Good2Go Automated Token Exchanger (GATEr)

Ariocie Liang (arliang2)
Henry Guan (henryg3)
John Kim (jaehank2)

ECE 445 Final Report, Senior Design, Spring 2022
Team 38
TA: Akshatkumar Sanatbhai Sanghvi
May 05, 2022

Abstract

This project aims to create a product that streamlines and automates a service called Good2Go that is prevalent in the U of I dining halls. The original service provides a container that a student can use to take food to go from the dining halls in exchange for a token or an old container. The design focuses on automating the process of exchanging tokens and containers rather than having a student find a dining hall employee to complete the process. In this report, the project's design, requirements and verifications, and results are comprehensively detailed. Although there were small shortcomings and issues that affected the ideal final product, our team achieved a proof of concept design that provides a solution to the current problems of the Good2Go system, which is exposed to several security and labor issues.

Table of Contents

1) Introduction 1
1.1) Problem Statement 1
1.2) Solution 1
1.3) High Level Requirements 1
1.4) Subsystem Overview 2
2) Design 3
2.1) Design Procedure 3
2.1.1) UI Subsystem 4
2.1.2) Sensing Subsystem 5
2.1.3) Control Subsystem 7
2.1.4) Power Subsystem 8
2.2) Design Details 9
2.2.1) UI Subsystem 9
2.2.2) Sensing Subsystem 10
2.2.3) Control Subsystem 11
2.2.4) Power Subsystem.......... 11
3) Verification 12
3.1) UI Subsystem 12
3.2) Sensing Subsystem 12
3.3) Control Subsystem 12
3.4) Power Subsystem 12
4) Costs and Schedule 13
4.1) Costs 13
4.2) Schedule 14
5) Conclusion 15
5.1) Accomplishments and Uncertainties 15
5.2) Ethical Considerations 15
5.3) Future Work 16
6) References 17
7) Appendix A – Requirements and Verification Tables with Results 19
8) Appendix B – Verification Outputs 21
9) Appendix C – PCB Layout 24
10) Appendix D – PCB Schematic 25
11) Appendix E – Software Code 26

1. Introduction

1.1 Problem Statement

Since the pandemic began, more and more people have had to become isolated to avoid the spread of COVID-19. As a response to such measures, the U of I dining hall system implemented a token/container exchanging system named Good2Go (G2G) to allow students to take food from the dining halls. The system allows students to exchange an old container for a redeemable wooden token to then exchange for a new container [1]. The current process lacks a unified system across all U of I dining halls and relies heavily on human supervision and interaction. Furthermore, the intermediate process of having to receive a physical token first in order to exchange for a new container causes significant delay in the system when students can directly exchange a new container by returning an old one. The current system also proves to be a hassle for both parties involved in the process since it is too simple to have a single person be assigned to it, but occurs often and spontaneously enough to become bothersome.

1.2 Solution

To solve the above problem, we developed a machine that automates the process of exchanging tokens and containers. The machine is capable of storing digital tokens, removing any inconvenience caused by carrying a physical token such as concern for loss. The system not only allows an exchange between old containers and tokens but also a direct exchange between old containers and new containers. The goal of the machine is to transform the process into a semi-autonomous system by removing human to human interactions and also allowing users a choice to select between a digital token and a new container. This improves the efficiency, simplicity, and performance speed of the current G2G container system.

The machine activates upon a card swipe to strictly allow only U of I students for access using their iCards. A QR sensor and a load cell are used to detect validity of the containers using weight measurements and G2G QR code. The front of the machine has LED status messages to show user-friendly instructions as well as push buttons to take user input. Lastly the microcontroller and motors are used to process the internal state of the machine and execute the actions necessary to retrieve or dispense containers and digitally update a user's token count. The sensors are the main replacement of the human to human interaction with the role of authenticating valid containers and tokens. The validity of each container is defined by its QR code as well as the weight to ensure minimal food waste within the G2G containers.

1.3 High Level Requirements

The three high level requirements below outline the successful functionality of the machine:

1. The system should deliver the entire process from receiving an old container or token to dispensing a new container in less than 15 seconds, assuming minimal delay in the user's action.
2. The system should dispense exactly one new container at a time indefinitely without jamming and should never allow more than one container to be dispensed upon a single request

3. The system must correctly reject invalid G2G containers without QR codes or with weight that exceeds 7*g* compared to the base weight.

The first high level requirement ensured that the machine could handle a high number of users by leaving ample time to make their selection while minimizing delay to complete the exchange. The next high level requirement ensured that all exchanges were one-to-one; in other words, no user should ever accidentally receive more than one container or receive none due to a mechanical fault in the system. The final high level requirement ensured that fraudulent exchanges with invalid containers were rejected, and only fully validated containers could continue through the exchange process. All three requirements were met by the final design and will be further discussed in Section 2 of the paper.

1.4 Subsystem Overview

Figure 1 below outlines the four subsystems used in our machine. The power subsystem supplied 12*V* to the motors in the control subsystem, 3.3*V* for the LEDs and 5*V* to all other components including the microcontroller and sensor modules. The control subsystem had motors receiving different data signals from the main microcontroller to activate the two types of mechanical arms responsible for retrieval and dispensing system. The UI subsystem mainly handled reading user inputs from the push buttons and outputting LED status messages for instructions. Finally, the sensing subsystem was used as a method of validation for both the user and the container. The load cell and QR scanner were responsible for verifying the weight and the QR code of a G2G container while the card reader was placed to authenticate user information through iCards.

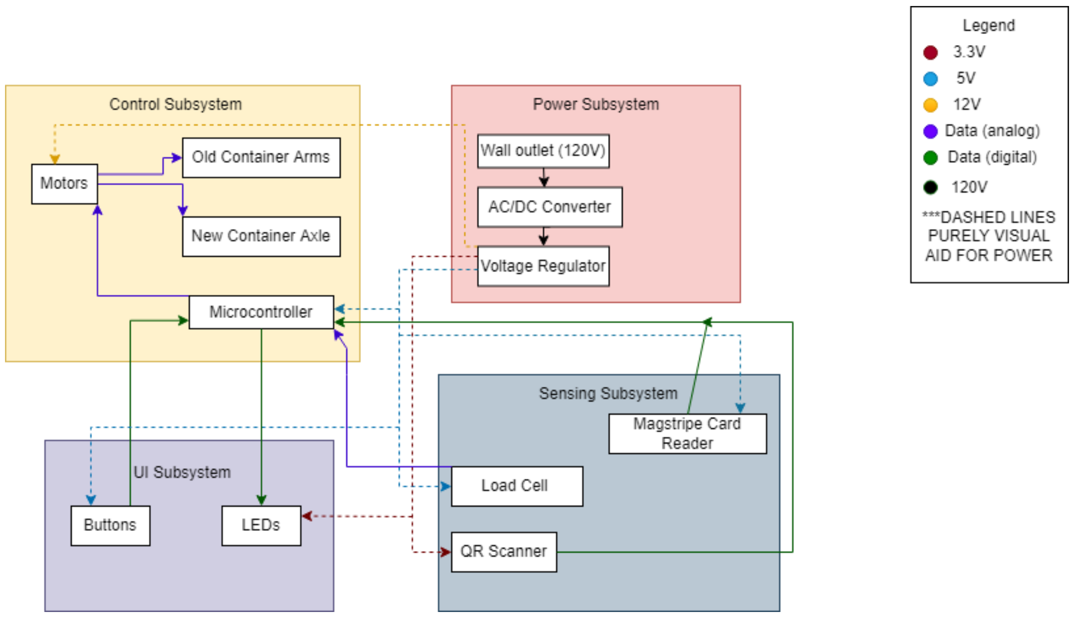


Figure 1. Block Diagram of the G2G Machine

2. Design

2.1 Design Procedure

With the current Good2Go system being only a service and not a product, the machine required complex design specifications accounting for different specs of each module. The scanning distance and field of view of the QR sensor required specific distance and tilt threshold from the containers while the load sensors and mechanical arms were placed in conjunction to parallelize the action of validating and retrieving returned containers. The base top-down view in Figure 2 also shows how returned containers are retrieved into an empty bin by the mechanical arms. The most intricate part of the mechanical design was the dispensing system, which required a mechanism to output exactly one container without any faulty behaviors such as physical jamming. This was done using a treadmill system powered by a continuous servo motor with an opening for exactly a single container height (1.25*in*) as shown in the bottom two diagrams of Figure 2.

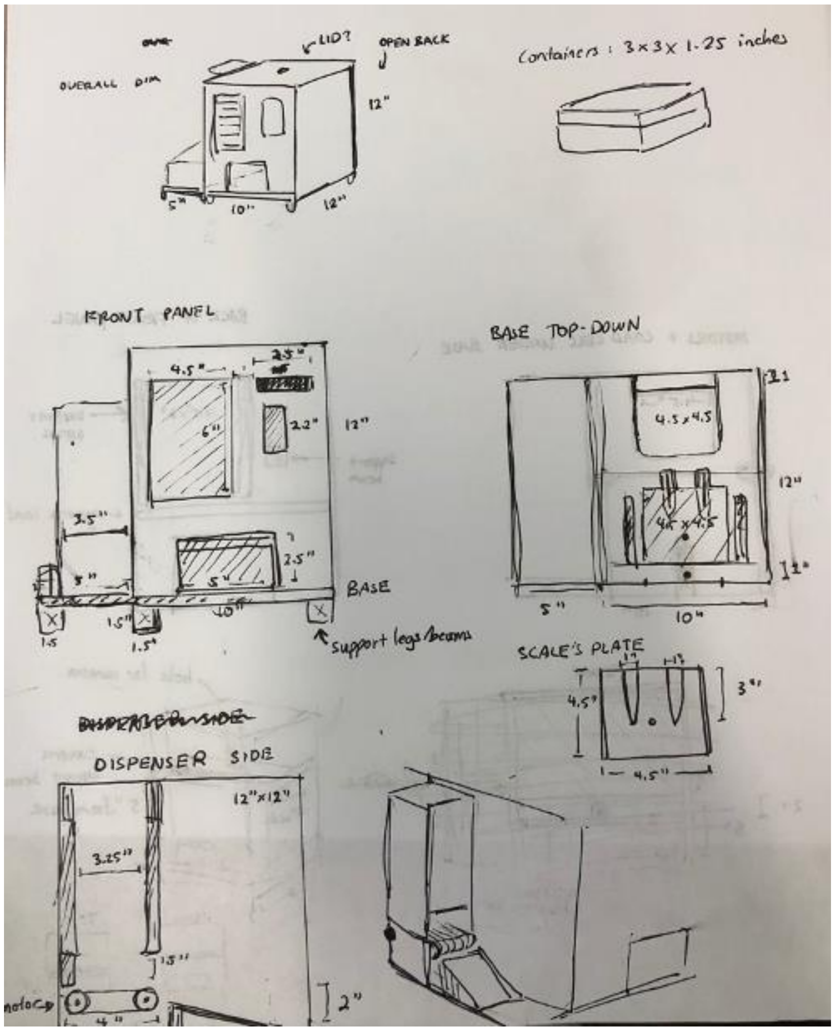


Figure 2: Design Specification of the G2G Machine

The design procedure also consisted of assigning another layer of state transition to the current G2G system to allow a direct exchange between old containers and new containers. Figure 3 below illustrates the finite state machine of the logic flow in which the initial state corresponds to the machine's idle state waiting for a card swipe.

