Home Pharm-Assist

Tyler Bednarz, Keynan Jones, Marc Le Roy, Ben Marks

Department of Electrical & Computer Engineering and Computer Science

Ohio Northern University

Ada, Ohio 45810

t-bednarz@onu.edu, k-jones.12@onu.edu, m-leroy@onu.edu, b-marks@onu.edu

Abstract

Approximately 30% of hospital admissions amongst elderly patients are related to prescription medications problems. These problems stem from overdoses, drug toxicity, and patients simply missing doses. Elderly patients on multiple medications can easily confuse prescription times, doses, purposes, colors, shapes, etc.

We propose to create a device which will dispense a patient's prescription dose at the exact administration time. This device is filled at a pharmacy with the patient's prescription medications by a licensed pharmacist. The pharmacist is responsible to program the device for proper drug dosages and administration times. Then, our product dispenses the appropriate dosage and alerts the patient to take the medication.

Our product will help elderly people to accurately take their prescribed medications. This device is primarily targeted towards elderly people because they tend to require multiple daily prescriptions. However, it could be used by anybody with similar requirements.

We plan to submit a working prototype to the conference that will highlight the device functionality.

Introduction

Based on research of current market solutions, there are not many home automated solutions to help senior patients with taking their medications. Many of the solutions are either too expensive, or difficult for the patient to use. The cheapest solutions on the market are approximately $800. These solutions are often not very helpful for the patient, as the patient still has to sort out their pills and put them in individual capsules that are then distributed at the given time. With our proposed design, the patient would not need to do this extra step. The pharmacist would simply dump the entire prescription regime into the pill hoppers and then programs the device with the corresponding dosage times and quantity. The patient would not need to have any interaction with the device except for taking the pill when it is dispensed.

Proposed Design

Full System

The proposed design will hold eight different prescriptions. The shell of the device is shown below in Figure 1. It consists of two main components: a rotating storage unit and an external housing. The rotating storage unit consists of eight cylinders used to contain the different types of pills, aligned in a circluar formation on the rotating table (shown in Figure 1). Each cylinder has a circluar opening at its base for the actuator and vacuum to enter, where the vacuum will grab hold of a selected pill type and lift it up for removal. The external housing has a hole of similar size in its base, which will align with the appropriate cylinder. The torque needed to turn the table to its designated location will be provided by the stepper motor. The motor's shaft will be secured in the extrusion shown in Figure 1, with a press fit and set screw.

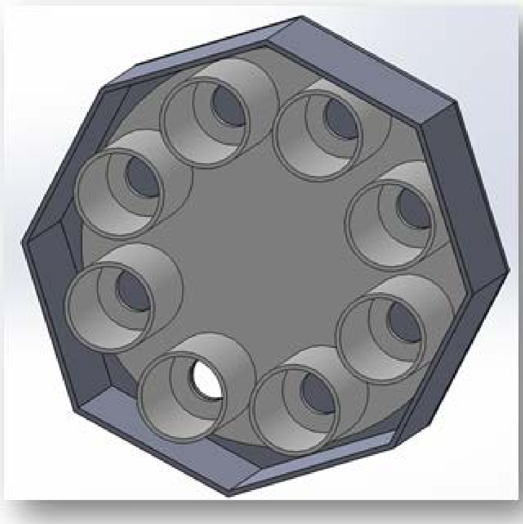
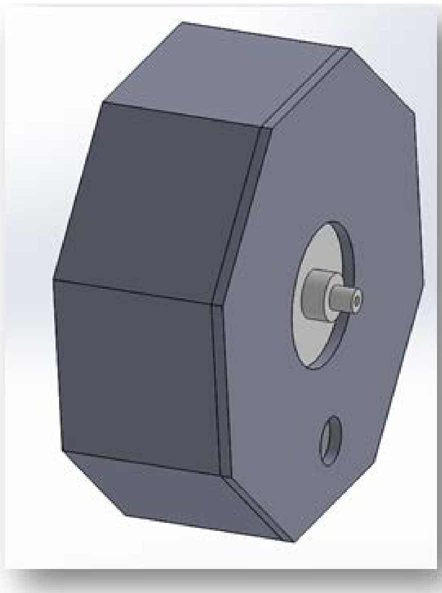



Figure 1: Top and Bottom views of device shell

When it is time to dispense a pill from a hopper, the stepper motor turns the rotating disc so that the desired hopper is directly over the linear actuator. The position sensor on the shaft of the stepper motor ensures that the hopper is in the correct position. Once the hopper is above the linear actuator, the actuator pushes up through the bottom of the hopper and the vacuum pump is turned on. The vacuum pump sucks a pill onto the end of the tube and holds it while traveling upwards. A sensor on the vacuum takes continuous samples of the vacuum voltage. Once the vacuum tube opening is covered with a pill, the vacuum voltage spikes, thus indicating a successful pick. Thus, providing an essential feedback loop to ensure successful pill picks.

If the pill is not correctly picked or is dropped mid-sequence, the device restarts the sequence. Once the actuator is fully extended, an alarm sounds to alert the patient that it is time to take their medication. Once the patient takes the medication from the vacuum hose, the vacuum pump shuts off and the actuator fully retracts to its normal position. The device then enters a standby mode until the next medication needs to be dispensed. An Arduino microcontroller controls all of the electrical mechanisms. The flowchart of the aforementioned sequence is summarized in Figure 2. Figure 3 shows an overall block diagram of the system.

