softwareEngineers — Socially Distanced Dispenser

Jake Moses, Quincy Mendelson, Jon Burkher, Justin Galante

Date of Submission

## Capstone Design ECE 4440 / ECE4991

## Signatures

## Statement of work:

In this section, each team member should provide 1 or 2 paragraphs describing their individual contributions to the project. This needs to be detailed and list several specific examples of work performed and how it fit within the context of the whole project.

Table of Contents

[Capstone Design ECE 4440 / ECE4991 1](#_Toc58337922)

[Signatures 1](#_Toc58337923)

[Statement of work: 2](#_Toc58337924)

[Table of Figures 4](#_Toc58337925)

[Background 5](#_Toc58337926)

[Constraints 6](#_Toc58337927)

[Intellectual Property Issues 8](#_Toc58337928)

[Detailed Technical Description of Project 9](#_Toc58337929)

[Project Timeline 9](#_Toc58337930)

[Test Plan 12](#_Toc58337931)

[Final Results 15](#_Toc58337932)

[Costs 15](#_Toc58337933)

[Future Work 15](#_Toc58337934)

[References 15](#_Toc58337935)

[Appendix 17](#_Toc58337936)

Error! Hyperlink reference not valid.Error! Hyperlink reference not valid.Error! Hyperlink reference not valid.Error! Hyperlink reference not valid.Error! Hyperlink reference not valid.Error! Hyperlink reference not valid.Error! Hyperlink reference not valid.**Error! Hyperlink reference not valid.**Error! Hyperlink reference not valid.Error! Hyperlink reference not valid.Error! Hyperlink reference not valid.Error! Hyperlink reference not valid.Error! Hyperlink reference not valid.Error! Hyperlink reference not valid.Error! Hyperlink reference not valid.Error! Hyperlink reference not valid.

## Table of Figures

[Figure 1 Original Gantt Chart 9](#_Toc58337996)

[Figure 2 Final Gantt Chart 11](#_Toc58337997)

[Figure 3 Bluetooth Test Plan 13](#_Toc58337998)

[Figure 4 Mobile Application Test Plan 14](#_Toc58337999)

**Error! Hyperlink reference not valid.**

**Abstract**

In the age of Covid-19, limiting the number of surfaces that are touched by multiple people is a key factor in slowing the spread. To help achieve this, the Socially Distanced Dispenser will serve as a contactless food dispenser, best deployed in a setting with many potential users such as a grocery store or a dining hall [1]. The dispenser will take user input from a smartphone application over a secure Bluetooth connection and automatically dispense the desired amount of food, limiting the required contact for any user to receive their food to their personal smartphone. Each module of the Socially Distanced Dispenser is self-sufficient aside from the occasional battery change and food item refill.

## Background

The inspiration behind this project was to create a product that in some way addressed the current situation of the world. Grocery stores or dining halls are high-risk zones for contraction of the virus, which poses the inconvenience of having to frequently sanitize common surfaces between customers. To ensure these common surfaces are cleaned, effort on either the customer’s end or an employee’s end has to be spent, and given the countless number of common surfaces, it is increasingly hard to guarantee that all surfaces are cleaned between each customer’s use. The Socially Distanced Dispenser addresses this issue by eliminating any need for contact in the first place, which allows for more focus and effort to be directed to other surfaces that require frequent cleaning.

The concept of contactless dispensers is not novel. Most hand sanitizer dispensers rely on infrared or photo sensors to dispense product without contact [2]. This approach, however, would not make sense in the context of our problem. Dispensers akin to the automatic hand sanitizer can only dispense a set, static amount. If the user wants more sanitizer than the set amount, they will have to trigger the dispensing mechanism repeatedly. If the user wants less than the set amount, that’s simply not an option and some product would have to go to waste. For the context of the hand sanitization problem, this is fine, as most users will be content with one dispense of the set amount. Food, however, is a different issue. Many users will desire many different amounts of food and having to repeatedly trigger the dispensing mechanism to receive the desired amount is inconvenient. And if the user wants less than the set amount, food will have to be wasted, which is not good practice.

  Another system that could be considered similar to the Socially Distanced Dispenser is Coca Cola’s ‘Freestyle Beverage Dispenser’. Coke’s dispenser allows users to choose from a wide variety of drinks by having users scan a QR code on their smartphone, which then directs them to a web application where users can select their beverage [3]. This approach saves the user from having to download an app and from having to form a Bluetooth connection with the machine, but the overhead is far greater than what the Socially Distanced Dispenser would need. A web application is far out of scope as each module only dispenses one type of food item and the customization options are limited simply to quantity. The Freestyle Beverage Dispenser allows users to choose from hundreds of different items and customization options which necessitates serving large amounts of user-facing views and having a database. In addition to the large overhead of a web application, the price of a single module of the Freestyle Beverage Dispenser can range from $2000 to $11,500 [4]. So, while having to download an app is inconvenient for the customer, it’s a one-time action. The inconvenience would be diminished with each use of the Socially Distanced Dispenser.

