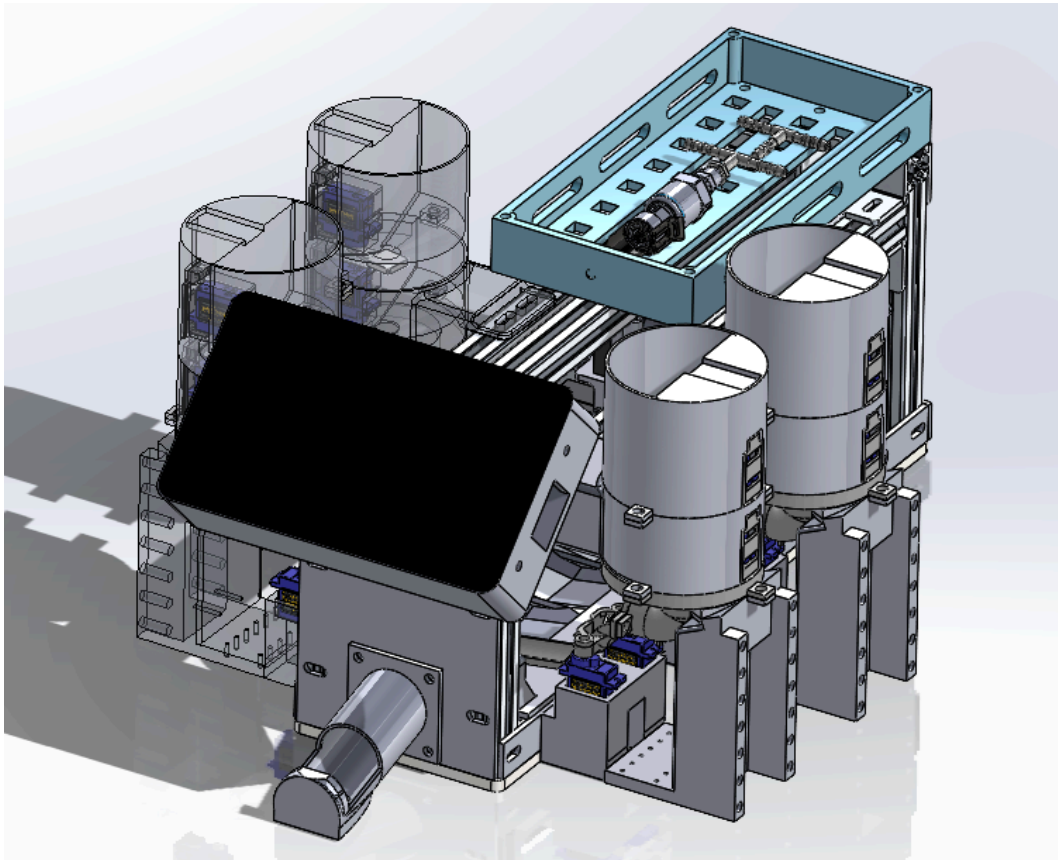


Automated Medication Manager

MAE162E Project Report

Group #1

ManyMeds™

Aaminah Malik, Alice Taylor, Bradley Jackson, Onur Calisir, Raya Shafeeq, Shubh Raval

June 14, 2024

Abstract

This report details the progress that was made to make a complete automated medicine dispenser project. The inspiration for the mechanism came from the personal interests of some of our team members relating to medicine adherence. The initial stages of the project included looking into existing designs of automatic medicine dispensers and gauging what factors from each were worth carrying over into our novel design. We quickly learnt that most if not all of the existing designs had shortcomings related to cost and complexity. These two elements became some of the first parameters that we set as design requirements including making a product that is affordable and easy-to-use. After the initial brainstorming and design phases we tried and passed a few dispensing mechanisms such as an agitation plate mechanism and PEZ mechanism. Each of these used a form of actuation that wasn't reliable enough for the action we were hoping to obtain, including tackling the two biggest design concerns: (1) dispensing differently sized pills and (2) dispensing one pill at a time. The key when developing the final design was to simplify the initial design as much as possible in which the agitation plate, slide-and-gate, and PEZ mechanism were all used in conjunction. The general idea was to add as much redundancy as possible between the point at which the pill is added by the user and the point at which it is dispensed into the pill receptacle because this ensures to a sufficient degree that the mechanism can manage the drop of a single pill. Further testing and development, however, made us realize that rather than over-complicating the system with all these redundancies the design could be simplified and still give us the desired result. With the proper programming that allowed the simplified mechanism to be responsive to a user input, we were able to arrive at the current, final design. As for the user interface and software components of this project, programming the Raspberry Pi to interface with our Arduino Mega in the intended manner proved difficult at times but once the Raspberry Pi was established as the input every component operated as the code instructed it to. All in all, we are pleased with the outcome of our two quarters of design development and testing. This report first details the preliminary design and results from Winter Quarter and then discusses the final design and results from Spring Quarter.

Table of Contents

List of Figures

List of Tables

1 Project Scope

1.1 Problem Statement

Taking medications in a timely manner and with the correct dosages, as important as it is, isn't much of a burden to the general public who are able to keep track of all their medicines and, more importantly, are physically able to organize them and take them at their prescribed times during the day. However, the same cannot be said about the over 60 million Americans who are over the age of 65. Nearly 9 out of 10 of these individuals require daily prescription medications for a host of health concerns including heart disease, diabetes, and metabolic syndrome. Furthermore, survey results taken in the year 2020 indicate that over 50 percent of the elderly population in the United States finds it very challenging to keep up with their prescriptions. Unfortunately, older patients who tend to require several medications a day are also the individuals who exhibit lower rates of medicine adherence. More specifically, for individuals requiring 4 medications a day, the average medicine adherence rate is only 50%. This number further decreases as age of the patient and number of medications increases.

1.2 Client Identification and Recognition of Need

The difficulty that individuals over the age of 65 face when trying to stick to their medication plans is a problem that will always exist and may even be exacerbated as time goes on with the onset of aggressive cancers and novel virus strains. Apart from these external factors, elderly patients are inevitably at significantly higher risk for nonadherence compared to younger subjects due to functional decline which presents degraded cognitive skills, visual impairment, and lack of manual dexterity. Not only is there the chance of missing medications, but more pressing is the possibility of taking the *incorrect* medication or dosage. There is a growing need given these possibilities that a system be designed to facilitate the process by which elderly patients take their medications. It turns out that there are already several products in the public health sector that have been designed specifically to address this need; however, the price of current automated medication dispensing products overshadows the benefits as many patients are not financially able to acquire the expensive services provided by brands such as MedMinder Siemens, Hero, and MedaCube. Many aren't even aware that this solution exists and others may be hesitant to purchase due to the complex setup process and unfriendly user interfaces. So, the

challenge remains to design an economical and user-friendly automated medication dispenser for individuals over the age of 65.

Given this general overview of the problem and need statement, both the high-level and low-level design requirements were considered in order to create the foundation for our pill dispensing mechanism. Based on our understanding of the market need, we wish to create a polished and compact dispensing device that is lightweight, a fraction of the cost of its competitors (~$600), and most importantly includes a user interface that is friendly for elderly patients who are visually and/or physically impaired. Elaborating on this final requirement, the device should include an interactive feature in the form of a screen that requires minimal user inputs and feedback. This will allow patients to have a more legible and enlarged feed of information regarding their medications which is easier to read and comprehend than the small text found on pill containers. The device should also inherently eliminate the need for the individual to sort the medications by day by having a compartment for the patient to simply pour all the pills from their container into. The inner workings of the machine will make sure to properly dispense the correct type and amount of pill.

As for the low-level design requirements, the main objective was to create the central mechanism that is responsible for dispensing the pills one at a time (for the shapes and sizes chosen by the design team). The mechanism is assembled out of 3-D printed renderings of several components that are discussed and shown in the later sections of this report. The design is open to using multiple iterations of the central mechanism (several hoppers) for different pill sizes or using one central mechanism (master hopper design) for different pill sizes. Each mechanism is fitted with DC motors, servo motors, gears, and vibration plates to actuate its different components.

1.3 Project Goals and Objectives

The main goal with this project is to create a mechanism that is able to accurately dispense medication when prompted to do so by the user (whether that is the elderly patient or their caretaker) and requires reloading once every 90 to 120 days. Specific objectives include that the design must be able to accommodate and store at least a 120 day supply of pills of varying sizes. More importantly, the device must be able to dispense one pill at a time from the storage container of pills. This includes accurately dispensing the pills when prompted to or on schedule. Another objective is to include a sensor that is able to detect whether or not the correct pill(s) was

dispensed and possibly have an incorporated troubleshooting or calibration step. Finally, it is very important that the device is fit with an easy-to-use User Interface that contains simple commands that require basic responses either from the patient themselves or from a caretaker.

These goals seem adequate and achievable given the remaining progress that needs to be made. However, the following are some reach goals that we would like to try and accomplish if time and budget permit. This includes a robust calibration process that uses weight sensors to verify the type and number of pills in the pill receptacle after dispensing is complete. Additionally, there is room to improve or add-on to the software/UI features by incorporating a touch screen, mobile app connectivity, and a barcode scanner for rapid pill identification. These again, are possible stretch goals that can be added on to the base mechanism in future versions of this project.

2 Project Planning and Task Definition

2.1 Task Identification

For the initial dispensing mechanism design, we split the team into 3 different design groups. The first group focused on an agitation plate design, based on an existing pill-counting device in which pills sat on a plate and a central rotating agitator pushed them through an off-ramp in a single file line. The second design was a slide and gate concept where a sloped gate kept the pills in a single file line and then a series of gates stopped the flow of pills and allowed for a single one to be let through at a time. The third group worked on a sort of PEZ dispenser design where hoppers that hold pills sit in a circle and a spring-controlled latch allows for a single pill to be released.

Each group worked on printing various iterations of the design. Finally, we created a design that incorporated pieces of each group's subsystem to make a more robust final design that filters pills through multiple steps. By adding these redundant filters, we can decrease the likelihood that extra pills are dispensed. With this final prototype created by the whole team, half of the team worked to set up the Arduino code and wiring to control the servo and DC motors to validate the mechanism.

In the Spring quarter, we started out by modifying our design to fix the problems with the hopper/filtration system that we ended the Winter quarter with, and worked to develop a frame and delivery system so that the pill dispense system could handle multiple medications. After the design phase, we moved onto manufacturing. This included a lot of 3D printing (and iterations)

for the hoppers, and more machining for the frame, since it was made out of 80/20, aluminum, and acrylic sheets. Once the individual components were manufactured, we tested them for tolerances and integration with electrical components. Verifying and developing wiring and code to control the stepper motor, linear actuator, pressure transducer, vacuum pumps, vibrational motors, servos, and motor controllers took a lot of time, and since cheap components were purchased, they often had to be repaired or replaced. After testing individual components, we started developing a plan for assembling everything. This required careful electronic planning, since enough voltage and amperage had to be supplied to power the numerous components that made up the system, so we started with a wiring diagram to plan out all the necessary components and find a way to power the system with a wall plug. Once electronics were attached to each sub assembly (hopper and frame), code was developed and tested. Then, the final assembly was put together and tested as a full system, although we found issues with certain electronics and wire management.

2.2 Timeline

2.2.1 Winter Quarter (past progress)

Our first week as a group was week 3 of the winter quarter, which was spent generating as many project ideas as possible. The pill dispenser was an idea that two different members of our team had separately based on their grandparent's experience with medication, so by week 4 we had decided to pursue that. We spent the rest of the week looking for existing smart pill dispensers and discussed possible mechanisms that may need to be involved. In week 5, we formally chose our project and worked on coming up with a timeline and tasks associated with the project to prepare for our idea pitch meeting at the start of week 6. Week 6 is when we started design discussion and modeling for various dispensing mechanisms while also trying the decide on a range of pill sizes to accommodate. We also did some background research on various components we were considering such as scales, motors, and vacuum suction. In week 7 we had another meeting to present our initial CADs and we were advised to simplify as much as possible and use gravity for our pill sorting, so we switched up our initial design and started working on the 3 components of our final design. In week 8 we presented the 3 separate designs each group had worked on and made some final revisions before printing our first round of prototypes. In week 9 we had printed prototypes and started testing them and refining designs before meeting with Professor Hong. The professor encouraged us to choose a main design or pieces of various designs and start working on one integrated concept. After that meeting, we spent the rest of the quarter designing our final hopper which contained aspects of each design.