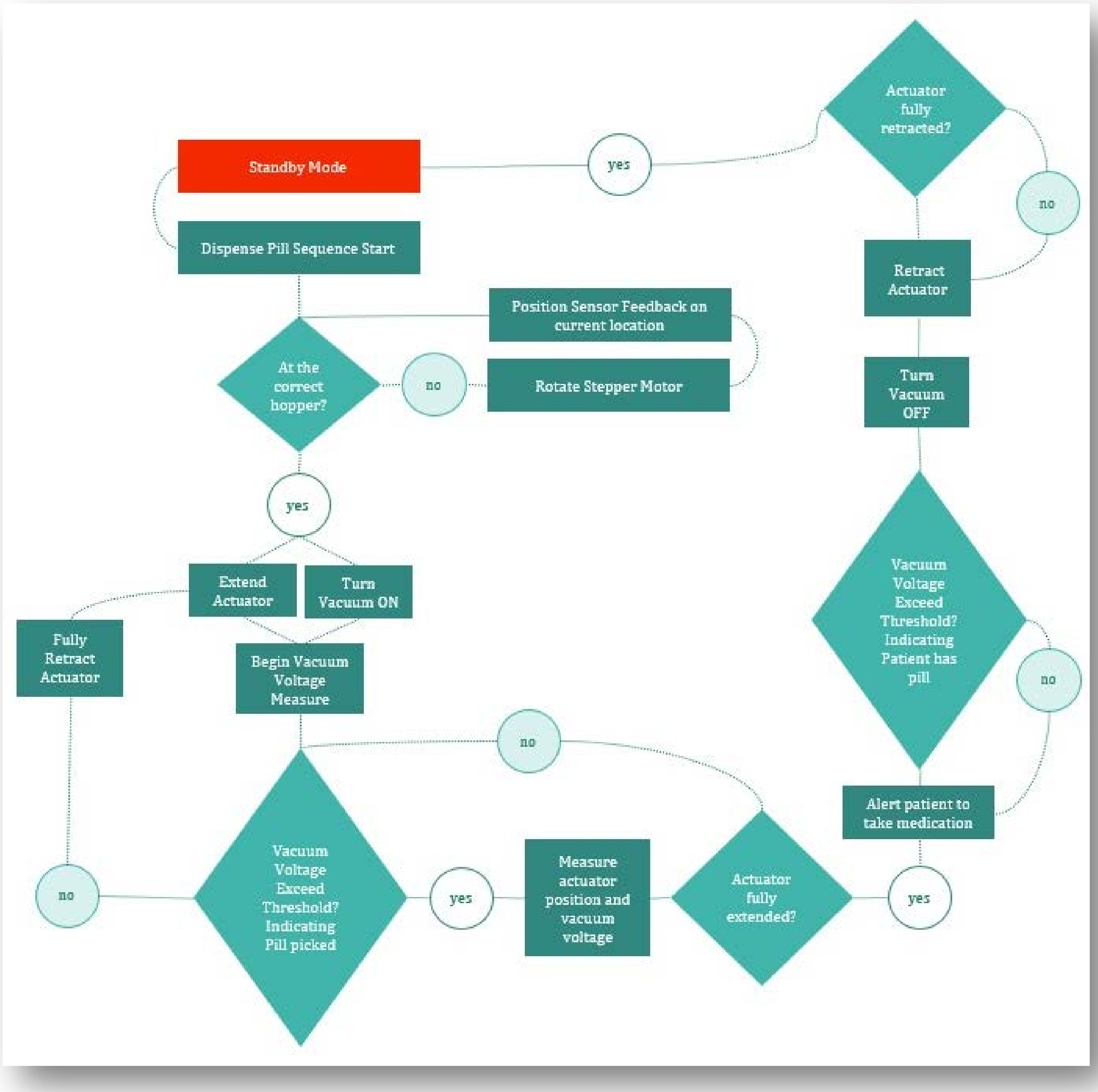

Standby Mode
yes
Actuator fully retracted?
no
Dispense Pill Sequence Start
Retract Actuator
Position Sensor Feedback on current location
At the correct hopper?
no
Rotate Stepper Motor
Turn Vacuum OFF
yes
Extend Actuator
Turn Vacuum ON
Vacuum Voltage Exceed Threshold? Indicating Patient has pill
no
Fully Retract Actuator
Begin Vacuum Voltage Measure
no
Alert patient to take medication
no
Vacuum Voltage Exceed Threshold? Indicating Pill picked
yes
Measure actuator position and vacuum voltage
Actuator fully extended?
yes

