MED-I-CAN

By

Alyssa Dias

Kyle Hand

Eric Shen

Final Report for ECE 445, Senior Design, Fall 2018

TA: Dongwei Shi

12 December 2018

Project No. 26

Abstract

Med-I-Can is an automatic pill dispensing device targeted at nursing homes and assisted living communities. It allows for a caregiver to set a schedule for their patient or patients and verify that the patients are taking their medication, without having to explicitly sort medications into daily doses. Through several verification methods, Med-I-Can is able to accurately dispense and record medication information with certainty.

Contents

1. Introduction

1.1 Objective

A common issue among the disabled and elderly communities, is the need for taking a wide variety of medications at least once a day. Typically, multiple patients depend on much fewer caregivers (i.e.: group homes, nursing homes, assisted living spaces); this becomes a time-consuming, tedious task, and can be detrimental if the incorrect medications are administered.

Our goal is to create a solution that will simplify the task for caregivers, allowing patients to take a step closer to independence, reduce the risk of human error, and give family and caregivers more time to interact with their patients. Our idea is to create an app-controlled physical system (similar concept to a vending machine) that can serve an individual, but is scalable to multiple people. Our idea sets itself apart by allowing one caregiver to manage medication for multiple users, to create preset dosages/timing for each patient, and modify based on the well-being of each individual (for example, if an individual has a cold, they can add that preset of specific cold medication to the regular medication routine).

1.2 Background

A partial solution to this problem exists by giving timed dosages, but does not allow for a caregiver to adjust medicine types or dosages. It additionally requires all the pills to be sorted and grouped by the caregiver manually. This only solves the timing issue, but doesn't eliminate the time-consuming process of sorting, and allows for human error.

Other solutions that exist have no timers and rely on one of two things: the patient remembering to take their medication at the appropriate time, or assistance from the caregiver.

Our solution sets itself apart from competitors solutions by allowing the caregiver to simply deposit pills into our predefined containers, and then, using simple interface, program the patient's pill schedule into the app. In order to succeed with our project, we must be able to provide a platform that can accurately, and dependably, dispense a patient's medication according to the caregiver's desired schedule.

2 Design

2.1 Block Diagram

Figure 1 shows the block diagram for our solution. It's broken into six main sections: Mechanical, Power, Processing, Bluetooth, Sensors, and the App. These were designed and built independently, then integrated together. This design satisfies the need to be a programmable device with a schedule, one that can dispense accurately, and fulfill safety checks, such as not dispensing a new round of medications if all the prior medication has been removed. It incorporates systems that are scalable from an individual user to multiple, and allows for simple control/caregiver input via the app.

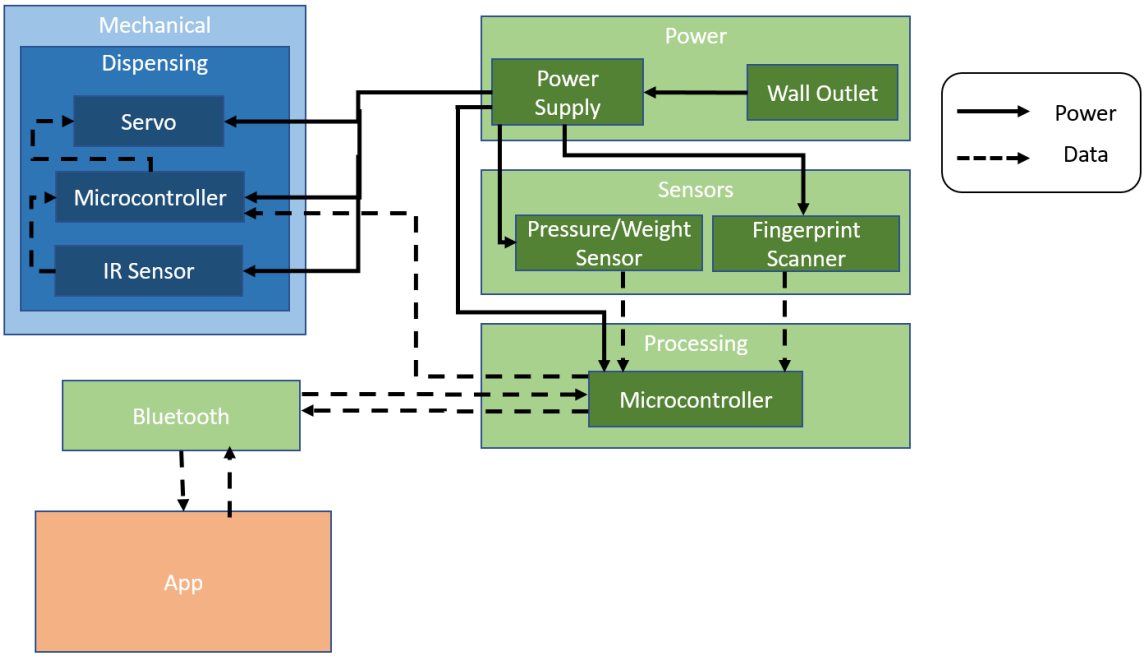

Figure 1 - Block Diagram

2.2 App

The User App is the main interface for the caregiver that sets up the device. It has been laid out and implemented in Python, using the tkInter library, to communicate with the rest of the sub systems. Using Bluetooth through a UART connection with the main PCB, the User App sends information regarding the patient's pill schedule, making changes and updates where necessary, as well as receives and view a history of the patient's activity using Med-I-Can, including missed doses. This module relies heavily on connecting to the main control portion of the project through Bluetooth.

The primary concern with the caregiver app was creating a simple user interface that was simultaneously feature rich. To do this, we implemented a single page, per user, application where all information regarding the patient could be accessed easily without the need to scroll or navigate to another page. Figure 2 shows the exact layout of the caregiver interface and the simplicity of changing,

updating, adding, and deleting medications as well as keeping detailed notes and viewing medication history logs.

