Automatic Toothpaste Dispenser

By
Renjie Fan
Yanbo Chen
Haoyu Tian

ECE445 Final Report, Spring 2019
Professor: Jing Jiang
TA: Soumithri Bala

Apr 30, 2019

Team No.38

Abstract

This report documents the design, implementation, testing, and verification of our Automatic Toothpaste Dispenser project. The blueprint of our project is to build a product that could replace those current toothpaste dispensers in the market. The prototype we finally built has the following functions: automatically dispensing and refilling toothpaste, identifying users through RFID tag, and uploading data to Android Application.

Contents

1 Introduction

1.1 Purpose

Several kinds of manually operated toothpaste dispensers have been designed by different manufacturers such as iLife Tech and ECOCO. In general, the mechanism applied is based on the action that the user pushes the trigger inside the dispenser with the toothbrush. This mechanism seems simple and user-friendly at the first look. Unfortunately, the trigger will soon be covered by dry toothpaste. This major problem is widely reflected in customer reviews. For example, a user named "William J Leep" said that "the dispenser itself is fairly well made and easy to use. But, it dispenses too much and misses the brush 1/2 of the time"[1]. In addition, these manual dispensers do not have any mechanism to control the amount of toothpaste coming out of it and functionality to record the usage of toothpaste.

To tackle this problem, we plan to design and implement an automatic toothpaste dispenser based on the CYBLE-214015-01 EZ-BLE™ Creator Module, which supports Bluetooth Low-Energy(BLE) wireless communication, Radio Frequency Identification, and Android application with Bluetooth functionality, to provide a solution for the problem stated above.

1.2 Functionality

Our project has three main functionalities: User recognition, Automatic Dispensing Mechanism, and Smartphone Interaction.

User Recognition: The RFID reader of our automatic toothpaste dispenser is able to identify 3 different users through 3 default RFID tags attached on toothbrushes and dispense a certain amount of toothpaste which can be configured on our Android application by users.

Automatic Dispensing Mechanism: After the RFID tag attached on a toothbrush is detected by the RFID reader, the mechanical unit can accurately dispense 0.25mL(low) or 0.5mL(high) of toothpaste.

Smartphone Interaction: Our Android application allows users to choose between the high and low amount of toothpaste and save their choices on the Android application. It can also retrieve the amount of toothpaste used by different RFIDs from the dispenser and display them as a chart so that parents can use it to monitor whether the children are using their toothpaste with a correct amount.

1.3 Subsystem Overview

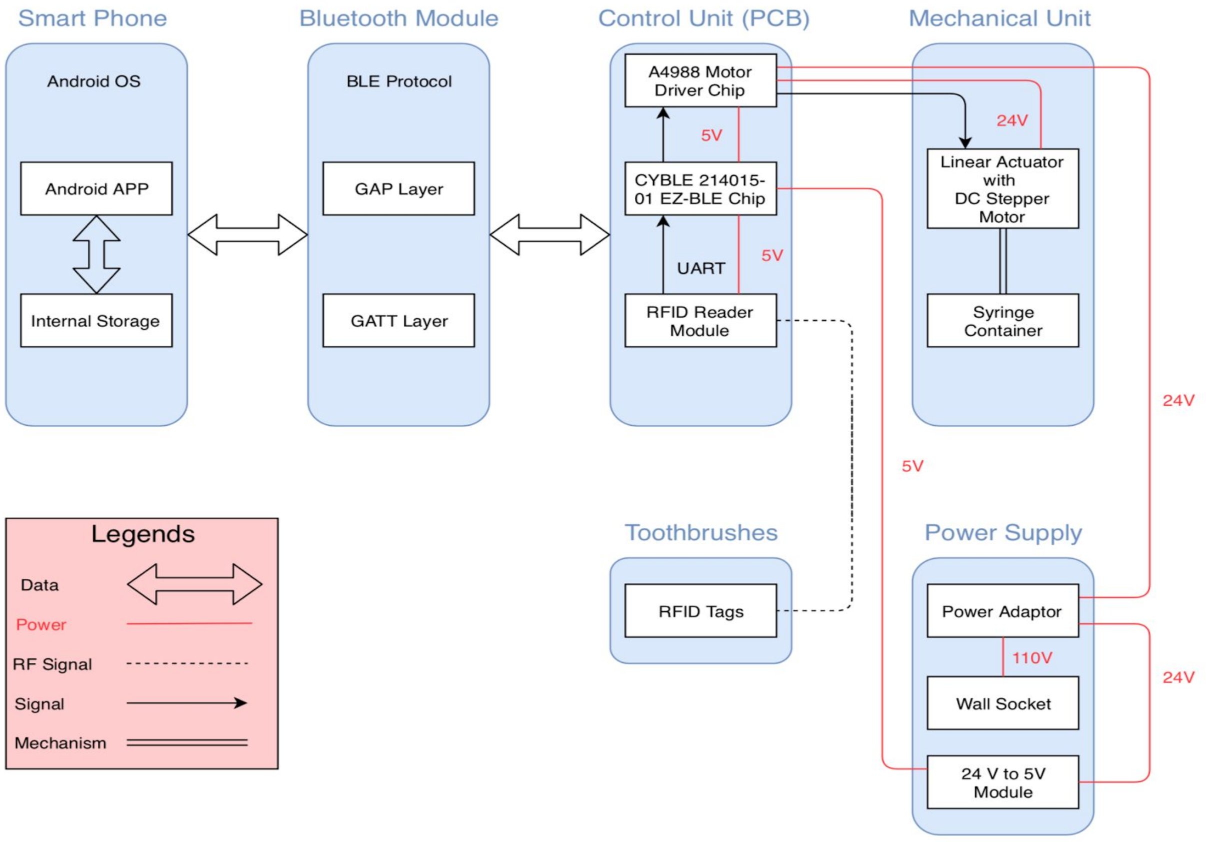


Figure 1: Block Diagram

Our design is divided into six components as shown in Figure 1. The control unit is the center of our project and it controls many core process: read RFID Tag information and identify different users, send PWM signal to the stepper motor driver and store the user's data at local storage. The mechanical unit will receive PWM signal from the control unit and then the stepper motor driver can process the signal to drive the motor in the linear actuator to dispense toothpaste from the syringe container or draw toothpaste from the toothpaste tube to refill the syringe. The power supply will power the whole system by connecting to a wall socket. The smartphone and the Bluetooth module will work together to send signals to the control unit to configure different users' desired amount of toothpaste or send a refill request to the control unit. It will also retrieve users' usage data from the control unit when the user wants to update the user usage on the application. The RFID tags in the toothbrushes module are used to distinguish the users.

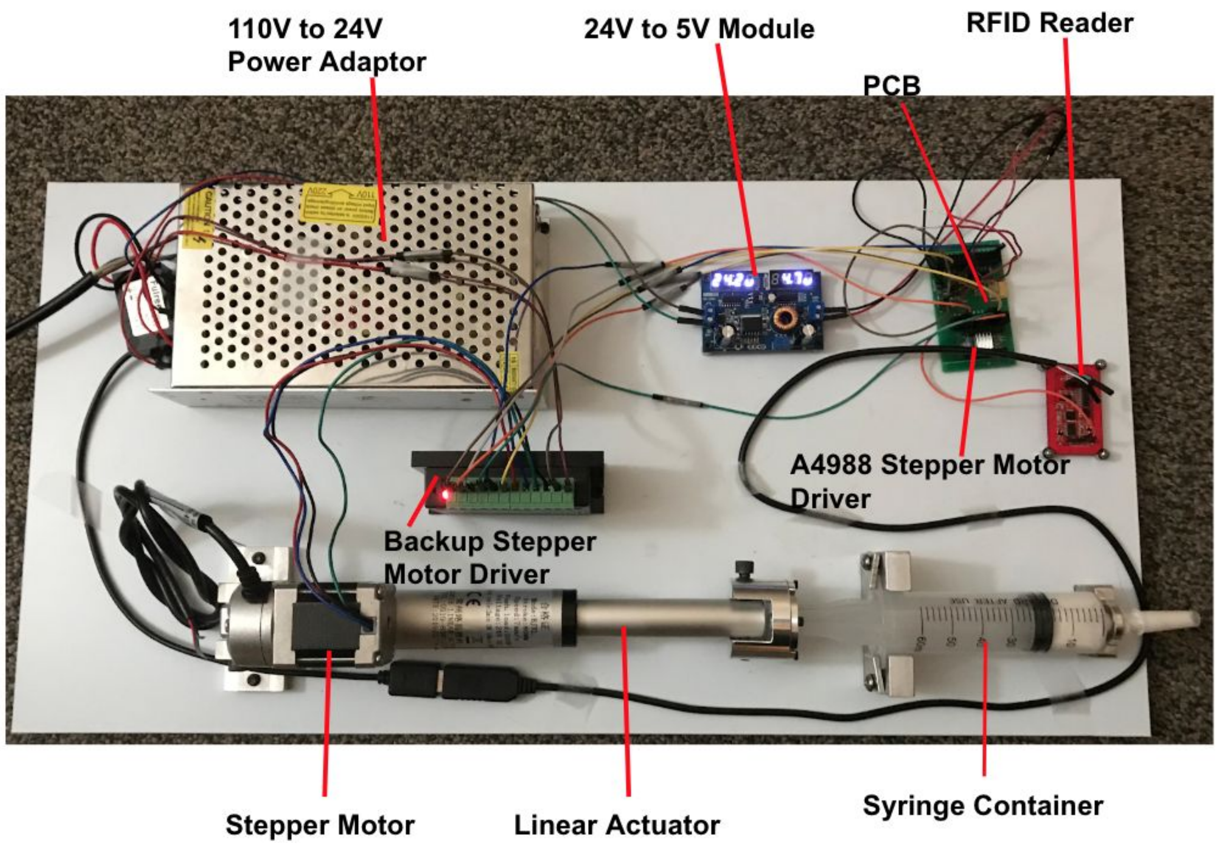


Figure 2: Physical Overview

Figure 2 is the physical overview of our project and each component is labeled on the graph. Our dispenser prototype is easy to use. Once the user put the toothpaste tube at the exit of the syringe and press refill button on our Android application, the linear actuator will move backward to draw toothpaste from the toothpaste tube into the syringe container. During the dispensing process, the linear actuator will move forward and toothpaste will be pushed out of the syringe container.

2 Design

2.1 Control Unit

Our control unit includes CYBLE-214015-01 Microcontroller chip sponsored by Cypress Semiconductor, A4988 Stepper Motor Driver Chip, and these two parts are directly integrated on our PCB. RFID Reader module was bought online and it is able to communicate with our microcontroller chip through UART protocol.

2.1.1 CYBLE-214015-01 Microcontroller

Schematic

Since we are using CYBLE-214015-01 microcontroller sponsored by Cypress Semiconductor, we used their supporting software, PSoC Creator to program the microcontroller.