Figure 2: Arduino Programming Flowchart

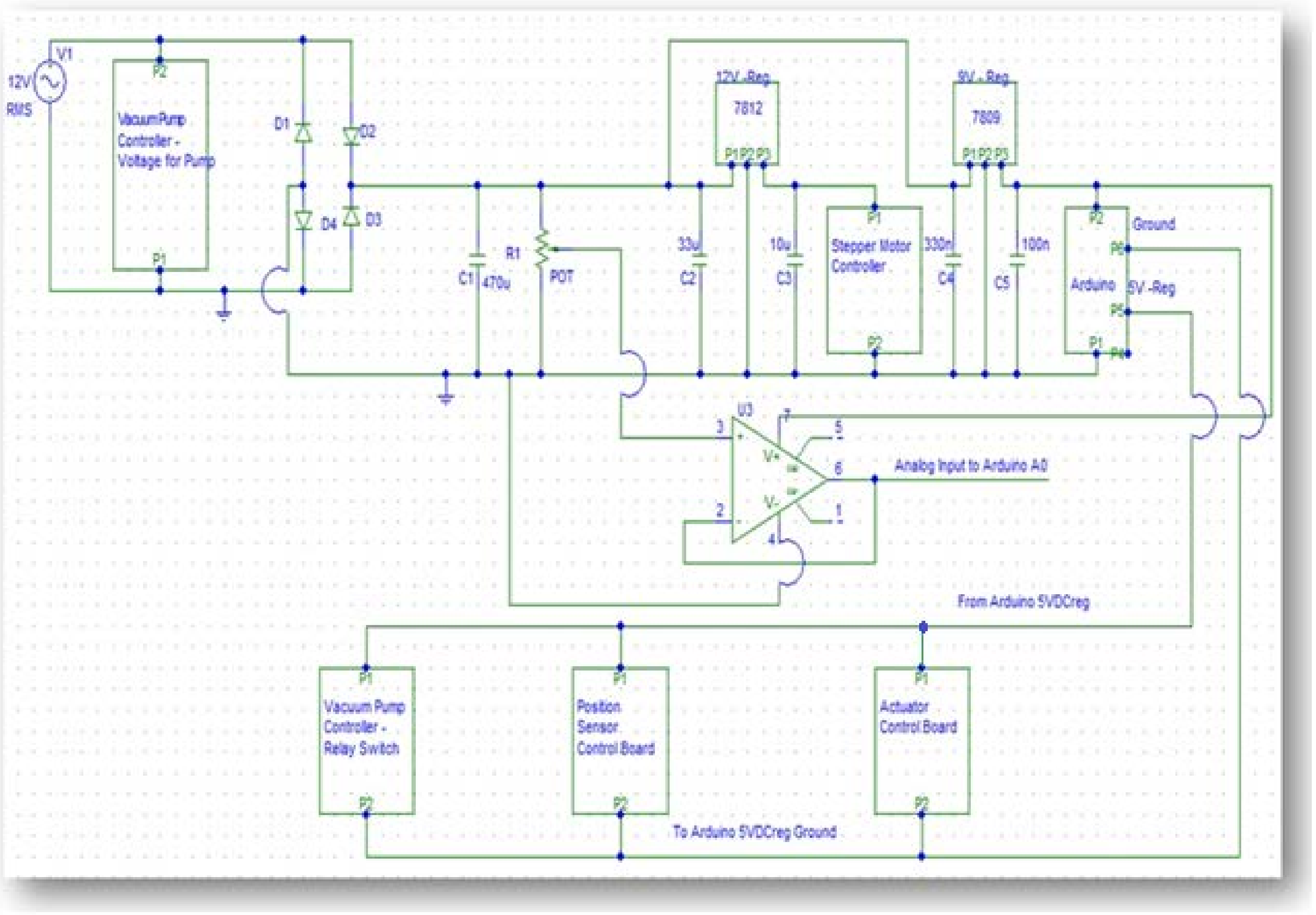
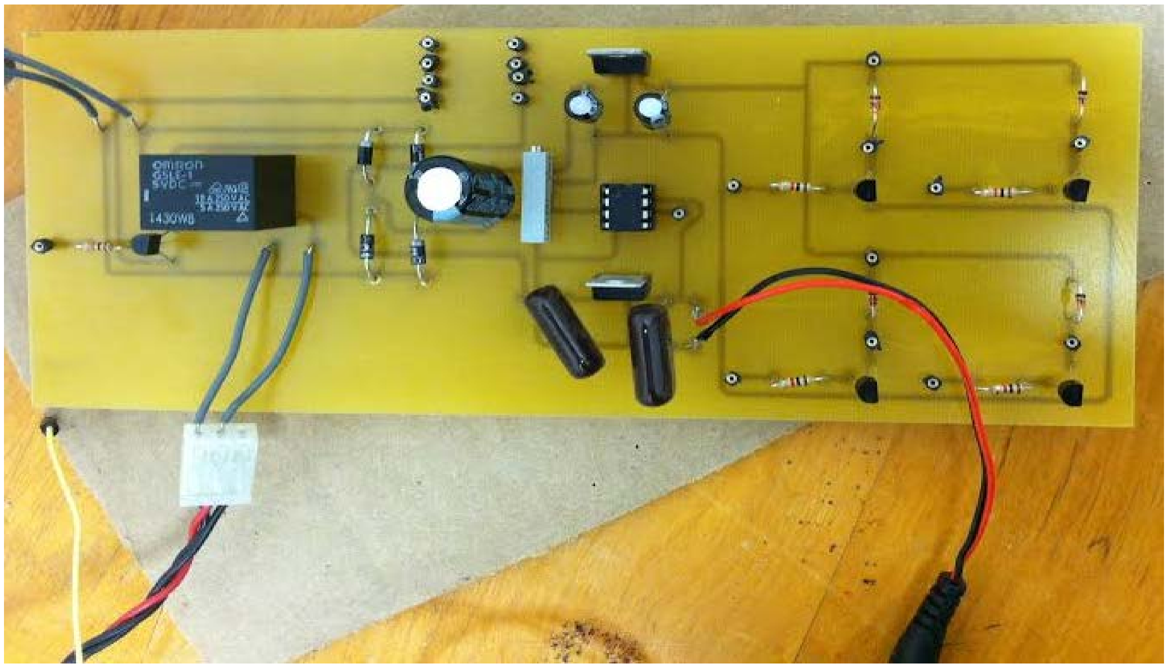


Figure 3: Overall System Block Diagram/ Schematic

Power Distribution Circuit

The prototype is powered through a wall outlet. An AC wall-adapter will step the 120 VAC down to 12V AC. This 12 VAC provides the power to the vacuum pump and to the AC to DC converter. After the AC to DC converter, the now DC voltage is connected to a high resistance potentiometer and voltage buffer. This component serves as the vacuum voltage sensor. Two voltage regulator rails are in parallel with the converted DC voltage, thus creating the 12 VDC and 9 VDC regulated rails. The 12 VDC rail solely provides power to the stepper motor. The 9 VDC rail solely provides power to the Arduino Uno board. All other devices (vacuum pump relay switch controller, position sensor, and linear actuator control board) are powered through the regulated 5 VDC provided by the Arduino board. The final printed circuit board (PCB) and overall power distribution circuits are shown below.



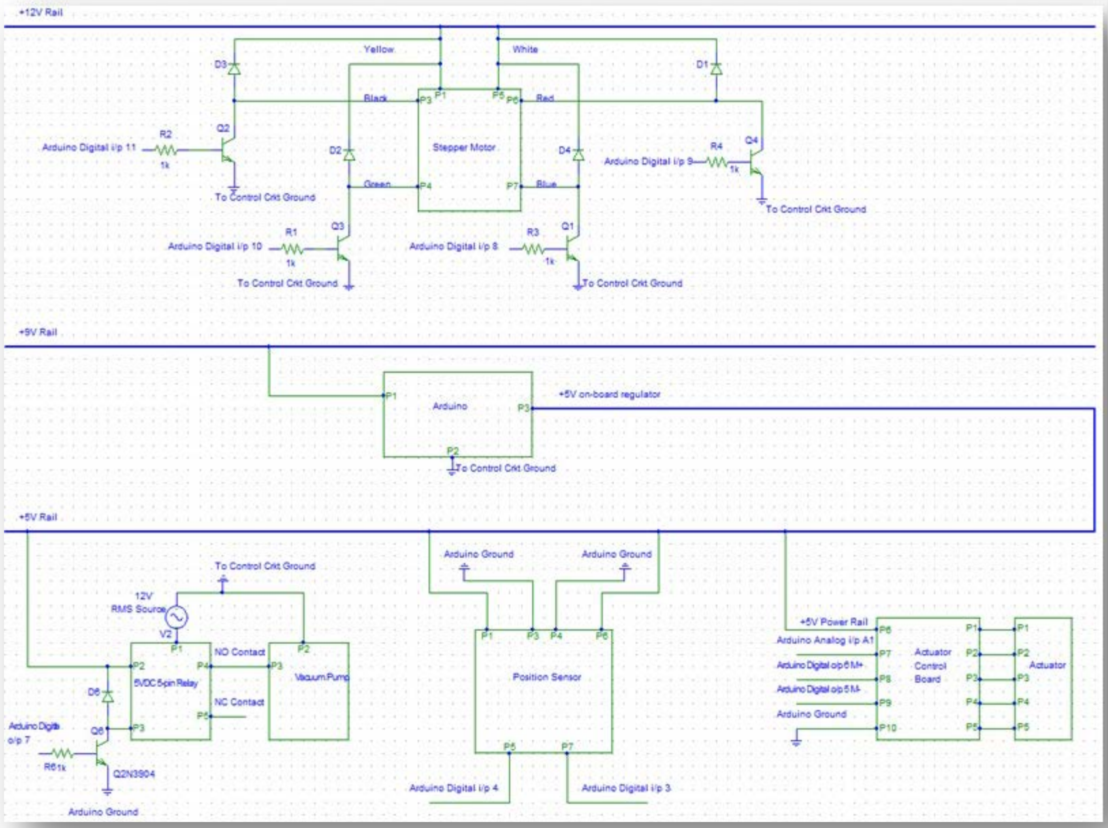


Figure 4: Power PCB (Top) & Power Distribution Circuit (bottom)

