MTRX3760 - Project 2 A Tally Tale

Report of Project 2

Group ID: **Super Tally**
Created By:

SID	Date
460386784	08/11/2019
470318966	08/11/2019
470023233	08/11/2019
470416309	08/11/2019

Introduction

Within modern retail stores and supermarkets, it is an overall priority for the company to increase revenue and efficiency of the store while still upholding a positive consumer experience. Companies therefore, are looking to specifically employ new robotic systems from self checkout to inventory robots with the overall priority to reduce workloads, increase stock accuracy, eliminate human errors and enhance the stores marketing opportunities. This report will highlight the critical aspects of the tally bot project.

Overview

This project report aims to model the employment of an autonomous inventory robot. The turtlebot will be programmed to demonstrate the expected operations of an inventory robot. More specifically, this would involve mapping of the shelves, identification of the product on a shelf by scanning via a camera, the ability to send and receive information between the robot and PC regarding the inventory, have an efficient movement through the supermarket and have a user-friendly GUI.

The physical demonstration of the turtlebot will involve a model of a range of shelves typically found in a supermarket (at camera level). The turtlebot will then follow the aisles and scan the products. The scan must return a variety of information concerning the product and its location in the supermarket as a whole.

The turtlebot being modified will be a *Turtlebot 3 Burger platform* with a camera attached. Furthermore, it is a restriction that no significant hardware can be introduced. The movement of the turtlebot must follow the aisles and be able to maneuver through them without collision or any major complications causing it to stop for extended periods of time or causing damage to the environment and itself.

Specifications and Requirements

The project has a variety of restrictions and requirements of the turtlebot and the system design in general. These include:

- The store must have at least five different kinds of items.
- The autonomous movement of the turtlebot.
- The modelled environment is to be a small scale of a supermarket space.
- The turtlebot must be able to identify and keep a record of the items on each shelf in the spaces.
- Save the inventory to disk in a human-readable (text) format.
- Provide a mechanism for answering user input.
    - At least two kinds of questions must be answered/accounted for.
- Ability to handle the addition of new kinds of items to the store shelves as well as handle any new shelves being introduced into the store.

Advanced Requirements(optional)

The project has a range of more complex outcomes that will result in a more independent system that will operate with greater data processing and more efficiently. The optional requirements for such a system include:

- The detection and output of changes to the inventory between scans.
    - Printed to the disk in a human-readable (text) format.
- Build a map of the store using RViz or similar simulation program.
- Use the store map to more efficiently conduct the inventorying operations.
    - May require the run of the turtlebot twice (discussed in the *System Design*).
- Detect the changes to the store layout and report on the changes to the disk file.
- Ability to find an item of a certain type in the store on request.
    - Report where the item can be found therefore, answering a third question that is *'where can a certain type of item be found?'* when requested as a user input.
        - Provide the readable identification of the shelf (not the QR code).
- Ability to suggest reorganisation of the store for similar items to be better grouped.
- Create an automatic "docking" (stop) charge station when the inventory scan is complete.
    - I.e. have a set end position for the turtlebot to stop at when the full store scan is completed.
- Work with another team to collaboratively map a larger store.
- Present a GUI on the host PC that is user friendly.

Assumptions

There are a variety of assumptions to be made for the basis of the project to allow for the design of the system to be consistent among a range of shelf arrangements and stores and be a realistic model. These assumptions include:

- Having a constant docking position in the entire model of the system.
- Using realistic double sided shelves, however, only using a single layer in terms of vertical axis (further denoted in the figure below).
- Design all the shelves to be uniform and equal in appearance to each other.
    - Shelf has a single spot to contain an item and each spot is uniform in size and shape (square slot for the product to sit on).
    - The shelf spots can be represented as an array, therefore information can be easily assigned into a vector data structure.
    - The camera is to be mounted on the left side of the turtlebot, therefore, using a left wall follower program will allow for efficient movement through the aisles.
- Upon starting the left wall follower will initialize the movement of the turtlebot and aim to locate the first shelf.
- The initial plan is to incorporate JSON files for the inventory, and the GUI initializes the JSON file to answer the questions that are inputted by the user.
    - The JSON file is on the host PC.

- Use a HASH table for the data structure.
    - Initial tests have been made with vectors.

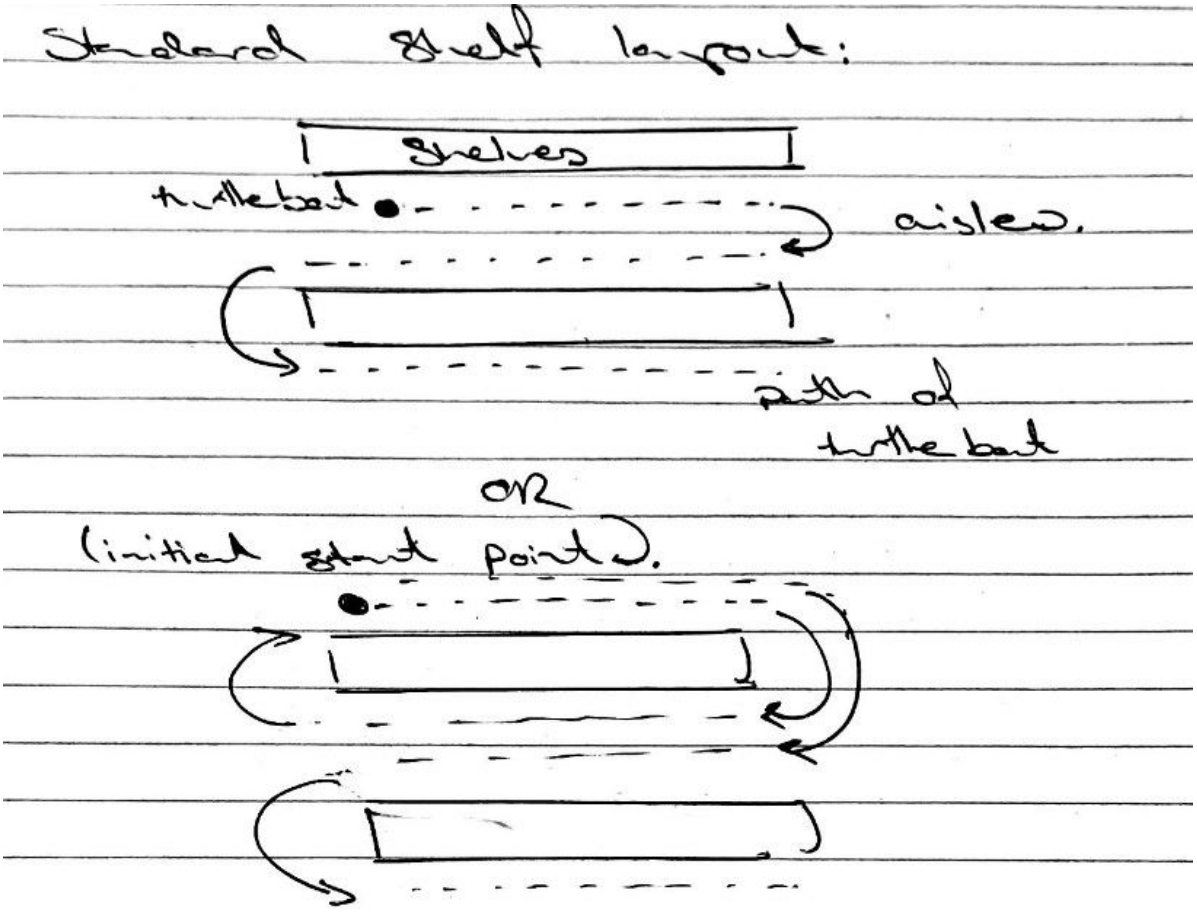


Figure 1: Model of the Standard Shelf

System Qualities

The following section highlights the three main priorities in the design of the system that potential investors will be looking into:

Maintainability

- Simple design with minimal hardware.
- Program is planned to be reusable.

Extensibility

- Handles the addition of new items and shelves.
- Ability to utilize alternative power sources.

Expenses

- Low cost components.
- Low maintenance.
- Can utilize cheaper power sources.