Task:	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10
Ideas										
Research										
Initial CAD										
3 Design CAD										
Prototype and test										

Table 1. Winter Quarter Gantt Chart

2.2.2 Spring Quarter (past progress)

The beginning of Spring Quarter was spent developing a finalized assembly for the complete system. The final design incorporated many levels of complexity beyond the hopper that we ended Winter Quarter with. We decided that the hoppers that contain each type of pill would be mounted onto a frame with a delivery plate and a vacuum system to deliver the desired pills to a final outlet spout. After finalizing the various components of the design (around week 3 although modifications were made for weeks following), we started to order the necessary materials to build the prototype. Materials included 80/20, aluminum sheets, acrylic sheets, fasteners, and various electrical components. We tried to make sure all orders were placed early enough, by week 5, but when components broke some later orders had to be placed. Once orders began to arrive, manufacturing began around week 4. The 80/20 was cut into the appropriate sizes for the frame using a mill, the aluminum was laser cut and bent to be brackets, and an acrylic base plate was laser cut. At the same time, 3D printed components were printed and reprinted to achieve desired quality and tolerances. Along the way, prototypes of the hoppers were tested and assembled. By week 6, the frame was assembled and it was time to move onto wiring up electrical components. Throughout manufacturing time as electronic components arrived, they were tested and modified to fit the needs of the project. Wiring the components onto the frame took careful planning to ensure that wires could be managed and wouldn't get in the way of moving parts, and that all components could fit onto the wiring space available on the Arduino Mega and the motor controller boards, and everything would receive enough power to run properly. Once the plan was made, wiring began and final code started being developed in week 8. This process took a lot of testing and modifying as components broke and new issues arose. By

week 10, all components were assembled and the rest of the time was spent testing and tweaking code and electronics, including a Raspberry Pi 3B+ with a touchscreen for the user interface, until the very end.

Task:	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10
Design										
Manufacture										
Electronic testing										
Assembly										
Coding										
Testing										

Table 2. Spring Quarter Gantt Chart

3 Literature Review

3.1 MedMinder

Fig 1: MedMinder

MedMinder describes itself as an “all-in-one pharmacy and adherence solution.” For $2700, MedMinder pairs an automatic pill dispenser with home-delivered trays containing the patient’s prescriptions. The device itself can be described as a smart pillbox, with trays containing weeks’ worth of pills that may be inserted into the device. These trays may be locked, and open only when the patient is to take their prescriptions. The device also features an interactive screen

that provides reminders to take medication and even allows family members to record personalized messages for the patient. It uses WiFi to connect the device to a medical professional and allow for the user to optionally contact them in emergencies.

This device has a similar need statement to our device. With its easy-to-use interface with customizable reminder messages and its optional locking functionality, MedMinder intends to provide patients who struggle with taking their prescriptions a user-friendly and safe way to take their medications regularly. This product excels with its interactive screen, and the customizable reminder messages and connectivity with medical professionals are exceptional means by which to ensure that patients are up-to-date with their medications.

Though convenient, the device paired with the home-delivered trays sums to $2700, making it very expensive. The trays also must come from the MedMinder pharmacy, limiting the patient's options for their pharmacy of choice. The device itself is limited in that it only works with the MedMinder trays, and the trays come presorted rather than allowing the user to insert pills that get sorted by the device. In spite of these issues, MedMinder provides a good baseline for a well-designed user interface. Its interactive screen and smart features may serve as inspiration for the screen on our device.

3.2 Hero

Fig. 2: Hero

Hero is a device that allows users to dispense up to ten different kinds of pills at the push of a button. When it is time to take the prescription, the device sends a reminder to the user, prompting them to push a button to dispense the pills into a cup. Hero features a small form-factor dispenser with a small screen and an app that tracks prescription regimens, even including pills that may not fit in the device. This solution also includes protective measures, such as password protection, maximum dispensing limits and low stock warnings. Hero uses a

subscription model, with users paying at least $1079.76 for a three year subscription. Patent documentation (Fig. 2) reveals the mechanism Hero employs, which consists of a cylindrical carousel divided into separate compartments that different pills are collected in. A suction-based grabber is mounted to an arm that may translate vertically and in one direction horizontally. This is used to pick up a single pill, and drop it into the cup.

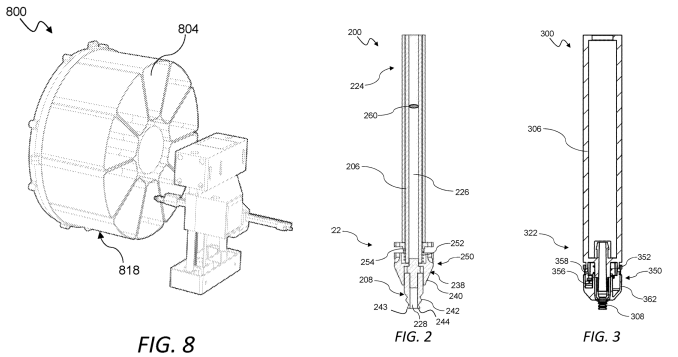


Fig. 3: Hero Patent Figures

This product also has a similar needs statement, using reminders and a simple interface to ensure users are able to keep track of their prescriptions. This product's successes lie in its sleek form and convenient interface. A smartphone user would benefit from the phone app's many features.

These features come at a high cost. The subscription model sums up to a large cost, especially excessive as it lacks the pill delivery service the MedMinder has. This high cost and the smartphone interface being required are both major faults in the product. In spite of this, the patents provided a good starting point for our design. Additionally, an optional smartphone-based interface may make our product more convenient without requiring that the user own a smartphone.

3.3 MedaCube

Fig. 4: MedaCube

MedaCube is a pill dispenser that is capable of storing and dispensing up to 16 different prescriptions for a 90 day period. This product dispenses a pill out of a small drawer that pops in and out. This particular dispenser targets caretakers, describing itself as a “strongbox for medication,” preventing the device from being opened unless it is unlocked for the caretaker to restock it. A PIN allows caretakers to access the device and change prescription information without allowing the patient access. The MedaCube is capable of sensing any tilting or handling, alerting the caretaker in this event. This device also tracks non-pill form prescriptions and as-needed medications, reminding the patient to take these medications or allowing the caretaker to set limits. Prescriptions can also be changed remotely.

The mechanism is similar to Hero’s, with its patent diagrams (Fig. 5) showing a similar vacuum system with a rotating cylindrical storage system.

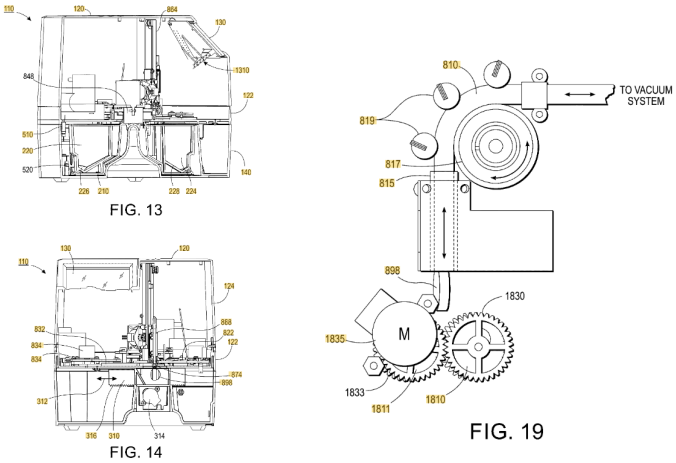


Fig 5: MedaCube Patent Figures

The need statement is very similar to the other two, in that it aspires to improve patient adherence to taking their medications. The MedaCube excels as a tool for caretakers, with its many safety features being excellent particularly for patients who may take more dangerous prescriptions. At $1800, it is rather expensive though it is a single, up-front payment. It is also a rather bulky design that may be able to be simplified. In spite of these issues, the MedaCube provides a very good baseline from which we may be able to design a safe product that is very convenient for caretakers in particular.

3.4 Rx-4 Automatic Pill Counter

Fig. 6: Rx-4 Automatic Pill Counter

The Rx-4 Automatic Pill Counter provides pharmacists with a fast and convenient means by which to count pills. It is capable of counting 500 pills per minute, and is compatible with 95% of pills. The device uses a central agitator to count a single pill at a time. This particular device serves a different purpose than the previous products. It intends to accelerate the process of sorting pills, and its target audience is exclusively pharmacists. The Rx-4 Automatic Pill Counter performs its task of counting pills extremely well, though it also shares the issue of being costly at $2499. Though not directly linked to our product, this device does require that a single pill is counted at a time, making it a very useful resource for designing a similar mechanism.

4 Preliminary Design and Experimental Results

4.1 Concept Generation and Evaluation

For our concept generation phase, we decided to divide it into three sub-teams to tackle the problem of dispensing one pill in different ways. They were as follows:

Agitator Team:

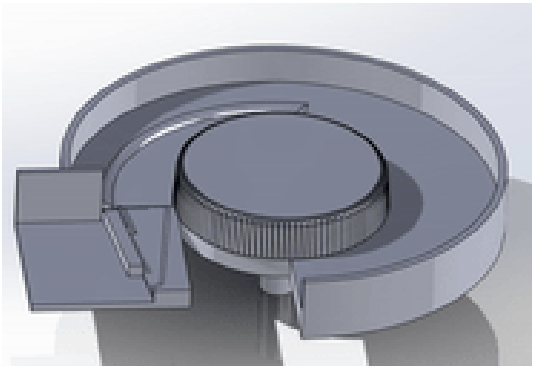


Fig. 7 Agitator Initial CAD

The agitator plate idea came from an existing product, the Rx-4 Automatic Tablet and Pill Counter, which uses an angled outer plate with a slightly off-centered rotating agitator to send pills into a container in a single file line so that they can be easily counted. We thought we could adapt this idea for our purposes of dispensing singular pills since one of the first steps was trying to get pills into an ordered line before selecting just one of them. As seen in the figure above, we created a simplified plate and agitator design, where the plate is angled in towards the agitator, and the agitator has small ridges to push the pills. Ideally, the pills are pushed in an ordered fashion down the tapered ramp, where a sliding gate would be opened enough for one to fall out at a time. Then, a sliding rake was created to sweep onto the flat off-ramp section, pushing the isolated pill off the side to be collected. Part of the design for this concept included a rack and pinion mechanism that was designed to allow for translational motion of a rack arm that would move pills across the stage. We found that conventional spur gears had dimensions that weren't flushing well with the design which is why the rack and pinion was designed to give good power transmission while agreeing with the very precise/tight dimensions of the stage.

From our initial prototype, we learned that we needed to make the outer plate smaller and more angled toward the agitator. The agitator itself needed larger ridges to "grab" onto the pills and move them towards the off-ramp. Additionally. the ramp itself was too shallow for the pills to slide down without a much faster agitator. So we made some changes to our initial print, and decided to incorporate the slide and gate design (see below) in place of the off-ramp design seen in the figure.

Slide and Gate Team:

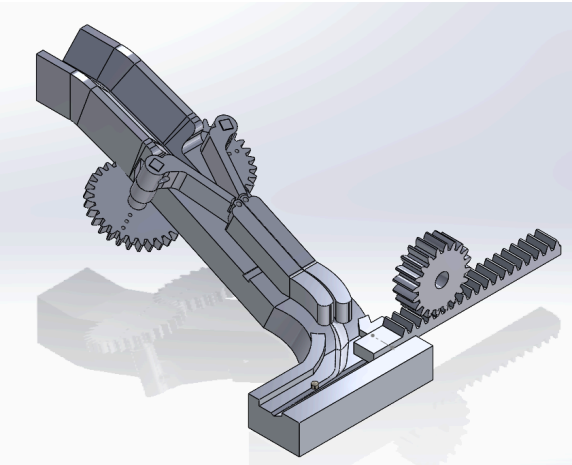


Fig. 8 Slide and Gate Initial CAD

The “slide and gate” design was inspired by a mechanism used to sort hex nuts during their manufacturing. This design relies on gravity to line the pills up, regardless of size. The “V” shaped profile would prevent the pills from stacking over or next to one another. The gate portion would serve as a funnel to further sort the pills and allow few at a time to pass through. This gate would be controlled by a pair of gears and a servo motor. A small shaft would connect the gate to a linkage. This linkage would move perpendicular to the slide, guided by a small joint and channels on both the slide and the linkage. The linkage would further line up pills up to the end channel, where a rack and pinion would push a single pill off of the slide.