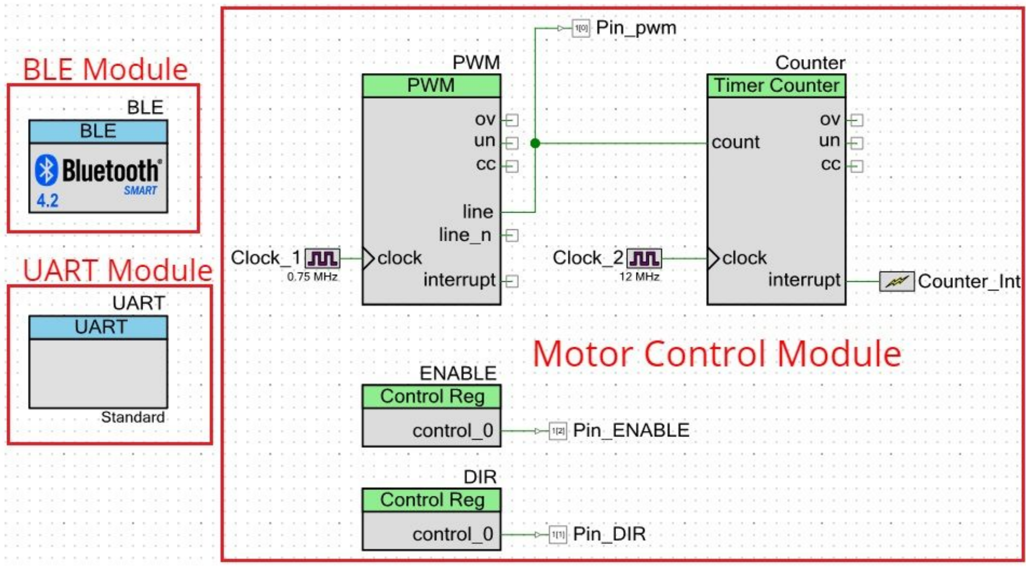


Figure 3: Schematic on PSoC Creator

Figure 3 above is the overall schematic for the microcontroller. The schematic can be roughly divided into three sections: BLE module, UART module, and motor control module.

BLE Module

BLE is a wireless communication protocol defined by Bluetooth SIG. It has a protocol stack to handle the data transmission as shown in figure 4.

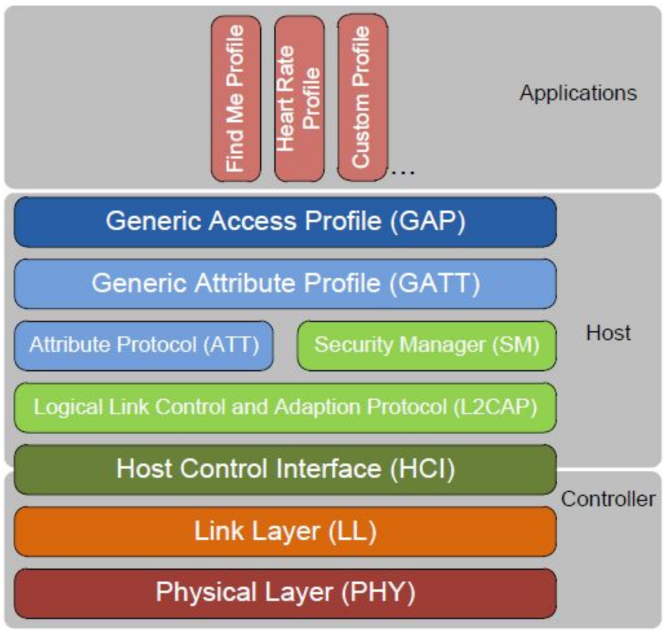


Figure 4: BLE Protocol Stack

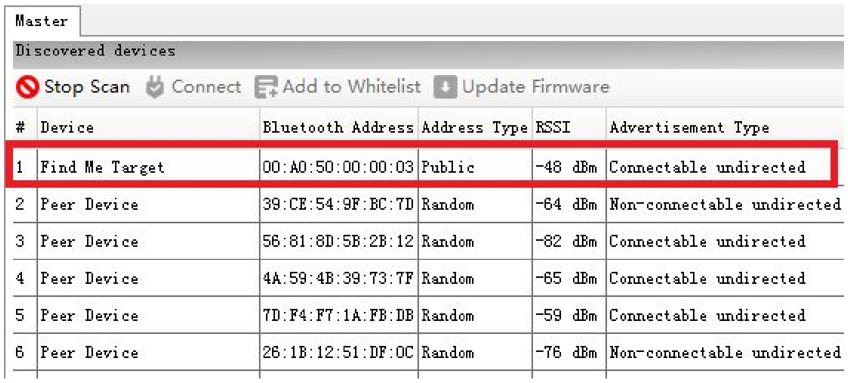
In our project, CYBLE-214015-01 has already provided a BLE stack for us. In a BLE protocol stack, GAP defines whether the device role is slave or master and GATT defines the methods to access the data defined in ATT.

Specifically, the microcontroller acts as a GAP peripheral which advertises itself to the Android smartphone, the GAP central. In addition, the microcontroller acts as a GATT server which contains toothpaste usage and settings and user's Android smartphone acts as a GATT client which reads data from the microcontroller.

	GAP	GATT
Android Phone	Central	Client
CYBLE-214015-01	Peripheral	Server (Database)

Table 1: GAP & GATT Device Role

After we established the BLE module, we tested with BLE connection by using a BLE dongle as the GAP central and an evaluation board(same chip as our PCB) as the GAP peripheral. As figure 5 shown, our evaluation board with a profile named "Find Me Target" is found by the dongle.


Master

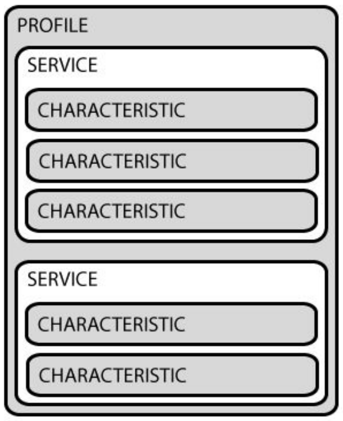
Discovered devices

Stop Scan | Connect | Add to Whitelist | Update Firmware

#	Device	Bluetooth Address	Address Type	RSSI	Advertisement Type
1	Find Me Target	00:A0:50:00:00:03	Public	-48 dBm	Connectable undirected
2	Peer Device	39:CE:54:9F:BC:7D	Random	-64 dBm	Non-connectable undirected
3	Peer Device	56:81:8D:5B:2B:12	Random	-82 dBm	Connectable undirected
4	Peer Device	4A:59:4B:39:73:7F	Random	-65 dBm	Connectable undirected
5	Peer Device	7D:F4:F7:1A:FB:DB	Random	-59 dBm	Connectable undirected
6	Peer Device	26:1B:12:51:DF:0C	Random	-76 dBm	Non-connectable undirected

Figure 5: Chip Advertisement Simulation

To provide data transmission between devices, GATT transactions are based on nested objects called Profile, Service, and Characteristics shown in figure 6.



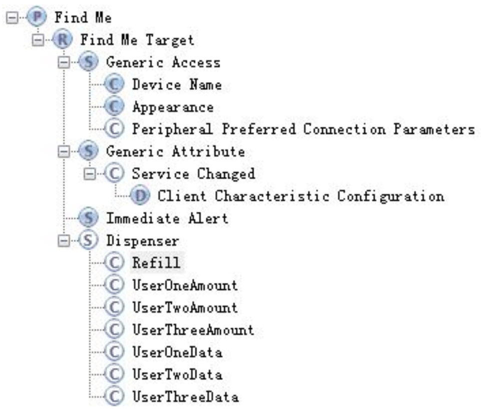
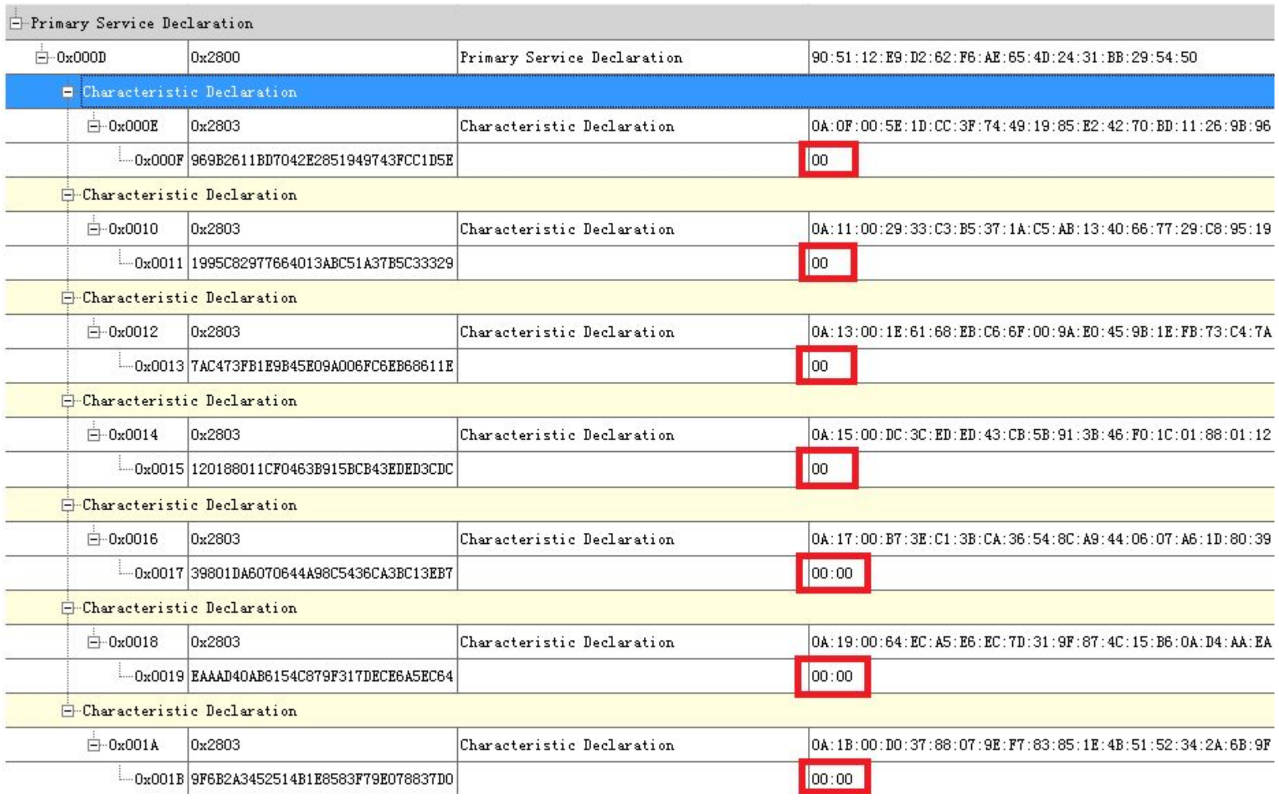

```
P Find Me
  R Find Me Target
    S Generic Access
      C Device Name
      C Appearance
      C Peripheral Preferred Connection Parameters
    S Generic Attribute
      C Service Changed
        D Client Characteristic Configuration
    S Immediate Alert
    S Dispenser
      C Refill
      C UserOneAmount
      C UserTwoAmount
      C UserThreeAmount
      C UserOneData
      C UserTwoData
      C UserThreeData
```

Figure 6: GATT Transaction Structure

Figure 7: Our BLE Profile

For the project, we defined our own BLE profile named as "Find Me" shown in figure 7. Under the service "Dispenser", we have seven characteristics. The characteristic "Refill" is used to indicate whether the refilling is processing: 1 means that the dispenser is refilling and 0 means that the dispenser is not refilling. The next three characteristics are used to indicate each user's toothpaste amount setting: 1 stands for the high amount and 0 stands for the low amount. The last three characteristics are three arrays of length two that used to store each user’s toothpaste usage: index 0 stands for the number of times that this user has been dispensed with a low amount and index 1 stands for the number of times that this user has been dispensed with a low amount.



