# MEDICAL KIT DISPENSER

By

Matthew Mitchell Chung

Dylan Edbert Hartato

Josh Leeman

Final Report for ECE445, Senior Design, Spring 2022

TA: Qingyu Li

May 4, 2022

Project 42

## Abstract

To address the shortening of operational hours of many convenience and drug stores due to the COVID-19 pandemic and the inaccessibility of the free medical supplies made available by the McKinley health center, our group created a medical dispensing device targeted to university students. The device is modular in design so that capacity can be increased depending on the foot traffic of the device location, and is integrated to online databases to keep track of a user's dispensing history and the available inventory.

The project successfully demonstrates the feasibility of this dispensing kit by implementing a single tray and showing that the design is generalizable to $2^n$ trays for a system using n-BUS lines. An issue encountered in using a RFID-based user identification is the encryption involved in an iCard; the ID of an iCard will change with every scan, and only readers with equipped with the proper decryption key with the user is able to correctly identify the user; this can be easily remedied if to be implemented in the U of I system.

# Table of Contents

# 1. Introduction

## 1.1 Problem

There have been instances during which medical necessities have been in need but are inaccessible, either due to how far the closest drug store is or the time of day during which such necessities are needed. For example, cold medicine is something that you often do not have at home and will only need when you are having a severe case of the sniffles––but circumstances are that you likely would not get such drugs if they are not relatively immediately available. Another scenario is when sometimes, the straps in our mask would snap off. Most people do not carry around a spare mask in their bag, which requires them to get another one from a store. In the era that we are currently in, addressing our illnesses and the safety of others as soon and as effectively as possible is out of everybody's best interest.

## 1.2 Solution

What we would like to do to address such issues is to build a modular vending machine that is targeted towards UIUC students and can be placed around campus. Our implementation of this machine is unlike any other vending machine that you can find either at ECEB or anywhere else for that matter. We would like to make it modular so that it can be as small (so that it can be placed in low-traffic areas) or as large (conversely, in high traffic areas) as it needs to be. A consequence of the modular design is that the trays that store inventory can be expanded vertically or horizontally to accommodate for every product size––a feature that is not found in any vending machine.

In addition, as this product is intended to serve the user more than to benefit the owner, the design of such device will be focused on ensuring that the user is able to obtain whatever product it is that they have ordered through a series of motion detectors. The vending machine is intended to provide goods that current students are able to obtain for free, either from McKinley or otherwise; however, such goods are often distributed to students on a quota. That is, students are able to dispense certain goods after some time period has elapsed. The software related to this device will thus serve two purposes: to track the user's past transactions to ensure that they are eligible to dispense a certain product, and to track inventory of the machine. Due to the required internet connection, an Arduino or Raspberry Pi will be used to make implementing the database-to-machine connection feasible for this project; however, the implementation of the actual machinery and any failsafe system will require at least 2 PCB boards; one to unify the BUS that connects to all the dispensing trays, the motion sensor, and the arduino so that the machine functions as intended, and the other to ensure that the individual trays dispenses an item when commanded.

Due to the modularity of the design and the implementation of the software, this machine can also serve as an all-in-one distribution center for goods that are often handed over to students as needed. While this machine is initially intended for distributing necessities, it can also be stocked with other items depending on where they are. For example, a machine at the ARC can also be used to vend sanitation wipes or some injury-related remedies.

## 1.3 Visual Aid

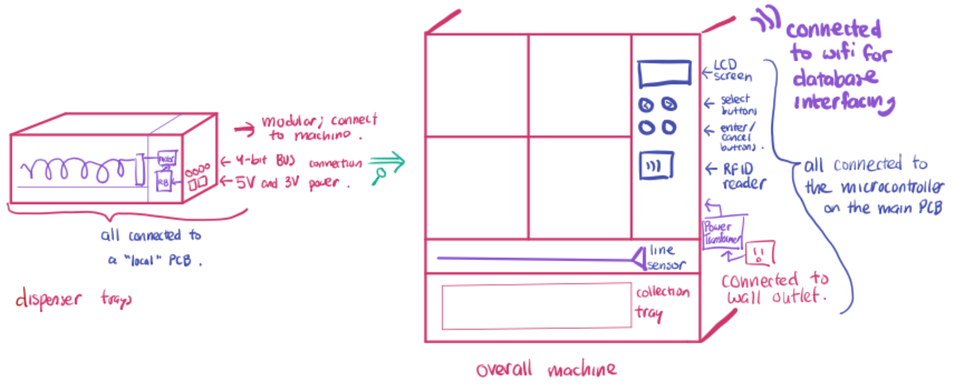

modular; connect to machine.
Motor
PCB
← 4-bit BUS connection
← 5V and 3V power.
all connected to a "local" PCB.
dispenser trays
connected to wifi for database interfacing
LCD screen
select buttons
enter/cancel buttons.
RFID reader
all connected to the microcontroller on the main PCB
Power Transformer
line sensor
connected to wall outlet.
collection tray
overall machine

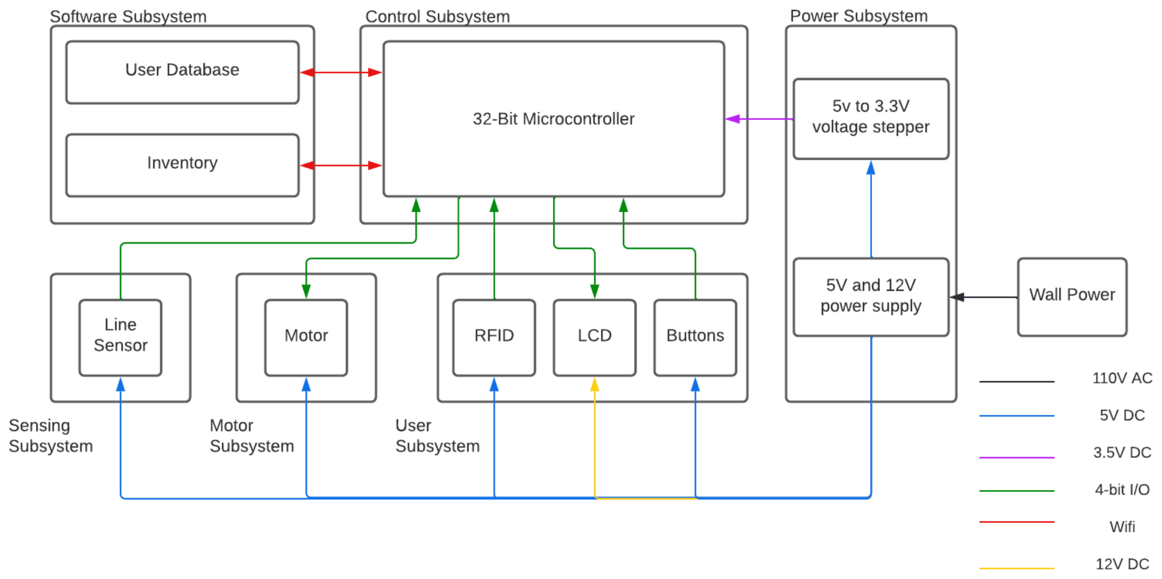
Figure 1. Physical Design of Medical Kit Dispenser

## 1.4 High-Level Requirements List

- ☐ The motion sensor should be able to detect if an item is dispensed by checking that the signal sent will be high, and should send a signal to the control module PCB. It should then update the user and inventory databases after it successfully dispenses a product but before dispensing the next product.
- ☐ The microcontroller should be able to read a user's identification using the RFID and successfully interpret the user ID, which prompts the LCD screen to show what the user can dispense; the user should be able to choose the product using the four buttons.
- ☐ When a product is chosen, the correct signal should be sent by the microcontroller to the BUS, and the corresponding module should activate the motor to dispense a product for five seconds before repeating up to three times.

# 2. Design

## 2.1 Block Diagram



(*) All our user I/O components will communicate with the 32-bit controller using generic I/O handling.

Figure 2. Block Diagram of Medical Kit Dispenser

For the project to be successful, the Medical Kit Dispenser will require two components: a hardware and a software component. The components board will be divided into five main units: controlling subsystem, motor subsystem, sensing subsystem, user subsystem, and the power subsystem. The control unit will consist of a 32-bit microcontroller. The power subsystem consists of a 5v power supply and a stepper that will power the control, sensing, motor and user subsystem. The user interface unit will consist of an RFID, LCD, buttons, and a line sensor. The RFID, LCD, buttons and line sensor will be connected to the 32-bit microcontroller in the control unit through wires to be connected to the PCB. The dispensing unit will consist of a motor  through a 4-bit BUS. The software components will consist of two items: user database and inventory. They both will be connected to the 32-bit microcontroller in the control unit through wires and connected to the internet through wifi.