The Socially Distanced Dispenser is a unique solution to the problem it addresses because it serves as a simple, cost efficient, easy to use, and appropriately scaled contactless dispenser. In addition, all of the aforementioned contactless dispensers were for liquid products whereas the Socially Distanced Dispenser serves solid products. This difference necessitates a completely different automated dispensing mechanism, which the Socially Distanced Dispenser provides.  The prototype we develop will also be portable and is essentially ready to use out of the box. Each module is self-contained, and no external connections are needed. So as soon as our dispenser is powered and the desired food item is loaded, it will be ready to go, which allows for simple installation and minimal maintenance.

This project draws directly from much of our past coursework. In order to dispense the food, a stepper motor will be attached to the knob of the dispenser. This motor will be controlled using a MSP430 microcontroller, incorporating knowledge gained from ECE3430: Introduction to Embedded Computing. The MSP430 will be connected to the Bluetooth module and the motor driver using a PCB, the circuit design of which will be informed by our experience in the Fundamentals of Electrical Engineering series (ECE2630, ECE2660, and ECE3750). We will also be using some signal processing and control systems techniques to allow communication between the motor and the microcontroller, conveying information such as how much to dispense or whether the dispenser is empty or jammed. The user interface also utilizes our previous classes, including Advanced Software Development and Mobile App Development (CS3240 and CS4720). These classes helped us learn to create clean and easy to use user experiences that do not discourage customers from using the product. The application will also need to establish a secure Bluetooth connection that does not jeopardize user privacy, which will require knowledge from Defense Against the Dark Arts (CS4630) and Introduction to Cybersecurity (CS3501). Finally, as 3D printing will be an aspect of our project, the experience of designing objects in CAD software from the Introduction to Engineering (ENGR 1620 and 1621) course will be useful as well.

## Constraints

*Manufacturability and Usability*

Our dispenser will most likely use two custom-made pieces: the enclosure for the moving parts of the machine, and the piece to attach the motor to the knob of the dispenser. A prefabricated enclosure may be purchased if one of an appropriate size and shape can be found. Otherwise, the enclosure will be made using plastic or wood, and will require some simple machining, such as use of a water jet or laser cutter. The motor attachment will be 3D printed, and will likely be a small and simple design, making it fast to print and cheap to produce.

The rest of our parts, including the dispenser itself, will be purchased off-the-shelf from various vendors, but primarily Digikey. The chips and other components that will be included in the PCB and large enough to solder easily, making the electronic components simple to manufacture.

Another aspect we must consider is how easy it is for a consumer to interact with our product. If it is difficult or time consuming for a user to connect to the dispenser, then the user will most likely choose not to do so and collect food elsewhere. We must keep this in mind as we design the mobile application to ensure the user can connect swiftly and without problems. On this topic, the application must be simple and intuitive enough to take roughly as much time as the user manually turning the knob. We must also take into consideration any complications the dispenser could have and how they should be dealt with. Problems such as mechanical or electrical failure, dispenser jams, and parts becoming dirty must all be carefully considered.

*Part Availability*

All parts that must be ordered from Digikey have been checked and have large quantities in stock.  Other board components, such as resistors and capacitors, can be taken from our lab kits from the Fundamentals of Electrical Engineering series. The physical dispensers we are using are in no shortage of stock and can easily be found at various retailers. Our main concern is that our ability to manufacture custom parts will be limited due to COVID-19 restrictions.

*Economic Constraints*

The main economic constraint in this project is that our budget is limited to $500 dollars. That means we must be careful in our spending so we can efficiently acquire needed quality parts and don’t overspend on areas in which cost can be reduced. To do this, we start with the major expenses first. We estimate the cost of our PCBs to be around $200. This accounts for the ordering of three boards, each of which will cost approximately $33, plus an additional $25 per board for shipping. Another expense is 3D Printing the parts needed to turn the rotary valve, which comes out to ~$60. Ordering the dispenser model should take about $40. The components needed for the board to function, which includes the MSP 430, the Bluetooth module, and the motor driver, comes out to $30, while the stepper motor can be acquired for $20. In total, acquisition of parts comes out to around $360, which leaves plenty of room for unexpected expenses, as well as space for expansion on the project. Since our project’s end users are customers, we want to ensure our product is easily affordable for our target market (grocery stores, dining halls, etc.) to achieve wider availability. Since the cost for one Socially Distanced Dispenser module is of around $140 and these places generally have sizable budgets, there shouldn’t be any major economic constraints.