Table 1: Cost Breakdown

Component	Quantity	Cost (AUD)
Turtle Bot (Platform 3)	1	1213.09
Camera (Raspberry Pi)	1	24.83
Software - Ubuntu/Windows - Command Line Interface - Notepad - Compiler - ROS	1	519.00
PC - Hard Drive (HDD) - RAM	1	<1000
Battery (Rechargeable)	2	(included in the Turtle Bot)
Tools - Zip Ties - Screw Driver	3	5.00
Monitor	1	100
Keyboard	1	25
Mouse	1	(include with above Keyboard)
Router (with Internet Connection)	1	60-100
Total Approximation:	-	*2596.92-2636.92*

System Design

Implementation Method

The project will be accomplished using ROS packages, libraries and open-source code correlating to the system design.
The core functionalities to be coded include:

- Scanning items on shelves
- Saving the inventory to disk.
- Top-level decision making.
    - Movement
    - Sending Information
    - Detection
- Navigation through the supermarket.
- Handling of the changes to the store layout.
- Detect the item types.

Initial Design Plan

All information presented in bold is specifically highlighting the planned design approach of the advanced components of the requirements of this project.

1. Creation of the store within simulation using ROS and RViz (or similar 3D simulation program).
    a. Create the placement of the shelves within the simulation.
    b. Define a starting and docking position.
        i. Most likely the same position to allow for consistency with the design of the project.

2. Allowing for the initial mapping of the store.
    a. Create an arrangement of shelves that would be found in a realistic supermarket.
    b. Have a method to identify each of the shelves (RFID) and have the tag on the shelf in the physical modelled setup.
    c. Utilize a participant to control the movement of the turtlebot upon initial mapping stage.
        i. To retain a complete autonomous system, the left-wall follower program can be incorporated to allow for the mapping to not rely on participants.
    d. A physical map of the store must be created.
        i. Therefore, a mapping run maybe utilized by the turtlebot using the wall follower program.
        ii. The following run of the turtlebot would be the taking of inventory.

1. **This would be much more efficient as the map has been created for the turtlebot to utilize position goals with.**

3. Creation of the robot to read items placed on the shelves.
    a. The turtlebot runs through the mapped layout autonomously.
        i. The turtlebot has saved *xy* position functions that will store its mapped positions and coordinates into a data structure.
            1. Will require programming to save the data into a structure.
            2. Use the data structure with the saved positions and employ position goals.
        ii. Reemployment of the same wall follower for the scanning of the items will allow for the previous *step (2)* to be completed alongside the detection of the shelves.
            1. Only one run of the turtlebot around the store is required.
            2. Use a data structure to assign the ID of the shelf and the ID of the item being scanned.
    b. Detect the shelves by scanning using the camera.
        i. Use the QR code at this point to save the shelf ID.
            1. Potentially when the QR code scan is triggered, the saved position of this event occurring can be utilized as a position goal.
    c. Create movement from point-to-point where an edge of a shelf is a point.
        i. The point of the shelf edge is the starting point of the scan.
        ii. Using goal seeking for each of the key coordinates can allow for a more flexible and accurate scan of the whole market layout.
    d. The camera detects objects on shelves.
        i. In situations where there is a shelf on each side of the robot, two runs are taken:
            1. A single sweep of one shelf (left-hand-side mounted camera),
            2. Then a 180 degree turn at the end of the aisle,
            3. Another sweep that records the other shelf which is now on the left-hand-side of the turtlebot.

Note: If the left wall follower is utilized as the single main movement system of the turtlebot, then the detection of all shelves becomes dependent on the shelf layout and the starting point. Therefore, the requirements of the project can be better satisfied by utilizing goal seeking with the *xy* position function within the turtlebot.

4. Allow for the detection of the items on the shelf.
    a. Identify the current shelf being scanned by the camera.
        i. Use the QR code scanning to identify the shelf being scanned.
        ii. Can create a vector data structure where each shelf has its own structure that further has items and properties saved within that structure.
    b. Identify the item that is being scanned.
        i. Implement a QR code detection system.

1. Unique QR code for each item but has a similar pre-ID that allows for the differentiation of what codes are shelf codes and item codes.

ii. Possible to implement shape detection however would require more programming to specifically identify the product.

c. Send the information to a data structure.

d. Within the advanced system design, the turtlebot must have a designated ending (stop) position or coordinate that the turtlebot retires to upon completion of taking inventory.

5. Creation of the data structure.
    a. Use a vector to hold:
        i. Position & Coordinates (*xy*)
        ii. Shelf ID
            1. Item ID
    b. Have a program that operates with the GUI to allow for the vector to create a counter of each item.
    c. Within the advanced system design, each inventory that is taken must be printed to the disk and therefore, changes between inventory should be explicitly printed to the readable file.

6. Creation of the user GUI.
    a. Must be able to provide information for the following user inputs:
        i. How many of an item type there are in the store?
        ii. What item is present on a particular shelf?
        iii. For the advanced system design, the program must be able to answer *'where a certain item type can be found in the store'*.
    b. Furthermore, the advanced system design must utilize a friendly user interface GUI that is more than the terminal.
        i. Terminal would not be utilized by users of the system, only participants in realistic deployment.
    c. For this section there would be three *cpp* files with each asking for the questions above as well as the **advanced system design question.**

Advanced System Design:

7. Work with another team to collaboratively map a larger store.
    a. Two turtlebots operating on a larger supermarket model.
    b. Having the ability to have a method to divide the store into sections.
        i. Account for the possibility for the wall follower to overlap into a section that the other turtlebot is assigned to and vice-versa.
    c. Have the output from the program to be combined either on two PCs for each section of the store or have one PC host two turtlebots.
        i. Gather all the information onto a single PC or have two PCs with correlating information being outputted to their respectable disks.
            1. Can upload the information onto an external drive like a cloud system, GitHub, etc.

Note:

- It would be realistic to have similar items on the same shelf and next to each other.
- These two user inputs must be answered or similar two questions (at the least).
    - Similar question could be *'on which shelf can a particular item be found?'*.
- Can utilize another language to create the GUI.

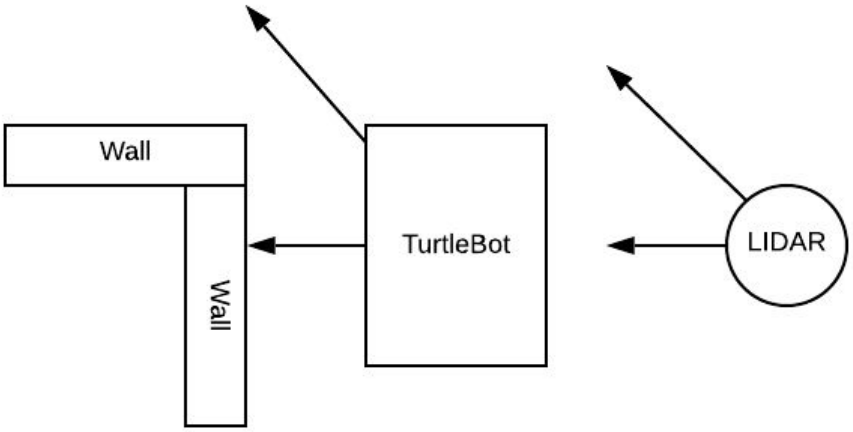


Figure 2: Displaying Detection of TurtleBot

Design Assumptions

The following section highlights all assumptions that were made in agreement for the project to have less open ended solution. Therefore, all members will be able to work towards a similar goal that is not shrouded with any conflicting or differing ideas on certain aspects mentioned:

- The questions that can be asked by the user are restricted to three, therefore the user will input selections rather than inputting a question.
    - Command line GUI design.
- There are at least five different types of items that will be detected by the turtlebot in the store.
    - New items can be added and new shelves can be added with the system being able to properly account for such events.
- The shelves are designed so that each spot" on the shelf has a single QR code representing an item.
- The store layout will be as denoted in ***figure 3.***
    - This was decided in the final system design (please see the *final system design* heading below).

