SDP Technical Report

Delta Robotics (Group 3)

Abstract

Pillip is a pill dispensing and delivery system designed to be used either in a care home or as a household appliance. The system is comprised of a pill dispenser and delivery robot and is interfaced through a web application. This document outlines the technical specification of the system including design decisions and further improvements.

I. Introduction

A. Problem

We have looked into medication and how it is dispensed and distributed and have found that:

- 70% of care home patients have not received the right medication dosage during their stay [3]
- 15% of adults take a potentially harmful medication combination[1]
- 5 of the top dietary supplements contribute to 50% of consumption[2]

B. Original Objective

The original objective was to design a prescription delivery system. The system would be composed of two parts. A dispenser that would dispense an appropriate prescription and a delivery robot which would then deliver it to the patient. The system could be used in 2 different use cases:

- Home use case - the dispenser would be placed in the users' home and it could be used to dispense their daily prescriptions
- Care home use case - the dispenser would be used in a care home to integrate with the delivery robot that could then deliver prescriptions to the appropriate door.

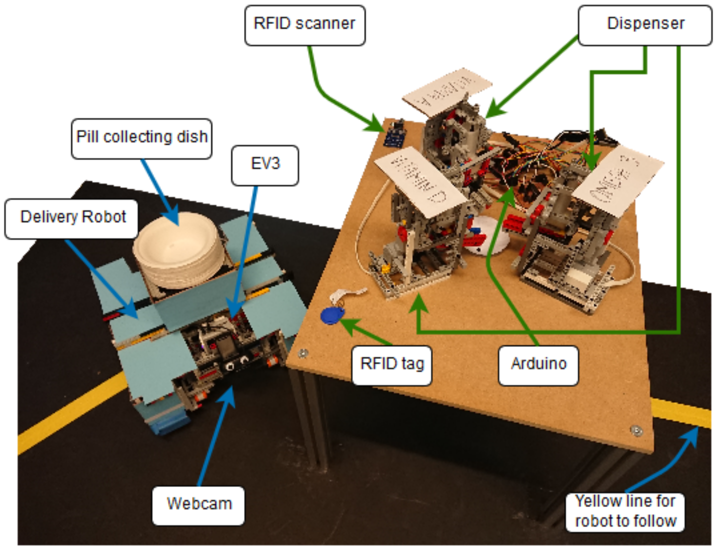

Figure 1: Pillip: Prescription Delivery System

II. Delivery

A. Background

The delivery robot was used to transport the pills once they had been dispensed. It worked together with the dispenser and the web application as part of the care home use case. The robot contained a Raspberry Pi working with an EV3. The EV3 was used to control the motors and everything else was handled by the Raspberry Pi. The two communicated over Wifi through a TCP connection created using the *Python Sockets module* [9]. Whenever the Pi established that it was time to start, stop or turn it sent a signal to the EV3 and this in turn controlled the motors.

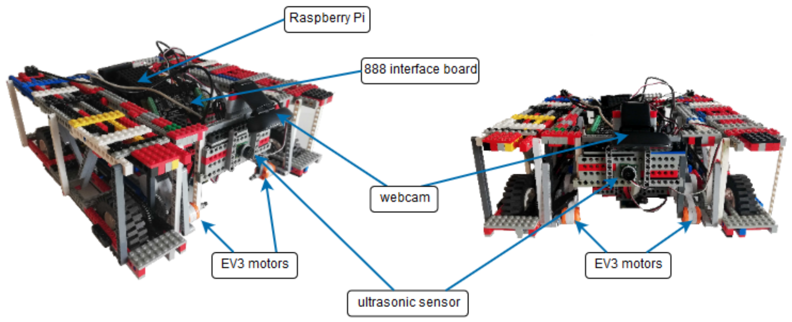

Figure 2: Front and Side views of the Delivery Robot

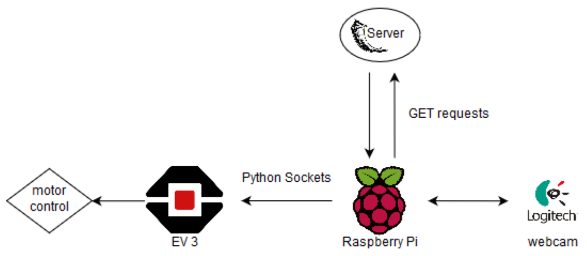

Figure 3: Communications between the various components within the Delivery Robot