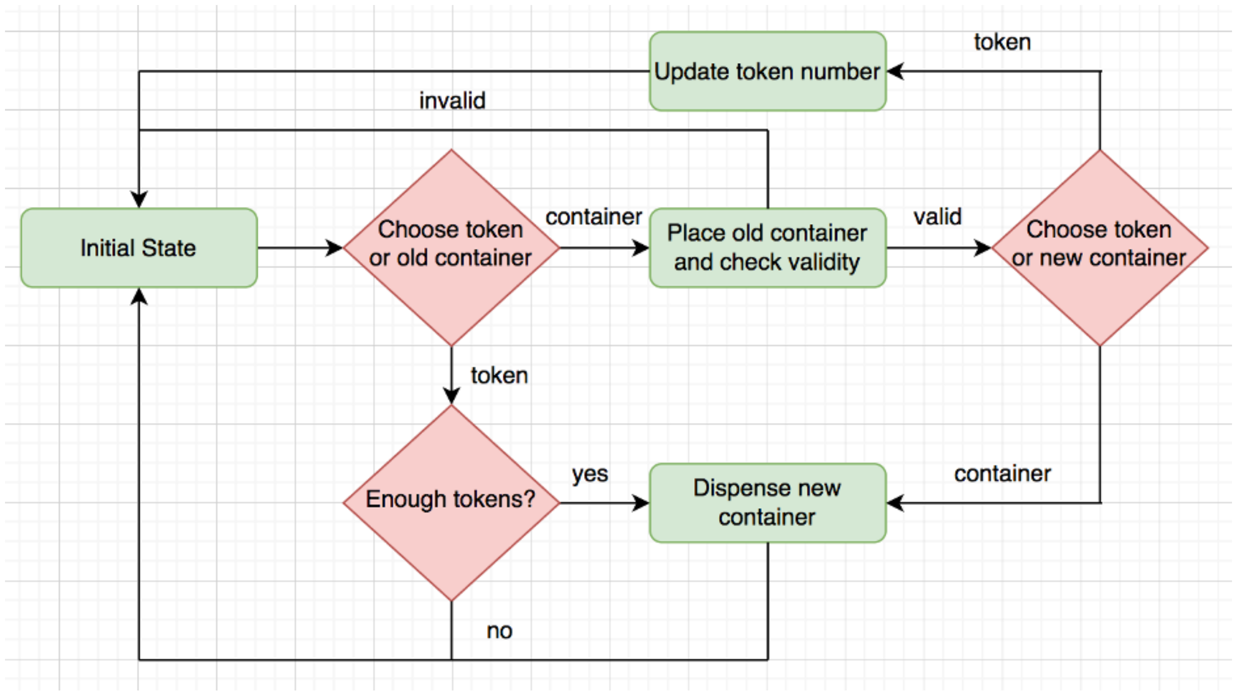


Figure 3: FSM of the G2G machine

2.1.1 UI Subsystem

The main design consideration for the user interface (UI) subsystem was between an LCD (liquid crystal display) screen and a multiplexer LED (light-emitting diode) status. An LCD screen had the advantage of displaying custom messages and easy to use libraries that had built in functions for an I2C or serial communication. However, the G2G machine had a similar functionality to that of a normal vending machine, and thus had little to no information to display to users. Most of the identity verification that happened under the hood was abstracted away and revealing sensitive information such as the student's UIN (university ID number) could have created security issues. Additionally, the project schematic used three microcontrollers (1 ATmega328P and 2 ATtiny85s) for managing the control logic with a limited number of digital pins, but even a relatively small 2x16 LCD screen required at most 14 pin connections [2]. An I2C adaptor could be used to cut down on the number of digital pins but incurred additional cost and was not included in the original PCB modification.

Thus, the final design decision was to use LED status messages, which also aided in visualizing the state transitions shown in Figure 3. Figure 4.1 below illustrates the schematic for the two push buttons as well as the LEDs that are connected to an 8-to-1 demux. The two push button pins labeled B0 and B1 correspond to a selection of token and container respectively, and are connected back to pin 23 and 24 of the ATmega. Each output of the demux made a serial connection with a 220Ω resistor before making a final connection with the positive anode of the LEDs to prevent excess current from burning any components [3].

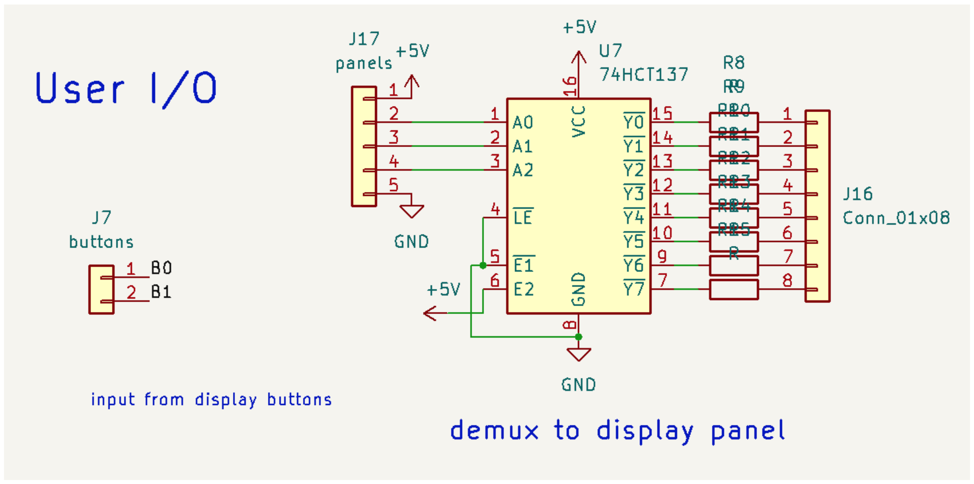
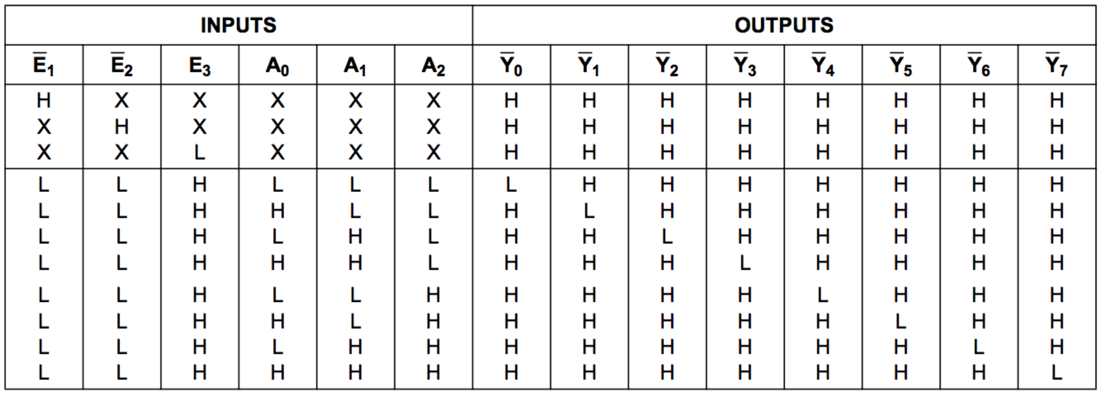


Figure 4.1: UI Schematic for Display and Buttons

The three select pins A0, A1, A2 controlled which output pin from Y0 to Y7 to set as LOW as shown in Figure 4.2 [4]. The HCT138 demux used was active low with NOR gates connected to each of the output gates. Thus, the G2G machine with LED status messages initially had all lights turned on until a state transition occurred to represent the current stage of the exchange.



INPUTS						OUTPUTS							
$\overline{E}_1$	$\overline{E}_2$	E_3	A_0	A_1	A_2	$\overline{Y}_0$	$\overline{Y}_1$	$\overline{Y}_2$	$\overline{Y}_3$	$\overline{Y}_4$	$\overline{Y}_5$	$\overline{Y}_6$	$\overline{Y}_7$
H	X	X	X	X	X	H	H	H	H	H	H	H	H
X	H	X	X	X	X	H	H	H	H	H	H	H	H
X	X	L	X	X	X	H	H	H	H	H	H	H	H
L	L	H	L	L	L	L	H	H	H	H	H	H	H
L	L	H	H	L	L	H	L	H	H	H	H	H	H
L	L	H	L	H	L	H	H	L	H	H	H	H	H
L	L	H	H	H	L	H	H	H	L	H	H	H	H
L	L	H	L	L	H	H	H	H	H	L	H	H	H
L	L	H	H	L	H	H	H	H	H	H	L	H	H
L	L	H	L	H	H	H	H	H	H	H	H	L	H
L	L	H	H	H	H	H	H	H	H	H	H	H	L

Figure 4.2: Function Table of 3-to-8 Demultiplexer (L = Low, H = High)

2.1.2 Sensing Subsystem

The sensing subsystem had a total of three different modules with QR scanner and load cell being used for the validation process of G2G containers while a magnetic card swipe was used for identity verification of the user. A popular choice for digital identification was using a RFID (radio-frequency identification) tag as it already had heavily detailed libraries compatible with Arduino code [5]. However, the G2G system was aimed to provide service exclusively to U of I students using their iCards. Thus, instead of introducing another layer of security ID, modifying a magnetic card reader suitable to read actual iCards was the main reason for such design choice.