Several iterations of the slide and gate were printed, with initial designs having slopes that were too shallow to allow for the pills to slide against the high-friction surface. The final iteration with an angle of 20 degrees proved to work well. This design ultimately proved to be overcomplicated with a number of issues. The linkage was likely too complex to serve as a redundancy, and the end channel and rack mechanism success rate was unknown.

PEZ Team:

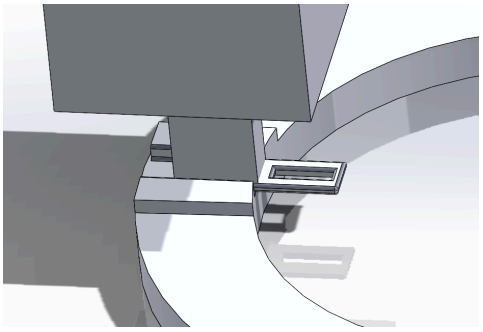


Fig. 9 PEZ Initial CAD

The PEZ design was inspired by the classic PEZ candy dispenser. It consisted of a hopper that funneled to a rectangular prismatic magazine. Pills would orient properly in the magazine as a result of selected geometry, pill size constraints, and potentially the assist of a miniature vibrating mechanism. The slide gate would be operated by some rack and pinion or four-bar mechanism. It would first push one pill out, which would then fall into the center. From there, the slide gate would retract until another pill was needed. This design takes advantage of gravity and a simple push and retract mechanism to dispense pills.

4.2 Selected Design

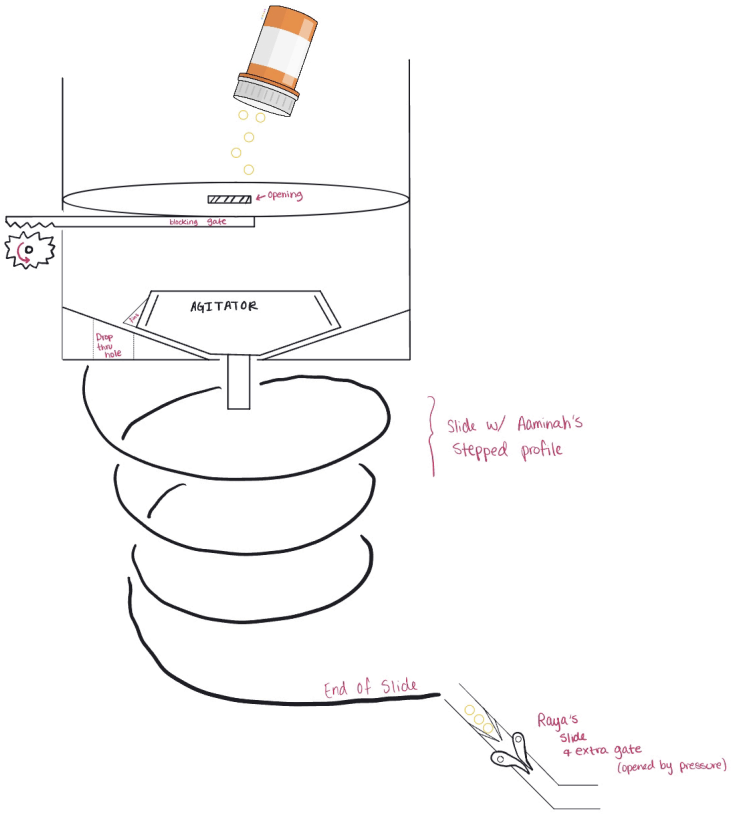


Fig. 10 Three-Stage Filtration Design

Our final design incorporates the successful aspects of our three preliminary subteam designs. It begins with a hopper that is loaded by the user from the top. Once pills are loaded, the

cap is put on and the machine should require no user interaction until pills have run out. The first stage consists of a funnel with an opening at the bottom. The opening is blocked by a gate that is controlled by a servo motor. From here, we can confidently say that a minimal amount of pills will fall through. Then, an agitator powered by a DC motor should drop fewer pills through the hole that feeds to the slide. The pill travels through the slide to the end which is blocked by a gate. At this gate, only one pill should be allowed to pass through as it is operated by pressure. This should inhibit more than one pill from passing through. The logic behind this is similar to that of a multi-stage water filtration system. We can ensure fewer pills will pass through each layer, but we can't say with high certainty that when presented with a large group of pills, that each layer would only permit one pill to pass on its own. This takes advantage of minimizing risk through probability, if each has a 10% chance of error, the final step should have a 0.1% chance.

4.3 Preliminary Analysis

As a team, we decided FEA or some form of computer-aided modeling would be too difficult to simulate the randomness of hundreds of pills. Thus, our testing has largely relied on iterative modeling and taken advantage of easy access to 3D printing. We began with evaluating the tolerance on Boelter Makerspace 3D printers to determine how we would need to modify our fittings' dimensions. We determined a necessary addition of 0.2 mm added to each hole diameter in order to properly fit components. Our first iterative correction was on the slope of the top hopper. The slope was too shallow and resulted in pills not falling through the hole as desired. The friction force between the pills and PLA was likely the main cause and as such, we printed a second model with a steeper slope that seemed to fix the problem. The current four "pronged" agitator model we have determined through testing should be reduced to a two prong model. This is because the four prongs were prohibiting pills from falling through the hole and instead just passing over the gap. On top of this, we have upgraded the motor to one that has better speed control. This was because the original Arduino UNO kit motors were inconsistent with speeds and had trouble overcoming the static friction force between the agitator and base of its container. As can be seen, a lot of our upgrades and component choices have been a result of trial and error. For the purposes of our project, this has shown to be the most efficient method of testing.

4.4 CAD Models

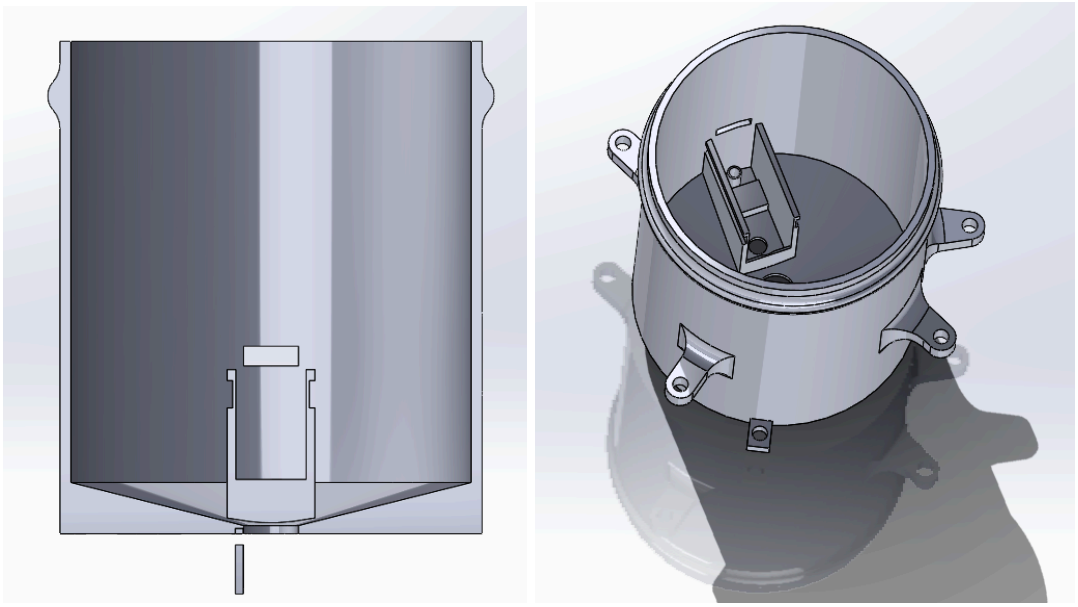


Fig. 11: W24 Funnel Design

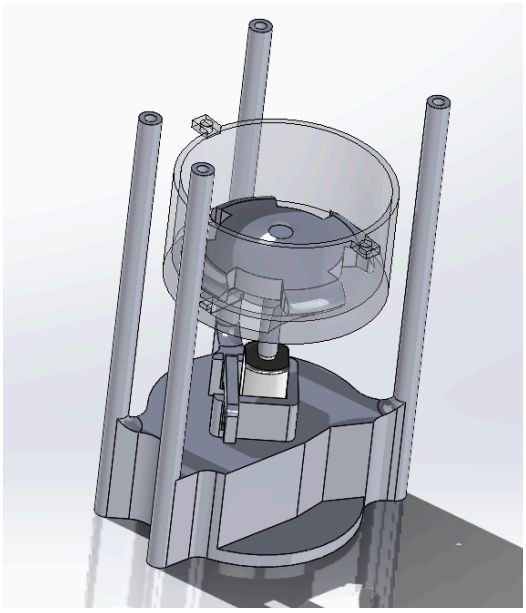


Fig. 12: W24 Agitator Design

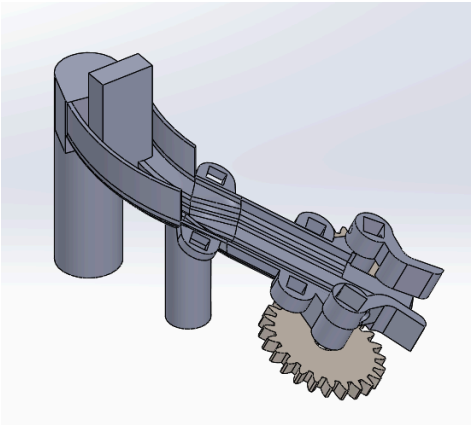


Fig. 13: W24 Slide and Gate Design

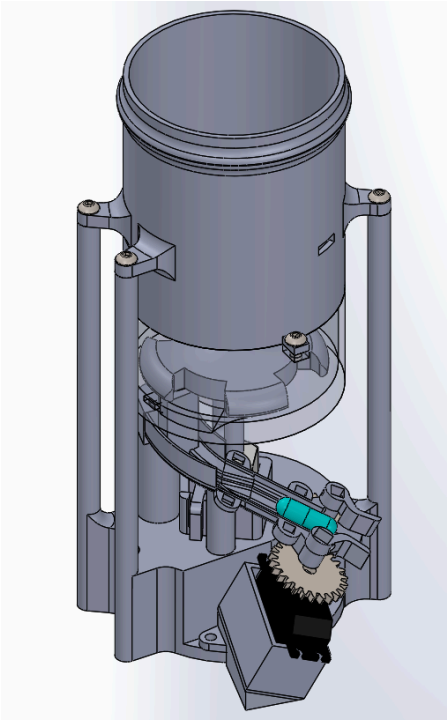


Fig. 14: W24 assembled CAD

The first subassembly of the final design is the funnel. This uses a servo with a rotating latch that opens and closes an orifice which allows for a small amount of pills to fall into the next agitator container hopper. This system also has a lid which serves to seal the pills off from air and allow them to be stored. It is shown in figure 11, overall the full gate container assembly contains 4 parts and 1 sg90 servo motor.

The following subassembly is the agitator hopper. Which uses a 6V DC motor to rotate a central agitator. Then the fallen pills into another orifice continuing along the journey towards dispensing. The whole idea is to provide another iteration of lowering of potential pills making it to the end of the dispense mechanism. This subassembly is shown in figure 12.

The final iteration of the slide and gate is given in Fig. 13. It is based heavily on the original slide and gate design, with a curve included to fit it within a smaller envelope. Earlier iterations of the slide featured multiple loops around the center axis, though these resulted in slopes that were too shallow, similar to earlier iterations of the original slide and gate. The rectangular extrusion near the top allows for the slide to be adhered to the agitator, and the bottom columns as well as the servo motor that is to be fastened to the gears may be attached to the base mount. The top of the slide features a stepped profile that allows the slide to work for different sized pills.