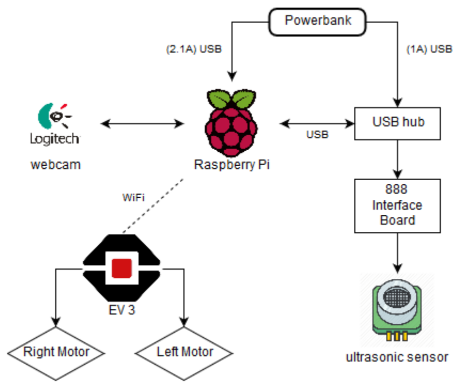

Figure 4: Component connection diagram for Delivery Robot

The Raspberry Pi handled the vision part of the operation. This was a yellow line on the floor which the robot would follow. The dispenser was marked by blue tape with every door defined by green tape. If the blue/green was detected at the right time, the robot would stop. The path had the following layout:

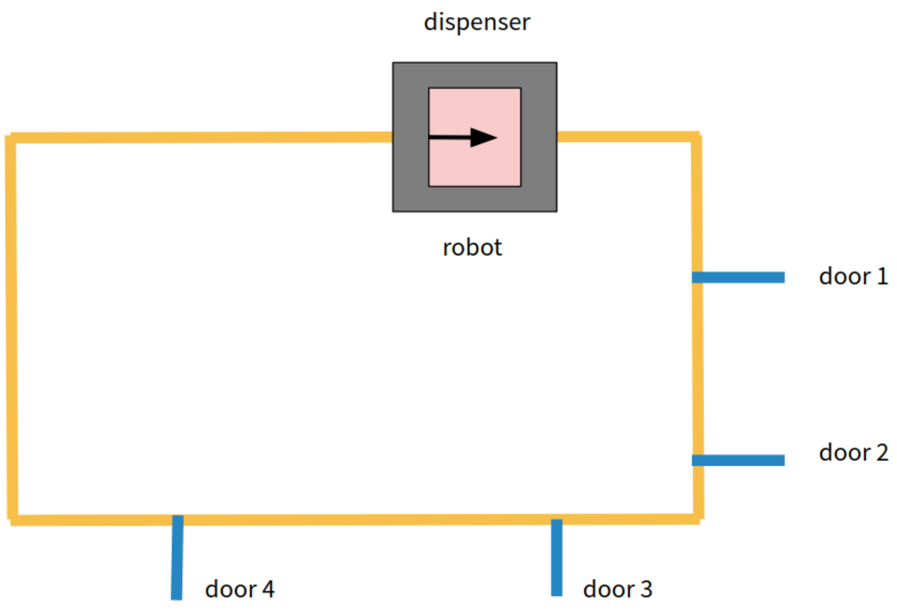

Figure 5: A simple circuit that the delivery robot would follow. The doors and dispenser are marked by tape.

B. Methods

The implementation of this design was carried out in stages. Firstly, the initial project proposal was drafted with the following features specified:

- A webcam which was to be used to allow the robot to follow lines to the doors of care home residents.
- An ultrasound sensor to be used for detecting and avoiding obstacles blocking the robot's path.
- A robust movement system enabling the robot to accurately and efficiently reach its destination
- A secure catch/release system to protect the payload of pills from being picked up by the wrong person. Pills would be packaged into a box at the dispenser. The box would then get released onto the robot. The robot would have a similar system in order to drop the box in front of the patient.
- A QR code scanner to allow authentication of care home residents.

These initial goals were each implemented with various rates of success. Many of them had to be changed from the first draft due to the reasons explained below.

1) Line following: The vision for the line following was put in place using a Logitech Webcam mounted on the front of the robot. This worked in conjunction with Python's OpenCV [7] module to effectively decide which way the robot should move. Each frame from the camera was split into two regions. Each of the regions were then analysed and the two largest yellow regions were selected (these are outlined in light green in Figure 6). OpenCV was then used to establish the centre of these regions and a line was drawn between these centroids.

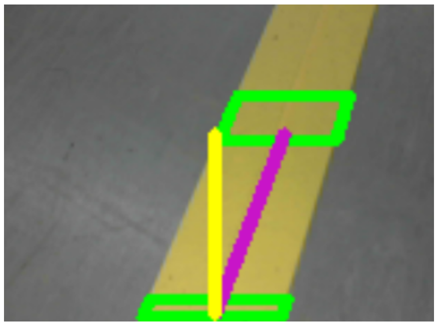
Figure 6: Frame from camera feed

This line between the centroids is pink in the camera feed shown in Figure 6, and represents the direction of motion the robot should aim for. The yellow line is simply a vertical line, representing the current direction of motion of the robot. In an ideal case, these two lines would overlap, as the robot would be perfectly following the correct direction. In reality, we allow the robot's direction to be within a threshold angle from the desired direction. So if the angle from the yellow line to the pink line is above a threshold angle, then the motors would be adjusted accordingly to rotate the robot until the angle fell back under the threshold angle. This is explained in more detail in the following pseudocode:

```
if angle > 15 and upper box (x coord) > lower box (x coord):
    send signal 'RIGHT' to EV3
else if angle > 15 and upper box (x coord) < lower box (x coord):
    send signal 'LEFT' to EV3
else if angle < 15:
    send signal 'FORWARD' to EV3
else if boxes dont exist:
    send signal 'STOP' to EV3
    # The robot is lost
```

2) Object Detection: The object detection was implemented using the ultrasound sensor as initially planned. The sensor returned a value which corresponded to the distance of the closest object in front of it. This value was constantly polled and if it fell below a certain constant then the motors were stopped.