Design Concepts

The project utilized the following design concepts:

Abstraction:

Employed in the majority of the programming of the image processing function to increase efficiency and readability.

Encapsulation:

Planned to be utilized in the implementation of the image processing, inventory and partially employed in the mapping. Furthermore, this is designed to increase the abstraction of functions and sort private and public data.

Inheritance:

Inheritance is aimed to take place in the classes of inventory creation when linking a program or the user interface function with the processing of collected camera data (QR scanning).

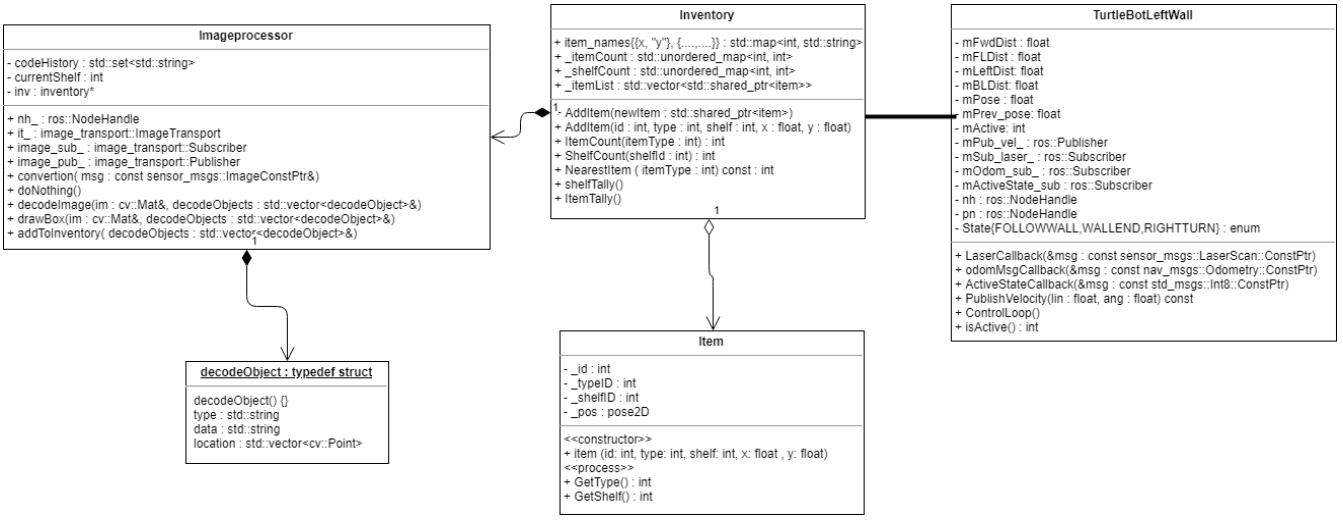


Figure 3 (a): UML Class Diagram

Polymorphism:

Polymorphism occurs within the project is appropriate for the design of the inventory function that may use aisles with a high priority in classes include various lower hierarchical classes such as the shelves and items on the shelves.

Design Patterns:

- *Iterators*

  Iterators are utilized within the inventory design of the turtlebot after image processing has taken place. Furthermore, the iterators will be employed in the design of the user interface program when new items and shelves will be processed by the system.

- *Singletons*

  Reduce the class instances initialized within the image processing, inventory and possible mapping functions. It is worth noting that mapping is incomplete, therefore, for future development singletons would be ideal in the implementation of the mapping function.

Standard Library Containers:

- *Algorithms*

  Various algorithms are employed in the creation of the user interface and the inventory system that will allow for the calculations of vector positions and counting of total items.

Final System Design/Level of Performance Achieved

This section highlights the achieved system design, the functionalities, the employment of the concepts and how they were implemented. The **final functionalities** of the turtlebot are:

- Autonomous maneuvering in a retail setting.
- Ability to record a variable number of items.
- Disk storage of inventory data.
- Friendly User interface for inventory data on host PC
- Handle the addition of new items and new shelves.
- Detect and print changes to inventory between scans

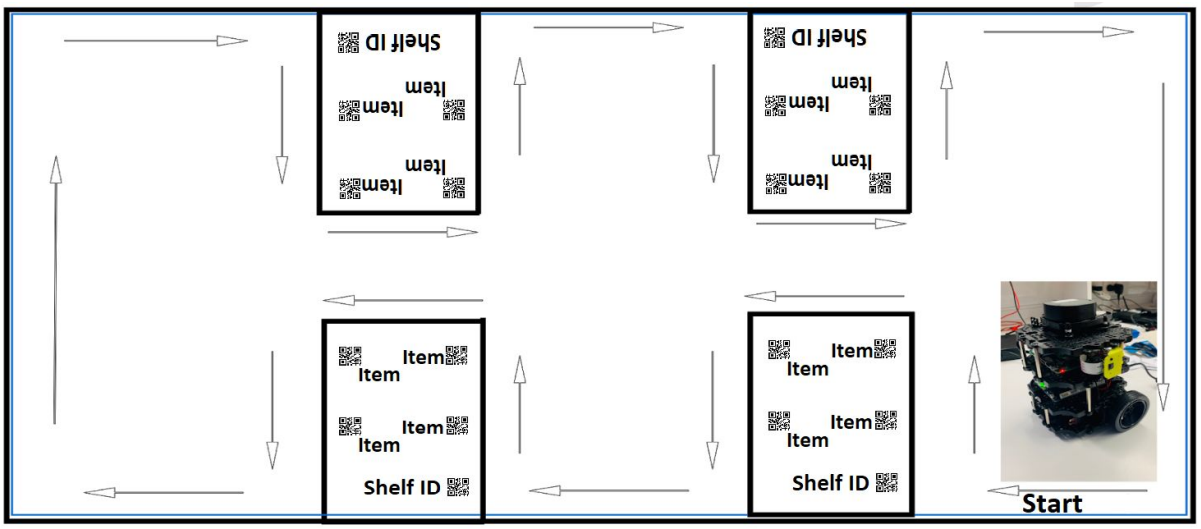


Figure 3 (b): Shelf Layout and Inventory Robot Path

The **design approach** of the system can be broken down into four main modules. They are:

Item Detection
- OpenCV & QR Scanner
- Camera

User Interface/Program
- Customized Command Line GUI

Obstacle Avoidance
- Light Detection and Ranging (LiDAR) System

Inventory
- Disk Storage with Human-readable Text.

Detect and print out changes to inventory between scans

For this functionality to work, a copy of recorded items from previous scan is saved in the file “oldInventory.txt” every time before a new inventory scan is carried out. At the completion of a new scan the new list of items are first saved in “inventory.txt” file whose data is later extracted and compared against the old list every time inventory check is required by an operator. When this happens the newly added items are displayed as in *figure 4*.

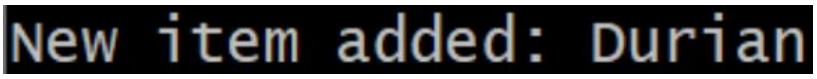


Figure 4 (a): Addition of New Item

In technicality, *std::vector* standard library containers are used to carry out this operation as shown in the flowchart below.