## 2.2 Physical Design

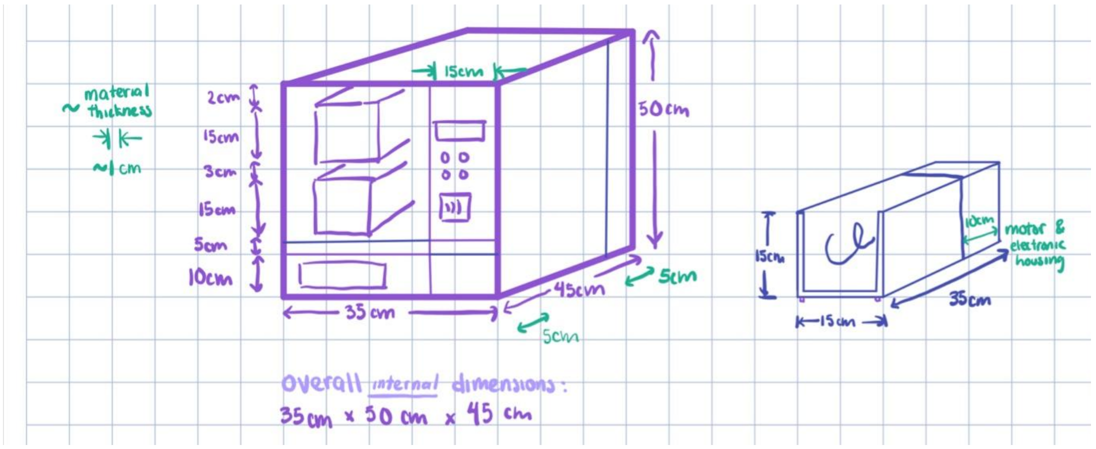


Figure 3. Physical Dimension and Design of Medical Kit Dispenser

## 2.3 Subsystem Requirements

### 2.3.1 Control Subsystem

This is the primary subsystem that ensures that the whole machine functions as intended. The microcontroller ensures that when a valid RFID signal is received, the user is able to select and receive products that they are eligible to dispense. This involves accessing the user database (subsystem 2.2.6) to ensure eligibility, and to display the eligible entries to the LCD screen (subsystem 2.2.4). When a valid input signal is received, a BUS signal is sent to the motors (subsystem 2.2.3) through a BUS, and when a signal is registered from the sensing modules (subsystem 2.2.2), the whole cycle repeats.

This subsystem primarily consisted of a Wifi-enables ESP-32 running firmware (Appendix C & Appendix B Figure 11) which receives input from the user subsystem, sensing subsystem and software subsystem and gives output towards the software subsystem, user subsystem and motor subsystem. The device first starts in the setup function. The device first blocks until it connects to wifi and detects that the RFID card reader is connected. It will then turn on the LCD screen with a welcome sign and initialize the input ports of 15, 18, and 19 and the output ports of 12, 13, 14, 27. With this the setup function is complete and the microcontroller moves on to the loop function.

The loop function is responsible will constantly be looping itself during the lifetime of the machine and does the main computation of the device. The first thing the device does is it resets all the output pins and the LCD display. Then the device blocks up and waits until a user taps an I-card on the RFID reader. When this occurs, the uin of the user is stored as a variable. The device then connects to the database online, sets up a JSON object with the users UIN inside and does a post request to the server. This post request creates a new row in the users database and users inventory database if the user has never used the vending machine before and else does nothing. The device then does a get request to the server with the endpoint of the devices location and users uin. The get request goes through the database and returns a list of items that the user has quota on and the vending machine has inventory of in that specific location. This list is returned to the device as a string which is then parsed and its contents placed in an array.

Given the array of objects the user can dispense, the device checks whether the user is not able to dispense any items. If the user is not able to dispense any items, the device returns from the loop function and restarts the loop. However, if the user is able to dispense at least one item, a loop is entered. This loop allows the user to use the buttons in the user subsystem to control the device. The left and right buttons when pressed changes the item printed on the LCD. The enter button breaks out of the loop and proceeds with the user's chosen item and the cancel button breaks out of the loop and restarts the loop function.

The device then takes the object selected by the user and finds the corresponding bus signal given the object. The signal is then sent through the bus to the motor until either a signal from the IR sensor is received signifying that the object has been dispensed or the motor tries to dispense the object 3 times. Depending on these outcomes, the device then connects to the server again and does a post request either updating the user inventory database for the users quota and inventory database for the stock of the item if the item is successfully dispensed or the request clears the stock of the item in the item database if the item could not be dispensed. The loop then ends and restarts waiting for an I-card to be tapped again.

### 2.3.2 Sensing Subsystem

The sensing subsystem will compromise all the sensors used. These sensors will detect whether an item has been properly dispensed. If an item has been properly dispensed, the object will pass through the line sensor and notify the microcontroller that the object has been dispensed, if not the microcontroller will know to retry. The sensing subsystem is basically an IR sensor that can detect within a certain range if there is an object that goes through. If it detects, it will send a high voltage, and this high voltage will be sent through logic gates so that it can send the appropriate signal to the microcontroller. The signal being sent to the microcontroller is always going to be three bits, so the information from the IR sensor is going to be compressed into a three bit information, which the microcontroller  will process.

### 2.3.3 Motor Subsystem

The motor subsystem is in charge of dispensing items. A signal will be sent by the microcontroller through the bus to the motor and will push the object down for dispensing. The signal sent by the microcontroller to the BUS will be four bits. This four bit information is going to contain the information on which tray to activate. The four bit information is going to be relayed to all the trays, and if the four bit information corresponds to that tray ID, it will then dispense that product. Overall, the motor subsystem is going to receive information from the microcontroller and check if the information corresponds to the tray ID through a series of logic and if it does, it will activate the motor.

### 2.3.4 User Interface Subsystem

The user subsystem comprises all the parts that the user will interact with including the LCD screen, RFID and buttons. The LCD screen is used so that users will be able to see what items they are able to dispense or choose what items to be dispensed. The LCD screen will connect to the control subsystem and what will be shown on the screens will be controlled by firmware. The buttons will be used for users to interact with the LCD screen and choose which product they would like to be dispensed. There are 4 buttons: left, right, enter and cancel. Each button when pressed will send a 3-bit signal to the control module which would be used in the firmware to determine the actions taken by the control module. The RFID module will be used to read the i-cards of users to identify who they are. The module will send the uin of the i-card to the control subsystem when tapped to be used and stored in the user database.

### 2.3.5 Power Subsystem

The power subsystem will be plugged into a standard wall plug and convert it to a 5V DC power supply. This will be used to power the user subsystem, the sensing subsystem and also the motor subsystem. From there the 5V power supply will be stepped down to 3V to power the 32-bit microcontroller. This subsystem basically contains a transformer that steps down 120V AC into a 5V DC. This 5V DC power is going to be routed to two different PCBS - the control PCB and the Motor PCB. The motor PCB operates at 5V, so it is going to power the entire PCB. The control PCB is going to operate at both 3.3V and 5V. The 5V is first going to be used to power the logic gates and LCD. Then the 5V power will be stepped down to 3.3V to power the microcontroller and the RFID module.

### 2.3.6 Database Subsystem

The software subsystem will compromise two parts: inventory and user database. The inventory component tracks the current items and corresponds that information to the LCD and the user database. The user database identifies users who are using the dispenser and inform the microcontroller to display what items can be dispensed based on the user quota. Another database is also used to connect these two databases. This database called the user inventory database will consist of information about the quota a user has on a specific item. These three databases are designed and implemented on SQL based on the db diagram (Appendix B Figure 13). Based on these three databases, a server is created on heroku which will house the databases and API calls can be made to update the databases or receive information from the databases.

The API calls are broken down into two endpoints: the inventory endpoint and user endpoint (Appendix D and Appendix B Figure 12). Within the inventory endpoint, a device can either call a get request or a post request. The get request will, given data on a users uin and device location, will run a query that will go through the inventory database and users inventory database and return a list of all the items on the machines location that has stock and the user has quota. The post request on the other hand can do two things. Receiving a JSON object consisting of information of the user's UIN, the chosen item and an item flag, the post request will first check the item flag. If the item flag is 1, a query will be made to the users inventory database and inventory database to decrement the quota of the user by 1 and the stock of the item by 1. On the other hand, if the item flag is 0, a query will be made to clear the stock of the item in the inventory database.

The other endpoint used is the user endpoint which consists of a post request. This post request receives a JSON object consisting of a user's UIN. From there a query will be made to check if the user already exists in the user database. If the user already exists, nothing will happen and a bad request will be sent. However, if the user does not exist, a query will be made to create a new row in the user database and a new row in the user inventory database for every item available in the vending machine.

## 2.4 Supporting Figures and Descriptions

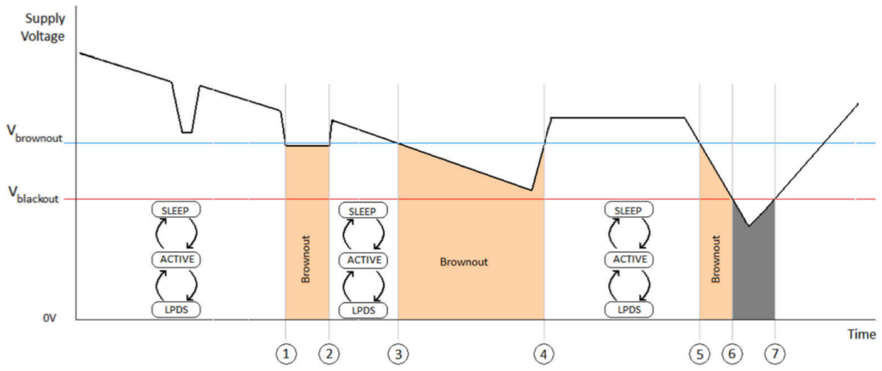


Figure 4. Brownout and Blackout Conditions of Microcontroller[2]

The supplied voltage must be above 2.1V but below 5V at all times to prevent brownout and blackout operation; as live data transmission is required for our device, it is imperative that the device does not enter the two aforementioned conditions.