Primary Service Declaration			
0x000D	0x2800	Primary Service Declaration	90:51:12:E9:D2:62:F6:AE:65:4D:24:31:BB:29:54:50
Characteristic Declaration			
0x000E	0x2803	Characteristic Declaration	0A:0F:00:5E:1D:CC:3F:74:49:19:85:E2:42:70:BD:11:26:9B:96
0x000F	969B2611BD7042E2851949743FCC1D5E		00
Characteristic Declaration			
0x0010	0x2803	Characteristic Declaration	0A:11:00:29:33:C3:B5:37:1A:C5:AB:13:40:66:77:29:C8:95:19
0x0011	1995C82977664013ABC51A37B5C33329		00
Characteristic Declaration			
0x0012	0x2803	Characteristic Declaration	0A:13:00:1E:61:68:EB:C6:6F:00:9A:E0:45:9B:1E:FB:73:C4:7A
0x0013	7AC473FB1E9B45E09A006FC6EB68611E		00
Characteristic Declaration			
0x0014	0x2803	Characteristic Declaration	0A:15:00:DC:3C:ED:ED:43:CB:5B:91:3B:46:F0:1C:01:88:01:12
0x0015	120188011CF0463B915BCB43EDED3CDC		00
Characteristic Declaration			
0x0016	0x2803	Characteristic Declaration	0A:17:00:B7:3E:C1:3B:CA:36:54:8C:A9:44:06:07:A6:1D:80:39
0x0017	39801DA6070644A98C5436CA3BC13EB7		00:00
Characteristic Declaration			
0x0018	0x2803	Characteristic Declaration	0A:19:00:64:EC:A5:E6:EC:7D:31:9F:87:4C:15:B6:0A:D4:AA:EA
0x0019	EAAAD40AB6154C879F317DECE6A5EC64		00:00
Characteristic Declaration			
0x001A	0x2803	Characteristic Declaration	0A:1B:00:D0:37:88:07:9E:F7:83:85:1E:4B:51:52:34:2A:6B:9F
0x001B	9F6B2A3452514B1E8583F79E078837D0		00:00

Figure 8: GATT Transaction Structure Simulation

From figure 8, we can see the simulation of our custom GATT transaction structure which is exactly the same as we mentioned above.

UART Module

For the connection between the RFID reader and the microcontroller, we chose the UART protocol because it is relatively easy to establish and configure.

In order to read the debug messages, we first set up the hardware so that the debug messages can be transmitted out from the evaluation board and be displayed on the terminal of our computer. In our final hardware setup, computer terminal and the wire connecting Pin P0[5] and Pin P12[6] are removed because we do not need message anymore. The hardware setup is shown in figure 9.

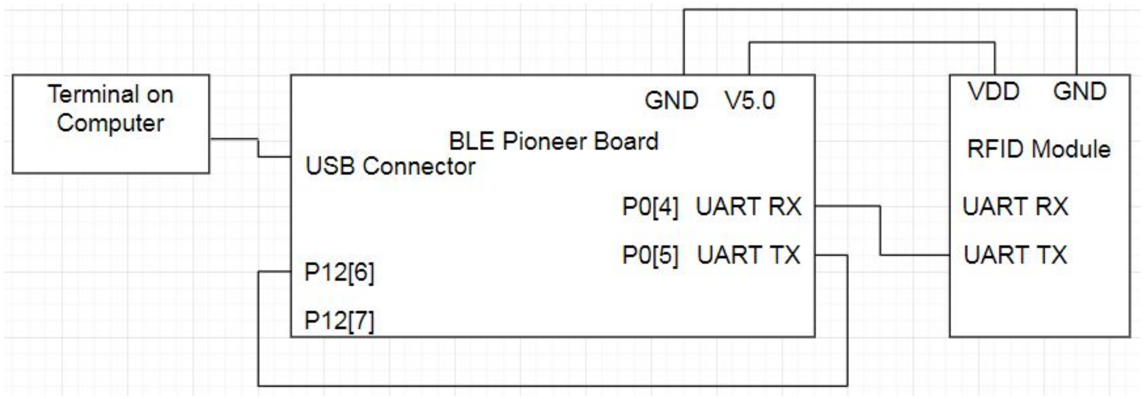
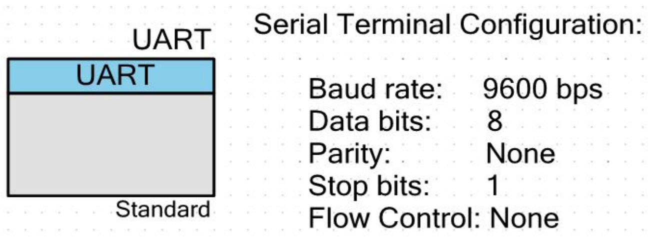


Figure 9: Hardware Setup for UART Connection Test

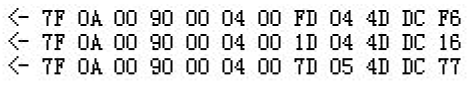
The basic configurations of the UART module are shown in figure 9. In addition, we prepared three sample RFIDs for testing shown in figure 9.



Serial Terminal Configuration:

Baud rate: 9600 bps
Data bits: 8
Parity: None
Stop bits: 1
Flow Control: None

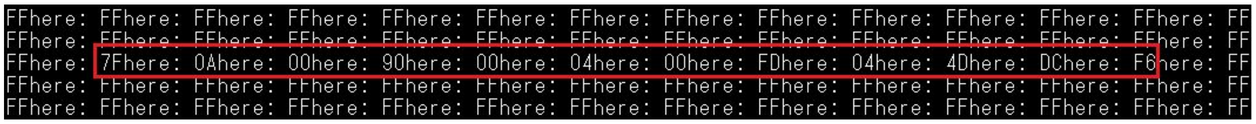
Figure 10: UART module basic



```
<- 7F 0A 00 90 00 04 00 FD 04 4D DC F6
<- 7F 0A 00 90 00 04 00 1D 04 4D DC 16
<- 7F 0A 00 90 00 04 00 7D 05 4D DC 77
```

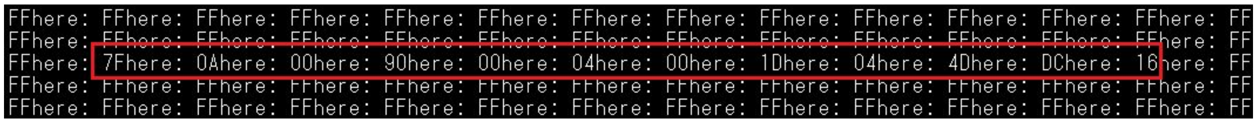
Figure 11: Three Sample RFIDs

At the beginning of the test, the UART connection was able to transmit data but the last three pairs of hex data are missed. Then we checked the wiring and increase the oversampling rate but neither of them worked. After we increased the RX buffer size from 8 bits to 16 bits, the problem was solved. To explain this weird behavior, our idea is that the UART protocol runs faster than our firmware. In our firmware, we can only pull out one byte from RX buffer at a time and need to push it onto the “printf” statement so that the debug message can be displayed on the terminal. When we try to pull out the next byte, the RX buffer might be overflowed already so the data at the end of the flow are missed. The successful UART data transmission messages are shown in figure 12, 13, and 14.



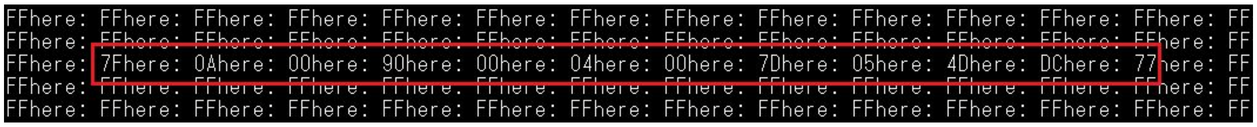
```
FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FF
FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FF
FFhere: 7Fhere: 0Ahere: 00here: 90here: 00here: 04here: 00here: FDhere: 04here: 4Dhere: DChere: F6here: FF
FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FF
FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FF
```

Figure 12: No.1 Sample RFID



```
FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FF
FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FF
FFhere: 7Fhere: 0Ahere: 00here: 90here: 00here: 04here: 00here: 1Dhere: 04here: 4Dhere: DChere: 16here: FF
FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FF
FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FF
```

Figure 13: No.2 Sample RFID



```
FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FF
FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FF
FFhere: 7Fhere: 0Ahere: 00here: 90here: 00here: 04here: 00here: 7Dhere: 05here: 4Dhere: DChere: 77here: FF
FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FF
FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FFhere: FF
```

Figure 14: No.3 Sample RFID

Motor Control Module

Since we are using a stepper motor to drive the linear actuator, we need to feed the stepper motor a certain number of pulses so that the linear actuator can move for the desired length.

