Warehouse Automation

Design Document

sddec20-10
Client: N/A
Advisor: Dr. Diane Rover

Jacob Ramsey-Smith
Amir Hamza
Stamatios Morellas
Jack Creighton

Team Email: sddec20-10@iastate.edu
Team Website: sddec20-10.sd.ece.iastate.edu

Executive Summary

Development Standards & Practices Used

- MVC style coding in the backend.
- Agile and Test Driven Development practices will be used

Summary of Requirements

- Have some method to take inventory in a warehouse
- Create a module design which can be integrated into different warehouses
- The gathered data must be transmitted to a web server via wireless communication
- Data in the server must be viewable in a web application

Applicable Courses from Iowa State University Curriculum

- CprE 288: Embedded Systems
- ComS 309: Software Development Practices
- ComS 319: Construction of User Interfaces
- EE 333: Electronic System Design

New Skills/Knowledge acquired that was not taught in courses

Most knowledge required in this project has been taught in the courses mentioned above.

Although most of the relevant knowledge and concepts were covered in previous courses, the Software Team was new to working with some technologies such as React, Express, and MongoDB.

This was also the first time anyone on the team has worked with Webots.

Table of Contents

List of figures/tables/symbols/definitions

1 Introduction

1.1 Acknowledgment

The team appreciates Dr. Rover for choosing us for the warehouse automation project. Our team is excited to contribute efforts in this regard and would like to deliver a project that is sustainable, easily integrative, and cost-effective. Our team has a wide range of skills that can be combined to deliver an effective product that meets all the requirements.

1.2 Problem and Project Statement

Warehousing contributes up to 30% of the cost of logistics in most developed economies. Billions of dollars of capital are locked up at a time, in high-volume-high-value inventory. Since the rise of the internet, eCommerce has exponentially increased in consumer demand and efforts need to be made to keep up with this demand. Traditional warehouse workers are falling short and need technological integration to help keep track of the high volume of inventory moving in and out of the warehouse.

Our team will research, design, simulate and demonstrate a warehouse inventory automation system using an application called Webots. The team will need to realistically scope the project for the time, resources, and expertise available. The team is responsible for creating a small warehouse environment appropriate for the proposed solution. This application will serve as a demonstration to potentially initiate a larger automated warehouse testbed for the department with industry collaboration.

Autonomous aerial drones will be utilized to scan the QR codes for each pallet at different heights in an aisle. Our team will be designing and building an aerial drone to scan QR codes. This drone will be operated manually by an operator and then set to autonomous if desired. Data from the drone will be transmitted to the server. A website will also be developed to view the gathered data and run analytics on it.

1.3 Operational Environment

Due to COVID-19 our team decided to change our project from requiring an in person operating environment to having a fully simulated operation environment. This flexibility allowed our team to be off campus and still work on the project.

Because our environment is simulated the drone is acting under optimal conditions. However, to account for a more realistic environment, some testing is done for realistic constraints, such as distance from box and minimum required lighting. There are also safety hazards that must be taken into consideration. As such, workers would have to stay out of designated areas during drone operations.

1.4 Requirements

1.4.1 Engineering Constraints and Non-functional Requirements

Non-Functional Requirements:

- **Market requirements:**
    - Create a module design which can be integrated into different warehouses
    - Develop a cost-effective system
    - Ensure low running cost
    - Ensure project requirements are being met through seeking feedback

- **System Requirements:**
    - Drone must be capable of communicating with the server
    - Develop a web application which reliably displays the warehouse data gathered by the drone
- **Administrative Requirements:**
    - Assign tasks on Trello to each member of the team
    - Weekly team meeting to discuss progress and address concerns
    - Weekly team meeting with the faculty advisor to discuss progress and future goals

Engineering Constraints:

- Webots drone was limited to using a camera because there was no QR code scanner object
- Boxes in a warehouse can be stacked at a variety of heights, so the object scanning the inventory must be capable of changing heights
- The team worked under the assumption that people worked in the warehouse so their safety must be taken into consideration

1.4.2 Functional Requirements

- The drone should be able to fly to the height of most warehouses
- The done must scan QR codes off of the inventory
- The data gathered from the inventory must be transmitted to the server
- The transmitted data must be viewable for intended users and uses

1.5 Intended Users and Uses

The intended users for our project are the companies operating warehouses for their inventory. Our product will be used to automate inventory tracking in those warehouses. This will enable warehouse owners to keep track of their inventory effectively and will result in long term savings through better analytics and increasing personnel efficiency.

1.6 Assumptions and Limitations

- **Assumptions:**
    - There is sufficient space between the aisles for operating aerial vehicles safely
    - The drone is not recommended for outdoor use
    - Since the location is indoors, no extra water or dust protections will be installed
    - The environment is well lit for the drone to scan barcodes
    - Inventory is well organized and evenly spaced
    - The person controlling the drone is well trained
- **Limitations:**
    - The QR code must be large enough for the drone to pick up
    - The QR code must be clean enough for the drone to pick up
    - The drone shall always operate in the range of a wireless access point

    Refer to Appendix II to view more limitations from initial design

1.7 Expected End Product and Deliverables

The expected end product will be a simulation of a drone with partial autonomous capabilities. The drone shall be capable of transmitting data acquired from QR codes to a backend server.

A web application will also be shipped with the drone which will display the data received by the server from the drone. This data will be updated in real-time as items are scanned. The user will also have the option to run analytics on the existing data to see which aisle has been the busiest or which product has been shipped the most etc.

Project Deliverables		Due
1	Project Plan	January
2	Research Hardware Components	February
3	Research Software Components	February - April
4	Construct Project Schematics	March
5	Perform Cost Analysis	March
6	Components Ordering	August (originally April, delayed due to COVID)
7	Product Assembly	September
8	Software Testing and Debugging	September - November
9	Hardware Testing and Debugging	September - November
10	Poster	November 15th
11	Final Report	November 15th
12	Final Presentation	November 19th
13	Product Delivery	November

Figure 1: Project Deliverables

1.8 Related Work and Literature

Related Work (Similar Companies):

- Ware
    - Ware creates autonomous drones that function in a very similar manner to our project, capturing barcodes, identifying inventory locations, then sending inventory reports to the cloud.
- FlytBase
    - Flytbase has a program called Flytware that operates in a similar fashion to our project. Complete drone fleet automation software.

Related Literature:

- There is a lot of related literature on drone technology and A.I. that can be used in autonomous systems. This can be viewed in Section 6.2.

2. Revised Project Design

This following is an overview of the team's project design.