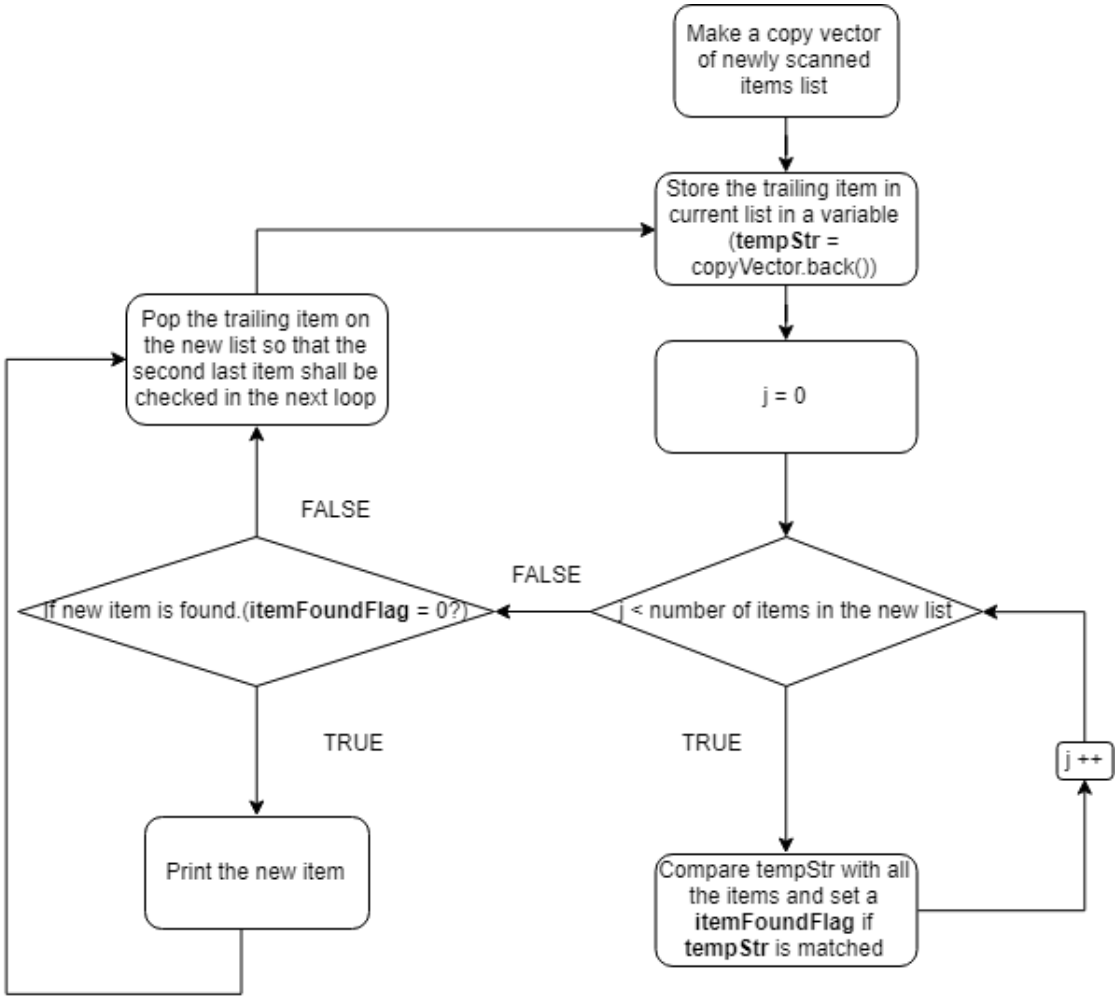


Figure 4 (b): Operations

The physical operation of turtlebot demonstrates that the PC runs the turtlebot on remote connection. The turtlebot starts immediately on request. The ROSCORE and decision making node runs on the turtlebot. The stop function was tested and operated independently, however integration has not taken place yet but can be completed in the near future.

The wall follower program was written to identify objects at set angles of 30-45 degrees (adjustable) to the LIDAR on the left side of the turtlebot (front left side). The readings would be taken at less than one second intervals with the configurable velocity on the servos and the direction was controlled using the wheel speeds individually (toggling different speeds of the wheels at certain times or in specific cases). The control loop allows for this speed configuration.

Testing

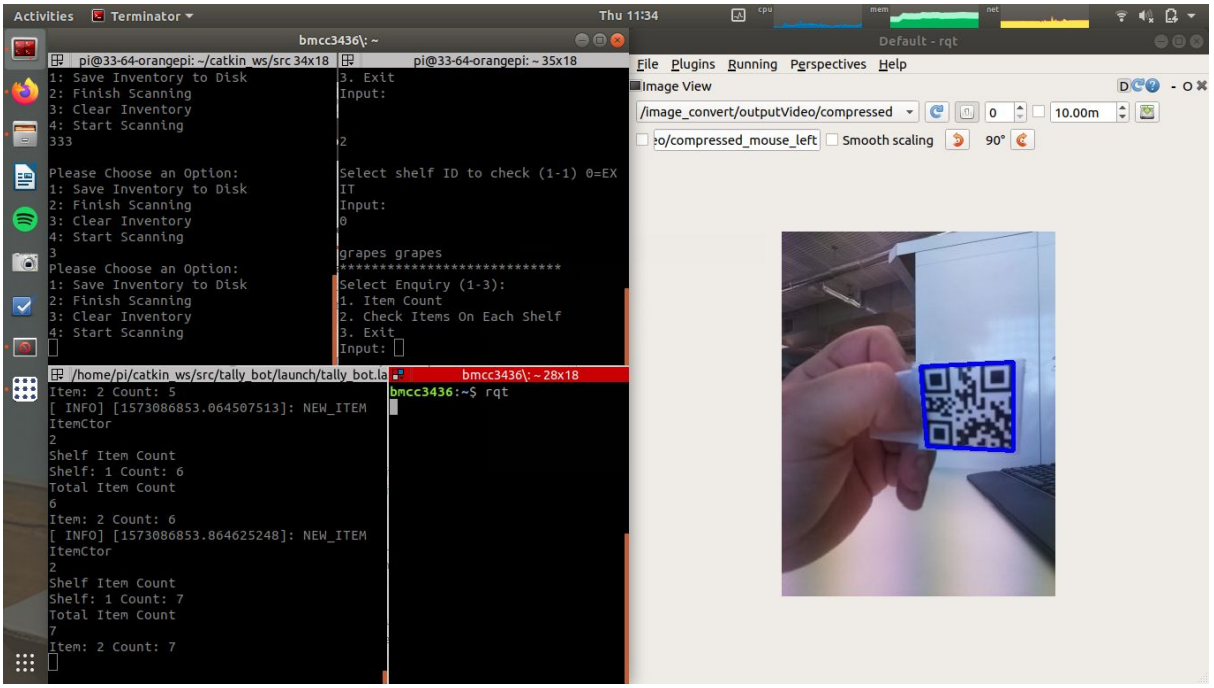
Simulation Testing

The simulation testing involved the employment of modelling software to visualise the program by running the code upon each change in the coding alongside utilization of modelled shelves. The exact movement of the turtlebot could then be deduced from the simulation and therefore, a similar behaviour can be expected when tested with the physical setup.

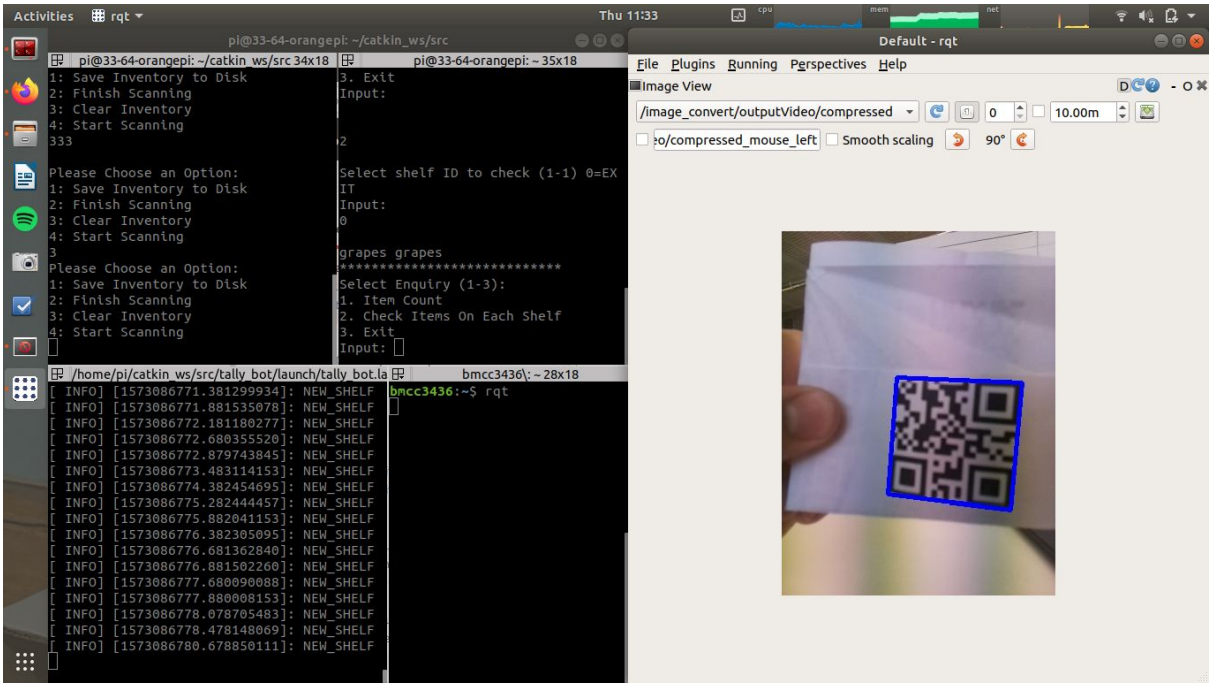
SImilar to the testing of the left wall follower system, the simulations remained with similar results as the turtlebot would follow the left hand side wall of the modelled store successfully. For future applications of this system, it would be appropriate for the alteration of the wall follower algorithm. This alteration should allow for a more flexible application of the turtlebot in a variety of store/shelf layouts. Ultimately this modification would result in:

- Ability to follow aisles that are more realistic in layout.
- Ability to track more stock.

QR Scanner and Camera Tests



Screenshot 1: Item Scanning



Screenshot 2: Shelf Scanning

Physical Testing

The testing stage involved the utilization of physical setups resembling a store with single layer shelves. These shelves would ideally be boxes with QR codes representing the item. The QR codes were stuck physically onto the box using an adhesive. As the turtlebot would detect the box, it would maneuver itself through the modelled aisle (gap between the boxes) with the camera on the left scanning the item codes.

It was concluded that the need for QR codes for shelves was unnecessary and therefore, the shelf ID can be determined using the movements of the turtlebot. Upon each rotation that occurs at a 180 degree angle, would mean that one shelf has been fully detected and the second shelf's detection process is about to begin. Using this algorithm and the detection of the wall, a shelf counter can be created that is accurate to its real time detection of new shelves.

The first physical test had displayed various bugs with the system due to the synchronised operation of the many functions. For example, the wall follower, when operated as a single node, worked with great efficiency and accuracy, however alongside the image processing the turtlebot would suffer from a varying wall following range, increased possibility of collisions and varying distances from the wall results in the QR scanning to be random. Overall, the turtlebot follows the wall too closely and therefore, the QR code cannot be fully scanned.

To solve these issues found from testing, it was first proposed that smaller QR codes are to be used. Another possible solution was to increase the distance to follow from the wall however, resulted in various random outcomes such as the turtlebot taking wider turns and possibility of collisions. For the future of this project, it would be ideal to revise the wall

follower code and constantly make changes with further testing, however due to time constraints this was not possible as of yet.

