Final Report

Team 03

Cole MacInnis

Spencer Tingle

Nick Thorpe

MECH 460: Conceive and Design

Department of Mechanical and Materials Engineering

Queen's university

Table of Contents

Executive Summary *1*

1. Background *2*

1.1 Introduction 2

1.2 Design Benefits 2

2. Problem Definition *3*

2.1 Scope and Constraints 3

2.2 Design Criteria/ Customer Needs 4

2.3 Functional Specifications and Metrics 4

3. Design Methodology *6*

3.1 Problem Breakdown 6

Positioning and Logic 6

3.2 Obstacle Detection Methods 6

3.3 Navigation Methods 7

3.4 Positioning and Logic 7

4. Design Options *8*

4.1 Design One: Depth Camera 8

4.2 Design 2 - Inductive Wire 8

4.3 Design 3 – Magnet Strip 9

4.4 Design Proposal 9

5. Depth Camera Selection *10*

5.1 Camera Model 11

5.2 Camera Line following Algorithm 11

5.3 Image Processing 12

5.4 Camera Image Calibration 13

5.5 Disparity Mapping 13

5.6 Graphical Interface 14

5.7 Camera Mounting and Placement 16

6. Logic System *17*

6.1 QR Code Checkpoints 17

6.2 Straightaway 18

6.3 Intersection 19

6.4 Parking 19

7. Microcontroller.. *20*

8. Line Following Material *21*

9. Economic Analysis.. *21*

10. Future work *23*

11. Conclusions... *23*

References... *24*

Appendix A – Quality Function deployment.. *25*

Appendix B – Logic System Flow Chart *26*

Appendix C – Gantt Chart *29*

List of Tables

Table 1: A table of the specified design requirements with a brief explanation of each...4

Table 2: List of the measurable engineering specifications used to meet customer needs with a brief definition of the requirement and explanation of how each specification will be quantified. 5

Table 3: Weighted evaluation matrix comparing the proposed designs to the design criteria. The ranking is based on a scale of 1-5 on how well the design meets the criteria. 9

Table 4: Comparison of stereo depth and time of flight methods for depth cameras [8] [7]. 10

Table 5: Waveshare IMX219-36 binocular camera module specifications 11

Table 6: Microcontroller specifications[14]...20

Table 7: Cost of parts table for entire design. 22

List of Figures

Figure 1: Example of how a factory map simplifies to a connected node map. Pictured left is a factory layout, and right is the corresponding node map. Note that the thickness of lines is used to represent connection weightings for some optimization algorithms. 8

Figure 2: Waveshare IMX219-36 binocular camera module 11

Figure 3: Graphical representation of the Hough Transformation function. 12

Figure 4: Example OpenCV's chessboard calibration method for camera calibration [10]. 12

Figure 5: Visual of stereo calibration and rectification to project images to a common plane[11] 13

Figure 6: The geometric relationship for generating a disparity map [12]. 13

Figure 7: The results of applying the stereo mapping techniques to the stereo-driving dataset[12] The left and right images are overlayed before processing [left], then the frames are undistorted and rectified [center]. The disparities are calculated and transformed to a depth map [right]. Object bounding boxes are overlayed to highlight blob detection in the depth map. 13

Figure 8: GUIs for stereo camera calibration and disparity map tuning. The chessboard stereo-calibration program from ROS is depicted on the left [13]. The disparity map tuning GUI window from Kaustubh Sadekar[13] is depicted to the right. 15

Figure 9: A proposed layout of the motion slider setup. The lower limit of each slider is -1, which allows the user to choose the default parameter. 15
Figure 10: Motorized base and camera mounting setup [right]. 16
Figure 11: Impact of camera height on sight lines for a 50º vertical FOV over several iterations. The chosen camera placement is low and straight [gray], where the camera first sees the ground 20 cm in front of it. The limitation of high placement is illustrated by the 1 m and straight FOV [blue], and the 1 m tilted down 25º [green]. The vertical and horizontal axes represent the respective vertical and forward axes of the robot and are measured in m. 16
Figure 12: Checkpoint logic illustration. The lists of destination options correspond to their respective path choices as denoted by the arrows. 17
Figure 13: Warehouse floor barcode that has a protective covering 18
Figure 14: Visual example of the force vector field. Point A is the bottom of the line, point B is the top of the line, and point C is an obstacle. The bottom right corner illustrates the resulting free body diagram. Blue indicates the robot and its FOV. 18
Figure 15: Preliminary control diagram for the PID Controller on straightaways 19
Figure 16: VSIO drawing of the factory floor. Specifically, this drawing details the route from assembly line 4 to packaged goods. 21
Figure 17: Quality function deployment comparing design alternatives. 25
Figure 18: Main loop. Structured around the straightaway routine with parking and intersection routines called when the appropriate flag is thrown. 26
Figure 19: Function definitions used in main loop 27
Figure 20: Intersection Routine 28
Figure 21: Gantt Chart detailing the project timeline. 29

Executive Summary

SigmaPoint is looking for a robotic solution to their parts delivery system to reduce downtime due to the slow delivery of parts. The design will have a speed limit of 5 km/h, a maximum load of 9.1 kg, and must not be mutually exclusive with manual operation. It must navigate safely and accurately while avoiding obstacles and people. Although there Is no imposed budget, the design should result in a net positive economic benefit for the company.

The robot's design comprises three key subsystems: environment navigation, positioning, and object detection. The final design addresses these subsystems with the Waveshare IMX219-83 stereo depth camera. The camera leverages its depth vision using computer vision processing to identify obstacles. This process involves camera calibration, depth mapping, and blob identification to return the coordinates of nearby obstacles. The images are simultaneously processed to identify yellow tape for path following. A force balance is applied to the environment features, and a PID controller performs direction decisions. Lastly, it reads QR codes to determine its position at intersections and turns. The position is compared with an internally stored "connected node map," and the A* algorithm is performed to assess path-finding decisions.

1. Background

The following section provides relevant background to the project.

1.1 Introduction

SigmaPoint is a 21-year-old Ontario-based company that manufactures complex electronics to address the needs of industrial consumers in fields such as nuclear, medical, medical, and aerospace. Through lean manufacturing strategy, they have successfully grown their business in a competitive environment. SigmaPoint is looking for a robotic solution to their parts delivery system that follows the same direction of cutting waste and limiting process downtime. Currently, parts are delivered by employees who manually push racks to their target destination, but there are several drawbacks to this approach.

SigmaPoint runs five different production lines, and each production line relies on the parts rack delivery system to supply materials and maintain process uptime. Each of these lines runs with a different parts delivery model; some lines pull parts from a stockpile in storage, while others push parts directly from one process to the next. Each part is associated with a barcode. Its status is tracked as it progresses through production; however, SigmaPoint does not currently track the physical location of parts in the plant. Lately, the factory has used around 20-40 parts racks per day, but this number is subject to change depending on-demand requirements.

1.2 Design Benefits

Robotic parts racks will benefit SigmaPoint in many ways. Namely, implementing this automated process will increase process efficiency; thus, increasing production output and saving the company money. Additionally, incorporating automation will elevate the current manufacturing process and put the company in a competitive position with respect to others in the industry because automation and smart manufacturing are emerging and growing fast. Although this system may remove the job of pushing carts, new jobs may appear that involve robot upkeep and calibration. Specifically, a new job strictly focused on robot calibration and coordinating robots for each assembly line in the factory. Direct environmental impacts are not as evident for this project; however, if these racks and parts are to be replaced, they should be disposed of sustainably.

Additionally, the manufacturing and shipping of parts used have an environmental impact and add to the overall carbon footprint of the design. If parts need to be shipped long distances, these modes of transportation typically release large amounts of carbon emissions. Semiconductors and rare metals used in electronics have a significant environmental impact because they need to be mined, processed, and distributed. There are also social issues with the way some materials are gathered, such as the cobalt mines in the Democratic Republic of Congo, where the work environment is harsh, and there are many unethical operations.

2. Problem Definition

Problem Statement

Transportation of racks adds a step to the workflow, and the employees responsible for pushing racks may not deliver them immediately since they have other tasks. Process downtime due to slow delivery of parts is avoidable and has a costly effect on productivity. In addition, a recent survey of the plant suggests that full racks can obstruct the pusher’s visibility. Worker responses indicate that it is common to experience near misses that could lead to a safety incident or have damaged materials.

Stakeholder Needs

SigmaPoint believes that integrating a robotic driver with the parts delivery system could address safety concerns, improve process uptime, and free up workers to focus on other tasks in the factory. They are looking to build a “from the ground up” solution specifically tailored for their needs, which allows them complete control over further customizations in the future.

For the parts delivery system to be successful, it must appeal to the needs of all stakeholders, including the project manager, workers in the factory, servicers, and production customers.

2.1 Scope and Constraints

The objective is to enhance current manufacturing operations at SigmaPoint by developing a control system that automatically moves parts racks to specific locations in the shop, thus avoiding an employee pushing them physically. The design should be built from the ground up to create an innovative solution specifically for SigmaPoint. This solution should easily integrate into the company’s current infrastructure and manufacturing process. To uphold current safety requirements, the total mass on the racks must not exceed 9.1 kg. Additionally, the racks must not travel faster than footspeed, approximately 5 km/h, and automation should not be mutually exclusive with manual operation. Lastly, the automated racks must navigate the shop environment safely by accurately detecting and maneuvering around people and objects, stopping, parking securely, and being electrostatic discharge safe.

The **mechanical design** of the robot and **power sources** are not within the scope of the control system design and will be dealt with by the mechanical design team (team 2). Furthermore, **cataloging and tracking parts** during the manufacturing process is **not within the scope of the control system design** as the client has specified that a current procedure exists and would instead prioritize accurate and safe control of the robotic racks. Lastly, the **client explicitly stated a budget is not being imposed**; however, the **final design** will include an economic analysis.

At the end of this project, the client will be provided with the following:

- A description of sensors, navigation systems and electrical hardware used to automate the parts racks.

- Flow charts outlining the logic that will be used to develop the software.

- Simulated camera calibration and object detection

2.2 Design Criteria/ Customer Needs

The customer needs are specific aspects to the design that the stakeholders indicated they want included. After consulting the clients at SigmaPoint the following needs were deduced as seen in Table 1. These customer needs are implemented in the QFD seen in

Table 1: A table of the specified design requirements with a brief explanation of each.

Customer Need	Definition
Move from point A to point B autonomously	The rack should be able to navigate between two specified points autonomously without human intervention.
Avoid/stop for obstacles autonomously	The racks should be able to stop or avoid obstacles that are obstructing the pathways. This function should be autonomous however, a manual shut off for safety should be included.
Lightweight	Design must not exceed rack weight capacity or weigh so much that racks cannot effectively carry the product.
Ease of operation	Intuitive interface. Should not require extensive training to operate. Should not require consistent interaction to fix issues like calibration.
Scalable to the environment	The design should easily adjust to various sizes of operations. For example, upgrading the system from being used in one assembly station to all five assembly stations should be feasible.
Easily integrated into the current manufacturing process	Extensive changes to the current manufacturing process should not be required.
Speed control	Rack speed should be controlled and be easily adjustable.
Automation with capability for manual use	Racks should be able to move autonomously while still having the ability to be operated manually if required.
Robust design	Sensors picked that are known to be reliable. The control system should be equipped with failsafe behaviours if something unexpected happens. The microcontroller or computer system should have adequate memory and processing power to handle operations.