3) Gearing: Next, the robot was built with the drive system geared-down using a 6/48 gear ratio to allow greater torque with less speed leading to greater control. This was built alongside a rigid and stable body to ensure the robot did not struggle to deal with the weight of the payload. Finally, tracks were attached to the wheels to create a 4x2 drive system for further control.

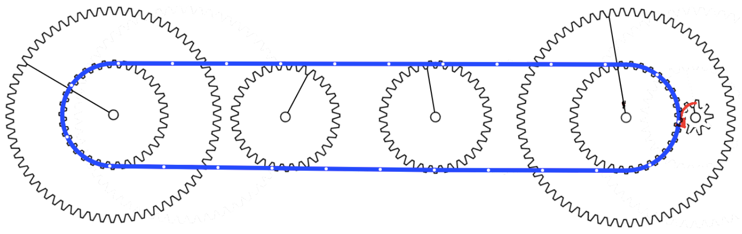
Figure 7: gear system : the smallest gear starts to rotate (caused by the motor), which leads the medium-size gear to rotate (ratio is 6/48, the diagram is not drawn to scale). As medium-size gear rotates, the belt rotates and further leads other medium-size gears to rotate. Simultaneously, the largest gears (the wheel of the robot) rotate as the medium-size gears rotate.

4) Catch release: The catch release system was not implemented. There were various reasons for this. First of all, the line following took far longer than expected to put in place. While it was easy to get a system that worked *fairly* well it took significantly longer than our initial estimates to produce something that would run to a sufficiently high standard. In addition to this, there were issues with the QR code scanning as detailed below and it was decided that attention should be focused on this area as opposed to a complex catch/release system. Instead of a catch release system it was assumed that the patients will pick the medication on their own from the top of the robot.

5) QR codes: The front mounted camera was also used to add the functionality for QR code scanning to the robot. When the robot reached the correct door, it waited for a QR code to enter the frame. When one was detected, it was decoded and checked to see if it matched the unique code given to the expected recipient. This allows the system to confirm that the correct resident had received the correct medication.

C. *Results*

After reaching this point, there were several well defined, testable points which were decided upon:

- How accurate was the robot when taking a pill from the dispenser to a door?
- How long did the robot take to deliver the pills to a door?
- What was the average speed of the robot and is this scalable to a care home environment?
- What sonar threshold should be used for effective object detection?
- How accurate was the QR code scanner?

Quantitative analysis of the above points was carried out and the following results established:

1) Accuracy of delivery: 10 tests of 3 pills to door 1 resulted in a 60% accuracy of delivery of the pills.

2) Time for delivery: The same test resulted in successful deliveries having a mean delivery time of 45 seconds to door 1.

3) Speed of Robot: The robot was set to travel around a closed circuit of 4.16m. It did this 10 times, finishing with an average of 82.3s. This led to the robot having an average speed of 5.05cm/s.

4) Object Detection Accuracy: The robot was set to travel 1.5m in a straight line, with three different objects at the end of the line: a wall, a box, and a scarf. For each object, the sonar threshold was varied from 25cm to 150cm, and the robot's object detection and subsequent avoidance was observed (three trials for each of the 10 thresholds). For the box and wall cases, a sonar thresholds between 50cm-150cm had 100% object detection success, however for the scarf case, 75cm was the minimum threshold that had 100% object detection and avoidance success. As a result we set 75cm as our sonar threshold.

5) Accuracy of the QR Code Scanner: The robot was placed at the correct door and a QR code was scanned from a phone. This was done 10 times and was 100% accurate.

D. *Discussion*

The delivery accuracy of the robot being only 60% is fairly low. In a care home scenario, it would not be acceptable and significant time/resources would have to go into improving this statistic. There were various reasons for this being so low. Firstly, the door was not always detected by the webcam. This could have been improved by making the tape marker larger or make it a more vibrant colour. Next, the webcam occasionally had latency issues. This was very difficult to fix was it was unclear as to why this was. However, the use of a faster network as well as perhaps something with more computing power than a Raspberry Pi/EV3 combination would have helped here. As well as this, the gearing system on the robot was poorly designed but would have taken a great deal of time to rebuild. It sometimes slipped and caused the robot to fail to reach its destination. In hindsight, much more time would have been devoted to building a robust and reliable gearing system instead of continually patching up the one we had. The circuit which was tested was significantly smaller than one which would be put into place in a care home. It was not a true representation of the environment and thus the speeds and times calculated above cannot be taken to be tested in the real world. They do, however, give a good estimate of what these values may be. One point which was pleasing was the accuracy of the QR codes. This can be attributed to how reliable QR codes are as a means of data transmission.

