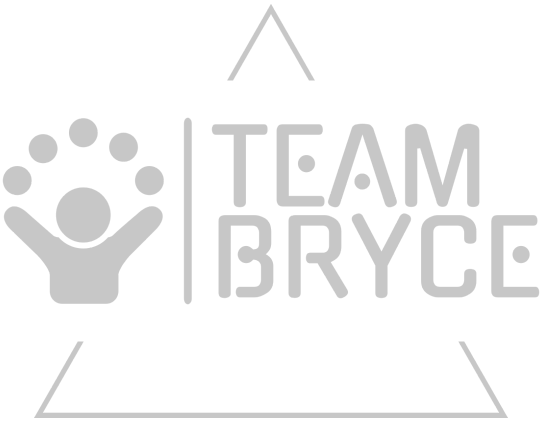
MICHIGAN ENGINEERING
UNIVERSITY OF MICHIGAN

Warehouse Robot Final Report



Submitted April 18, 2023

Team Bryce
1440 Hubbard St
Ann Arbor, 48109

Naila Garcia
Elliot Kupchik
Samit Mohapatra
Haytham Tang
Product Engineers

Submitted in response to SoW #ENGR100-850-W23

Table of Contents

I. Executive Summary 2

II. Introduction 3

Problem Statement

Full-scale Warehouse Robot Constraints and Goals

Prototype Comparison to Full-Scale

III. Stakeholder Considerations 4

Impact on Warehouse Workers

Safety Considerations

Design Input

IV. Design & Development 5

Gantry (XY Motion)

Scissor System (Z Motion) 7

Claw System (Payload Grip) 8

Power Design of Mechanical Systems

Fully Integrated Mechanical Systems of Prototype 9

User-Interface Software

Computer-Arduino Serial Connection 14

Data Storage System 16

V. Vision Of The Full-Scale System 17

VI. Conclusions and Recommendations 18

VII. Appendices 19

Appendix A. UI Software Code

Appendix B. Stakeholder Memo References

I. Executive Summary

With the increase in demand for rapid delivery of goods purchased online, high-efficiency warehouses have popped up all over the world. As a result, the demand for manual labor has increased in the industrial sector. Unfortunately, the current labor market can not fulfill this demand, so warehouses remain understaffed and employees are overworked. Over the duration of a long shift, manual labor is unable to consistently maintain quick and precise distribution of heavy items for warehouses. Our team aims to improve working conditions and warehouse efficiency with a system capable of moving heavy payloads. This final report details the design and specifications of the prototype.

The robot consists of three mechanical subsystems: the gantry frame, a scissor lift mechanism, and a claw mechanism.

The gantry frame was repurposed from a laser cutter and consists of a 2D gantry supported with steel bars. The scissor and claw mechanisms hang from the frame, their motion in the X and Y directions driven by two stepper motors.

The scissor lift mechanism consists of crossed straight plastic pieces for movement and other parts for support. Lego parts were used as a scaled-down material alternative to the metals that would be used in an industrial setting. The bars of the scissor lift are constrained within side panels to prevent twisting of the bars from the axle to reduce the strain on the servo responsible for lifting.

An aluminum claw was acquired and attached to a servo. One arm of the claw with one gear is rotated by the servo, causing the other arm and gear to rotate too. This ensures that when the claw pieces move in one direction, the other claw piece moves with the same force in the opposite direction.

All of the motors in the mechanical subsystems were controlled with an Arduino Leonardo microcontroller and motor drivers.

Included with the robot are a custom user interface, data transmission software, and data storage software. The prototyped user interface, made in Processing, is a 5x5 grid representing the warehouse inventory with buttons for item reallocation, retrieval, and storage. Data transmission occurs through data packets sent between the computer and arduino. Storage information is saved in structs via electrically erasable programmable read-only memory (EEPROM). For the first iteration of the design, each part was independently developed.