Stepper Motor & Position Sensor

The stepper motor is the first major component used in the pill dispensing sequence. The motor must precisely rotate the correct pill hopper to the desired spot above the actuator and vacuum pump. We chose a Jameco Reliapro motor for this task. We chose this particular motor because of a few separate reasons relating to the project. First, the motor is very precise, as the motor is capable of making 200 steps, equating to 1.8 degrees/step. Due to the precision and accuracy needed to move the hopper over a small opening, this made the motor extremely desirable. Finally, the motor can support the weight load of the pills and rotating disc combination.

The chosen Parallax position sensor is capable of sensing 144 positions. Since the motor is extremely accurate, the need for an extremely accurate sensor is diminished. The sensor is simply a feedback device to ensure the stepper motor moves to the correct position, and does not move without an input from the Arduino board.

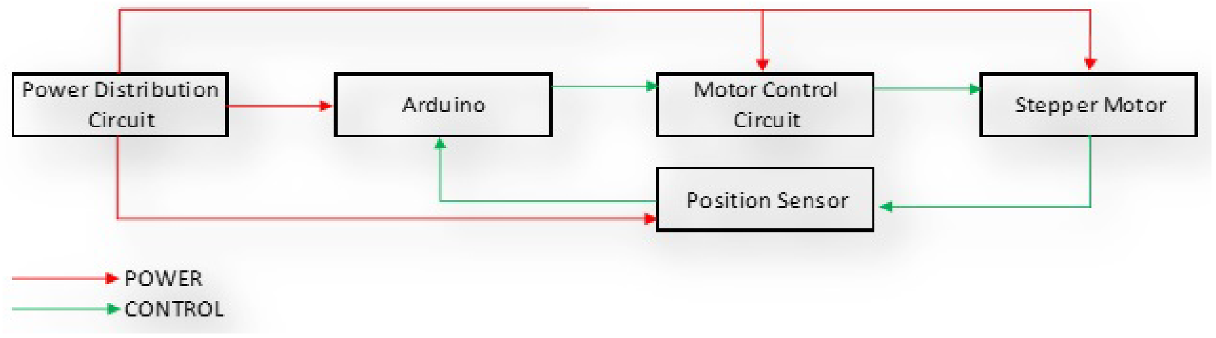


Figure 5: Block diagram of stepper motor subsystem

As shown in Figure 5, the position sensor works as a feedback for the stepper motor. Both the motor control circuit and sensor are connected to the Arduino board. The Arduino board sends control instructions to the stepper motor control circuit. While the stepper motor shaft is moving the position sensor then sends the corresponding voltage values to the Arduino board. In turn, determining the exact position of the stepper motor.

The stepper motor is of unipolar type and is controlled from the Arduino using a transistor switching circuit. A unipolar stepper motor has 6 leads, 2 for power and the other 4 are the motor coils. The power source is applied to the motor and then each coil is connected to its own transistor. When the transistor's gate is supplied sufficient current it completes the circuit allowing the coil to energize. Energizing the coil makes the driveshaft turn 1.8 degrees. In order to continuously turn the stepper motor the transistors must be turned on and off in a sequence.

The Arduino is used to operate the sequence such that only one transistor is triggered one at a time. In order to turn the stepper motor in the clockwise direction the first coil is turned on then the second and so on (1, 2, 3, 4). To turn the stepper motor in the counter clockwise direction the sequence is executed in reverse (4, 3, 2, 1). Repeating the sequence 25 times allows for one full rotation.