Overall, the delivery robot proved to be successful. This is not to say it was without challenges. The time taken to implement the vision and line following was severely underestimated. This led to the hardware team not being able to complete all of the initial aims as defined above.

In addition to this, the integration with the rest of the system was extremely tricky. As the delivery robot added another two systems which needed to be on the same network to communicate, this caused many problems when trying to get everything to work together from a central server (on a different network). In hindsight, a different method of communication would be used between the Raspberry Pi and the EV3 such as serial to ensure they could work in any location on any network.

III. Dispenser

A. *Introduction*

The dispenser was used to allow the users to take out their medication when they needed it. It was also integrated with the delivery robot to allow for delivery of medication in the care home.

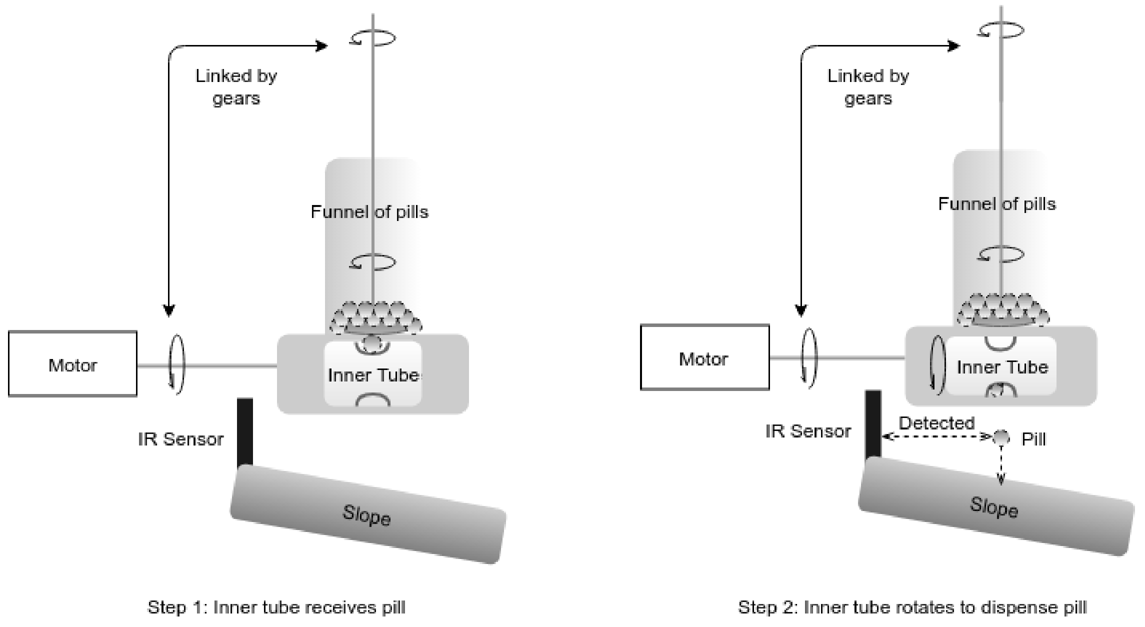

Figure 8: Dispenser cross-section

B. Background

We wanted to create a pill dispenser with the following characteristics:

- Accurate in dispensing only 1 pill at a time. We wanted to ensure that the dispenser will be 100% accurate in the number of pills it dispenses.
- Dispenses any custom combination of pills. Pills can be dispensed as prescriptions with any number of each type of pill included
- Scalable to any number of different pills. The dispenser should be modular and allow for extra dispenser units to be purchased separately.

C. Method

1) Dispenser 3D Design: Many prototypes were constructed along the way. Those made out of Lego did not allow enough precision to dispense 1 pill at a time. In order to ensure that the above requirements were met a 3D printed custom part was designed order to fulfill the above requirements. Spherical objects were chosen to represent the pills in order to simplify the problem due to their uniform nature. The dispenser was designed specifically to allow one pill dispensing at a time with characteristics such as speed being less important. *Fusion 360* [5] was used for the design.