A full scale implementation would increase the customizability and durability of the robot. Better materials would be used and stress testing would be performed to determine the optimum setup. The software would allow for varying grid sizes, the movement of multiple items in one command, and automatic sorting. Finally, better safety features, like emergency buttons would be implemented to make the robot warehouse-ready.

II. Introduction

Problem Statement

With the rise of online shopping through retail sites and shipping magnates like Amazon, UPS, and FedEx, warehouses for package management have sprung up globally. Consequently, a pressing need for warehouse floor workers and managers has developed. Unfortunately, there are only so many employees willing to work awkward hours at a breakneck pace. Thus, there is always a shortage of labor, forcing workers to push themselves harder to get the job done. Many end up fatigued or injured.

Full-scale Warehouse Robot Constraints and Goals

In order to reduce the physical stress of warehouse operations, we have designed an automated package sorting system to assist with inventory management and carrying heavy objects across the warehouse. This gantry-based robot system will be capable of safely carrying upwards of 150 kg of boxes at three meters per second and will span the entirety of the warehouse's floor. A user interface with customizable features will be included to set each robot's specifications from a central or local location, allowing managers to adapt to any changes within the warehouse.

This design is optimal for the safety and efficiency of carrying boxes. First, the robot only operates within its own frame to prevent any unpredictable motion that could injure workers operating near the robot. Second, the minimal hardware combined with the programmability of the robot results in consistency in various tasks. Finally, the size of the robot allows for multiple to be used in various areas of the same warehouse, doing tasks in parallel.

Prototype Comparison to Full-Scale

To show our design's potential capabilities, we have built a small-scale functional system that reflects the important attributes of the final product. It includes a 616 x 212 x 557 mm 2D gantry system repurposed from a laser cutter, a claw system constructed from Lego pieces, and fully developed Arduino software for the user interface and the Arduino-computer communication. This prototype is supposed to be capable of carrying a maximum of three kilograms, and numerous payload safety tests will be performed to ensure this.

III. Stakeholder Considerations

The problem that our robot aims to solve is one that impacts several different parties involved in the daily functions of a warehouse. With a goal of ethical and efficient design, research was conducted to scope the impact that our robot would have, both on those directly and indirectly affected. We interviewed experts in different fields including an IOE professor, a Physics professor, and the director at Bursley residence hall. In addition, we read peer-reviewed articles that explained the features of existing robots as well as concerns with their use.

Impact on Warehouse Workers

Warehouse workers are one of our most directly affected stakeholders. Due to our design, they might either lose their job, be required to take over the operation of the product, or end up working near the robot. Because of this potential impact, we decided to investigate how the robot would affect them mentally and physically. The interviews we conducted revealed that workers would appreciate more automated help in order to decrease their workload but also that there are concerns regarding robot proximity and errors in possibly unsound new technology. The research articles that we consulted corroborated this information, detailing how both the efficiency of the warehouse and workers' concerns about unemployment increase when autonomous robots are implemented. However, from what we gathered, the stress from being overworked can be worse than or equivalent to the stress of possible unemployment and robot proximity. The interviewees and other research articles shed light on the question of how we can alleviate possible strains that the robot could cause on the warehouse workers. One of the ways we can do such a thing is to implement safety precautions and protocols; we could install emergency stop buttons for example. (See Appendix B: 1, 2, and 3)

Safety Concerns

Ensuring the safe operation of the robot plays a pivotal role in the design. Thus, IOE professor, Don Chaffin, with expertise in the health of workers in industries, has provided advice regarding how to operate the robot safely. He pointed out several ways of ensuring the safety of workers. For example, he suggested confining the area and making sure that no one is allowed within the safety perimeter. Another approach is to apply object/movement detection on the robot so it can pause or move in another path to avoid contacting people when there are workers nearby. As our final design involves the gantry being attached to storage racks, we decided to cordon off the area where the gantry will operate and incorporate protocols for when human interaction with the gantry is needed.