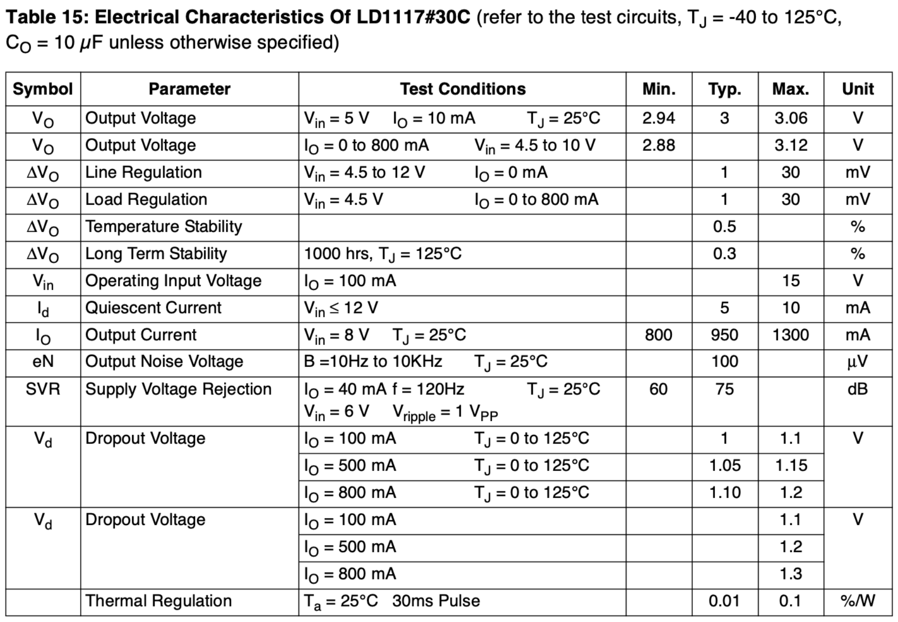


**Table 15: Electrical Characteristics Of LD1117#30C** (refer to the test circuits, $T_J$ = -40 to 125°C, $C_O$ = 10 $\mu$F unless otherwise specified)

| Symbol | Parameter | Test Conditions | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|
| $V_O$ | Output Voltage | $V_{in}$ = 5 V  $I_O$ = 10 mA  $T_J$ = 25°C | 2.94 | 3 | 3.06 | V |
| $V_O$ | Output Voltage | $I_O$ = 0 to 800 mA  $V_{in}$ = 4.5 to 10 V | 2.88 | | 3.12 | V |
| $\Delta V_O$ | Line Regulation | $V_{in}$ = 4.5 to 12 V  $I_O$ = 0 mA | | 1 | 30 | mV |
| $\Delta V_O$ | Load Regulation | $V_{in}$ = 4.5 V  $I_O$ = 0 to 800 mA | | 1 | 30 | mV |
| $\Delta V_O$ | Temperature Stability | | | 0.5 | | % |
| $\Delta V_O$ | Long Term Stability | 1000 hrs, $T_J$ = 125°C | | 0.3 | | % |
| $V_{in}$ | Operating Input Voltage | $I_O$ = 100 mA | | | 15 | V |
| $I_d$ | Quiescent Current | $V_{in} \leq$ 12 V | | 5 | 10 | mA |
| $I_O$ | Output Current | $V_{in}$ = 8 V  $T_J$ = 25°C | 800 | 950 | 1300 | mA |
| eN | Output Noise Voltage | B =10Hz to 10KHz  $T_J$ = 25°C | | 100 | | µV |
| SVR | Supply Voltage Rejection | $I_O$ = 40 mA f = 120Hz  $T_J$ = 25°C<br>$V_{in}$ = 6 V  $V_{ripple}$ = 1 $V_{PP}$ | 60 | 75 | | dB |
| $V_d$ | Dropout Voltage | $I_O$ = 100 mA  $T_J$ = 0 to 125°C | | 1 | 1.1 | V |
| | | $I_O$ = 500 mA  $T_J$ = 0 to 125°C | | 1.05 | 1.15 | |
| | | $I_O$ = 800 mA  $T_J$ = 0 to 125°C | | 1.10 | 1.2 | |
| $V_d$ | Dropout Voltage | $I_O$ = 100 mA | | | 1.1 | V |
| | | $I_O$ = 500 mA | | | 1.2 | |
| | | $I_O$ = 800 mA | | | 1.3 | |
| | Thermal Regulation | $T_a$ = 25°C  30ms Pulse | | 0.01 | 0.1 | %/W |

Figure 5. Electrical Characteristics of the 5V to 3.3V stepper[4]

The stepper is able to output a stable voltage of above 3V under a 5V input voltage. Assuming proper functionality, this ensures that the microcontroller is always active.

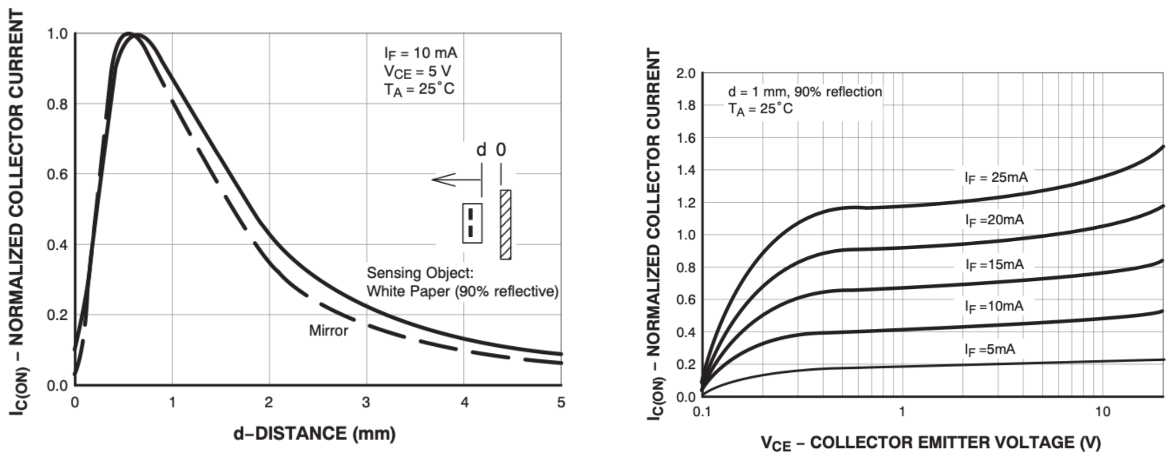


Figure 6. IV Curves of the Line Sensor as a function of distance and $V_{CE}$[3]

The line sensor will be designed to be active low; that is, it will have a "high" output when no item is dispensed and "low" when otherwise. As the device operates using a phototransistor, it will detect an object by an instantaneous lack of reflectivity that causes the photocurrent to decrease. The current can be passed through a resistor and connected to the microcontroller to probe a "high" or "low" state.

## 2.5 Circuit Diagram

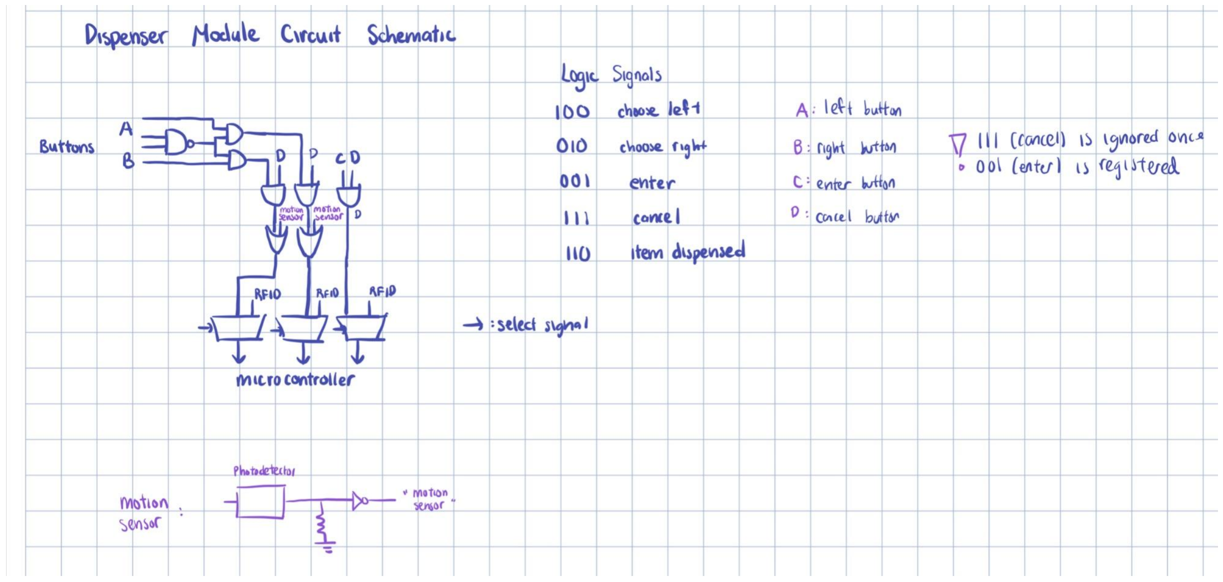


Figure 7: Circuit Diagram of Relevant Electronic Components Requiring Logic Design

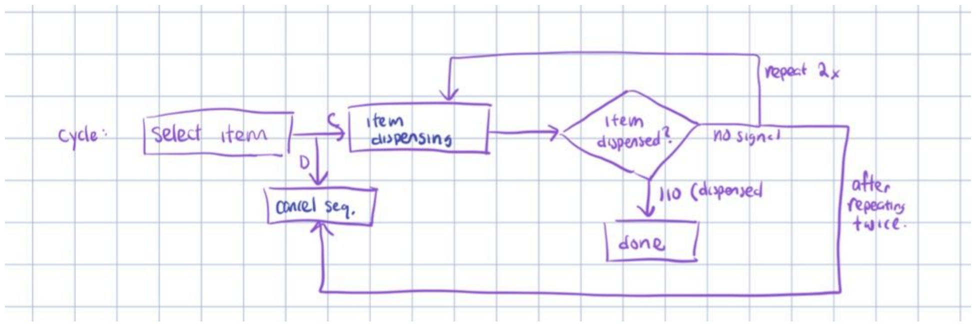


Figure 8: Functional Block Diagram of Circuit in Figure 7

Figure 7 shows the logic implementation required for our device; our microcontroller has 24 general I/O pins, and 16 pins will be dedicated to the LCD screen and four pins to the BUS. Consequently, we have four pins to drive the remaining logic; therefore, a series of states will be used to break down the signals from each device. The circuit diagram in Figure 7 shows how three input pins and one output select pin is sufficient to drive the remaining logic, which involves the input from the RFID sensor, the buttons, and the motion sensor, as facilitated by a series of multiplexers and logic gates. We will be implementing an I2C protocol for communications between the RFID module and the microcontroller, and the pins compatible for such communication have been assigned as such.