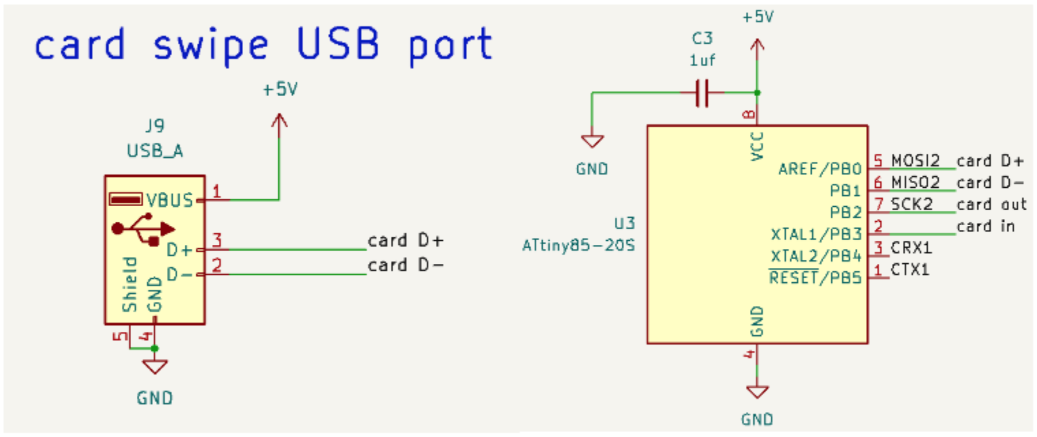


Figure 5.1: Sensor Schematic for Magnetic Card Swipe

The left component in Figure 5.1 shows the USB port of the magnetic card swipe with D+ and D- pins used to transfer data between the host (microcontroller) and the device (card reader). The card swipe had its own smaller microcontroller unit shown on the right of Figure 5.1 that connected back to the main microcontroller. The ATtiny was responsible for reading the raw card data while the main microcontroller ATmega decided on the validity of the user based on the input from the ATtiny.

In addition to the USB card reader, the G2G machine required a mechanism to detect the amount of food waste within the returned containers. Using a typical scale had two major issues as it was too big to fit in the downscaled machine and also did not have the accuracy to measure extremely light plastic containers. A load cell offered a solution to both of these problems as a small transducer to convert force into measurable electrical output. Although it required a calibration process for accurate measurements, using an amplifier allowed a digital reading of small weights. The specific type of the load cell used was a strain gauge bending beam load cell, which represented the load of an object placed with a degree of voltage change [6]. The bending beam load cells offered low-profile construction for limited space and were flexible at relatively low forces, making them suitable for lower capacity applications such as the G2G containers [7].

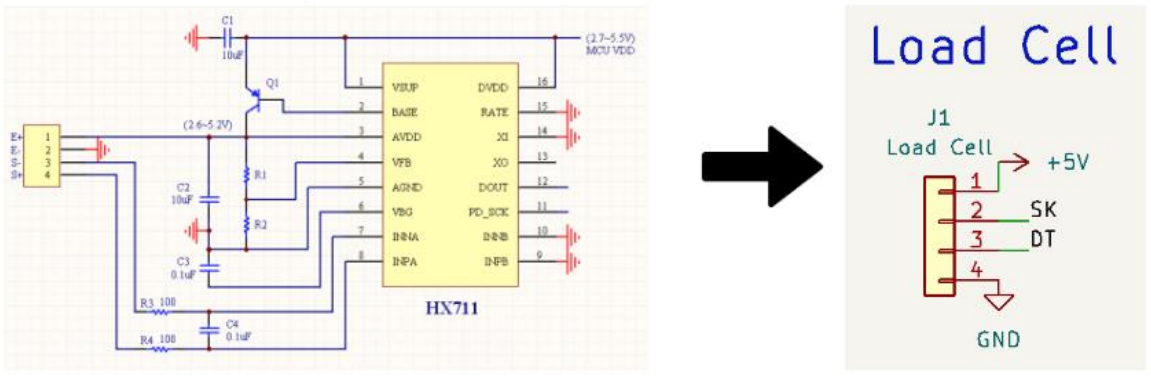


Figure 5.2: Sensor Schematic for Load Cell

The initial four wires labeled E+, E-, S-, S+ on the very left of Figure 5.2 were taken from the load cell and connected to the HX711 amplifier to amplify the output signal. The two outputs of HX711 labeled as PD_SCK and DOUT were then connected to SK and DT respectively, which stands for serial clock and data out [8]. The final output of the load cell was fed back to the main microcontroller at pin 15 with digital reading of the weight. The load cell was used to reject any containers that weighed more than 10*g* with the base weight of the G2G container being approximately 3*g*, allowing a maximum 7*g* of food waste.

Lastly, a QR scanner module was used to verify the object placed on the load cell as a valid G2G container through the use of QR codes. The main difference between QR codes and barcodes was the amount of information that could be stored as QR codes had another physical dimension from which data could be written and scanned [9]. The initial design decision was to use a barcode scanner as the machine required nothing more than a static ID verification for the G2G containers. However, barcodes required a linear scan and were thus susceptible to errors based on different orientations and placements. To allow more flexibility and account for possible human errors, QR scanner was used to offer a more stable solution.

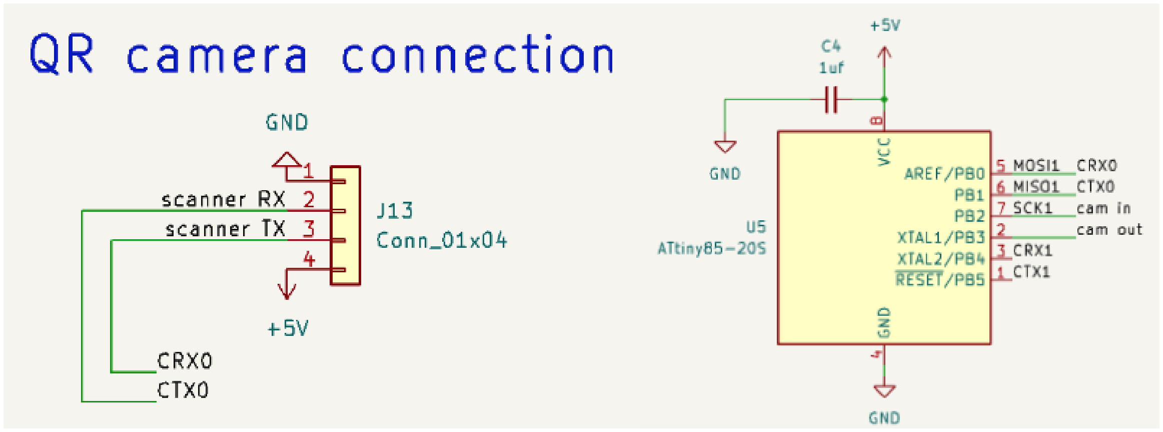


Figure 5.3: Sensor Schematic for QR scanner

The QR scanner module utilized a serial port for an UART communication with an ATtiny microcontroller using the Rx (receive) and Tx (transmit) pins as shown in Figure 5.3. The Rx pin of the QR was used to receive any data from the microcontroller, which was mostly composed of UART scan commands to change some of the default settings of the module. The Tx pin was used to send any read QR codes to the Rx pin of the microcontroller to determine a valid G2G container.

2.1.3 Control Subsystem

Once all the validation was complete by the sensing subsystem, the main microcontroller was responsible for sending any appropriate data to the motors, which were responsible for safe and accurate execution of dispensing and retrieving G2G containers.

The final product incorporated a non-continuous servo motor for the retrieval system and a continuous servo motor for the dispensing system. The two main reasons for using a servo motor instead of a stepper motor was because of the closed loop control it provided along with an increased torque at high speed. The stepper motor had an advantage of providing high torque at low speed and was easy to use [10]. However, the design of the control subsystem had significant weight placed on the motors for not only the mechanical arms attached to the shaft but also the G2G containers placed on top of the treadmill. This also induced frictional force that further cut down on the natural torque provided by the motor, making servo motors the ideal choice for the machine.

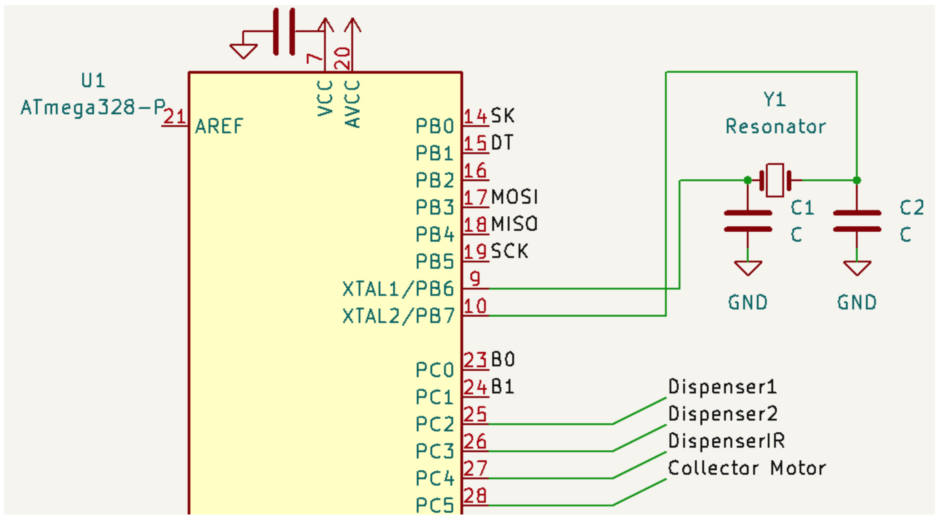


Figure 6: Control Schematic for Servo Motors

The two servo motors received commands from the main microcontroller regarding position data and pulse signals from pin 27 and 28 with pins 25 and 26 being unused. The non-continuous motor would turn its shaft based on the desired angular position sent from the ATmega while the continuous motor received information on both the direction of rotation as well as the RPM.