*Environmental Impact*

The day-to-day use of the Socially Distanced Dispenser will not have much of an environmental impact, besides perhaps saving resources spent on sanitization materials. The main environmental concern is during the manufacturing and the end-of-life phases. The Socially Distanced Dispenser consists of a fair amount of plastic, which poses concerns for when it’s time to dispose of the dispenser [11]. The batteries used to power the dispenser will contain toxic materials and will need to be changed out as use of the dispenser persists [12]. In regard to the electronic components such as the Printed Circuit Board, there are standards for responsibly recycling and reusing electronic waste that certified electronic recyclers must follow [13]. To minimize the environmental footprint of the dispenser, we will encourage owners to utilize a certified electronic recycler at the end of the dispenser’s life.

*Sustainability*

The only maintenance required to ensure smooth, sustained operation is battery replacement and occasional cleaning of the dispenser to adhere to FDA cleanliness requirements. The lifespan of the Socially Distanced Dispenser is limited to the battery and cleaning material supply of the owner, but this should be of little concern to established grocery stores or dining halls. A possible improvement to the project would be to perhaps connect multiple dispensers and motors to one controller and allow users to select from the options as opposed to having an individual PCB for each dispenser. In the event of degradation of the dispenser, all parts should be easily replaceable and reproducible as mentioned in the *Manufacturability and Usability* section.

*Health and Safety*

The Socially Distanced Dispenser qualifies as a vending machine as defined by the Food and Drug Administration in its 2017 FDA Food Code. According to these guidelines, we must provide tight-fitting covers to protect the food in the container from customer tampering. Our dispenser is designed to hold shelf-stable, dry food such as rice or cereal, and is therefore exempt from many of the guidelines defined by the FDA for temperature-controlled vending machines. We must also, however, provide information about the cleaning procedures required for our device. Per FDA regulations for bulk food available for customer self-dispensing, we will also provide a place on our dispenser to display a label containing the name, ingredient list, and nutrition information of the food item contained in the dispenser [14].

Our project also involves moving parts, namely the stepper motor used to actuate the dispensing mechanism, which poses the concern of a user potentially injuring themselves. This is discussed further in the ***Standards*** section of the proposal.

*Ethical Considerations*

As far as normal operation of the dispenser goes, the only ethical concern would perhaps be that the dispenser is only accessible to users with smartphones. In this day and age, however, essentially all smartphones have Bluetooth capability, and in 2019, 81 percent of adult Americans owned a smartphone and 96 percent owned cellphones in general [15]. So, the ethical constraint on the usage of our project is relatively insignificant as an overwhelming majority of all potential users have the ability to operate the dispenser. A potentially more tangible opportunity for ethical concerns to arise would be when the user establishes a Bluetooth connection. To ensure that no malicious third party can invade users’ privacy, extra care will be taken when forming the connection and when transmitting data to and from the dispenser. All Bluetooth standards will be followed to achieve this.

### Intellectual Property Issues

In this section you should discuss the patentability of your project. You should include references to 3 US patents whose claims encompass material similar to your project and explain why (or why not!) you feel your project might be patentable in light of those claims. Your project does not have to be patentable, but you need to explain why, or why not. You should list specific claims in the patents and explain of the are dependent or independent claims.

## Detailed Technical Description of Project

This should a very detailed section. It should include the points. (Not bulleted in your report!) It should explain in sufficient detail that would allow a 4th year undergrad to exactly duplicate your results at the beginning of their fall semester.

* What it is
* How it works
* Components used – All should be referenced2]
* Design Decisions and tradeoffs
* Block Diagrams
* Schematics (use sections of schematics and explain thoroughly and legibly)
* Board Layouts (explain why layout decisions were made)
* Problems and design modifications

## Project Timeline

Throughout our project there were many tasks that were able to be completed in parallel, and then towards the culmination of the project all of these tasks were polished and sequentially tested and built into the final dispenser. Our original Gantt Chart— depicting our predicted timeline— is shown below, followed by the final Gantt Chart which shows our actual timeline:

![A picture containing text, writing implement, stationary, pencil

Description automatically generated

Figure 1 Original Gantt Chart

Originally, our team estimated that we would be able to complete the project by Thanksgiving break and then focus on just performing the demo after break. However, due to difficulties with our voltage regulator—will go into more detail later on in this report—we ended up not completing the project until after break, due to the need to wait for a final part order. In addition to this differentiation, in our original timeline we only scheduled 1 instance of soldering, PCB design, and part ordering, whereas in reality we had 2-3 instances of each of those tasks; this was due to the fact that errors in our first PCB iteration had to be corrected in a second board send-out. This makes up the main differences between the Original Gantt Chart and the Final Gantt Chart, the other differences being small date differences due to the mis-estimation of how long it would take to accomplish a specific task. An example of this small timeline difference is depicted in the time we estimated it would take to get the motor turning via the Bluetooth, as we estimated that this would only take about a week, however due to difficulties with the setup on the PCB we were not able to get Bluetooth-triggered motor turns until after 3 weeks.