2) Dispenser Design: Dispenser was designed as demonstrated on Figure 7. 3D printed tube would rotate inside another tube. The inner tube had a hole in it big enough to fit a single pill. The tube would rotate inside and the pill would fall through the bottom of the outer tube. EV3 motors were used to spin the inner tube. A slope was used underneath the dispenser to collect pills dropping. IR sensors were used at the end of the slope in order to ensure that only one pill at a time would drop from the dispenser. The IR sensors were used due to a small enough range to be able to detect the pills dropping but at the same time not to pick up interference from the surroundings. An alternative for pill counting was considered, a pressure sensor, but was rejected due to their small precision and installation difficulty. The motion of the motors was also translated upwards at a 90 degree angle to allow for a mixer inside the funnel of pills. This improvement was introduced as a result of the pills getting stuck in the funnel. Alternatives such as vibration motors were explored but eventually abandoned due to their unreliable nature.

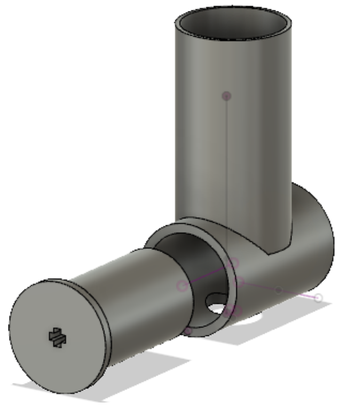
Figure 9: 3D design

3) Dispenser state machine: A state machine was used on the dispenser side to ensure that the system was only performing one task at a time. The state machine script is running on a machine to manage all of the dispenser activity.

Home use case

```
state = WAIT_FOR_READY
if state is WAIT_FOR_READY:
    wait for a "ready" from the Arduino Xino
    state = WAIT_FOR_RFID
elif state is WAIT_FOR_RFID:
    wait for an RFID tag to be scanned
    save the UID of tag
    state = GET_PRESCRIPTION
elif state is GET_PRESCRIPTION:
    get prescription for UID from server
    state = SEND_TO_ARDUINO
elif state is SEND_TO_ARDUINO:
    send prescription to Arduino Xino over radio
    state = WAIT_FOR_COMPLETE
elif state is WAIT_FOR_COMPLETE:
    wait for a complete from Arduino Xino
    state = COMPLETE
elif state is COMPLETE:
    state = WAIT_FOR_READY
```

Care home use case

```
state = WAIT_FOR_READY
if state is WAIT_FOR_READY:
    wait for "ready" from the Arduino Xino
elif state is GET_SERVER_DATA:
    wait for a UID from a scanned RFID tag
elif state is GET_PRESCRIPTION:
    get the next scheduled delivery from server and check if status is PENDING
    state = SEND_TO_ARDUINO
elif state is SEND_TO_ARDUINO:
    send prescription to Arduino Xino over radio
    state = WAIT_FOR_COMPLETE
elif state is WAIT_FOR_COMPLETE:
    wait for complete from Arduino Xino
    state = SEND_COMPLETE
elif state is SEND_COMPLETE:
    make request to server to notify that status changed to DISPENSED
```

4) Arduino Xino RF: An Arduino was used as the controller for the motors. There were 3 motors connected to the Arduino as well as 3 IR sensors. Arduino was used as it provided a low level access to the motor speed and the sensor readings. Arduino Xino specifically was used as it allows communication through radio. This means that the script can be running on a remote machine and the dispenser does not have to be connected to it directly. After the dispensing was completed the Arduino Xino communicated with the state machine to alert it about a task being done.

The general flow of the motor controller is as follows:

```
while True:
    send "ready" to state machine
    wait for data incoming through radio
    parse the incoming string in the format "first, second, third" to extract the numbers of pills
    while pills1 < first:
        keep spinning the motor for dispenser 1
        keep checking the sensor for dispenser 1 to see if pill detected
        sleep

    ...
    repeat for pills2 < second and pills3 < third
    ...
    send "complete" to state machine
```

5) Arduino UNO: Another Arduino was added as the pin outs on the motor shield were not enough. This Arduino is used to read the UID of a RFID tag. Arduino UNO was used for this task as its pin outs are very well documented and it is easy to connect it to the RFID tag. The RFID tag also required access to all of the Arduino pins which was not possible with the Arduino Xino due to its motor shield. *A RFID tag RC522* [8] was used in this setup.

The flow of the Arduino UNO is as follows:

- Wait for an RFID tag to be scanned
- Extract the UID of the tag
- Send UID of the tag to state machine

6) Communications: There are several connections in this dispenser setup:

- Arduino Xino - Machine: radio connection. Machine uses RF stick that transmits on a pre-agreed frequency with the Arduino Xino.
- Arduino UNO - Machine: serial connection. Arduino UNO is connected via USB cable to the machine and transmits over serial.
- Machine - Server: machine communicates with the server through making GET requests.