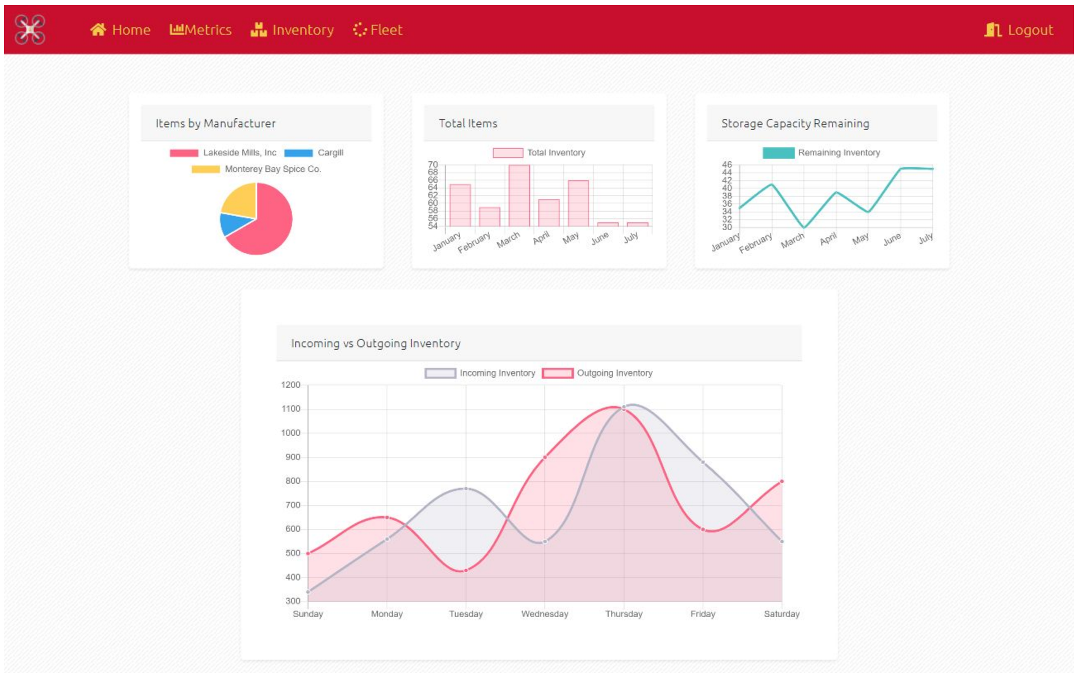
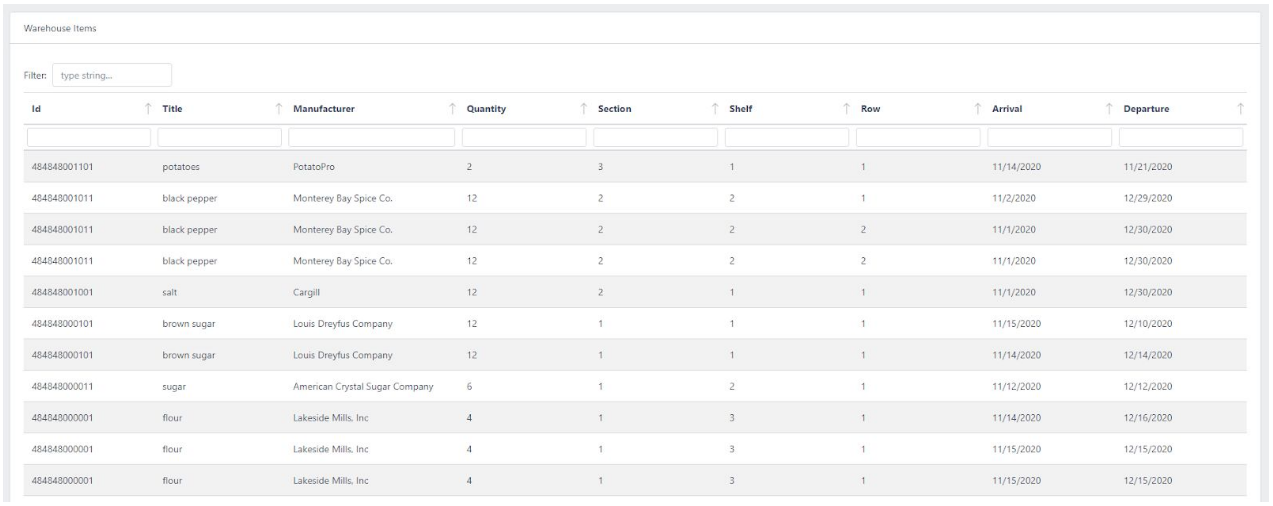


Figure 2: Web Application Dashboard



Warehouse Items

Filter: type string...

Id	Title	Manufacturer	Quantity	Section	Shelf	Row	Arrival	Departure
484848001101	potatoes	PotatoPro	2	3	1	1	11/14/2020	11/21/2020
484848001011	black pepper	Monterey Bay Spice Co.	12	2	2	1	11/2/2020	12/29/2020
484848001011	black pepper	Monterey Bay Spice Co.	12	2	2	2	11/1/2020	12/30/2020
484848001011	black pepper	Monterey Bay Spice Co.	12	2	2	2	11/1/2020	12/30/2020
484848001001	salt	Cargill	12	2	1	1	11/1/2020	12/30/2020
484848000101	brown sugar	Louis Dreyfus Company	12	1	1	1	11/15/2020	12/10/2020
484848000101	brown sugar	Louis Dreyfus Company	12	1	1	1	11/14/2020	12/14/2020
484848000011	sugar	American Crystal Sugar Company	6	1	2	1	11/12/2020	12/12/2020
484848000001	flour	Lakeside Mills, Inc	4	1	3	1	11/14/2020	12/16/2020
484848000001	flour	Lakeside Mills, Inc	4	1	3	1	11/15/2020	12/15/2020
484848000001	flour	Lakeside Mills, Inc	4	1	3	1	11/15/2020	12/15/2020