Overall, the main causes of the differentiation of dates between our proposed timeline and the actual timeline were the need to have multiple board send-outs as well as part orders. This also meant that had to do additional integration tests later on in the project and pushed back our project completion dates. Thankfully, since we were planning on having our project done before Thanksgiving break even with the delays we encountered, we were able to have our demo completed by the deadline of December 10th.

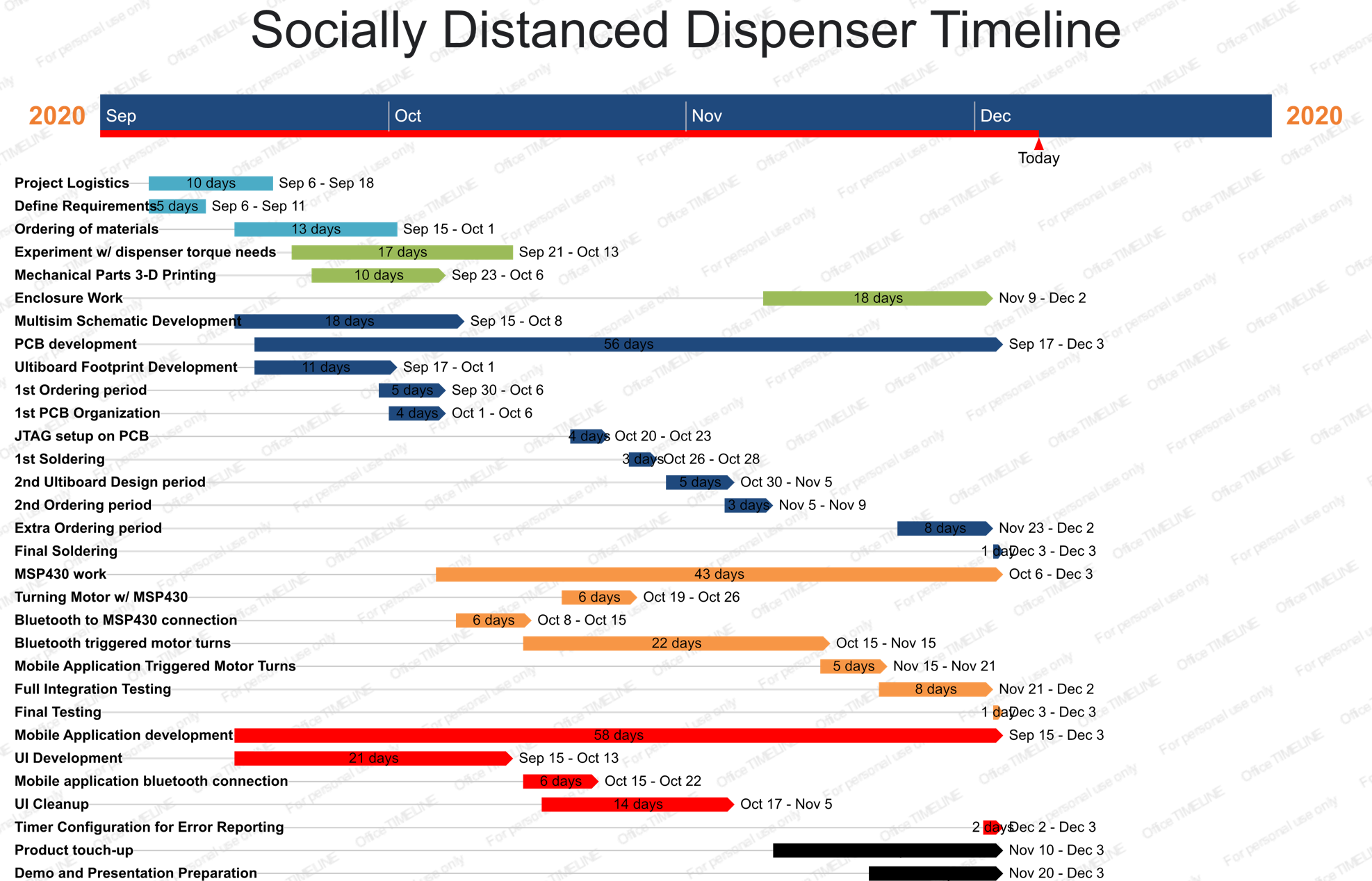


Figure 2 Final Gantt Chart

At the onset of this project there were multiple tasks that were worked on in parallel: the mobile application development, MSP430 embedded code development, and the PCB Multisim/Ultiboard layout/construction. This allowed each of us to specialize in a specific part of the project— primary and secondary roles are discussed below— and then later on sequentially test and add each of the relative components to the project. As can be seen in the Final Gantt Chart above, the schematic, mobile application, and embedded code were all done in parallel in the months of September and October—MSP430 work was delayed due to delay in receiving the Launchpad. In the month of November (as well as late October), as all of the pieces of the project started to come together, we began performing more tasks sequentially. Especially after the first PCB came in Quincy and Jake had to sequentially go through and test the JTAG, motor, Bluetooth, and Hall Effect Sensor in order to figure out what we needed to do for our second board. At the same time, Jake was working with Jon and Justin on ensuring that the HM-11 Bluetooth Module effectively communicated with their mobile application. Towards the culmination of the project, the entire team came together to serially go through all of the tests on the final board (with the new voltage regulator) and then add smaller features to the product to ensure that it was ready for the demo.

**Who did What**

**Jake Moses**

* Primary: JTAG on-board debugging, configuration of the HM-11 Bluetooth module, as well as was responsible for writing all of the embedded code— Motor, Hall Effect Sensor, & Bluetooth — that was put in the MSP430FR2311 chip.
* Secondary: development of the Ultiboard footprints for different PCB components— independently worked on JTAG header and Bluetooth header footprints— as well as other hardware debugging & construction.

**Quincy Mendelson**

**Jon Burkher**

**Justin Galante**

## Test Plan

You should show the test plan from your proposal and explain how you followed this plan or how you modified it. You should explain each of your testing procedures, and how you divided your system into testable sub modules. If testing caused a partial redesign of your device, you should explain how you arrived at that conclusion and how it influenced your redesign.