Similar to the original slide and gate, this step in the final design serves to ensure that the pills are in-line and only allowed to exit the entire assembly when the gate is open. Though not implemented in this iteration, small square tabs lie on the sides to allow for compliant flaps to be mounted onto the slide, which may help in lining up the pills further. Figure 14 shows this all assembled into the fully finalized early prototype of an omni-pill dispenser. The main intention was by taking advantage of the law of percentages the likelihood of success can be increased via filtering subassemblies. The next steps aim to improve manufacturability and serviceability of these components and create a more robust design that is better able to ensure the dispensing of only 1 pill of any size.

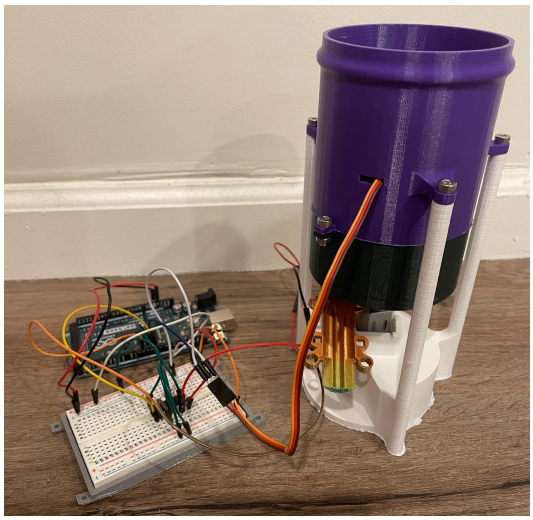


Fig. 15: W24 assembled physical prototype with circuitry included

5 Final Design and Experimental Results

5.1 Hopper Design and Components

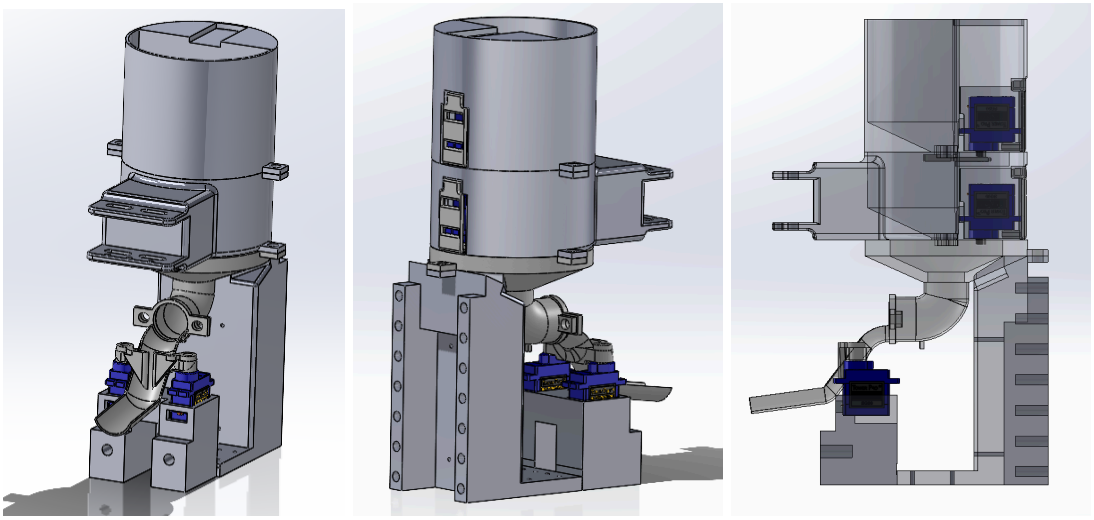


Fig. 16: Final Hopper Assembly CAD

The final design of the hopper mechanism (see CAD in Fig. 16 above and final assemblies in Fig. 17 below) followed the same idea of the hopper that was designed in winter quarter, but with modifications to make it more functional. The first major change was to remove the agitation plate that was between the container and the slides. The component was replaced with another latched container because we found that the agitation plate dispensed too many small pills, and would sometimes crush the pills as it spinned. By using two latched containers, we were able to filter out 1-3 pills for a variety of sizes, from about 1 mm - 10 mm diameter. The latches were controlled by servo motors. The first servo would open the first latch, revealing a 14 mm diameter hole, which seemed to be the optimal size to allow for a few pills to fall into the next container based on several design iterations (10 mm, 12 mm, 14mm, 16 mm). After the first latch closed, the second latch opened, allowing for a few pills to fall into the funnel portion, and down to the gates on the slide. Then, 2 servos slowly opened the gates in 5 degree increments to control the speed at which the pills fell. After opening fully, they swung shut, returning the hopper system to its initial ‘closed’ position. The angle and length of the slide were iterated to find the optimal dimensions such that the pills would fall from the slide to the delivery plate on the frame without over shooting. The 4 servos were connected to a PCA9865 motor controller, which in turn connected to an Arduino Mega and a 9V battery. Additionally, a grounding component was added to the hopper assembly to support the main components, offer mounting

points for the frame and an outer casing, and control the position of the hopper system relative to the frame.

Fig. 17: Final Assembled Hoppers

Although the hoppers were fairly reliable for some pill sizes, specifically small round pills, we had some issues with larger pills getting stuck in the various filtration steps. Even with the small pills, sometimes there were jams at the latch openings, or too many pills (>5) would be dispensed at once. The problem of too many pills dispensed was addressed using mechanisms in the frame, which will be described in the following section. For the problem of pills jamming at the latch opening, we decided that adding some sort of agitation element would help jostle the pills, allowing them to fall into the hole. Additionally, we increased the slope of the container so that pills would naturally fall towards the opening. To achieve agitation, we used vibrational motors to shake the top and middle containers and encourage pills to fall. Although the intensity of the vibration could not be controlled, we found that the duration and placement of the small motors could help us achieve the desired results. We found that the vibration was more effective when it occurred in parallel with the latch opening, rather than trying to shake the pills up before opening the latch. Initially we mounted the motors inside the container with the pills (see blue and red wires coming out yellow hopper in Fig. 17 above), but we decided that this was not a clean placement since the motors could fall and come in contact with pills, and the vibration was too intense for smaller pills. We then tried mounting the motors on the outside of the container, but this placement seemed to dampen the effect of the vibrations for moving the pills. Finally, we decided to mount the motors on the lid of the hopper. We modified the lids to have a central

opening for the wires to come out of, which also helped to suspend the motor away from the pills in case the adhesive on the back of the vibrational motor failed. With additional time, we would've added a loosely held in place piece on the lid-motor assembly to touch against the pills and ensure that the vibration effectively altered the orientation of the pills in the container.

5.2 Frame Design and Components

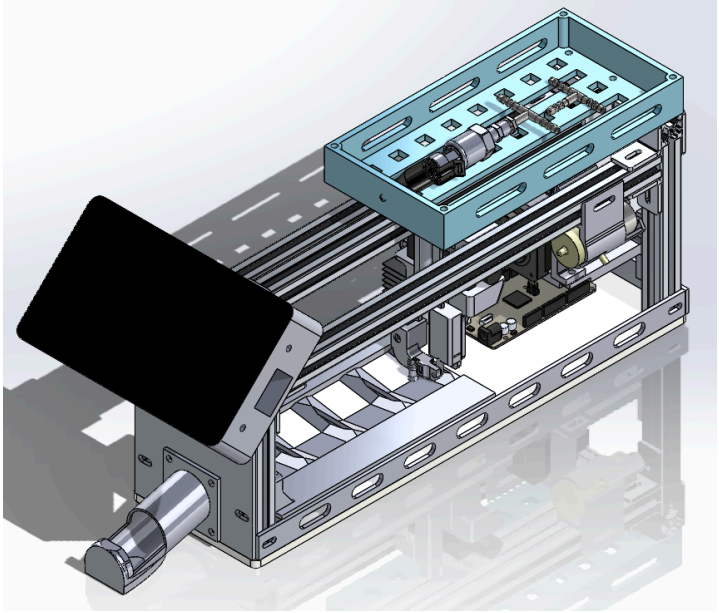


Fig. 18: Final Frame Assembly CAD

The frame served as the body of the final device. It was the main design and manufacturing component for the spring quarter, as it incorporated more materials and electrical components than the hoppers. It features an 80/20 skeleton for modularity and ease of mounting, held together by aluminum brackets and an acrylic base. The brackets were a custom design for the desired assembly, and manufactured by laser cutting aluminum sheets and bending the pieces accordingly. The back of the frame houses the main electrical components, including an Arduino Mega, SparkFun EasyDriver board, and L298N H-bridges for power distribution. The frame also had the final dispensing mechanism, which serves as the final filtration step in the off chance that more than one pill is dispensed by the hopper. Using two vacuum pumps in series, a stepper motor + ball screw, a linear actuator, and tubing, a "pick" mechanism was created. Pills fell from the hopper into the divided base plate (to keep medications separate). Then, the "pick" mechanism carriage traveled along the ball screw to the base plate with the pill (DoF #1) and dropped down to the plate (DoF #2) to grab the pill. A pressure transducer (seen in the blue top plate in Fig. 18 above) is used to detect a change in the pressure readings to determine whether or

not a pill has been successfully picked up. If a pill is picked up, the “pick” mechanism continues along the ball screw and deposits the pill in the spout at the front of the device. We used a thin film pressure sensor to indicate that the mechanism had reached the front of the frame and sent a drop signal to shut off the vacuum. If a pill was not picked up, the “pick” mechanism moved forward a few millimeters and repeated the process until it found a pill.

The plan for the final integrated code was to trigger Hopper X’s dispensing code if the “pick” mechanism could not detect a pill within the section of the base plate associated with Hopper X. Aside from the dispensing, the frame was also equipped to be plugged into the wall for power requirements and supported the case for the Raspberry Pi display which served as the User Interface. Since this project was meant to help elderly patients adhere to medication schedules, the UI was designed to be simple, with simple dispensing buttons to indicate each medication. With more time, we would have liked to develop a UI that could take a 1-time schedule input to dispense pills each day with information like time and number/type of medication required. This would have been an added creative feature, but the simple click and go commands that were included were sufficient in allowing a user to select the medicine they needed and receive it in 30-60 seconds.

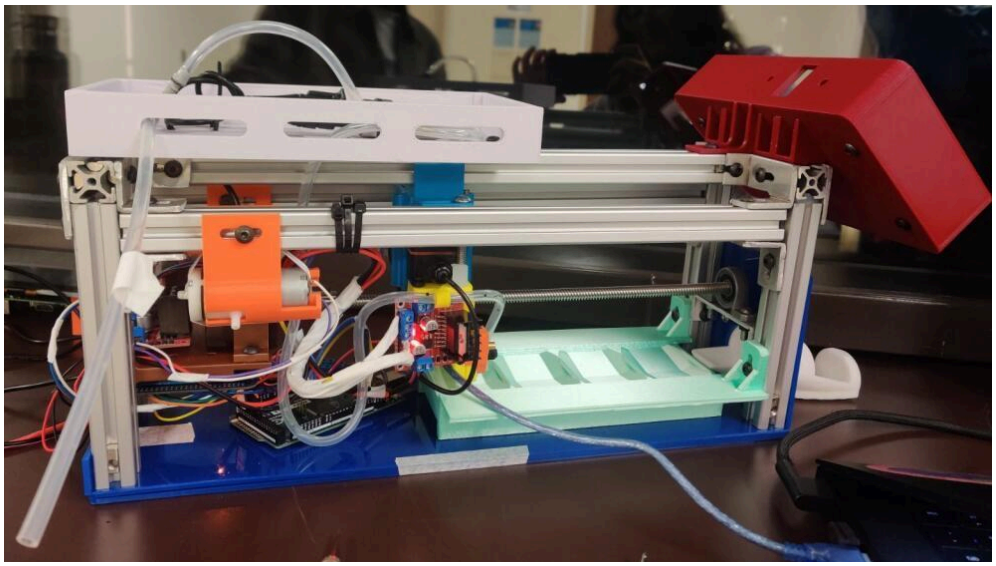


Fig. 19: Final Assembled Frame

The frame and its many components came together very well. There were of course iterations necessary for 3D printed parts for tolerancing and as new components were added. For example, the resin printed spout had to be reprinted so that it had a steep enough slope for the dispensed pills to fall to the end. Additionally, wire management and power distribution became

difficult with the number of electrical components and the translational motion of these components, but by grouping the wires and soldering them to be as short as possible the wire management was done quite well. The open structure and the modularity of the frame allowed for the accommodation of these components. The vacuum pumps and linear actuator and stepper had their issues along the way, working one day and then failing the next. As the system came together, troubleshooting became even more complicated; however, our team members were able to fix electronics or find cheap replacement options to ensure that the "pick" mechanism successfully delivered pills to the spout and that the vacuum pumps and linear actuators worked as intended in the end. Through careful electrical planning, wires were soldered into place to avoid disconnections and tubing was allowed to travel with the carriage using the slots available in the frame.

