BCIT SCHOOL OF ENERGY

Mechatronics and Robotics Engineering Technology

ROBT 4491
Mechatronics Final Project

FINAL PROJECT REPORT:

AUTOMATED TREAT LABELER

Produced By:	
Wyatt Johnson	A01324595
Wendel Pena	A01038564
Zy Jarvis	A01323665
Date:	May 22, 2024
For:	Isaiah Regacho

ACKNOWLEDGMENTS

We must express our thanks to the Mechatronics and Robotics faculty for their support and encouragement throughout the project development: Isaiah Regacho, Andrew Friesen, and Edwin Cheung. Completion of the project would have not been accomplished without the guidance and support from our instructors and peers. As well as Open Range for supplying everything needed to complete the project.

ABSTRACT

This report is regarding the labeling automation project for the people at Open Range Natural Dog Treats by the second year Mechatronics and Robotics Diploma students. We plan on solving the problem of requiring an employee to hand feed and receive treats from your labeling machine to decrease risk of repeated stress injuries caused by repetition in the employee and have a reliable replacement. Our solution to this problem is to use camera vision to work in tandem with an ABB robot to grab a variety of treats in the work area.

CONTENTS

FIGURES

TABLES

1 INTRODUCTION

The project was created to automate the treat labeling process done by Open Range's dog treat labeler by eliminating the need for repetitive action of loading and unloading the labeler to be done by an employee. This document will split the project into 4 different subsystems: machine vision, the robot gripper, the ABB robot, and communication. It explains each subsystem and shows the design and implementation applied to each one.

2 SYSTEM OVERVIEW

The 21 Jump Treat Labeler is a proof-of-concept dog treat picking and placing Robot developed for the Open Range Pet Treat company. It labels dog treats by locating them using computer vision and picks them up with a 6 degree of freedom robot. The system starts with a worker that will spread treats onto a table with a uniform background in range of the robot, making sure the treats are spread out evenly. The camera is mounted above the table. The worker will attach the gripper for the treats and the robot process will start when the work area is cleared. Once activated, the robot will grab a single treat from the table and place it onto the labeler. The robot will then wait until the labeler is finished and retrieve the treat and drop it into a box to be packaged. The robot continues until all treats are picked and placed. The system will notify the worker once there are no remaining treats and the worker can safely remove the box of labeled treats. To restart the process with more treats, press the button and the robot will repeat the process.