**Bluetooth Test Plan**

The original test plan (from our proposal) is shown below:

Diagram

Description automatically generated

Figure 3 Bluetooth Test Plan

**Mobile App Test Plan**

Diagram

Description automatically generated

Figure 4 Mobile Application Test Plan

**Motor Test Plan**

**Hall Effect Sensor Test Plan**

**Integration/Overall Testing**

## Final Results

In this section you should explain the functionality of your final device in detail. You should honestly assess and explain which of the success criteria defined in your proposal you met and which you did not.

## Costs

In this section, you should outline your costs, with a detailed spreadsheet in your appendix. You should also consider how costs would change if you were to manufacture in 10000-unit quantities, i.e., look at Digikey to get estimates of costs in large quantities, and consider if automated equipment could be used to assemble your device and how that might influence costs.

## Future Work

In this section you should offer suggestions as to how the project might be improved or expanded upon if a future group of students wished to create a new project based upon yours. You should consider difficulties that were not foreseen at the beginning and offer advice on pitfalls to watch for.

# References

[1] Pullen, J., 2020. *5 Technologies Changing the Restaurant Industry*. [online] msnbc.com. Available at:   <http://www.nbcnews.com/id/48959179/ns/business-small\_business/t/technologies-changing-restaurant-industry/#.X1WNXHlKiUk> [Accessed 7 September 2020].

[2] urdesignmag. 2020. *An Insight on How the Best Automatic Soap Dispensers Work*. [online] Available at: <https://www.urdesignmag.com/technology/2018/05/14/an-insight-on-how-the-best-automatic-soap-dispensers-work/> [Accessed 6 September 2020].

[3] Magloff, L., 2020. *'Freestyle' Beverage Dispenser Offers Restaurants Contactless Pouring*. [online] Springwise. Available at: <https://www.springwise.com/innovation/food-drink/coca-cola-vending-machine-qr-codes-covid> [Accessed 6 September 2020].

[4] Cost Aide. 2020. *How Much Does Coca-Cola Freestyle Cost In 2020?*. [online] Available at: <https://costaide.com/coca-cola-freestyle-cost/> [Accessed 7 September 2020].

[5] Components101. 2020. *HC-05 - Bluetooth Module*. [online] Available at: <https://components101.com/wireless/hc-05-bluetooth-module> [Accessed 7 September 2020].

[6] “Stepper Motor Control Using MSP430™ MCUs.” Texas Instruments, Dallas, Texas, Sep-2017.