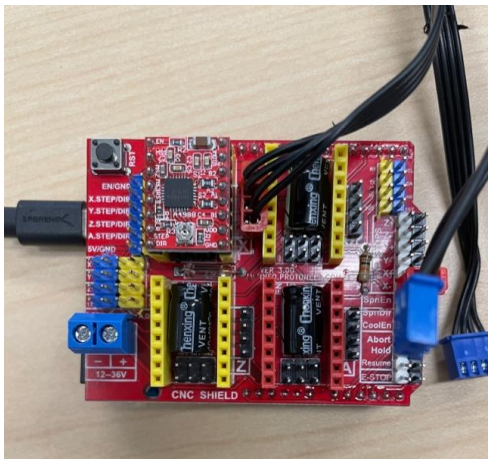
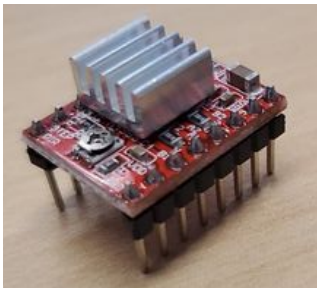
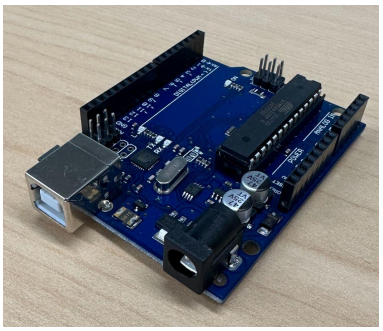
IV. Design & Development

Gantry (XY Motion)

The gantry system consists of a repurposed Kentoktool laser cutter connected to a programmed Arduino Leonardo microcontroller through motor controllers. To get the required height for the claw to fit in the frame of the gantry, metal supports are installed in each corner, bringing the height up to half a meter. The laser system was taken off the gantry to free up the mount for the claw. As for movement, by setting the turn angles of the motors with the Arduino Java function write(), we can move the claw to a specific XY coordinate to pick up or drop off a package. After every command, the gantry would return to the origin to keep any offsets from occurring.

Table 1: Gantry Materials

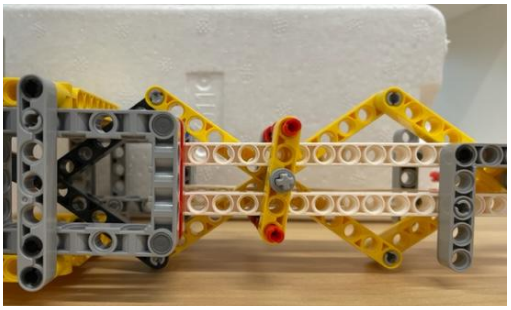
Kentoktool Laser Cutter	
Metal Support Beams x4 (0.5 m)	

Motor Controller	
Motor Driver	
Arduino Leonardo	
Dupont Wires (Jump Wires)	

Scissor System (Z Motion)

The scissor lift mechanism consists of crossed straight plastic pieces for movement and other parts for support. Lego parts were used as a scaled-down material alternative to the metals that would be used in an industrial setting. The bars of the scissor lift are constrained within side panels to prevent twisting of the bars from the axle to reduce the strain on the servo responsible for lifting. In order to increase the range of motion that the lift is capable of, a lego bar was attached to the servo arm, increasing its reach. A string was tied from the tip of the extended arm to the bottom of the lift, so that when the servo fully rotates, the mechanism lifts the claw around 8-10 cm.

Table 2: Scissor System Materials

Lego Scissor System Body	
MG 996R Digital High Torque Servo	

Claw System (Payload Grip)

The claw system is repurposed from a geared metal claw mechanism found in-shop. The claw is attached to a servo motor which controls the gripping motion by rotating one of the gears. The other gear rotates as well due to the connection to the other gear and the claw opens/closes. Finally, the claw is attached to the scissor system to provide it with Z motion. Modifications would include attaching cut neoprene to the gripper tips to decrease the pinpoint force while maintaining the friction force to prevent slippage.