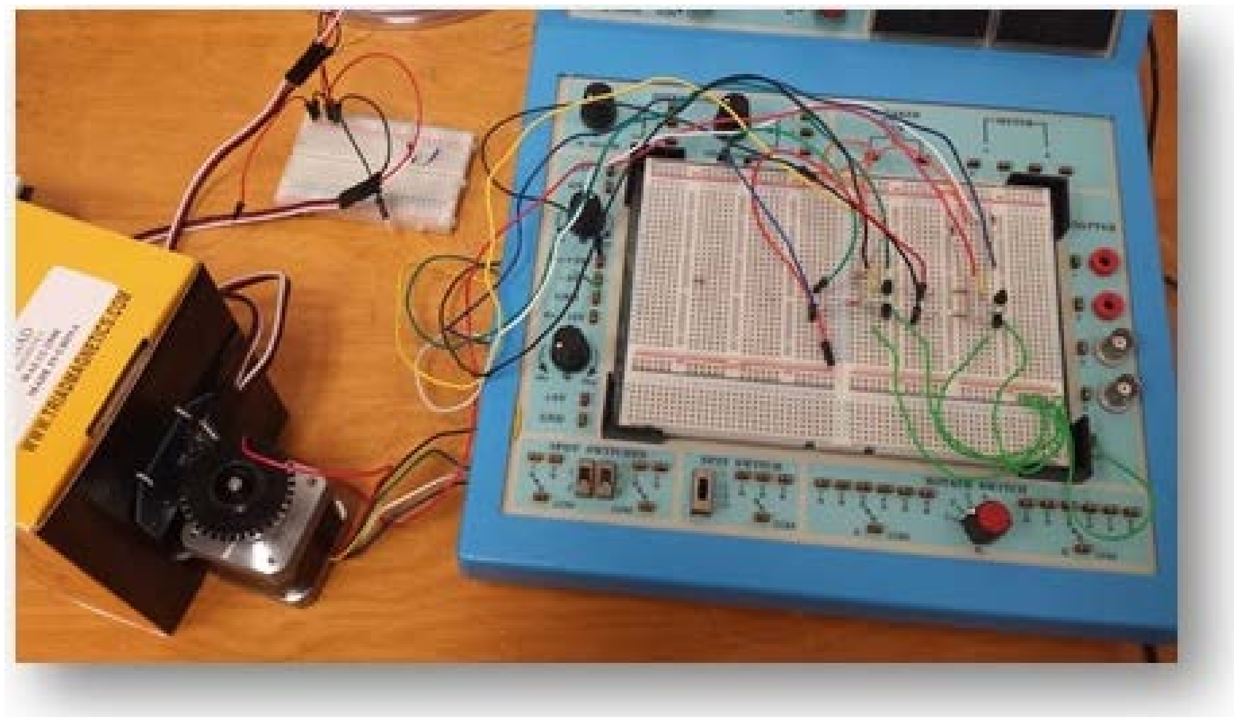


Figure 6: Stepper motor and position sensor circuits

The position sensor operates with the use of a 36 tooth gear mounted on the stepper motor's shaft. This position sensor has an entering sensor (CH1, yellow) and an exiting sensor (CH2, blue). The clockwise rotation of the motor would trigger the position sensor so that it would first detect the front edge of the gear tooth passing by the entering sensor (CH1, yellow) and then detected by the exiting sensor (CH2, blue). The counter-clockwise rotation of the motor triggered the position sensor in the opposite manner. The counter-clockwise motor rotation is represented by Figure 8. The exiting sensor would first see the edge of the gear tooth and then be detected by the entering sensor. This resulted in CH2 leading CH1. Using the sensor's outputs, the Arduino can easily keep track of the stepper motor's current position.

The position sensor was tested (the Figure 7 setup) using an oscilloscope to monitor the outputs (Figure 8). For the final prototype, the outputs of the position sensor will be read into the Arduino. After this is accomplished, the Arduino can use these readings for position control. Included below are pictures from the circuit design and testing of the motor and sensor. Look to Figure 4 for the stepper motor driver circuit schematic.

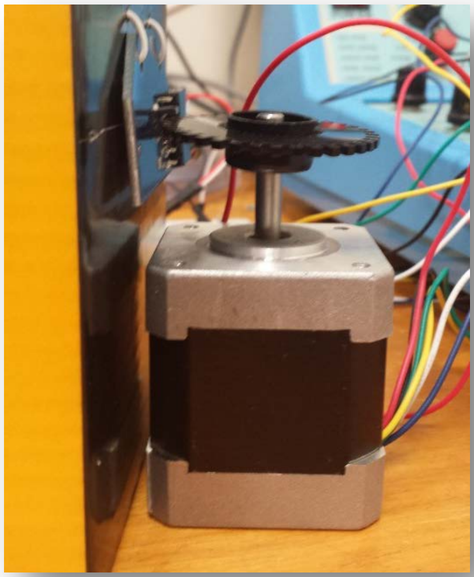


Figure 7: Test setup for position sensor

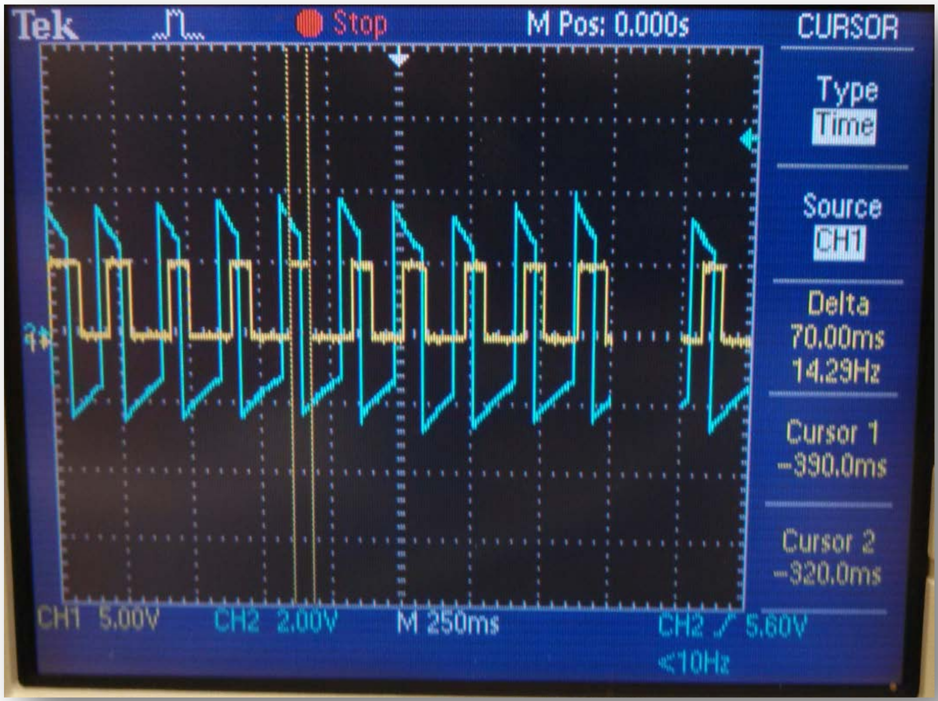


Figure 8: Oscilloscope Output signals of position sensor

Actuator

From the initial system requirements laid out by the group, the linear actuator in the device must only lift a pill approximately 4 inches. For these reasons, we chose a Firgelli L16 actuator. The length of the actuator is 100 mm (≈3.937 in.). The actuator also includes a built-in position sensor. By applying a positive voltage (+5V from the Arduino) and ground (Arduino Ground) to the “P+” and “P-“ terminals, respectively, the actuator acts like a potentiometer while extending and retracting. The analog voltage of the actuator at any given time during extension or retraction can be read from the “P” pin on the actuator control board. This is helpful so that the extension of the stroke can be manipulated if needed. However, at this time, the position will simply be monitored to control other sequence processes. The direction of the actuator (extending or retracting) is controlled via the h-bridge within the actuator control board. Thus, requiring only two digital inputs to the “M+” and “M-“control pins The entire block diagram is shown below.

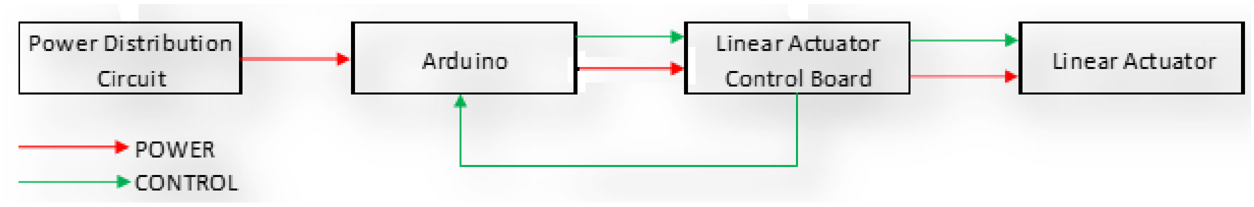


Figure 9: Linear actuator block diagram

Independently testing the actuator proved successful. The actuator is able to completely extend and retract given specific Arduino commands. Also, the group successfully monitored the position of the actuator through an Arduino analog input pin during the extension and retraction movements. Whenever the actuator is fully retracted, this gives an analog voltage of +5V (1023 analog value), and 0V (0 analog value) when the actuator is fully extended. The movement of the actuator will be used to control the vacuum in the final prototype. If the actuator is being extended, the vacuum will switch on. When the actuator is being retracted, the vacuum will switch off. Figure 10 depicts the system prior to testing.