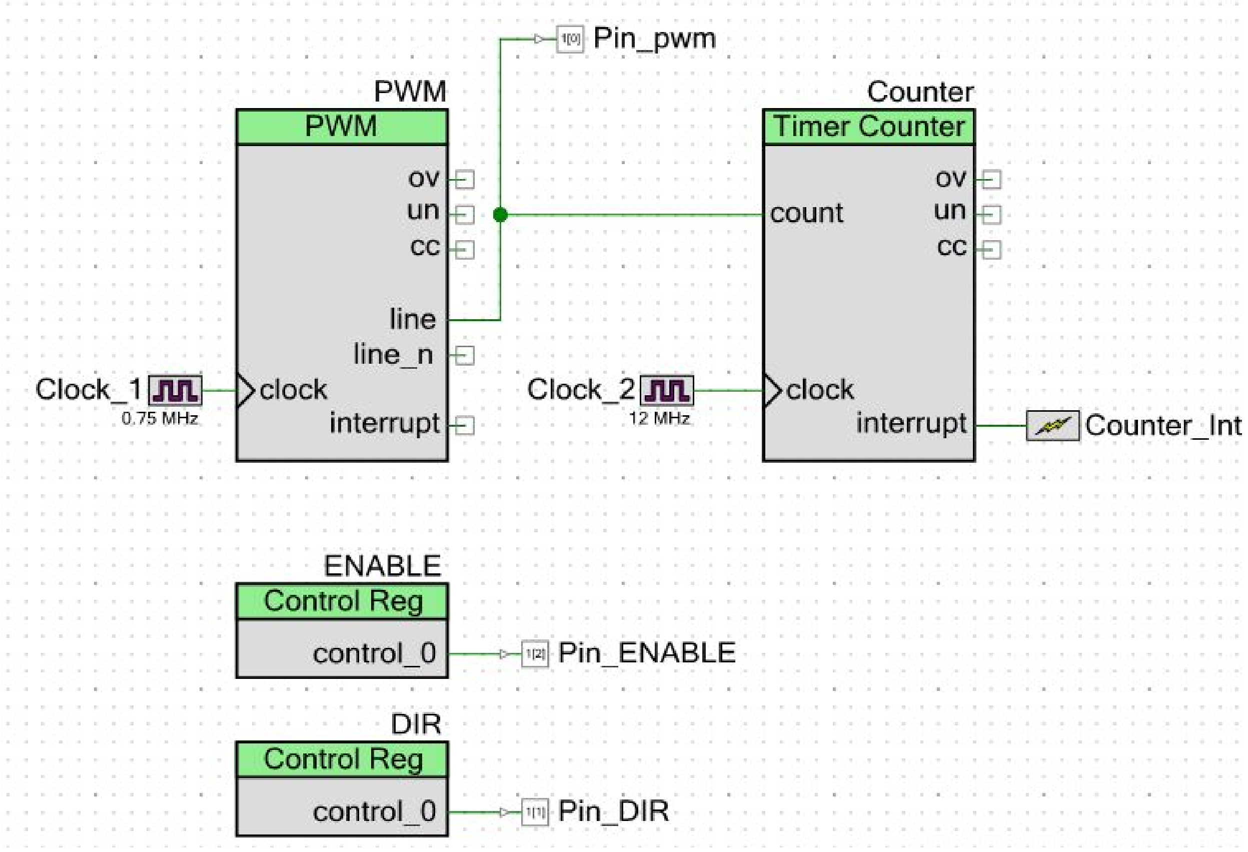


Figure 15: Schematic of the Motor Control Module

In addition to the PWM module, which generates PWM signals for the stepper motor through Pin_pwm, we also added a Counter module so that it can generate an interrupt to stop both the PWM and itself once a certain number of PWM pulses has been generated.

We used two control registers to feed ENABLE and DIR of our stepper motor. When DIR = 0, the linear actuator push forward; when DIR = 1, the linear actuator retract backward. ENABLE is always fed with 1.

For the basic settings of PWM, we set the period value to 20000 and the compare value to 10000 which means that the output PWM signal is of 10000/20000 = 50% duty cycle. To calculate the output frequency of PWM, we use the following formula shown in figure 16. The value of output clock frequency is just the reciprocal of time per pulse.

$$\text{Time per pulse} = \frac{\text{Period} * \text{Prescaler}}{\text{Input Clock Freq}}$$

Figure 16: Formula to calculate PWM output frequency(reciprocal of time per pulse)

PWM(50% duty cycle, prescaler = 1)			Time(sec, per 10 mL)
Clock(MHz)	Period value	Output Pulse(Hz)	
3	20000	150	22.61s
1.5	20000	75	44.57s
0.75	20000	37.5	89.66s

Table 2: Test Data of Stepper Motor

After testing with the stepper motor, we derived data shown in table 2. The last column labeled as "Time" stands for the time used for the linear actuator to dispense 10 mL of toothpaste. We tried three different input clock frequency. In order to avoid any possible leap of the plunger which can lead to error in amount dispensed, we selected the last row as our final setting for PWM to stabilize the dispensing speed. The final period value of Counter is calculated by the equation shown in figure 17.

$$\frac{89.66s * 37.5Hz}{\frac{10mL}{0.25mL}} = 84\ pulses\ per\ 0.25mL$$

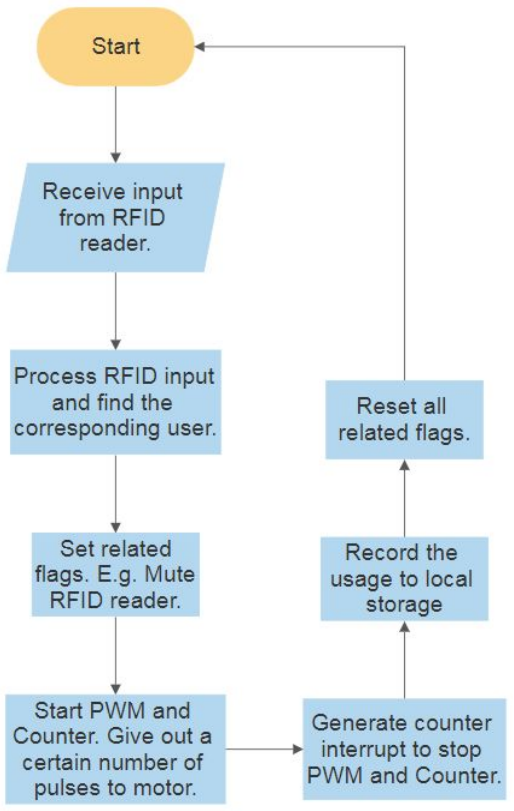
Figure 17: Equation to Determine the Period Value of Counter

Workflows

There are three main workflows in the program of our microcontroller: dispensing workflow, refilling workflow, and usage data update workflow respectively shown in figure 18, 19, and 20.

Dispensing workflow are very similar to refilling workflow: set related flags to prevent weird behavior, start and stop PWM and Counter, and reset all related flags. The only difference is that DIR is set to 1 for the linear actuator to retract back and then set to 0 after refilling is completed.

For the usage data update, the usage data in local storage will be upload to GATT database only when the Android application sends a read request to the microcontroller.


Start
Receive input from RFID reader.
Process RFID input and find the corresponding user.
Set related flags. E.g. Mute RFID reader.
Start PWM and Counter. Give out a certain number of pulses to motor.
Generate counter interrupt to stop PWM and Counter.
Record the usage to local storage
Reset all related flags.

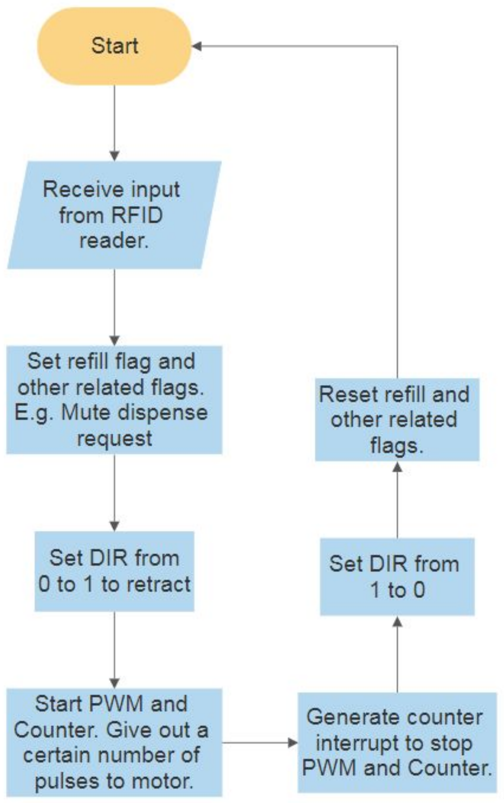

Start
Receive input from RFID reader.
Set refill flag and other related flags. E.g. Mute dispense request
Set DIR from 0 to 1 to retract
Start PWM and Counter. Give out a certain number of pulses to motor.
Generate counter interrupt to stop PWM and Counter.
Set DIR from 1 to 0
Reset refill and other related flags.

Figure 18: Dispensing Workflow **Figure 19**: Refilling Workflow

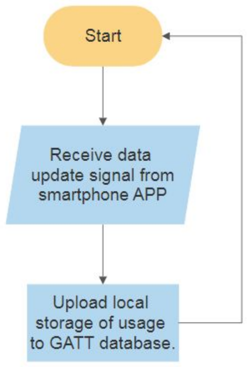

Start
Receive data update signal from smartphone APP
Upload local storage of usage to GATT database.

Figure 20: Usage Data Update Workflow

2.1.2 A4988 Stepper Motor Driver Chip

We use A4988 to build our own step motor control on PCB. Our step motor is 24 V and this chip can supply up to 35V. The power of our motor is 40W so its maximum input current is about 1.7 A and this A4988 chip can supply up to 1.5 A current. The figure below is a typical implementation and our wire connections are almost the same except we directly connect SLEEP and RESET pin together so the driver can always work. We connect step motor to OUT1AB,2AB and use the default step mode which is the full mode. And when our microcontroller sends one pulse to the STEP pin of the chip will make the step motor move about 1/200 turns and in this way, we can very precisely control the movement of our stepper motor.

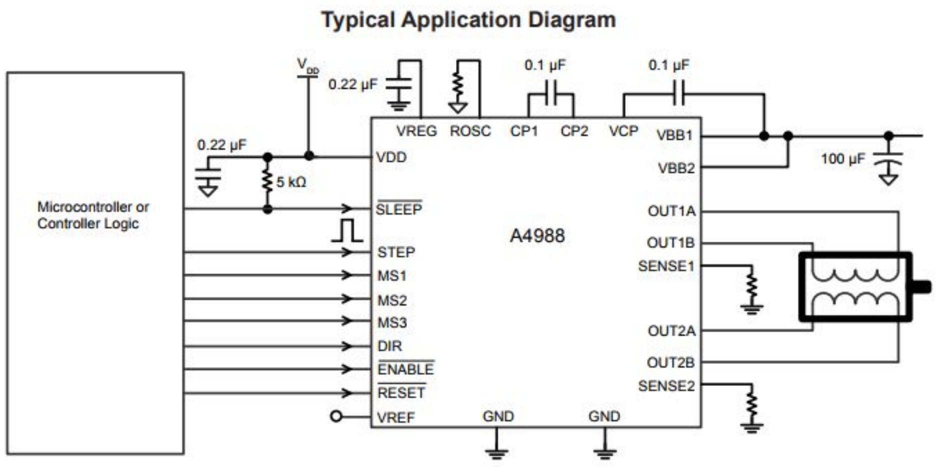


Figure 21: Typical implementation of A4988[8]

There is a problem we find out later after we have our prototype build in the machine shop. Because our mechanical design requires the motor to provide a big amount of force, we have to input at least 1.5A to the motor so that the refill or dispensing process can work. However, the design of the A4988 chip itself has a flaw on heat dissipation. Even with the heat sink, the A4988 chip will burn with a 1.5A output after relatively long time usage and our refill process takes a long time to finish. In fact, we have burned two A4988 modules when we test the final product. Since we directly implemented the A4988 chip on our PCB with our microcontroller, I have to replace A4988 stepper motor driver with a 4A current rating stepper motor driver. The code we use is exactly the same and the wire connection is the same for this backup driver (see figure 22).