Table 3: Claw Mechanism

Metal Gear (Solid) Claw	
MG 996R Servo	

Power Design of Mechanical System

The prototype design is primarily powered by a 24V generator which directly connects to all of the motors. Depending on the situation, the Arduino might receive power from the generator using a voltage regulator or a separate 5V source. The Arduino then receives electrical signals as commands from the computer and sends out its own commands to all of the motors.

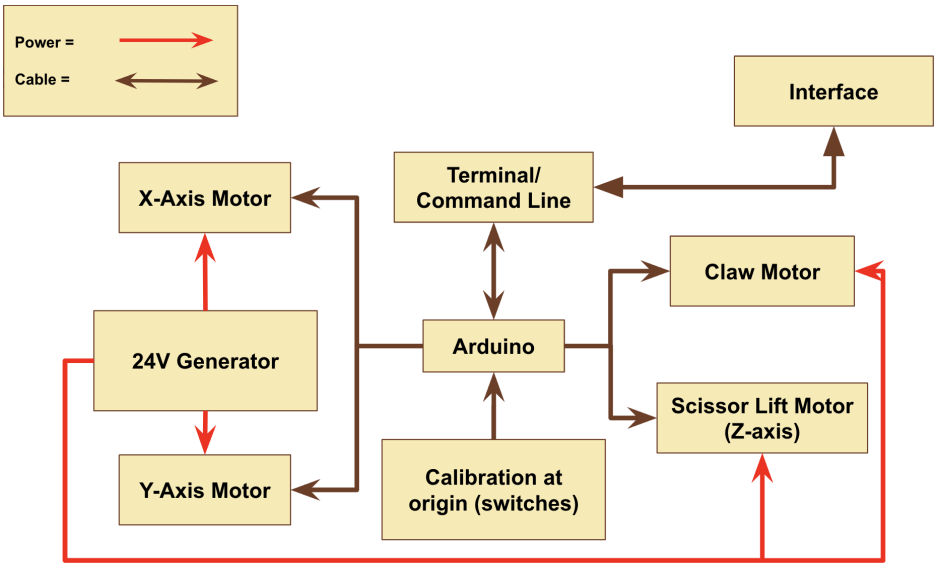


Figure 1: *Power Design of All Mechanical Systems Together*

Integrated Mechanical System

The fully integrated mechanical system contains the gantry frame, the scissor lift, the claw, the Arduino, the motor controllers, and the 24V source. Simple commands can be run on the system to move all of the motors.

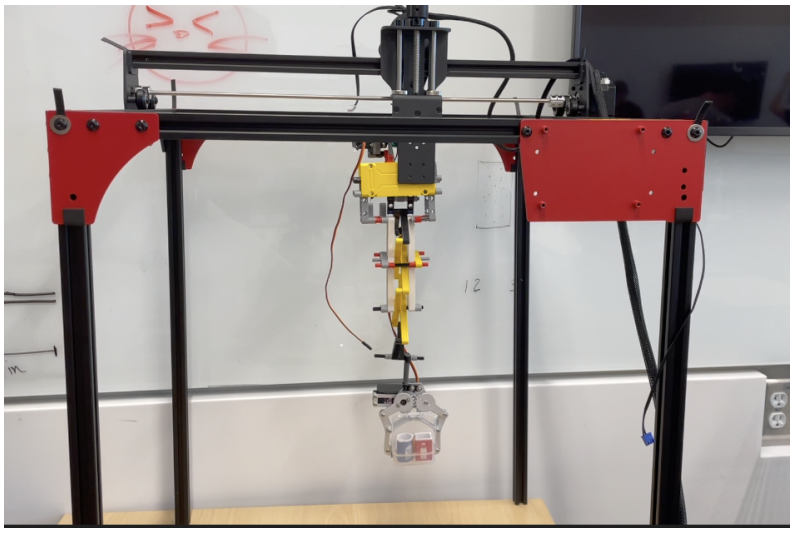


Figure 2: *Fully Integrated Mechanical Systems of Prototype*

User-Interface Software

We designed a friendly user-interface application using Processing software that allows the user to command and interact with the robot. The user gives their input mostly by clicking options on the screen. The default screen is composed of a 5×5 black and gray grid board that represents a $40\ cm \times 40\ cm$ map of the warehouse. Each cell of the grid is $8\ cm \times 8\ cm$ based on our prototype gantry system.