Figure 3: Inventory Table

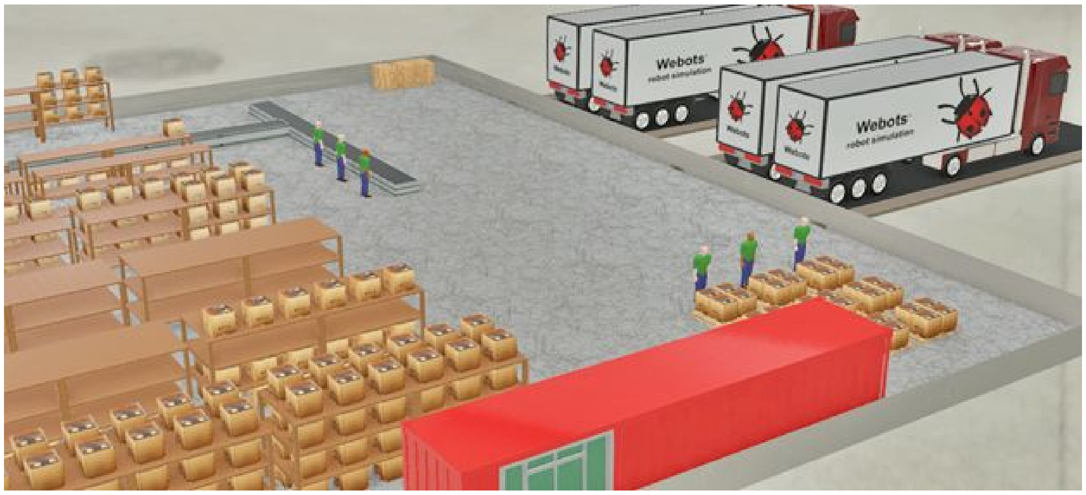


Figure 4: Warehouse Layout

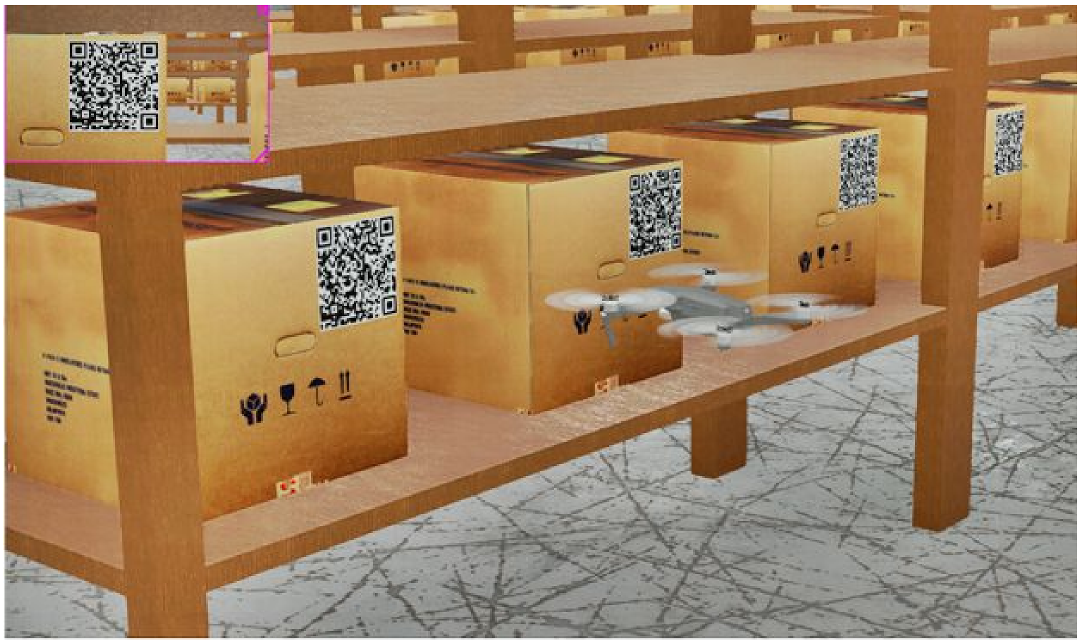


Figure 5: Drone Scanning

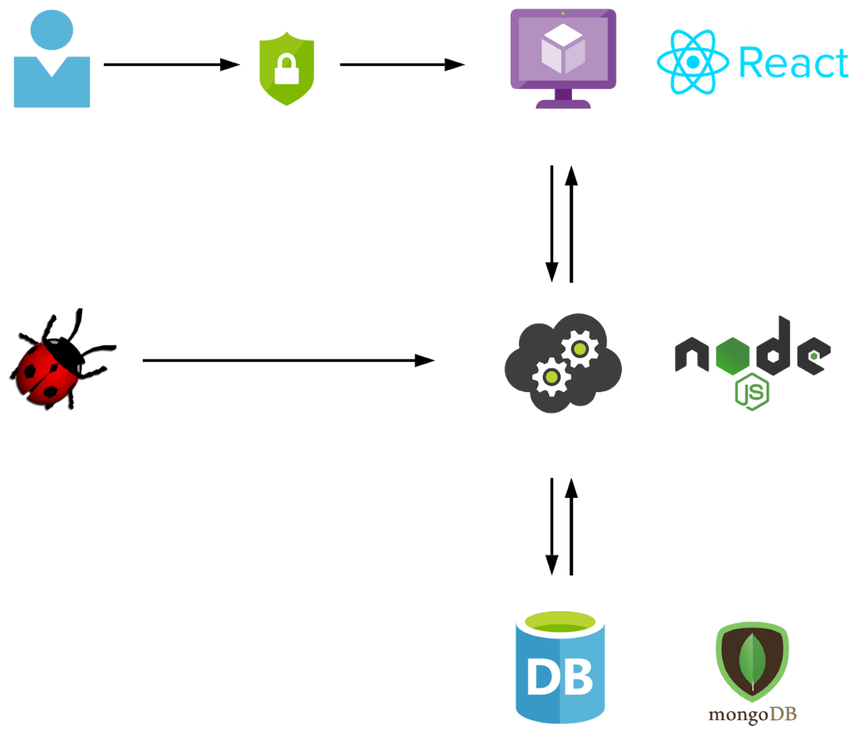


Figure 6: Project Design Overview

Our web application consists of 5 main pages: A homepage, an inventory page, a metrics page, and a fleet management page. The following image shows the state diagram of the web application.

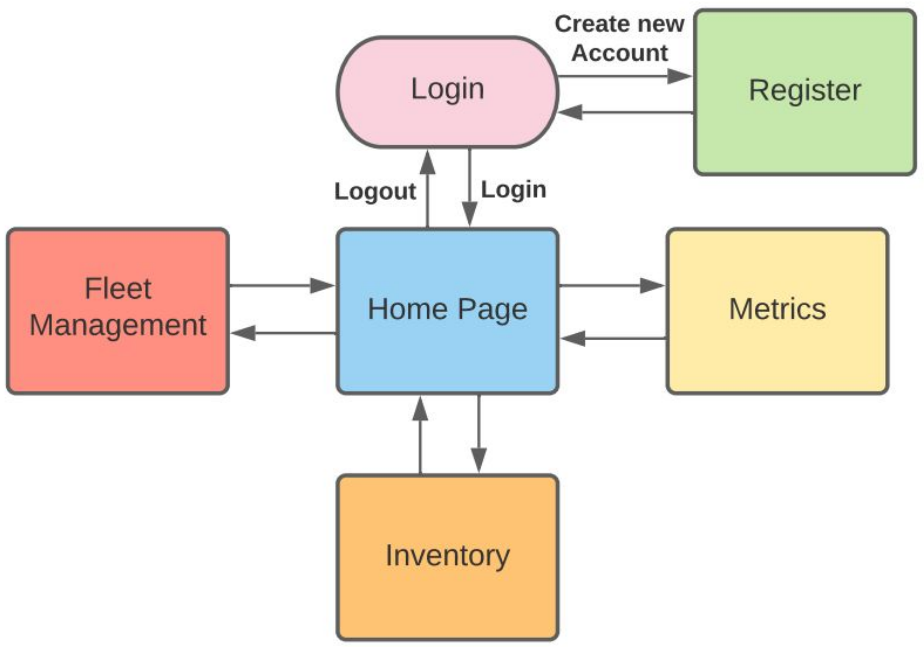


Figure 7: Web Application State Diagram

For the webots environment, the team set up a sample warehouse and inventory with QR codes. A drone was then added to the warehouse to scan the QR codes and send the stored information from the QR codes to the server.

State transition diagram for the webots code controlling the drone

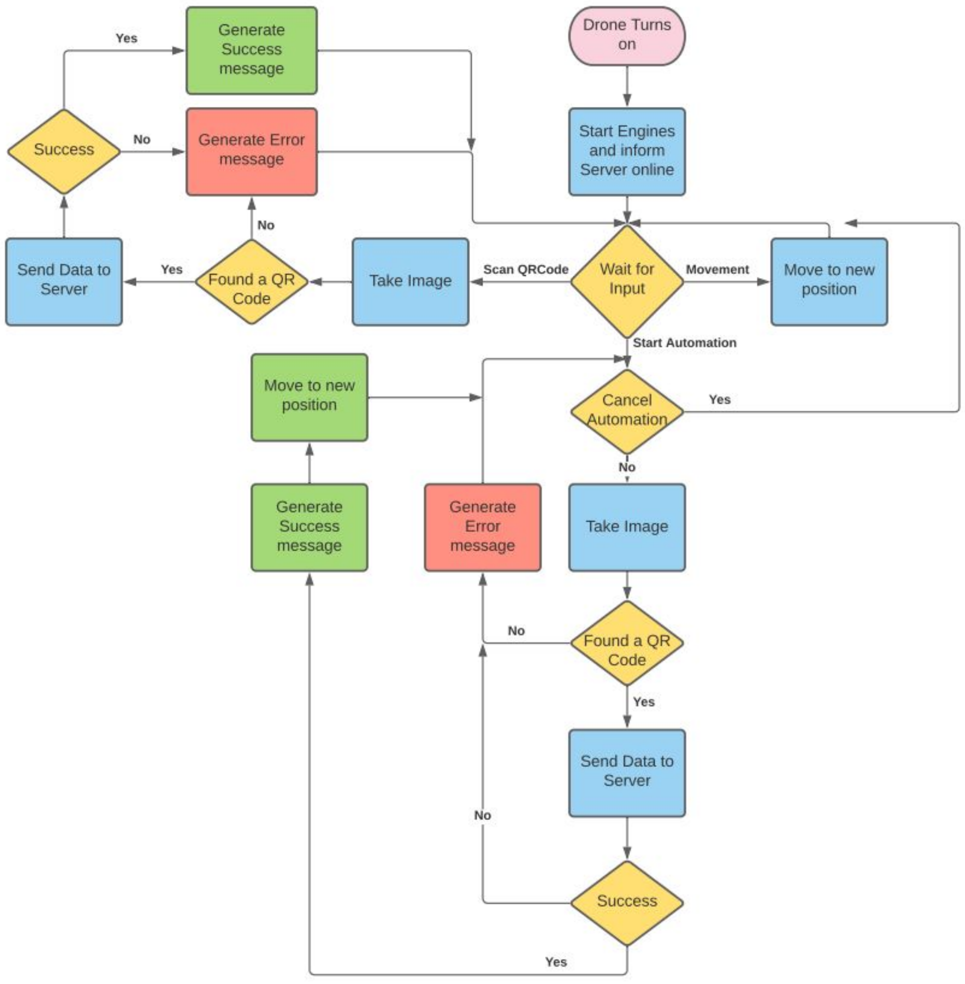


Figure 8: Webots State Diagram

3. Implementation Details

Webots:

- The drone is equipped with a camera. Since there is no barcode scanner in the webots environment, the team had to create a work around. The team decided to take an image of the QR code and then process it using Google's Zebra Crossing library.
- After the image has been successfully processed, the data is put into a JSON object and sent to the server in an HTTP Request.
- There are no IR or LIDAR sensors on the drone, so it should be defined as limited automation, as it has to be aligned manually and if the drone gets off course it has to be manually reset.