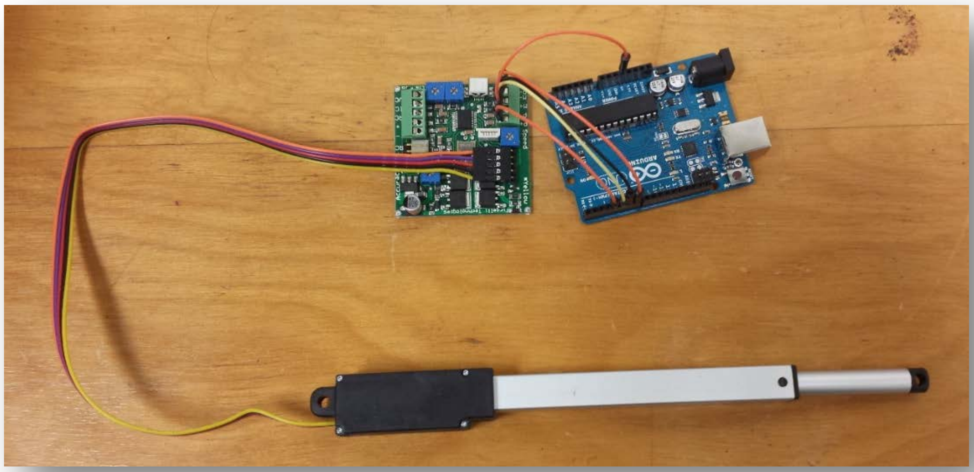


Figure 10: Linear actuator subsystem prior to testing

Vacuum Pump

The vacuum pump chosen only needed to generate enough suction to pick and hold a single pill. The vacuum pump is controlled through the Arduino board, and a vacuum controller. Referring to Figure 4, the vacuum controller consists of a 5VDC 5-pin Relay and a few components.

The +5VDC from the Arduino is connected to the positive energizing terminal of the relay, and the negative energizing terminal is connected to the collector terminal of an NPN transistor. The emitter NPN terminal is connected to the Arduino's common ground. The NC contact of the relay is left unconnected, while the NO (the switching) contact is connected to the positive lead of the vacuum pump, while the vacuum pump ground lead is connected to the overall AC voltage ground (refer to Figure 3). A diode is placed in parallel between the energizing terminals of the relay. This is to ensure minimal inductive kickback when the vacuum turns off, essentially protecting the relay and other components. The cathode is placed on the relay's positive terminal, and the anode is placed on the negative terminal. The 12VAC from the wall-adapter is connected to the control pin of the relay. A digital output from the Arduino is finally connected to a 1K-Ohm resistor, which is connected in series with the base terminal of the NPN transistor.

Once the linear actuator begins to move, a digital output from the Arduino energizes the above control circuit and the vacuum pump switches on. A voltage sensor connected in parallel to the vacuum pump monitors the voltage once it is switched on. When a pill is sucked up by the vacuum, the voltage spikes and the sensor will relay that information back to the Arduino. If the voltage sensor detects a sudden drop in voltage before reaching the full extension position of the actuator, the sequence will then reset. Thus, ensuring the patient receives their medication.

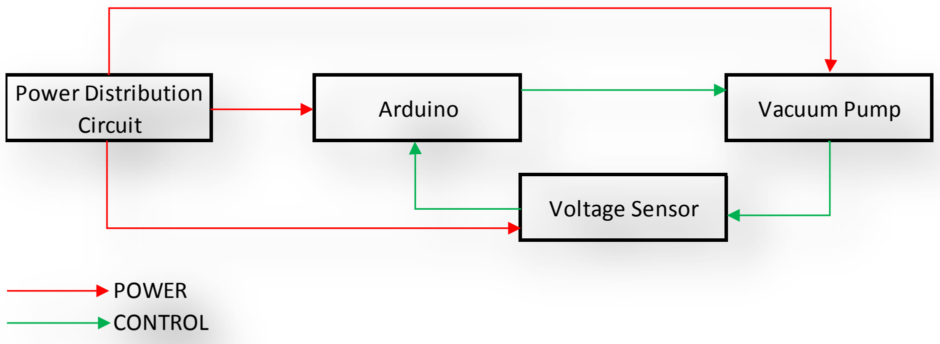


Figure 11: Vacuum pump block diagram

The vacuum pump controller and sensor successfully passed initial testing procedures. As previously stated, the actuator movement should control the vacuum for the final prototype. Below is a picture of the vacuum during testing.

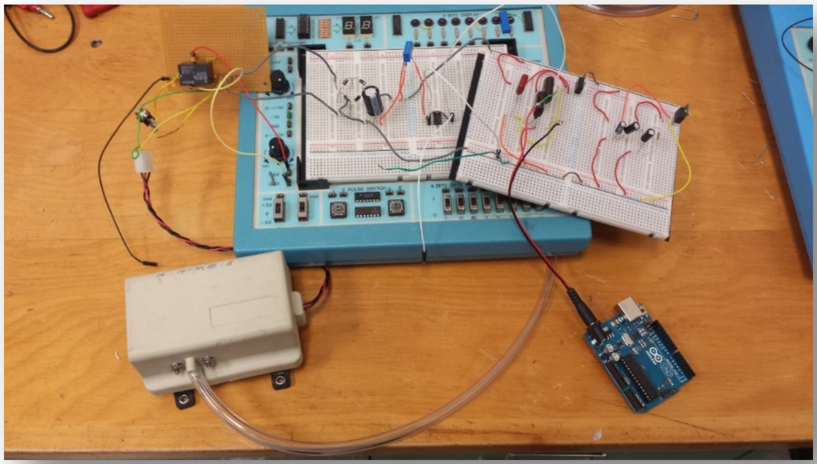


Figure 12: Vacuum system during testing

Arduino I/O

The Arduino Block in Figures 3 and 4 are explained in more detail in Figure 13. Figure 13 shows the interconnections between the Arduino Uno microcontroller and all other I/O components. The entire system implementation uses nine of the thirteen available Digital I/O ports, and two of the six available Analog I/O ports.