When the user starts the application, the screen will exhibit the grid board as well as three different options for modes: Reallocation, Retrieval, and Storing. The program will prompt the user to select a mode. The following is the figure of the starting status of the program. There are three buttons for the modes, and each has a daffodil background color and blue text colors. The program has an “enum” class that stores the selected mode. After the user clicks the a button, the program will clear the page and present a new layout. The page will remain still if the user does not click any button.

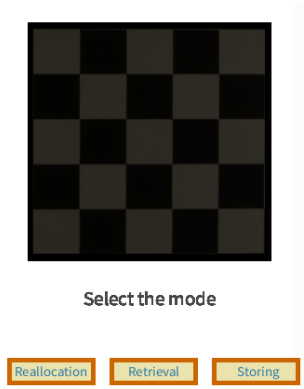


Figure 3: *Starting Status of U.I. Software*

Reallocation Mode:

Reallocation Mode is selected when the user wants to move an existing object in the warehouse to another position. As the following figure shows, if the user picks Reallocation Mode, the program will prompt the user to pick the starting position, which is the location of the object they want to relocate. The user selects this position by clicking a corresponding grid on the screen.

Figure 4: *Page Prompting the User to Pick From Position*

After the user selects a valid position, the grid will be highlighted blue. Then, the program will create a confirmation message that asks the user to confirm the choice. The user just needs to click the "YES" or "NO" button in the figure shown below.

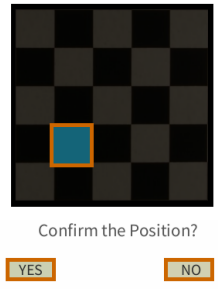


Figure 5: *Confirmation Page for From Position*

If the user clicks “NO”, the program will clear the grid as well as the confirmation message. It will return to the stage asking the user to pick a “from” position. If the user clicks “YES”, the grid will be locked and can not be selected again. Then the program will prompt the user to click the destination position, where the object will be reallocated to. This procedure is shown in the figure below, which is very similar to the step of picking the “from” position. The user again selects another grid for the destination position.

Figure 6: *Page Prompting the User to Pick Destination Position*

If the user chooses a destination position that has already been selected as the “from” position, the terminal will print out the message “Please select another position”, clearly asking the user to pick a different position.

```
Now Choose the Destination
Please select another position
Please select another position
Please select another position
Please select another position
Please select another position
```

Figure 7: *Terminal Output for Invalid Selection*

After the user has selected a valid destination position, the destination grid will be highlighted in green. The program will generate the confirmation page again to prompt the user to confirm their choice. As Figure 8 below shows, the user will again be required to click the “YES” or “NO” button to confirm their selection. If the user clicks the “NO” button, the program will clear the selected destination grid and go back to the previous step, asking the user to pick the destination again. If the user clicks the “YES” button, the program will store the two positions and continue. It will first calculate the horizontal and vertical distances between the origin (the bottom left grid) and the “from” position. Then, it will calculate the distance between the “from” position and the destination position. Finally, it will calculate the distance between the origin and the

destination position. The calculated distances are used to command the gantry to move accordingly.

Figure 8: *Confirmation Page for Destination Position*

Retrieval Mode:

The retrieval mode is selected when the user wants to fetch an object from a specific location. The grabber will move to the particular location confirmed by the user, pick the object up, and return to the origin to drop off the object.