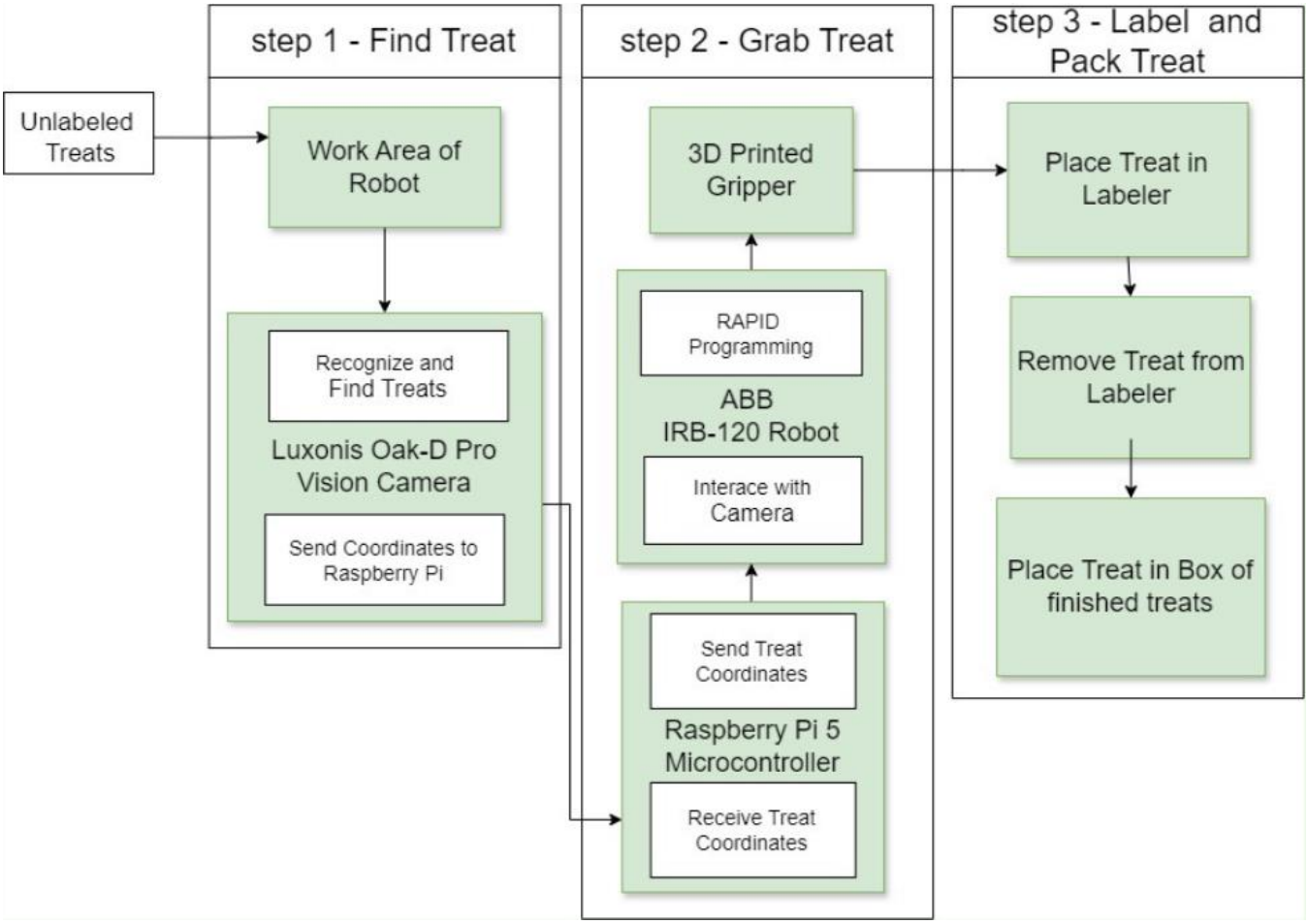

Figure 1: Treat Labeling System Overview

SYSTEM REQUIREMENTS

The requirements for operating the automated dog treat labelling system will be described in this section. We will highlight necessary hardware components and supplement amenities for the function of the system. All listed features are essential for the operation of the system by design:

Table 1: System Requirements

Requirement	**Detail**
Robot	ABB IRB-120 to manipulate the treat in space
Vision Camera	Luxonis OakD-Pro to detect the treat
Microcontroller	Raspberry Pi 5 to relay information to and from the robot and camera
Product	Open Range Dog Treats to be labelled
Workspace	Visualization of Robots Work Envelope to ensure treats are placed in range
Labeler	Industrial Labeler to label the treats
Parallel Gripper	MPG-plus 64
End Effector	3D Printed PLA Gripper to pick-up the treats
Employee	A Person to place the treats on the table
Box	A Box to place the labeled treats in

VISION

The vision subsystem is fixed to view the work area in front of the robot. The camera deploys a vision model that allows it to recognize treats and gather coordinates. It will locate the treats and send their coordinates to the microcontroller through USB 3. The microcontroller communicates with the camera to get it to continuously scan the table and return coordinates to the microcontroller. The camera coordinates are filtered and sent to the robot.

ROBOT GRIPPER

The robot gripper is attached to the ABB robot and is opened and closed by the pneumatics attached to the robot. The gripper is actuated by the robot to grab and carry treats from the table to the labeller, and finally to the box of finished treats.

ABB ROBOT

The ABB robot collects the information from the microcontroller when it is ready and moves to the position of the treat to pick it up and move it to the treat labeler. It waits for the treat to be labeled then moves it to labeled treats box.