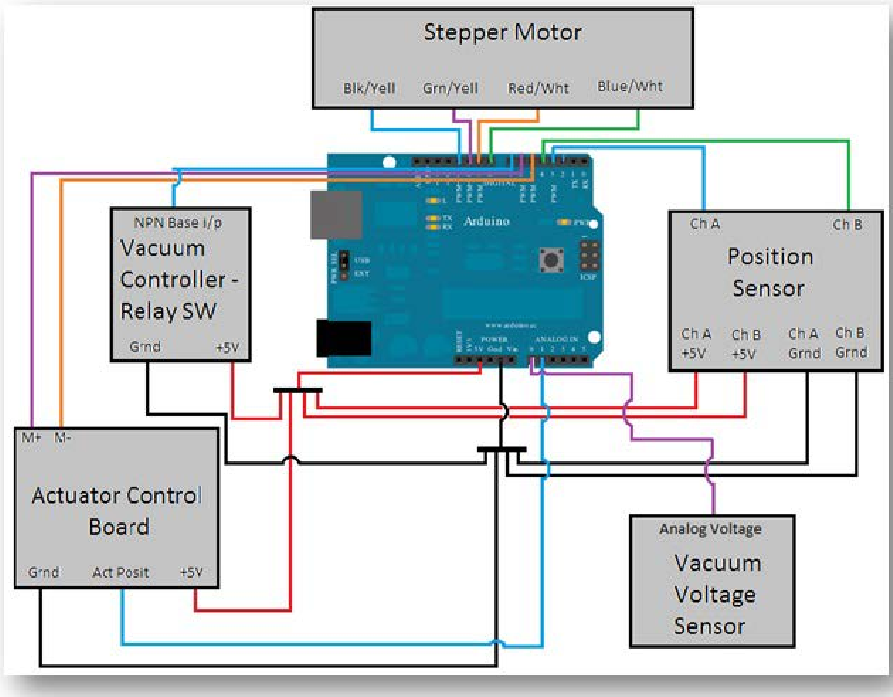


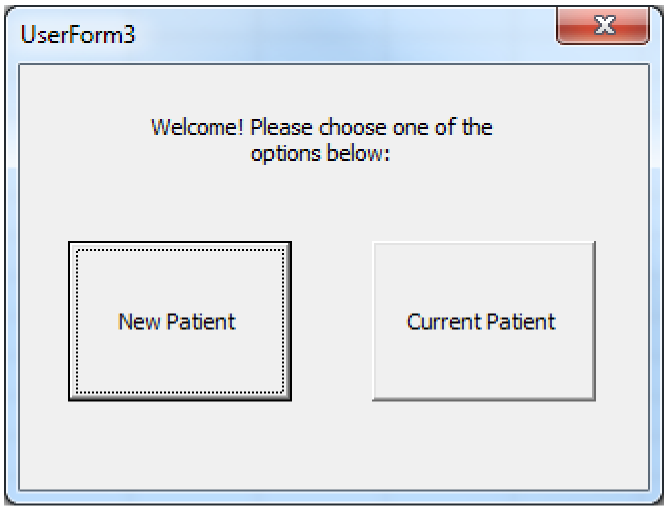
Figure 13: Arduino Connections to I/O components

User Interface

In order to ensure ease of use and compatibility, our user interface is made with the help of Microsoft Excel VBA. Microsoft Office products are common in most all work environments, thus, our interface software will be compatible with any computer that has Microsoft Office Excel. Once the caregiver makes the appropriate selections within the interface they will arrive to the prescription page for the patient's device. Here, the caregiver enters all medications with their associated doses into the two leftmost columns. Each prescription entry is directly correlated to one of the pill hoppers in the device. Next, the caregiver selects the dispensing time by "checking" the appropriate box for the desired time. Once all selections are made, the caregiver selects "Update Patient Prescription Information" and allows the software to finalize programming the device.

Once the caregiver clicks the aforementioned button, the software sends the information to the device's microcontroller (Arduino) via serial communication through a USB connection. The software sends data for each bin and the selected times for dispensing to the Arduino boards. This Arduino holds all of the bin data needed for the selections. The internal program on the Arduino periodically checks the current time with the selection times received from the user interface. Once there is a match, the bin number is sent to a second Arduino, which controls the pick sequence.

Figure 14 shows the initial display prompt when the pharmacist opens the program. Here, they choose from the two options.



Welcome! Please choose one of the options below:

Figure 14: Initial Display Prompt

Figure 15 represents the display prompt for a patient already in the system. Here, the pharmacist will enter the patient's ID number, as well as their personal ID number. Essentially ensuring only the correct personnel can update the patient's device settings. After correctly authenticating the personnel and patient, the program displays Figure 17.

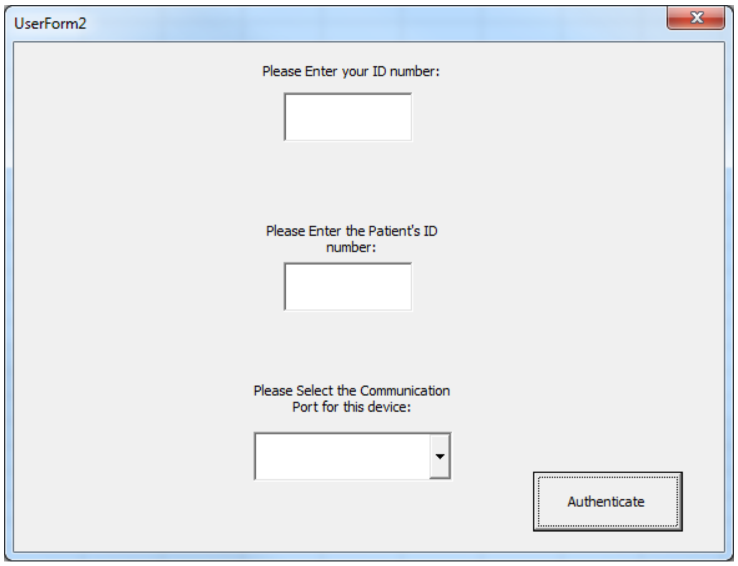

UserForm2
Please Enter your ID number:
Please Enter the Patient's ID number:
Please Select the Communication Port for this device:
Authenticate

Figure 15: Existing Patient Information

Figure 16 depicts the display prompt for entering a new patient into the system. Here, the pharmacist enters the new patient information as well as validating their identity for safety precautions. After clicking "Add New Patient", the program displays Figure 4.