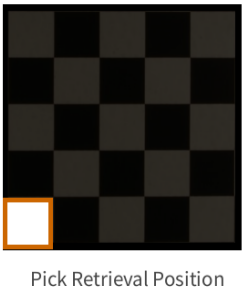


Figure 9:

Page Prompting the User to Pick Retrieval Position with White Grid Marking the Location of the Origin

Once the user has selected a valid retrieval position, the grid will be highlighted with green color. Next, the program will create a confirmation message and ask the user to confirm their choice. The user will click either the “YES” or “NO” button on the screen as Figure 10 displays. If the user clicks the “NO” button, the green grid will be cleared and the program will go back to the last step to ask the user for a retrieval position. If the user clicks the “YES” button, the program will store the retrieval position and calculate the horizontal and vertical distances between the origin and the retrieval location. Finally, it commands the gantry motors to move to the retrieval location, pick up the object, and return it to the origin.

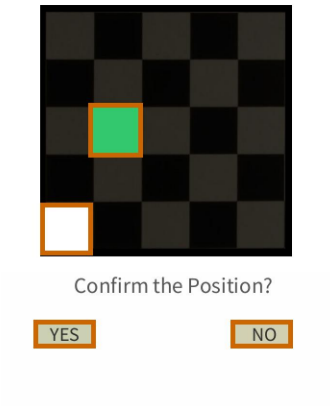


Figure 10: *Confirmation Page for Retrieval Position*

Storage Mode:

To run the storage mode, the grabber will pick the object at the origin, move to the location specified by the user, drop off the object, and return to the origin. When the user selects the storing mode, the program will prompt the user to select the position at which they want to place the object.

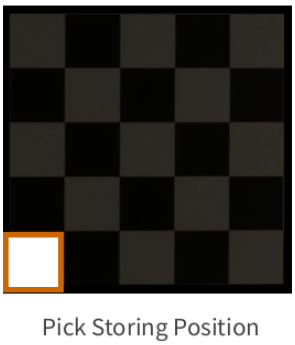


Figure 11: *Page Prompting the User to Pick Storing Position*

After the user has selected a valid grid position, it will be highlighted with blue color. The program will then create a confirmation message that asks the user to confirm their choice. The user needs to either click the “YES” or “NO” button on the screen shown in Figure 12 below. If the user clicks the “NO” button, the program will clear the blue grid and ask again for a storage position. If the user clicks the “YES” button, the program will save the selected position and calculate the horizontal and vertical distances between the origin and the storage location. Finally, it commands the gantry to move to the storage location, drop off the object, and return to the origin for recalibration.

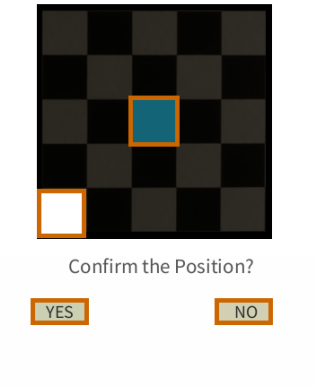


Figure 12: *Confirmation Page for Storing Position*

Computer-Arduino Serial Connection

To ensure a stable connection between the computer (sender) and the gantry system's main Arduino (receiver), our sending system will first encode the user's commands as a string of bytes, which are transmitted over a serial port. Each segment of bytes is used to indicate different data values, such as the coordinates of a requested cell from which the user wants to retrieve an item. Special char values are placed between data segments, which allows the receiver to determine what each data segment represents.

When the Arduino receives an encoded message from the computer, it will read through every byte and utilize the preset formatting rules of the data packet to determine if the message has been corrupted in any way. If the Arduino identifies any issues with the integrity of a message, the message will be ignored and a re-request for the string of bytes will be sent back to the computer.

This process will take place between any two sending and receiving processing components and ensures that the robot does not malfunction in environments that can affect the integrity of the gantry robot's electronic components and connections. Below are illustrations of the system.