5.3 Final Assembly and Analysis

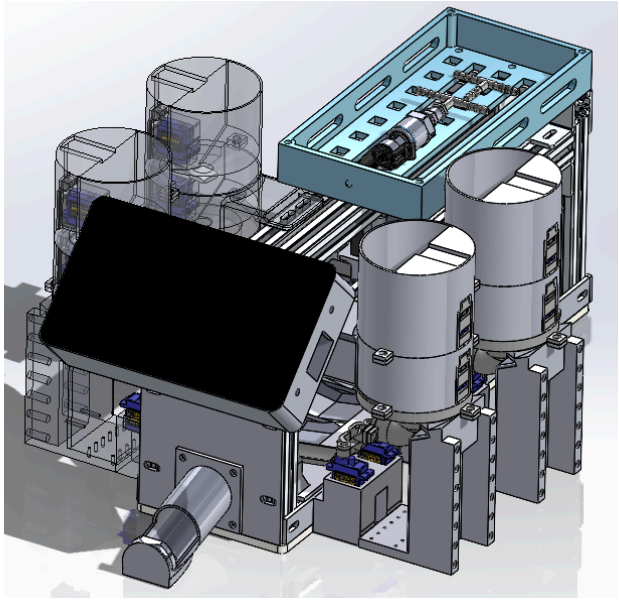


Fig. 20: Final Assembly CAD

The final assembly (CAD in Fig. 20 above, and physical prototype in Fig. 21 below) came together as a fully developed proof of concept project. We are pleased with the mechanical design of the system, and would've used additional time to manufacture an outer casing that was designed for the system to have a sleek, appealing final look. Up until the final assembly, we ran into minor tolerancing errors since many of our components were on the millimeter scale and it took us a while to get used to the appropriate tolerancing for these units while keeping the precision of the 3D printers in mind. Due to the number of components, each design iteration took at least a full day to print. The code for the frame was complex and extensive to control the

position of the carriage through the stepper motor, while also managing the linear actuator, the vacuum pumps, and the pressure transducer. This project included more electrical and software complexity than we originally expected but this proved to be a great learning experience. We were able to design, manufacture, and assemble two complex systems (hoppers and frame) that together gave us the medication management system we were looking to create.

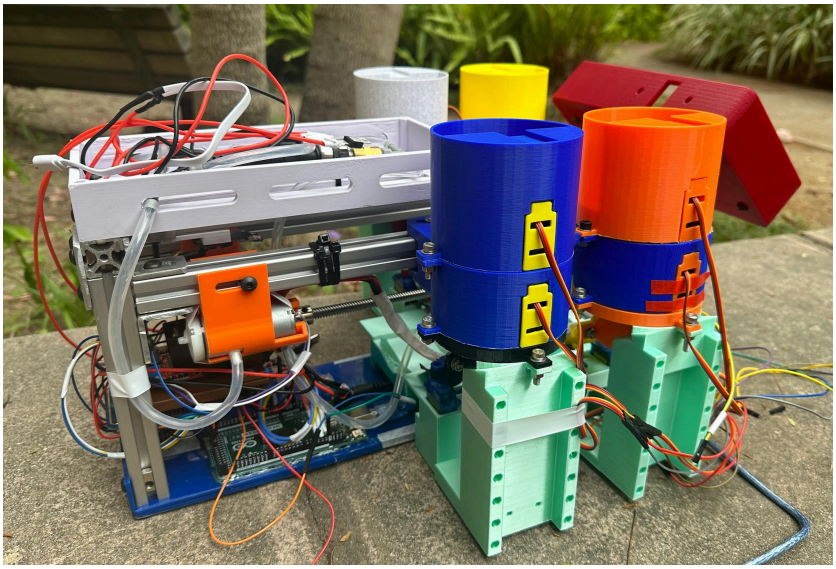


Fig. 21: Final Assembly

5.4 Electrical System/ Code Description

The complete Wiring Diagram as well as the Logical Flow chart for this project can be found in the Appendix. Both of these schematics greatly formed the basis of our project's electrical system as well as our code. All electronic components were wired to an Arduino Mega that controlled both subsystems. To power every component, a 12V DC adapter plugged directly into a wall outlet, which was then inserted into a power distribution board. This board powered all components, with a 5V power board reducing the voltage for components with smaller voltage requirements. The Raspberry Pi 3B+ was powered separately, with two USB Micro B cables powering both the Raspberry Pi and its touchscreen display.

The four SG90 micro servo motors for each hopper were attached to a PCA9685 Servo Driver, which in turn was connected to the Arduino Mega and a 9V battery. The vibration motor on each hopper's lid was also attached to the Arduino Mega. The code for this system triggered the topmost servo to open the first latch and turned the vibrational motor on at the same time. The latch in the base of the hopper subsequently closed and after a 0.5 second delay, the second latch was opened by triggering the second servo. At this point the vibrational motor was again turned

on. When the latch completed its cycle it closed and the vibrational motor also turned off. Finally, the servo motors at the bottom of the structure on opposite sides of the slide opened the slide gates in 5 degree increments to let the pill drop onto the delivery plate before closing shut. At this point the pill had completed its route through the hopper subsystem and the hopper was back to its starting position. In the final assembly, the code was to be triggered by the user selecting the pill that corresponds to a specific hopper and can also be triggered if the “pick” system does not detect a pill in the corresponding section of the delivery plate.

A stepper motor was controlled using an Sparkfun EasyDriver stepper motor controller, which was soldered onto a solderable breadboard with each wired connection being soldered and connected to either the motor, Arduino, or power distribution board as well. Two vacuum pumps were powered with a L298N H bridge which was then connected to both the Arduino and distribution board. A pressure transducer was powered directly from the Arduino. Its signal was input into one of the analog inputs on the Arduino. A thin film pressure sensor was wired to the Arduino with its signal input into an analog input and its ground connected to one of its ground inputs. A bump sensor was wired similarly. A linear actuator is wired to yet another L298N H bridge and its wires were taped together to prevent the wires from interfering with the motion of the tubing which was close to and moved with the carriage. This H bridge was also connected to both the Arduino and power distribution board.

The code for the picking function initially read for a user input from the Raspberry Pi touchscreen and then triggered the hoppers to start dispensing once the input was detected. The latches opened and the gates were incrementally opened until a pill (or a few depending on the size) were released. The stepper was then turned on to cycle through a certain number of steps until it was over the delivery location for the hopper in question. After this, the vacuum pumps were turned on and the current pressure was read. The linear actuator was turned on until it reached its full stroke and then it was pulled back. From here, the pressure was read again and if a change in the pressure was not read, the stepper motor moved forward one step and the process repeated. If a pressure change was read, then the stepper motor turned on and carried the carriage to the spout until the thin film pressure sensor read contact. At this point the linear actuator lowered and the vacuum pumps turned off in order for the pill to fall into the spout. From here, the linear actuator was raised and the stepper motor turned on in the reverse direction until it came in contact with the bump sensor at the other end of the frame where everything was homed. The system was then ready for its next user input.

6 Summary and Discussion

Over these two quarters, we were able to design a device that is capable of autonomously delivering a pill to a user with their given input. We were able to design two mechanisms that were fully functioning alone, with the hopper being capable of delivering a small amount of pills when prompted and the frame and pick mechanisms being able to carry a pill from any location on the device to the delivery spout given the user input from the Raspberry Pi touchscreen.

Taking the design we had developed in Winter quarter, we began by opting to make major changes to this design while also designing a vacuum-based pick mechanism for final delivery of the pills. We simplified the hopper design substantially, only using two latched containers and a slide and gate. Four of these were connected to the frame and pick assembly, which uses two vacuum pumps in serial, a stepper motor, and linear actuator to move a pill from the delivery points of those hoppers to the spout. The Raspberry Pi was programmed to communicate with the Arduino Mega that was controlling every electronic component and sending signals when the user desired a pill to be delivered.

Mechanically, the device was assembled using many different methods, including plasma cutting, bending, laser cutting, lathing, and other manufacturing methods, with the majority of parts being 3D printed. We had some difficulties with maintaining tolerances, though much of the design had large enough tolerances that they were easy to fix.

Major challenges were faced when managing the sensors. We initially planned to use IR sensors for most applications, but found them to be difficult to work with due to the fact that they were simply not precise enough to catch the motion of a fast-moving, small pill. The hoppers used timing to deliver pills, while the pick subsystem used a thin film pressure sensor and bump sensor. Both of these sensors also proved challenging, though the thin film pressure sensor eventually worked as intended. Other challenges were faced when managing electronics, with difficulty ensuring wires stayed soldered or in their corresponding pins, and with managing the lengthy code that caused two laptops to crash. In spite of this, the electronics largely worked as intended and the software we developed also demonstrated this.

To improve our product in the future, the hopper mechanism may be redesigned with its long-term storage kept in mind. For example, sealing each hopper and potentially using refrigeration to support the requirements of any type of pill over a months-long timespan are some potential features to look into. Some redundancies may be able to be removed with the addition of better tolerancing and sensors as well although this would entail significant iterative

testing that goes beyond proof of concept. The latter point can be applied to the pick mechanism. If more time is spent on tuning the software, one sensor may be able to be removed and the location of the carriage can be stored as a number of steps. Additionally, more time can be spent improving the UI to be more user-friendly and contain more features. Including a timer feature in the Raspberry Pi code would have been an interesting addition that would automatically deliver a medication at a predefined time.

If we circle back to the concerns that were posed at the end of the winter quarter it can be concluded that the main aspects that we were hoping to address were acknowledged in the spring quarter. Namely, there were concerns regarding (1) reliability, (2) robustness, (3) jamming, (4) pill verification, and (5) user interface. The concerns regarding reliability were simply related to whether or not the device would be able to consistently drop one pill at a time. Fortunately, the slide gates and latch systems that were incorporated into each individual hopper were successful in offering enough filtration to only allow one pill to fall through to the receiving plate under the influence of gravity.

Apart from dispensing one pill at a time there was also the concern of whether or not the mechanism design would be able to handle different sizes of pills (robustness). The two options available to us was to either use one design for all pill sizes which had features incorporated into it that accounted for size differences or use a few versions of one design that were tweaked slightly to accommodate different pill sizes. Fortunately, what we found was that rather than relying on the mechanical design to deal with different pill sizes, we could turn to the servo code for the slide gates. The code that was able to incrementally open the slide gates made it such that different sized pills could be used in a “universal” hopper and slide design without any significant modifications to account for different pill sizes. The openings in the bases of both joint hopper containers were adequately large enough for the largest pill size that we were testing and not too big to where an uncontrollable amount of our smallest pill size would fall through.

Another concern was the possibility of jamming occurring in any of the mechanical assemblies such as the agitation plate mechanism located in the floor of the hopper or the arms of the slide-and-gate mechanism. The slide gate design was completely modified along with the profile of the slide to ensure that these gates had enough range of freedom to move back and forth as intended without jamming with the slide or with a pill. This required that the profile of the slide gates was the negative profile of the slide and that the clearance between the gates and the slides was smaller than the thickness of the smallest pill. As for the agitation plate, some early testing confirmed that the rotating parts were very susceptible to jamming and also damaged the

smaller pills. Therefore, the agitation plate was completely taken out of the design and replaced with an additional hopper and latch that weren't prone to jamming.

These concerns related entirely to the mechanical design portion of the project, but one concern relating to software was having an accurate pill sensing mechanism that tracked the movement of the pill through the delivery plate. For the purpose of detecting a pill drop an IR sensor was included and although it faced difficulties with noisy data it worked well enough to indicate a pill drop. The more robust sensor used was the thin film pressure sensor at the far end of the frame where the spout was. This sensor detected contact with the carriage which alerted the vacuum pumps to turn off and allow the pill to drop into a spout. The code for the vacuum pumps, as mentioned in more detail earlier in this report, relied on changes in pressure differential to detect a pill. Therefore, the concern of a pill sensing mechanism was alleviated.