Web Application:

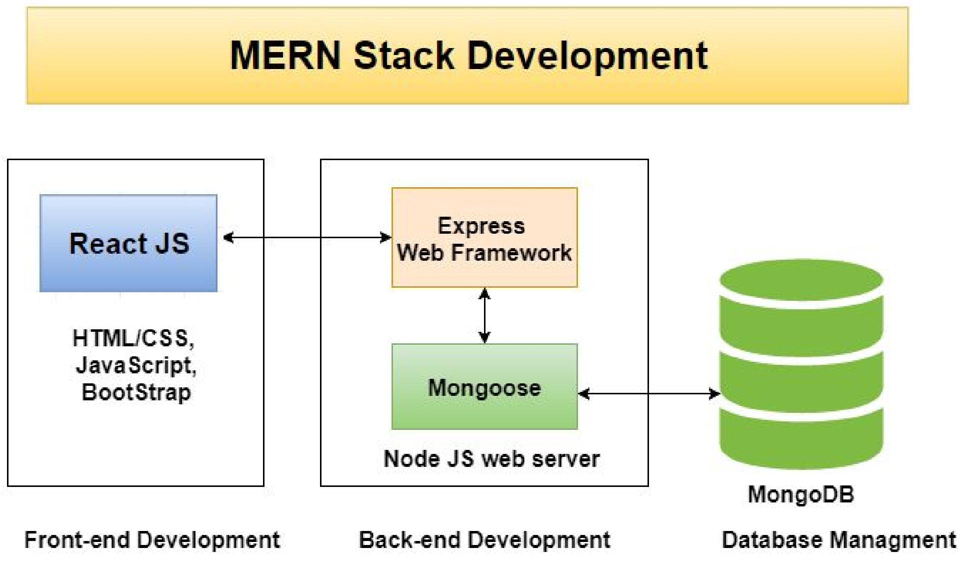


Figure 9: MERN Stack Architecture

- Our web application was built using the "MERN" Stack: MongoDB, Express, React, and Node.js. We have split our components into three parts:
    - Client Application (Front-end):
        - To design our user interface, we used Adobe Xd to construct what we visualized to be the expected outcome. Since we built the front-end using React, we chose to use a third-party components library called CoreUI.
        - CoreUI made it significantly easier for us to implement views for the pages, tables, etc, and to implement features such as table sorting and filtering. Since all of the components are pre-styled, this allowed us to focus less on the UI and more on the functionality and features we wished to implement.
        - We chose to use React since it provides a nice front-end framework for JavaScript component-based development. There are also a lot of resources available for React, and it is rather easy to pick up assuming a developer has a basic understanding of JavaScript.

- Server Application (Back-end):
    - The back-end was built using Express and Nodejs
        - The primary responsibility of the server is to handle incoming HTTP requests from Webots and the Client
        - POST requests from webots add new items to DB
        - GET requests from client to access item data
    - Uses mongoose to read/write DB

- Database:
    - The current Schema is simple, meant only to keep track of inventory of one warehouse. We used one table to keep track of our users and another to store the items in the warehouse.
    - Support for multiple warehouses could be added in the future by creating a table for the warehouses and adding a "warehouse_id" field to the item schema to keep track of which warehouse an item belongs to.
    - Has a whitelist to ensure read/write access is restricted to trusted IPs.

Schemas:

Item Schema:

Field	**Type**	**Description**
_id	ObjectId	Unique ID for each item *(generated by MongoDB)*
location.wsection	String	The section of the warehouse that the package is located in
location.wshelf	Number	The shelf of the warehouse section that the package is located in
location.wrow	Number	The row of the warehouse section that the package is located in
productId	Number	ID associated with a product
title	String	Name of the product
manufacturer	String	Manufacturer of the product
quantity	Number	The number of products in a given package
weight	String	The weight of the product
arrival	String	The arrival date of the package to the warehouse Format: MM/DD/YYYY
departure_scheduled	String	The scheduled departure date of the package from the warehouse Format: MM/DD/YYYY

Figure 10: Item Schema

User Schema:

Field	**Type**	**Description**
_id	ObjectId	Unique ID for each user *(generated by MongoDB)*
emailVerified	Boolean	Identifier for if the user has verified their email
name.first	String	The user's first name
name.last	String	The user's last name
email	String	The user's email
password	String	The user's password which is securely encrypted by the database

Figure 11: User Schema

4. Project Timeline, Estimated Resources, and Challenges

4.1 Project Timeline

- This is the team's Gantt chart with project deliverables. These are the assigned projects throughout the semester with defined due dates.

  2nd Semester task timeline:

8/17/2020 - 8/31/2020	- Decision to switch projects to the webots platform and familiarized themselves with the new application - Started setting up a web application repository.
8/31/2020 - 9/14/2020	- Set up skeleton code for web application - Set up drone controller - Found a way to implement QR codes into environment
9/14/2020 - 9/28/2020	- Connected web application to database - Started attempt to connect webots and server via curl
9/28/2020 - 10/12/2020	- Due to complications implementing curl changed drone controller to Java now using HTTP requests
10/12/2020 - 10/26/2020	- Team decided to interpret QR codes from images on webots side of codebase to more closely represent using a barcode scanner - Started user login implementation on web application
10/26/2020 - 11/9/2020	- Implemented partial automation for drone - Added endpoints for metrics and status determined from data sent from Webots

	- Implemented drone status
11/9/2020 - 11/15/2020	- Finished user login - Worked on poster - Worked on Final Report - Worked on Final Presentation

Figure 12: 2nd Semester timeline

4.2 Feasibility Assessment

There are several aspects about a simulated environment that would have to be reassessed if the project were to translate to a real world scenario. The team has done some testing to make sure that the project is feasible, mentioned in section 5. However, there are many more tests that would have to be done to ensure the feasibility of translating this to a real environment. A few examples would be testing the amount of dust in a warehouse that could possibly cover the QR code or finding a warehouse environment that is consistent enough for an autonomous program to work. Several products doing this do exist, as mentioned in section 1.8.

4.3 Personnel Effort Requirements

Tasks	**Effort Requirements**
Creating Webots Environment	This is a lower amount of effort. While there is some labor involved creating all the objects and making it look realistic, It is not too complicated of a task
Creating UI	This will require a medium to a large amount of effort. Designing the UI from scratch gives us a lot of freedom with displaying data; this will require researching the appropriate metrics to display to the user. From a technical standpoint, the admin console is intended to be a responsive web application, so the learning curve of learning a responsive web framework should be the main thing to watch out for here.
Maintaining Inventory Database	It should be a small amount of effort as long as we can properly scan and transmit data to the server.
Creating Drone Controller	This is a large amount of effort. The controller code involves moving the drone, taking photos of the QR code, processing the QR code, sending that data to the server, and implementing the automation process.

Figure 13: Effort Requirements

4.4 Other Resource Requirements

No other resource requirements are necessary. To see list of previous resource requirements, view appendix II.

4.5 Financial Requirements

The team has a $500 budget constraint for our project. Our revised project didn't utilize this budget. To see the previous plan for the budget refer to cost analysis in Appendix II.

5. Testing and Implementation

5.1 Interface Specifications

- Webots communicates with the backend server using HTTP Requests.
    - Webots sends status updates telling the server if the drone is online, taking inventory, in its automation process, or offline
    - Webots sends data contained in QR codes that the drone has scanned to the server
- The client communicates with the server by sending HTTP GET requests to get and display item data.
- The server interacts with the database by using mongoose and its methods such as save() and find() for updating and accessing the database.