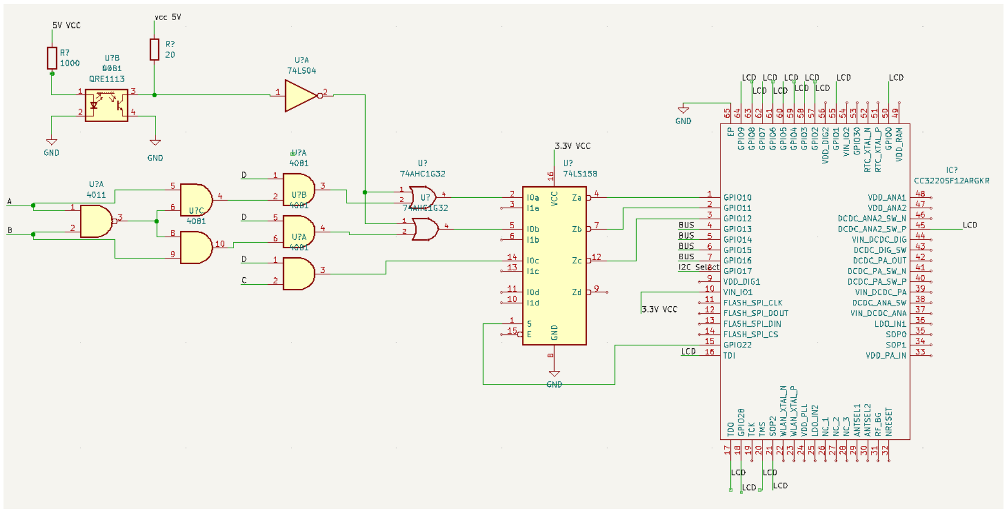


Figure 9: Circuit Diagram of Controller Module; Implementation and Integration of Circuit in Figure 7

The circuitry required to implement the dispenser module and activating the motors is relatively simple; if the appropriate signal is being transmitted across the BUS, then a series of NXOR gates should transmit a 1 signal; if all four signals are active, then the motor should detect a high signal and run for one cycle, which will be directly clocked by the 32-bit microcontroller across the BUS.

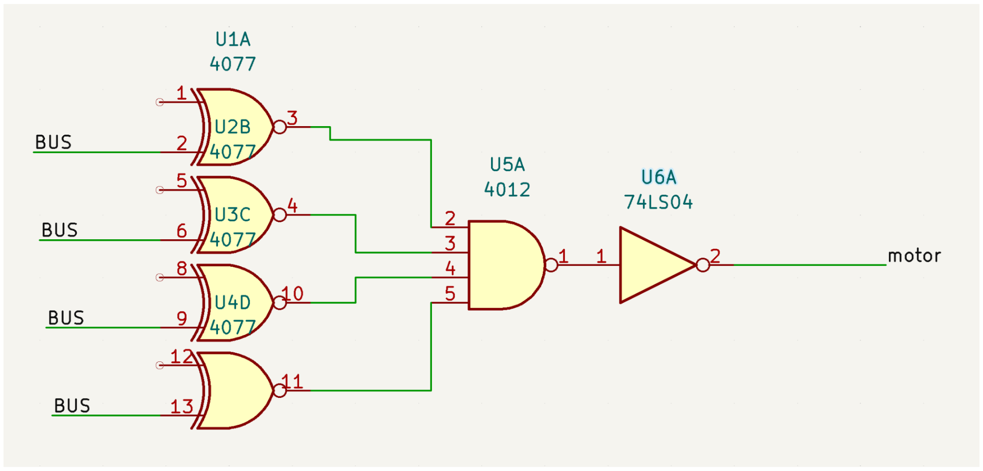


Figure 10: Circuit Diagram of the Dispenser Module

## 2.6 Tolerance Analysis

The most prominent software issues will likely be errors in communication between the machine's microcontroller and an online database; an example would be an unsuccessful update of the inventory database and the user database. However, as the reliability of the internet connection that the device will rely on to communicate with the online database is out of our control, our tolerance analysis will simply be that an update to inventory and the user information will be done prior to the next dispensing cycle.

For hardware, the voltage response of the line motion sensor depends on reflectivity, which itself depends on the distance between the object and the sensor and of the material being dispensed. The $V_{CE}$ of the phototransistor saturates at 0.3V; as such, assume that the target voltage entering the photoresistor is 0.5V during the "on" state. To that end, as we are using a 20 Ohm resistor to pin down the voltage under flowing current, and as shown in Figure 6, choose the collector voltage to be 0.2A. When the reflectivity is measured to be low, assume that the reflectivity can yield a normalized current within the range of $I_n \in [0, 0.3]$, implying a current of $I \in [0, 0.06]\, A$ when an item is sensed to have dispensed.[3] Therefore, during the "low" state of the motion sensor, the voltage being transmitted to the logic will be calculated below:
By design, choose $V_C$ = 5V and R = 20 Ohms. Assume that I = 0.06A as the upper current limit during the off state and I = 0.2A during the on state. Then using V = IR, during the on state, the voltage should be V = 5 - (20*0.2) = 1V, and during the off state V = 5 - (20 * 0.2 * 0.3) = 3.8V. The trigger voltage for the AND gates is 3V, meaning that the logic gates are able to differentiate between ON and OFF states well within our tolerance ranges. [5]

There can also exist a time delay in communication between the hardware components that are difficult to determine ahead of time. For example, the time required for the RFID reader to completely read an ID card and transmit the time to the microcontroller is a function of clocking speed, the size of data to be transmitted, and processing time by both the microcontroller and the RFID reader. As such, while we are predicting that the operation should take five seconds, we will need to determine the total processing time; the processing time should be consistent and we expect that the variance should be in a range of +/- 1 second.

# 3. Verification

## 3.1 Control Subsystem

There are four requirements for the control subsystem, the first one being that the correct bus signal be sent when a product is chosen. To verify this requirement, the motor subsystem is set to the signal 1001 corresponding to a mask. Then the mask is chosen on the vending machine and tested to see if the motor turns on, which it did. Then a different item is chosen on the vending machine to test whether the motor would still turn on. Thus a pill was chosen to be dispensed on the machine and the vending machine motor did not turn on, determining that the correct bus signal is being sent to the motor when a product is chosen.

The other requirement of the control subsystem is the correct products should be displayed on the LCD screen. To verify this requirement, the inventory database is stocked with 10 items, 5 of which had stock,  and a user was set up on the user database with a quota on half of the items. Then the machine was started with that user and using the buttons we went through all the items that were displayed on the LCD screen. To our avail, 3 items were shown on the LCD screen corresponding to the items which had inventory and the user had quota on, demonstrating that the LCD screen displayed the correct products the user could display.

The third requirement of the control subsystem is that an RFID signal be received and correctly interpreted by the microcontroller. To do this we connected the RFID module to the microcontroller and tapped an I-card to the device. From there we could see that the uid of the i-card could be displayed and obtained demonstrating that the RFID module is able to obtain data from I-cards and that data be correctly interpreted by the microcontroller.

The final requirement of the control subsystem is that when a product is dispensed or indispensable, the databases be updated accordingly. To verify this requirement, an item is dispensed twice, once with the IR sensor being tripped and one without the IR sensor being tripped. When the item was dispensed while the IR sensor was tripped, a change could be seen in the inventory database that the stock of that item decremented by 1 and the quota of the user corresponding to the item decremented by 1. When the IR sensor was not tripped, a change could be seen in the inventory database where the stock for the given item was set to 0, thus demonstrating that the databases were updated correctly when an item is dispensed/indispensable.

## 3.2 Sensing Subsystem

With the sensing subsystem, we had to try and verify whether the IR sensor is behaving as expected by using LCDs. To do so, we tested the circuitry of the signals delivered by the IR sensor using 3 LCDS that are connected to the MUX. If a signal is delivered from the IR sensor, the LCD should show a 011. This means that the LCD should be off - on - on in that specific order. When we tested out the logic and connected the sensor to the MUX, it displayed the correct signals through the LCDs.

We then try to connect the signals from the sensor to the microcontroller and try to dispense a product. As it tries to dispense, sending a signal from the IR sensor should stop the motor and update the inventory. Initially, sending a signal from the IR sensor did not stop the motor. Since we know that it is sending the correct signal based on the verification beforehand, we know that the problem is in the software and we were able to fix that in the end.

## 3.3 Motor Subsystem

To verify whether our motor subsystem works, what we have done is artificially send BUS signals from a power source and also set the Tray ID to a specific ID by setting it to high and low as well. We connected all the high pins with jumper cables to the power source and all the low pins with a jumper cable to ground. With this, we were able to simulate the bus signals and tray IDs. For example, to check whether our motor subsystems work as intended, we tried to match the tray ID and BUS signal to be 1001. When the BUS signal and tray ID match, the motor activates. To check this, we also tried when the BUS signal and tray ID mismatches, and the motor does not turn on. Hence, it is working as intended. To ensure that it works completely, we tried it with all the possible 4 bit combinations, and the motor only turns on when the BUS signal and the tray ID match.