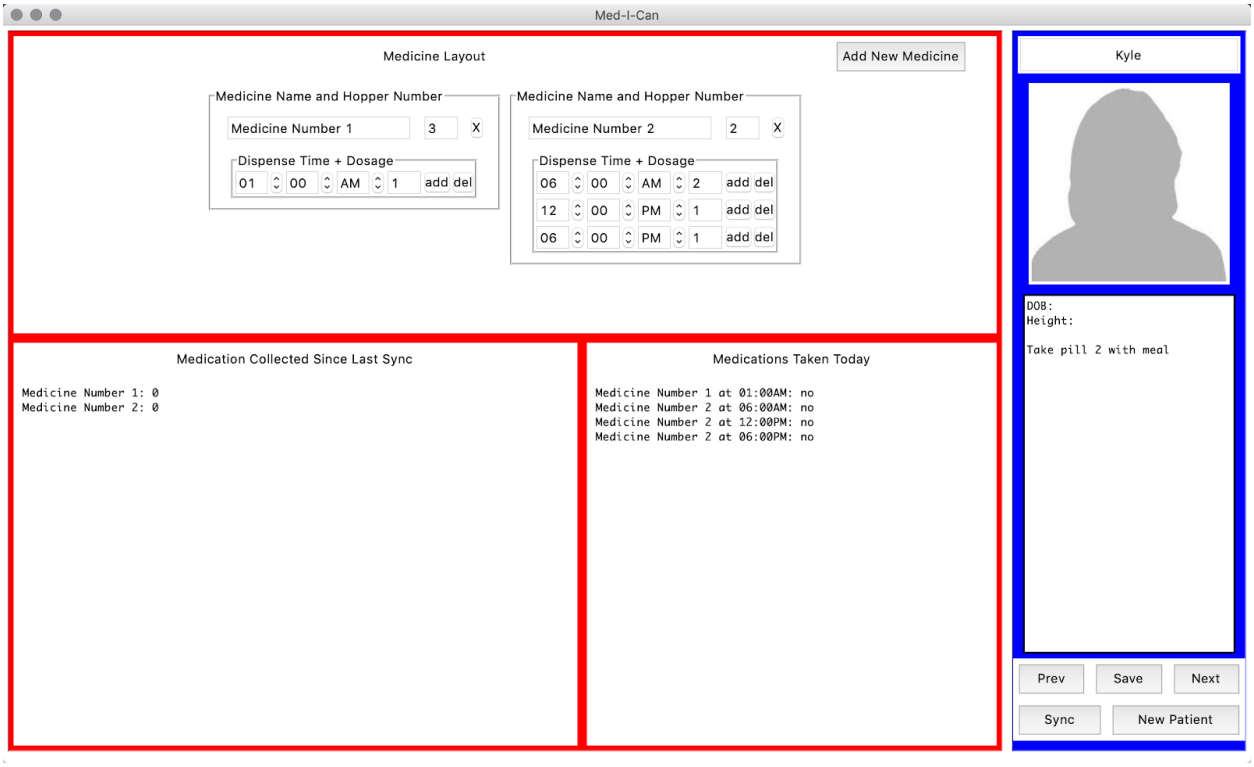

Figure 2 - Caregiver App Screen

2.3 Bluetooth

The Bluetooth module acts as a convenient method of allowing the caretaker to exchange information between the app and the main device. This includes adding new patients, updating to the medication each patient requires and retrieving the history of past dispenses. We chose the RN-42 due to the ease of use of UART and price considerations. However, throughout the process of working on the project, we found that implementing a server and using a Wi-Fi module would better suit our needs. This would allow the caretaker to sync information with the device from anywhere, as well as receive real time notifications from the device if there are any issues. The device is also equipped with a real time clock, or RTC, to determine the current time. Allowing the device to connect to the internet would alleviate our need for the RTC.

2.4 Processing

2.4.1 Microcontroller

The microcontroller in the processing unit acts as the main control for the device. The controller handles initializing and processing all other sensors and modules, saving patient information, syncing with the app, and communicating with the dispensing unit to signal what medication must be dispensed. We

chose to use the ATMEGA328 with Arduino boot-loaded because it was cost efficient and had specifications sufficient to meet our needs, as our project didn't involve any strenuous computations. In addition, the boot-loaded Arduino gave us access to many libraries that helped us interface with the various sensors we used, such as the load cell or fingerprint reader. The ATMEGA328 performed very well for us. However, in the future, we would also add external storage to the device, as we found the amount of internal storage that we could use on the microcontroller insufficient if we wanted the device to care for many patients (10+).

After initialization, the dispensing process begins with the patient interacting with the fingerprint scanner. The corresponding user's medical information is then retrieved from memory. The controller then determines which medications need to be dispensed and if the current time is close enough to the scheduled dosage time. If the medication meets all requirements, the address of the servo containing the appropriate medication is then sent to the microcontroller in the dispensing unit through UART. After which, the dispensing unit then dispenses a single pill and returns an ACK signal. Upon receiving the ACK signal, the main controller then repeats the process until the required number of dosages have been dispensed and no other medications that meet all conditions remain. The device is only ready to intake another patient's fingerprint once the cup containing the medication has been taken, emptied and replaced back into the device.

2.5 Sensors

2.5.1 Fingerprint Scanner

The fingerprint scanner acts as our method of patient identification. In our initial design we used a RFID card reader for patient recognition, however there were concerns about patients potentially misplacing their identification cards. As such, we decided to instead use a fingerprint scanner instead. The fingerprint scanner works through software serial and simply returns a unique user ID for each patient. This user ID is then used by the controller to determine where the patient data is stored in memory. When used in conjunction with the app, this also allows for easy updating of the patient information.

2.5.2 Pressure/Weight Sensor

We used a 100 g Load Cell connected to an amplifier breakout board to detect if there are pills in the cup that are ready for collection. This load cell was specifically chosen because it should be able to detect changes that are around the granularity that we need which was roughly 1 g difference. Accounting for a mounting plate made of acrylic, 100 g is just the right upper bound to measure the medication and pills dispensed.

Initially the load cell was connected directly to the microcontroller. This was not a viable solution because the sensitivity of both the load cell combined with the sensitivity of the microcontroller was not high enough to detect any change in readings. For this reason, we went with a breakout amplifier circuit that amplified the signal as well as discretized it for simple integration with the microcontroller.

As a unit, these two pieces worked perfectly with very high accuracy.

2.6 Mechanical/Dispensing

2.6.1 Physical Design/Reservoir System

The mechanical system consisted of five replications of the reservoir system. Each reservoir system included a hopper to store pills, a servo, a customized disk to adapt to the servo wheel and transport pills, and a set of IR sensors. In the sections below, we will discuss each of the electrical components separately. It is important to note that, for the purposes and scope of this project, M&Ms were used to simulate pills. They have (mostly) uniform size and weight, while varying in color.

The servo rotated between 0° (LOAD State), 90° (CHECK State) and 180° (DROP State). It began at Load, rotated to Check, where IR sensors (see 2.6.2) were activated. If the sensors detected an M&M, the servo would continue to rotate to the drop state and dispense a single M&M at a time. If an M&M was not detected at Check, the servo would reverse and go back to Load (and cycle back as many times as necessary, until a single M&M was dropped). This process was activated by the microcontroller (see 2.6.3) until the necessary pills were dispensed.