2.3 Functional Specifications and Metrics

The engineering specifications are measurable criteria used to determine how the customer's needs will be met. A description of these specifications and how they will be quantified can be seen in Table 2. These criteria are included in the QFD seen in Appendix A

Table 2: List of the measurable engineering specifications used to meet customer needs with a brief definition of the requirement and explanation of how each specification will be quantified.

Functional Requirement	Description of Requirement	Quantification
Weight of control system hardware	Control system hardware should not reduce the weight capacity of the rack significantly.	Max weight the racks can carry is 20 lbs. The total control system hardware should be maximum of 3 lbs leaving 17 lbs available for products on the rack.
Surface resistance	Racks need to be ESD safe.	All materials with a surface resistivity less than 10^{11} ohms need modification to be ESD safe [1].
Sensor range	The sensor range should be suitable for an indoor environment. It should allow for easy detection of objects to ensure smooth driving and breaking.	Racks should have 360-degree sensing capability. The sensor range varies depending on the type. Ideally, a close-range sensor that detects ranges from 0 to 5m will be used [2].
Sensor accuracy	Sensors need to detect obstacles, people accurately and navigate the facility.	Many close-range sensors for robot navigation are accurate to about 0.01 mm [2].
Velocity	Racks need to operate at walking speed. If the racks move too fast, the payload is at risk and compromised stability.	The control system will span a velocity range of 0-5 km/h to maintain safe speeds in the factory environments.
Microcontroller processing power	Processing power should withstand a wide range of operational requirements and run code quickly.	Typical hobby Arduino microcontrollers have 16Mhz processors[3]. It is likely the design will need more powerful controllers than this. The processing power of 1 GHz or higher is ideal.
Microcontroller memory	Memory should be large enough to ensure smooth computer operation and code execution.	Arduino microprocessors have various memory capabilities ranging from 2KB SRAM/32KB flash to 96KB/512KB flash [3]. A microcontroller with 2 GB or more of RAM is desirable.
Training hours for operation	Training hours to use the system hours should be minimal.	New workers should be able to use the system confidently within an 8-hour workday.
Installation time	Installation times should be minimized to improve the ease of installation.	The system installation time should not delay factory operations by more than one day of work. Note this is specific to the SigmaPoint factory.
Operational steps	Operational steps for rack function should be clearly defined, easy to follow, and as condensed as possible.	Aim to have less than four manual steps that need to be carried out by workers for the racks to operate autonomously.
Operation sequence	The sequence of operation is defined by the user. Based on locations in the shop.	Racks can move between all defined checkpoints in the building.
System shutdown and conversion to manual use	The automated racks should be able to shut down and resume manual use if required.	The shutdown time should be minimized and take no longer than 5 seconds.

3. Design Methodology

The overarching design problem has been broken down into four main categories specifically, obstacle detection, navigation, positioning and logic system, and the microcontroller. Each category has been tackled sequentially as the following components are all dependent on the last. A summary of the design process and research in each category prior to formulating potential design solutions is outlined as follows.

3.1 Problem Breakdown

Obstacle detection

Object detection is used to inform the system of unexpected obstacles. Detecting obstacles early and accurately gives the robot adequate time to stop and avoid collisions. Static object detection is relatively straightforward and can be accomplished with a simple IR send/receive sensor. Detecting dynamic objects such as pedestrians or other carts poses a greater challenge since the controller must look for obstacles in its peripheral vision.

Navigation

The navigation system is crucial to ensure the rack can move from point A to point B along a set path. The movement should be smooth and controlled while maintaining the ability to maneuver around potential obstacles. Various navigation methods are explored in section 3.3 below.

Positioning and Logic

A logic system is necessary to ensure the control system can be scaled factory-wide and allow the rack to travel from point A to point B. The cart may encounter a variety of scenarios during a route where decisions must be made at intersections or key points in the factory where the cart may need to stop. A checkpoint system is developed as described in section 3.4 to enable this decision-making ability.

Microcontroller

This component will be selected last once a final solution is determined. Processing power and memory can vary substantially depending on what is being used for the subsystems discussed above. For example, absolute navigation systems are more computationally expensive than relative navigation. Regardless, it is expected a more powerful microcontroller than the typical hobby Arduino will be required. As mentioned in Table 2, a microcontroller with at least 2GB of RAM and 1 GHz of processing power is ideal.

3.2 Obstacle Detection Methods

A good object detection system should and react quickly and effectively to unexpected obstacles while considering scalability issues. Further, a solution should consider implementation costs, computational requirements, and power draw. Three detection methods that were considered are an infrared proximity sensor, ultrasonic sensor, and depth camera.

An active sensor sends energy into the environment as an electromagnetic or ultrasonic wave, then waits for the signal to return. The time of return is used to collect information about the distance to nearby objects. An example of this would be ultrasonic sensors or LiDAR. Environments with several active sensors are subject to interference; this “crosstalk” is a critical limitation to creating a scalable design. Some sensors, such as ultrasonic sensors, can use filtering methods to mitigate the challenges of crosstalk while others (e.g. time-of-flight depth cameras) struggle to perform reliably. In contrast, the passive sensor reads energy already present in the environment. An example of a passive sensor is the stereo-depth camera.

3.3 Navigation Methods

The most common navigation methods are direct, absolute, and relative navigation. Direct navigation involves following a predetermined path such as a line, magnetic strip, or embedded wire [4]. Absolute navigation uses a local positioning system to inform the controller of its location with respect to the target destination. The robot drives within the system’s boundaries and artificial intelligence (AI) is allowed to make path-finding decisions [4]. In general, the controller compares its current position with the requested position and makes a move towards the target while avoiding obstacles and boundaries. Relative navigation systems use computer control systems to assign tasks and direct the vehicle by estimating trajectories [8]. The relative position and orientation are calculated with respect to a known starting point, orientation, and velocity using data from motion sensors and rotation sensors [9]

After evaluating the three major navigation categories, guided navigation is a good balance between flexibility and reliability. Guided navigation is simpler, more rigorous and cost-effective compared to absolute navigation. Comparing guided navigation with relative navigation, it is found that guided is more reliable because there is less error accumulation, is more scalable, and has an easier setup/operation. Therefore, the design options generated will only consider different guided navigation techniques.

3.4 Positioning and Logic

This report proposes a logic flow called the *connected node* method. This method simplifies map-based approaches by approximating each intersection as a ‘node,’ and the controller only stores information about which other nodes are adjacent. The cart travels along a guided path until it approaches an intersection. Each intersection has a checkpoint marker that contains data to inform the robot both of its current position and about the respective directions of adjacent nodes. This differentiates from traditional step based directional logic as it does not need to compute its exact location akin to a GPS system. Figure 1 below shows a visual depiction of how a factory layout is simplified to a connected node map.

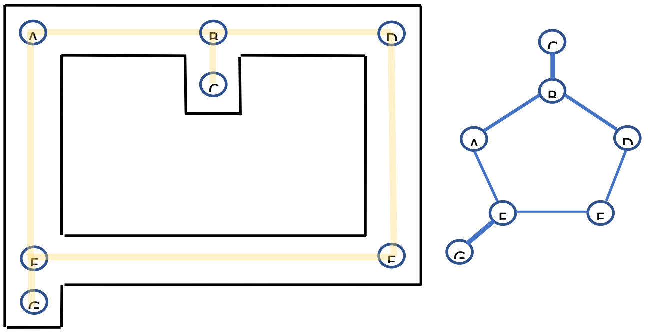


Figure 1: Example of how a factory map simplifies to a connected node map. Pictured left is a factory layout, and right is the corresponding node map. Note that the thickness of lines is used to represent connection weightings for some optimization algorithms.

4. Design Options

The following section outlines the three potential design solutions and the selection process based on the customer needs and the design criteria specified. Each design is characterized by its object detection method, and its line detection and logic system are detailed within. A weighted evaluation matrix and a quality function deployment are used to analyze the proposed solutions.

4.1 Design One: Depth Camera

The first design employs one depth camera that to address the line following, object detection, and checkpoint identification subsystems. This prioritizes efficiency and scalability, and ease of implementation at a higher cost of development, computational cost, and power efficiency.

The line following uses computer-vision algorithms to identify and follow the path. A bright yellow line is taped to the floor as the trail marker. The camera will capture the floor in front of it and follow various pre-processing techniques to identify the line's position with respect to the robot. The camera will also run an object detection algorithm to identify obstacles.

The design will employ the connected nodes method as its logic system uses QR codes at checkpoints along the path where a decision must be made (straight, left, or right). Selecting this method prioritizes scalability, path efficiency and ease of implementation that are considered throughout this design.

4.2 Design 2 - Inductive Wire

Option two is a wire guide system comprised of a series of wires embedded in the factory floors. The robot analyzes the transmitted signals from the wires to make sure it keeps following the correct path. The object detection method for this system will use three diverging sharp sensors. The connected nodes system will be employed with RFID tags will be used as a checkpoint system. At each checkpoint, an RFID tag will indicate what decision must be made based on user inputs for start and end destination. The robot may need to continue straight, turn left, turn right, or stop and park. This method has a simpler and less computationally expensive system; however, it is much less scalable and more invasive to implement as wires need to be embedded in the floor.

4.3 Design 3 – Magnet Strip

Design option three uses a magnetic guide rail system to navigate the factory environment. There are two magnetic strips on the floor, mapping the desired pathway with magnetic sensors attached to the rack. These sensors relate rack position to the magnetic fields being emitted by the tape. An ultrasonic sensor will be implemented for object detection and avoidance as it is a lightweight, inexpensive, and reliable sensor for the controlled environment. The connected nodes method and QR code checkpoints will be implemented as a logic system to account for any decisions that need to be made along the robot's path. A QR code scanner will read each checkpoint and relay the information to the control system.

Table 3: Weighted evaluation matrix comparing the proposed designs to the design criteria. The ranking is based on a scale of 1-5 on how well the design meets the criteria.

Criteria: 1 = Poor 2 = Between 3 = Met 4 = Between 5 = Surpassed	Move from point A to point B autonomously	Avoid/ stop for obstacles autonomously	Lightweight	Ease of operation	Scalable to the environment	Easily integrated into the current manufacturing process	Speed control	Automated with capability for manual use	Robust Design	Adaptable	Low cost	Total Score
Weights (1-5):	5	5	3	3	4	3	4	5	3	4	3	
Depth Camera	5	5	4	4	5	4	5	4	3	4	3	188
Inductive wire	5	3	3	3	1	2	5	3	2	2	3	150
Magnetic strip	5	4	3	4	4	4	5	4	3	4	4	169

4.4 Design Proposal

After carefully weighing critical criteria against each design, the depth camera solution is quantitatively superior to other methods. Using tape as the guide rail addresses ease of implementation criteria since the only set up for the guide rail is laying down the bright yellow tape to identify paths. Using CV for line recognition allows for "smart" evasive maneuvers where a robot could avoid stationary path-blocking obstacles by temporarily departing from the line. Investigating evasive maneuvers will not be explored at this point, but it highlights the freedom that CV proposes over other options.