A final concern, and one that was also related to software, was designing a user interface that required minimal user input and was especially friendly for elderly individuals and their caretakers. We wanted to strike a balance between how complex we needed the set-up of the machine to be so that it could interface with the Arduino and operate each electrical component as intended and how simple we wanted the commands to be. Fortunately, we were able to develop the touchscreen such that clicking on the correct quadrant of the screen directly talked to one of the hoppers and initiated the pill drop. This interface was very easy to interact with as a user although it was quite difficult to code for the developer. All in all, the touchscreen was one of the highlights of the project that really brought it together and gave it the polished look that we were looking for.

The main safety concern with our project has less to do with the parts and pieces of the machine and more to do with the overall purpose of the machine. As long as the machine is dispensing accurately, which we were able to prove through testing, there are no other significant safety concerns related to our project. That in and of itself ties into the greater ethical impact of our project. If the machine doesn't dispense correctly, as with any other automated pill dispenser, the user runs the risk of missing a medication or taking the incorrect medication. Our job was to create a dispensing mechanism with enough redundancy that mitigates this danger. Having been successful in completing this project, the positive ethical impact of this project is huge. Taking medication daily is often taken for granted by individuals who are able to do so without any physical strain, but there is a large group of people who have the opposite experience. By catering our product specifically towards an elderly audience, our hope to make taking medications easy for patients with visual impairments, lack of basic motor skills, forgetfulness, etc. was fulfilled.

Overall, the biggest changes to be made are in time management. The final product achieved what we sought out to accomplish, however, we could have spent more time on electronics and programming to create a much more fleshed out product. Our mechanical design worked very well and robustly, and so putting more time towards the software portion of the design could have yielded better results. In spite of this, we still managed to create a working prototype that is easy to use, works for a wide range of pill sizes, and has the desired carrying capacity that we were looking to have. Some areas of interest for future iterations of this project include having more than four hoppers, having more features in the user-interface, including more degrees of freedom in the “pick mechanism”, etc..

References

“Features.” *MedMinder*, https://www.medminder.com/features/

“MedaCube Features.” *MedaCube*, https://www.medacube.com/pages/how-it-works

“Rx-4 Automatic Tablet & Capsule Counter.” *RxCount*, 2023, https://rxcount.com/?gad_source=1&gclid=CjwKCAjwte-vBhBFEiwAQSv_xQFC77E3dr4omgipEr0ziEQLMkj-p6gRtTMd9VIGR7SV_oIIvoLK1BoC_GQQAvD_BwE

Appendix

Bill of Materials

Date	Vendor	Qty	Unit	Unit Price	Catalog#	Description	Total Price	URL	Reason
03/13/24	Amazon (Bojack Electron)	1	pk(2)	$6.99	1803BK/1803B	Bojack Low Voltage DC Motor Speed Controlled PWM (adjustable driver switch)	$6.99	onur	Hopper Latch control
03/13/24						Tax	$0.66		
03/14/24	Amazon (WZHUIDA)	1	pk(270)	$9.99	None listed	M4 Assorted Hex Socket Head Cap Bolts (6/8/12/20/30mm) Screw, Nut, Washer	$9.99	onur	Fasteners for Hopper
03/14/24						Tax	$0.95		
03/18/24	Amazon (tatoko)	1	pk(20)	$13.99	None listed	10mmx3mm Mini Vibration Motors DC 3V 12000rpm Flat Coin Button-Type Micro DC Vibrating Motor	$13.99	https://www.amazon.com/tatoko-Vibration-Button-Type-Vibrating-Appliances/dp/B07Q1ZV4MJ	Vibration for Hopper (pill movement)
03/18/24						Tax	$1.33		
04/20/24	Amazon (Smraza Direct)	1	pk(10)	$17.99	1803BK/1803B	SG90 9G Micro Servo Metal Geared Motor Kit	$17.99	onur	Servo for Hopper
04/20/24						Tax	$1.71		
04/24/24	Amazon (ELEGOO)	1	pk(120)	$6.98	B01EV70C78	ELEGOO 120pcs Multicolored Dupont Wire 40pin Male to Female, 40pin Male to Male, 40pin Female to Female Breadboard Jumper Ribbon Cables Kit Compatible with Arduino Projects	$6.98	link	Wiring connections
04/24/24	Amazon (Hanaive)	1	pk(16)	$26.99	B0BBS54FYR	Hanaive 16 Pcs SG90 9g Micro Servo Motor Mini Servo Motors RC	$26.99	link	Servo Motors for

						Servo Kit for Robot Helicopter Airplane Car Boat Control			motion control
04/24/24	Amazon	2	pk(2)	$8.85	B07T1CHY58	2Pcs Thin Film Pressure Sensor 20g-2Kg DF9-16 High Precise Force Sensitive Resistor Force Sensor Pressure Sensor Resistance Type Thin Film Pressure Sensor Force Sensing Resistor	$17.70	link	Pressure sensor for pill pick-up verification
04/24/24	Amazon	1	pk(5)	$6.99	B00XAGSWR4	5 PCS IR Infrared Obstacle Avoidance Sensor Module for Arduino Smart Car Robot 3-Wire	$6.99	link	Infrared sensor for pill verification
04/24/24	Amazon (iexcell)	1	pk(100)	$9.96	B073H3GBJM	iexcell 100 Pcs M5 x 10 Thread Pitch: 0.8 mm Alloy Steel 10.9 Grade Hex Socket Button Head Cap Screws Bolts Kit, Black Oxide Finish	$9.96	link	M5 fasteners for assembly
04/24/24	Amazon	1	pk(100)	$16.68	B07VHNGBWJ	100-Pack 2020 2040 2060 Roll in Spring Loaded T Nut M5 for 20x20 20 Series Aluminum Extrusions 6mm Slot Aluminum Profile Accessories (m5)	$16.68	link	T-nuts from frame attachments
04/24/24	Amazon (Legines)	1	pk(2)	$5.99	B07487DR2Q	Legines Brass 1/8" NPT Female × 1/8" NPT Female Coupling Coupler Hex Head Pipe Fitting 1200psi (Pack of 2)	$5.99	link	Fastener fittings
04/24/24	Amazon	1	each	$18.39	B0BM6B6B73	Pressure Transducer Sender Sensor with Connector, Stainless Steel 1/8”-27 NPT Pressure Transducer Sensor Compatible for Oil Fuel Air Water (30 PSI)	$18.39	link	Pressure transducer for vacuum pump
04/24/24	Amazon (ReliaBot)	1	pk(2)	$4.99	B079HQ386R	2PCs Tr8x8 Lead Screw Brass Nut (Acme Thread, 2mm Pitch, 4 Start, 8mm Lead) for Diameter 8mm T8 Lead Screw of 3D Printer Z Axis	$4.99	link	Ball screw nut for vacuum pump
04/24/24						Tax	$10.88		
04/24/24	McMaster	1	ea	$19.33	8973K913	Corrosion-Resistant 3000 Series Aluminum Sheet, 0.125" Thick, 12"X12"	$19.33	link	Frame building material

04/24/24	McMaster	1	ea	$27.47	5537T101	T-Slotted Framing, Single Four Slot Rail, Silver, 20 mm Square, Solid, 8 ft. length	$27.47	link	
04/24/24	McMaster	1	ea	$0.77	5218K543	Plastic Barbed Hose Fitting, Inline Tee Adapter for 3/16" Hose ID, 1/8 NPT Male	$0.77		
04/24/24	McMaster	1	ea	$0.74	5218K502	Plastic Barbed Hose Fitting, Tee Connector for 3/16" Hose ID	$0.74		
04/24/24	McMaster	1	pk(50)	$8.87	90327A125	"Alloy Steel Low-Profile Socket Head Screws Hex Drive, Zinc Plated, M5 x 0.8 mm Thread, 10 mm Long"	$8.87		
04/24/24	McMaster	1	ea	$11.22	1441T12	"Slippery UHMW Polyethylene Film Adhesive-Back, 12"" Wide, 0.005"" Thick, 2 ft Length"	$11.22		
04/24/24						Tax	$6.50		
						GRAND TOTAL:	**$254.06**		

Non-PAC Orders:

Who	Where	When	What	Total
Shubh	CVS	3/5/24	Pills for testing	30.51
Shubh	Amazon	3/14/24	Linear Act, Pump	25.61
	“”	3/11	Resin	10.94
	“”	4/2	Electronics ($20 soldering)	35.02
	“”	4/3	Tubing	9.84
	“”	4/4	Another pump	9.98
	“”	5/8	M5 screws	9.84

	“”	5/13	Brass rods	8.74
	“	5/29	breadboard	9.84
Bradely	Amazon	5/20	Wall plug, expander, PCB	28.10
Onur	Amazon	5/14	Vibration motor	15.21
	“”	5/9	PWM driver, battery	29.08
	“”		Mini servo	13.13
	“”	5/29	Arduino Mega	20
Raya	Amazon	5/21	M3 M4 M5 kit	14.99
			Total:	**250.83**

Hopper Code

```
// Very Important To include in Final Code

#include <Wire.h>

#include <Adafruit_PWMServoDriver.h>

Adafruit_PWMServoDriver pwmController1 = Adafruit_PWMServoDriver(0x40);

Adafruit_PWMServoDriver pwmController2 = Adafruit_PWMServoDriver(0x41);


// Arduino Connections

const int OE1Pin = 2;     // OE pin for PCA9685 board 1


const int delayBetweenHoppers = 1000; // Do not need this at the final code probably


void setup() {

  Serial.begin(115200);

  Wire.begin();
```

```
  // Initialize PCA9685 board 1
  pwmController1.begin();
  pwmController1.setPWMFreq(50);  // Analog servos run at ~50 Hz updates

  // Initialize PCA9685 board 2
  pwmController2.begin();
  pwmController2.setPWMFreq(50);  // Analog servos run at ~50 Hz updates

  pinMode(OE1Pin, OUTPUT);

  digitalWrite(OE1Pin, LOW); // Enable PCA9685 board 1

  //initializeServos() Keep it commented out in final code

  // Disable both PCA9685 boards after initialization
  digitalWrite(OE1Pin, HIGH);
}

void loop() {

  // Function to call as needed.
  delay(500);
//  dispenseHopper1();
// delay(delayBetweenHoppers); // Wait for 5 seconds

//  dispenseHopper2();
//  delay(delayBetweenHoppers); // Wait for 5 seconds
```

```
//  dispenseHopper3();
//  delay(delayBetweenHoppers); // Wait for 5 seconds

 //dispenseHopper4();
//  delay(delayBetweenHoppers); // Wait for 5 seconds

  delay(100000); //Super long delay so I have time to re-upload the code for testing

}

void initializeServos() {
  // Maybe wont be used but keep it

  // Initialize PCA9685 #1
  setServoAngle(pwmController1, 0, 0);
  setServoAngle(pwmController1, 1, 0);
  setServoAngle(pwmController1, 2, 0);
  setServoAngle(pwmController1, 3, 90);

  setServoAngle(pwmController1, 4, 0);
  setServoAngle(pwmController1, 5, 0);
  setServoAngle(pwmController1, 6, 0);
  setServoAngle(pwmController1, 7, 90);

  // Initialize PCA9685 #2
  setServoAngle(pwmController2, 0, 0);
  setServoAngle(pwmController2, 2, 0);
```

```
  setServoAngle(pwmController2, 3, 0);
  setServoAngle(pwmController2, 4, 90);

  setServoAngle(pwmController2, 5, 0);
  setServoAngle(pwmController2, 6, 0);
  setServoAngle(pwmController2, 7, 0);
  setServoAngle(pwmController2, 8, 90);
}

// Code

void dispenseHopper1() {
  digitalWrite(OE1Pin, LOW); // Enable PCA9685 board 2
  dispenseServos(pwmController2, 0, 2, 3, 4);
  digitalWrite(OE1Pin, HIGH); // Disable PCA9685 board 2
}

void dispenseHopper2() {
  digitalWrite(OE1Pin, LOW); // Enable PCA9685 board 1
  dispenseServos(pwmController1, 0, 1, 2, 3);
  digitalWrite(OE1Pin, HIGH); // Disable PCA9685 board 1
}

void dispenseHopper3() {
  digitalWrite(OE1Pin, LOW); // Enable PCA9685 board 2
  dispenseServos(pwmController2, 5, 6, 7, 8);
  digitalWrite(OE1Pin, HIGH); // Disable PCA9685 board 2
}
```

```
void dispenseHopper4() {
  digitalWrite(OE1Pin, LOW); // Enable PCA9685 board 1
  dispenseServos(pwmController1, 4, 5, 6, 7);
  digitalWrite(OE1Pin, HIGH); // Disable PCA9685 board 1
}

void dispenseServos(Adafruit_PWMServoDriver &controller, int servo1, int servo2, int servo3, int servo4) {
  setServoAngle(controller, servo1, 45); // Adjust angles as needed for open position
  delay(1000); // Adjust delay as needed
  setServoAngle(controller, servo2, 45);
  delay(2000); // Time servos are open

  // Return to closed positions
  setServoAngle(controller, servo1, 0);
  setServoAngle(controller, servo2, 0);
  delay(3000); // Delay before next action

  int currentAngle = 0;

  // Increment until angle is 45
  while (currentAngle <= 45) {
    setServoAngle(controller, servo3, currentAngle);
    setServoAngle(controller, servo4, 90 - currentAngle);
    currentAngle += 5;
    delay(1000); // Adjust delay as needed
  }
```

```
  // Reset servo positions
  setServoAngle(controller, servo3, 0);
  setServoAngle(controller, servo4, 90);
  delay(3000);
}
// Crucial
void setServoAngle(Adafruit_PWMServoDriver &controller, uint8_t servoNumber, int angle) {
  // Convert angle to pulse length in microseconds
  int pulseLength = map(angle, 0, 180, 500, 2500); // Map angle to pulse length
  Serial.print("Setting servo ");
  Serial.print(servoNumber);
  Serial.print(" to angle ");
  Serial.print(angle);
  Serial.print(" (pulse length ");
  Serial.print(pulseLength);
  Serial.println(")");

  // Set PWM pulse length
  controller.writeMicroseconds(servoNumber, pulseLength);
}
```