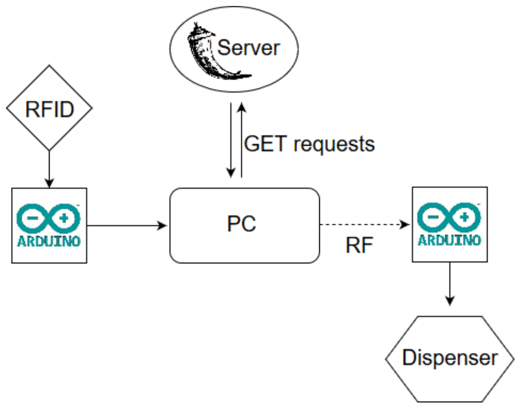

Figure 10: Home use case communications

D. Results

1) Quantitative Analysis: It was agreed that the dispenser was going to be built to allow for:

- Accurate dispensing, ensuring that the prescriptions dispensed will always be correct was the top priority.
- Speed of dispensing, ensuring that the prescriptions will be distributed timely
- Minimize human intervention, ensuring that the dispenser setup requires as little human intervention as possible.

Across the course quantitative testing was conducted to ensure that the above criteria are being met. The following results have been achieved:

- Accuracy - 99% dispensing accuracy on 100 pill trials
- Time taken - 184.86s for 12 sets of 8 pills, 15.4 per set = 1.93s per pill
- Human intervention - required 1 out of 100 times

2) Weak Points: Even though the system worked properly a lot of the time, potential failure points were identified:

- The axle didn't sit flush in the 3D printed part and therefore could slip out
- The motor speed of the dispenser was not enough or too much for the amount of pills in the dispenser. When the dispenser was empty very often motors would spin too fast whereas when the dispenser was full the motors didnt spin fast enough.
- The IR sensor was pointing at the Lego walls of the funnel and therefore detecting a pill when it wasn't there resulting in a miscalculated pill amount.

In retrospect we would address the above issues in the following way:

- Create a better suited 3D design addressing this issue. The changed would include increasing the size of the axle or increasing the precision of the 3D print.
- Use other motors rather than EV3 motors. Ensure a constant power supply to the motor shield. Implement a motor power scaling that will adjust the speed of the motors to the number of pills in the dispenser.
- Create a more sturdy structure without using the Lego for the dispenser. Potentially fully 3D print the entirety of the dispensing unit.

E. Discussion

We would have changed or added the following things if we could attempt this project again:

- Use a Raspberry Pi instead of a laptop. This would allow to reduce the number of setup steps the user would have to follow.
- Use a different board to the Arduino, one that would better suit our needs to allow all of the devices to be connected to the same board. This would reduce the amount of hardware required
- Use radio communications for all of the system.

IV. Application

A. Introduction

A web application was used to create a user interface to control the system when deployed in a care home. It provided access to the database that stores information about the system's users, allowing for input through a series of forms. A derivative application was created to compliment the dispenser when being used as a standalone system in the home with reduced functionality.

B. Background

The Python web framework Flask [4] was used to implement the web application. Flask was chosen as there are many libraries that assist with much of the functionality required such as form handling and database management. Using well established libraries not only speeds up development but ensures important security features are not overlooked. Bootstrap was used to style the application providing a more appealing user interface.

Using the python package *Jinja* [6] allows for templates to be defined which can then be rendered with different information from the backend database. This makes it very simple to create pages that will change in content based on the current user for example.

The backend consists of a SQLite database; communication to it is handled by *SQLAlchemy* [10]. This powerful package allows for querying across multiple tables very quickly which is vital in making the app feel responsive for users.

WTForms [11] allows for HTML forms to be created and handled very easily along with custom input validation. Data can then easily be entered into the backend database.

C. Methods

1) Care Home: In the context of the care home it was decided that two types of accounts would be used: doctor accounts and patient accounts. This would allow for tight control over what data could be seen and modified by each user group. New patients and doctors can be registered by existing doctor accounts. Doctors can then create prescriptions for patients linked to them. Patients can only see information stored about themselves whereas doctors can see information for any patient registered to them.

Each patient is assigned a room number and one or several delivery times. A patient is then assigned a combination of pills to be delivered at each time slot.

Doctors are responsible for populating the database with the contents of each dispenser. The quantity and expiration date of each type of pill is tracked allowing the application to warn if intervention is required.

Pill deliveries were orchestrated through the web server using an API that would expose the next delivery to the dispenser and delivery robot. The web server would also listen for changes of state from either part of the system to allow indirect communication between both devices. This is discussed in greater depth in the next section.

Initially deliveries were generated once daily and the API would iterate through them. A time slot based scheduler was decided upon during development of the project. This allowed for users to have multiple, distinct prescription deliveries throughout the day, making the system a lot more practical. Time intervals of 10 minutes each would dictate the current data on the end point.