Figure 3 shows the overall view of the reservoir system.

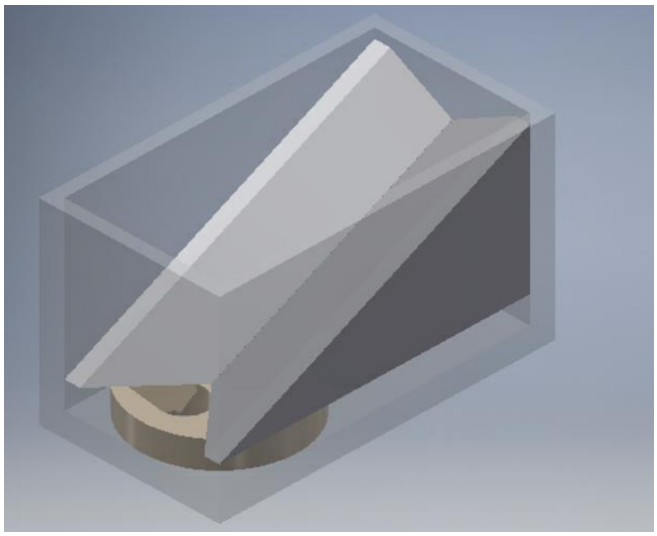

Figure 3 - Isometric view of reservoir

Below is the empty hopper. The two bigger holes are for the Check and Drop states - the slot in the disk aligns with both of these for the respective states, and one has the emitter within it as a part of the IR sensor subsystem.

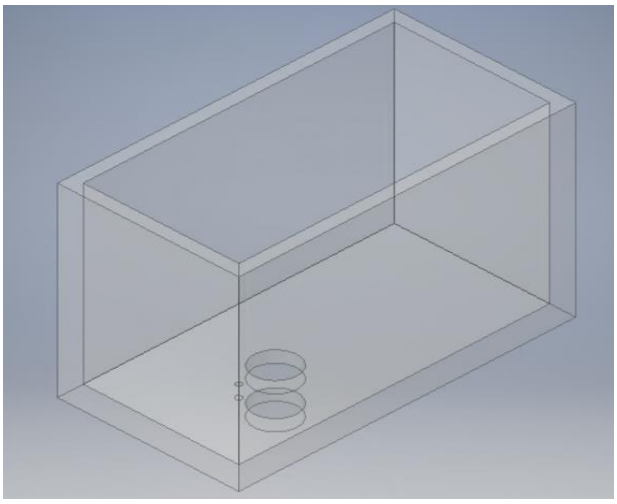

Figure 4 - Isometric view of hopper

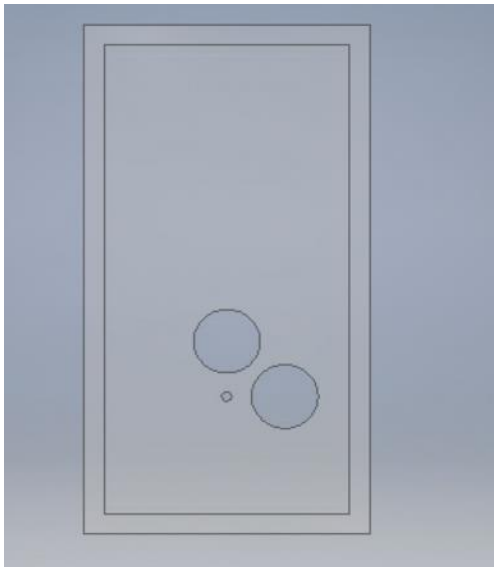

Figure 5 - Top view of hopper

Below, figure 6 shows the ramp within the Reservoir system. The purpose of the ramp is to align the M&Ms, so they can filter into the Load slot properly and avoid getting jammed.

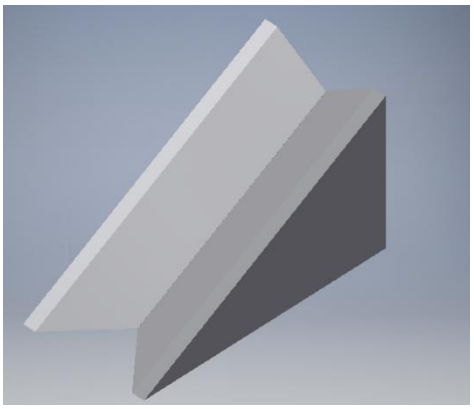

Figure 6 - Ramp insert

2.6.2 Servo

Since the servo came with very specific “wheels”, it was imperative that we create a special disk to adapt to our needs; in this case, the disk needed to have a slot to hold one pill at a time and be able to rotate within the hopper without getting jammed. Figure 7 and figure 8 show the first disk design; it contains a single slot with the dimensions of a single M&M; the circumference is 0.4094” (just over that of an M&M), and the depth of the center circle is that of a single M&M, 0.02756”. The ramp adjacent to it was created to allow for any additional M&Ms in the reservoir to gradually be pushed out of the slot as the disk rotated. This physical design ensured that only one M&M could sit in the disk at a time, and therefore, only one could be dispensed at a time. This was a specific requirement (discussed in further detail in Section 3), and ensured the quantity dispensed was correct (no room for error).

It was designed before obtaining the premade wheels that came with the servos, so it doesn’t have the ability to attach to the wheel or C1 spline on the servo itself.

