the SPI bus, the equation in Fig. 4 yields a weight update from the scale every $.36\mu s$.

$$DataRate(\frac{bytes}{s}) = 32(bits) \times \frac{(SCLK_{SPI}(\frac{bits}{s})}{N_{SPI}(bits)} + \frac{SCLK_{I^2C}(\frac{bits}{s})}{N_{I^2C}(bits)} + \frac{BaudRate_{Serial}(\frac{bits}{s})}{N_{Serial}(bits)}) \quad (1)$$

Figure 4: Data rate calculation

5.2 Load Cell

The load cell was taken from NEXT-SHINE kitchen scale and unfortunately did not come with any part numbers so the actual verification of the load cell capability had to be done after integration with the device. However, the NEXT-SHINE scale was found to operate within the advertised conditions. The load cell, shown in Fig. 5, fit well into the model because of its small size. Its dimensions (LxWxH) were found to be 47mmx10mmx6mm. The load cell was determined to function like a Wheatstone bridge based on the V+, S+, S-, and V- connections in the scale as shown in Fig. 6. This allowed us to splice the connection and plug it into the load cell amplifier described in Sec. 5.3.

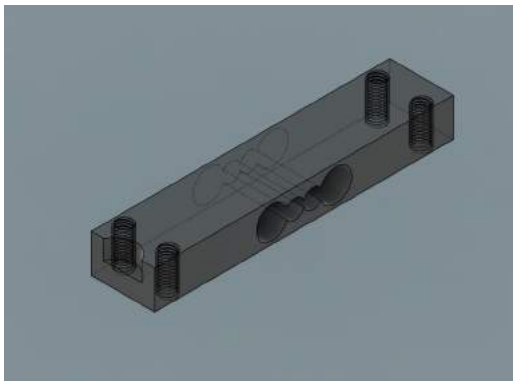


Figure 5: 3D model of load cell

Figure 6: Load Cell Connection to NEXT-SHINE kitchen scale

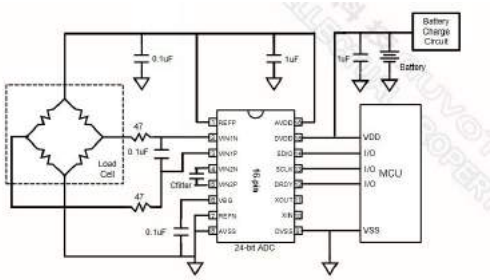


Figure 7: NAU7802 typical application schematic

5.3 Load Cell Amplifier

The load cell amplifier communicates through I^2C with the MCU as described in Sec. 5.1. Due to the usage of a SparkFun evaluation board for the NAU7802[1], a library[4] was able to be used for the driver code. The NAU7802

model takes in a voltage differential from a Wheatstone bridge connection detailed in Fig. 7. It outputs data from its internal ADC in the form of thee 8-bit packages.

5.4 2.4GHz Transmitter

For Bluetooth transmission, the nRF24L01[3] was used. It broadcasts at 2.4GHz which is the frequency Bluetooth communication broadcasts at. In order to send the information to the application, the device encodes the weight information into the manufacturer data to get around proprietary Bluetooth communication protocols. Fig. 9 shows the scale broadcasting information and encoding 86g (0x56 in hexadecimal) in the manufacturer data. This value updates on every loop of the MCU describes in Sec. 5.1 and shown in Lst. 1.