Although the depth camera is more computationally expensive and has higher power consumption than other options, its improved performance is ideal for this design. The faster framerate addresses point-to-point efficiency since faster read times create faster response times. This means the robot can move quickly, knowing that it will react faster. Using QR codes is also easy to install as they will be stickers placed on the ground.

This design is an elegant solution for the end-user in terms of scalability, ease of implementation, ease of use, and efficiency. The drawbacks of this solution lie within the complexity of its underlying code. Research and development will require significant work to coordinate these subsystems together.

5. Depth Camera Selection

Depth cameras determine the distance to each pixel to provide 3-D context. Three standard methods for determining pixel distances are structured light, stereo depth, and time-of-flight (ToF)[5]. The stereo depth camera takes pictures with two side-by-side cameras and compares their features to triangulate the distance between each pixel [6]. The time-of-flight method analyses the length of time light has been in flight which can be related to depth using kinematic equations [7]. Structured light is a specific type of stereo depth camera where light is projected towards a scene and uses the camera to observe how it deforms around the environment [7]. Table 4 below compares some main properties of stereo and ToF depth cameras.

Table 4: Comparison of stereo depth and time of flight methods for depth cameras [8] [7].

Depth Technique	**Pros**	**Cons**
Stereo Depth	• Commonly used technology with many resources and libraries online. • No crosstalk. • Higher resolution • Faster Framerate • Less expensive	• Performs poorly on flat surfaces such as walls. • Higher power consumption • Working distance 2m or less • More computationally expensive • Needs ambient light
Time of Flight	• Commonly used technology with many resources and libraries online. • Works in low lighting conditions • Lower power consumption • Working distance from 0.4 m to 5 m • Less computationally expensive	• Susceptible to crosstalk • Lower resolution • Slower framerate • Hardware more expensive • Dead spot

Both stereo depth and time of flight cameras are suitable for the robotics parts racks system. Evaluating the pros and cons of camera types, it is determined that a stereo depth camera is the most ideal for this application. It is justifiable to sacrifice a more computationally expensive option to achieve a higher resolution and faster framerate, increasing object detection abilities. There is no dead spot for the stereo camera, whereas ToF cameras typically have a dead spot when objects are closer than 0.4m. Additionally, the stereo depth camera is not susceptible to interference from other cameras (crosstalk), which is ideal for scalability when implementing this factory-wide system. Although lighting affects the stereo depth camera more than ToF cameras, it is not a concern as the operating environment is controlled with ambient light.

5.1 Camera Model

The specific stereo depth camera model selected for this application is the Waveshare IMX219-83 binocular camera module. The hardware can be seen in Figure 2.

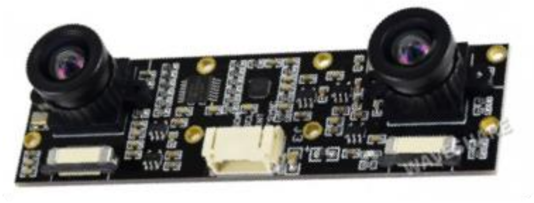


Figure 2: Waveshare IMX219-36 binocular camera module

Table 5 below summarizes the fundamental properties of the selected depth camera. The most important properties of the camera for effective obstacle detection and navigation are the resolution and the field of view. The resolution must be high enough that the camera can effectively see the surroundings and accurately identify the line. Eight megapixels is more than adequate to achieve this requirement and can capture clear and detailed images. The field of view specified in Table 5 enables the camera to see the front environment while travelling at various heights. The proposed mounting position is described in Section 4 – Camera Mounting**.**

Table 5: Waveshare IMX219-36 binocular camera module specifications

Resolution	8 megapixels 3280x2464 (per camera)
Sensor	IMX219
Focal Length	2.6 mm
Field of View	83° diagonal 73° horizontal 50° vertical
Distortion	<1%
Microcontroller Compatibility	Jetson and Raspberry PI

5.2 Camera Line following Algorithm

The Hough Transformation is a technique available apart of the Open CV library that will be used to program the camera on the robot. This technique represents a line in polar coordinates (r,θ). For a given X and Y coordinate, a line defined by (r,θ) can be represented as seen in Eq 1**,** where r is between 0 and 2 pi radians. Plotting r on the y ais and θ on the x-axis, a sinusoid is produced.

$$r_\theta = x_0 \cos(\theta) + y_0 \sin(\theta) \tag{1}$$

Multiple sinusoids are produced from plotting every X and Y coordinates gathered in an image. The intersection of these curves corresponds to the line where the respective X and Y coordinates lay. Therefore, a line can be detected by finding the number of intersections between curves. A threshold can then be defined to determine the minimum number of intersections corresponding to the yellow line. This exact threshold will need to be resolved through physical testing and is easily modified in the code. An example of the graphs produced from plotting three sets of X and Y coordinates using (eq 1) is seen below in Figure 3.

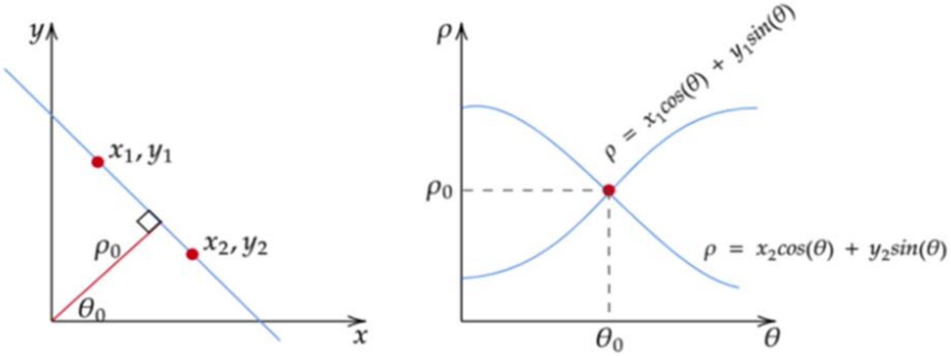

y
x_1, y_1
ρ_0
x_2, y_2
θ_0
x
ρ
$\rho = x_1 cos(\theta) + y_1 sin(\theta)$
ρ_0
$\rho = x_2 cos(\theta) + y_2 sin(\theta)$
θ_0
θ

Figure 3: Graphical representation of the Hough Transformation function.

This line detection method is highly versatile because lines of various orientations can be detected, including lines that are parallel and orthogonal to the robotic rack. Furthermore, the algorithm can be adapted to detect lines and shapes despite them being broken or discontinuous, meaning the rack can continue following the line even if some obstacles were covering the tape—for example, a piece of paper.

5.3 Image Processing

Unlike expensive stereo cameras such as the intel real sense [9], the wave share does not perform on-board depth processing. All depth estimation and object detection processing must occur on the raspberry pi microcontroller. There are many methods and libraries dedicated to these tasks, such as SLAM [10] (simultaneous location and mapping) or neural network-based models. The ideal solution for this project should be straightforward so future engineers can easily continue this work. It should consider the processing limitations of the raspberry pi while producing results that are accurate, consistent and in real-time. The OpenCV image processing library is a computer vision library designed in C++ and Python that focuses on creating easy-to-use and lightweight functions for computer vision. It is the standard for hobby CV projects because of its detailed documentation and breadth of help resources.

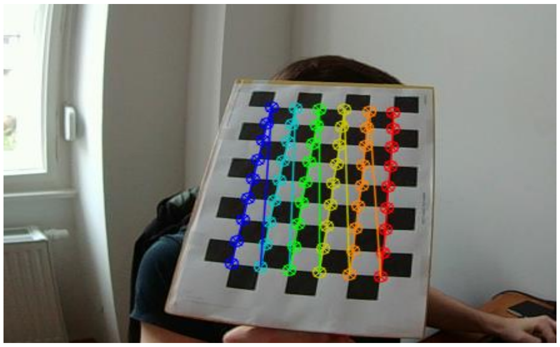


Figure 4: Example OpenCV's chessboard calibration method for camera calibration [10].

5.4 Camera Image Calibration

Before performing a stereo analysis on stereo images, the cameras must be calibrated such that the images are aligned in the same coordinate system. First, each camera must be calibrated to remove lens distortions. Next, the images must be rectified to reproject them on the same plane as depicted in Figure 7. OpenCV offers one-step procedures for each of these tasks, which involves using a printed checkerboard to automatically fit them to a shared plane. Checkerboard identification was tested on Ali Yasin Eser's [11] calibration photos in Figure 4. The rectification process was also tested with the stereo-driving dataset, and the results are depicted in Figure 5

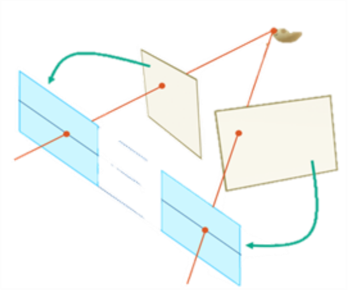


Figure 5: Visual of stereo calibration and rectification to project images to a common plane[11]

5.5 Disparity Mapping

The procedure to extract depth from a set of stereo images is derived with a simple geometric relationship. For two identical cameras O and O', that have perfectly aligned coordinate systems, a common feature, X, is identified at x and x' in each respective camera. The real-world distance z to feature X is proportional to the width between cameras B, the cameras focal length f, and inversely proportional to the disparity between points $x - x'$ [12]

$$disparity = x - x' = \frac{Bf}{Z} \text{ (2)}$$

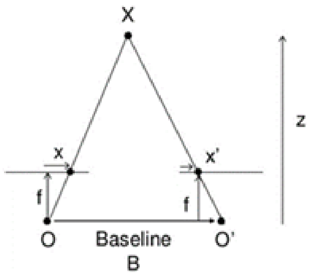


Figure 6: The geometric relationship for generating a disparity map [12].

OpenCV offers the SGBM [10] (semi-global-block-matching) algorithm to generate disparity maps. It is an effective and quick method; however, the tuning process is involved. The SGBM parameter tuning interface will be a part of the GUI section of this report, but the tuning procedure will be created once the team acquires the hardware. The SGBM method was tested on the stereo-driving dataset to create a depth map and is pictured in Figure 7 below.