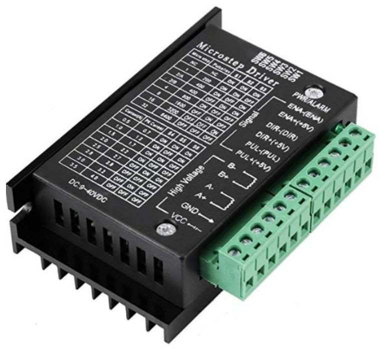


Figure 22: Backup stepper motor driver[6]

2.1.3 RFID Reader Module

The RFID reader module, Y13R, is capable of reading an S50 high-frequency RFID tag and its maximum reading distance is 6 cm. The actual read distance depends on the size of RFID tag and bigger RFID tag has longer read distance. It has UART, IIC and RS232 interface and we are using UART to connect with our microcontroller. The RFID reader can automatically read the ID information inside the RFID tags and send to the microcontroller. It takes 2 seconds for the RFID reader to trigger next read command so there will be no multiple reading problem. The working current of the RFID reader is 20mA and the sleep current of the RFID reader is 1mA so the power consumption is low and it is safe to use (The reading distance require us to put this reader close to the user).

2.2 Bluetooth Module

All BLE operations (our project only has Read/Write the GATT database) are triggered by the Android side, and the Microcontroller side program is interrupted by BLE operations and then handle them.

When the user triggers a BLE operation, the Application firstly requests to switch on the smartphone's Bluetooth. Then its back-end BLE protocol interface initializes and scans for other BLE device services. Once the device with the correct service UUID is found, the Application begins to build a connection between the smartphone and the microcontroller. The last step of establishing the BLE connection is discovering all GATT characteristics (databases) by UUIDs. The Application then interacts with GATT characteristics to executes the Read/Write operations. The last step is disconnecting with the microcontroller and closing the Bluetooth[11]. The code of the BLE interface is given in Appendix C.

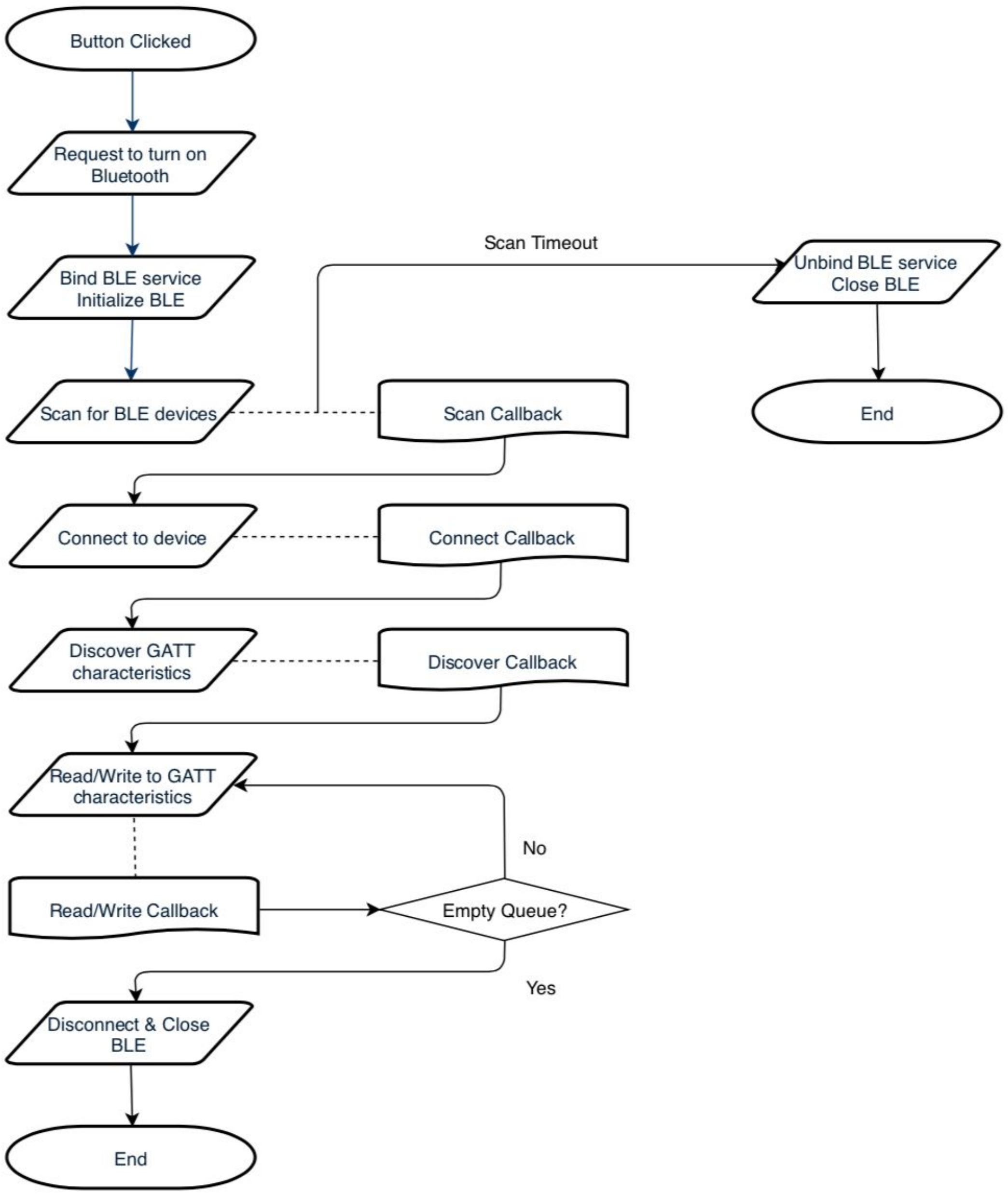


Figure 23: Android Side BLE Flowchart

We set the maximum device scanning time to 20 seconds to prevent the program from deadlocking. All Read/Write operations are queued to maintain thread safety because a BLE operation can only be executed after the previous one has called back. The code of the BLE operation queue is given in Appendix C.

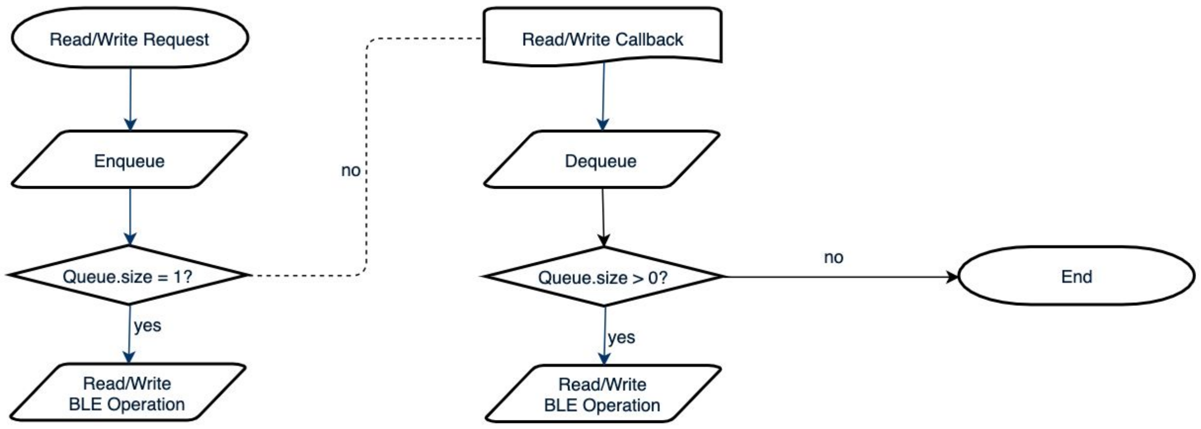


Figure 24: Read/Write Operation Queue Flowchart

2.3 Smartphone

The Android Application is the product's user interface. The Application contains the main page, which includes button navigator to subpages, and three subpages with different functions, and the back-end BLE protocol interface.

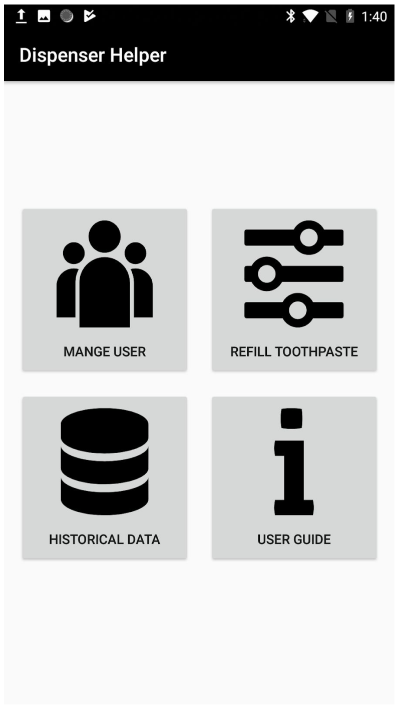
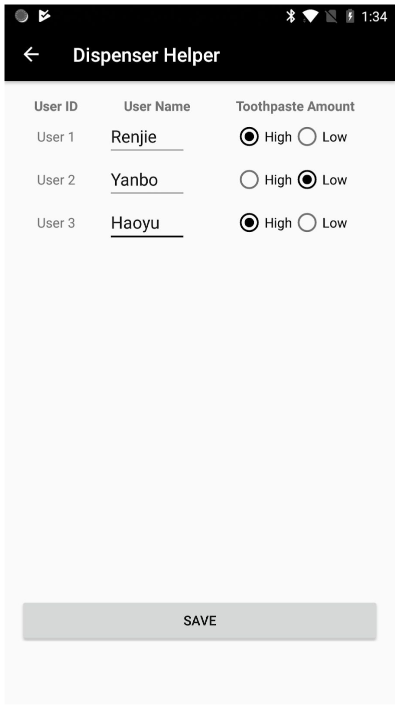


Figure 25: Application Main Page



User ID	User Name	Toothpaste Amount
User 1	Renjie	[✓ High] Low
User 2	Yanbo	High [✓ Low]
User 3	Haoyu	[✓ High] Low