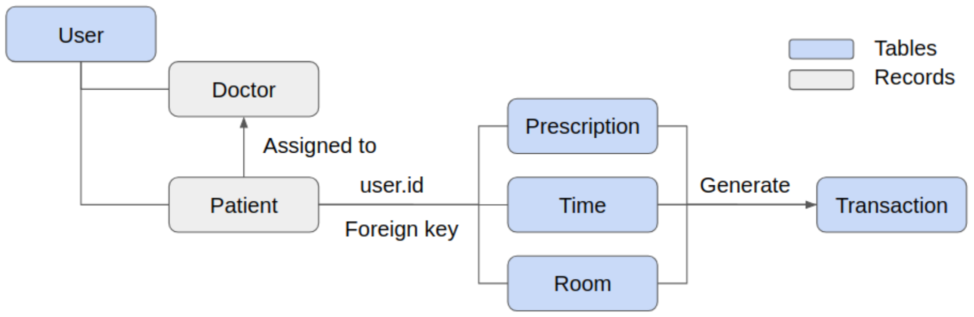

Figure 11: Database Structure

In the database, there is a table called *User* which has all information about the doctors and patients. It includes email addresses and hashed passwords used in login system as well as the identity ('p' for patient, 'd' for doctor) and doctor assignments of patients. With user.id as the foreign key, table *Prescription*, *Time* and *Room* were created as the attributes of patients. These separate tables facilitate the manipulation of the database and allow more flexibility for adding new data. After the patient was registered and the prescription was added, the server would generate a list of new transactions in the *Transaction* table for each timeslot of every patient. By means of the *Transaction* table, the database records the current day's transactions to be completed as well as a full history of medication received by each patient for analysis in the future.

2) Home: To operate in conjunction with the dispenser, a stripped down version of the care home web app is used. In this system, users manage their own prescriptions and can dispense them using unique RFID tags. This system was designed to be simple to use due to the non technical target market.

Dispenser contents are managed the same as above with the exception that any user has access to do so.

D. Testing

1) Usability Testing: This was done on random participants around Appleton Tower to test the care home use case and ensure mainly that the application was easy to use, especially for people with a non-technical background. Several questions that could be rated from 1-5 (with 1 being the lowest score, 5 being the highest) were asked to the participant after going through the web application with them. An optional field for open-ended comments was available for the participants to voice any concerns about the web app. The questions that were asked as well as the results were as follows:

- How intuitive is it to navigate the web app?
  - 5: 80% of responses
  - 4: 20% of responses
- Do button descriptions match the function?
  - 5: 80% of responses
  - 4: 10% of responses
  - 3: 10% of responses
- Are the forms understandable to fill in?
  - 5: 100% of responses

2) Application Testing: The web app was tested using Locust to measure how the app would do under an abnormal day's load in a care home, especially with multiple concurrent users. The app was run from a local server and tested with 500 simulated users at 5 users generated per second. The script tested different users concurrently logging into doctor's accounts and making different /send and /receive requests to the server. The results from the tests are as follows:

- Success rate of requests: 100%
- Max response time: 4519 ms

3) Results: From the Usability Testing, the software team implemented the popular comment of adding auto-complete to as many fields of the forms as possible to aid filling them in. Most of the feedback was positive with regards to the ease of use and the participants had no trouble using the web app. Meanwhile, the load testing assured the team that the web app was coded robust enough to handle more than the usual stress of a care home use.

E. Discussion

The following points were brought to attention during development of the application. Below outlines how they were addressed along with some ideas for further improvements the system:

1) Security: The applications for the care home and home were both initially built on HTTP protocol. For the sake of security, they were both switched from HTTP to HTTPS. In addition a preshared key is used to authenticate at the two endpoints to avoid exposing data.

2) Big Data: Through the *Transaction* table, the database stores a full history of deliveries. In the future, this data could be analysed to perhaps optimise delivery schedules or predict medication usage to help with stock management.

3) User Interface: Considering the physical condition of patients and their non-technical nature, the user interface could be made more user friendly such as using a larger font and buttons and include more guidance. Attempts to alleviate user struggle include the use of selection fields in forms where possible.

4) Limitations of RFID tags: A drawback of using RFID tags is that each user can only dispense their pills once per day. This may present problems for users who must take different medication throughout the day. They could however use multiple RFID tags tied to different prescriptions to work around this. The system could also dictate what is dispensed based on the time the user scans their tag. This would allow users to have distinct combinations of pills at different times of the day but adds complexity to the system and may result in users being unable to dispense vital medication if they miss the allotted time.

V. Communication between systems/Integration

A. Integration

Once satisfied with the performance of each subsystem, they were brought together using a Star integration methodology. This was deemed suitable as there were only three pieces of the system. Initially the dispenser and web app for home were brought together. This proved to be relatively easy as the protocol for communication had been agreed upon before development. The delivery robot was then introduced to the system. The main challenge was creating a way for each subsystem to communicate which is outlined below.