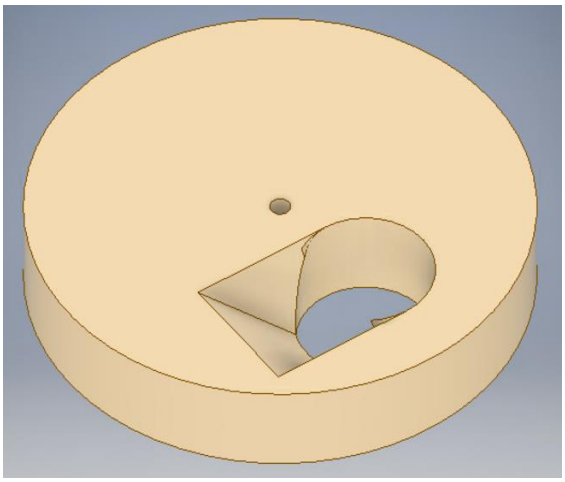

Figure 7 - Top isometric view of disk

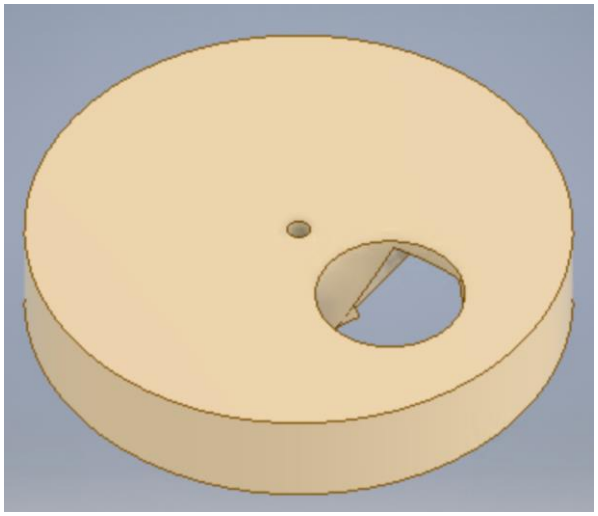

Figure 8 - Bottom isometric view of disk

Below, figures 9 and 10 show the top and bottom views of the adapted disk design, to ensure compatibility with the servo wheel. The shape of both, the ramp and slot for the M&Ms remained unchanged. The only difference is the cutout for the servo wheel to be glued in.

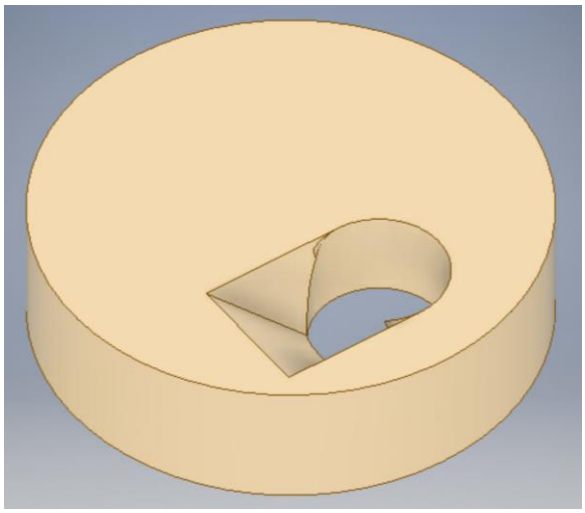

Figure 9 - Top isometric view of new disk

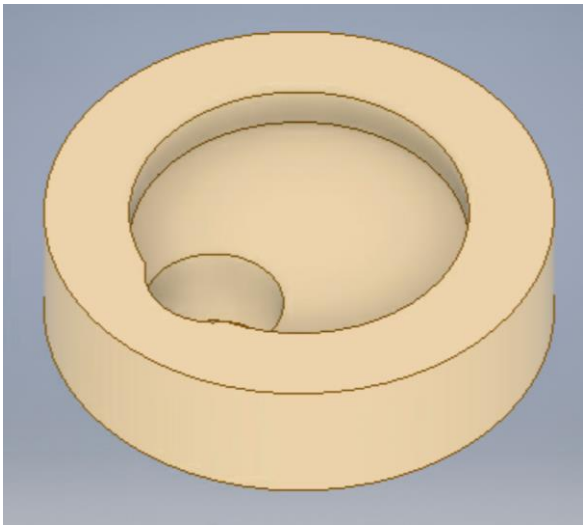

Figure 10 - Bottom isometric view of new disk

2.6.3 IR Sensor

In terms of the three states (Load, Check and Drop), Check is the only state applicable to the IR sensors. The sensors came in pairs, consisting of one emitter (diode) and one detector (NPN transistor). Both needed to be connected in parallel. The general circuit layout is shown below in figure 11.

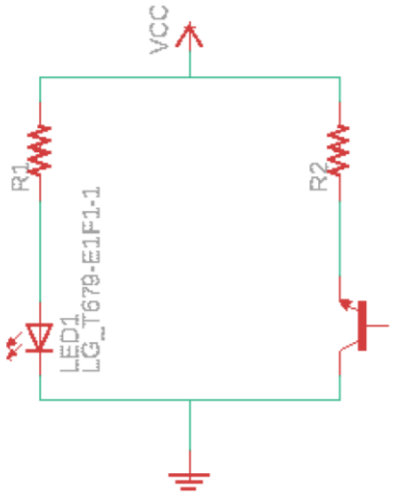

Figure 11 - IR sensor schematic

The next step was to calculate and assign values to the components. The voltage being given to each servo system is 6 V, so VCC = 6 V. The value of R1 is determined below. The data sheet for the emitter specifies the maximum current allowed, which is 50 mA.

$$R1 = 6\,V \div 0.050\,A = \ 120\,\Omega \quad (1)$$

The calculation for R2 is very similar - the difference is the maximum current allowed. In this case, the test current is 1 mA.

$$R2 = 6\,V \div 0.001\,A = 6000\,\Omega \quad (2)$$

For both R1 and R2, these are the minimum resistance values needed to ensure the emitter and diode aren't damaged.

After the circuit was set up and functioning with the Arduino, values needed to be measured to understand what the tolerance would be for determining PILL vs. NO PILL. Table 1 shows the values measured for the respective obstacles and M&M colors.

Table 1 – IR Sensor Readings

M&M Color/Obstacle	Analog Value
No Obstacle/No Pill	34 - 43
Green	646 - 658
Red	650 - 658
Orange	649 - 659
Brown	646 - 658
Yellow	640 - 649
Blue	646 - 652

To ensure ambient light did not have an effect on the values, these values were recorded in different types of lighting (darkness, regular housing lighting, and with a flashlight over the sensors), as well as varying the distance from the obstacle to the emitter and detector.

Though these values were all above 600, the tolerance value used in the Arduino code is 200; in other words, if, when the IR sensor is activated, the value obtained is < 200, the system returns "NO PILL". If the value obtained is > 200, the system returns "PILL".

2.6.4 Microcontroller

We used an ATMEGA328 microcontroller for the same reasons stated above. The controller in the dispensing unit has two main requirements. Being able to interpret instructions from the main controller in the processing unit and being able to dispense a single pill from a specified servo. The controller

accomplishes this through an array containing many function pointers. Each function pointer corresponds to a function that dispenses a single pill according to the process mentioned above from a specific servo. The signal received from the main controller is used an index into this array. After the pill is dispensed, an ACK signal is sent back to the main controller.