[7] "Visual Studio IDE, Code Editor, Azure DevOps, & App Center - Visual Studio", *Visual Studio*, 2020. [Online]. Available: https://visualstudio.microsoft.com/. [Accessed: 14- Sep- 2020].

[8] "Code Composer Studio (CCS) Integrated Development Environment (IDE)", *Texas Instruments*, 2020. [Online]. Available: https://www.ti.com/tool/CCSTUDIO. [Accessed: 14- Sep- 2020].

[9] "What is Multisim™?", *Ni.com*, 2020. [Online]. Available: https://www.ni.com/en-us/shop/electronic-test-instrumentation/application-software-for-electronic-test-and-instrumentation-category/what-is-multisim.html. [Accessed: 14- Sep- 2020].

[10] "Fusion 360 | 3D CAD, CAM, CAE & PCB Cloud-Based Software | Autodesk", *Autodesk.com*, 2020. [Online]. Available: https://www.autodesk.com/products/fusion-360/overview. [Accessed: 14- Sep- 2020].

[11] Sedaghat, L., 2020. 7 Things You Didn’t Know About Plastic (And Recycling). [online] National Geographic Society Newsroom. Available at: <https://blog.nationalgeographic.org/2018/04/04/7-things-you-didnt-know-about-plastic-and-recycling/> [Accessed 14 September 2020].

[12] Des.nh.gov. 2020. All About Batteries. [online] Available at: <https://www.des.nh.gov/organization/commissioner/pip/factsheets/hw/documents/hw-23.pdf> [Accessed 14 September 2020].

[13] US EPA. 2020. Certified Electronics Recyclers | US EPA. [online] Available at: <https://www.epa.gov/smm-electronics/certified-electronics-recyclers#:~:text=Currently%20two%20accredited%20certification%20standards,e%2DStewards%C2%AE%22).> [Accessed 14 September 2020].

[14] Food and Drug Administration, "FDA Food Code 2017", U.S. Department of Health and Human Services, College Park, MD, 2017.

[15] Pew Research Center: Internet, Science & Tech. 2020. Demographics of Mobile Device Ownership and Adoption in The United States. [online] Available at: <https://www.pewresearch.org/internet/fact-sheet/mobile/> [Accessed 14 September 2020].

## Appendix

In this section you should include helpful information that does not fit into the above categories but will be helpful in understanding and assessing your work. Complete code listings should be in this section, and detailed cad drawings.

**Embedded (MSP430) Code**

**Main.c**

#include <bluetooth.h>

#include <msp430.h>

#include <motor.h>

#include <MSP430FR2xx\_4xx/driverlib.h>

#include <sensor.h>

volatile unsigned char ReceivedValue = '\0';

int num\_turns;

bool startTurning = false;

void send();

void motorTurns(int turns);

int main(void)

{

WDTCTL = WDTPW | WDTHOLD; // stop watchdog timer

PMM\_unlockLPM5();

InitializePins(); // initializes bluetooth, sensor, and motor connection with msp

\_enable\_interrupts();

/\* ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

\* Main function

\* ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

\*/

//\*

while(1){

ENABLE\_SLEEP; //set sleep high

LOW\_NENBL; // set enable to low to turn on

ENABLE\_CONFIG;

ENABLE\_DIR;

LOW\_M0;

LOW\_M1;

DISABLE\_STEP;

while (ReceivedValue == '\0'); // wait until user connects to the device and sends a value

// TB0CTL |= MC\_1; // set Timer B to upmode

switch (ReceivedValue){

case '1':

ReceivedValue = '\0';

motorTurns(1);

send();

break;

case '2':

ReceivedValue = '\0';

motorTurns(2);

send();

break;

case '3':

ReceivedValue = '\0';

motorTurns(3);

send();

break;

case '4':

ReceivedValue = '\0';

motorTurns(4);

send();

break;

default:

UARTSendString("Please select a different choice");

ReceivedValue = '\0';

break;

}

}

//\*/

}

void send(){

if (ReceivedValue == 'z'){

ReceivedValue = '\0';

UARTSendString("Error Dispensing");

}

else{

UARTSendString("Successfully Dispensed");

}

return;

}

void motorTurns(int turns){

startTurning = true;

num\_turns = turns + 1;

while(num\_turns){

DISABLE\_STEP;

\_delay\_cycles(5000);

ENABLE\_STEP;

\_delay\_cycles(5000);

}

DISABLE\_STEP;

HIGH\_NENBL;

startTurning = false;

// //Motor set up

// TB0CTL |= MC\_0; // stops timer b

// TB0R &= 0; // Resets Timer B count to 0

return;