2.1.4 Power Subsystem

The machine was intended to draw power from a wall outlet, and thus required an AC/DC converter as a component. The 12*V* output of the converter was designed to feed into the voltage regulator as shown in Figure 7 to output a steady 5*V* supply to the rest of the modules.

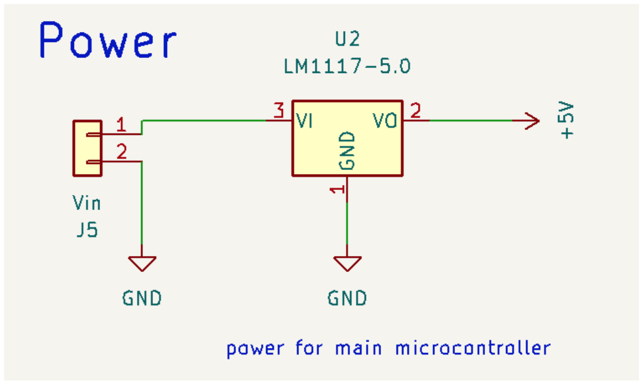


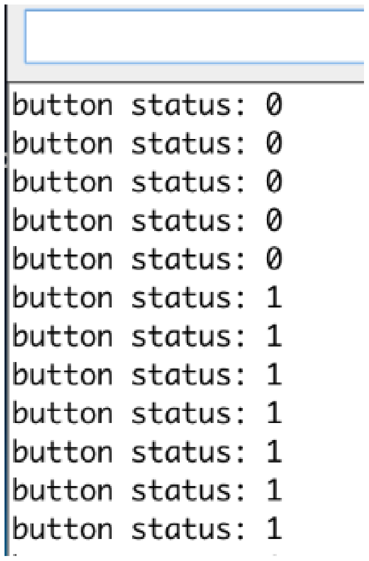
Figure 7: Power Schematic for Voltage Regulator

2.2 Design Details

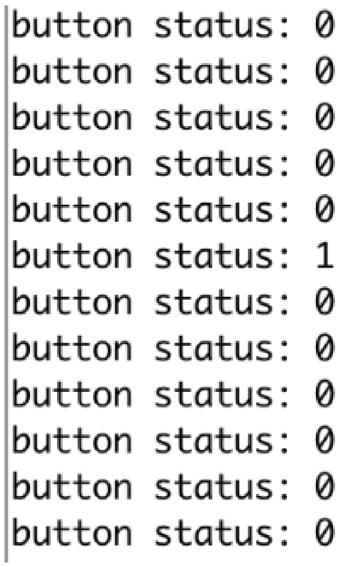
2.2.1 UI Subsystem

The two push buttons used for the machine operated in a maintained-contact mode that stayed on upon a press and turned off when pressed again. Thus, the change in the status of the push buttons made by the users were measured by the difference in the ON and OFF states rather than keeping track of when exactly the press was detected. This allowed multiple other processes with higher priority such as the USB task and UART serial communication to run while saving the initial state of the push button to detect a change when needed.

One small modification made from the original PCB design was the addition of a pull-down 10KΩ resistor for a serial connection with the push buttons. This was done as a fix for an incorrect digital reading of the button status as shown in the serial monitor outlined in Figure 8. The left side of Figure 8 shows the correct readings of the push button that switched state from OFF (0) to ON (1) upon a button press. However, the button status momentarily reflected an ON state when the push button remained unpressed as shown on the right side of Figure 8.



```
button status: 0
button status: 0
button status: 0
button status: 0
button status: 0
button status: 1
button status: 1
button status: 1
button status: 1
button status: 1
button status: 1
button status: 1
```



```
button status: 0
button status: 0
button status: 0
button status: 0
button status: 0
button status: 1
button status: 0
button status: 0
button status: 0
button status: 0
button status: 0
button status: 0
```

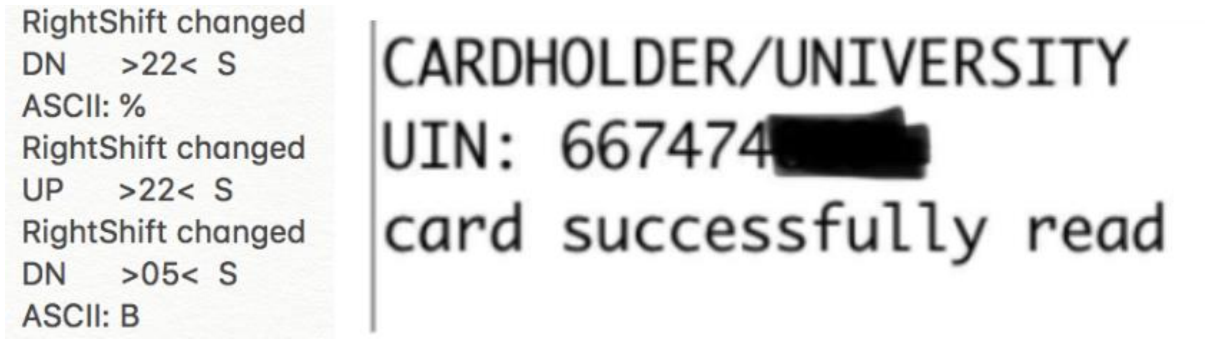
Figure 8: Serial Monitor Output of Button Status

This was due to a floating input in the microcontroller pin with no defined logic state, leaving the appearance of a button press that falsely triggered the circuit [11]. The use of a pull-down resistor forced the behavior of the push button to have a digital reading of LOW (0) when there was no connection between the two legs of the button while a digital reading of HIGH (1) when the connection was made from a press.

The LED status messages were used to represent the six different states of the FSM, and required writing to the A0, A1, and A2 digital pins of the demux to set the three select bits. The Y0 output was used to represent the initial state of the machine waiting for a card swipe, followed by Y1 to Y5 representing the select, retrieval, dispense, update, and invalid states respectively.

2.2.2 Sensing Subsystem

As one of the most complex parts of the project, the magnetic card reader required interfacing with a HID (human interface device) Keyboard as card swipes imitated the same interrupt behavior as that of a keyboard. Each card swipe responded to both key-down and key-up interrupts along with any special character key presses such as SHIFT and CAPS. The left image of Figure 9 below shows the raw data output read from the keyboard interrupts of a card swipe while the right image shows the output after modification in the driver code. The data parsing allowed each iCard swipe to identify as a university card and to recognize the UIN (university identification number) of the user.



```
RightShift changed
DN     >22<  S
ASCII: %
RightShift changed
UP     >22<  S
RightShift changed
DN     >05<  S
ASCII: B
```

```
CARDHOLDER/UNIVERSITY
UIN: 667474[redacted]
card successfully read
```

Figure 9: Output Data of Magnetic Card Swipe

The important parts of the load cell design consisted of determining the load cell resolution as well as the minimum weight measurable. The graph in Figure 10 expresses the relationship between the FSO (full scale output) and the amount of load on the load cell. Calculating such minimal weight was critical as the G2G containers needed accurate measurements for detecting small amounts of food waste. Equation (1) takes the maximum load to assess the output voltage (display's input sensitivity) required to give a readout of the smallest measurable weight [12].

$$\frac{max\ measuring\ range}{number\ of\ divisons\ \times\ max\ load} = \frac{output\ voltage}{excitation\ voltage} \tag{1}$$

Using the specification of the load cell from the data sheet in Equation (1) gave a final input sensitivity of 0.72μV, an upper bound of the output voltage to read a weight of 0.3g.

$$output\ voltage = \frac{1500g\ \times\ 3.6mV}{5000\ \times\ 1500g}\ =\ 0.72\ \mu V \tag{2}$$

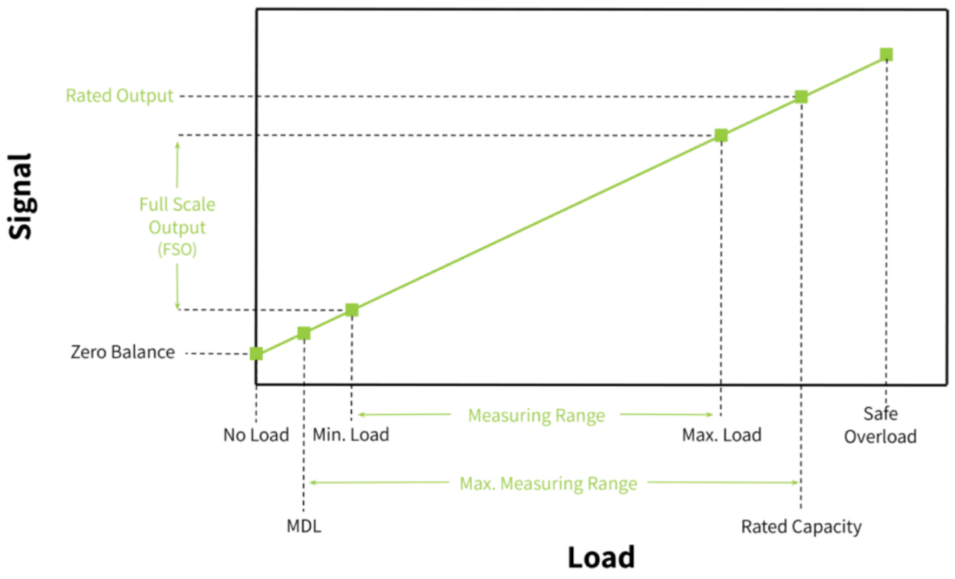


Figure 10: Relationship Graph Between Load and Load Cell Signal

The QR scanner had a built in image recognition algorithm and most of the work needed was changing the default behaviors of the module that were incompatible with the intended design of the machine. The module's baud rate had to be changed from 9600 to 115200 to be synchronized with the card reader and the scanning mode had to be changed to run continuously to not enter a sleep mode when inactive for a period of time. Sending specific UART scan commands in the datasheet allowed the QR scanner to enter a continuous scanning mode by setting bit0 to 1 at address 0x02 [13].

2.2.3 Control Subsystem

The two servo motors used for the retrieval and dispensing system operated in different modes. The non-continuous motor for retrieving received position data from the microcontroller, going from 0° to 120° with a delay of 10*ms* between each angle. This was done to minimize any safety issues as the retrieval motor was placed near the bin exposed to user contact.