Frame Code

```
#include <math.h> // for abs

////////////////////////////////////////////////////////////////////////////////
// Hopper Constants
////////////////////////////////////////////////////////////////////////////////
const int HOPPER1 = 1;
```

```
const int HOPPER2 = 2;

const int HOPPER3 = 3;

const int HOPPER4 = 4;


////////////////////////////////////////////////////////////////////////////

// Stepper Constants

////////////////////////////////////////////////////////////////////////////

//Declare pin functions on Redboard

#define stp 39

#define dir 37

#define MS1 41

#define MS2 43

#define EN 45


//Declare variables for functions

char user_input;

unsigned long x;

int y;

int state;

//int resetEDPins;


// Step to Distance Constants

const int stepsPerRevolution = 200; // Based on Motor Spec Sheet

const float pitch = 2.0;

const int microStep = 8;

const float stepDistance = pitch / (stepsPerRevolution * microStep); // mm/step


// Hopper Positions

const float hopperPositions[6] = {12,20,40,60,80,169};
```

```
////////////////////////////////////////////////////////////////////////////
// Linear Actuator Constants
////////////////////////////////////////////////////////////////////////////
const int extend = 11; // L298N IN3 Pin orange
const int retract = 12; // L298N IN4 Pin green
const int enaPinLin = 13; // L298N ENA Pin

////////////////////////////////////////////////////////////////////////////
// Vacuum Pump Constants
////////////////////////////////////////////////////////////////////////////
const int suckVac1 = 51; // L298N IN1 Pin
const int suckVac1Alt  = 49; // L298N IN2 Pin
const int enaPinVac1 = 53; // L298N ENA Pin
const int suckVac2 = 48; // L298N IN3 Pin
const int suckVac2Alt = 50; // L298N IN4 Pin
const int enaPinVac2 = 52; // L298N ENB Pin

//////////////////////////////////////////////////////
// Sensor Constants
//////////////////////////////////////////////////////
#define sensor A1
#define transducer A0
#define sensorBegin 4

void setup() {

  ////////////////////////////
  // Stepper Setup
  ////////////////////////////
  pinMode(stp, OUTPUT);
```

```
pinMode(dir, OUTPUT);
pinMode(MS1, OUTPUT);
pinMode(MS2, OUTPUT);
pinMode(EN, OUTPUT);
resetEDPins(); //Set step, direction, microstep and enable pins to default states
Serial.begin(9600); //Open Serial connection for debugging

//////////////////////////////
// Linear Actuator Setup
//////////////////////////////
pinMode(extend, OUTPUT);
pinMode(retract, OUTPUT);
pinMode(enaPinLin, OUTPUT);
digitalWrite(extend, LOW);
digitalWrite(retract, LOW);
digitalWrite(enaPinLin, HIGH);

//////////////////////////////
// Vacuum Setup
//////////////////////////////
pinMode(suckVac1, OUTPUT);
pinMode(suckVac1Alt, OUTPUT);
pinMode(enaPinVac1, OUTPUT);
pinMode(suckVac2, OUTPUT);
pinMode(suckVac2Alt, OUTPUT);
pinMode(enaPinVac2, OUTPUT);
digitalWrite(suckVac1, LOW);
digitalWrite(suckVac1Alt, LOW);
digitalWrite(suckVac2, LOW);
digitalWrite(suckVac2Alt, LOW);
```

```
  digitalWrite(enaPinVac1, HIGH);
  digitalWrite(enaPinVac2, HIGH);

  ////////////////////////////
  // Sensor Setup
  ////////////////////////////
  pinMode(sensor, INPUT);
  pinMode(transducer, INPUT);
  pinMode(sensorBegin, INPUT);

}

void loop() {
hopperRunThrough(HOPPER1);
}

////////////////////////////////////////
// Transducer Functions
////////////////////////////////////////
float movingAverageTransducer(){
  float transducerReading = 0;
  int transRunThroughs;
  for(int i = 0; i<10; i++){
    float transValue = analogRead(transducer);
    transducerReading += transValue;
    transRunThroughs++;
  }
  return transducerReading/transRunThroughs;
}
```

```
////////////////////////////////////////

// Hopper Functions

////////////////////////////////////////

void hopperRunThrough(int userInput){

float stepsTraveled = 0;

delay(5000);

linearRetract();

delay(1000);

vacuumOn();

delay(2000);

digitalWrite(EN, LOW);

SmallStepMode(hopperPositions[userInput],stepsTraveled);

float initialTransReading = movingAverageTransducer();

Serial.println(initialTransReading);

linearExtend();

delay(5000);

linearStop();

delay(1000);

linearRetract();

delay(5000);

linearStop();

delay(1000);

float finalTransReading = movingAverageTransducer();

Serial.println(finalTransReading);

if(abs(finalTransReading - initialTransReading)>30){

  moveToEnd();

  moveToEnd();

  moveToEnd();

  moveToEnd();

  moveToEnd();
```

```
  }
else{
  for(int i = 1; i<8; i++){
    vacuumOn();
    delay(2000);
    digitalWrite(EN, LOW);
    SmallStepMode(3,stepsTraveled);
    float initialTransReading = movingAverageTransducer();
    Serial.println(initialTransReading);
    linearExtend();
    delay(5000);
    linearStop();
    delay(1000);
    linearRetract();
    delay(5000);
    linearStop();
    delay(1000);
    float finalTransReading = movingAverageTransducer();
    Serial.println(finalTransReading);
    if(abs(finalTransReading - initialTransReading)>20){
      moveToEnd();
      break;
      }
  }
}

vacuumOff();
linearExtend();
delay(5000);
linearRetract();
```

```
delay(5000);
moveToBegin();
}
//////////////////////////////////////////
// Stepper Functions
//////////////////////////////////////////
// 1/8th microstep foward mode function
void SmallStepMode(float distance,float & stepsTraveled){
  Serial.println("Stepping at 1/8th microstep mode.");
//// Decide if travel is forwards or backwards
  if (distance > 0){
    digitalWrite(dir, LOW); //Pull direction pin low to move "forward"
    }
  else if (distance < 0){
    digitalWrite(dir, HIGH); //Pull direction pin high to move "backwards"
  }
  else{
    return;
  }
//// Pull MS1, and MS2 high to set logic to 1/8th microstep resolution
  digitalWrite(MS1, HIGH);
  digitalWrite(MS2, HIGH);
//// Calculate Steps Needed
  unsigned long stepsNeeded = distToSteps(distance);
//// Move Desired Steps
  for(x = 1; x<=stepsNeeded; x++) { //Loop the forward stepping enough times for motion to be visible
    digitalWrite(stp,HIGH); //Trigger one step forward
    delay(1);
    digitalWrite(stp,LOW); //Pull step pin low so it can be triggered again
    delay(1);
```

```
  }
  stepsTraveled = stepsNeeded;
}

// Moves one big step
void getToSteppinBigSteppa(){
//// Pull MS1, and MS2 low to set logic to 1/8th microstep resolution
  digitalWrite(MS1, LOW);
  digitalWrite(MS2, HIGH);
  digitalWrite(dir, LOW);
  digitalWrite(stp,HIGH); //Trigger one step forward
  delay(1);
  digitalWrite(stp,LOW); //Pull step pin low so it can be triggered again
  delay(1);
}

// Moves one big step
void getToSteppinBigSteppaREV(){
//// Pull MS1, and MS2 low to set logic to 1/8th microstep resolution
  digitalWrite(MS1, LOW);
  digitalWrite(MS2, HIGH);
  digitalWrite(dir, HIGH);
  digitalWrite(stp,HIGH); //Trigger one step forward
  delay(1);
  digitalWrite(stp,LOW); //Pull step pin low so it can be triggered again
  delay(1);
}

// Moves To End
void moveToEnd(){
```

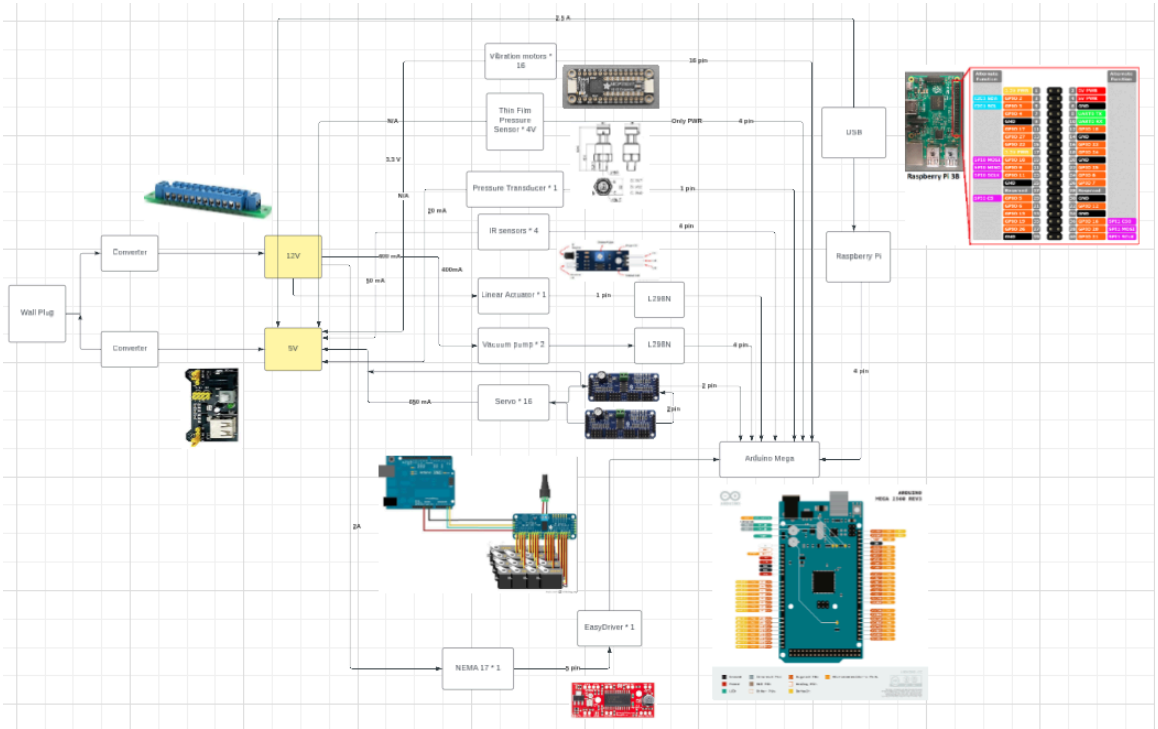
```
  while(true){
    digitalWrite(EN,LOW);
    getToSteppinBigSteppa();
    int sensorValue = analogRead(sensor);
    Serial.println(sensorValue);
    // Check if the sensor value is zero
    if (sensorValue == 0) {
      int sensorValue = analogRead(sensor);
      Serial.println(sensorValue);
        if (sensorValue == 0)
          { Serial.println("You've hit the end!");
            break;}
    }
}
}

void moveToBegin(){
  for(int i = distToSteps(75);i>=1; i--){
    digitalWrite(EN,LOW);
    getToSteppinBigSteppaREV();
    int sensorValue = analogRead(sensor);
    Serial.println(sensorValue);
//    // Check if the sensor value is zero
//        if (sensorValue == 1)
//          {break;}
}
}



//Reset Easy Driver pins to default states
```

```
void resetEDPins()
{
  digitalWrite(stp, LOW);
  digitalWrite(dir, LOW);
  digitalWrite(MS1, LOW);
  digitalWrite(MS2, LOW);
  digitalWrite(EN, HIGH);
}

// Convert distance to steps based on info
unsigned long distToSteps(float distance){
  unsigned long stepCount = distance/stepDistance;
  stepCount/=10;
  stepCount*=2;
  stepCount*=20;
  stepCount/=17;
  if(distance>=100){
    stepCount*=20;
    stepCount/=17;
  }
  return stepCount;
}

//////////////////////////////////////////
// Linear Actuator Functions
//////////////////////////////////////////
// Extend the linear actuator
void linearExtend(){
  digitalWrite(retract, LOW);
  digitalWrite(extend, HIGH);
```