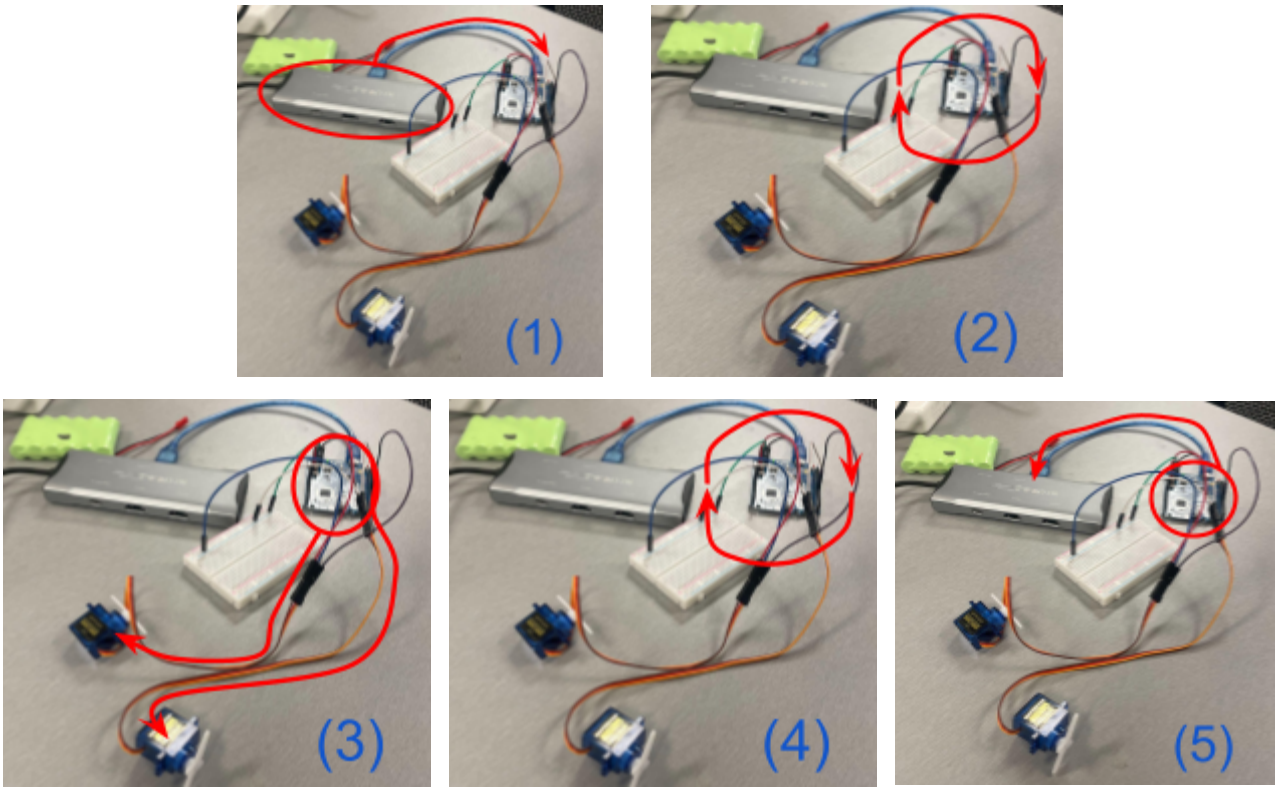


Figure 13: *(1) Initial Data Transmission from COM to Arduino (2) Arduino Internally Processes Commands Given by COM (3) Arduino Sends Signals to Auxiliary Servos to execute COM instructions (4) Arduino Internally Counts Steps Taken for Operations until Execution Completes or Interrupted for Emergency Stop (5) Confirming Successful Operation from Arduino to COM via Serial Connection*

Data Storage System

Over the course of a warehouse's daily operations, the gantry robot will keep an up-to-date log of all items contained within the warehouse's gridspace. This will be done through the usage of electrically erasable programmable read-only memory (EEPROM), flash memory storage, and retrieval protocols, where data structs will be created to store the contents of individual cells on a cell array of Arduino structs.