}

// For UART interrupt (communication with Bluetooth)

#pragma vector = USCI\_A0\_VECTOR

\_\_interrupt

void USCIAB0RX\_ISR(void)

{

ReceivedValue = UARTReceiveByte(); // read user input

if (ReceivedValue == 'z'){

num\_turns = 0;

}

UCA0IFG &= ~UCRXIFG;

}

// For Hall Effect Sensor Interrupt

#pragma vector=PORT1\_VECTOR

\_\_interrupt void Port\_1(void)

{

P1IFG &= ~SENSOR\_BIT; // Clear P1.1 IFG

if (startTurning){

if (num\_turns){

num\_turns--;

} else {

HIGH\_NENBL;

}

}

}

**bluetooth\_motor\_sensor\_setup.c**

#include <bluetooth.h>

#include <motor.h>

#include <sensor.h>

void InitializePins()

{

// Software reset enabled. USCI logic held in reset state.

UCA0CTL1 |= UCSWRST;

// Select USCI UART functionality.

UCA0CTLW0 |= UCSSEL\_\_SMCLK + UCRXEIE;

// Baud Rate calculation

UCA0BRW = 6; // 1000000/115200 = 8.68

UCA0MCTLW |= UCOS16 | UCBRF\_8 | 0x20; //0xD600 is UCBRSx = 0xD6

// UCBRSx value = 0xD6 (See UG)

UCA0BR1 = 0;

P1SEL1 &= ~(BIT6 | BIT7); // USCI\_A0 UART operation

P1SEL0 |= BIT6 | BIT7;

SET\_RECEIVE\_AS\_AN\_INPUT;

SET\_TRANSMIT\_AS\_AN\_OUTPUT;

// SET\_STATE\_AS\_AN\_OUTPUT; // Bluetooth state pin (not needed)

// SET\_EN\_AS\_AN\_OUTPUT; // Bluetooth enable pin (not needed)

// Motor Configurations

SET\_SLEEP\_AS\_AN\_OUTPUT;

SET\_NENBL\_AS\_AN\_OUTPUT;

SET\_STEP\_AS\_AN\_OUTPUT;

SET\_DIR\_AS\_AN\_OUTPUT;

SET\_M0\_AS\_AN\_OUTPUT;

SET\_M1\_AS\_AN\_OUTPUT;

SET\_CONFIG\_AS\_AN\_OUTPUT;

SET\_NFAULT\_AS\_AN\_INPUT;

// Hall Effect Sensor Configuration

SET\_SENSOR\_AS\_AN\_INPUT;

UCA0TXBUF = 0; // initialize transmit buffer to 0 (for UART communication with Bluetooth)

UCA0CTL1 &= ~UCSWRST; // Initialize eUSCI (gets out of Reset state)

UCA0IE |= UCRXIE; // Enable USCI\_A0 RX interrupt

P1IE |= SENSOR\_BIT; // Enable HallEffectSensor Interrupt

P1IES |= SENSOR\_BIT; // make interrupt falling edge

P1IFG &= ~SENSOR\_BIT; // Clear interrupt flag

}

void UARTSendByte(unsigned char SendValue)

{

while (!(UCA0IFG & UCTXIFG)); //wait to be ready

UCA0TXBUF = SendValue;

while ((UCA0STATW & UCBUSY));

}

unsigned char UARTReceiveByte()

{

while (!(UCA0IFG & UCRXIFG)); //wait until ready to read

unsigned char ReceiveValue = UCA0RXBUF;

while ((UCA0STATW & UCBUSY));

return ReceiveValue;

}

void UARTSendString(unsigned char \*str)

{

if (str != NULL) {

while (\*str != '\0') {

UARTSendByte(\*str);

str++;