B. System statuses

In order to integrate all the system components a status was used for each of the care home deliveries. The status was kept as a field in *Transaction* table. The following statuses were available:

- PENDING - delivery has not started yet, the delivery robot is moving underneath the dispenser
- UNDER - delivery robot is underneath the dispenser ready for the pills
- DISPENSED - the dispenser has completed dispensing
- DELIVERED - the delivery robot has delivered the prescription to the correct door

C. API

Coordination of the full system is handled by the API hosted on the web server. There are two endpoints:

- **/send:** serves information on the current prescription to be dispensed and delivered
- **/receive:** listens for state changes and updates this information in backend

Both endpoints are secured using a preshared key to authenticate requests are from legitimate hosts. This ensures all data shared will not be exposed and the systems integrity is maintained.

The payload received when querying the **/send** endpoint will be dependent on the scheduler mentioned in the web application section. Every payload will be of the form:

```
{"dispenser1/2/3": x, "room_number": y, "status": "PENDING/UNDER/DISPENSED/DELIVERED", transaction_id: z}
```

The dispenser attributes denote the quantity of pills to be dispensed from each of the three dispensers. *room_number* is simply the number the deliver to and *transaction_id* allows the devices to keep track of the current delivery and update its status on the /receive endpoint. *status* is used to keep track of the current system state and allows the various systems to coordinate their actions as mentioned above.

The **/receive** endpoint expects a request with a payload of the following structure:

```
{"key": "secretkey", "status": "UNDER/DISPENSED/DELIVERED", "transaction_id": x}
```

The relevant transaction in the database will then be queried and updated with the new status. By querying the API, each subsystem will know the current state of the whole system. When not performing an action, a device will continually query the /send endpoint until a change of state is seen. For example the dispenser will continually wait for a transition from "PENDING" to "UNDER" triggered by the delivery robot. At this point it knows to dispense the necessary pills and once done will inform the system by sending "DISPENSED" to the /receive endpoint.

VI. Reflection

Throughout the project several challenges were encountered that highlighted flaws with the initial plans. From these challenges important lessons were learned that will be extremely useful when attempting new projects. Some examples of these are listed below:

- We severely underestimated the amount of time required to perfect the vision system for the delivery robot. If the project was to be undertaken again it would definitely be advisable to devote more time to the hardware design of the robot.
- Assigning subteams to members was largely based on personal preference. Whilst this ensures that members will be satisfied with the work they have to undertake it may not have been the best way to ensure each member's expertise are used to the fullest. It may be favourable to not use a subteam system and instead allow members to work on any area they feel capable of undertaking.
- In the initial project plan, many of the goals were scheduled to be completed in rapid succession, neglecting the fact that previous goals might not necessarily be achieved on time. It is clear now that time should be allocated for unforeseen interruptions in the schedule.

Overall the group worked well together by communicating regularly and making key decisions on design and goals together.

VII. Main achievements

During the project several milestones completed were a lot more challenging compared to the rest of the project. The main achievements included:

- 3D design: Initially there was a lot of doubt whether we will ever be able to produce a design capable of consistently distributing only 1 pill at a time. utilizing CAD to design a custom piece which managed to achieve almost perfect accuracy was a big achievement.
- Vision: Utilizing vision for the line following was a challenge due to the use of the Logitech webcam - but after a lot of experimentation with different colours of tape and thresholding using HSV values, and with the addition of the EV3 in week 10 to have greater control over the motors, we managed to get the robot following the lines more accurately and smoothly.
- System Communication: Using the web servers API, we managed to achieve a way for the whole system to coordinate with low latency and high accuracy.

References

[1] *Dangerous Drug combinations*. URL: https://consumer.healthday.com/general-health-information-16/prescription-drug-news-551/1-in-6-seniors-takes-dangerous-combos-of-meds-supplements-study-709168.html.

[2] *Dietary supplements*. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1255934/.

[3] *Dispensing errors in care homes*. URL: https://www.nhs.uk/news/older-people/drug-errors-in-care-homes/.

[4] *Flask Documentation*. URL: http://flask.pocoo.org/docs/1.0/.

[5] *Fusion 360 Documentation*. URL: https://www.autodesk.co.uk/products/fusion-360/students-teachers-educators.

[6] *Jinja Documentation*. URL: http://jinja.pocoo.org/docs/2.10/.

[7] *OpenCV Documentation*. URL: https://opencv.org/.

[8] *RFID RC522 Documentation*. URL: https://randomnerdtutorials.com/security-access-using-mfrc522-rfid-reader-with-arduino/.

[9] *Sockets Documentation*. URL: https://docs.python.org/3/library/socket.html.

[10] *SQLAlchemy Documentation*. URL: https://docs.sqlalchemy.org/en/latest/.

[11] *WTForms Documentation*. URL: https://wtforms.readthedocs.io/en/stable/.