Figure 26: Manage User Page

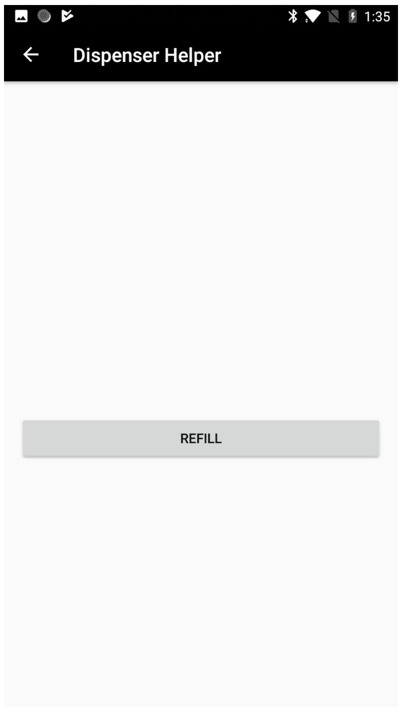
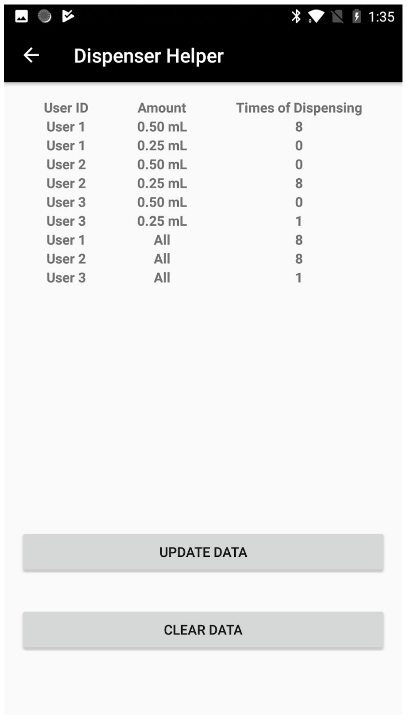


Figure 27: Refill Toothpaste Page



User ID	Amount	Times of Dispensing
User 1	0.50 mL	8
User 1	0.25 mL	0
User 2	0.50 mL	0
User 2	0.25 mL	8
User 3	0.50 mL	0
User 3	0.25 mL	1
User 1	All	8
User 2	All	8
User 3	All	1

Figure 28: Historical Data Page

The Manage User page (figure 26) allows the user to set the amount of dispensing toothpaste. The High amount corresponds to 0.50 mL, and the Low amount represents 0.25 mL. The Refill Toothpaste page (figure 27) has a button to trigger the refilling process. The Historical Data page (figure 28) contains a table of user history data, in which each row represents how many times the dispenser has dispensed a certain amount of toothpaste for a user. The buttons on the three pages are the triggers of establishing BLE connection with microcontroller and sending data to or retrieving data from its GATT database. The Manage User save button triggers the write operations to the userOneAmount, userTwoAmount, userThreeAmount characteristics. If the user sets the amount to High, the value would be 1, otherwise, the value is 0. The refill button writes a 1 to the Refill characteristic when the user hits it. The update data button reads the user history data, size 2 arrays, from the userOneData, userTwoData, userThreeData characteristics.

The Application is compatible with all Smartphones that run Android OS of version 6.0 or higher. We originally planned to use the mobile database framework as the solution of data storage, but after we calculated the required volume of storage space is small, we realized that the Android built-in internal storage is the more appropriate choice.

Each user has 3 pieces of data

Name:** At most 10 characters -> **10 bytes

Toothpaste Amount:** Boolean -> **1 byte

History Data:** Integer array (size = 2) -> **2 bytes

Total Volume Required = (10 + 1 + 2) * 3 = 39 bytes

The Android internal storage is the best solution for a small amount of data because it's secure (not accessible to other Applications) and can be fast read or updated[10].

Edge cases are fully considered and handled, for example, when the Save button on the Manage User page is clicked, new user settings will be updated to both the internal storage data file and the microcontroller GATT database. We set a protective mechanism to avoid the data file being updated when the Bluetooth connection fails, therefore, we guarantee that the Application and the microcontroller always have the same user settings.

2.4 Mechanical Unit

2.4.1 Syringe Container

The main mechanical component is a syringe, which functions as the buffer between the toothpaste tube and the users' toothbrushes. The plunger of the syringe is controlled by the linear actuator. To dispense toothpaste, the linear actuator will move forward and put toothpaste out of the syringe. To refill toothpaste, we first put a toothpaste tube on the exit of the syringe. Then the step motor on the linear actuator rotates inversely so linear actuator can move backward and draw toothpaste into the syringe. There is also a cap to prevent any toothpaste leak from the syringe. For the refill process, it is very important to make sure only a small volume of air can get inside the syringe otherwise the dispensing process would only push air out of the syringe instead of toothpaste.

2.4.2 Linear Actuator with stepper motor

We use the Linear Actuator to push and pull the plunger of the syringe. The length of the actuator is about 80mm and this is the same number as the maximum moving length of the plunge. The stepper motor on this Linear Actuator is 24V and the linear actuator can provide about 200 N force with 1.7A input to the stepper motor. The force needed to draw toothpaste into the syringe is equal to the force needed to lift a 10 kg stuff so 200 N force is a safe choice. The maximum moving speed of the linear actuator is roughly 5 mm/s and our designed speed during the operation is about 1 mm/s. We can easily control the turns of stepper motor through our A4988 motor control chip so we can precisely control the output toothpaste.

2.5 Power Supply

We have a 110V AC to 24V DC power adaptor and a 24V to 5V module. The power adaptor is connected to a wall socket to power the entire project. Since our stepper motor driver requires 1.5A and the microcontroller and RFID reader require less than 300mA, the 10A current rating for this power adaptor is a very safe value. The 24V to 5V module is used to power the microcontroller and RFID reader. The 2A current rating of this module is also big enough because the total current consumption of this module is less than 300 mA.

2.6 RFID Tags

We used the sticker type RFID tag and I chose the S50 type tag with 13.56 Mhz working frequency (refer as a high-frequency RFID tag) and its size is 15mm*30mm. I use this type of RFID Tags because only this type of RFID tag can be small enough to stick on a toothbrush. Our RFID tags have 2.5cm reading distance and this is a desirable value because we don’t want any miss read if the toothbrush with RFID tag is put close to the dispenser.

3 Design Verification

3.1 Syringe Airtightness and Air leakage

In our mechanical design, the syringe airtightness is a very important factor because we are trying to draw toothpaste from the toothpaste tube into the syringe through the small nozzle. During the refilling process, some air might enter the syringe. Another source of air is the toothpaste tube, which usually contains a small portion of air due to the manufacturing technique. Therefore, after refilling, some air is always trapped inside the syringe. When the RFID reader identifies the user and starts the dispensing process, the air inside syringe comes out first. In result, we set the maximum air leakage inside the syringe can be no more than 15 ml when the refilling process is complete. We have tested the refilling process many times to make sure of that and the below chart is the result.

Refilling Speed (ml/second)	Volume of Air Gap (ml)
0.25	13.5
0.5	14.3
1	14.8
2	16.4

Table 3: Relationship between Refilling Speed and Airtightness

After testing different refilling speed, we know that the longer time the refilling process takes, the less air leak into the syringe. Besides, when we tested with a more expensive toothpaste, the air gap inside the syringe decreased, and we speculate that it happens because the quality of the expensive toothpaste is higher and thus contains less amount of air.

3.2 Android Application

Our requirement of the Android Application loading time is within 3 seconds. We tested its performance by two tools: electronic stopwatch and Android Studio Debugger system performance evaluation tool. The testing process is launching the APP for ten times (killing background process first) and calculating the average loading time. The actual loading time is 0.92 second in normal mode and 1.78 seconds in low battery level mode.

3.3 Bluetooth Module

The initial requirement on the longest time of Bluetooth connection is within 30 seconds. We first tested the Bluetooth performance extrinsically by an electronic stopwatch, but we then realized that the connection speed was too fast to be identified by human eyes. So we then set breakpoints at the code of BLE callback functions to measure the length of the system pending. The actual connection time is not strictly stable and is between 1~3 seconds, which is much shorter than our expectation.

4 Cost & Schedule

4.1 Cost Analysis

We have 3 people in our group and use 40$/ hour as our hourly salary. We will put 10 hours/week into this project and there are about 16 weeks for us to do this project. Total labor cost: 3*(40*12*16)*2.5=$57,600 in total.

Component	Cost
CYBLE PSoC 4 BLE pioneer board	$30*1=$30
CYBLE MiniProg3	$99*1=$99
CYBLE 214015-01 microcontroller chip	$18*2=$36
Sticky RFID tags	$0.24*16=$3.84
Y13R RFID reader	$6.66*1=$6.66
USB to TTL line	$1.15*1=$1.15
110V AC to 24V DC Power adaptor	$20*1=$20
24V to 5V module	$9.95*1=$9.95
Linear Actuator with stepper motor	$80*1=$80
Connection wire	$0.1*20=$2
Syringe	$2.99*2=$5.98
Colgate toothpaste (Pack of 12)	$12.99 *1 = $12.99
Labor	$57600
Total	$57907.6

4.2 Schedule

Week	Member	Task
2.11	Haoyu Tian	Start writing the design document and buy the necessary components.

	Renjie Fan	Start writing the design document and buy the necessary components.
	Yanbo Chen	Start writing the design document and buy the necessary components.
2.18	Haoyu Tian	Detailed Mechanical part design.
	Renjie Fan	Learn to develop with CYBLE BLE Pioneer Baseboard.
	Yanbo Chen	Learn to develop with CYBLE BLE Pioneer Baseboard.
2.25	Haoyu Tian	Finish Mechanical part design and order mechanical part prototype from the machine shop.
	Renjie Fan	Complete mobile App UI design.
	Yanbo Chen	Build a sample Bluetooth connection between smartphone and board successfully.
3.4	Haoyu Tian	Develop the connection between the CYBLE board and the motor control board.
	Renjie Fan	Develop mobile App framework.
	Yanbo Chen	Work on RFID module.
3.11	Haoyu Tian	Enable CYBLE Board to control the movement of the motor and start to develop RFID reader connection.
	Renjie Fan	Develop mobile App database function and learn Android Bluetooth programming.
	Yanbo Chen	Start working on data transfer between the smartphone and the board.
3.18	Haoyu Tian	Spring break
	Renjie Fan	Spring break
	Yanbo Chen	Spring break
3.25	Haoyu Tian	Develop CYBLE board to change the configuration for the corresponding RFID.

	Renjie Fan	Build Bluetooth communication between mobile App and CYBLE board.
	Yanbo Chen	Finish data transfer function between smartphone and board
4.1	Haoyu Tian	Test and debug the whole project without the mobile App. Make sure RFID reader and mechanical part work correctly when using CYBLE board to control.
	Renjie Fan	Test mobile App’s User Interface and function of users’ settings.
	Yanbo Chen	Enable user to configure the amount of toothpaste dispensed through the mobile App
4.8	Haoyu Tian	Assembly all the components on a platform and improve product appearance.
	Renjie Fan	Collaborate with Yanbo on general software testing and debugging.
	Yanbo Chen	Collaborate with Renjie on general software testing and debugging.
4.15	Haoyu Tian	Collaborate with teammates on general software-hardware testing and debugging.
	Renjie Fan	Collaborate with teammates on general software-hardware testing and debugging.
	Yanbo Chen	Collaborate with teammates on general software-hardware testing and debugging.
4.22	Haoyu Tian	Write final reports and demo project
	Renjie Fan	Write final reports and demo project
	Yanbo Chen	Write final reports and demo project
4.29	Haoyu Tian	Group presentation
	Renjie Fan	Group presentation

	Yanbo Chen	Group presentation

5 Conclusion and Ethics

5.1 Accomplishments

All requirements are tested and verified, and we have successfully demonstrated all the required functionality. We have integrated the CYBLE-214015 microcontroller chip and the A4988 Stepper Motor Driver Module on our PCB. The microcontroller is perfectly soldered and programmable. The RFID module can sense RFID tags within the range of 5 cm. The device can identify user by the RFID tag attached on the toothbrush, and then dispense the preset amount of toothpaste to the user. The mechanical unit is able to dispense a different amount of toothpaste and retract the syringe plunger to refill toothpaste. The bug-free Android Application has all expected function, user can set toothpaste dispensing amount, trigger refilling process, and viewing the history data. The loading time of the Application is within 2 seconds. The performance of BLE connection is beyond our expectation and a GATT operation can usually be executed and finished within 1~3 seconds.