The continuous motor for dispensing went through precise calibration as the functionality required a specific pulse of 1.5*ms* to stop the motor from rotating. Even a small drift would cause significant rotation over time, resulting in possible dispensing of new containers. The adjustment was made using a potentiometer with PWM (pulse width modulation) to send a 1.5*ms* pulse for the stopping condition of the servo motor [14].

2.2.4 Power Subsystem

The power subsystem plugged into a standard US wall outlet that could output 12*V* from 120VAC and could convert the voltage to the adequate DC voltage required by the other components in each subsystem. The converter had a built-in 10:1 transformer to step down from 120*V* to 12*V*, which was fed as an input to the voltage regulator to supply a steady 5*V*. The two stepper motors drew a lot of current and thus required a separate power supply, which was taken from a battery pack of 5*V*.

3. Verification

3.1 UI Subsystem

The requirement for the UI Subsystem was to correctly keep track of a user's token count within the magstripe card (Appendix A, Table 4.1). Another requirement was to have the system correctly keep track of any user input through the push buttons (Appendix A, Table 4.2). To verify both of these requirements, we exchanged a single container for a token in the first iteration, and used the token to exchange for a new container in the next iteration to verify a successful token update. In the final iteration, we attempted to spend a non-existent token to check the validation process of the user's token count, making sure that no new container was dispensed.

3.4 Sensing Subsystem

The requirement for the sensing subsystem was to have a successful completion of a QR scan and accurate weight measurement of the G2G containers within a $1g$ tolerance (Appendix A, Table 7.1). To verify the stated requirement, we checked for a valid QR code scan from the serial monitor and made comparisons of the load cell measure against a known container weight. The other requirement was to have the system correctly execute retrieval and dispensal for valid G2G containers (Appendix A, Table 7.2). This was verified by checking if the system rejected containers that were overweight ($> 10g$) or had an invalid or non-existent QR code while continuing the exchange process for all valid containers.

3.3 Control Subsystem

The requirement for the control subsystem was to have the G2G machine complete the entire exchange process within 15 seconds from start to finish, assuming minimal delay on the user's side (Appendix A, Table 6.1). To verify the stated requirement, we conducted several repetitions of completing an exchange, timing it from start to finish, ensuring that the maximum total time taken did not exceed 15 seconds. The other requirement was to have the machine dispense exactly one new container upon a dispensal request without any mechanical issues (Appendix A, Table 6.2). Similarly, this was also verified by conducting several repetitions of completing an exchange, ensuring no skew or drift in the motors were causing the machine to dispense a container more than once.

3.2 Power Subsystem

The power subsystem required each component to be powered in accordance within a 5% variance of the rated voltage outlined in Figure 1. We used a multimeter to check the connection between our $12V$ supplied power, ensuring that it delivered power to each voltage regulator first. The rest of the components were probed to check for a steady voltage supply of $5V$ with a maximum variance of 5% of the rated voltages (Appendix A, Table 5).

4. Costs & Schedule

4.1 Costs

Table 1: Cost Breakdown of Each Component

Part	Part Name	Price Per Unit	Quantity	Price
Push Button	GSW-21	$5.18	2	$10.36
White LED	57P7143	$0.47	6	$2.82
Red LED	HLMP1340	$0.39	1	$0.39
Servo Motor	HS-311	$13.54	1	$13.54
Servo Motor	FS5103R	$11.95	1	$11.95
Load Cell Sensor	TAL221	$9.95	1	$9.95
QR Scanner	SKU 14810	$49.99	1	$49.99
Microcontroller	X000048	$7.20	1	$7.20
MSR90 Magstripe Card Reader	MSR90	$20.23	1	$20.23
Total Cost:				$126.43

The labor costs for the team members were calculated from the 2020-2021 annual Illini Success Report [15]. Ariocie is an Electrical Engineering major with an average salary of $80,296. John and Henry are Computer Engineering majors with an average salary of $105,352. A 10 hours per week work for a total of nine weeks came out to be 90 hours of work per engineer. Thus, the total labor costs for an Electrical Engineering major was 1 engineer × 2.5 × $38.60/hr × 90 hours = $8,685 and for two Computer Engineering majors was 2 engineers × 2.5 × $50.65/hr × 90 hours = $22,792.50.

The current labor rate for machine shop workers in the state of Illinois is $33,000/year [16]. Skee Aldrich and Gregg Bennett worked 13 hours during the semester for this project. Thus, the total labor costs for two machine shop workers came out to be 2 workers × 2.5 × $15.87/hr × 13 hours = $1031.55. The total labor cost across the machine shop workers and engineers was $32,509.05.

Table 2: Labor Cost Breakdown

Name	John Kim	Ariocie Liang	Henry Guan	Machine Shop
Rate	$50.65	$38.60	$50.65	$15.87
Hours Worked	90	90	90	13

Labor Support Rate	$75.98	$57.90	$75.98	$23.81
Total Labor Costs	$11,396.25	$8,685.00	$11,396.25	$1,031.55

4.2 Schedule

Table 3: Weekly Schedule of Each Team Member

Week	**John Kim**	**Ariocie Liang**	**Henry Guan**
2/21	Complete Design Document.	Complete Design Document. Decide on parts to use.	Complete Design Document. Decide on parts to use.
2/28	Complete Design Review. Work on PCB for approval for first round orders.	Complete Design Review. Work on PCB for approval for first round orders.	Complete Design Review. Work on PCB for approval for first round orders.
3/7	Work on software code.	Solder and mount components to the PCB. Test PCB	Solder and mount components to the PCB. Test PCB
3/14	Spring Break	Spring Break	Spring Break
3/21	Continue working on software code.	Revise PCB for second round orders.	Work with John on software code.
3/28	Complete Individual Progress Report.	Complete Individual Progress Report.	Complete Individual Progress Report.
4/4	Address software issues with the G2G system.	Solder and mount components to the second PCB. Test PCB.	Solder and mount components to the second PCB. Test PCB.
4/11	Complete software code. Unify system for final testing.	Test additional hardware and address issues. Unify system for final testing.	Work with John on software code. Unify system for final testing.
4/18	Mock Demo. Begin final demo preparation.	Mock Demo. Begin final demo preparation.	Mock Demo. Begin final demo preparation.
4/25	Final Demo. Work on presentation and final paper.	Final Demo. Work on presentation and final paper.	Final Demo. Work on presentation and final paper.
5/2	Final Presentation and Final Paper	Final Presentation and Final Paper	Final Presentation and Final Paper

5. Conclusion

5.1 Accomplishments and Uncertainties

The final product of the G2G machine was able to meet all three of the high level requirements stated in Section 1.3 of the paper. The project provided a proof of concept design that provides several beneficial effects on the labor cost and user satisfaction of the G2G take-out system by improving efficiency and eliminating any human delay in the current system. The final product allows direct exchange between old containers and new containers as well as digitizing all user related information in the iCard. This further adds to the practicality of the machine, allowing current U of I students an easy access without any additional purchases of security IDs.

Despite having the full functionality achieved, the project was not able to meet some of the requirements for the individual subsystem specifications. The power subsystem required an AC/DC converter, but the product itself did not arrive on time to incorporate in the final design. The issue was partially remedied by having a voltage regulator that fulfilled half of the power subsystem requirements. Additionally, the full PCB schematic was not incorporated into the project as several critical components such as the USB cable and specific capacitors were missing due to miscommunication within the team.

5.2 Ethical Considerations

Throughout the duration of the project, we kept ethics and safety a top priority. In Code I1 of the IEEE Code of Ethics, it states that we must "hold paramount the safety, health, and welfare of the public" [17]. The code was closely applied to the initial goals of the project as it required creating a mechanism of retrieval and dispensing that was free of mechanical faults to minimize the possibility of physical injury.

Additionally, the automation of the current G2G container exchange system could have introduced a few security issues. With the aim of removing any human supervision, the sensor modules were required to achieve complete and accurate functionality. To avoid any possible malicious attempt to steal or damage the components of the machine, we ensured that all parts of our device were concealed and abstracted away from the user while running thorough testing of any mechanical faults. With how digitized the world is becoming, exposure of even the smallest personal information could lead to detrimental consequences. Thus, we also ensured that all user information including the UIN and digital tokens were confidential and protected. This was in alignment with codes 1.6 and 1.7 in the ACM Code of Ethics, which states that we must "respect privacy" and "honor confidentiality", respectively [18]. We vow to uphold any user's personal information and privacy by keeping all data internally protected.

Lastly, there were also safety concerns that applied to the development of our project when working inside the lab. In accordance with federal and state regulations as well as the campus lab policies, we made sure not to work alone in the lab, avoid bringing food or drinks, not remove any test equipment from the lab station, and clean up after ourselves so that the workspace was hazard-free [10].

5.3 Future Work

For the current G2G machine, it uses on-chip memory to store user information such as the token number, and thus has a limited number of 10,000 write cycles. Although the number may most likely be enough to keep track of all the current token updates, having a relational database using SQL could make the system more durable. Additionally, the growth of the G2G system may introduce different sizes and types of containers, and may require a strict deadline for users to return what they borrowed. These extra pieces of information can easily be added if the system utilizes a database to maintain the structure, also allowing greater scalability to be used in all dining halls.

Lastly, the use of a display screen could enhance user experience with on screen instructions and also present several selections rather than just two choices between tokens and containers. Although the LED status messages were more suitable for the current state of the machine, digital screens would allow greater flexibility to any potential design changes to the product.

References

[1] University Housing, “Good2Go carry-out program,” *University Housing*. [Online]. Available: https://housing.illinois.edu/Dining/Locations/Good2Go-Carry-Out-Program. [Accessed: 30-Apr-2022].

[2] T. A. Team, “Liquid Crystal Displays (LCD) with Arduino: Arduino documentation,” *Arduino Documentation | Arduino Documentation*. [Online]. Available: https://docs.arduino.cc/learn/electronics/lcd-displays. [Accessed: 03-May-2022].

[3] “Resistors for led circuits: Resistor applications: Resistor Guide,” *EEPower*. [Online]. Available: https://eepower.com/resistor-guide/resistor-applications/resistor-for-led/#. [Accessed: 03-May-2022].