COMMUNICATION

An ethernet communication system connects the Raspberry Pi 5 microcontroller and ABB IRB-120 robot together using a TCP/IP client-server model. This connection allows all position information to be shared from the Luxonis Oak-D Pro camera to the ABB robot.

3 VISION

The camera uses a vision model that can locate and differentiate between treats. The model is trained with a dataset of photos that are annotated to tell the model what is and is not a treat. The model is deployed to the camera from the Raspberry Pi. The Raspberry Pi communicates with the camera to extract the coordinates of the treats from the camera. The Raspberry Pi filters the coordinates where only treats in the range of the robot are sent to the robot.

VISION REQUIREMENTS

- A Raspberry Pi 5 microcontroller
- Luxonis Oak-D Pro camera
- Vision model
- A stand to hold the camera 150cm above the table
- An ethernet connection between the microcontroller and the robot
- USB 3 connection between camera and microcontroller
- Python
- Depth AI
- Roboflow Oak dependencies

VISION DESIGN:

HARDWARE DESIGN

The design of the hardware was very simple. The Oak-D Pro was connected to the Raspberry Pi through USB 3 to allow communication between the two and the Raspberry Pi was connected through ethernet to the robot to allow communication by TCP/IP.

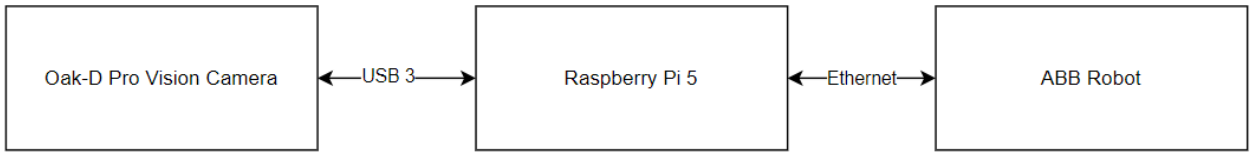

Figure 2 - Hardware Design Block Diagram

The figure below shows the schematic of the stand to hold the camera above the table. The stand needed to be above 150 cm to stay out of the maximum reach of the robot. After building the stand and running the robot it was discovered that the stand was very resonant. So, another beam was added in the middle to stabilize the stand and reduce the movement of the camera at the end of the stand.

Figure 3 - Camera Stand Schematic

SOFTWARE DESIGN

The figure below shows the process the machine vision system went through to gather treat coordinates for the robot to go to. The process starts by extracting the vision model from the Roboflow website and deploying it to the Oak-D Pro. Then establishing a connection with the robot through TCP/IP. The microcontroller then crops the camera view and filters the treat coordinates to only give coordinates to the robot that it can reach. Every time the microcontroller

goes through this process it checks to see if the robot has sent the okay string. If it has, coordinates are sent to the robot. If there is no okay string, the process continues and gets the most recent treat coordinates. The process will end if the user terminates the program.