5.2 Uncertainties

We are unsatisfactory that we have such a time-consuming refilling process, which takes 89.66 seconds to refill 10 mL of toothpaste. See Table 2, we set the clock frequency to 0.75 MHz, which results in a relatively low pulse frequency. The reason that we have to slow down the process control is a faster refilling process leads to a larger amount of air gap inside the syringe container. See Table 3, there is a trade-off between the refilling speed and the volume of the air gap. Considering that the quality is more important than the speed, we decided to choose the securer plan.

We initially planned to collect the user usage data on a daily, monthly, and yearly basis rather than just the number of times of dispensing. The original plan requires the microcontroller to maintain a real-time clock so that it can record the exact time of each dispensing. However, we tried several times to implement the real-time clock and didn’t succeed. Ultimately, we used the backup plan that the microcontroller only updates the number of time of dispensing to each user.

5.3 Future Work

As an electronic device that will be mainly used in bathrooms, our product prototype is too big and not waterproof. Therefore, the most crucial improvement is shrinking its size and adding a waterproof cover.

Another potential design alternative is improving the airtightness of the toothpaste container. Our current design is lack of an effective method to reduce the volume of air trapped inside the syringe after the refilling process. One possible way is to design a new refilling mechanism to prevent air from entering the syringe, for example, screwing toothpaste tube onto the container instead of simply plugging the syringe nozzle into the toothpaste tube. Another feasible method is pushing the syringe plunger forward after refilling to squeeze the air out. On the other hand, a container with better airtightness allows us to shorten the length of the refilling process, which would be a great improvement to our project.

5.4 Ethical Considerations

Our design includes using power adaptors with high current and this toothpaste needs to place in the restroom so we must prevent any water leak into our project otherwise it might hurt the users of this dispenser. This may be a violates of IEEE # 1 code that "to hold paramount the safety, health, and welfare of the public, to strive to comply with ethical design and sustainable development practices, and to disclose promptly factors that might endanger the public or the environment."[4] We will try our best to improve the physical design of this project to overcome this potential risk.
Our design of mechanical components might refer to the mechanical design of manual dispenser on the Internet. Therefore, this would potentially be a violation of #2 of the IEEE code of ethics - to avoid real or perceived conflicts of interest whenever possible [4]. We currently come up with a mechanism that is significantly different from those manual dispensers. In the following modifications of our design, we will try to avoid using the ideas that appeared in the existing products.

Reference

[1]William J Leep, Amazon Product Review, ‘Messy!’, 2018. [Online]. Available: https://www.amazon.com/iLifeTech-Toothpaste-Dispenser-Automatic-Squeezer/product-reviews/B00NRW4LAS/ref=cm_cr_dp_d_hist_1?ie=UTF8&filterByStar=one_star&reviewerType=all_reviews#reviews-filter-bar [Accessed Feb 2019].

[2] John-he-928, GitHub project, Tencent/WCDB, 2019. [Online]. Available: https://github.com/Tencent/wcdb [Accessed Feb 2019].

[3]Krishnaprasad MV, ‘Getting Started with PSoC® 4 BLE’, 2018. [Online]. Available: https://www.cypress.com/file/141171/download [Accessed: Feb 2019].

[4]IEEE, ‘IEEE Code of Ethics’, 1990. [Online]. Available: http://ewh.ieee.org/cmte/substations/posted_documents/ieee_codeofethics.pdf [Accessed Feb 2019].

[5]Amazon, ‘Frienda 4 Pack Large Plastic Syringe for Scientific Labs and Dispensing Multiple Uses Measuring Syringe Tools (60 ml and 100 ml)’, 2019. [Online]. Available: https://www.amazon.com/gp/product/B07KY5K58W/ref=ppx_yo_dt_b_asin_title_o00__o00_s00?ie=UTF8&psc=1 [Accessed Feb 2019].

[6]Taobao, ‘步进电动推杆机’, 2019. [Online]. Available: https://detail.tmall.com/item.htm?spm=a230r.1.14.6.3530440e95tKya&id=520789835915&cm_id=140105335569ed55e27b&abbucket=19 [Accessed Feb 2019].

[7]Taobao, ‘RFID 读写器 Y13R’, 2019. [Online]. Available: https://item.taobao.com/item.htm?spm=a230r.1.14.1.36532633L3NxDm&id=557513420947&ns=1&abbucket=19#detail [Accessed Feb 2019].

[8]Pololu,’DMOS Microstepping Driver with Translator And Overcurrent Protection’, 2019. [Online]. Available:https://www.pololu.com/file/0J450/a4988_DMOS_microstepping_driver_with_translator.pdf [Accessed Feb 2019].

[9]cypress,’CYBLE-214015-01: EZ-BLE™ Creator Module’, 2019. [online]. https://www.cypress.com/documentation/datasheets/cyble-214015-01-ez-ble-creator-module [Accessed Feb 2019].

[10] Google, Android documentation, Data and File Storage Overview, 2019. [Online]. Available:
https://developer.android.com/guide/topics/data/data-storage [Accessed
09-Mar-2019].

[11] Google, Android documentation, Bluetooth Low Energy Overview, 2019. [Online] Available:
https://developer.android.com/guide/topics/connectivity/bluetooth-le.html [Accessed
11-
Mar-2019].