[4] *Datasheet 74HC/HCT138*. [Online]. Available: https://www.digchip.com/datasheets/parts/datasheet/364/74HC_HCT138-pdf.php. [Accessed: 03-May-2022].

[5] “RFID scanner - full tutorial,” *Arduino Project Hub*. [Online]. Available: https://create.arduino.cc/projecthub/shubamtayal/rfid-scanner-full-tutorial-6518db. [Accessed: 03-May-2022].

[6] O. Engineering, “Load Cells & Force sensors,” *https://www.omega.com/en-us/*, 29-Apr-2022. [Online]. Available: https://www.omega.com/en-us/resources/load-cells#:~:text=Strain%20gauge%20load%20cells%20are%20a%20type%20of%20load%20cell,gauge%20when%20it%20undergoes%20deformation. [Accessed: 03-May-2022].

[7] “How does a bending beam load cell work?: Working Principle,” *HBM*, 09-Sep-2019. [Online]. Available: https://www.hbm.com/en/2973/how-does-a-bending-beam-load-cell-work/. [Accessed: 03-May-2022].

[8] “Tal221 miniature load cell - cdn.sparkfun.com.” [Online]. Available: https://cdn.sparkfun.com/assets/9/9/a/f/3/TAL221.pdf. [Accessed: 02-May-2022].

[9] “QR code vs Barcode: Why the difference matters,” *Paysley Blog*, 10-Feb-2022. [Online]. Available: https://paysley.com/blog/qr-code-vs-barcode/. [Accessed: 03-May-2022].

[10] “Stepper vs Servo,” *Tutorial: Stepper vs Servo*. [Online]. Available: https://www.amci.com/industrial-automation-resources/plc-automation-tutorials/stepper-vs-servo/. [Accessed: 03-May-2022].

[11] D. Lancaster, “Do push buttons need resistors?,” *Electronic Guidebook*, 08-Jan-2021. [Online]. Available: https://electronicguidebook.com/do-push-buttons-need-resistors/. [Accessed: 03-May-2022].

[12] “What is the lowest weight a load cell can measure?,” *Tacuna Systems*, 31-Mar-2022. [Online]. Available: https://tacunasystems.com/knowledge-base/load-cell-lowest-weight/. [Accessed: 03-May-2022].

[13] *Waveshare Barcode Scanner Module*. [Online]. Available: https://www.waveshare.com/w/upload/archive/3/3c/20180623102926%21Barcode_Scanner_Module_User_Manual_EN.pdf. [Accessed: 02-May-2022].

[14] Adafruit, *Continuous Rotation Servo - FeeTech FS5103R*. [Online]. Available: https://media.digikey.com/pdf/data%20sheets/adafruit%20pdfs/154_web.pdf. [Accessed: 02-May-2022].

[15] "Part of University of Illinois," *Box*. [Online]. Available: https://uofi.app.box.com/s/aoply09y5kf6i36es8v3bl758n2lfl08. [Accessed: 02-May-2022].

[16] Zippia, "Average Machine Shop Worker Salary in Illinois," *AVERAGE MACHINE SHOP WORKER SALARY*. [Online]. Available: https://www.zippia.com/machine-shop-worker-jobs/salary/. [Accessed: 02-May-2022].

[17] "IEEE code of Ethics," *IEEE*. [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html. [Accessed: 02-May-2022].

[18] "ACM Code of Ethics," *Code of Ethics*. [Online]. Available: https://www.acm.org/code-of-ethics. [Accessed: 02-May-2022].

Appendix A - Requirement and Verification Tables with Results

Table 4: UI Subsystem R&V Table

Requirement	Verification	Verification Status
1. Correctly keep track of a user's token count.	1. Test the system to see if it can detect if the user has invalid tokens, as well as if the system is able to store a token and use said token.	Yes, refer to Figure 11, Appendix B.
2. Correctly keep track of user input.	1. Test to see if the system executes the intended action on the push button the user selects.	Yes, refer to Figure 12, Appendix B.

Table 5: Power Subsystem R&V Table

Requirement	Verification	Verification Status
1. Power all components in accordance with their rated voltage within 5% variance.	1. Use a multimeter for each component, ensuring that each component's measured voltage is within 5% variance of its' rated voltage	Yes, refer to Table 8, Appendix B.

Table 6: Control Subsystem R&V Table

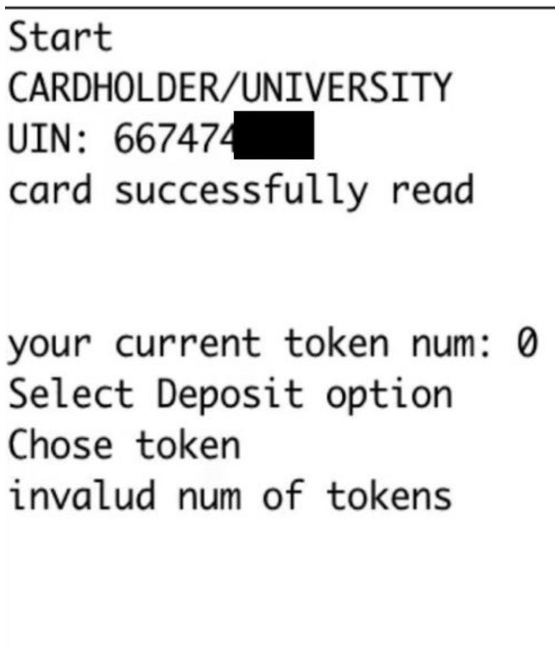
Requirement	Verification	Verification Status
1. Machine should complete an exchange within 15 seconds, from start to finish.	1. Utilize repetitive tests and time how long it takes to complete an exchange.	Yes.
2. Dispensing exactly one container upon receiving a dispensal request without any mechanical issues.	1. Repetitively complete exchanges for new containers, and check to see if exactly one container is dispensed.	Yes.

Table 7: Sensing Subsystem R&V Table

Requirement	Verification	Verification Status
1. Successful completion of a QR scan and accurate weight measurement of the G2G	1. Check to see if QR enables a correct scan and that the load cell accurately measures the known mass of a container.	Yes, refer to Figure 14, Appendix B.

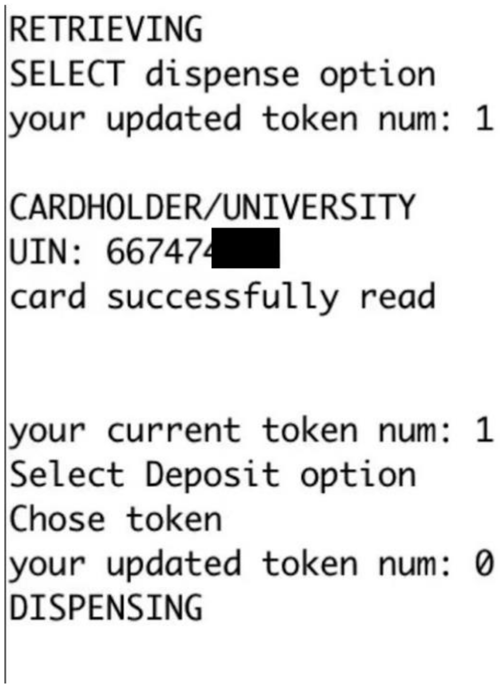
containers within a 1*g* tolerance.		
2. Correctly execute retrieval and dispensal only for valid G2G containers	1. Reject containers that are overweight or have an invalid/no QR code, while executing retrieval or dispensal for valid containers.	Yes, refer to Figure 15.

Appendix B – Verification Outputs



```
Start
CARDHOLDER/UNIVERSITY
UIN: 667474[redacted]
card successfully read


your current token num: 0
Select Deposit option
Chose token
invalud num of tokens
```

Figure 11: Serial Monitor Output of Invalid Token Use

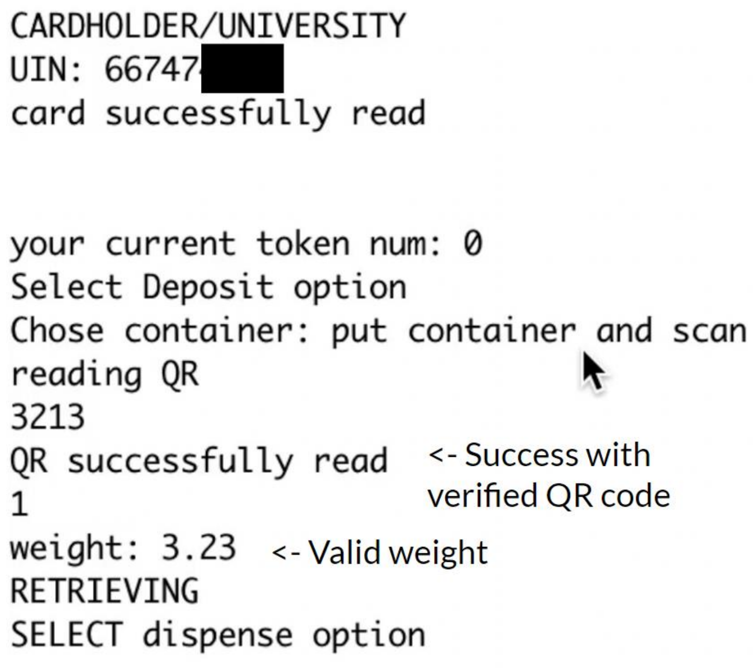


```
RETRIEVING
SELECT dispense option
your updated token num: 1

CARDHOLDER/UNIVERSITY
UIN: 667474[redacted]
card successfully read


your current token num: 1
Select Deposit option
Chose token
your updated token num: 0
DISPENSING
```