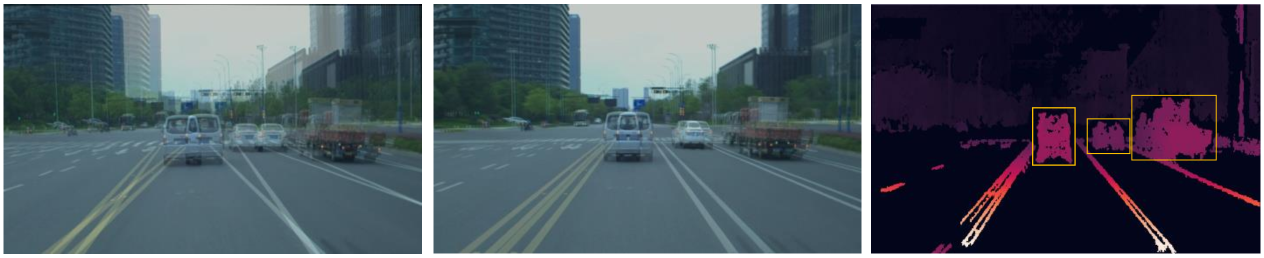


Figure 7: The results of applying the stereo mapping techniques to the stereo-driving dataset[12] The left and right images are overlayed before processing [left], then the frames are undistorted and rectified [center]. The disparities are calculated and transformed to a depth map [right]. Object bounding boxes are overlayed to highlight blob detection in the depth map.

Once a depth map is created, the remaining challenge is detecting objects. One lightweight solution to object detection is blob detection. This method detects regions of the image that significantly contrast with its surroundings and assigns a bounding box to each blob, such as the yellow ones pictured in Figure 7. The depth map will identify features on the floor plane, such as the guideline. Floor plane features may be mistaken as blobs, so one more consideration must be taken to account for this.

There are two appealing solutions to account for accidental floor-blob detection. The simplest solution is to place the camera such that its optical center does not intersect with the floor plane. If the center of the camera's vision cannot see the floor, it can be assumed that all features in the top half of the depth map will not contain features from the ground plane. It will no longer recognize objects that are below the optical center. Therefore, the camera must be placed very close to the ground to account for this, and it will be unable to identify very short objects. The second method to address this is to transform the perspective of the camera so that all floor features belong to a ground plane. The ground plane can be identified using the Hough-transform method described in the line-identification section of the report, and the transform is performed using OpenCV's warp perspective method. This method is more rigorous, but it is unclear if it is necessary. Due to processing constraints, the former solution will be considered for now. However, implementing the latter is minimally involved and will be considered in the testing phase.

5.6 Graphical Interface

The GUI (graphical user interface) is a critical feature to ensure a quality experience for the end-user. The GUI serves three purposes: to initialize nodes and to calibrate the camera system and PID controller.

The OpenCV community has several GUI-based calibration programs that will be used for camera calibration. ROS-stereo employs a clean user interface to perform user-guided chessboard calibration [13]. It automatically computes the distortion matrixes and calculates the transformations required to rectify the images. This program has extensive documentation and has a breadth of helpful resources online. Kaustubh Sadekar has created an OpenCV tutorial with open-source code to easily calibrate the parameters for the disparity function. The code used in this tutorial is replicated in several other tutorials as well. There are many parameters that can be fine-tuned; therefore, it will be necessary to create a step-by-step manual to help guide the user towards the optimal result. It is not feasible to design this manual right now, but it will be created once the team acquires the hardware and becomes familiar with the tuning process. These tuning GUIs are depicted below (Figure 8).

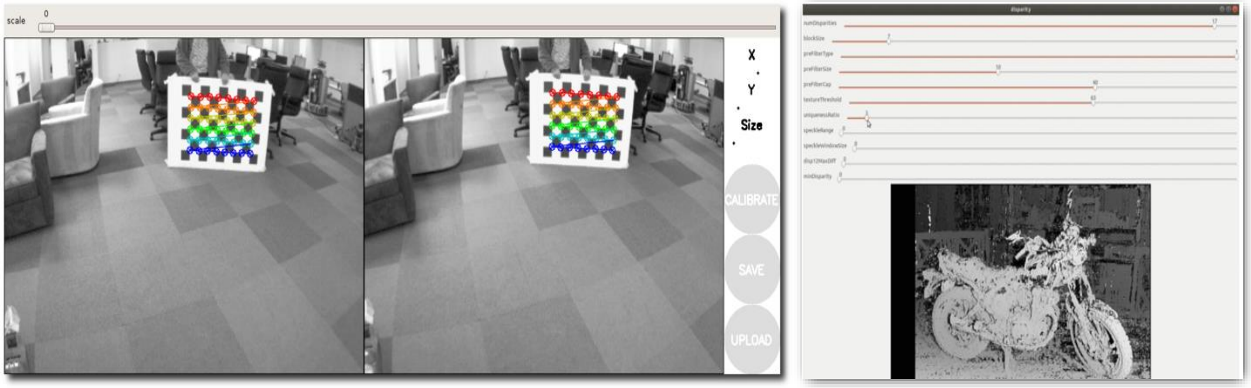
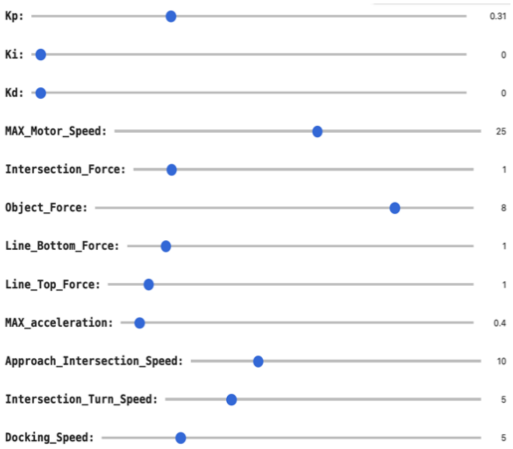


Figure 8: GUIs for stereo camera calibration and disparity map tuning. The chessboard stereo-calibration program from ROS is depicted on the left [13]. The disparity map tuning GUI window from Kaustubh Sadekar[13] is depicted to the right.

The PID controller ensures the robot makes smooth and effective corrections to return to the proper course. It must be specifically tailored to this use ; therefore, it will be designed from scratch. The PID controller has several coefficients that must be tuned to ensure smooth and safe travel. The microcontroller will save the results of the most recent PID errors and display them as a plot beside the sliders for the user to reference. The manual will also contain examples of bad error plots and the adjustments required to fix them. A preliminary design for the sliders is shown in Figure 9 below.



Kp:	0.31
Ki:	0
Kd:	0
MAX_Motor_Speed:	25
Intersection_Force:	1
Object_Force:	8
Line_Bottom_Force:	1
Line_Top_Force:	1
MAX_acceleration:	0.4
Approach_Intersection_Speed:	10
Intersection_Turn_Speed:	5
Docking_Speed:	5

Figure 9: A proposed layout of the motion slider setup. The lower limit of each slider is -1, which allows the user to choose the default parameter.

5.7 Camera Mounting and Placement

The depth camera will be mounted on the front of the motorized unit designed by the mechanical team (team 02) at the bottom of the rack. The camera will be mounted from the rear using the premade fastening holes with additional hole cutouts for the lenses to fit into. The mounting configuration can be seen in

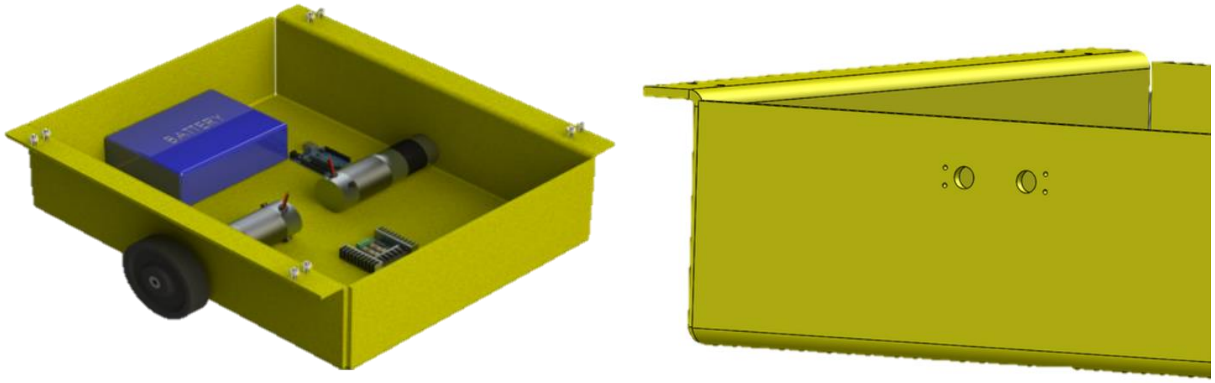


Figure 10: Motorized base and camera mounting setup [right].

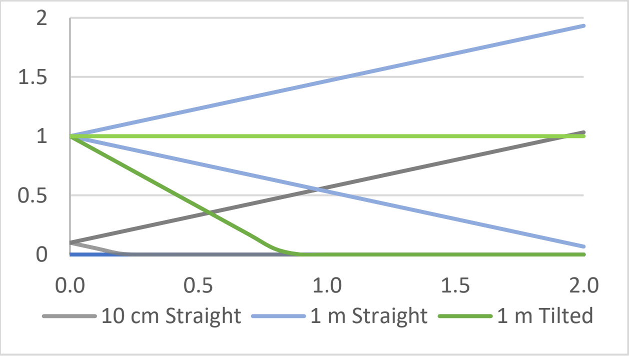


Figure 11: Impact of camera height on sight lines for a 50º vertical FOV over several iterations. The chosen camera placement is low and straight [gray], where the camera first sees the ground 20 cm in front of it. The limitation of high placement is illustrated by the 1 m and straight FOV [blue], and the 1 m tilted down 25º [green]. The vertical and horizontal axes represent the respective vertical and forward axes of the robot and are measured in m.

Because the stereo cameras will be used for QR Codes, line following, and object detection, placement is critical for optimal performance. As explored in the Image Processing section, aligning the camera's plane to be orthogonal with the ground could potentially resolve any uncertainty from mistaking floor features as an object. Given the camera's vertical FOV of 50º, incorrect camera placement could create critical dead-spots in the camera's vision. Higher camera placements lead to decreased floor vision as illustrated in Figure 11. Placing the camera low to the ground provides adequate floor vision for QR code reading and line identification. The caveat to this placement is that the vertical field of vision is reduced significantly. It is reasonable to assume that any object encountered will be connected to the ground at its base, therefore a large field of vision above the camera is not critical. The increased glare from the floor at low placements was considered but is dismissed due to the low reflectivity of the ESD flooring.

6. Logic System

The logic system serves three primary functions: observing the environment, performing direction decisions, and acting on those decisions. The logic system operates in a loop which will perform a routine depending on if the robot is in a straightaway, at an intersection, or at the destination where it will perform the docking sequence. A detailed flow chart summarizing the logic system is available in Appendix B

6.1 QR Code Checkpoints

In SigmaPoint's facility, the autonomously guided cart must decide which path to take at any given pathway junction. First, the cart verifies its current location using QR codes located at each junction. The cart then assesses a list of the facility's junctions within its onboard memory and chooses its next direction of movement according to the destinations available from the junction it is located at. This list is based on the connected nodes, and the logic described in this section reflects the cart's utilization of this list.