## 3.4 User Interface Subsystem

With the user interface subsystem, there are two devices that we have to check for, the RFID and the LCD. With the LCD, we want it to show the available products that users can dispense. To do this, we first try to establish a connection with the microcontroller and try to print “Hello World”. After a connection has been established and “Hello World” can be seen on the display, we then try to see if it can communicate with the database and print the items properly. To do this, we populate the database with data and try to see if the objects are printed onto the LCD screen. We were able to populate the data with 3 items and the LCD screen will display the items that the user is eligible for and the items that the machine has inventory on.

With the RFID system, we want to know if each user will be identified correctly. To do this, we first establish a connection between the RFID and the microcontroller. After a connection has been established, we then try to print out the ID of each card whenever it is tapped. Everytime a user taps a card, we check if the ID is constant and see if it is sent over to the database correctly. We found that the I-Card has some encryption issues so we decided to test and verify the RFID reader through other RFID compatible cards. All other cards that we tested send over their corresponding ID to the database correctly. The only cards that have their UID constantly changed is the I-Card, and we believe this is because there are some encryption protocols put into place. After having a user tap their card, we try to dispense a product and check with the database if it behaves accordingly. There are two scenarios that we tested. The first scenario is when a user successfully dispenses a product; the database should update the machine’s inventory and decrement the user’s quota. The second scenario is when a user dispenses a product, but the item fails to dispense; the database should update the inventory to zero and make sure to not decrement the user’s quota.

## 3.5 Power Subsystem

To verify whether our power subsystem is able to convert from a 120V wall power into a 5V power using a transfer, we used a multimeter to check the voltage that is being delivered by the transformer. We put one end to the output voltage of the transformer and the other end unto ground. We confirmed that the voltage output being read by the multimeter is indeed 5V.

After confirming that the voltage output is 5V, we used a multimeter again to check the voltage output after stepping it down. We connected one end of the multimeter on the output after the step-down voltage and the other end on ground. We confirmed that the voltage output is indeed 3.3V and we moved on to the other parts of the project.

## 3.6 Database Subsystem

There is one requirement to verify that the database subsystem works properly, which is that when a product is dispensed or indispensable, the databases be updated accordingly. To verify this requirement, an item is dispensed twice, once with the IR sensor being tripped and one without the IR sensor being tripped. When the item was dispensed while the IR sensor was tripped, a change could be seen in the inventory database that the stock of that item decremented by 1 and the quota of the user corresponding to the item decremented by 1. When the IR sensor was not tripped, a change could be seen in the inventory database where the stock for the given item was set to 0, thus demonstrating that the databases were updated correctly when a item is dispensed/indispensable.

# 4. Cost and Schedule

## 4.1 Cost Analysis

### 4.1.1 Labor

Assume that an electrical engineer responsible for designing and assembling the circuitry and electronics of this project is paid $45/hour in compensation. A time estimate for the construction and assembly of this implementation of this machine is 3 hours. The PCB design and assembly of this machine, which includes compatibility testing and simulations, will likely take 12 hours per partner. The software design component will likely take 48 hours to implement and debug, and integrating the microcontroller with the software will take 24 hours. It is expected that it should take around 2 hours of labor for the machine shop to create the housing for the dispensing machine; it is reasonable to assume that they are paid $40/hour in compensation.

### 4.1.2 Parts

| Item (linked) | Quantity | Cost (USD) |
|---|---|---|
| ESP32 MCU | 1 | 8.95 |
| RFID Module | 1 | 39.95 |
| Metal Pushbutton | 4 | 19.8 |
| 120V to 12 and 5V Transformer | 1 | 20 |
| 5V to 3.3V Stepper | 1 | 2.10 |
| 16x2 LCD Screen | 1 | 9.95 |
| Object Reflection Sensor | 1 | 2.95 |
| 12V DC Motor | 1 | 16.95 |
| Vending Machine Spirals | 1 | 8 |
| Spiral Adapter | 1 | 2 |
| MUX | 1 | 0.53 |
| Inverter | 1 | 1.32 |
| XOR Gate | 1 | 0.64 |
| 4-Input AND Gate | 1 | 0.37 |
| 2-Input AND Gate | 2 | 0.37 |

| Item (linked) | Quantity | Cost (USD) |
|---|---|---|
| ESP32 MCU | 1 | 8.95 |
| RFID Module | 1 | 39.95 |
| Metal Pushbutton | 4 | 19.8 |
| 120V to 12 and 5V Transformer | 1 | 20 |
| 5V to 3.3V Stepper | 1 | 2.10 |
| 16x2 LCD Screen | 1 | 9.95 |
| Object Reflection Sensor | 1 | 2.95 |
| 12V DC Motor | 1 | 16.95 |
| Vending Machine Spirals | 1 | 8 |
| Spiral Adapter | 1 | 2 |
| MUX | 1 | 0.53 |
| Inverter | 1 | 1.32 |
| OR Gate | 1 | 0.37 |
| **Purchasing Total** (assuming 10% tax) | | 134.25 (147.68) |
| **Labor** | | 4980 |
| **Grand Total** | | 5128 |

## 4.2 Schedule

| Week | Josh Leeman | Dylan Hartato | Matthew Chung |
|---|---|---|---|
| 1 | Design Document. Design the PCB for the Module. | Design Document. Design the PCB for the Control Module. | Design Document. Design the idea and skeleton for the backend software. |
| 2 | Design Document is Due. Get the PCB design approved so that it can start. Start the PCB design on KiCad. | | Design Document is Due. Fill out Google form to place an order, so that we can start early. |
| 4 | Test PCB functionality and of individual components. | PCB Soldering. | Complete Software Code to work with Backend. When finished, help assemble the housing and slots |
| 5 | Integration of hardware components, error testing. | Integration of hardware components with microcontroller | Integration of databases with microcontroller, error testing. |
| 6 | Assembly of final build and combine the hardware subsystem into a single packaging | Make the different subsystems work and debug the problems related to integration | Final software debugging, ensure compatibility of analog signals with microcontroller. |
| 7 | Finish up and make sure everything works for the Mock Demo. Try to keep on testing. This week is generally kept empty just in case something happens and extra time is needed. | Finish up and make sure everything works for the Mock Demo. Try to keep on testing. This week is generally kept empty just in case something happens and extra time is needed. | Finish up and make sure everything works for the Mock Demo. Try to keep on testing. This week is generally kept empty just in case something happens and extra time is needed. |
| 8 | Mock Demo | | |
| 9 | Final Demonstration. Work on presentation and Final Paper. | | |
| 10 | Final Demonstration and Final Paper Due | | |

# 5. Conclusion

## 5.1 Accomplishments

After working on this project throughout the semester, we have successfully completed all the high level requirements that we have set for ourselves. Although we did not end up using our PCB, we were able to mimic our PCB design onto the breadboard and make it work. Firstly, we successfully connected and integrated the RFID and LCD with the database and backend. We successfully used the RFID system to identify a user, and send that data over to the database. With the LCD, we got it to communicate with the database to display the available items depending on the information that it receives from the database. The server also worked as intended, as it keeps track of each user's data and inventory consistently during each GET and POST request. Second, we successfully implemented our tray PCB properly and it was working as intended. When the signal sent over by the BUS matches the tray's ID, it will activate the motor as intended. We tested the behavior of this PCB using different tray IDs and different BUS signals. We concluded that the PCB works as intended as the motor turns on only when the BUS signal and tray ID matches.

## 5.2 Uncertainties

With our project, we have three problems, which are the IR sensor, the University's I-Card, and the PCB's design. The IR sensor that we implemented in our project worked as intended; it will stop the motor from dispensing when the sensor is triggered. However, the IR sensor only has a working range of 1 cm. Because of this, we could not implement it based on our original design as a 1cm range will not work in a dispenser. We successfully implemented the functionality of it in our project, but we had to manually tap the sensor for it to trigger. Initially, we thought that by changing the resistor values we could increase the range of the IR sensor. We were only able to extend the range from 5 mm to 1cm.

Secondly, the University of Illinois's I-Card does not work as intended with the RFID reader. After running numerous tests, we noticed that the information being sent in the I-card is encrypted. The card's ID is always scrambled and random everything it comes into contact with the RFID reader. To test our theory, we tested it with numerous other cards that contain an RFID chip and it reads the data consistently every time. Other cards that we used will always send the same ID, but the University's I-card always has the numbers scrambled. As a result, we were unable to make it work with an I-card, but it works with any other RFID compatible cards.

Lastly, a problem that we encountered was in our PCB design. After soldering all our components onto the PCB, we tried to flash the MCU but it did not work. After going through various forums and asking around, we found that we needed to set PIN 3 in the ESP32 MCU to be High while we flash the code, then set it to low when it is done flashing. Our PCB design sets PIN 3 to an Active HIGH by connecting it to the power source. Because of this, our MCU is always in a flashing state and it can never exit that state. As a result, we cannot use our PCB in the final product.

## 5.3 Future Work

If this project is to be continued after the class, some next steps that we could do is to first fix the problems that we had. That is to first tackle the I-Card encryption issue. The I-Card encryption issue can be fixed by using a card swipe instead of a RFID reader. With a card swipe, the user's information can be parsed properly and it will not be encrypted. This will allow the program to take in the user's ID consistently. Another problem that we can improve upon is the IR sensor. As mentioned before the IR sensor only has a small working range. To fix this, a bigger and stronger IR sensor can be used for it to detect if an object has been dispensed. However, with a stronger IR sensor, it will take a larger footprint and it will cost more to implement as it is a more expensive part. Lastly, some future work that could be added unto this is to have a mobile application. The mobile application can be used to show where these machines are located and what stock each machine has. This can help fully integrate everything together and it can also help provide information to people on whether they can dispense a product they are looking for.

## 5.4 Ethical and Safety Considerations

Every piece of technology has its risks, and such risks can range from abuse of collected information or risk of injury to the user from unintentional misuse. While the implementation of our device requires us to collect some data regarding the user's vending history, none of the information should be considered sensitive. However, user information should not be divulged unless absolutely necessary to ensure privacy, and as such the user information database and inventory database should be implemented independently to ensure that those who have access to inventory are not able to access user information without proper credentials.