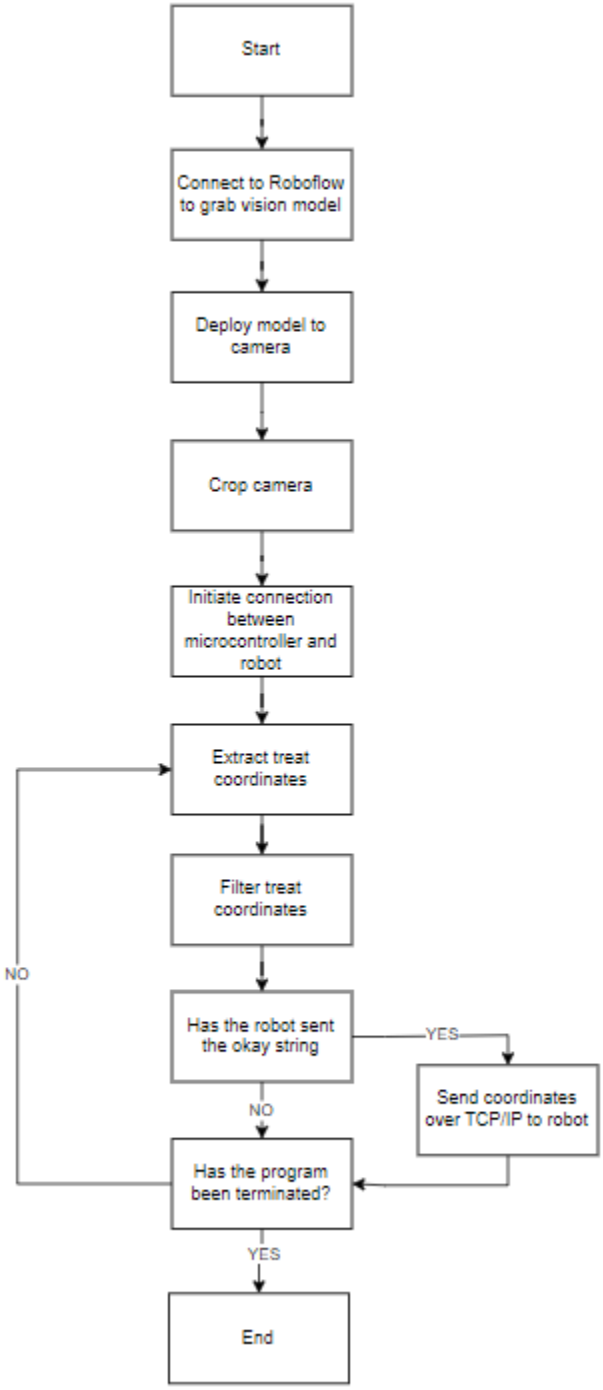

Figure 4 - Vision Software Flowchart

The program used these modules:

- Depth AI platform
- Roboflow Oak
- pySocket
- Threading

VISION IMPLEMENTATION:

SOFTWARE IMPLEMENTATION

The links below are to the project GitHub. The links provided are for the python file drafts and the modules used in the python files.

https://github.com/wdj1319/21JumpTreat/blob/main/machine_vision_python_files.zip

https://github.com/wdj1319/21JumpTreat/blob/main/machine_vision_modules.zip

4 ROBOT GRIPPER

The robot gripper end effector is tasked with grabbing various dog treats to work in tandem with a labeling machine. The gripper loads and unloads the treats onto the labeler, then drops the labeled treat into a finished box to be then packaged away. Treat coordinates are acquired and relayed to the ABB robot from the camera vision and raspberry pi 5. Overall, the implementation of this gripper will eliminate the need for a human worker to manually grab and label the dog treats.

ROBOT GRIPPER REQUIREMENTS

- 3d printer and PLA filament
- ABB IRB-120 Robot
- MPG-plus 64 Parallel Gripper
- Treat profiles
- SolidWorks software

TREAT TYPES

The gripper is tasked to grab the following Open Range dog treats (depicted left to right in photo):

- 12” Bully Stick
- 12” Chomper
- 6” Bully Stick
- 12” Bully Braid
- 12” Flat Chomper

Figure 5: Open Range Dog Treats

ROBOT GRIPPER DESIGN:

HARDWARE DESIGN

The final gripper prototype consists of a three-pronged scissor mechanism that is designed to grab onto any of the different treat types. We opted for a scissor mechanism design to maximize our stroke length for the large variance in treat sizes.

The parallel gripper provided to us by BCIT is the MPG-plus 64 which features a total stroke length of 20mm. The treat sizes range from 10mm to 60mm and thus proved inadequate to use a parallel end effector. Upgrading to the scissor mechanism, our gripper could attain a total stroke length of 80mm, which proved sufficient for grabbing all rounded treat types.

The gripper also features a torsion spring loaded tip that contours to the edge of the table where the treats are spread onto. This is to allow the gripper to come into constant contact with the table to be able to scoop under the treats ensuring a consistent and secure grip.