5.2 Hardware and Software

The revised project has no hardware pieces. The previous hardware testing phase will be described in Appendix II.

Software used in the testing phase will consist of:

- Java code used to control the webots drone
- Java code used for drone automation
- Java code used to send QR code data to server
- The UI querying the database. This will test that the database is being updated accurately and the UI is working properly.

5.3 Functional Testing

- Drone is capable of flight
- Drone can send data from QR code to Server
- Data from webots can be displayed to user on web application

5.4 Non-Functional Testing

- Drone flight is steady
- Drone can capture QR code from a good distance
- Drone can capture QR code in not perfect light
- Web application has login process that is secure
- Can view data metrics and analytics

5.5 Process

While the tests have changed since the previous semester, the process is still the same.

The following flow diagrams describe the process to test each method indicated in Section 5.2. During the Integration testing stage, additional tests will be done to meet non-functional requirements, such as building a stand with a tether to safely test the drone.

Software Smoke Testing process is described in Appendix II: Webapp Demo/Testing

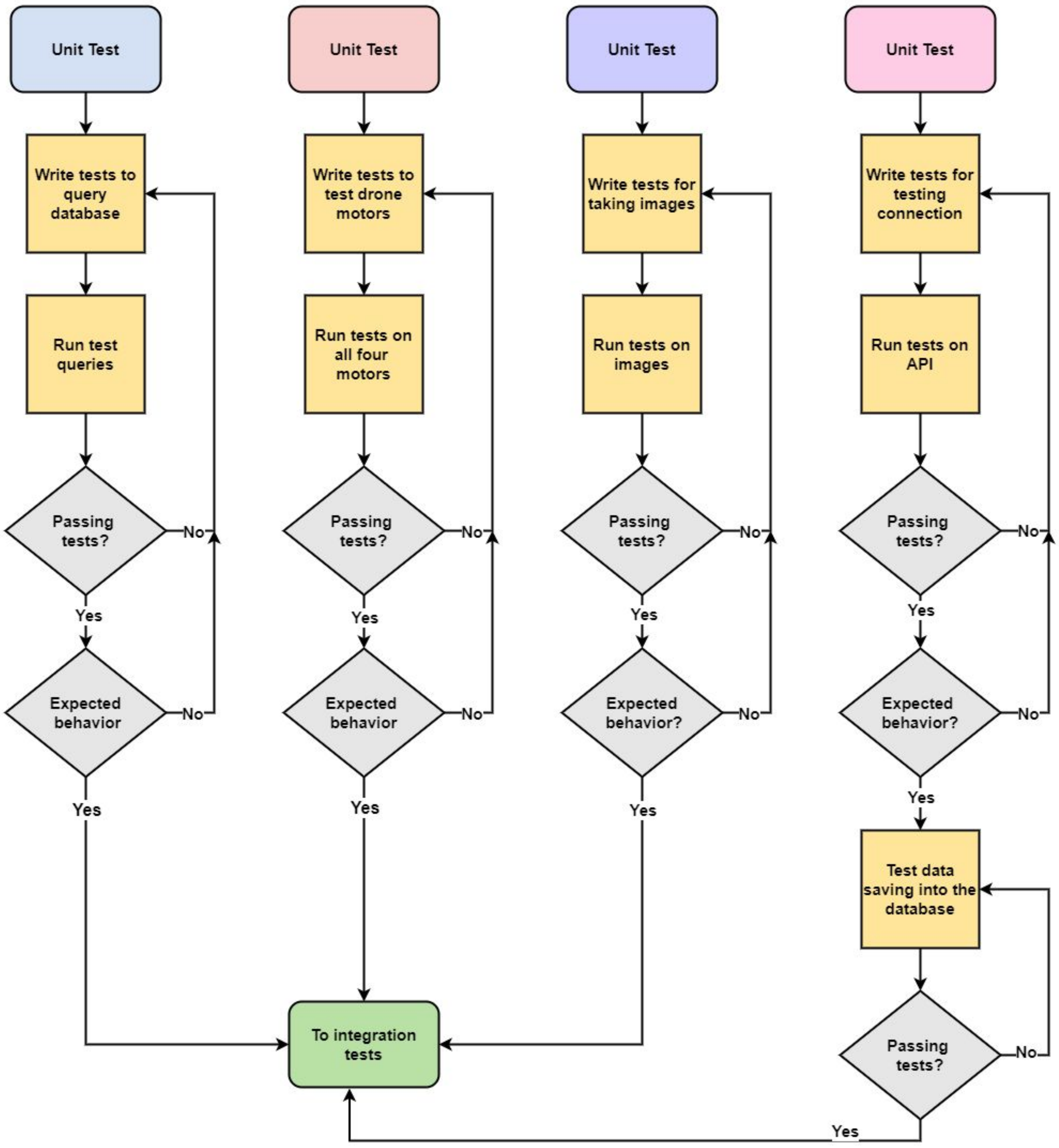

Unit Test
Unit Test
Unit Test
Unit Test
Write tests to query database
Write tests to test drone motors
Write tests for taking images
Write tests for testing connection
Run test queries
Run tests on all four motors
Run tests on images
Run tests on API
Passing tests?
No
Yes
Expected behavior
Expected behavior?
Yes
No
Test data saving into the database
To integration tests