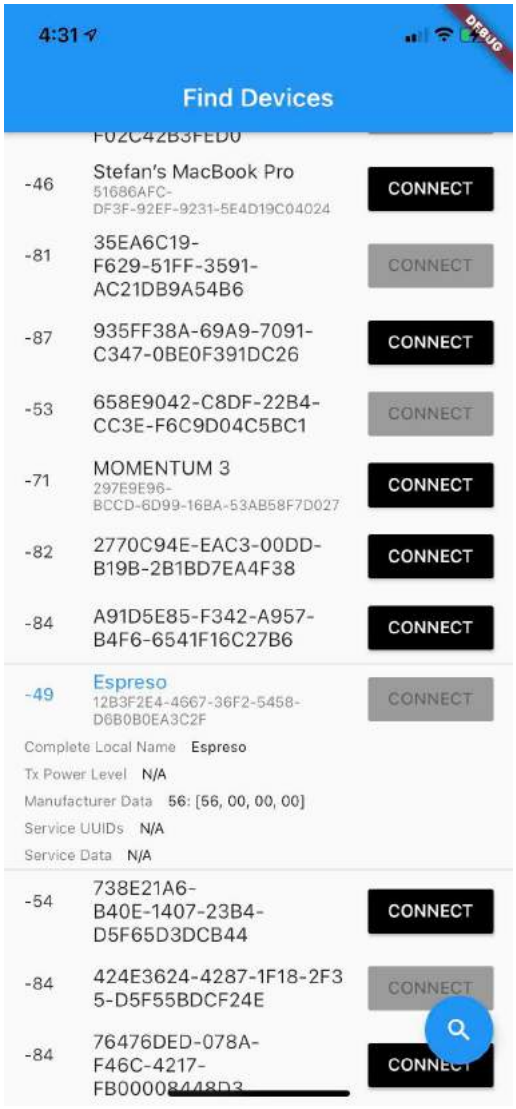


Figure 8: The broadcast device name and manufacturer data containing the current weight on the scale

5.5 LCD

The LCD is a SparkFun 16x2 SerLCD. It communicates with the MCU through I^2C as described in Sec. 5.1 and displays the current weight on the scale every loop in the MCU as shown in Fig. 2.

5.6 Buttons

The Buttons on the scale are responsible for starting the on board timer and zeroing or taring the scale. Both buttons are shown next to the LCD in Fig. 2. The buttons are simply Pressed-On/Release-Off buttons connected to GPIO pins on the MCU. The tare function simply takes 100 samples of the current weight on the scale and averages it. This value becomes the new baseline value used in the loop calculation shown in Lst. 1. The timer button starts a timer from the time of press and stops it on a subsequent press. Because the device perpetually broadcasts information as described in Sec. 5.4, the time kept by the scale is just for the user to maintain normal coffee scale functionality.

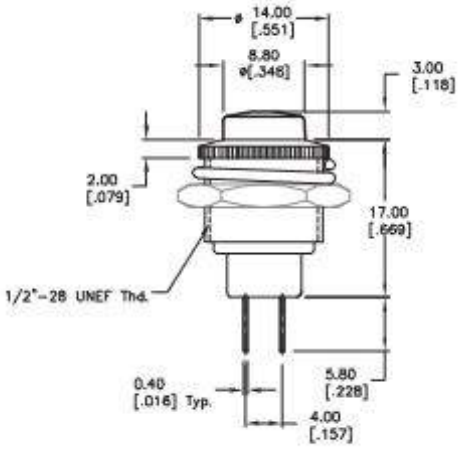


Figure 9: Mechanical drawing of scale buttons

6 TEST & VALIDATION

The team performed various tests on the speed of the application and the accuracy of weight measurements from the scale itself. The data was uploaded to storage and downloaded back to the dashboard consistently within five seconds, as was hoped at the beginning of the project.

6.1 Results for Weight Accuracy

The device quality was verified using a commercial scale of known accuracy to cross reference the weights of coffee grounds. It was found that the percent error was well below ten percent for various measurements. This data is displayed in 10 and 11.

Figure 10: Line graph displaying measured weight error

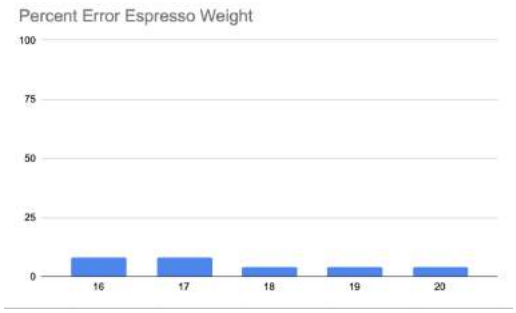


Figure 11: Bar graph displaying percent error in espresso weight

7 PROJECT MANAGEMENT

7.1 Schedule

The team's schedule changed drastically towards the end due to the PCB, which contained the entire system, not arriving. This forced the entire hardware side of the the system design to pivot to the risk mitigation plan and forced many of the tasks in the middle of the schedule to simply preparing for the late arrival of the PCB. Things like writing preliminary embedded code for the STM32 were completed but never fully realized because of the pivot to an available MCU.