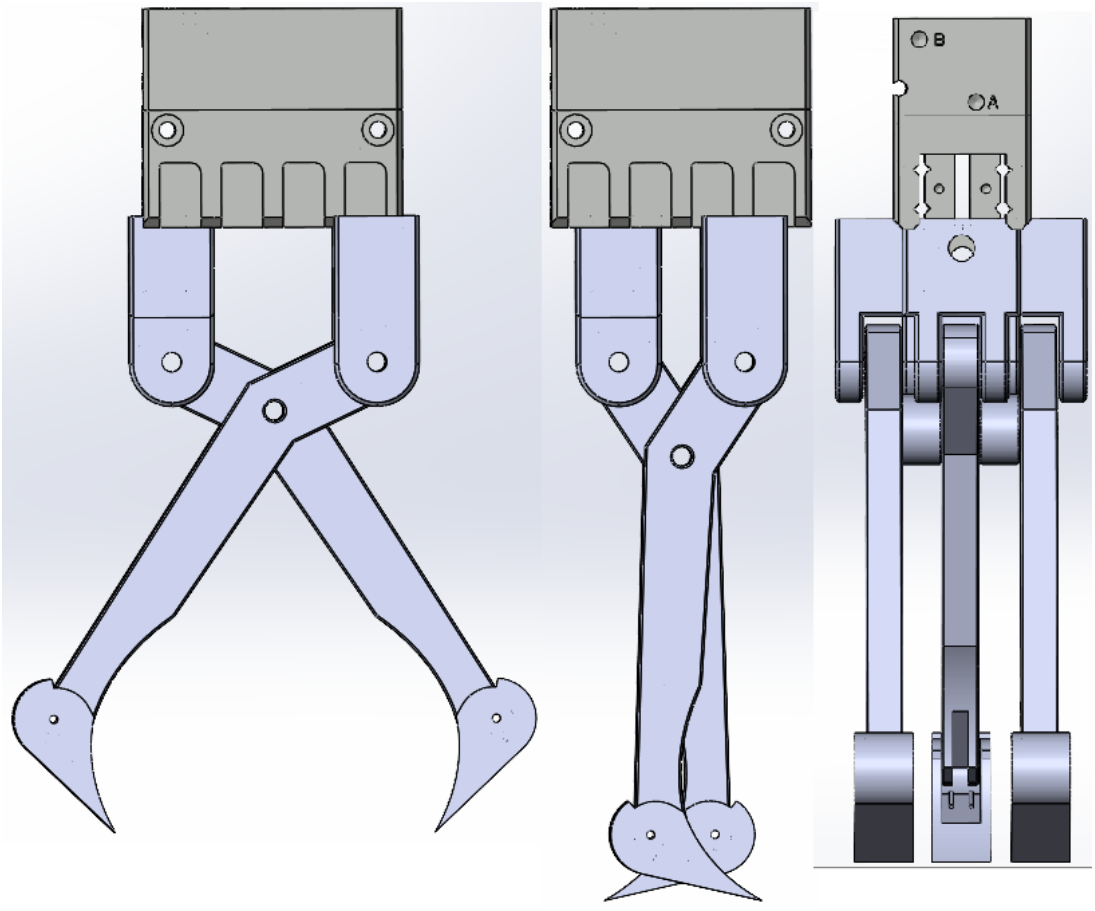

Figure 6: Final Gripper Solidworks Design

ROBOT GRIPPER IMPLEMENTATION:

HARDWARE IMPLEMENTATION

The final gripper is 3D printed with PLA for its affordability and light weight. The end effector is mounted onto the parallel gripper using M6 bolts/nuts. The parallel gripper is controlled through pneumatics. Below is the gripper fully printed and attached to the ABB robot.

Figure 7: Final Prototype Gripper on ABB Robot

View the link below to the SolidWorks files for all iterations of the robot gripper:

https://github.com/wdj1319/21JumpTreat/blob/main/mechatronics_final_project_solidworks.zip

5 ABB ROBOT

The ABB IRB-120 robot is a 6 degree of freedom robot with an ABB robot controller. The robot receives position information and repeatedly performs a labeling sequence with one dog treat at a time.