After a command is sent from the computer to the Arduino, the Arduino will parse through the elements of the cell array placed in its long-term storage. Upon doing so, it will save the data of the state and contents for each cell in active memory with separate 2D cell arrays, which can be quickly modified as new commands come in from the computer. Once the user finishes sending commands, the Arduino will save changes to data values by updating the structs that correspond to modified cells.

To reiterate, the changes made by a user to the position and contents of warehouse cells will be stored by updating the structs stored in the system via EEPROM. This ensures that the state of the warehouse's storage contents will be correctly stored by the Arduino after a usage session.

V. Vision of the Full-Scale System

Because of the limited amount of time and budget, we are unable to create a full-scale robot that would perform exactly as we expected it to. As a result, we have included the following important features for our full-scale system.

Gantry & Claw

The full-scale system will utilize heavy-duty motors for x and y direction motion. This provides high levels of torque for our gantry to move and also maintains a reasonable speed for factory operation. We will apply reinforced metal beams to support the gantry, increasing its payload capacity. We will upgrade the cable-carrying sheath to prevent pinch points or tangling, ensuring that our robot can operate smoothly. Instead of using Lego pieces for our scissor system, we will employ more flexible and resilient materials that can safely stretch to provide a greater claw range of motion. Lastly, the full-scale system will have custom-made claws with different shapes to adapt to specific items in a warehouse.

Full-Scale U.I. System

We will also upgrade the U.I. software to run on a mobile/web application so that it can be directly downloaded for operation. Instead of fixing the grid board to be 5×5, the U.I. software will provide the option of adjustable internal grid dimensions. This allows the robot to be customized to the specifications of the warehouse. In addition, the full-scale U.I. will have the cell-grouping option, allowing individual items to take up multiple cells. It will also allow the user to select multiple objects of the same type and reallocate/retrieve/store them all in one command. Finally, we will display the data storage information about each grid and object along the side of the grid.

VI. Conclusions and Recommendations

The primary goal of this project was the development of an automated system that alleviates the intense workload on package shipping center workers. From our research, we determined that the most important requirements for this system are sufficient payload capacity, payload safety, consistency, and a user-friendly interface. Thus, we included a 2D gantry with stepper motors for consistent mobility and metal supports for payload weight redistribution, a padded extending claw mechanism for payload safety, and an intuitive UI design for easy operation. As the prototype does not have the exact capabilities of the full-scale system, there will be modifications to the industry-ready robot that are not present in the model. The gantry will be made with reinforced steel, the claw system will use a linear actuator, and the data storage system will be much larger to account for the warehouse size.

Any potential improvements to the design might include better materials for physical support, automatic sorting for boxes without worker input, and higher-quality motors for faster delivery. It is especially important for the claw to be large and resilient enough to carry multiple objects stably. Stress testing may be done to determine what parts of the gantry frame need the most support. Even with these modifications, however, a warehouse only runs as fast as its slowest process. Decreasing sorting times requires greater external transport capacity so that items do not pile up (faster truck shipments). The U.I. software can be implemented into a mobile or web application with greater flexibility so it can adapt to different sizes of warehouses. Improvements for the U.I. software should enable the user to select multiple objects of the same type and operate reallocation/retrieval/storage simultaneously. In addition, it will be straightforward for the user to check the status of the warehouse if the U.I. software can display the storage information of each cell.

If there is any confusion or recommendation for the design of this warehouse robot, feel free to reach us.

Naila Garcia: garnaila@umich.edu
Elliot Kupchik: ekupchik@umich.edu
Samit Mohapatra: samitm@umich.edu
Haytham Tang: yunxuant@umich.edu

VII. Appendices

Appendix A. UI Software

Link for Github Repo: https://github.com/haytham918/ENGR-100

Appendix B. Citations and References

1. Warehouse Robot Scoping Report by Haytham Tan
2. Warehouse Robot Scoping Report by Samit Mohapatra
3. Warehouse Organizing Robot Design Scope Report by Naila Garcia
4. Warehouse Robot Scoping Report by Elliot Kupchik