}

}

}

bluetooth.h

#ifndef BLUETOOTH\_H

#define BLUETOOTH\_H

#include <msp430.h>

#include <stddef.h>

/\* ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

\* UC0RX USCI\_A0 receive data input in UART mode

\* ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

\* GPIO : P1.6

\* ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

\*/

#define USCIA0\_RECEIVE\_BIT BIT6

#define USCIA0\_RECEIVE\_PORT P1IN

#define USCIA0\_RECEIVE\_DDR P1DIR

#define SET\_RECEIVE\_AS\_AN\_INPUT USCIA0\_RECEIVE\_DDR &= ~USCIA0\_RECEIVE\_BIT

/\* ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

\* UC0TX USCI\_A0 transmit data output in UART mode

\* ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

\* GPIO : P1.7

\* ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

\*/

#define USCIA0\_TRANSMIT\_BIT BIT7

#define USCIA0\_TRANSMIT\_PORT P1OUT

#define USCIA0\_TRANSMIT\_DDR P1DIR

#define SET\_TRANSMIT\_AS\_AN\_OUTPUT USCIA0\_TRANSMIT\_DDR |= USCIA0\_TRANSMIT\_BIT

#define BLUETOOTH\_EN\_BIT BIT5

#define BLUETOOTH\_EN\_PORT P1OUT

#define BLUETOOTH\_EN\_DDR P1DIR

#define SET\_EN\_AS\_AN\_OUTPUT BLUETOOTH\_EN\_DDR |= BLUETOOTH\_EN\_BIT

#define BLUETOOTH\_STATE\_BIT BIT0

#define BLUETOOTH\_STATE\_PORT P2OUT

#define BLUETOOTH\_STATE\_DDR P2DIR

#define SET\_STATE\_AS\_AN\_OUTPUT BLUETOOTH\_STATE\_DDR |= BLUETOOTH\_STATE\_BIT

void InitializePins();

void UARTSendByte(unsigned char SendValue);

unsigned char UARTReceiveByte();

void UARTSendString(unsigned char \*str);

#endif

**motor.h**

#ifndef MOTOR\_H

#define MOTOR\_H

#include <msp430.h>

#include <stddef.h>

#define SLEEP\_BIT BIT0

#define SLEEP\_PORT P1OUT

#define SLEEP\_DDR P1DIR

#define SET\_SLEEP\_AS\_AN\_OUTPUT SLEEP\_DDR |= SLEEP\_BIT

#define ENABLE\_SLEEP SLEEP\_PORT |= SLEEP\_BIT

#define NENBL\_BIT BIT4

#define NENBL\_PORT P2OUT

#define NENBL\_DDR P2DIR

#define SET\_NENBL\_AS\_AN\_OUTPUT NENBL\_DDR |= NENBL\_BIT

#define LOW\_NENBL NENBL\_PORT &= ~NENBL\_BIT

#define HIGH\_NENBL NENBL\_PORT |= NENBL\_BIT

#define STEP\_BIT BIT6

#define STEP\_PORT P2OUT

#define STEP\_DDR P2DIR

#define SET\_STEP\_AS\_AN\_OUTPUT STEP\_DDR |= STEP\_BIT

#define ENABLE\_STEP STEP\_PORT |= STEP\_BIT

#define DISABLE\_STEP STEP\_PORT &= ~STEP\_BIT

#define DIR\_BIT BIT5

#define DIR\_PORT P2OUT

#define DIR\_DDR P2DIR

#define SET\_DIR\_AS\_AN\_OUTPUT DIR\_DDR |= DIR\_BIT

#define ENABLE\_DIR DIR\_PORT |= DIR\_BIT

#define DISABLE\_DIR DIR\_PORT &= ~DIR\_BIT

#define M0\_BIT BIT3

#define M0\_PORT P1OUT

#define M0\_DDR P1DIR

#define SET\_M0\_AS\_AN\_OUTPUT M0\_DDR |= M0\_BIT

#define LOW\_M0 M0\_PORT &= ~M0\_BIT

#define HIGH\_M0 M0\_PORT |= M0\_BIT

#define M1\_BIT BIT2

#define M1\_PORT P1OUT

#define M1\_DDR P1DIR

#define SET\_M1\_AS\_AN\_OUTPUT M1\_DDR |= M1\_BIT

#define LOW\_M1 M1\_PORT &= ~M1\_BIT

#define HIGH\_M1 M1\_PORT |= M1\_BIT

#define CONFIG\_BIT BIT1

#define CONFIG\_PORT P2OUT

#define CONFIG\_DDR P2DIR

#define SET\_CONFIG\_AS\_AN\_OUTPUT CONFIG\_DDR |= CONFIG\_BIT

#define ENABLE\_CONFIG CONFIG\_PORT |= CONFIG\_BIT

#define NFAULT\_BIT BIT4

#define NFAULT\_PORT P1IN

#define NFAULT\_DDR P1DIR

#define SET\_NFAULT\_AS\_AN\_INPUT NFAULT\_DDR &= ~NFAULT\_BIT

#endif

**sensor.h**

#ifndef HALLEFFECTSENSOR\_H

#define HALLEFFECTSENSOR\_H

#include <msp430.h>

#include <stddef.h>

#define SENSOR\_BIT BIT1

#define SENSOR\_PORT P1IN

#define SENSOR\_DDR P1DIR

#define SET\_SENSOR\_AS\_AN\_INPUT SENSOR\_DDR &= ~SENSOR\_BIT

#endif

Delivered in a Word .docx format with track changes turned on