2.7 Power

2.7.1 Power Supply

The Power Supply module was implemented with a simple wall adapter. It plugs into a barrel socket to be used with the device. Since the majority of parts needed +5 V source, it was trivial to select a Power Supply that converted wall 120 V AC into 5 V DC. Other power divisions were handled with voltage regulators and dividers on a per component basis in those modules.

3 Design Verification

3.1 App

The caregiver app does not have any quantifiable verifications other than communication with the Bluetooth module on the device. If our app is successful then it will be able to communicate over Bluetooth, or another serial device, to our Med-I-Can dispenser. We can see that this is the case with the design of the app and the array that is printed out when the sync button is pressed. Because of this output, we know that the sync data can be serialized and sent to the Bluetooth controller.

3.2 Bluetooth

Unfortunately, we were not able to fully implement the module due to technical difficulties. We wrote the code on the app and main controller that allowed exactly one patient's worth of data to be written to and read from serial output and input. Although we were able to turn on the chip, we were not able to enter the configuration menu and send meaningful data between the app and the device. We are reasonably sure that the cause of the issue lies not with the written code but with the hardware, whether this be our PCB design that acted as a breakout for the chip or with the chip itself. Assuming that in the future we have a working server and Wi-Fi module, we should be able to easily adapt our code to work with the new implementation.

3.3 Processing

3.3.1 Microcontroller

The main controller was required to be able to communicate with the controller in the dispensing unit. This was verified by having the dispensing controller provide outputs to 5 different LEDs. We were then able to toggle a particular LED by sending an integer between one and five from the main controller to the dispensing controller through UART. Similarly, we were able to verify that the main controller could receive ACK signals by having the controller hang until the dispensing controller sent back an ACK signal.

Another requirement was that the controller must be able to successfully read inputs from the pressure sensor. We were able to confirm this by having the fingerprint reader not accept another fingerprint until the medication cup has been removed and then replaced back into the device empty.

3.4 Sensors

3.4.1 Fingerprint Scanner

The fingerprint scanner worked exactly as intended. Although there were sometimes relatively rare issues with getting the scanner to recognize a fingerprint depending on where the finger was placed on the scanner, the scanner never misidentified one patient for another. This was confirmed by printing the returned user ID to the serial monitor and continuous testing with different registered fingers.

3.4.2 Pressure/Weight Sensor

The sensor was tested using the microcontroller to determine its sensitivity to small changes. This was done by securing the load cell onto a block, in our case the base of the device, and fixing a mounting plate to the weight-bearing end of the load cell. The load cell and amplifier circuit was then connected to our microcontroller that had been loaded with a logging function. Starting with the plate and cup on it, we started taking readings. This effectively zeroed the scale with the cup on it. This was helpful in determining the presence of the cup on the device which was successful in 100 % of cases using many different disposable cups.

Following that verification, we tested to ensure that each pill could be read individually correctly. Weighing out several different M&Ms we found that each one weighed between 3.5 and 4.0 units. This is important to note since there is a wide range of weights. Going with the larger of the two measurements as our standard for pill weight ensured that we never mis-read a smaller number of pills for a larger number, i.e. 3 pills didn't get read as 4 pills. This estimation held true for about 10 pills at a time, which was our requirement, but inevitably was incorrect for greater than 12 pills, sometimes reading 11 pills as only 10. Figure 12 shows our measured values vs. the values implemented in the control logic and exemplifies how using the higher weight resulted in the correct number of pills being registered.

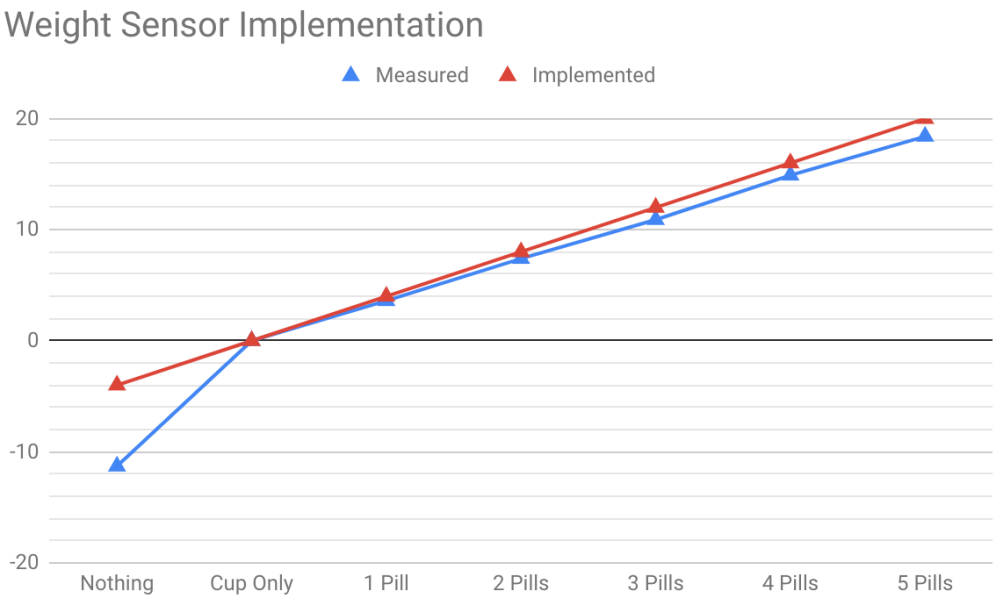

Figure 12 - Weight Sensor Verification

3.5 Mechanical/Dispensing

The design verification of the Mechanical/dispensing system was apparent in its ability to effectively load, check and drop multiple M&Ms. In the following subsections, the successes will be discussed in greater detail.

3.5.1 Servo

Once the servos were connected, testing them to ensure proper functionality was crucial. The first test was ensuring that they aligned properly. This was done with the slotted disk attached to the servo and checked visually. As the program was run, the servo and disk rotated and stopped at 0°, 90°, and 180° with precision and accuracy, thus completing the verification requirement.

3.5.2 IR Sensor

In order to test the IR sensor accuracy, a program was run in Arduino in which the sensor differentiated between NO PILL (value is < 200) and PILL (value is > 200). Then, our team performed a blind test - one team member viewed the Arduino Monitor, while the other would place an M&M between the emitter and detector, and acknowledged the pill's presence with 100 % accuracy.

The success was also noticeable in the testing of the full assembly - when an M&M dropped into the slotted disk, and moved to the Check state, the servo would continue to the Drop state. When the disk was empty, it would continuously rotate between Load and Check.