Figure 12: Serial Monitor Output of Token Usage

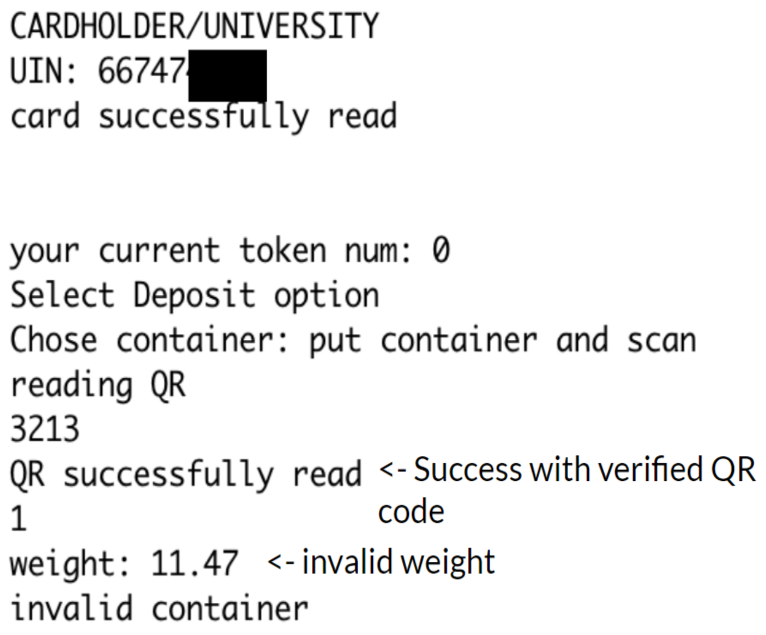
Component	Measured Voltage Rating
ATMega328P	4.997V
ATTiny85	4.996V
Load Cell	4.988V
Push Buttons	4.928V
Servo Motors	4.946V
Card Reader	4.898V
QR Scanner	3.295V
LEDs	3.323V

Table 8: Multimeter Readings of PCB Components



```
CARDHOLDER/UNIVERSITY
UIN: 66747[redacted]
card successfully read


your current token num: 0
Select Deposit option
Chose container: put container and scan
reading QR
3213
QR successfully read    <- Success with
1                          verified QR code
weight: 3.23  <- Valid weight
RETRIEVING
SELECT dispense option
```

Figure 14: Serial Monitor Output of Valid Container Detection



```
CARDHOLDER/UNIVERSITY
UIN: 66747[redacted]
card successfully read


your current token num: 0
Select Deposit option
Chose container: put container and scan
reading QR
3213
QR successfully read <- Success with verified QR
1                    code
weight: 11.47  <- invalid weight
invalid container
```

Figure 15: Serial Monitor Output of Invalid Container Detection

Appendix C – PCB Layout

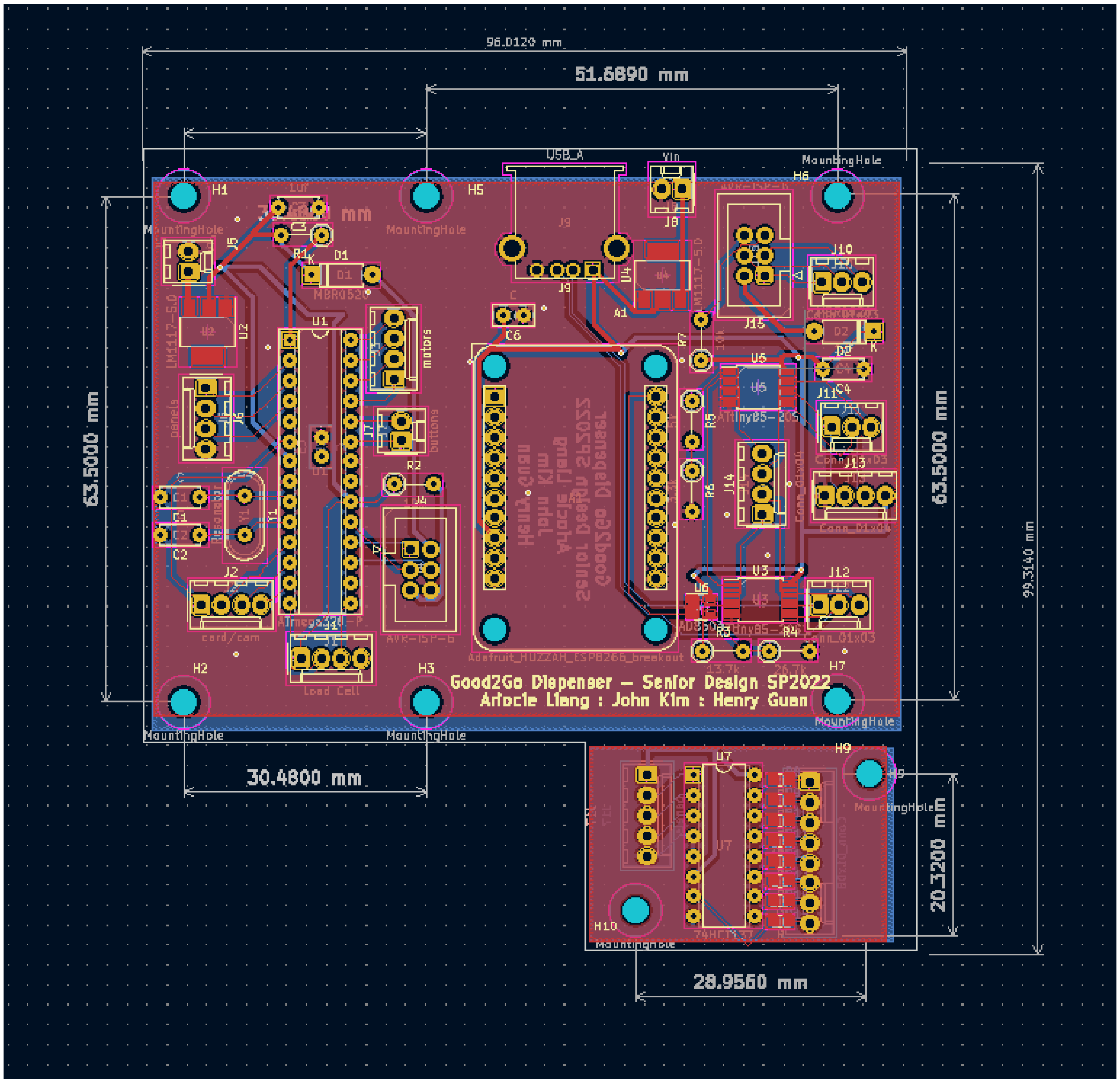


Figure 16: PCB Layout of the G2G Machine

Appendix D – PCB Schematic

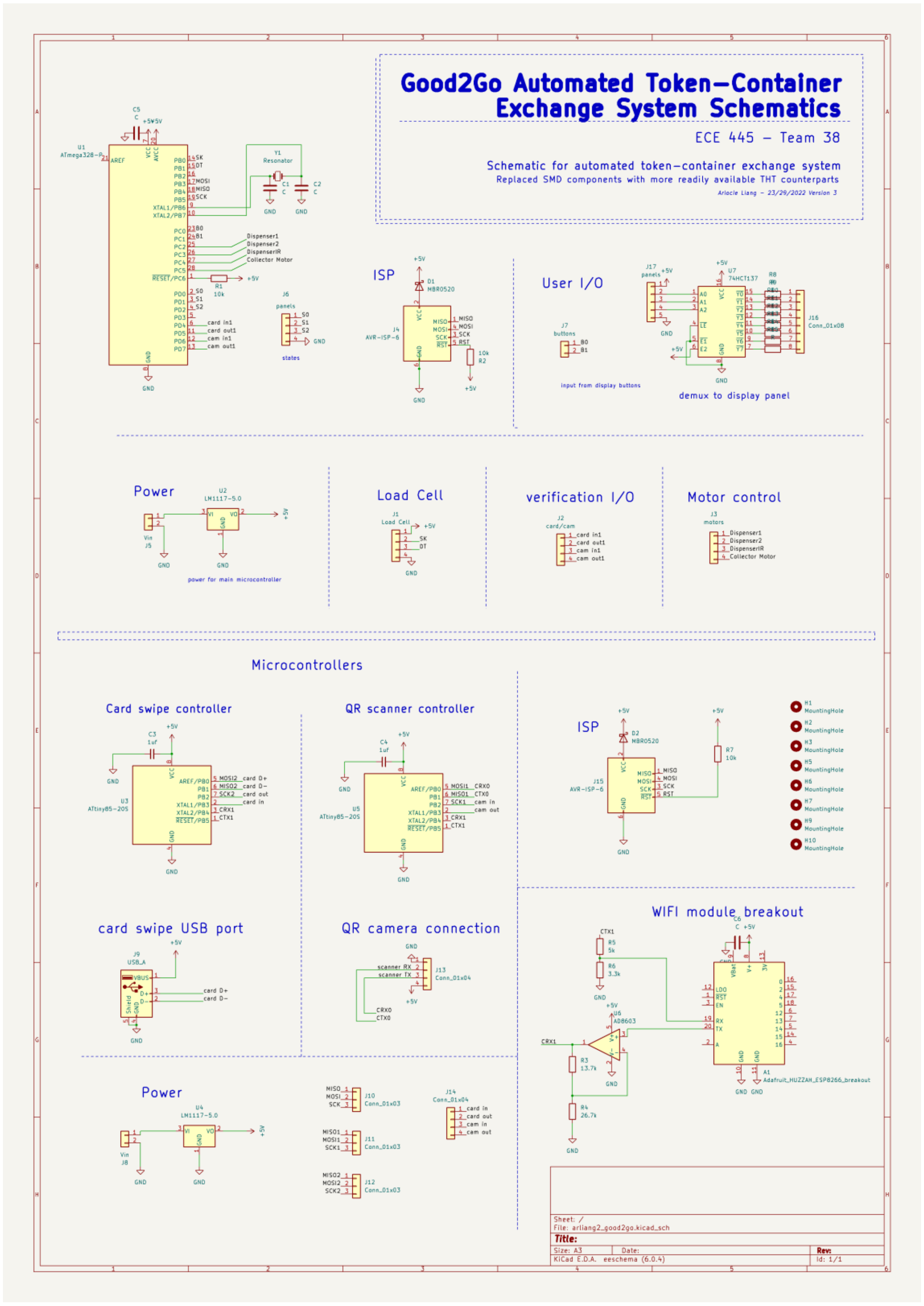

Good2Go Automated Token–Container Exchange System Schematics
ECE 445 – Team 38
Schematic for automated token–container exchange system
Replaced SMD components with more readily available THT counterparts
Arlocle Liang – 23/29/2022 Version 3
ISP
User I/O
demux to display panel
input from display buttons
Power
power for main microcontroller
Load Cell
verification I/O
Motor control
Microcontrollers
Card swipe controller
QR scanner controller
ISP
card swipe USB port
QR camera connection
WIFI module breakout
Power
ATmega328-P
ATtiny85-20S
LM1117-5.0
74HCT137
AD8603
Adafruit_HUZZAH_ESP8266_breakout
MountingHole
Sheet: /
File: arliang2_good2go.kicad_sch
Title:
Size: A3
Date:
Rev:
KiCad E.D.A. eeschema (6.0.4)
Id: 1/1