The exact functionality of the codes is essentially a location checkpoint. Moreover, the cart uses the codes to verify which junction the cart is currently located at within the facility. The carts are equipped with lists of destinations that each junction path leads to. As opposed to the QR codes containing the lists themselves, this logic is more scalable, as updating the QR codes individually is more tedious than adjusting the lists contained within each cart's memory. Depending on the communication methods of the carts, the entire fleet's firmware can be updated collectively to reflect any changes in navigation decisions that SigmaPoint may wish to make. An illustration of this logic is seen below in Figure 12**.**

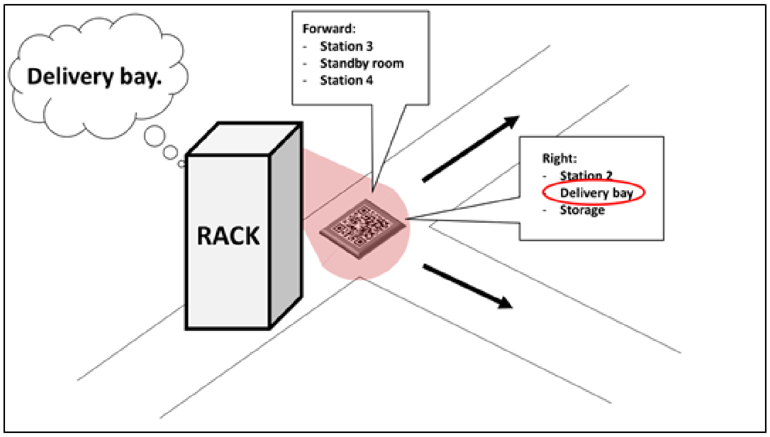


Figure 12: Checkpoint logic illustration. The lists of destination options correspond to their respective path choices as denoted by the arrows.

Figure 12 exhibits the desired functionality of the cart when assessing a junction within the facility. Presented with the available directions, the cart evaluates the destination options for each direction. Upon identifying the corresponding direction for its desired location, the cart follows this direction and ultimately arrives at the desired location. In some instances, this process may consist of multiple QR code checkpoints during the cart's journey throughout the facility, and it will achieve the same successful result.

A code installment method is shown in Figure 13 **.**

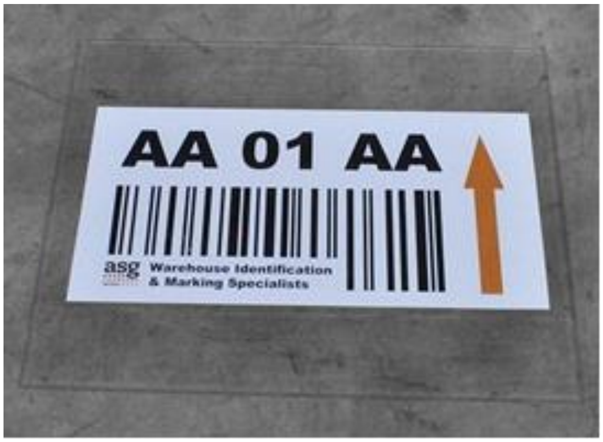


Figure 13: Warehouse floor barcode that has a protective covering

Lamination prevents peeling, abrasion, and other forms of damage to the barcode that impair its legibility. This same assembly can be applied to the QR codes as part of our design solution. This is not a cost intensive solution, and it withstands its operating environment.

6.2 Straightaway

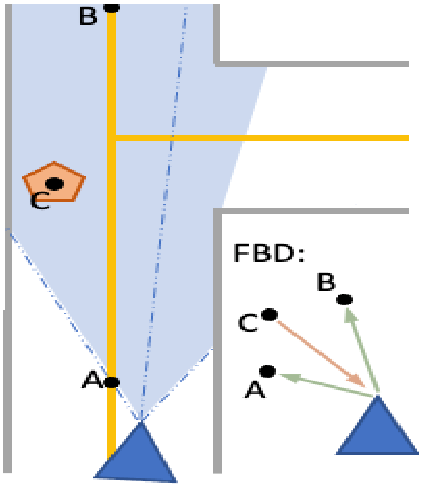


Figure 14: Visual example of the force vector field. Point A is the bottom of the line, point B is the top of the line, and point C is an obstacle. The bottom right corner illustrates the resulting free body

Along straight paths between checkpoints, the robot must check for obstacles, follow the line, and identify upcoming intersections. To maintain course, the computer uses a force-vector approach, where the attractive vectors encourage the robot to move in the correct direction, while repulsive forces slow the robot down from potential obstacles or walls.

First the program reads the most recent camera frame and processes it to identify any objects and returns the coordinates of the nearest one. Simultaneously, the line identification process returns the coordinates of the top and bottom of the vertical line. If a horizontal line is also detected, it will return the coordinates of the intersection. The coordinate of the top of the line, bottom of the line, and nearest objects are used to compute their respective force vector with a simplified version of newton's law of universal gravitation.

$$F_{12} = G\frac{m_1 m_2}{r^2} \qquad (3)$$

where G is replaced with a calibrated weight that is unique for each object. Appropriate weights will be calibrated during testing. A visual representation of this technique is pictured in Figure 14. Masses will be set to one as the main scaling in this equation is from the weighted constant G.

The force vectors will be resolved into x and y components and used as error in the PID controller. The max top speed of the left and right motors will be represented as $delta_{max}$. Initially the left and right wheel speeds $delta_L$ and $delta_R$ will equal $delta_{max}$, and then are adjusted according to the error due to horizontal force. Next, the error due vertical (forward facing) forces is used to scale these deltas further. The control diagram for this process is pictured in the Figure 15 below.

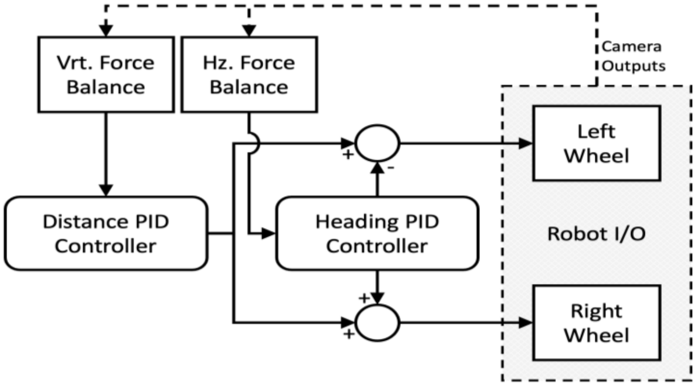


Figure 15: Preliminary control diagram for the PID Controller on straightaways

6.3 Intersection

The robot will slow down as it approaches intersection and stop once the QR code is at a readable distance. The robot will scan the QR code, which informs it of the ID of the current node, the IDs of adjacent nodes and their direction. It will check this information against the internal node map to determine the required direction change based on the user inputs for start and end destinations. Once the QR code is read, the robot will move forward until it is at the center of the intersection. The robot will identify that it is at the center of the intersection when the horizontal intersection line is no longer in its line of sight, roughly 20 cm away. It will move forward 20 cm with dead reckoning, then perform a rotation according to the instruction acquired at the QR code. It will then continue the straightaway routine.

6.4 Parking

The parking routine will follow the same process as the intersection routine, however once it performs the 90º turn, it will enter a short subroutine. A yellow line of tape is used to mark the back of the parking space, and the robot will creep forward until it is no longer in sight. The robot will then perform a 180º turn until the road line is in sight and orthogonal to the robot. At this point, it will enter a rest mode to conserve power until it receives a command.

7. Microcontroller

The data processing unit as part of our design will be a Jetson Nano Developer Kit B01. The control system requires object detection, line following and QR code checkpoint scanning sequences be ran simultaneously which requires a significant amount of RAM and processing power. As mentioned in the functional requirements a controller with more than 2 GB of RAM and 1 Ghz processing power is desirable. The chosen microcontroller exceeds these requirements, and the relevant specifications are summarized in Table 6 below.

Table 6: Microcontroller specifications[14]

Specification	Value
GPU	128 – core Maxwell
CPU	Quad -core ARM A57 @ 1.43 GHz
Memory	4 GB 64-bit LPDDR4 25.6 GB/s
Storage	microSD
Camera Connection	2x MIPI CSI-2 DPHY lanes

Apart from the technical specifications the Jetson microcontrollers are common and known to be reliable in the robotics field. The controller is also user friendly and has many teaching resources available online which is another appealing factor. As a result, this hardware will be easier to work with.

8. Line Following Material

Using the depth camera described in section 5.1 a bright yellow line will be installed centered on the floors which the robot must navigate. This route is illustrated in the VSIO drawing Figure 16 below and is specifically for assembly line four. The total distance of the route is 72m.

There are two main options manufacturing facilities implement for floor markings. The most common approach is to use specialized floor tape made of strong materials such as vinyl paired with strong adhesives [15]. The main advantages of using industrial tape are affordability, ease of implementation, and higher scalability, which are significant design criteria. Another option is to use floor marking paint. This method requires more time to install and typically has higher costs [15] while being less scalable.

Considering the factors discussed above related to the design criteria and customer needs outlined Table 1 and Table 2 the 3M Ultra Durable Floor Marking Tape 971 is recommended. This tape is designed for high activity and heavy traffic factories and can withstand scuffing from pallets and heavy equipment such as forklifts [16]. Given that SigmaPoints manufacturing facility does not operate with heavy equipment and large items that could damage the floors, the tape should withstand factory conditions.

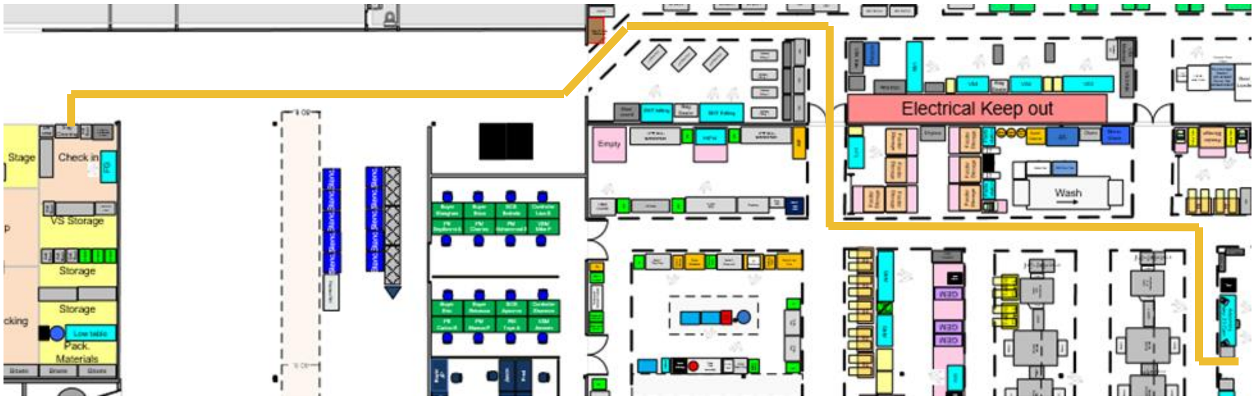


Figure 16: VSIO drawing of the factory floor. Specifically, this drawing details the route from assembly line 4 to packaged goods.