The primary safety concern is primarily electrical; the machine will house a 120v to 5v and 12v stepper, and a lower 5v to 3.3v stepper. To ensure that high voltage electrical hazards are minimized, the 120v stepper is enclosed and will be isolated from the majority of the electronics and human-to-machine contact points. In addition, the wattage of the whole machine is sufficiently low (~120W) such that the likelihood of a fire hazard from a malfunction of the transformer is very low.

Any wiring carrying the 12V voltage will run in the back of the machine from the transformers to the modules and then the motors, and thus the risk posed to the user is very small. The user interfaces are all going to be made from insulating material (such as plastics) to prevent any electrical injuries from occurring to the users.  The 5v and 3.3v electronics will be primarily housed on the PCB board, which itself is isolated from the buttons and screen that the user may touch; even so, the voltage is sufficiently low that it does not pose a significant hazard to the user.

In regards to the IEEE code of ethics, we are ensuring that we are going to follow the code of conduct, specifically in Article I, number 1 [1]. Our device is going to keep the privacy of others because we are going to only collect when a certain person has dispensed an item. Each user is going to be stored as an ID, not by their names, so user information is going to be ambiguous. Furthermore, the information is going to be held in the database, which is not accessible by anyone. Another article that we will follow is on the idea that we are not going to discriminate against others as stated in  Article II number 7 [1]. We are following this by ensuring that we are not going to purposely hand out more supplies to certain individuals. Since we are ensuring that everyone will have a quota on supplies, no one is going to have more than the others, at least purposefully.

## References

[1] “IEEE code of ethics,” IEEE, Jun-2020. [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html. [Accessed: 10-Feb-2022].
[2] “CC3220R, CC3220S, and CC3220SF SimpleLinkTM Wi-Fi® Single-Chip Wireless MCU Solutions,” Texas Instruments, May-2021. [Online]. Available: https://www.ti.com/lit/ds/symlink/cc3220sf.pdf?ts=1644333019728&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FCC3220SF . [Accessed: 17-Feb-2022].

[3] “Miniature Reflective Object Sensor” ON Semiconductors, 2019. [Online] Available: https://www.onsemi.com/pdf/datasheet/qre1113-d.pdf . [Accessed: 17-Feb-2022].

[4] “LOW DROP FIXED AND ADJUSTABLE POSITIVE VOLTAGE REGULATORS” STMicroelectronics, 2005. Available: https://www.sparkfun.com/datasheets/Components/LD1117V33.pdf. [Accessed: 17-Feb-2022].

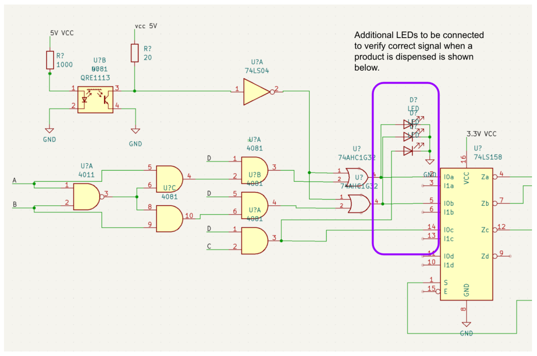
[5] M. #570786, “Sparkfun Line Sensor Breakout - QRE1113 (analog),” ROB-09453 - SparkFun Electronics. [Online]. Available: https://www.sparkfun.com/products/9453. [Accessed: 24-Feb-2022].

# Appendix A: Verification Table

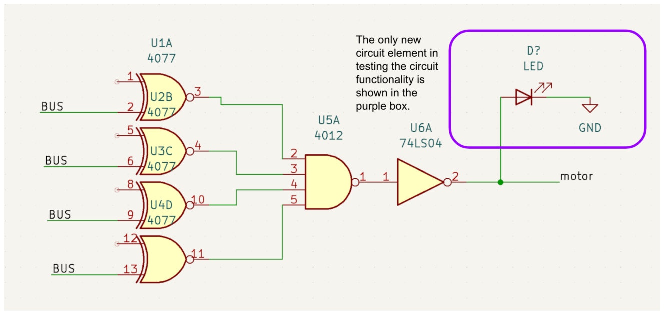
**Control System**

| Requirement | Verification | Verification Status |
|---|---|---|
| 1. The correct BUS signal should be emitted when a product is chosen; | 1. Connecting leads from the BUS to a series of LCDs on a breadboard and configuring the LCD's the different motors for the different products and testing if sending the correct signal will light up the LCD corresponding to the correct motor. | Yes |
| 2. The correct products should be displayed to the LCD screen; | 2. Tapping an ID card and adding various items to the inventory (both dispensable and indispensable) verifying that the correct product is shown on the LCD screen. | Yes |
| 3. An RFID signal should be received and correctly interpreted by the microcontroller. | 3. Scan an id-card through the RFID signal, dispensing an object and verifying that the database for the correct user is being updated. | Yes |
| 4. When a product is dispensed or indispensable, the microcontroller should update the inventory and user database correspondingly. | 4. Put in a fake entry into the inventory database assigned to a non-existent tray. An attempt to dispense that item should fail after three attempts, and the inventory should be cleared. | Yes |

**Sensing Subsystem**

| Requirement | Verification | Verification Status |
|---|---|---|
| 1. When motion is detected by the sensor, an active signal should be sent to the microcontroller. | 1. Try to dispense an object and verify that the inventory of the object is being updated. To test the circuitry if the correct signal is being dispensed, consider below; attach 3 LCDs to the input of the MUX as shown. If the correct signal is being transmitted, the LCD should signal 0 1 1 from top to bottom, where 0 is "off" and 1 is "on."   5V VCC<br>vcc 5V<br>R? 1000<br>R? 20<br>U?B 4081<br>QRE1113<br>GND<br>GND<br>U?A 74LS04<br>Additional LEDs to be connected to verify correct signal when a product is dispensed is shown below.<br>D? LED<br>3.3V VCC<br>U? 74LS158<br>U?A 4011<br>U?C 4081<br>U?A 4081<br>U?B 4081<br>U? 74AHC1G32<br>A<br>B<br>D<br>C<br>I0a<br>I1a<br>I0b<br>I1b<br>I0c<br>I1c<br>I0d<br>I1d<br>S<br>E<br>VCC<br>GND<br>Za<br>Zb<br>Zc<br>Zd | Yes |

**Motor Subsystem**

| Requirement | Verification | Verification Status |
|---|---|---|
| 1. When the correct signal is delivered across the BUS, the motor should activate for exactly one cycle on a rising edge. | 1. One can artificially send a BUS signal by connecting leads to the BUS entry ports of the dispenser module and either connecting it to 3V power or to ground to represent a "1" or "0" bit. Then, attach an LCD to the wires connected to the motor; if the LCD illuminates upon the correct signal being transmitted across the BUS, the device is deemed to be functional.   U1A 4077<br>U2B 4077<br>U3C 4077<br>U4D 4077<br>BUS<br>BUS<br>BUS<br>BUS<br>U5A 4012<br>U6A 74LS04<br>The only new circuit element in testing the circuit functionality is shown in the purple box.<br>D? LED<br>GND<br>motor | Yes |

**User Interface Subsystem**

| Requirement | Verification | Verification Status |
|---|---|---|
| 1. The correct products should be displayed to the LCD screen | 1. Adding products to the inventory and verifying that the correct product is shown on the LCD screen | Yes |
| 2. An RFID signal should be received and correctly interpreted by the microcontroller | 2. Scan an id-card through the RFID signal, try to dispense an object through that account and verify through the database if it labels that account as having dispense the object. | Yes |

**Power Subsystem**

| Requirement | Verification | Verification Status |
|---|---|---|
| 1. The power system must take in 120V and output a 5V DC current to the PCB | 1. Use a voltmeter to detect the output voltage from the power system and verify if it is 5V | Yes |
| 2. The power system must transform the 5V DC current and step it down to a 3.3V DC current | 2. Use a voltmeter to detect the output voltage from the step-down power converter and verify that it is 3.3V | Yes |

**Database Subsystem**

| Requirement | Verification | Verification Status |
|---|---|---|
| 1. When a product is dispensed or indispensable, the microcontroller should update the inventory and user database correspondingly. | 1. Scan an id-card, try to dispense an object and see if the database gets updated correctly. | Yes |