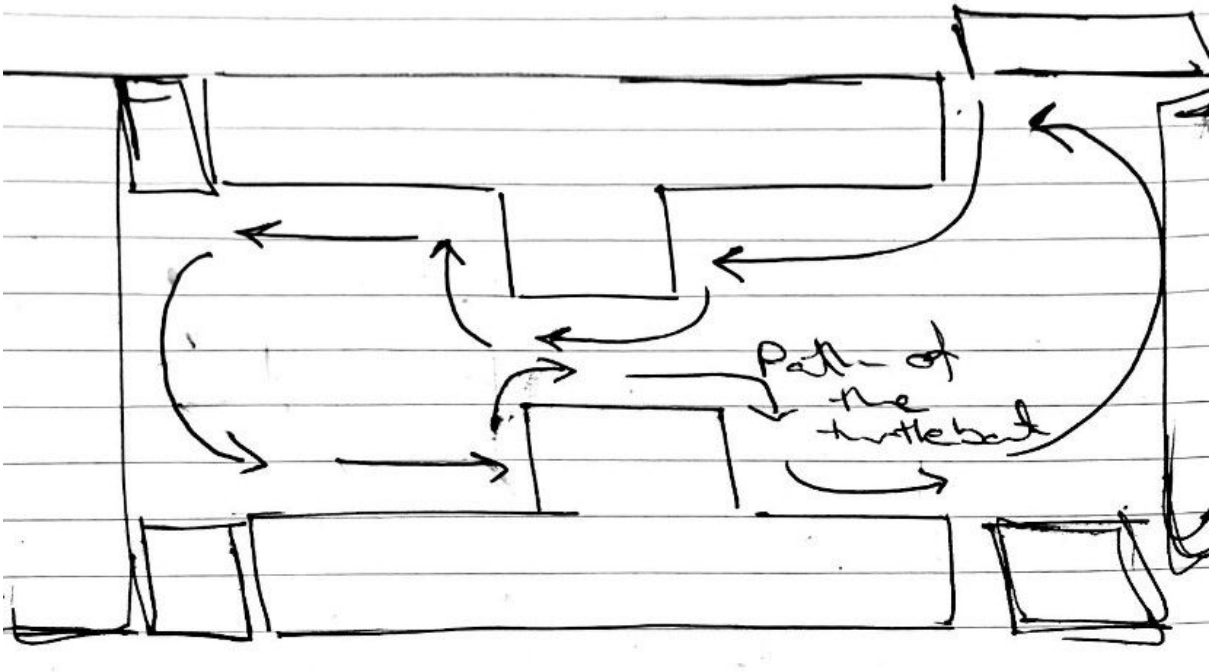


Image 1: Physical Testing Setup A

User Interface Testing and Results

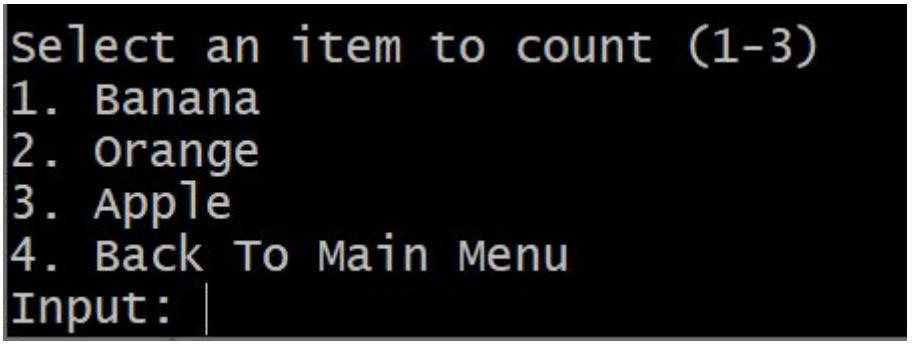


Figure 5 (a): Menu for UI

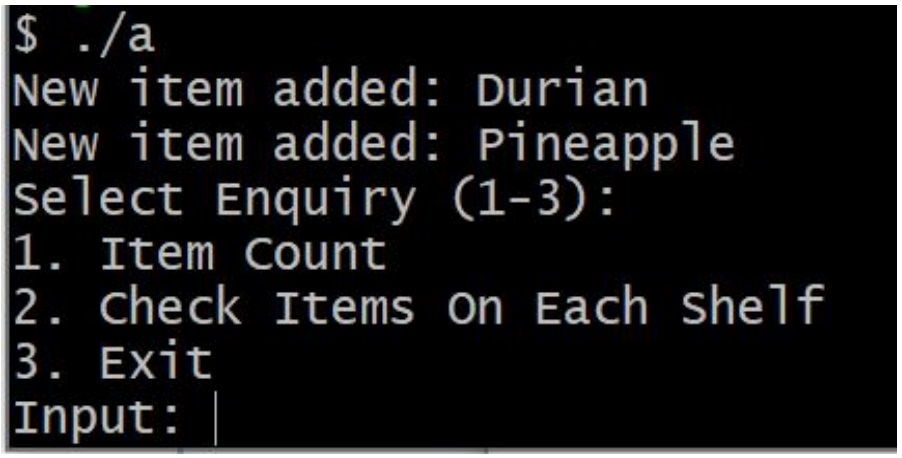


Figure 5 (b): Addition of an Item

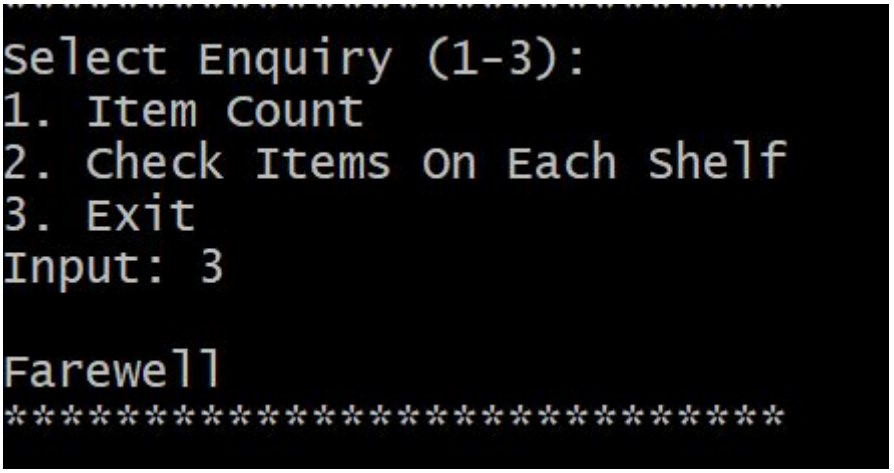


Figure 5 (c): Farewell Message

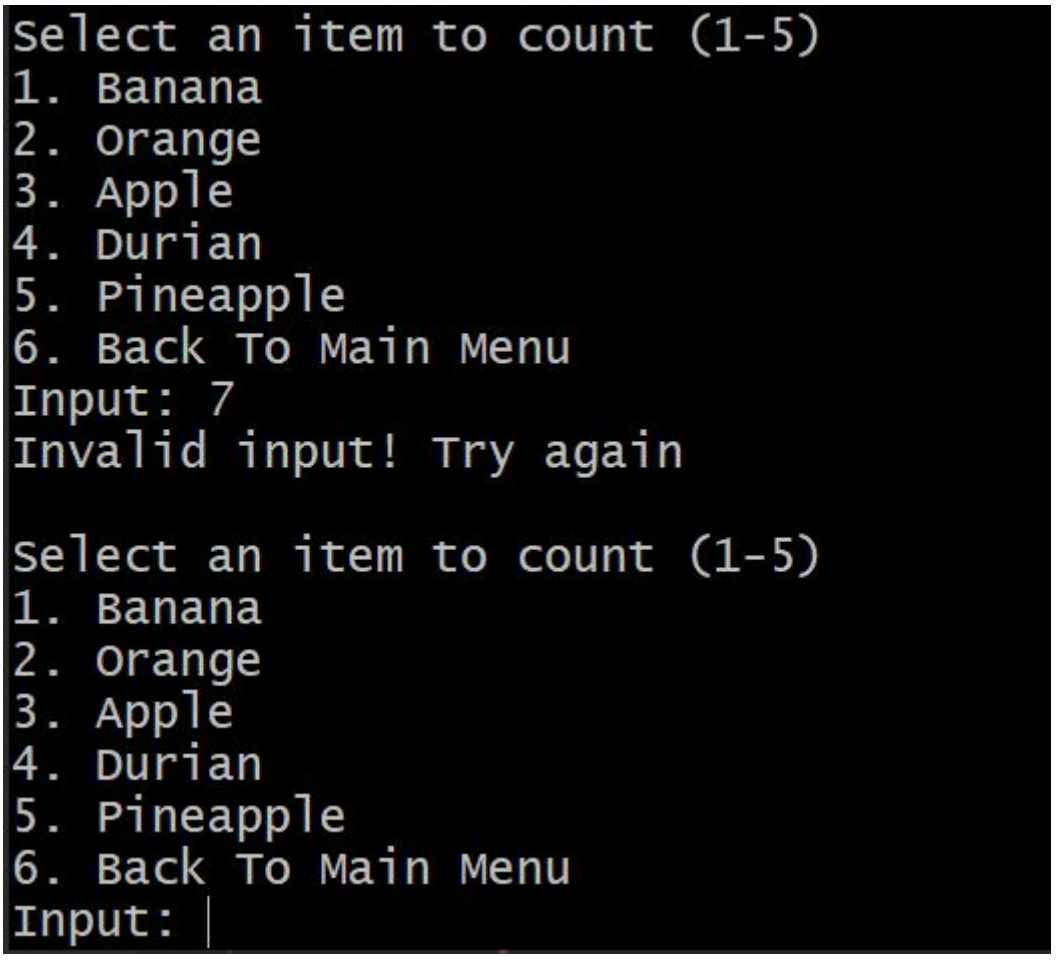


Figure 5 (d): Invalid Input Example

The UI is capable of taking a variety of invalid inputs such as too many characters, not enough characters, invalid characters including symbols and alphabets and blank inputs.