The application development also suffered from not having available hardware because it forced integration between the application side and the hardware to be pushed until after the newly ordered evaluation boards for each of the peripherals had been integrated.

7.2 Team Member Responsibilities

Each team member came into the project with fairly similar skill sets but slightly specialized in one way or another. The divisions of labor ended up falling on the lines of front end software, embedded software, and hardware. This changed slightly towards then end when Stefan had all of the hardware in his possession and became responsible for hardware integration and testing the embedded software on the full hardware system. General responsibilities were as follows:

Archana - Front End Software

- Firebase implementation
- Flutter implementation and integration
- Application User Interface

Stefan - Embedded Software

- Embedded Software Integration
- LCD/Button User Interface
- Physical Enclosure Manufacturing and Testing
- Hardware Integration

Harper - Hardware

- System Electrical Design
- PCB Design
- Load Cell Implementation

7.3 Budget

The budget is described in Tab. 1.

7.4 Risk Management

The team's approach to the project was understood to be high risk from the beginning. Between the custom on-PCB antenna to embedded peripherals as opposed to full evaluation boards, the project had many failure points. For this reason, the risk mitigation plans for the project encompassed on PCB risk mitigation such as:

- 0Ω Signal Test Points
- 2.54mm UART Through Holes
- Terminal Blocks for the Buttons, the LCD, and the Load Cell

The design was also made with well known peripherals that had evaluation boards should the project need to pivot away from the PCB entirely. When the PCB was unable to arrive due to stock shortages in China, it was helpful to be able to buy SparkFun evalution boards for all the peripherals and control them all with an Arduino that was laying around.

8 ETHICAL ISSUES

Espresso making as a hobby and community is one with a high barrier to entry in that the cost of the tools is high and the learning curve is steep. While this project is unable to do anything about the high cost of entry, it looks to lower the time and effort required to surmount the learning curve. The obvious implication of such a goal is more people will become involved in espresso making thereby consuming more coffee.

Table 1: Bill of materials

Description	Model #	Manufacturer	Quantity	Cost @	Total
Populated PCB	SPRSO00	PCBWay	1	$255.00	$255.00
Load Cell Connector	440146-4	TE Connectivity	3	$0.17	$0.51
Load Cell Connector Crimps	1734193-1	TE Connectivity	12	$0.14	$1.68
JTAG Programmer	ST-LINK/V2	STMicroelectronics	2	$22.61	$45.22
LCD	LCD-16396	SparkFun	3	$19.95	$59.85
LCD Cable	4399	Adafruit	3	$0.95	$2.85
Batteries	CR2032	LiCB	1	$5.99	$5.99
Power Switch	PV2F240NNM01	E-Switch	3	$5.87	$17.61
Buttons	RP3502ABLK	E-Switch	10	$1.68	$16.80
3D Printer PLA Filament	HZST3DSILKPLA	SHENGTIAN	1	$29.99	$29.99
Coffee Scale	n/a	NEXT-SHINE	2	$16.99	$33.98
Load Cell Amplifier	SEN-15242	SparkFun	1	$14.95	$14.95
2.4GHz Transmitter	nRF24L01	Makerfire	1	$12.88	$12.88
MCU	STM32F423CHU6	STMicroelectronics	3	$12.64	$37.92
					$535.23

Herein lies the primary ethical concern with this project: coffee consumption. On a global scale, coffee consumption is incredibly detrimental both environmentally and economically for countries which grow it. Coffee, as it is currently grown requires a very specific climate common in South America, Northern Africa, and many countries in Oceania. These regions are also subject to exploitation by many industries, coffee included. The mass growth of coffee to meet the global demand results in mass deforestation and low wages for the workers who endure such back-breaking labor.

The solution is to encourage users of this device, but also all coffee products, to purchase ethically sourced coffee. Ethically sourced coffees range in their detriment mitigation techniques from coffees that do not require deforestation, like shade grown coffees, to simply giving portions of the profits back to the countries which grow the coffee.

9 RELATED WORK

There are few related products to the product the team created. The main one of note is the Acaia Pearl and Lunar scales. These weighing scales have similar functionality to the product created by the team. These products are the only ones of importance on the market that can fit under an espresso machine, which was a key requirement in this project. With the original design including the PCB, our team believes that the budget would have been comparable to the cost of the Acaia products.