This tape would be a sound solution when the project is at the implementation stage; however, for the prototyping and testing phase it is advisable to run tests using cheaper yellow duct tape. The project budget accounts only for yellow duct tape as this is what will be used for the next stages in the project. The slight variances in color are not a concern as the processing techniques will be flexible enough to account for this.

9. Economic Analysis

The following section details the estimated cost of implementation for the control system. A total cost for the design combining the control system solution and the mechanical system is estimated to be $2319. This preliminary was presented to the clients and has been accepted by Sigma Point. Sigma Point is also willing to implement a contingency budget in case specific part need to be changed or more parts are required. Each component and its price are summarized in Table 7 below. It should be noted at this stage in the design a single roll of duct tape will be used for testing to reduce cost significantly. More robust tape can be selected based on client preference and specific application. If the 3M Ultra Durable Floor Marking Tape is implemented the cost will increase as each roll costs $195 and 3 rolls would be required to span the 72 m pathway.

Table 7: Cost of parts table for entire design.

Item No	Description	Supplier	Qty	Pkg	Cost USD	Cost CAD
1	Aluminum Chassis Manufacturing	Mcglaughlin Shop	1	1	-	250
2	Aluminum Chassis Stock	MetalPros	1	1	-	225
3	Galvanized Steel U Bolt	McMaster Carr	4	5	4.15	6
4	DC Gearmotor	Midwest Motion	2	422	844	1072
5	ESD Safe Rubber Wheel	McMaster Carr	2	1	23.58	30
6	Battery Pack	-	1	1	265	337
8	Motor Controller	Digikey	1	1	121.25	154
9	Stereo Depth Camera	Waveshare	1	1	-	45
10	Camera Extension Cables	Waveshare	2	1	-	16
11	Jetson Nano Dev Kit B01	Nvidia	1	1	-	100
12	64 GB UHS-1 Micro SD Card	Sandisk	1	1	-	26
13	Raspberry Pi Power Supply USB-C 5.1V 3A	Raspberry Pi	1	1	-	10
14	Acrylic Case for Jetson Nano B01 with Cooling Fan	Seeed Studio	1	1	-	25
15	Yellow Duct Tape	ULINE	1	1	-	7
16	Laminated QR Code	Staples	4	4	-	16
Total						$2319

SigmaPoint estimates that the system implemented will reduce downtime by approximately 30 minutes per shift. With three shifts per day, the reduced downtime equates to approximately 30 dollars in daily savings, assuming an hourly wage of $20 per hour (see Eq 3). These daily savings can be represented as yearly savings with a value of $10950 (see Eq 4). Using the initial cost of the project and the daily savings generated from the project as positive cashflows the payback period can be estimated as 77 days as shown in Eq 2 below.

$$Fractional\ Payback\ Period = \frac{Initial\ Investment - CF\ Opening}{Ending\ CF} + n_{open}(days) \qquad 4$$

$$= \frac{2319 - 2310}{2340} + 77\ days$$

$$= 77\ days$$

$$Daily\ savings = 10\frac{dollars}{shift} * 3\frac{shifts}{day} = 30\frac{dollars}{day} \tag{5}$$

$$Yearly\ savings = 30\frac{dollars}{day} * 365\frac{days}{year} = \$10950 \tag{6}$$

10. Future work

There are still some limitations in the design that need to be addressed. The next steps for this project involve physical testing to validate obstacle detection and line following abilities further. Specifically, different obstacles varying in size, colour, and movement will be analyzed. The line-follow testing will evaluate how well the rack can follow a line, recognize turns, and return to the path after avoiding an obstacle. Additionally, testing how the system performs if a checkpoint is missed must be considered.

All preliminary code will need to be refined and synthesized as object detection, and the line following code is separate. It is expected that this process will be arduous and will require debugging. The camera calibration processes will need to be documented so that future users can follow the necessary steps. Currently, it is uncertain how frequently the camera will need to be calibrated. Lighting and environment changes may influence calibration; however, it is uncertain how sensitive the camera model is to these conditions. It is expected testing will reveal this behaviour. A concise guide outlining the calibration procedure will be produced.

11. Conclusions

The final design solution solves the requirements for obstacle detection, navigation, and checkpointing, which in turn satisfies the client's needs. The solution is comprised of the stereo camera, the Jetson Nano microcontroller, QR codes, and the yellow tape line. Apart from physical components, the operational logic is included within this design solution as well. This is comprised of the computational methods detailed in sections 5 and 6. The stereo camera alone collects the necessary information to perform the required functionalities. This design eliminates the need for a variety of complex analytical instruments, thereby minimizing cost and onboard processing requirements. The client, SigmaPoint, approves of the solution and will provide funding for the project in future prototyping.

References

[1] “What does ESD SAFE mean.” https://kb.hakkousa.com/Knowledgebase/10094/What-does-ESD-SAFE-mean (accessed Oct. 01, 2021).

[2] R. B. Fisher and K. Konolige, “Range Sensors,” in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Berlin, Heidelberg: Springer, 2008, pp. 521–542. doi: 10.1007/978-3-540-30301-5_23.

[3] W. Says, “Types of Arduino Boards : Working and Their Comparision,” *ElProCus - Electronic Projects for Engineering Students*, Sep. 06, 2016. https://www.elprocus.com/different-types-of-arduino-boards/ (accessed Oct. 01, 2021).

[4] F. Gul, W. Rahiman, and S. S. Nazli Alhady, “A comprehensive study for robot navigation techniques,” *Cogent Engineering*, vol. 6, no. 1, p. 1632046, Jan. 2019, doi: 10.1080/23311916.2019.1632046.

[5] “Beginner’s guide to depth (Updated),” *Intel® RealSense^{TM} Depth and Tracking Cameras*, Jul. 16, 2019. https://www.intelrealsense.com/beginners-guide-to-depth/ (accessed Oct. 25, 2021).

[6] K. Ambrosch, M. Humenberger, S. Olufs, and S. Schraml, “Embedded Stereo Vision,” in *Smart Cameras*, A. N. Belbachir, Ed. Boston, MA: Springer US, 2009, pp. 137–157. doi: 10.1007/978-1-4419-0953-4_8.

[7] A. Kadambi, A. Bhandari, and R. Raskar, “3D Depth Cameras in Vision: Benefits and Limitations of the Hardware,” in *Computer Vision and Machine Learning with RGB-D Sensors*, L. Shao, J. Han, P. Kohli, and Z. Zhang, Eds. Cham: Springer International Publishing, 2014, pp. 3–26. doi: 10.1007/978-3-319-08651-4_1.

[8] “ToF (Time of Flight) vs Stereo Vision - 3D Imaging Technology Comparison,” *Latest Open Tech From Seeed*, Jul. 03, 2020. https://www.seeedstudio.com/blog/2020/07/03/3d-imaging-technology-comparison-tof-time-of-flight-vs-stereo-vision-m/ (accessed Oct. 25, 2021).

[9] “Intel® RealSense^{TM} Depth Camera D435.” https://store.intelrealsense.com/buy-intel-realsense-depth-camera-d435.html (accessed Dec. 09, 2021).

[10] D. Esparza and G. Flores, “The STDyn-SLAM: A stereo vision and semantic segmentation approach for SLAM in dynamic outdoor environments,” *arXiv:2010.09857 [cs]*, Mar. 2021, Accessed: Dec. 09, 2021. [Online]. Available: http://arxiv.org/abs/2010.09857

[11] “Understand and Apply Stereo Rectification for Depth Maps (Part 2),” *andreasjakl.com*, Dec. 02, 2020. https://www.andreasjakl.com/understand-and-apply-stereo-rectification-for-depth-maps-part-2/ (accessed Dec. 09, 2021).

[12] “Google Colaboratory.” https://colab.research.google.com/drive/1kMliQEzh6kASwOK562zvi6S_JA8WERyn?usp=sharing (accessed Dec. 09, 2021).

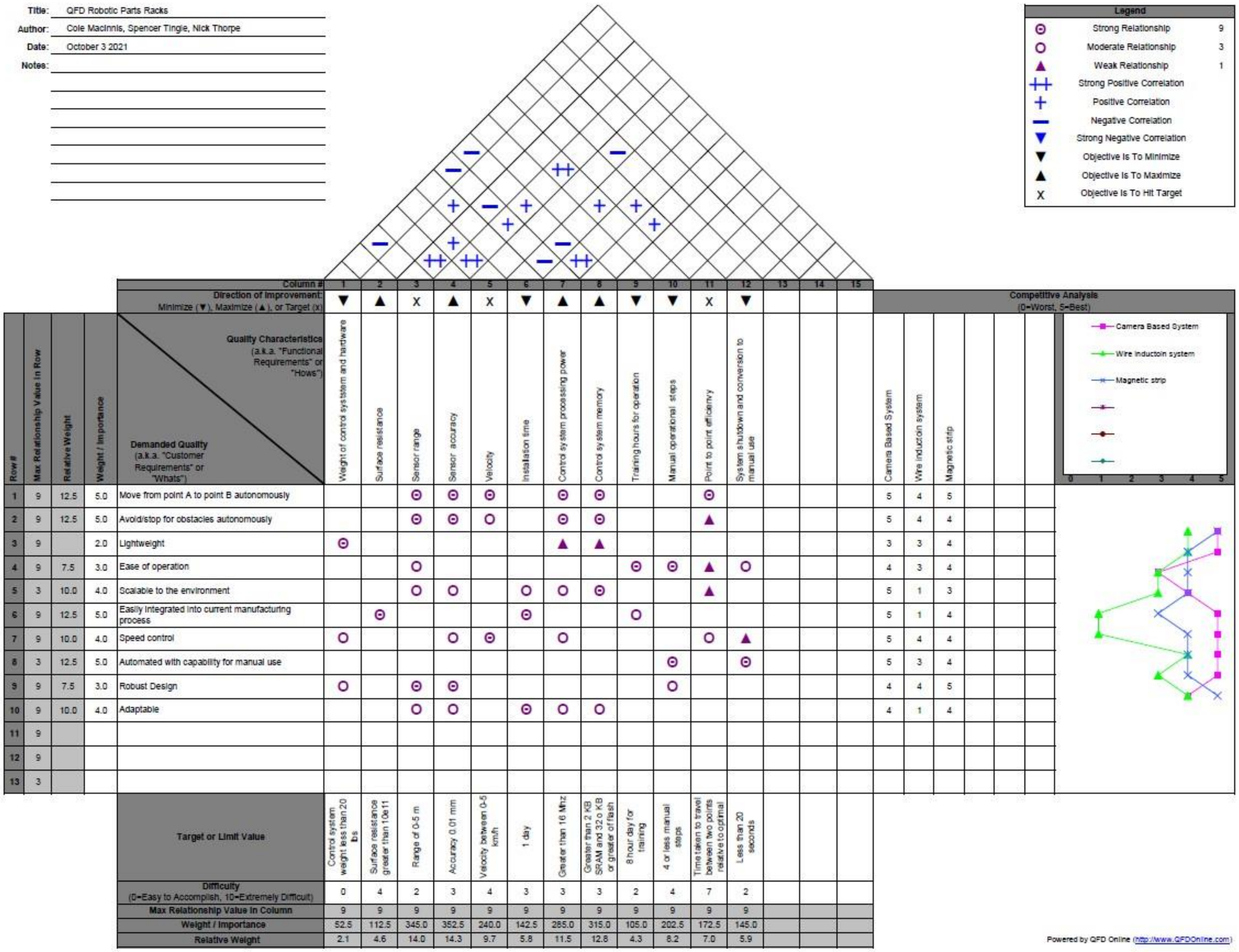
[13] “StereoSGBM (OpenCV 3.4.16 Java documentation).” https://docs.opencv.org/3.4/javadoc/org/opencv/calib3d/StereoSGBM.html (accessed Dec. 09, 2021).