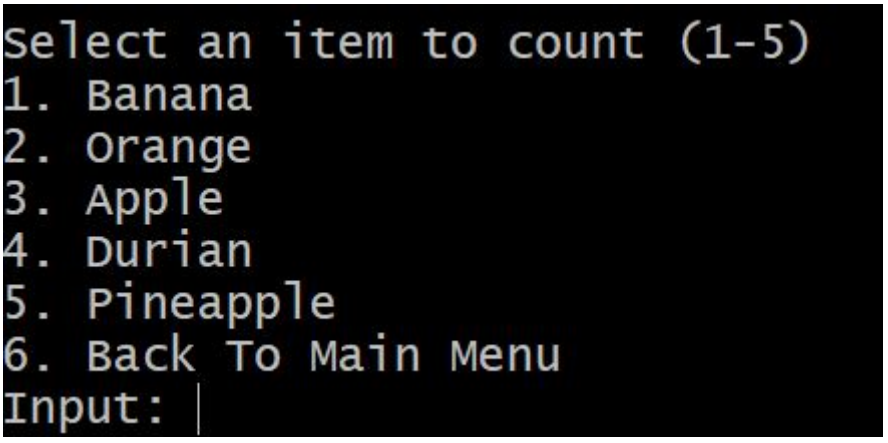


Figure 5 (e): Menu Adjusts itself with new Items Automatically

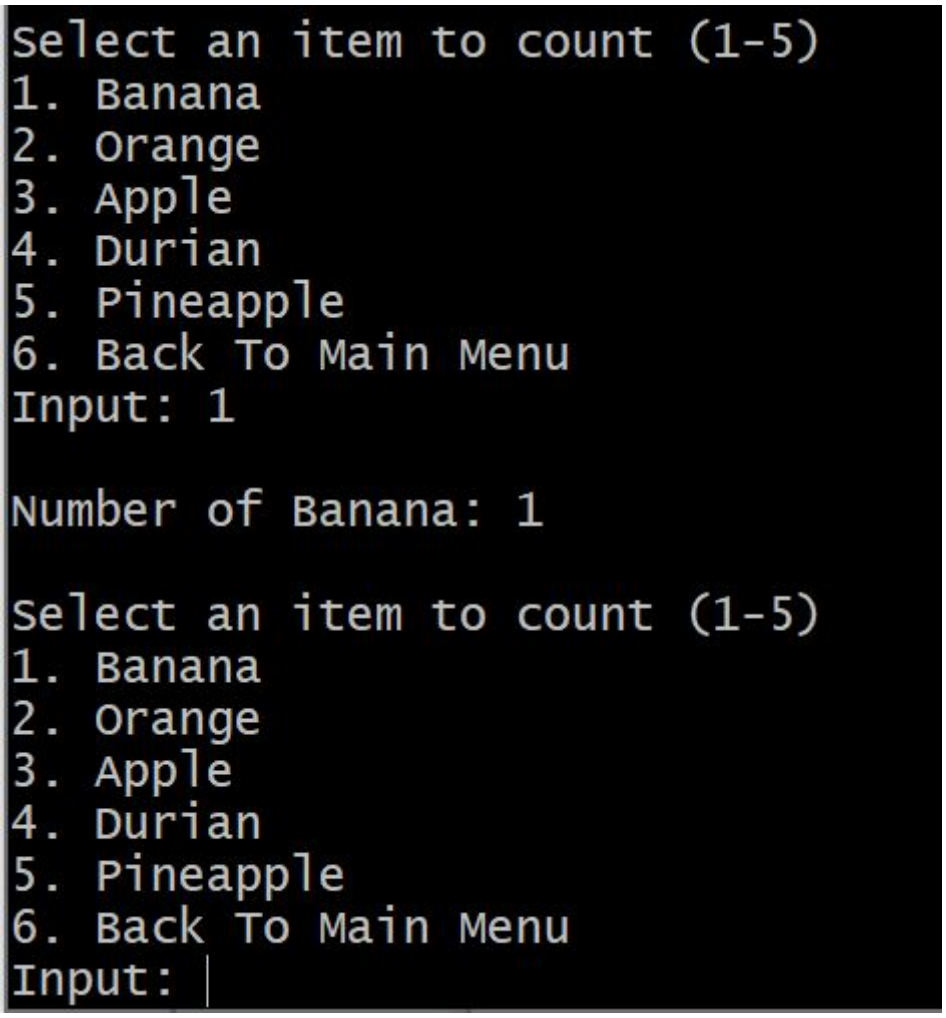


Figure 5 (f): Menu stays in the Loop

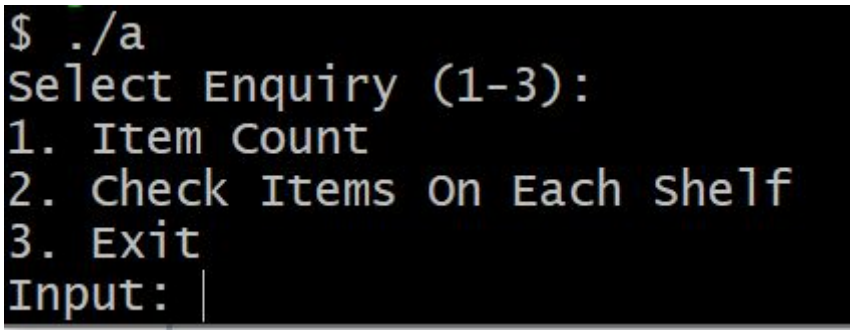


Figure 5 (g): No New Item Added

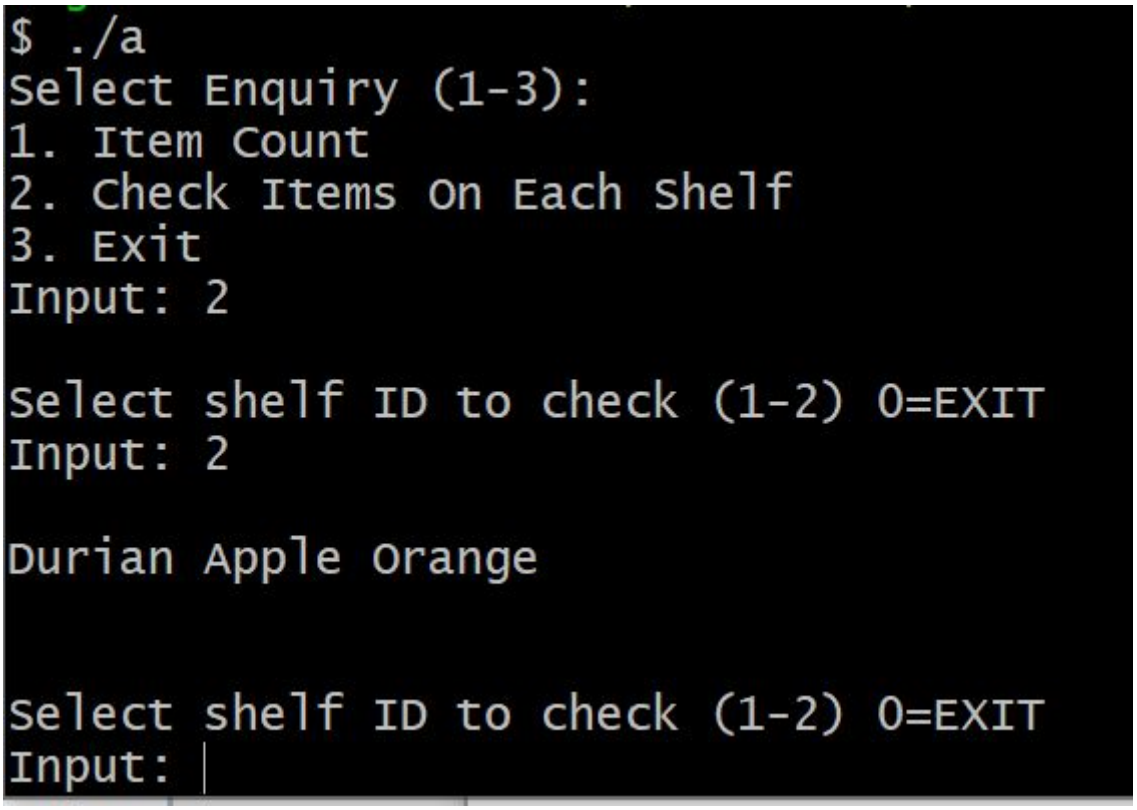


Figure 5 (h): Select Shelf ID

Discussion

The final demonstration of the project has displayed that the original scanning plan of going around the aisles or using the turning method as displayed in the figures below. Due to time constraints the final store setup is displayed below. For the future of this project it is recommended that the following changes:

- Alter the wall follower program to account for a more flexible store arrangement.
- Work on the accuracy of the movement for better QR scanning.
- Utilize a smaller set of QR codes.
    - Have to account for the store having variable sizes of items.
- Additional functions that are highlighted can be implemented in the near future.

Elements of the project that was completed well include the user interface, a portion of the advanced inventory function as mentioned in the level of performance and the overall follower program.