Figure 14: Unit Testing

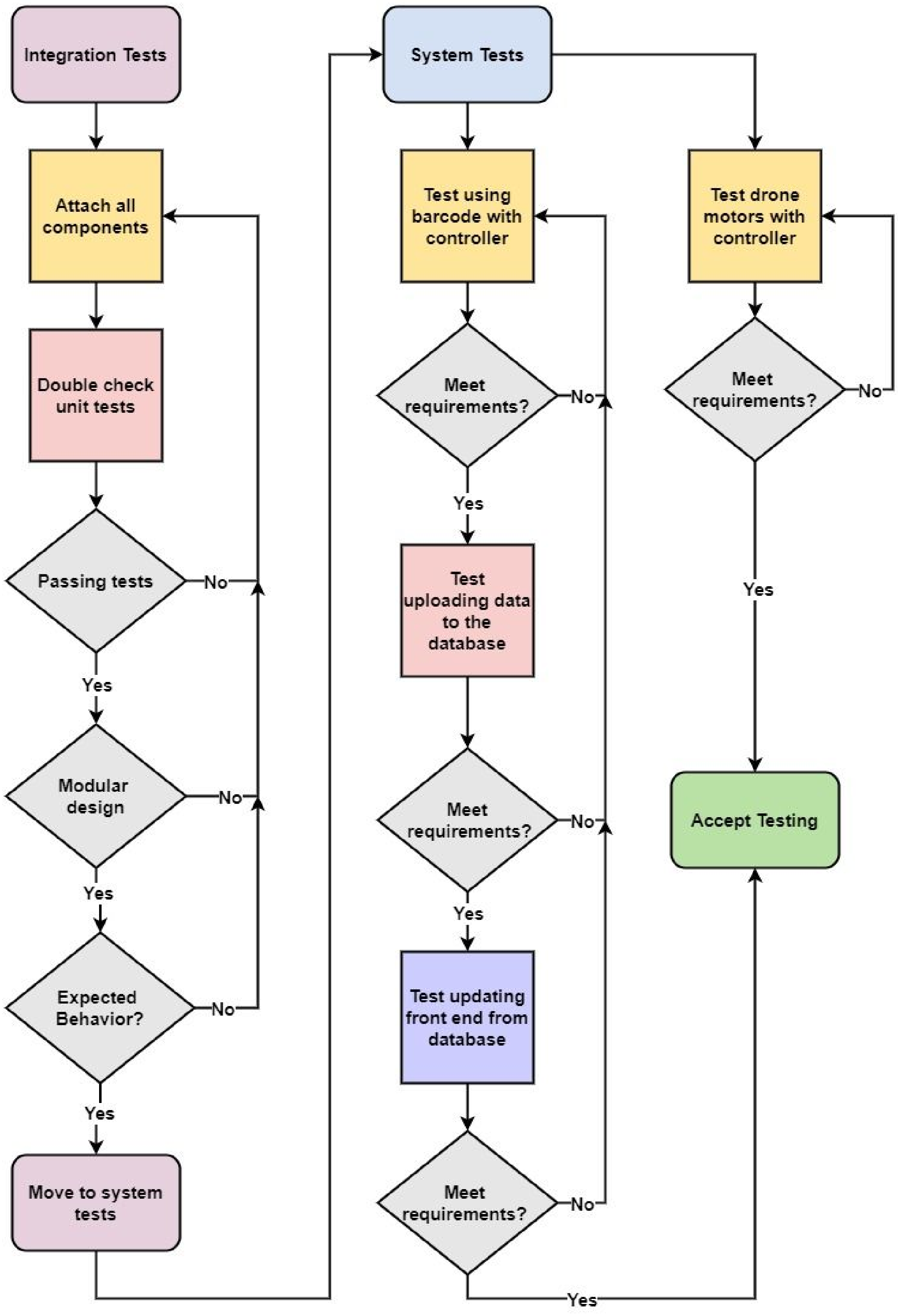


Figure 15: Integration & System Testing

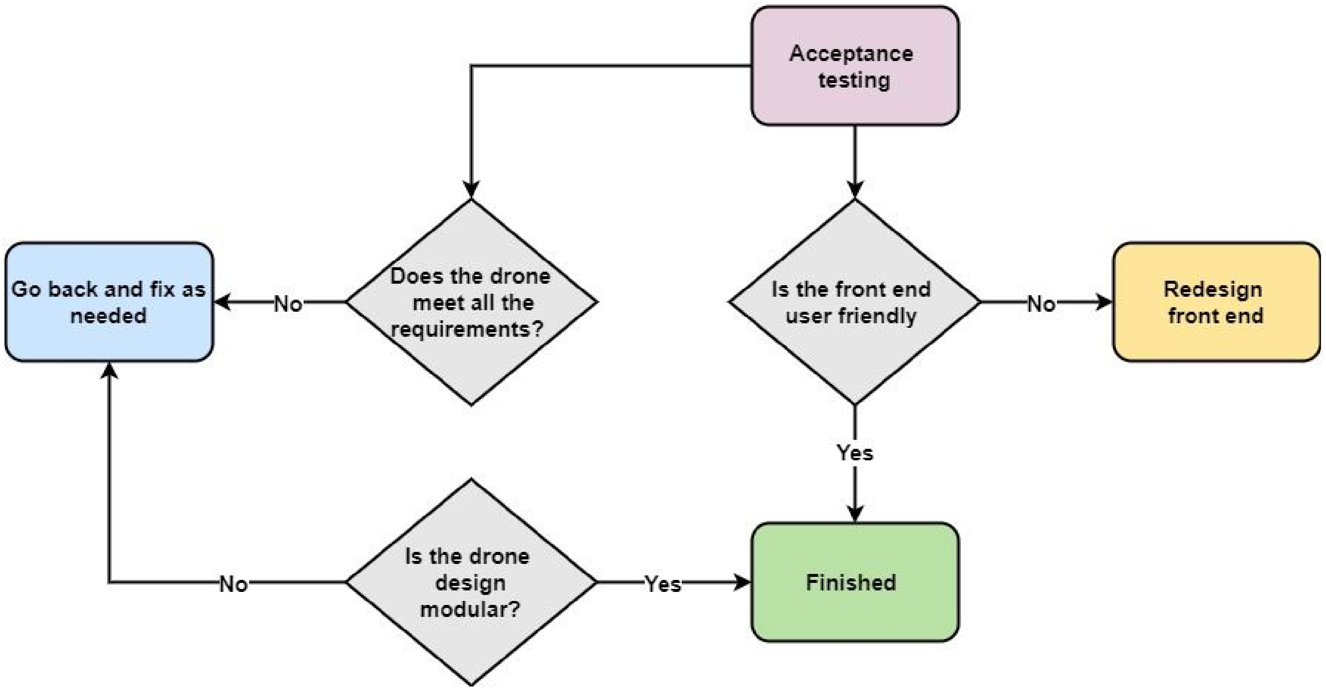

Acceptance testing
Does the drone meet all the requirements?
Is the front end user friendly
Go back and fix as needed
No
No
Redesign front end
Yes
Is the drone design modular?
No
Yes
Finished

Figure 16: Acceptance Testing

5.6 Results

- Testing QR code designs
  - Some QR codes that are generated cannot be scanned, so all generated QR codes must be scanned before being put into use
  - QR codes can be created with slightly over 2000 characters
  - QR codes can consistently be scanned with 500 or less characters
- Results of testing maximum distance Drone can successfully scan QR codes
  - Drone can pick up QR code from up to 1.7 meters away in a good photo
  - Drone can consistently pick up QR code from 1.4 meters away
- Testing light levels required to pick up QR code
  - Webots uses a luminosity level field that the team cannot find documentation for a direct translation to lumens for, so pictures will be included to give a feel of the brightness level
  - Webots can pick up the QR code at a minimum of .02 luminosity

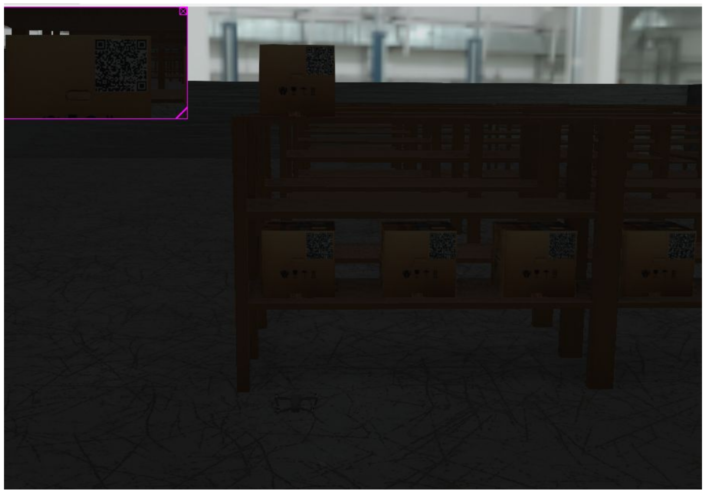


Figure 17: Warehouse light level .02 lumenosity

- Webots can pick up QR code consistently with light levels above .05 luminosity

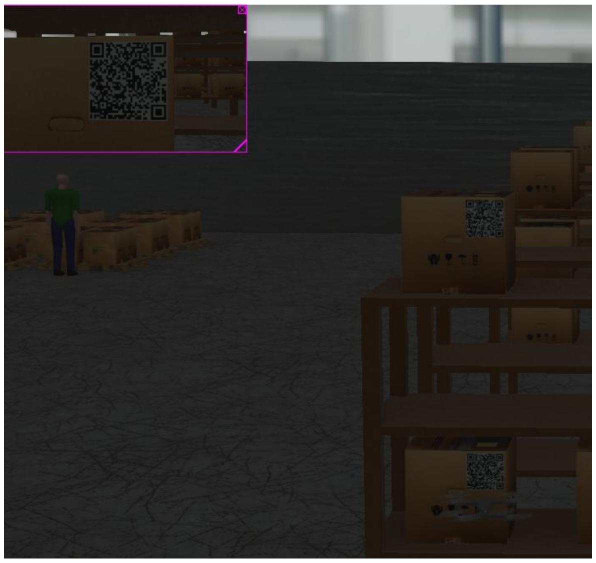


Figure 18: Warehouse light level .05 luminosity

- Testing Drone Automation
    - Drone eventually drifts off and must be reset, but it is functional and meets the team's needs and requirements.

6. Closing Material

6.1 Conclusion

The primary goal of our project is to increase warehouse efficiency through automation processes. To achieve this goal the team created a simulation of an aerial drone capable of taking inventory autonomously.

The original plan was for a fully autonomous drone, but after researching that plan and consulting with faculty we concluded that approach was not feasible due to circumstantial constraints. The new plan is to create a simulated drone and environment that uses partial automation due to constraints of both time and the simulation environment.