[14] “NVIDIA Jetson Nano Developer Kit-B01.” https://www.seeedstudio.com/NVIDIA-Jetson-Nano-Development-Kit-B01-p-4437.html (accessed Dec. 09, 2021).

[15] “Floor Marking Ideas for Warehouses,” *Creative Safety Supply*. https://www.creativesafetysupply.com/articles/floormarking-ideaswarehouses/ (accessed Nov. 24, 2021).

[16] “3M Floor and Safety Marking Tapes | 3M.” https://www.3m.com/3M/en_US/facility-safety-us/solutions/floor-safety/floor-and-safety-marking-tapes/ (accessed Nov. 24, 2021).

Appendix A – Quality Function deployment



Title: QFD Robotic Parts Racks

Author: Cole MacInnis, Spencer Tingle, Nick Thorpe

Date: October 3 2021

Notes:

Legend		
⊙	Strong Relationship	9
○	Moderate Relationship	3
▲	Weak Relationship	1
++	Strong Positive Correlation	
+	Positive Correlation	
−	Negative Correlation	
▼	Strong Negative Correlation	
▼	Objective Is To Minimize	
▲	Objective Is To Maximize	
x	Objective Is To Hit Target	

Column #	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Direction of Improvement: Minimize (▼), Maximize (▲), or Target (x)	▼	▲	x	▲	x	▼	▲	▲	▼	▼	x	▼			
Quality Characteristics (a.k.a. "Functional Requirements" or "Hows")	Weight of control syststem and hardware	Surface resistance	Sensor range	Sensor accuracy	Velocity	Installation time	Control system processing power	Control system memory	Training hours for operation	Manual operational steps	Point to point efficienvy	System shutdown and conversion to manual use			

Row #	Max Relationship Value in Row	Relative Weight	Weight / Importance	Demanded Quality (a.k.a. "Customer Requirements" or "Whats")	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Camera Based System	Wire inductoin system	Magnetic strip
1	9	12.5	5.0	Move from point A to point B autonomously			⊙	⊙	⊙		⊙	⊙			⊙					5	4	5
2	9	12.5	5.0	Avoid/stop for obstacles autonomously			⊙	⊙	○		⊙	⊙			▲					5	4	4
3	9		2.0	Lightweight	⊙						▲	▲								3	3	4
4	9	7.5	3.0	Ease of operation			○						⊙	⊙	▲	○				4	3	4
5	3	10.0	4.0	Scalable to the environment			○	○		○	○	⊙			▲					5	1	3
6	9	12.5	5.0	Easily integrated into current manufacturing process		⊙				⊙			○							5	1	4
7	9	10.0	4.0	Speed control	○			○	⊙		○				○	▲				5	4	4
8	3	12.5	5.0	Automated with capability for manual use										⊙		⊙				5	3	4
9	9	7.5	3.0	Robust Design	○		⊙	⊙						○						4	4	5
10	9	10.0	4.0	Adaptable			○	○		⊙	○	○								4	1	4
11	9																					
12	9																					
13	3																					

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Target or Limit Value	Control system weight less than 20 lbs	Surface resistance greater than 10e11	Range of 0-5 m	Accuracy 0.01 mm	Velocity between 0-5 km/h	1 day	Greater than 16 Mhz	Greater than 2 KB SRAM and 32 o KB or greater of flash	8 hour day for training	4 or less manual steps	Time taken to travel between two points relative to optimal	Less than 20 seconds			
Difficulty (0=Easy to Accomplish, 10=Extremely Difficult)	0	4	2	3	4	3	3	3	2	4	7	2			
Max Relationship Value in Column	9	9	9	9	9	9	9	9	9	9	9	9			
Weight / Importance	52.5	112.5	345.0	352.5	240.0	142.5	285.0	315.0	105.0	202.5	172.5	145.0			
Relative Weight	2.1	4.6	14.0	14.3	9.7	5.8	11.5	12.8	4.3	8.2	7.0	5.9			

Powered by QFD Online (http://www.QFDOnline.com)

Figure 17: Quality function deployment comparing design alternatives.

Appendix B – Logic System Flow Chart

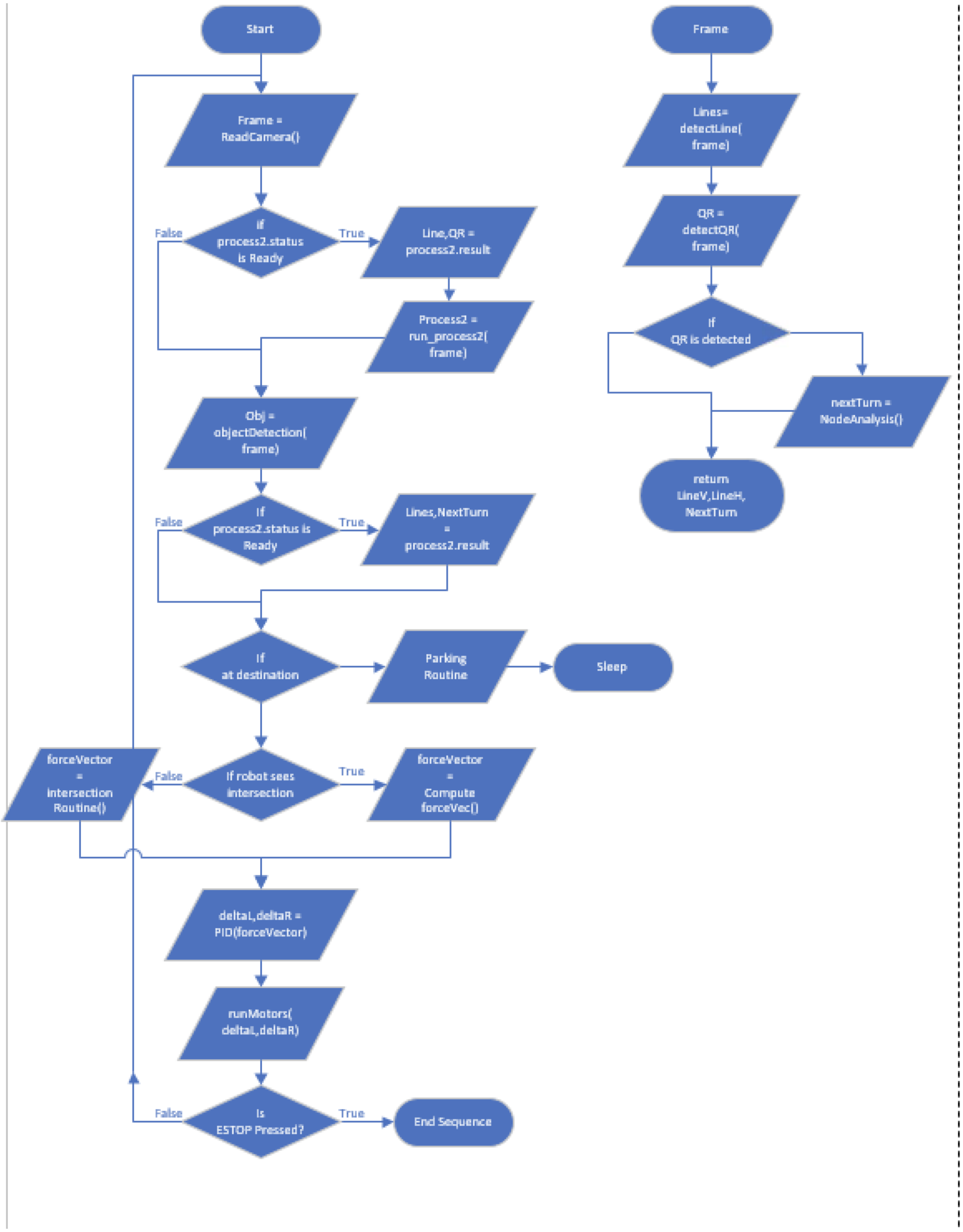

Start
Frame = ReadCamera()
If process2.status is Ready
False
True
Line,QR = process2.result
Process2 = run_process2(frame)
Obj = objectDetection(frame)
If process2.status is Ready
False
True
Lines,NextTurn = process2.result
If at destination
Parking Routine
Sleep
If robot sees intersection
False
True
forceVector = intersection Routine()
forceVector = Compute forceVec()
deltaL,deltaR = PID(forceVector)
runMotors(deltaL,deltaR)
Is ESTOP Pressed?
False
True
End Sequence
Frame
Lines= detectLine(frame)
QR = detectQR(frame)
If QR is detected
nextTurn = NodeAnalysis()
return LineV,LineH, NextTurn

Figure 18: Main loop. Structured around the straightaway routine with parking and intersection routines called when the appropriate flag is thrown.