Figure 17: PCB Schematic of the G2G Machine

Appendix E – Software Code

```
// for card reader - HID boot keyboard
#include <hidboot.h>
#include <usbhub.h>

// Satisfy the IDE, which needs to see the include statment in the ino too.
#ifdef dobogusinclude
#include <spi4teensy3.h>
#endif
#include <SPI.h>

#include "HX711.h"
#include <Servo.h>
#include <SoftwareSerial.h>
#include <EEPROM.h>
// #include <LiquidCrystal.h>

// loadcell
#define calibration_factor -7050.0
#define DOUT 4
#define CLK 3

HX711 scale;

Servo servoD; // continuous servo for dispensing
Servo servoR; // non-continuous servo for retrieving

SoftwareSerial qr(0, 1);  // Rx, Tx

// LCD Display
//int contrast = 60;
//const int v0 = A2, rs = 7, en = 8, d4 = 9, d5 = 10, d6 = 11, d7 = 12;
//LiquidCrystal lcd(rs, en, d4, d5, d6, d7);


String qr1 = "3213";
String qr2 = "3214";
String qr3 = "3212";

const int buttonL = 8;
const int buttonR = 7;

const int servoD_pin = 2;
const int servoR_pin = 6;

const unsigned long timeout = 10000; // timeout value for QR set to 2 seconds

int a0 = A0;
int a1 = A1;
int a2 = A2;

int pos = 0;
int num_tokens = 2;
bool card_valid = false;
bool start = false;

// card reader
String s = "";
bool card_read = false;
bool company = false;
bool unused = false;
bool uin = false;

String John = "667474863";

struct student {
  String netID;
  int addr;
  int tokenNum;
};

student jaehank2;
student henryg3;
student* studentList[2] = {&jaehank2, &henryg3};
int idx = 0;

class KbdRptParser : public KeyboardReportParser
{
    void PrintKey(uint8_t mod, uint8_t key);

  protected:
    void OnControlKeysChanged(uint8_t before, uint8_t after);

    void OnKeyDown  (uint8_t mod, uint8_t key);
    void OnKeyUp  (uint8_t mod, uint8_t key);
    void OnKeyPressed(uint8_t key);
};

void KbdRptParser::PrintKey(uint8_t m, uint8_t key)
{
  return;
};

void KbdRptParser::OnKeyDown(uint8_t mod, uint8_t key)
{
  uint8_t c = OemToAscii(mod, key);

  if (c)
    OnKeyPressed(c);
}

void KbdRptParser::OnControlKeysChanged(uint8_t before, uint8_t after) {

  return;

}

void KbdRptParser::OnKeyUp(uint8_t mod, uint8_t key)
{
  return;
}

void KbdRptParser::OnKeyPressed(uint8_t key)
{
  s += ((char)key);
  if (card_read == true && s.length() == 17){
    Serial.println();
    s = "";
    card_read = false;
    company = false;
    unused = false;
    uin = false;
    return;
  }
  else if (company == false && s.length() == 19){
    // Serial.println(s);
    s = "";
    company = true;
  }
  else if (company == true && s.length() == 21){
    Serial.println(s);
    s = "";
    unused = true;
  }
  else if (s.length() == 14 && unused == true && card_read == false){
    s = "";
    uin = true;
  }
  else if (s.length() == 9 && uin == true && card_read == false){
    Serial.print("UIN: ");
    Serial.println(s);
    if (s == jaehank2.netID){
      idx = 0;
      card_valid = true;
    }
    else if (s == henryg3.netID){
      idx = 1;
      card_valid = true;
    }
    else {
      card_valid = false;
    }
    card_read = true;
    s = "";
  }
};

USB     Usb;
//USBHub     Hub(&Usb);
HIDBoot<USB_HID_PROTOCOL_KEYBOARD>    HidKeyboard(&Usb);

KbdRptParser Prs;

void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
  #if !defined(__MIPSEL__)
    while (!Serial); // Wait for serial port to connect - used on Leonardo, Teensy and other boards with built-in USB CDC serial connection
  #endif
  qr.begin(115200);
  Serial.println("Start");
  if (Usb.Init() == -1){
    Serial.println("OSC did not start.");
  }
  delay( 200 );
  HidKeyboard.SetReportParser(0, &Prs);
  // initialize student info
  jaehank2.netID = "667474863";
  jaehank2.addr = 0;
  jaehank2.tokenNum = EEPROM.read(jaehank2.addr);

  henryg3.netID = "659439620";
  henryg3.addr = 1;
  henryg3.tokenNum = EEPROM.read(henryg3.addr);

  // initialize push buttons
  pinMode(buttonL, INPUT);
  pinMode(buttonR, INPUT);
  // have all LEDs on
  setLED(255, 255, 255);
  // initialize non-continuous servo motor
  servoR.write(0);
  servoR.attach(servoR_pin);
  servoR.write(0);
  // initialize scale
  scale.begin(DOUT, CLK);
  scale.set_scale(calibration_factor);
  scale.tare(); // Assuming there is no weight on the scale at start up, reset the scale to 0

}

void loop() {
  // put your main code here, to run repeatedly:

  // save initial button status
  int b1 = digitalRead(buttonL);
  int b2 = digitalRead(buttonR);

//  Serial.println(b1);
//  Serial.println(b2);
  Usb.Task();

  start = true;

  if (card_valid == true){
    Serial.println("card successfully read");
  }

  // TODO: set LED status for IDLE/init - Y0
  setLED(0, 0, 0);

  while (card_valid == true){
    // TODO: set LED status for SELECT DEPOSIT - Y1
    setLED(255, 0, 0);

    if (start == true){
      start = false;
      Serial.println();
      Serial.println();
      studentList[idx]->tokenNum = EEPROM.read(studentList[idx]->addr);

      Serial.print("your current token num: ");
      Serial.println(studentList[idx]->tokenNum);
      Serial.println("Select Deposit option");
    }

    // get number of tokens for the user
    // check pushbutton for using token
    if (b1 != digitalRead(buttonL)){
      Serial.println("Chose token");
      card_valid = false;
      // if not enough tokens, exit
      if (studentList[idx]->tokenNum <= 0){
        Serial.println("invalud num of tokens");
        // TODO: set LED status for invalid tokens - Y5
        setLED(255, 0, 255);
        delay(2000);

        break;
      }
      // else update token and dispense
      setLED(0, 0, 255);  // updated - Y4
      delay(2000);
      studentList[idx]->tokenNum -= 1;
      EEPROM.write(studentList[idx]->addr, studentList[idx]->tokenNum);

      Serial.print("your updated token num: ");
      Serial.println(studentList[idx]->tokenNum);
      // TODO: set LED status for dispense - Y3
      setLED(255, 255, 0);

      dispense();
      break;
    }

    // check pushbutton for returning old container
    if (b2 != digitalRead(buttonR)){
      setLED(255, 255, 255);  // Y7
      Serial.println("Chose container: put container and scan");
      card_valid = false;
      bool qrValid = QRcheck();
      Serial.println(qrValid);
      float weight = scale.get_units(5);
      Serial.print("weight: ");
      Serial.println(weight);

      if (weight <= 5.0 && qrValid == true){
        // TODO: set LED status for retrieving - Y2
        setLED(0, 255, 0);

        retrieve();
  //      b1 = digitalRead(buttonL);
  //      b2 = digitalRead(buttonR);
        // TODO: set LED status for SELECT DISPENSE - Y1
        setLED(255, 0, 0);

        Serial.println("SELECT dispense option");

        // spin until button pressed
        while (b1 == digitalRead(buttonL) && b2 != digitalRead(buttonR));

        // if user pressed token for dispense
        if (b1 != digitalRead(buttonL)){
          studentList[idx]->tokenNum += 1;
          EEPROM.write(studentList[idx]->addr, studentList[idx]->tokenNum);
          Serial.print("your updated token num: ");
          Serial.println(studentList[idx]->tokenNum);
          // TODO: set LED status for token updated - Y4
          setLED(0, 0, 255);
          delay(2000);

          break;
        }
        // else user pressed new container for dispense
        // TODO: set LED status for dispense - Y3
        setLED(255, 255, 0);

        dispense();
        break;

      }
      // invalid container
      else {
        Serial.println("invalid container");
        // TODO: set LED status for invalid container - Y5
        setLED(255, 0, 255);
        delay(2000);
        if (qr.available()){
          while (qr.available()){
            int t = qr.read();
            delay(5);
          }
        }

        break;
      }
    }
  }

}

void dispense() {
  Serial.println("DISPENSING");
//   return;
  servoD.attach(servoD_pin);
  servoD.write(0);  // rotate CW at full speed
  delay(2500);
  servoD.detach();
}

void retrieve() {
  Serial.println("RETRIEVING");
//   return;
  for (pos = 0; pos <=110; pos++){
    servoR.write(pos);
    delay(10);
  }
  for (pos = 110; pos >= 0; pos--){
    servoR.write(pos);
    delay(10);
  }
}

bool QRcheck() {
  unsigned long startTime = millis(); // timer starts
  String val = "";
  while (millis() - startTime < timeout){
    if (qr.available()){
      Serial.println("reading QR");
//      String val = "";
      while (qr.available()){
        int input = qr.read();
        val += input;
        delay(5);
      }
      Serial.println(val);
      if (val.length() > 4 || val == qr1 || val == qr2 || val == qr3) {
        Serial.println("QR successfully read");
        return true;
      }
      break;
    }
  }
  Serial.println("timed out");
  return false;
}

void setLED(int x, int y, int z){
  analogWrite(a0, x);
  analogWrite(a1, y);
  analogWrite(a2, z);
}
```