Our team believes that, while our project did not complete our original vision of a fully autonomous warehouse inventory solution, that the created project is a viable alternative for creating a more efficient warehouse environment.

Lessons Learned:

Throughout the year, our team was met with several setbacks. These setbacks were for a variety of reasons: complications due to COVID-19, issues with learning new technologies, or time management as individuals and as a team. The way our team overcame and dealt with these issues was by being flexible with the way our team thought about the solution to our project, consulting with our advisor for important decisions, as Dr. Rover had a large amount of valuable input, and made sure our team was in constant communication. The last one not only helped keep each other on track, but also allowed for those who had knowledge or experience in a certain area to assist others who might be having difficulties in that area. There is no sure way to know what setbacks a team will face, so the only solution is to be flexible and help each other.

6.2 References

[1] "Arduino drone brushless flight controller tutorial", *Electronoobs.com*, 2020. [Online]. Available: http://www.electronoobs.com/eng_robotica_tut9.php. [Accessed: 26- Apr- 2020]

[2] J. Brokking, "Brokking.net - Project YMFC-AL - The Arduino auto-level quadcopter - Home.", *Brokking.net*, 2020. [Online]. Available: http://www.brokking.net/ymfc-al_main.html. [Accessed: 26- Apr- 2020]

[3] H. Srivastava, "5 Best Free and Open Source Inventory Management Software", *Blog.capterra.com*, 2020. [Online]. Available: https://blog.capterra.com/the-top-5-free-inventory-software-systems/. [Accessed: 26- Apr- 2020]

[4] J. Schofield, "7 Important Inventory Metrics for Your Warehouse", *System ID Barcode System Blog*, 2020. [Online]. Available: http://www.systemid.com/learn/7-metrics-for-your-warehouse/. [Accessed: 26- Apr- 2020]

[5] S. Stone, "5 Key Warehouse Performance Metrics for an Effective Operation", *Cisco-eagle.com*, 2020. [Online]. Available: https://www.cisco-eagle.com/blog/2016/01/28/5-key-warehouse-performance-metrics-for-an-effective-operation/. [Accessed: 26- Apr- 2020]

[6] M. Lebied, "Inventory Metrics & KPIs – Best Practices Every Manager Should Know", *BI Blog | Data Visualization & Analytics Blog | datapine*, 2020. [Online]. Available: https://www.datapine.com/blog/inventory-metrics-and-kpi-best-practices/. [Accessed: 26- Apr- 2020]

[7] S. Cho, "10 Inventory Metrics You Need to Know - Inventory Management Metrics", Best Inventory Management Software for Small Businesses - EMERGE App, 2020. [Online]. Available: https://emergeapp.net/inventory-reports/10-inventory-metrics-for-smbs/. [Accessed: 26- Apr- 2020]

[8] J. Brokking, "Brokking.net - Project YMFC-AL - The Arduino auto-level quadcopter - Home.", *Brokking.net*, 2020. [Online]. Available: http://www.brokking.net/ymfc-al_main.html. [Accessed: 26- Apr- 2020]

[9] "Arduino radio controller NRF24", *Electronoobs.com*, 2020. [Online]. Available: http://www.electronoobs.com/eng_arduino_tut25_2.php. [Accessed: 26- Apr- 2020]

[10] "DIY Arduino RC Transmitter - HowToMechatronics", *HowToMechatronics*, 2020. [Online]. Available: https://howtomechatronics.com/projects/diy-arduino-rc-transmitter/. [Accessed: 26- Apr- 2020]

[11] "DIY Arduino RC Receiver for RC Models and Arduino Projects", *HowToMechatronics*, 2020. [Online]. Available: https://howtomechatronics.com/projects/diy-arduino-rc-receiver/. [Accessed: 26- Apr- 2020]

[12] "Supply Chain Metrics for Data Driven Leaders", *Klipfolio.com*, 2020. [Online]. Available: https://www.klipfolio.com/resources/kpi-examples/supply-chain. [Accessed: 26- Apr- 2020]

[13] "KPIs to Improve Inventory Management Process | USPS Delivers", *USPS Delivers*, 2020. [Online]. Available:
https://www.uspsdelivers.com/10-kpis-that-can-help-improve-your-inventory-management-process/. [Accessed: 26- Apr- 2020]

6.3 Appendices

Appendix I (Operation Manual):

Step-by-step instructions on how to setup/demo/test the system

- Download and set up Webots (For Windows)
    - Download Webots for your here: https://cyberbotics.com/
    - Download the google zebra crossing library here https://jar-download.com/artifacts/com.google.zxing Unizp it and add a CLASSPATH environment variable with the value of the path to the jar file in the Windows Environment Variables
    - Clone webots gitlab branch https://git.ece.iastate.edu/sd/sddec20-10.git
    - Readme describes where files inside should be placed, but A quick description will be provided here.
    - Protos should be placed inside of the Webots appdata location at ..\AppData\Local\Programs\Webots\projects\objects\factory\containers\protos\textures
    - There is a textures folder inside of the protos that should be combined with the existing textures folder
    - Wherever you create your environment with the world file (.wbt) there should be a controller folder. Inside of that folder, create a folder named mavicJavaTest and place the file called mavicJavaTest.java inside of there
    - Run Webots and under File -> Open World select Warehouse.wbt
    - Click on the drone by the shelves and on the left hand menu, edit the controller for the drone.
    - The mavicJavaTest.java file should pop up. Press on the gear wheel to compile that file
    - Restart the world environment and Webots should be set up
- Maneuvering drone-
    - Use arrow keys to rotate left and right and move forward and backwards
    - Use Shift + Left or Shift + Right to strafe left and right.
    - Use Shift + Up or Shift + Down to change altitude.
    - Use Control + Right or Control + Left to Start automation
    - Use Control + Up to cancel Automation
    - Use Control + Down to scan QR code
- Drone Automation
    - Drone must be lined up with a QR code at the end of a row of 6. Automation code was made specific to this warehouse environment so it knows where the gaps are and how long they are.
    - Drone starts by scanning the box in front of it, so there is no need to scan that box before starting automation.
- Install and run web application
    - Clone gitlab: https://git.ece.iastate.edu/sd/sddec20-10.git
    - Open a terminal and navigate to where you cloned the repository, then go to the "webapp" folder (the root of the project).
    - Run the following commands in the order they are listed:
        - npm install   (installs root dependencies)

        - cd client && npm install   (installs client dependencies)
        - cd server && npm install   (installs server dependencies)
        - npm run client-dev   (start the client)
        - npm run server-dev   (start the server)
    - Sometimes terminals don't like combining commands like in steps b and c, if so, do this instead:
        - cd client   (or cd server)
        - npm install   (installs dependencies in either client or server)
        - cd ..   (return to "webapp" aka the root directory)
    - Open http://localhost:5001/ in a browser
- Webapp Demo/Testing
    - After starting the client your browser will open http://localhost:5001/login
    - After logging in you will be brought to the dashboard homepage
        - A user account is not currently necessary to use the webapp, simply change "/login" to "/dashboard" in the url to get to the home page
    - From the home page you can access all the other pages (inventory, metric, fleet management).
    - The table on the inventory page contains the information for all the items in the database.
        - Each column in the table can be sorted in ascending or descending order by clicking the arrow at the top of that column
        - Each column can be filtered by entering a string into the text box at the top of that column
        - The entire table can be filtered in a similar way as the column filtering by entering a string in the text box located just above the table.
    - Smoke Testing: To test that the software is functioning correctly we test the functionality of each page
        1. Start at the homepage
        2. Click "Go to Inventory"
        3. Check that the inventory table is populated with item data
            a. If there are no items there is a problem with the client communicating with the server
            b. Optionality, scan a box in webots or send a POST request to the server to update the item database with a new item. Then, refresh the inventory page to check that the newly added item is the first entry in the table.
        4. Test sorting and filtering functionality of the table
        5. Click "Metrics" located at the top of the page in the navigation bar
        6. Check that the graphs are displaying data
        7. Click "Fleet" located in the navigation bar
        8. Check that drone information is being displayed
            a. Optionally, scan a box with the drone in webots and you should see its status change in the UI

Appendix II (Initial Design Version):

Other Resource Requirements:

- A room large enough to fly a drone
  - Dr. Rover mentioned she could put in a request to allocate space for our team somewhere.
- Shelves
  - Shelves are needed to simulate a warehouse environment and put goods at different elevations
- Boxes
  - Cardboard boxes or something similar to place barcodes on.
- Barcode printer
  - We will need this to print barcodes to test the scanner. We would like to check in with existing ISU resources to see if one would be available for use.