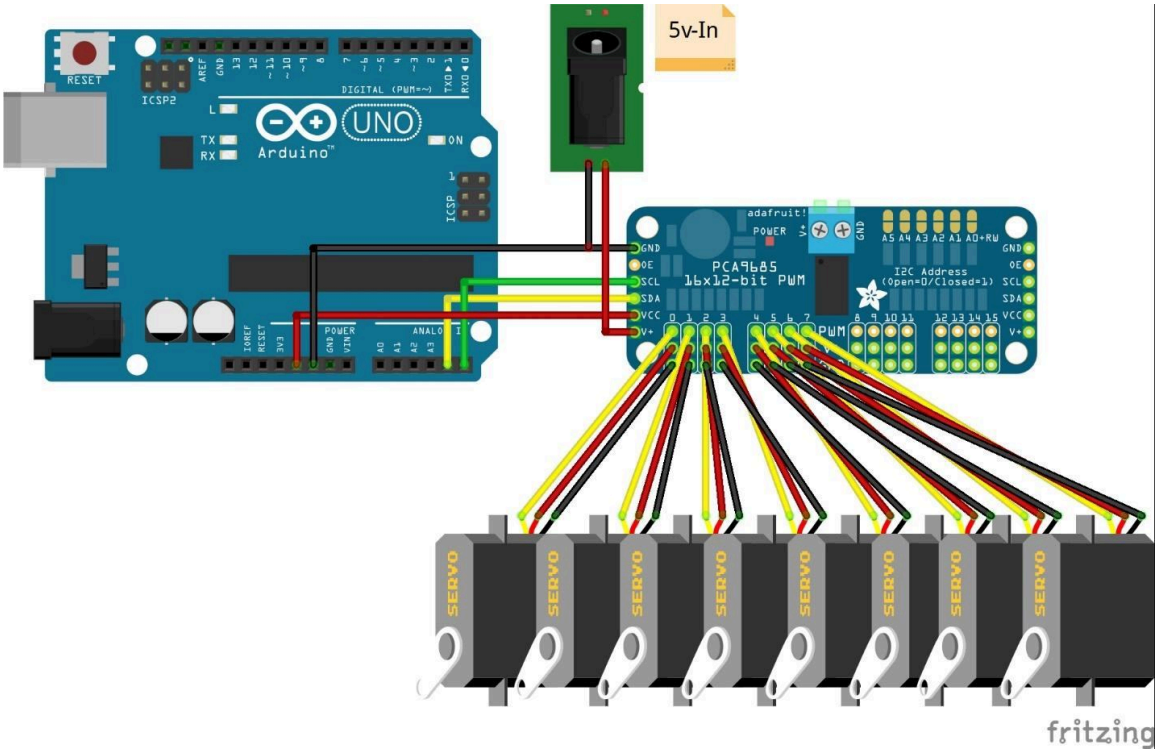
```
}

// Retract the linear actuator
void linearRetract(){
  digitalWrite(retract, HIGH);
  digitalWrite(extend, LOW);
}

// Stop the linear actuator motion
void linearStop(){
  digitalWrite(retract, LOW);
  digitalWrite(extend, LOW);
}

////////////////////////////////////////////
// Vaccum Functions
////////////////////////////////////////////
// Turn on the vacuum
void vacuumOn(){
  digitalWrite(suckVac1Alt, LOW);
  digitalWrite(suckVac1, HIGH);
  digitalWrite(suckVac2Alt, LOW);
  digitalWrite(suckVac2, HIGH);
}

// Turn on the vacuum (alternate motor direction)
void vacuumOn2(){
  digitalWrite(suckVac1Alt, HIGH);
  digitalWrite(suckVac1, LOW);
  digitalWrite(suckVac2Alt, HIGH);
```

```
  digitalWrite(suckVac2, LOW);
}

// Turn off the vacuum
void vacuumOff(){
  digitalWrite(suckVac1Alt, LOW);
  digitalWrite(suckVac1, LOW);
  digitalWrite(suckVac2Alt, LOW);
  digitalWrite(suckVac2, LOW);
}
```

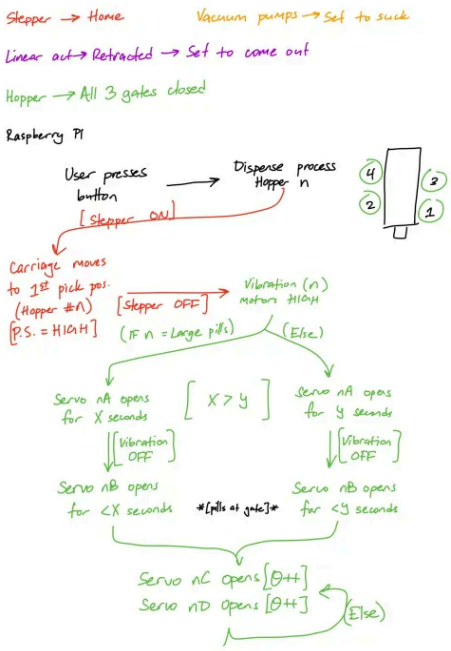
Electrical Diagram

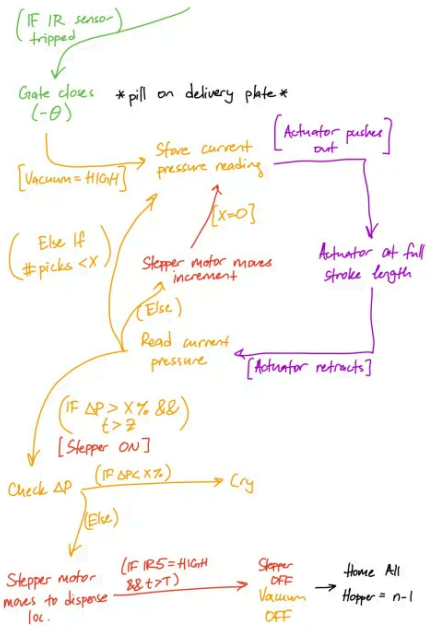


Hopper Wiring Diagram



Logical Flow Diagram



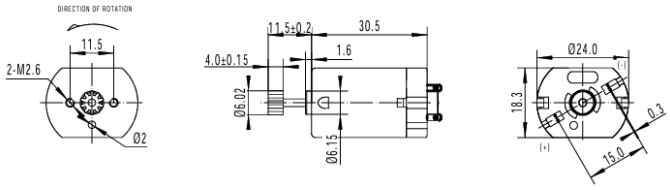


DC Motor Data Sheet

DC motor 6/9V

Item	Specification	Reference
Rated Voltage	6V DC	
No load speed	12000±15%rpm	
No load current	≤280mA	
Operating voltage	1.5-6.5V DC	
Starting Torque	≥250g.cm(according to ourself developed blade)	
starting current	≤5A	
Insulation Resistance	above 10Ω between the case and the terminal	DV 100V
Rotation Direction	CW:[+]terminal connected to the positive power supply,[-]terminal connected to nagative power,clockwise is deemed by the direction of the output shaft	
shaft gap	0.05-0.35mm	

TFK-280SA-22125 MOTOR



Pressure Angle	20°
Number Of TEETH	10
PITCH	0.5 Module
Outside Diameter	6.02

MOTOR Specification

Operating voltage: 6.0V DC

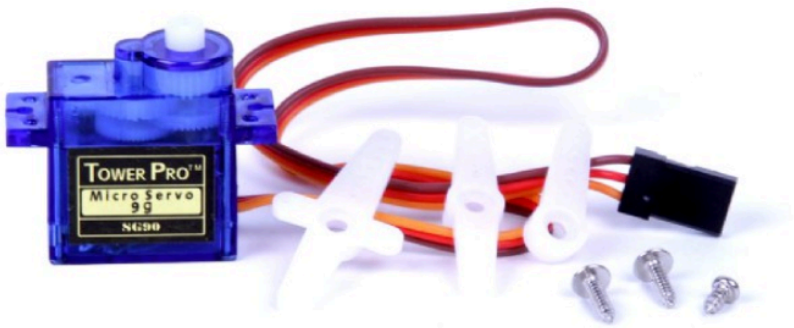
1、No Load Speed:7500±10%rpm
2、No Load Current:90mA (120mA max)
3、30g.cm Load Speed:6100rpm

4、Stall Torque:150g.cm min
5、Stall Current:2.6A (3.2A MAX)

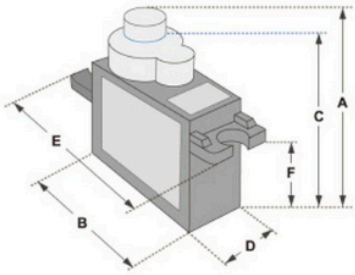
TITLE:TFK-280SA-22125	Customer No:	Drafted By: Kevin wang
		Checked By:
Dwg. NO: bm22125-002	Date: 2015.06.15	Approved By:
TT MOTOR (HK) INDUSTRAIL CO.,LTD	Scale: 1:1	Units: mm EDITION No: A
		Angular: =1° Decimal: X.XX ±0.05 X.X ±0.2 X ±0.3

Servo Motor Data Sheet

SERVO MOTOR SG90 DATA SHEET

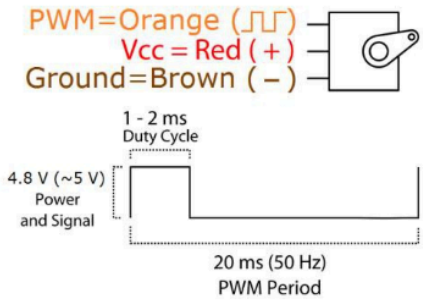


Tiny and lightweight with high output power. Servo can rotate approximately 180 degrees (90 in each direction), and works just like the standard kinds but smaller. You can use any servo code, hardware or library to control these servos. Good for beginners who want to make stuff move without building a motor controller with feedback & gear box, especially since it will fit in small places. It comes with a 3 horns (arms) and hardware.



Dimensions & Specifications
A (mm) : 32
B (mm) : 23
C (mm) : 28.5
D (mm) : 12
E (mm) : 32
F (mm) : 19.5
Speed (sec) : 0.1
Torque (kg-cm) : 2.5
Weight (g) : 14.7
Voltage : 4.8 - 6

Position "0" (1.5 ms pulse) is middle, "90" (~2ms pulse) is middle, is all the way to the right, "-90" (~1ms pulse) is all the way to the left.



Vibration Motor Data Sheet

11.外形图/Outside View