3.5.3 Microcontroller

We were able to verify that the dispensing controller was also able to communicate with the main controller though UART through the test we performed above for the main controller.

Like the IR sensor test, we were able to verify that the controller could correctly interpret the outputs of the sensors as the servo continuously alternated between the Load and Check state until an object was placed between the IR sensors. After which, the servo rotated to the Drop state. We were also able to verify the requirement that the controller be able to rotate a specific servo through practice. As mentioned earlier, this was implemented through an array of function pointers where the index into the array would specify a different servo.

3.6 Power

3.6.1 Power Supply

The supply voltage of the device was tested to ensure it was within an acceptable range for all of the devices that will be using it. In order to do this, we set up a simple test circuit that would put a light load on the power supply and allow us to measure the source voltage by measuring the component voltages within the circuit.

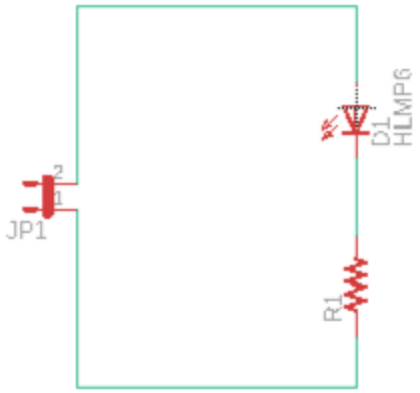

Figure 13 - The simple test circuit for Power Supply Module

Figure 13 shows the simple circuit that was used to validate the Power Supply Module. The jumper was supplied with the +5 V from the Power Supply and then voltage readings were taken over the LED and Resistor. The following table represents the readings taken in the circuit.

Table 2 – Test circuit voltages

Component	Voltage
LED	1.906 V
Resistor	3.318 V

Using the Kirchhoff Voltage Law, we can then determine the voltage over the power supply,

$$V_{Source} = V_{LED} + V_{Resistor} \quad (3)$$

which comes out to 5.224 V. This value is within 5 % of the required source voltage of 5 V, thus satisfying our requirements for our source voltage.

4 Costs

4.1 Parts

Table 3 – Parts Costs

Part	Manufacturer	Quantity	Retail Cost ($)	Bulk Cost ($)	Actual Total Cost ($)	Bulk Total Cost ($)
ATMEGA 328P-PU	Microchip Technology	2	$2.14	$1.78	$4.28	$3.56
HS-422 Servo	Hitec	5	$9.95	$9.95	$49.75	$49.75
TAL221 Load Cell	HT Sensor	1	$9.95	$8.96	$9.95	$8.96
HX711 Load Cell Amplifier	Sparkfun	1	$9.95	$8.46	$9.95	$8.46
IR Sensors	Sparkfun	5	$1.95	$1.76	$9.75	$8.80
Fingerprint Scanner	Sparkfun	1	$35.95	$32.36	$35.95	$32.36
Qwiic Cable	Sparkfun	1	$1.50	$1.35	$1.50	$1.35
Wall Adaptor	Sparkfun	1	$5.95	$5.36	$5.95	$5.36
DC Barrel Power Jack	Sparkfun	1	$1.25	$1.13	$1.25	$1.13
Real Time Clock	Sparkfun	1	$15.95	$15.95	$15.95	$15.95
RN-42 Bluetooth	Roving	1	$18.95	$18.95	$18.95	$18.95

Total					**$164.23**	**$154.63**

4.2 Labor

Assuming a wage of $35 per hour, and assuming each group member puts in 12 hours a week, for 10 weeks. The final total labor costs approximately:

$$3 \text{ people} \times \$35 \text{ hours} \times 12 \text{ hours per week} \times 10 \text{ weeks} \times 2.5 = \$31{,}500 \quad (4)$$

5 Conclusion

5.1 Accomplishments

Though this project did not culminate in a working prototype, there were several successful modules within the project. All submodules worked individually except for the Bluetooth module, but had some failures in connecting them.

The biggest accomplishment of the system was the mechanical implementation of the dispensing subsystem. Through some design and testing, we were able to determine the perfect size and shape for each dispensing hopper, as well as the disk. These parts were crucial to the functioning of our design and worked flawlessly.

Along with the dispensing subsystem, the control subsystem managed everything flawlessly. Through that subsystem, we were able to set up a state machine to assign a user's ID to their schedule and properly spin the correct dispensing reservoir.

In order to measure those dispensed medications, we properly implemented a load cell to measure the change in weight of the cup used to collect the medicine. The load cell can measure up to 10 pills correctly at the same time.

Finally, to control scheduling and program the device, we implemented an app for caregivers to use. We have been successful in creating a simplistic design that allows for the caregiver to see all information pertaining to a patient on a single page.

5.2 Ethical Considerations

In dealing with devices that cross into the healthcare domain, there are many components to be considered including information privacy, malicious use of the system, and medical regulations.

Since we deal with sensitive information, such as an individual's medications and medication schedule, we had to worry about potentially leaking this information, intentionally or unintentionally. This makes us susceptible to possible breaches of Association for Computing Machinery (ACM) codes 1.6 and 1.7 (1), which require respecting privacy and honoring confidentiality. We needed to make sure that we are securing the user's data correctly and making sure that we do not access information needlessly.

We solved these concerns by encrypting information within the app subsystem and only stored critical information on the actual device. Information on the device is restricted to the timings and the dispensers. The app itself has information such as the patient's name, which medications are set to dispense, and when they are set.

When working with medicine and peoples' medicine schedule, we must consider the users' health. The Institute of Electronics and Electrical Engineers (IEEE) Code of Ethics also takes this into account with rule 9 (2). In terms of our project, we must ensure that pills are dispensed correctly every single time without any missed doses or double doses. This is guaranteed through the implementation of the IR

Sensors. Additionally, the dimensions of the slot in the disk holds exactly 1 pill. With both the mechanical and sensor design, we ensure that the correct number of pills is dispensed every time.

Due to the many regulations related to group homes and nursing homes, this device will most likely be targeted towards individuals, their families, and caregivers. Because families and individuals aren't governed by any one group, there isn't a specific list of qualifications to pass for medicine dispensing devices. Primarily, we will have to show these groups that the device is consistent, dependable, and easy to use. This will be taken into account during the entire design process and tested through interviews with potential users of our device.

There are several other safety concerns that we will have to address. The first would be that of a malicious user attempting to access the individual's medication. While this would already be a problem for someone using the device, we can make it even more secure by locking up the medication and only dispensing it when the user presents their unique identification card. In addition, we could also prevent a case in which the user forgets to take their medication on time. To combat that, we are going to have alerts sent to the caregiver when the individual has missed a dosage at a certain time. This can help the caregiver keep track of how often the user is missing doses and if more intervention might be necessary.