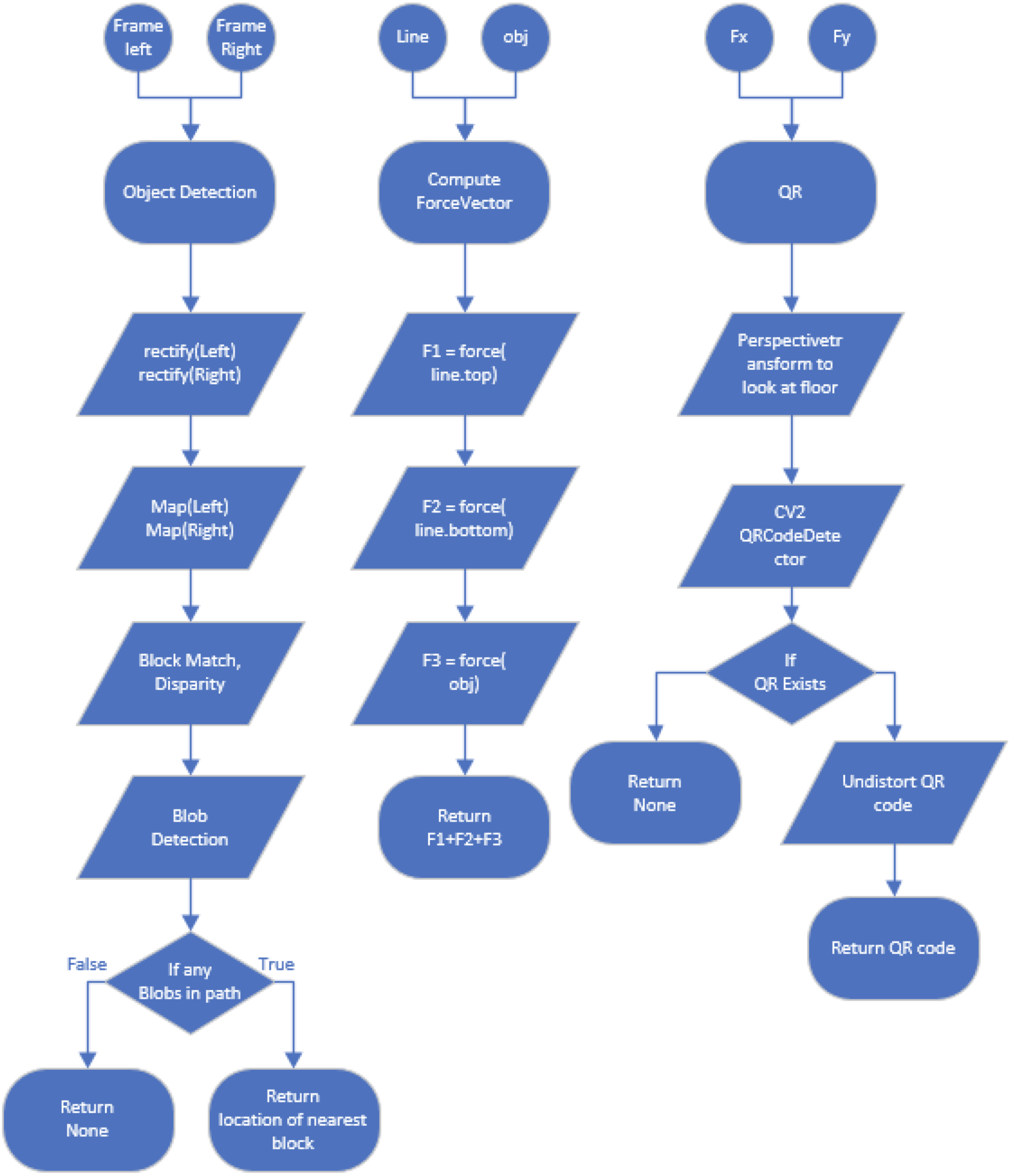


Figure 19: Function definitions used in main loop

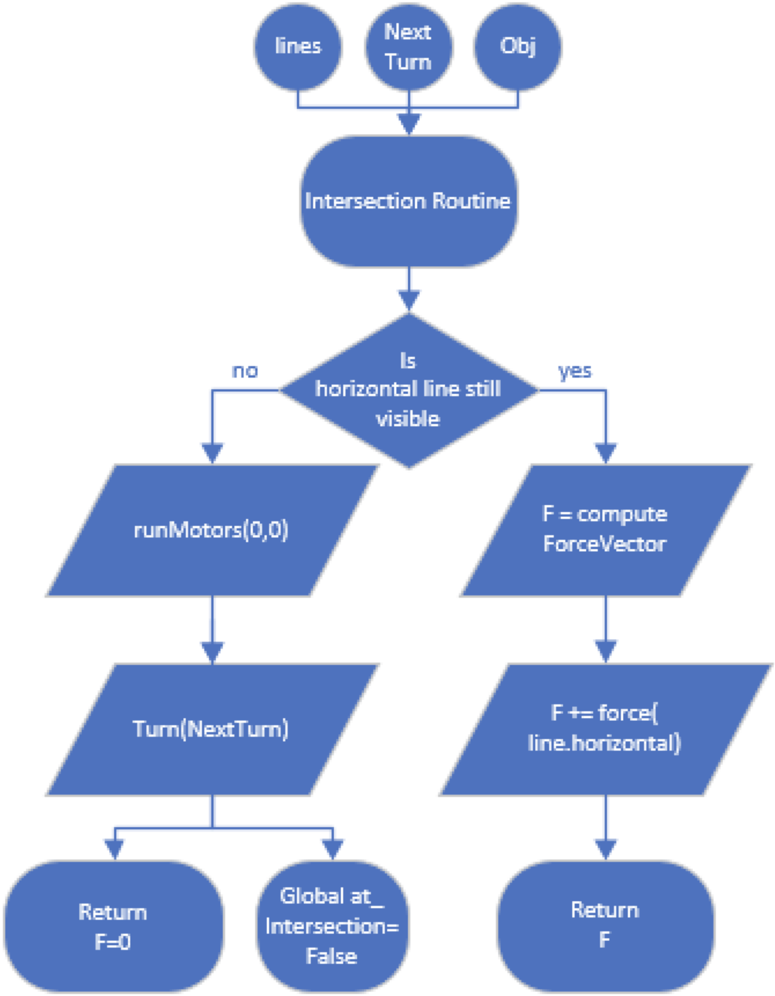
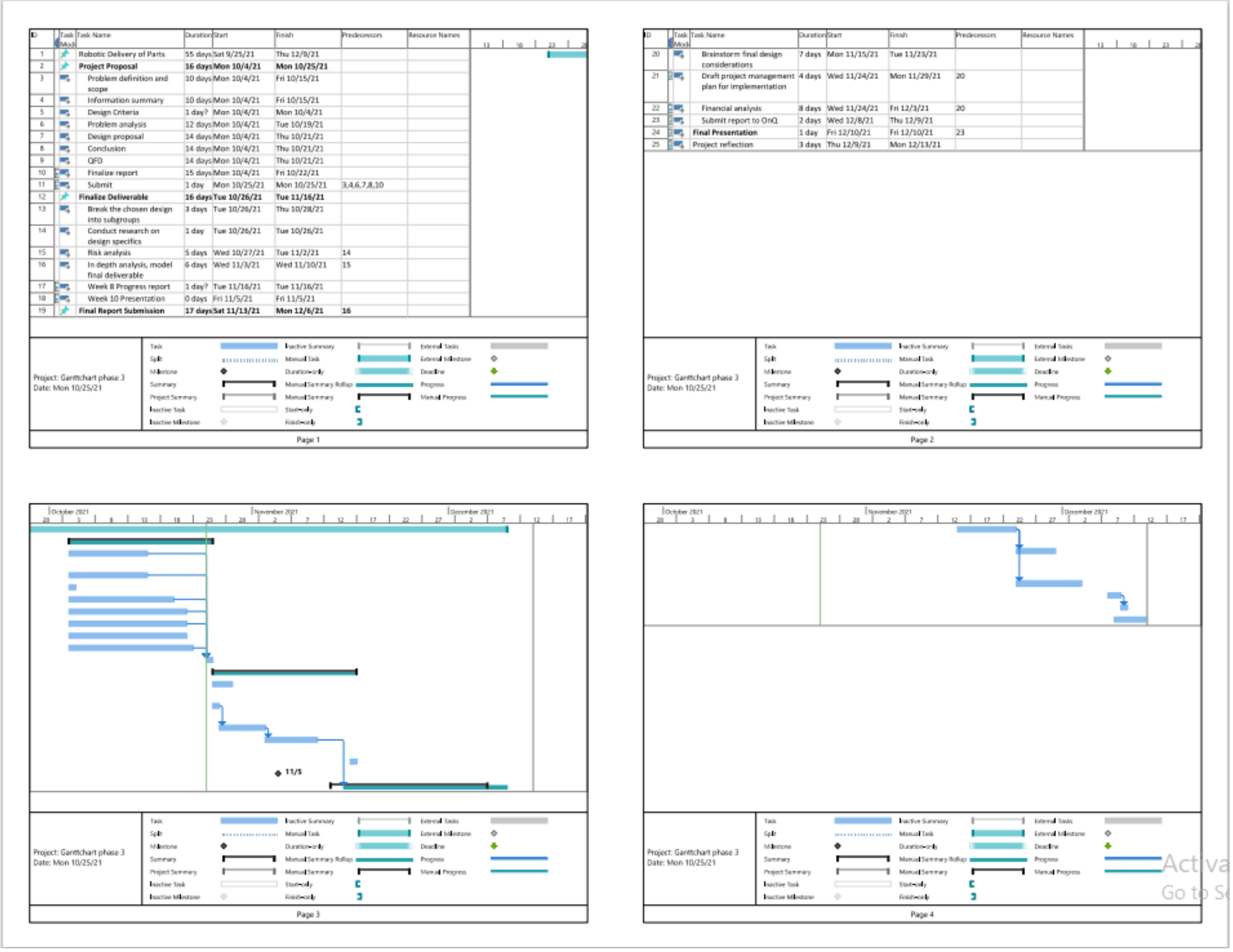


Figure 20: Intersection Routine

Appendix C – Gantt Chart



ID	Task Mode	Task Name	Duration	Start	Finish	Predecessors	Resource Names
1		Robotic Delivery of Parts	55 days	Sat 9/25/21	Thu 12/9/21		
2		**Project Proposal**	**16 days**	**Mon 10/4/21**	**Mon 10/25/21**		
3		Problem definition and scope	10 days	Mon 10/4/21	Fri 10/15/21		
4		Information summary	10 days	Mon 10/4/21	Fri 10/15/21		
5		Design Criteria	1 day?	Mon 10/4/21	Mon 10/4/21		
6		Problem analysis	12 days	Mon 10/4/21	Tue 10/19/21		
7		Design proposal	14 days	Mon 10/4/21	Thu 10/21/21		
8		Conclusion	14 days	Mon 10/4/21	Thu 10/21/21		
9		QFD	14 days	Mon 10/4/21	Thu 10/21/21		
10		Finalize report	15 days	Mon 10/4/21	Fri 10/22/21		
11		Submit	1 day	Mon 10/25/21	Mon 10/25/21	3,4,6,7,8,10	
12		**Finalize Deliverable**	**16 days**	**Tue 10/26/21**	**Tue 11/16/21**		
13		Break the chosen design into subgroups	3 days	Tue 10/26/21	Thu 10/28/21		
14		Conduct research on design specifics	1 day	Tue 10/26/21	Tue 10/26/21		
15		Risk analysis	5 days	Wed 10/27/21	Tue 11/2/21	14	
16		In depth analysis, model final deliverable	6 days	Wed 11/3/21	Wed 11/10/21	15	
17		Week 8 Progress report	1 day?	Tue 11/16/21	Tue 11/16/21		
18		Week 10 Presentation	0 days	Fri 11/5/21	Fri 11/5/21		
19		**Final Report Submission**	**17 days**	**Sat 11/13/21**	**Mon 12/6/21**	**16**	
20		Brainstorm final design considerations	7 days	Mon 11/15/21	Tue 11/23/21		
21		Draft project management plan for implementation	4 days	Wed 11/24/21	Mon 11/29/21	20	
22		Financial analysis	8 days	Wed 11/24/21	Fri 12/3/21	20	
23		Submit report to OnQ	2 days	Wed 12/8/21	Thu 12/9/21		
24		**Final Presentation**	1 day	Fri 12/10/21	Fri 12/10/21	23	
25		Project reflection	3 days	Thu 12/9/21	Mon 12/13/21		

Project: Ganttchart phase 3
Date: Mon 10/25/21

Activa
Go to Se

Figure 21: Gantt Chart detailing the project timeline.