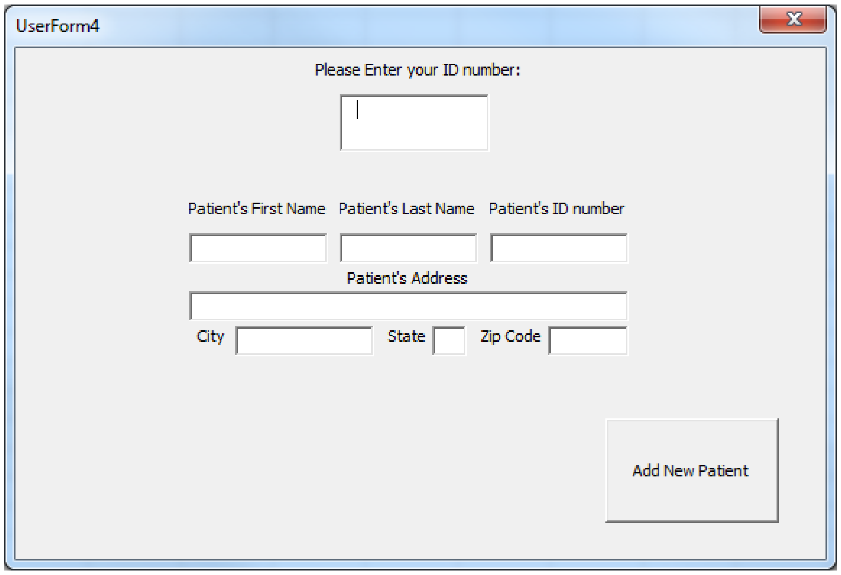
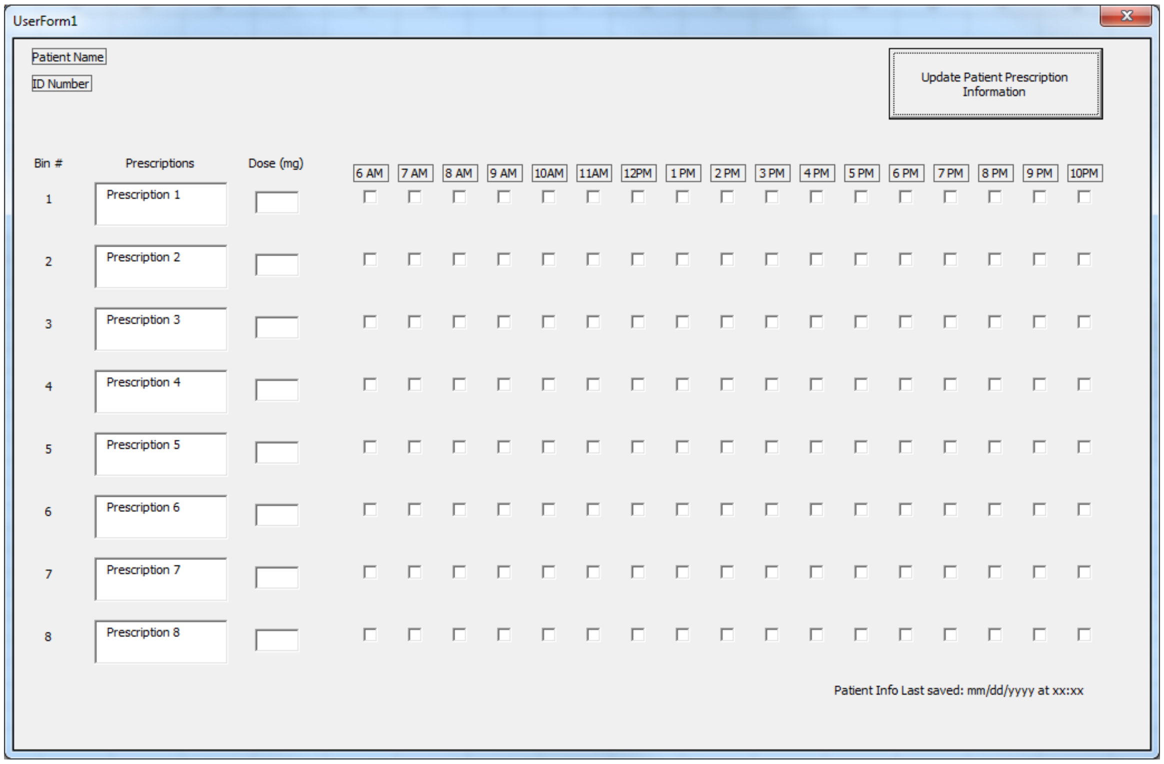


Figure 16: New Patient Information Prompt

Figure 17 shows the current patient prescription information. This window allows the caregiver to make all necessary selections in order to dispense a patient's prescription regiment.



Bin #	Prescriptions	Dose (mg)	6 AM	7 AM	8 AM	9 AM	10AM	11AM	12PM	1 PM	2 PM	3 PM	4 PM	5 PM	6 PM	7 PM	8 PM	9 PM	10PM
1	Prescription 1		☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
2	Prescription 2		☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
3	Prescription 3		☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
4	Prescription 4		☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
5	Prescription 5		☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
6	Prescription 6		☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
7	Prescription 7		☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
8	Prescription 8		☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐

Patient Info Last saved: mm/dd/yyyy at xx:xx

Figure 17: Patient Dosage Information

Currently, the user interface is still in the prototype stages and only the page layouts (Figures 14-17) are completed. Our group plans to integrate this with our final prototype for our final presentation in May. However, we do hope to have user interface design working for a functionality display at the conference in April.

Conclusion

The proposed solution meets all of the goals that were laid out prior to the start if this project. The group wanted to accomplish three main goals with the device. First, the device needed to be universally able to dispense any shape and size pill. With the staggering number of different medications on the market, there is not a uniform shape pill, which makes it difficult to make a simple pill dispenser. Secondly, the group wanted simple interactions with the pharmacist, as well as the patient. The interface for the pharmacist was created with simplicity in mind, but was also created to be effective in dispensing a pill. On the patient side, the interaction with the device could not be simpler. The patient simply needs to pick up the medication when the alert sounds. The final goal that the group had was to make the product cheaper than the current market solutions. Through research, it was found that similar devices tend to cost somewhere in the range of $800-$1000. The group was given an initial budget of $500, but the prototype itself is expected to cost approximately $400. If the device were to be mass produced, the price would be cheaper than that as well.