Since this device is connected to the wall outlet, there is some danger of electrocution when developing or using the device. In order to mitigate this chance, we will insulate all wires and cords both within the device and going to the wall outlet as well as using the three-pronged, grounded, outlet plug. In addition, any developers working on this device should unplug the device before working with it.

5.3 Future Work

Future work on Med-I-Can revolves around the caregiver experience with the device. We would like to see caregiver-device interactions be moved from Bluetooth to Wi-Fi, add more memory to the microcontroller, have a more scalable physical design, and adjust the hopper disks to accept different types/styles of pills.

The first modification would be to remove Bluetooth communication in favor of Wi-Fi communication. This would open the possibility for a back-end server to be able to take information from the device on a real-time basis and could send push notifications to the caregiver's app, providing a more updated view of patients' medication.

Secondly, the current state of the microcontroller is that it uses on-board memory to store the patient and hopper data. This can get tricky when storing multiple users' data. In order to make the device more scalable, off chip storage should be added to the device to store extra users' information.

Along with adding memory to make the hardware more scalable, creating a more scalable physical design would also help expand the device to be used with more patients.

Finally, our current design works exclusively with one type of pill. To work with a larger array of medications, Med-I-Can needs to have an interchangeable disc and hopper designs that can be substituted out for different specific medications.

References

(1) ACM Code 2018 Task Force. “ACM Code of Ethics and Professional Conduct.” Association for Computing Machinery, www.acm.org/code-of-ethics.

(2) “IEEE Code of Ethics.” IEEE - Advancing Technology for Humanity, www.ieee.org/about/corporate/governance/p7-8.html.

(3) Hallock, Betty. “M&M's Mega -- with Three Times as Much Chocolate -- Hits Stores.” Los Angeles Times, Los Angeles Times, 25 Apr. 2014, www.latimes.com/food/dailydish/la-dd-mms-mega-three-times-chocolate-hits-stores-20140411-story.html.

(4) Science Buddies Staff. "M&M Geometry." Science Buddies, 28 July 2017, https://www.sciencebuddies.org/science-fair-projects/project-ideas/Math_p022/pure-mathematics/mms-geometry. Accessed 4 Oct. 2018.

(5) Hallock, Betty. “M&M's Mega -- with Three Times as Much Chocolate -- Hits Stores.” Los Angeles Times, Los Angeles Times, 25 Apr. 2014, www.latimes.com/food/dailydish/la-dd-mms-mega-three-times-chocolate-hits-stores-20140411-story.html.

(6) “SCHEMATICS – INFRARED EMITTER DETECTOR”. Society of Robots, https://www.societyofrobots.com/schematics_infraredemitdet.shtml
(4) “DC Analysis of Circuits with BJTs”. UIUC ECE342 Staff Coursenotes, https://courses.engr.illinois.edu/ece342/fa2018/secure/handout/HO18.pdf

(7) “Controlling a Servo with an Arduino”. Servo City, https://www.youtube.com/embed/T8nCN10MjNE?rel=0&autoplay=1

(8) “Servo – Hitec HS-422”. Sparkfun, https://www.sparkfun.com/products/11884

(9) “Infrared Emitters and Detectors”. Sparkfun, https://www.sparkfun.com/products/241

(10) “LTE-302” Lite-On Electronics, https://www.sparkfun.com/datasheets/Components/LTE-302.pdf

(11) “LTE-301” Lite-On Electronics, https://www.sparkfun.com/datasheets/Components/LTE-301.pdf

Appendix A: Requirement and Verification Table

Table 4 – System Requirements and Verifications

Component	Requirement	Verification	Verification status (Y or N)
AC/DC Converter	The converter must be able to convert 110 V from the wall outlet to a voltage of 5 ± 0.5 V	Plug the converter into a wall outlet, then measure the voltage between the V+ and ground outputs with a multimeter to ensure that the output voltage is within the acceptable range.	**Y**
Microcontroller (Main)	1. The controller must be able to communicate through UART with 2 devices 2. The controller must be able to read the outputs from the pressure sensor. 3. The controller must be able to read an acknowledgement from the dispensing unit.	1. Another microcontroller will be set to toggle an LED when it receives a UART signal from our main controller. It should successfully toggle 9/10 times. This should be tested with 2 different UART port pairs. 2. A blind test will be run 10 times. An observer should be able to differentiate between cup only, cup + meds, and no cup 9/10 times. 3. A blind test will be run 10 times. An observer should be able to differentiate between an acknowledgement and no acknowledgement 10/10 times.	**Y**
Bluetooth	The microcontroller must be able to communicate with a computer that is running our app through Bluetooth.	The controller will be set to toggle an LED when it receives a Bluetooth command from the app. It should successfully toggle 9/10 times.	**N**

Pressure Sensor	The pressure sensor must be sensitive enough to a degree under the weight of one pill (estimated to be 1 gram)	A random number of pills between 0 and 10 are to be placed in a cup on top of the sensor and the number of pills in the cup must be able to be correctly determined from the output of the sensor 10 out of 10 times	**Y**
Thumbprint Reader	The reader cannot make any mistakes when identifying different users	Two different fingers will be scanned on rotation, the scanner must be able to correctly identify the patient 10 out of 10 times	**Y**
Servos	1. Each servo must be able to rotate between 0-155° without getting jammed 2. Servos must be able to accurately line up with three positions: Loading, Check, Drop/Dispense.	1. Servo will be at “neutral”/loading, and an M&M will be placed into the wheel slot. Servo should turn clockwise and counterclockwise without getting jammed. Should correctly do this in 10 out of 10 trials. 2. When servo stops at each destination, it should align such that the M&M is fully visible from below Check area and drops without jamming in Drop area. Should correctly do this in 10 out of 10 trials.	**Y**
IR Sensor	Sensor should detect/differentiate between pill (any M&M, regardless of color) and no pill.	A blind test will be done, with one group member watching sensor output, and the other either loading pill into slot, or leaving it empty. When an M&M is placed in wheel slot in loading and twisted to Check state sensor will recognize pill. When wheel slot is empty and twisted to Check state, sensor will return “no pill” output. In 15 trials, Sensor will accurately determine pill vs. no pill with 100 % accuracy.	**Y**
Microcontroller	1. The microcontroller must be able to communicate with	1. Another microcontroller will be set to toggle an LED when it receives a UART signal from our main controller. It should	**Y**