ABB ROBOT REQUIREMENTS

- 120V 60Hz Outlet connection
- Stationary work area
- Ethernet connection between the microcontroller and the robot
- USB 3 connection between camera and microcontroller
- Python
- RobotStudio RAPID programming
- Actuator to activate the gripper
- Gripper to pickup treats

ABB ROBOT DESIGN:

SOFTWARE DESIGN

The RAPID code begins by sending the robot to the home position and preparing sockets for an incoming connection with our Raspberry Pi 5 microcontroller. Once connected, the robot code waits for a button to be pushed. Upon pushing the button, the code enters an infinite loop. In this loop the robot code sends a string to the microcontroller requesting treat position information. The robot code then receives the position information and moves the robot through its sequence of picking up the treat, labeling the treat, placing the treat in a box and returning home.

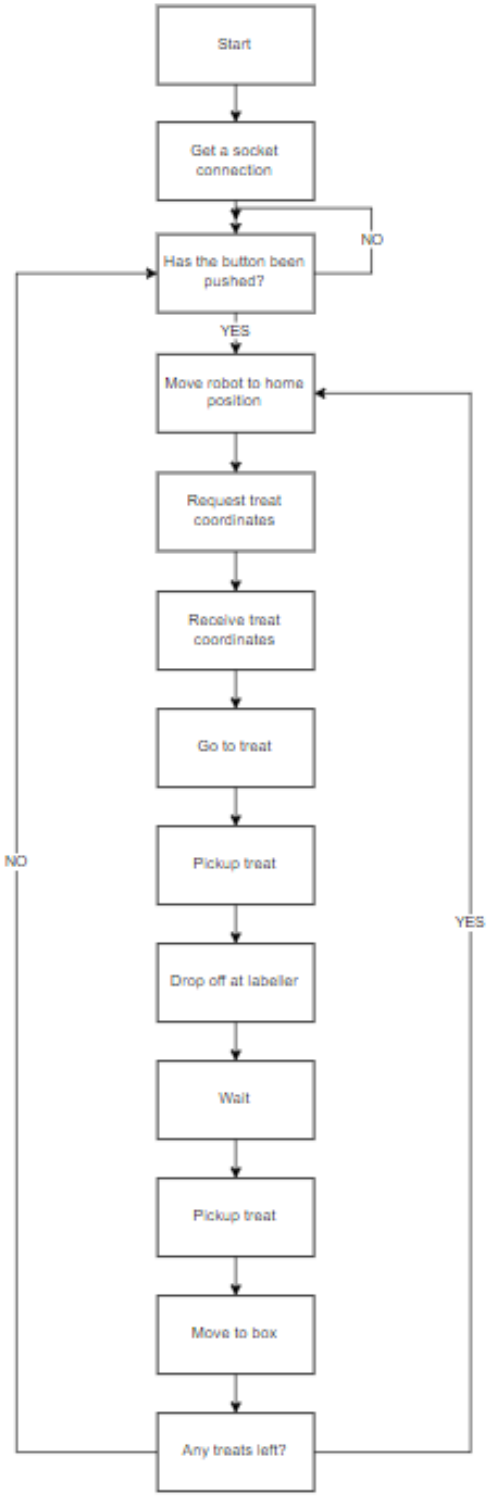

Figure 8 - ABB Robot Process Flowchart

ABB ROBOT IMPLEMENTATION:

HARDWARE IMPLEMENTATION

Added Hardware only includes an ethernet cable connecting the ABB robot controller to the Raspberry Pi 5 microcontroller.

SOFTWARE IMPLEMENTATION

Code written for the ABB IRB-120 exists in the ABB robot controller and uses the RAPID coding language in ABB RobotStudio. All code for the robot can be found in the link to the project Github

6 COMMUNICATION

Communication allowed the system to share position information from the detection by our Luxonis OAK-D Pro camera to the robot controller. It was housed in the Raspberry Pi 5 microcontroller and the ABB Robot controller. The code for our communication was implemented in the Python and RAPID coding languages.