10 SUMMARY

The final product was able to meet most of the design requirements that were set forth with the caveat that they occurred through the risk mitigation plan. The device functions as a normal coffee scale and can be used in place of any espresso scale one might find in the kitchen of an espresso enthusiast. Additionally, data is transferred to the application which displays the weight vs, time. The application is capable of data logging and providing brief advice on improving the espresso quality.

Had there been more time for this project, first and foremost, the PCB would have arrived allowing for a more compact and efficient implementation. Additionally, more time would have been able to be allocated to integration and developing more features for the product like more specific improvement advice or advice based on the experiences of other users.

10.1 Future Work

Since the assembled PCB and the replacement STM32s are arriving after the deadline for the project, it would be interesting to try and see this project through. All members of the team subscribe to "espresso culture" and could possibly benefit from a finished and polished version of this project. Other version of the device exist but it's well understood that they do not accomplish what they need to and the niche largely goes unfilled.

10.2 Lessons Learned

The main lessons learned from this are ordering evaluation boards for all components of the PCB before it arrives. This way, in the worst case, there is a very large and disjointed version of the finished product that only needs to be recreated in the more sleek and compact version offered by the arriving PCB. The design requirements were able to be met but to a much lesser extent because of how long it took to pivot to the risk mitigation plan. In the future, the risk mitigation plan should just be an in-tandem plan.

Glossary of Acronyms

- SPI – Serial Peripheral Interface
- I^2C – Inter-Integrated Circuit
- MCU – Microcontroller Unit
- LCD - Liquid Crystal Display
- GPIO - General Purpose Input/Output
- PCB - Printed Circuit Board
- IDE - Integrated Development Environment
- JTAG - Joint Test Action Group
- UART - Universal Asynchronous Receiver-Transmitter
- PLA - Polylactic Acid