(Dispensing)	another microcontroller through UART 2. The microcontroller must be able to read the output from the IR sensor. 3. The microcontroller must be able to control the appropriate servo when needed	successfully toggle 9/10 times. This should be tested with 2 different UART port pairs. 2. A blind test will be run 10 times. An observer should be able to differentiate when a M&M is within the slot on the disk 10 out of 10 times 3. The controller must cause the servo with the correct address to spin 10 out of 10 times	
App	1. The app must allow for the caretaker to make changes to doses and medications 2. Check some of the history for previously dispensed medication 3. See if a certain patient has gotten their medication yet	1. A new dosage will be scheduled on the app and then sent to the device. The device should dispense this new dosage correctly 10/10 times. 2. A dosage will be dispensed and then the “history” will be brought up. The history should reflect the dispensed dosage 3. A dose will be scheduled to dispense and the “current activity” page will be brought up. The dosage should show as “scheduled” until dispensed where it will change to “dispensed.”	**Y**

Appendix B: Processing FSM

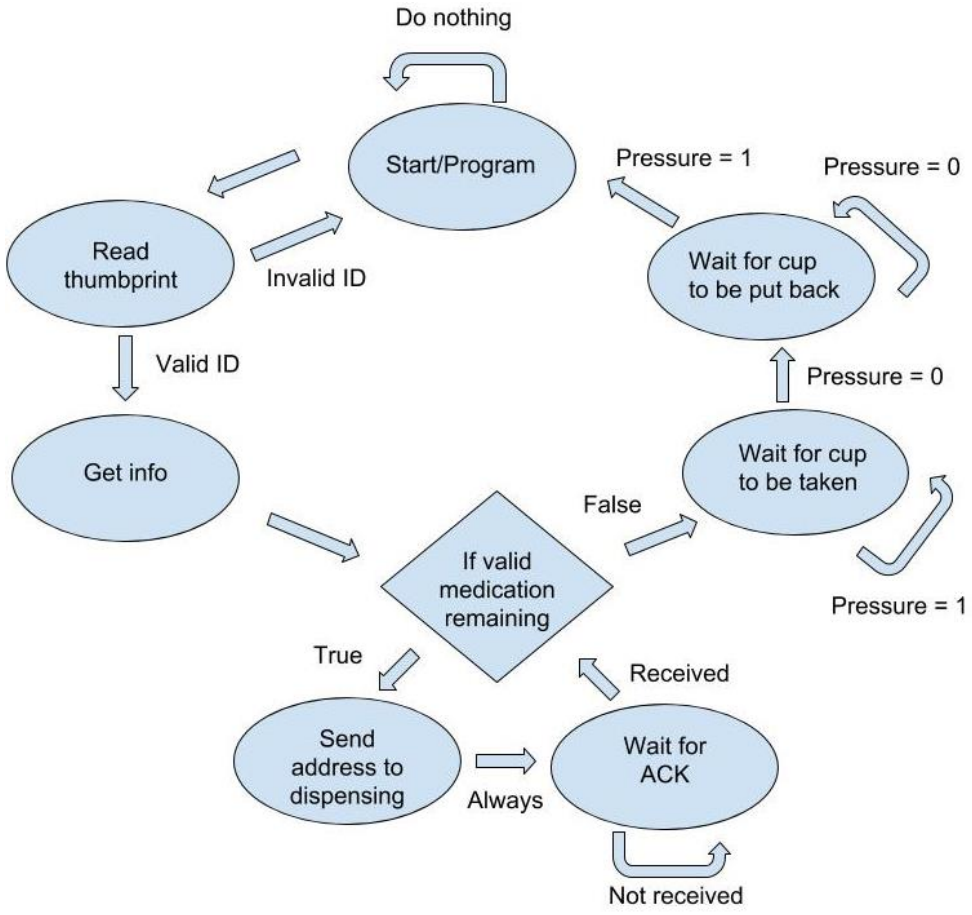

Figure 14 - Main Control FSM

Appendix C: Dispensing FSM

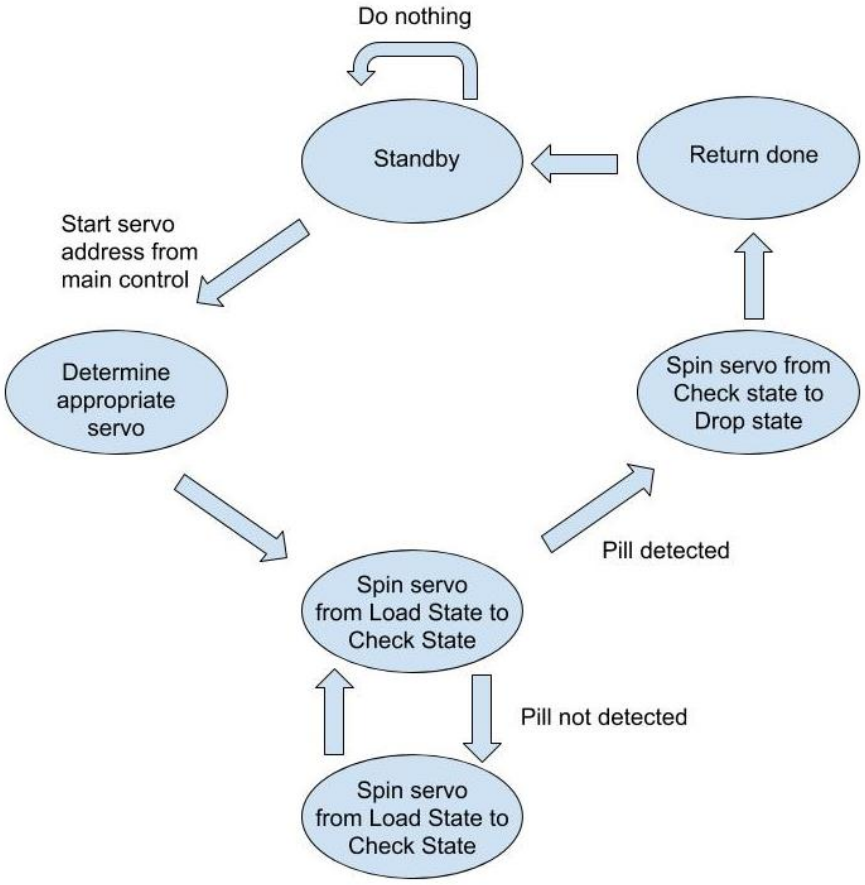

Figure 15 - Dispensing FSM