COMMUNICATION REQUIREMENTS

- Position information from camera
- TCP/IP server-client connection model
- Raspberry Pi 5 microcontroller
- Ethernet connection to ABB robot controller
- RobotStudio software and RAPID programming language
- Python programming language

COMMUNICATION DESIGN:

HARDWARE DESIGN

The microcontroller is attached to the side of the table and acts as a hub for the connections of the subsystems. The Luxonis Oak-D Pro camera is mounted above the table and connected to the Raspberry Pi 5 microcontroller via a micro-USB and the ABB robot is communicating through ethernet connection to the robot controller.

Software Design

The Luxonis Oak-D Pro camera sends data to the Raspberry Pi 5 that is collected by a Python program. In Python, a server was coded on our microcontroller to host the robot on a local network. This network was made using the pySocket and Threading Python modules. Next, the ABB robot connects to the Raspberry Pi's server as a client. Once connected and in the home position, the robot pings the microcontroller by sending an agreed upon beginning string. This string is received and recognized by the microcontrollers code. All strings are in Ascii format. Once the beginning string is recognized, the Raspberry Pi sends a response string containing the position information which is constantly updated via the camera code thanks to threading. This position information is then processed and used by the robot to know the treats position in space and the server waits for the next ping.

Communication Implementation:

Software implementation

All code regarding communication in the Raspberry Pi 5 was written in the Raspberry Pi Geany in the Python coding language. The code that dealt with communication in our robot was written in the RAPID coding language while connected to our robot controller via ABB RobotStudio. All code for communication can be found in the link to the project Github

7 System Verification

System Operation

The project achieved its goal of automating the treat labeling process. The system now fully automates the loading and unloading actions, eliminating the need for an employee to perform these repetitive tasks. As a result, the labeling process operates correctly, meeting the intended objectives and enhancing overall productivity.

Performance Metrics

The system meets the specifications required by our sponsor company. The system is able to grab and label 1 treat every 5 seconds in a controlled environment where the treats are uniformly placed, vertical, and spaced a minimum of 26 mm apart from one another. We tested this by running the system and marking the time it took to label every treat and averaging the time to get the time to label a single treat.

8 PROJECT SUMMARY

The goal was to automate the loading and unloading process to work in tandem with an automatic treat labeler.

SYSTEM SPECIFICATIONS

The automated treat labeling system for Open Range has met all the outlined requirements. The ABB IRB-120 robot, along with the Luxonis OakD-Pro vision camera and the Raspberry Pi 5 microcontroller, automate the loading and unloading of treats. Overall, the project has proved successful at achieving its goals, providing a reliable solution for the treat labeling process.

RECOMMENDATIONS/FUTURE WORK

The first recommendation for future work is to purchase a cheaper option for the robot. The current robot is overqualified and under utilized for such a task and a less expensive robot could do just as good of a job for less of a price. A future addition to this project would be a full coverage safety gate for the system. Something like this would allow for maximum speed, therefore maximising output and minimizing safety risks for operators and others working in the immediate area. There are also improvements to be made to the cameras ability to get position information for varying orientations of the treat.

9 CONCLUSION

The outcome of our project met the milestones we set out to achieve when first tasked with automating the treat labeler. For this reason, we can be confident in saying that our project was a success. Along the way we encountered problems and found solutions. The first issue we were met with was a result of using hardware supplied by BCIT for our vision system. Problems with the SICK PIM60 and PIXY 2 cameras arose when we found their FOVs (Frame of View) were far too narrow, and the Pixy 2 was unable to detect treats at the required range. Secondly, we were only marginally familiar with TCP/IP through the communication protocols we learned in our program (UART, SPI, I2C). Newly, we were tasked with creating the communication protocol from both sides of the connection. Resolving the issues resulted in the project being very successful and allowed us to learn many different aspects of mechatronics and apply them as one in a full system.