# Appendix B: Software Charts

## Firmware Flowchart

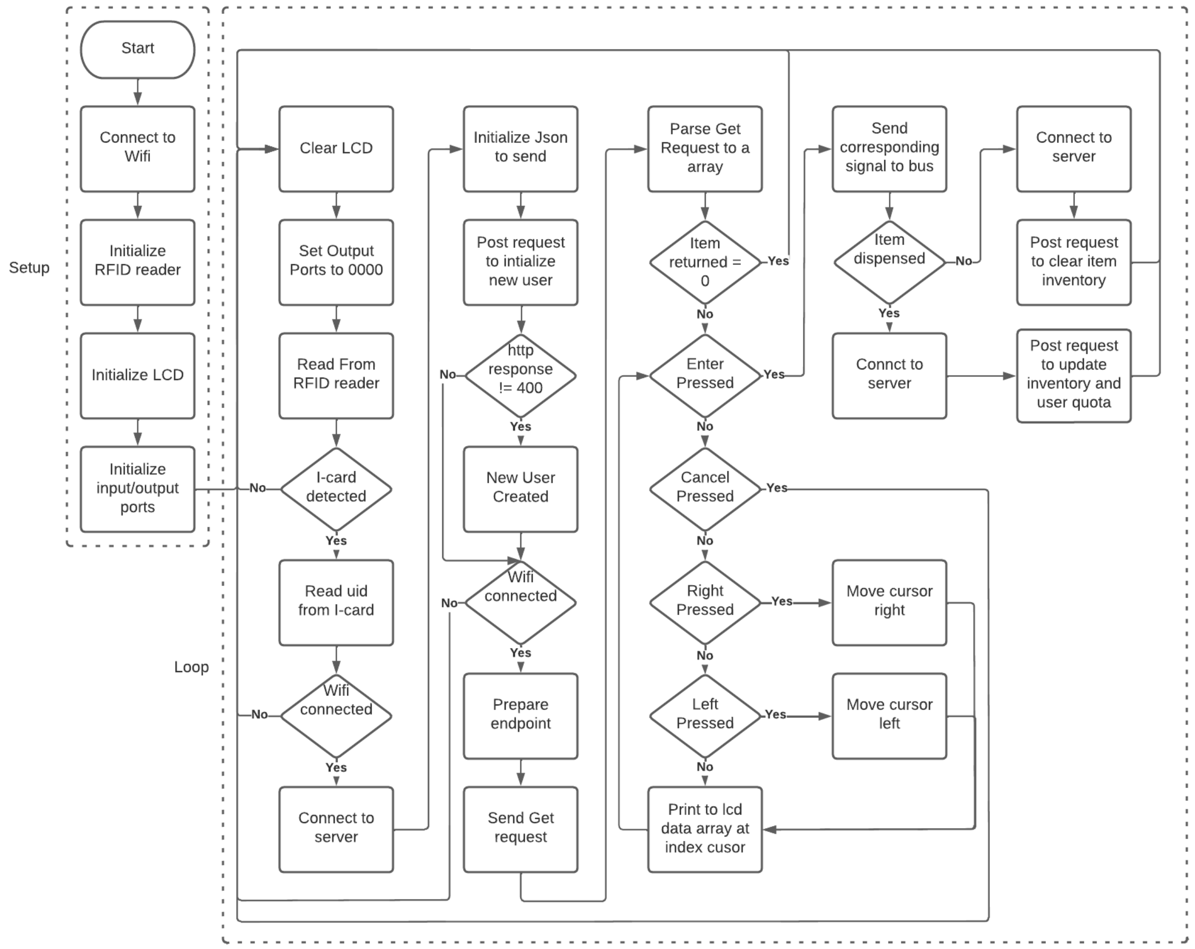


Figure 11: Circuit Diagram of the Dispenser Module

## Database Flowchart

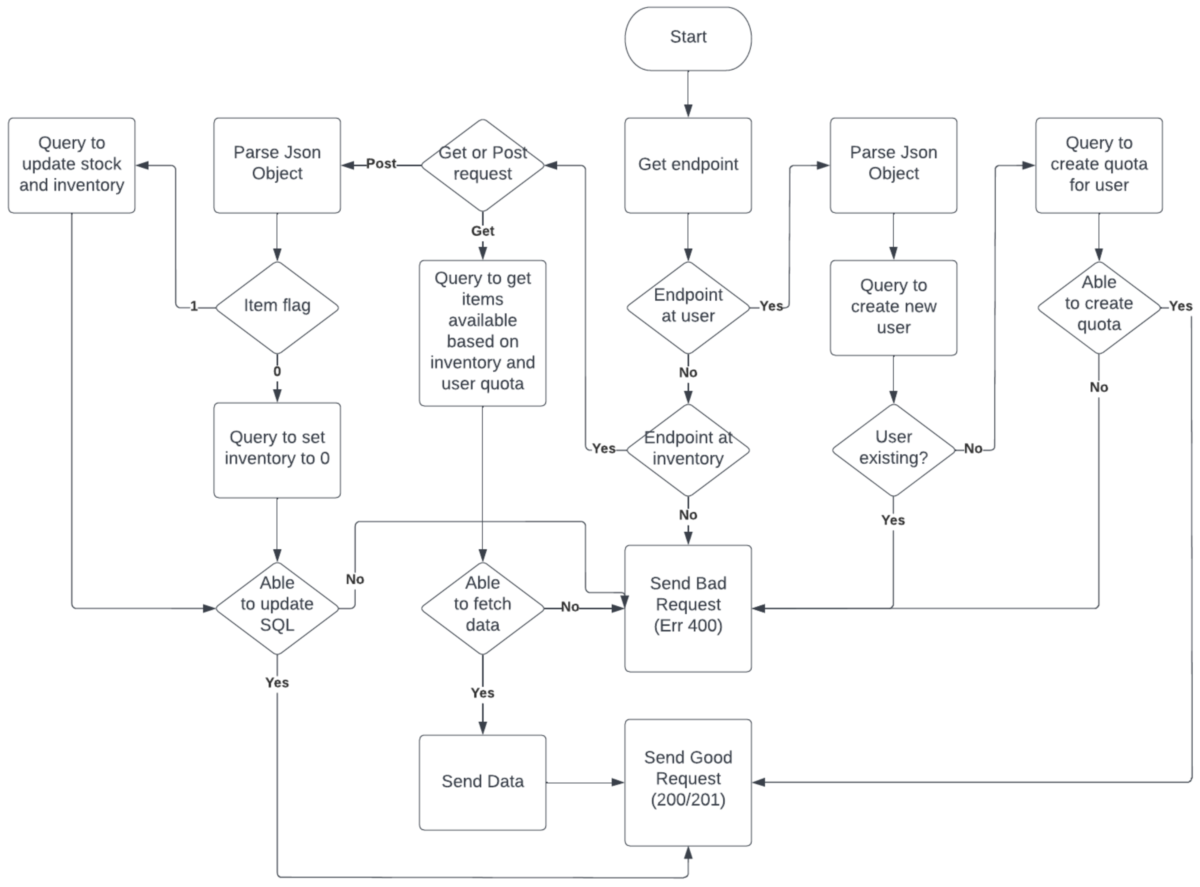


Figure 12: Circuit Diagram of the Dispenser Module

## Database Table

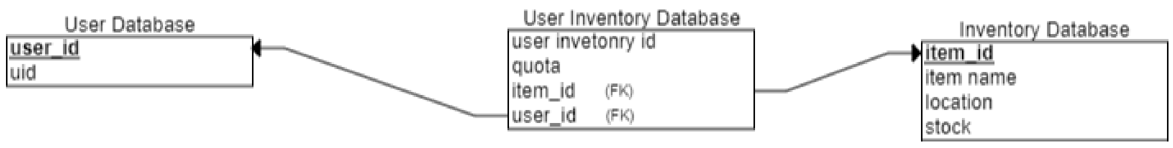


Figure 13: Circuit Diagram of the Dispenser Module

## Appendix C: Firmware Code

Code to firmware: https://github.com/chunghwaa/Medical-Dispenser-Kit.git

```
#include <WiFi.h>
#include <HTTPClient.h>
#include <ArduinoJson.h>
#include <StreamUtils.h>
#include <LiquidCrystal_I2C.h>
#include <Wire.h>
#include <SPI.h>
#include <Adafruit_PN532.h>

#define PN532_SCK  (4)
#define PN532_MOSI (17)
#define PN532_SS   (5)
#define PN532_MISO (16)
#define PN532_IRQ   (2)
#define PN532_RESET (3)

int leftbit = 19;
int midbit = 18;
int rightbit = 15;
int bus1 = 13;
int bus2 = 27;
int bus3 = 14;
int bus4 = 12;

Adafruit_PN532 nfc(PN532_SCK, PN532_MISO, PN532_MOSI, PN532_SS);

int lcdColumns = 16;
int lcdRows = 2;

LiquidCrystal_I2C lcd(0x27, lcdColumns, lcdRows);

const char* ssid = "**********";
const char* password = "**********";

const char* serverName = "https://ec-board.herokuapp.com/user";

String geti_path = "https://ec-board.herokuapp.com/inventory?location=eceb&uid=";
```

```
void setup() {
	Serial.begin(9600);

	WiFi.begin(ssid, password);
	Serial.println("Connecting");

	while(WiFi.status() != WL_CONNECTED) {
		delay(500);
		Serial.print(".");
	 }

	Serial.println("");
	Serial.print("Connected to WiFi network with IP Address: ");
	Serial.println(WiFi.localIP());

	nfc.begin();
	uint32_t versiondata = nfc.getFirmwareVersion();

	if (! versiondata) {
		Serial.print("Didn't find PN53x board");
		while (1); // halt
	 }

	Serial.print("Found chip PN5"); Serial.println((versiondata>>24) & 0xFF, HEX);
	Serial.print("Firmware ver. "); Serial.print((versiondata>>16) & 0xFF, DEC);
	Serial.print('.'); Serial.println((versiondata>>8) & 0xFF, DEC);

	nfc.SAMConfig();

	lcd.init();
	lcd.backlight();

	pinMode(leftbit, INPUT);
	pinMode(midbit, INPUT);
	pinMode(rightbit, INPUT);
	pinMode(bus1, OUTPUT);
	pinMode(bus2, OUTPUT);
	pinMode(bus3, OUTPUT);
	pinMode(bus4, OUTPUT);

	digitalWrite(bus1, LOW);
	digitalWrite(bus2, LOW);
	digitalWrite(bus3, LOW);
	digitalWrite(bus4, LOW);
```

```
}
void loop() {
	lcd.clear();
	lcd.setCursor(0, 0);
	lcd.print("WELCOME");

	digitalWrite(bus1, LOW);
	digitalWrite(bus2, LOW);
	digitalWrite(bus3, LOW);
	digitalWrite(bus4, LOW);

	uint8_t success;
	uint8_t uid[] = { 0, 0, 0, 0, 0, 0, 0 };
	uint8_t uidLength;

	Serial.println("Waiting for a RFID Card");
	success = nfc.readPassiveTargetID(PN532_MIFARE_ISO14443A, uid, &uidLength);

	if (success) {
		Serial.println("Found an ISO14443A card");
		Serial.print("  UID Length: ");
		Serial.print(uidLength, DEC);Serial.println(" bytes");
		Serial.print("  UID Value: ");
		nfc.PrintHex(uid, uidLength);
		Serial.println("");

		int uin = uid;
		Serial.print(uin);

		Serial.println("Posting JSON data to server...");

		if (WiFi.status()== WL_CONNECTED) {

			HTTPClient http;

			http.begin("https://ec-board.herokuapp.com/user");
			http.addHeader("Content-Type", "application/json");

			StaticJsonDocument<200> doc;
			doc["uid"] = uin;

			String requestBody;
			serializeJson(doc, requestBody);

			int httpResponseCode = http.POST(requestBody);
```

```
        if(httpResponseCode>0){

                String response = http.getString();
                Serial.println(httpResponseCode);
                Serial.println(response);

        }
        else {

                Serial.printf("Error");
        }

}

else{
        break;
}

String dataArr[20];
int counter = 0;

Serial.println("Getting JSON data from server...");

if (WiFi.status() == WL_CONNECTED) {
        HTTPClient http;

        Serial.println("getting inventory data");
        geti_path = geti_path + uin;

        Serial.println(geti_path);
        http.begin(geti_path.c_str());

        int httpResponseCode = http.GET();

        if(httpResponseCode>0){

                String response = http.getString();

                Serial.println(httpResponseCode);
                Serial.println(response);

                int flag = 0;
                String temp = "";
                for (int i = 0; i < response.length(); i++){
                    char c = response[i];
                    if (c == ':'){
                        flag = 1;
                        continue;
                    }

                    if (flag == 1){
                        if (c == '}'){
                            temp[temp.length() - 1] = '\0';
                            temp.remove(0,1);
                            Serial.println(temp);
                            flag = 0;
                            dataArr[counter] = temp;
                            counter += 1;
                            temp = "";
                        }

                        else{
                            temp += c;
                        }
                    }
                }
            }

            else {
                Serial.printf("Error");
            }
        }

        else {
            Serial.println("WiFi Disconnected");
            break;
        }

        if (counter == 0){
            lcd.print("No Valid Items");
            return;
        }

        String command = "";
        int item_chosen = 0;
```

```
while(1){
	command = "";

	if (digitalRead(leftbit) == LOW){
		if (digitalRead(midbit) == LOW){
			if (digitalRead(rightbit) == HIGH){
				command = "enter";
			}
		}
		else{
			if (digitalRead(rightbit) == LOW){
				command = "right";
			}
		}
	}
	else{
		if (digitalRead(midbit) == LOW){
			if (digitalRead(rightbit) == LOW){
				command = "left";
			}
		}
		else{
			if (digitalRead(rightbit) == HIGH){
				command = "cancel";
			}
		}
	}

	lcd.setCursor(0, 0);
	lcd.clear();
	lcd.print(dataArr[item_chosen]);

	if (command == "cancel"){
		return;
	}
	else if (command == "enter"){
		break;
	}
	else if (command == "left"){
		if (item_chosen > 0){
		item_chosen -= 1;
		}
	}
```

```
        else if (command == "right"){
                if (item_chosen < counter -1 ){
                        item_chosen += 1;
                }
        }

        delay(100);
}

lcd.clear();
lcd.print("Dispensing");

Serial.println(dataArr[item_chosen]);

int item_flag = 0;
unsigned long startTime;

for (int i = 0; i < 3; i++){
        startTime = millis();
        if (dataArr[item_chosen] == "mask"){
                digitalWrite(bus1, HIGH);
                digitalWrite(bus2, LOW);
                digitalWrite(bus3, LOW);
                digitalWrite(bus4, HIGH);
        }
        else if (dataArr[item_chosen] == "advil"){
                digitalWrite(bus1, LOW);