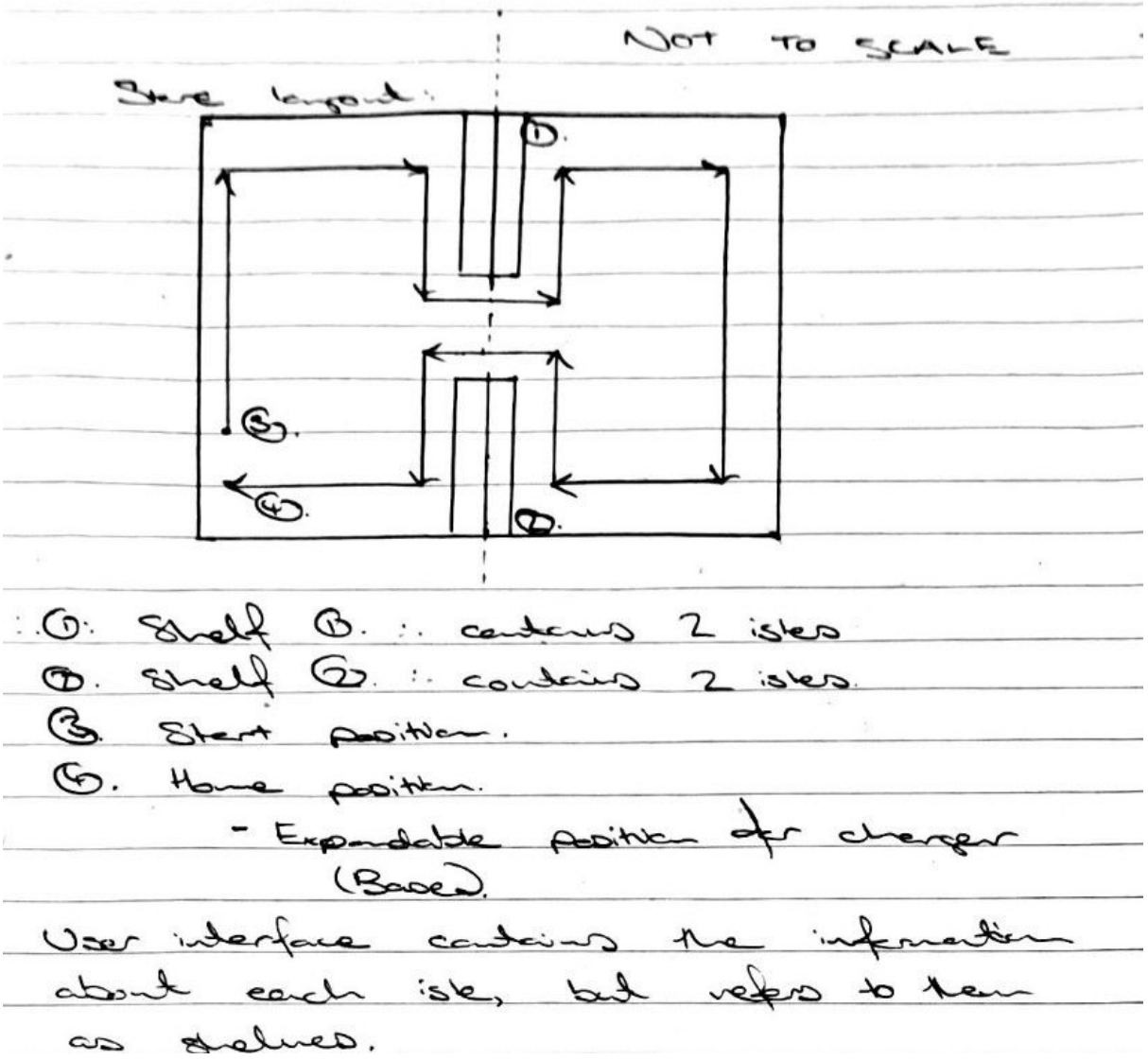


Figure 6: Final Store Layout (Used for Demonstration & Testing)

Conclusion

In summary, the basic requirements/functionalities of the project were all met to an acceptable standard. This system would now be capable of integration to a physical store with some further development and testing.

Appendix

Meeting Minutes

Meeting 1

Date: 14/10/2019
Time: 11am-2pm

Notes:

- Planned the store layout.
- Overall discussion of the approach to the system.
- Sharing ideas of the most effective method of scanning.
- Discussing possible movements of the turtlebot in the store.
    - Only utilizing the left wall follower.
    - Adjusting the follower code to scan an aisle by going around it.
    - Adjusting the follower code to scan down an aisle, turn around and scan back up the aisle.

Meeting 2

Date: 28/10/2019
Time: 11am-2pm

Notes:

- Commenced on the creation of the report.
- Start the installation of necessary packages.
    - Starting OpenCV configurations.

Meeting 3

Date: 02/11/2019
Time: 8pm-1am

Notes:

- Further changes to the system design.
    - Changing the arrangement of the store.
- Started the user interface program.
- Started the camera configurations with the QR scanning program.

Meeting 4

Date: 03/11/2019 - 04/11/2019
Time: 12pm-11am

Notes:

- Completed the project.
    - Attempted advanced inventory program to state where the item is in the store.
    - All required functions are completed.
- Completed the QR scanning program.
- Creation of the inventory program
- Inventory integrated with the user interface program.

Project Plan

The majority of the design approach utilized within the project was that of a *Waterfall Project Management Methodology.* Most of the hours spent on the project was completed by all members when meeting up together with the equipment, performing testing and progressing on the required outcomes.

Tasks

The following is a simplification of the tasks that are involved in the project:

1. *Mapping*
2. *Navigation*
3. *Detection*
4. *Data Handling*

A majority of the tasks were completed sequentially with the involvement of all group members as some tasks had large reliance on each other and had many subtasks.
For example, the data handling would tie into GUI design and output but would rely on the detection system therefore, in reality most the the team had worked together on each task to fully complete the objectives before moving onto the next.
The following table denotes some of the strong elements of this approach to the system design as well as some difficulties faced when using this approach.

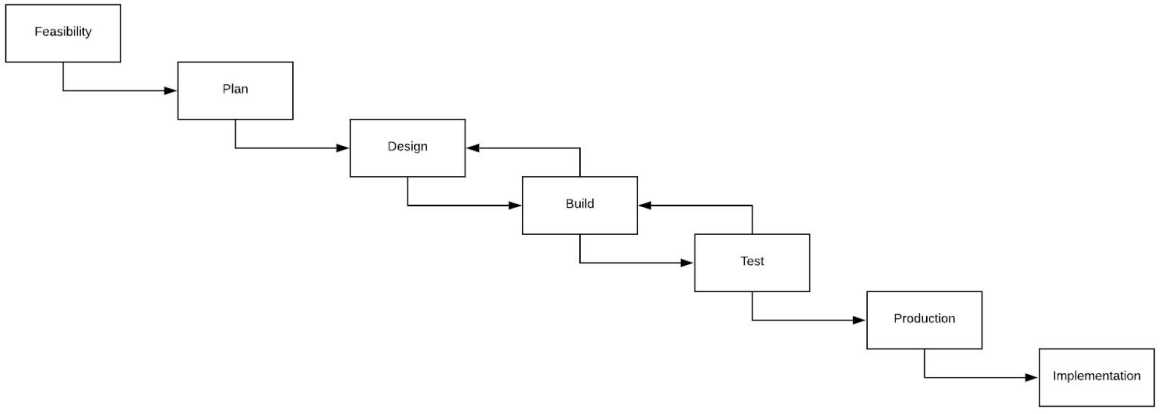


Figure 7: Representation of the Project Approach

Within this representation the testing phase highlighted issues and therefore, the project approach was flexible in allowing for the rework of certain aspects of the project. This was also permitted by the time of the project.

Table 2: Pros & Cons of Waterfall Design Approach

Pros	Cons
Simple to plan a project and most of the tasks allocation is sequential.	The project plan requires individuals to be together for most of the progress to be made.
Most communication that affects the project directly is consistent for all members, hence all members discuss a similar aspect of the project at hand at the same time.	There is less communication made over the internet or GitHub and similar services that are used.
Less independent reliance, therefore, no hindering of the project progress if one individual is not satisfying deadlines.	Time is hindered by there being less work being completed away from group meetings.
Each objective is satisfactorily achieved on the terms of all group members.	The tasks as a whole require greater time for completion as there are all/most members working on the same objectives at one time.
Collaboration is still employed (GitHub, Cloud Sharing) & the individual shared components are much smaller.	There is little reliance on collaborative services over the internet with some services that are available being unused.
Within this specific case, the tasks were completed faster as all members would work on aspects that are smaller for each task. *Look at in depth breakdown below for steps taken by specific individuals.*	Each individuals impact on the project is dependent on their effect on the progress during the meetings.
Error checking is completed with higher detail and more accuracy with multiple people working on the same code or checking code at once.	Only one person can make changes to the project and test at once.
Multiple ideas are shared at once with each task that the group collaboratively completes at the same time as it is being completed.	
Do not need to wait for a single person to complete their part for progress.	

Team Coordination

The following section highlights in summary the roles that each group member completed in the project:

Table 3: List of Member's Contribution in the Project

SID	Roles
460386784	- Assisted in the employment of the user interface. - Configuration and Calibration of the QR scanner. - Configuration of the camera. - Attempted the mapping advanced system requirement. - Created the image processing system. - Configured and majority created the inventory.
470318966	- Assisted in the creation of the presentation.
470023233	- Installation of the camera. - Creation of the shelf counter user interface. - Integrated user interface code with other participants code and the main system. - Implemented advanced functionality of detection of new items between scans - Assisted with project report - Created the pitch presentation.
470416309	- Creation of the shelf list in the user interface. - Error checking of the user interface. - Partial completion of the advanced item finder. - Assisted in the item counter. - Writing of the project report.

References

ROBOTIS e-Manual. (2019). Retrieved 22 September 2019, from http://emanual.robotis.com/docs/en/platform/turtlebot3/overview/

Robots/TurtleBot - ROS Wiki. (2019). Retrieved 22 September 2019, from https://wiki.ros.org/Robots/TurtleBot

(2019). Retrieved 10 November 2019, from https://projects.raspberrypi.org/en/projects/getting-started-with-picamera