Appendix A: Schematic and PCB Layout

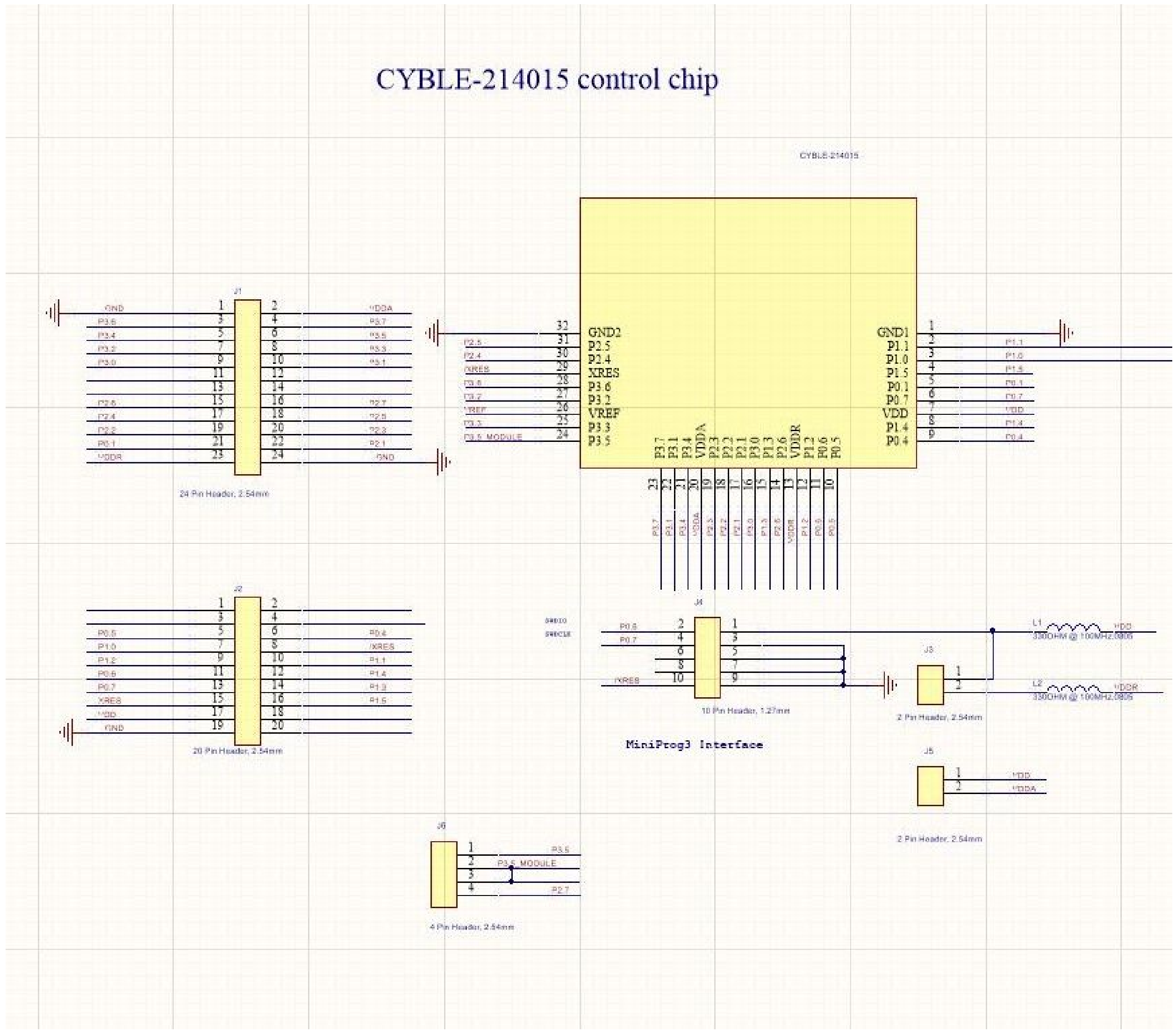


Figure 29: CYBLE-214015 control chip schematic

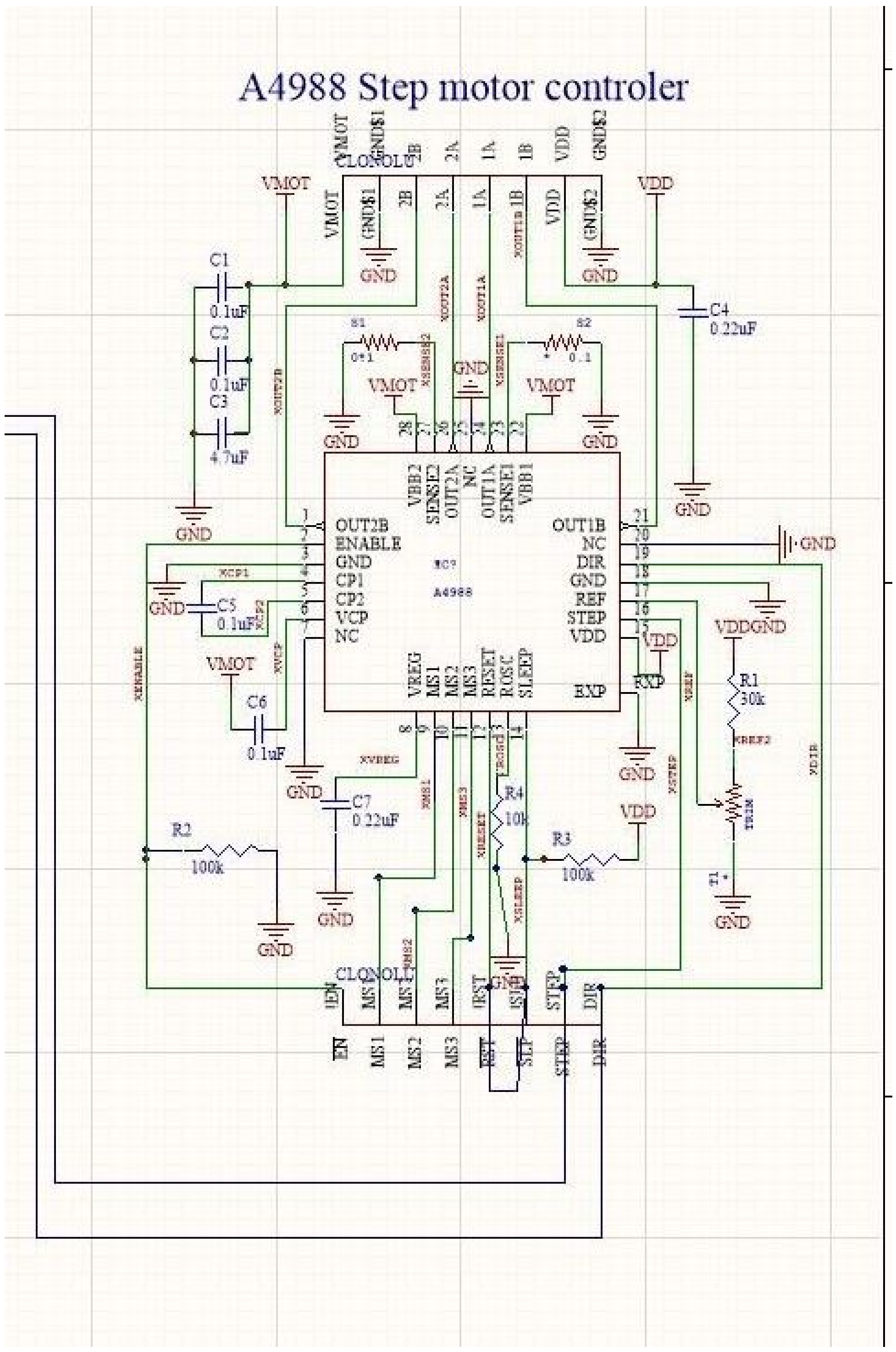


Figure 30: A4988 stepper motor driver chip schematic

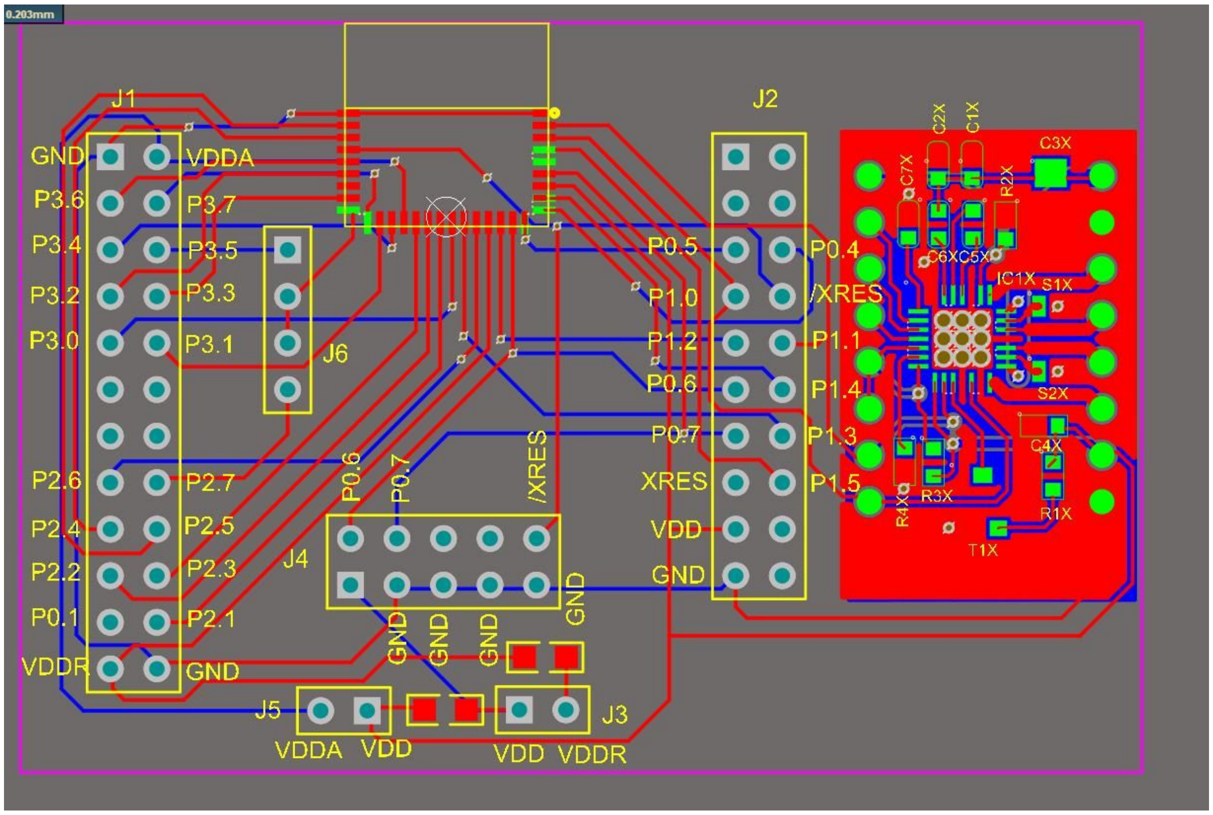


Figure 31: PCB layout

Appendix B: Requirements and Verifications Table

Smartphone Application

Requirement	***Verification***	***Verified? (Y/N)***
The APP's loading time is about 3±0.5 seconds.	*Start the APP ten times (kill its background process each time before starting it). Calculate the average time spent on loading by using an electronic stopwatch.*	Y

Bluetooth Module

Requirement	***Verification***	***Verified? (Y/N)***
The Bluetooth connection provided by the module should enable user smartphone to connect in 30 seconds after the Bluetooth module on PCB starts working. After the connection is established, the data can be transferred between the board and the smartphone.	*If the connection between the smartphone and the board is successfully established, the smartphone should display "Connected". If the board is then turned off, the smartphone should display "Disconnected".*	Y

Control Unit

CYBLE-214015-01 EZ-BLE™ Chip

Requirement	***Verification***	***Verified? (Y/N)***
1. The input voltage to the VDD is about 4.7±0.5V.	1. *Use a multimeter to measure the voltage supply of the VDD input of the PCB to see if it falls in 4.7±0.5v.*	Y

RFID Reader

Requirement	***Verification***	***Verified? (Y/N)***

1. Operating voltage between 4.7±0.5V. *2. It can read the RFID tag within 2±1 cm and output the corresponding ID.*	*1. Use a multimeter to measure the voltage supply of the VDD input of the PCB to see if it falls about 4.7v.* *2. We have software to test the function of the RFID reader. The RFID reader part on PCB can be directly connected to our computer with USB wire (TTL). Put the RFID tag close to our PCB and see if the ID is displayed on our computer.*	*1.Y* *2.Y*

A4988 Stepper Motor Driver

Requirement	***Verification***	***Verified? (Y/N)***
1. The input Voltage to the VDD pin should be about 5±0.5V. *2. The chip can keep working for 150±20 seconds without burning the a4988 chip.*	*1. Use a multimeter to measure the input voltage of this A4988 module. Also, we can directly connect this a 4988 part of our PCB to a stepper motor and send a signal to STEP pin to see if the motor can move correctly.* *2. Also, we can directly connect this a 4988 part of our PCB to a stepper motor and keep sending a signal to STEP pin for 3 minutes to see if it will burn.*	*1.Y* *2.Y*

Mechanical Part

Linear Actuator with Linear Actuator

Requirement	***Verification***	***Verified? (Y/N)***
1. The Actuator can provide about 200 N (+	*1.A. Connect the motor to our a4988 motor control PCB and send a signal*	*1.Y*

infinite / -10N) torque to lift things up. *2. The pushing length of the actuator should be about 80±2 mm.*	*to the step input and see if the linear actuator can start moving.* *B. Put a 20 kg stuff on top of the actuator and see if it can push up. If it does, then it can provide about 200N torque.* *2. We use motor control to move the push rod inside the actuator to its maximum position and measure the length of it. The error of this number will only affect the capacity of our syringe.*	*2.Y*

Mechanical Components

Requirement	***Verification***	***Verified? (Y/N)***
1. After completely fill the syringe with toothpaste, the air inside the syringe should be less than 15±2ml. *2. The syringe should be airtight when the front cap is on the syringe and the input hole is blocked.* *.*	*1. We manually fill the syringe with toothpaste. After that, we start pushing the plunge until toothpaste starts to come out. Because the air will be pushed out first so we can check the scale on the syringe to see if the volume of the air gap is about 15ml.* *2. Block the input hole of toothpaste on the syringe. Put the cap on the syringe and pull the plunger. An airtight syringe should allow the plunger to bounce back to the original position after being released.*	*1.Y* *2.Y*

Toothbrushes(RFID)

Requirement	***Verification***	***Verified? (Y/N)***

The RFID can be read within 3 cm by our RFID reader with the correct value.	*A. Connect the RFID reader to a computer and use a software (provided by seller) to read the signal from the RFID reader.* *B. Put RFID on a toothbrush and hold the toothbrush to approach the reader.* *C. Check software to identify the maximum range that the RFID reader can sense RFID tags.*	*Y*

2.8 Power supply

Requirement	***Verification***	***Verified? (Y/N)***
1. The power adapter supplies 24±0.5V voltage for the motor. *2. The power adapter can supply 10±0.5A current.* *3. The 24V to 5V voltage reduction module can supply about 5±0.5V.*	*1. Connect the power adapter to the wall socket and use a multimeter to measure the output voltage of the power adapter.* *2. Use a multimeter to measure the output current of the power adapter. Also, directly connect the power adapter to the motor to see if the motor works correctly.* *3. Use a multimeter to measure the output voltage of the voltage reduction module.*	*1.Y* *2.Y* *3.Y*

Appendix C: Code

BLE Protocol Interface

https://github.com/jamesfrj/Android-BLE-Project/blob/master/app/src/main/java/com/example/dispenserhelper/BluetoothLowEnergyService.java

BLE Operation Queue

https://github.com/jamesfrj/Android-BLE-Project/blob/master/app/src/main/java/com/example/dispenserhelper/CharacteristicQueueObject.java