Previous Cost Analysis:

The team laid out the items needed to complete our project in a spreadsheet listed here:

Drone Cost Analysis

Index	Item	Cost/Item	Shipping	Quantity	Total	Link	Purpose
1	Drone Frame	$19.99	$0.00	1	$19.99	Frame	To mount components on
2	Propeller Motor	$15.99	$0.00	4	63.99	Propellers	For thrust
3	Gyroscope Sensor	$10.99	$0.00	1	$10.99	Sensor	For Orientation and balancing
4	Lipo Battery	$33.99	$0.00	2	$67.98	Battery	Powering the drone
5	Lipo Battery Charger	$38.99	$0.00	1	$38.99	Charger	Charging the batteries
6	Miscellaneous	$50.00	$0.00	1	$50.00		Wiring, pcb, transistors, capacitors, heat, heat shrink, etc.
7	Arduino Nano		$0.00		0		Borrow from ETG
8	Scanner	$33.00	$0.00	1	$33.00	Scanner	Scanning barcodes

Figure 19: Drone Cost Analysis

RC Controller Cost Analysis

Index	Item	Cost/Item	Shipping	Quantity	Total	Link	Purpose
1	Transmitter	$10.99	$0.00	1	$0	Transmitter	Transmitting data to the receiver
2	Receiver	$5.99	$0.00	1	$5.99	Receiver	Receiving data from the transmitter
3	2 Joystick	$9.99	$0.00	1	$9.99	Joysticks	Controlling the drone

4	Toggle Switches	$8.56	$0.00	1	$8.56	Switches	Switching between radio channels
5	x4 9V battery	$7.48	$0.00	1	$7.48	Battery	Powering the controller
6	Arduino Nano		0		0		Borrow from ETG
7	Miscellaneous	$30	$0.00	1	$30		Wiring, pcb, transistors, capacitors, heat, heat shrink, etc.
8	Battery Connector	$4.99	$0.00	1	$4.99	Connector	Connecting to Arduino

Figure 20: RC Controller Cost Analysis

Item Breakdown:

Item	Total Cost
Drone	$285
RC Controller	$78.00
Budget	**$500.00**
Remaining	$137

Figure 21: Cost Breakdown by Item

As you can see the team is under budget and plans to use remaining funds to replace parts that might break during construction or testing phases if need be.

Hardware Testing:

Hardware used in the testing phase will consist of:

- The drone and its respective components
- The controller and its respective components.
- The barcode scanner

Individual tests will be performed checking the motor controllers and motors to confirm each one is functioning properly. Then the remote channels will be tested to make sure the Arduino controller is receiving a signal. The barcode scanner needs to be activated when a channel switch is toggled.

Software Testing:

Functional Testing:

- Initially, the battery will be tested to make sure it works. Then the motors will be hooked up to the battery to confirm each of the motors work.
- Separately the flight controller (the Arduino board) will be hooked up to confirm that works and run unit tests on that with the barcode scanner. This includes a test to confirm the board can receive a signal from the controller.

- Then the flight controller and motors will be hooked up to make sure the whole system is functional.

Non-Functional Testing:

- While testing the motors of the drone, the team must ensure that the motors will be able to lift the drone and hover with no more than half throttle. The purpose of this is to not burn out the motors and ensure longevity of the drone. The drone may also have more equipment attached in the future and the drone must be able to carry that as well.
- During initial flight tests, the drone will be tested using a tether to safely ensure flight capabilities.
- The Drone will also be tested for general usability and if the control setup is easy and intuitive.

Limitations

- A single drone shall operate for a minimum of 20 minutes on a single charge
- The drone shall be no larger than 300 mm in length
- The drone shall have a swappable battery
- The drone shall use a lithium-ion battery

Hardware-Software Relation

<u>**Relation Between Hardware and Software Overview**</u>

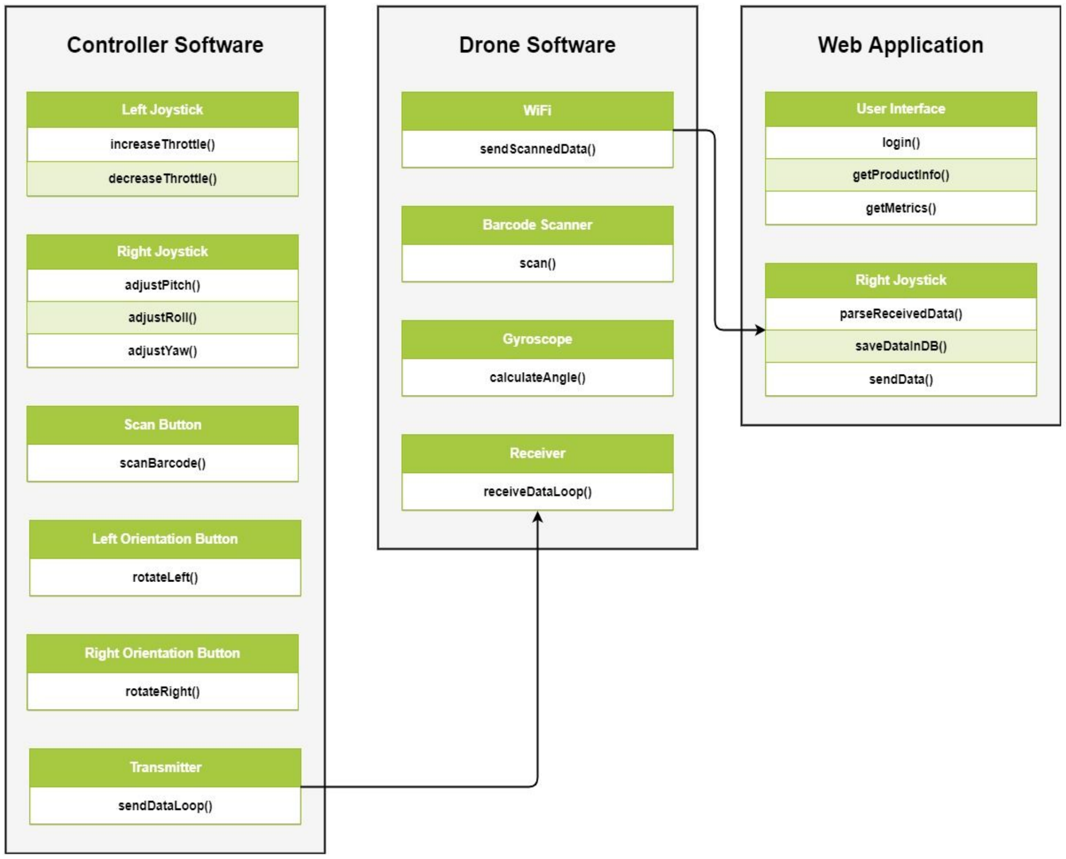


Figure 22: Hardware-Software Relation

Appendix III (Other Considerations):

Considerations for further development:

- Using multiple warehouses
    - To use multiple warehouses the web application and data sent there will have to be updated to reflect those changes
- Using multiple drones
    - The drone controller is bound to the keyboard. Within the simulate a different library would have to be used to inform the drones not to respond to the same controller
    - In the real world each drone would probably have a different controller so this wouldn’t be an issue
- Full autonomy
    - More sensors would need to be added to ensure consistency of drone flight throughout autonomy. An example of this would be using IR sensors as a homing beacon so the drone knows where to fly to or how to orient itself.
    - Full autonomy can be as complicated as you want it. A group would have to determine what level of autonomy they want to achieve and would be viable in the real world