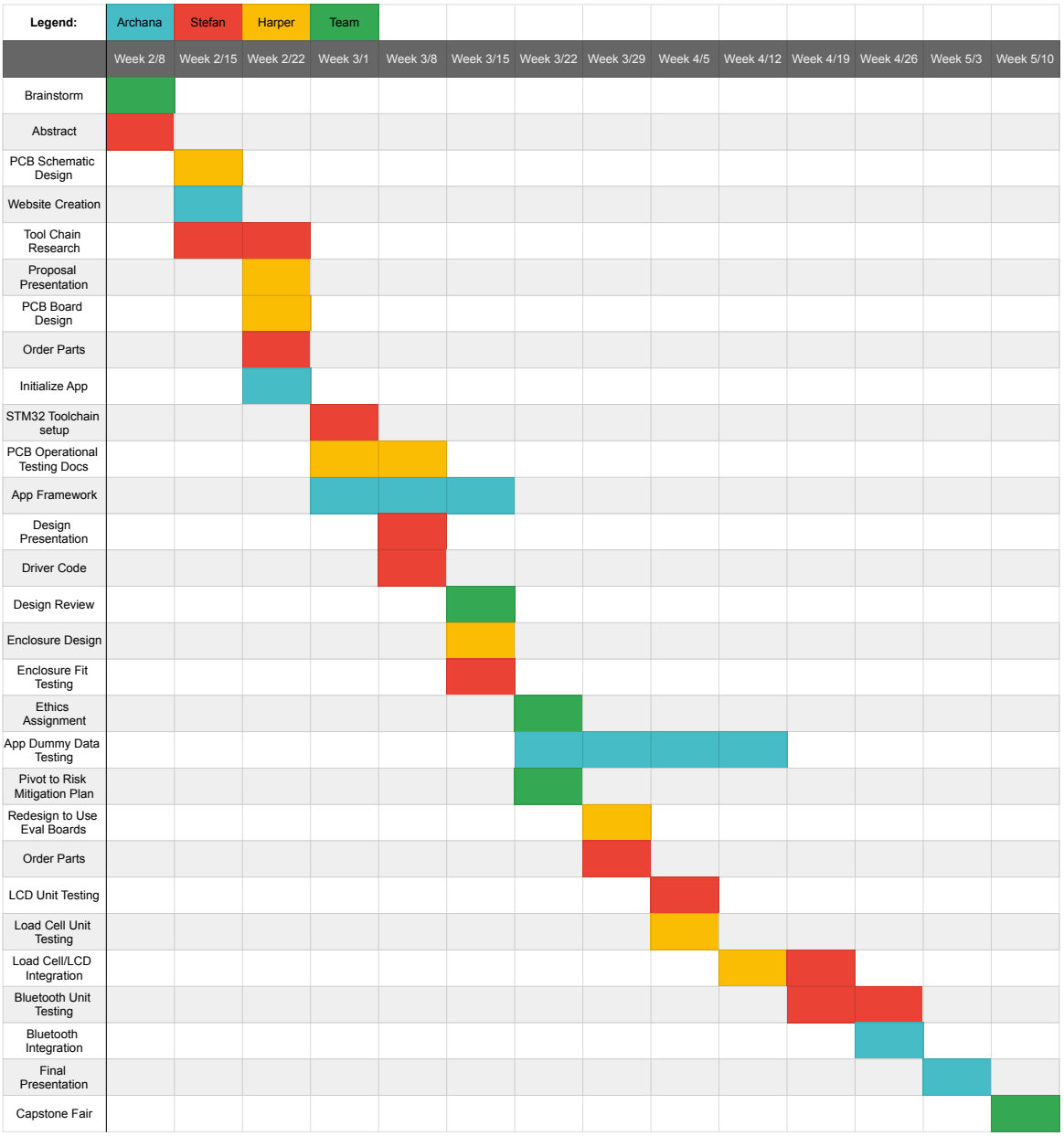
References

[1] Nuvoton Confidential. "NU78-02 24-bit ADC". In: 1.7 (Jan. 2012), pp. 1 –42.

[2] Atmel Corporation. "ATmega328P [DATASHEET]". In: 4 (Jan. 2015), pp. 1 –294.

[3] Nordic Semiconductor. "Product Specification v1.0". In: *nRF24L01+ Single Chip 2.4GHz Transceiver* 1 (Sept. 2008), pp. 1 –2.

[4] SparkFun. "SparkFun$_Q$*wiic*$_S$*cale*$_N$*AU*7802$_A$*rduino*$_L$*ibrary*". In: 1.0.4 (Jan. 2020).


Legend:
Archana
Stefan
Harper
Team
Week 2/8
Week 2/15
Week 2/22
Week 3/1
Week 3/8
Week 3/15
Week 3/22
Week 3/29
Week 4/5
Week 4/12
Week 4/19
Week 4/26
Week 5/3
Week 5/10
Brainstorm
Abstract
PCB Schematic Design
Website Creation
Tool Chain Research
Proposal Presentation
PCB Board Design
Order Parts
Initialize App
STM32 Toolchain setup
PCB Operational Testing Docs
App Framework
Design Presentation
Driver Code
Design Review
Enclosure Design
Enclosure Fit Testing
Ethics Assignment
App Dummy Data Testing
Pivot to Risk Mitigation Plan
Redesign to Use Eval Boards
Order Parts
LCD Unit Testing
Load Cell Unit Testing
Load Cell/LCD Integration
Bluetooth Unit Testing
Bluetooth Integration
Final Presentation
Capstone Fair

Listing 1: MCU loop code

```
void loop()
{
  if(myScale.available() == true)
  {
    if (digitalRead(0) == 0) baseline = zero_scale();
    long currentReading = (myScale.getReading()-baseline)/val2gram;
    SerialUSB.print("Reading: ");
    SerialUSB.println(currentReading);
    nrf_service_data buf;
    buf.service_uuid = 0x180A;
    float temp = currentReading;
    buf.value = temp;

    SerialUSB.print(".");
    //Needs to be large enough to hold the entire string with up to 5 digits
    char tempString[100];
    // transmit to device #1
    Wire.beginTransmission(DISPLAY_ADDRESS1);

    //Put LCD into setting mode
    Wire.write('|');
    //Send clear display command
    Wire.write('-');
    sprintf(tempString, "Weight %dg",currentReading);
    Wire.print(tempString);
    Wire.endTransmission(); //Stop \(I^2C\) transmission
    sprintf(tempString, "Weight %dg",currentReading);
    btle.begin(tempString);
    btle.advertise(0,0);
    //btle.advertise(0,0);
    btle.hopChannel();
  }
}
```