                digitalWrite(bus2, LOW);
                digitalWrite(bus3, LOW);
                digitalWrite(bus4, HIGH);
        }
        else if (dataArr[item_chosen] == "pill"){
                digitalWrite(bus1, LOW);
                digitalWrite(bus2, LOW);
                digitalWrite(bus3, HIGH);
                digitalWrite(bus4, HIGH);
        }

        while(millis() - startTime <= 5000){
                        if ((digitalRead(leftbit) == HIGH && digitalRead(midbit)
                            == HIGH) && digitalRead(rightbit) == LOW){
                                item_flag = 1;
                                break;
                        }
        }

        if (item_flag == 1){
                Serial.println("done");
                break;
        }

        startTime = millis();

        digitalWrite(bus1, LOW);
        digitalWrite(bus2, LOW);
        digitalWrite(bus3, LOW);
        digitalWrite(bus4, LOW);

        while(millis() - startTime <= 2000){
                if ((digitalRead(leftbit) == HIGH && digitalRead(midbit) ==
                    HIGH) && digitalRead(rightbit) == LOW) {
                        item_flag = 1;
                        break;
                }
        }

        if (item_flag == 1){
                Serial.println("done");
                break;
        }
}



lcd.clear();
if (item_flag == 1){
        lcd.print("Item Dispensed");
}
else{
        lcd.print("Dispense Fail");
}
delay(1000);



Serial.println("Posting JSON data to server...");
```

```
        if (WiFi.status()== WL_CONNECTED) {

            HTTPClient http;

            http.begin("https://ec-board.herokuapp.com/inventory");
            http.addHeader("Content-Type", "application/json");

            StaticJsonDocument<200> doc;

            doc["uid"] = uin;
            doc["flag"] = item_flag;

            if (dataArr[item_chosen] == "pill"){
                doc["item_name"] = "pill";
            }
            else if (dataArr[item_chosen] == "advil"){
                doc["item_name"] = "advil";
            }
            else if (dataArr[item_chosen] == "mask"){
                doc["item_name"] = "mask";
            }

            String requestBody;
            serializeJson(doc, requestBody);

            int httpResponseCode = http.POST(requestBody);

            if(httpResponseCode>0){

                String response = http.getString();

                Serial.println(httpResponseCode);
                Serial.println(response);

            }
            else {
                Serial.printf("Error");

            }
        }
    }
}
```

## Appendix D: Database Code

Code to database code: https://github.com/chunghwaa/Medical-Dispenser-Kit.git

### Code for user endpoint:

```
const router = require("express").Router();
const client = require("../db.js");

router.post("/", async (req, res) => {
        const uid = req.body.uid;
        console.log(uid);

        try {
                const new_user = await client.query(
                 "INSERT INTO UserData(uid) VALUES($1) ON CONFLICT DO NOTHING
                RETURNING id",
                [uid]
                 );

                let id;

                if (new_user.rows.length > 0) {
                        id = new_user.rows[0].id;
                } else {
                        const user_id = await client.query(
                        "SELECT id FROM userdata WHERE uid=$1",
                        [uid]
                        );
                        id = user_id.rows[0].id;
                }

                const inventory_items = await client.query(
                "SELECT id FROM inventory LIMIT 1;"
                );

                console.log(inventory_items.rows);

                 const new_useritem = await client.query(
                "INSERT INTO usersinventory(inventory_id, quota, user_id)
                VALUES($2, 2, $1), ($3, 2, $1), ($4, 2, $1)",
                [
                id,
                inventory_items.rows[0].id,
			inventory_items.rows[0].id + 1,
			inventory_items.rows[0].id + 2,
			]
			);

			res.status(201).send("Created User");
		} catch (err) {
			res.status(400).send("Bad Request");
			console.log(err.message);
		}
});

module.exports = router;
```

**Code for inventory endpoint:**

```
const router = require("express").Router();
const client = require("../db.js");

router.get("/", async (req, res) => {
        const location = req.query.location;
        const uid = req.query.uid;

        try {
                const user_id = await client.query(
                "SELECT id FROM userdata WHERE uid=$1;",
                [uid]
                );
                const inventory = await client.query(
                "SELECT item_name FROM inventory WHERE id IN (SELECT inventory_id
                FROM UsersInventory WHERE quota > 0 AND user_id=$2) AND location=$1
                AND stock > 0;",
                [location, user_id.rows[0].id]
                );
                res.status(200).send(inventory.rows);
        } catch (err) {
                res.status(400).send("Bad Request");
                 console.log(err.message);
         }
});

router.post("/", async (req, res) => {
        const uid = req.body.uid;
        const item_name = req.body.item_name;
        const flag = req.body.flag;
        console.log(uid);
        console.log(item_name);
        console.log(flag);
```

```
if (flag == 1){
        try {
                const user_id = await client.query(
                "SELECT id FROM userdata WHERE uid=$1;",
                [uid]
                );

                const item_taken = await client.query(
                 "UPDATE inventory SET stock = ( CASE WHEN (stock > 0) THEN
                (stock - 1) ELSE 0 END) WHERE item_name = $1 RETURNING id;",
                [item_name]
                );

                const inven_id = item_taken.rows[0].id;
                const u_id = user_id.rows[0].id;

                const quota_updated = await client.query(
                "UPDATE usersinventory SET quota = ( CASE WHEN (quota > 0) THEN
                (quota - 1) ELSE 0 END) WHERE inventory_id = $1 AND user_id = $2",
                [inven_id, u_id]
                );

                res.status(201).send("Updated Inventory");
        } catch (err) {
                res.status(400).send("Bad Request");
                console.log(err.message);
        }
}
else{
        try {
                const item_taken = await client.query(
                "UPDATE inventory SET stock = 0 WHERE item_name = $1;",
                [item_name]
                );
```

```
				res.status(201).send("Updated Inventory");
			} catch (err) {
				res.status(400).send("Bad Request");
				console.log(err.message);
			}
		}
});

module.exports = router;
```