ECE 445

SENIOR DESIGN LABORATORY

FINAL REPORT

Automated guided vehicle for cargo delivery in factories

ZHENGJIE WANG
(zw65@illinois.edu)
XUHONG HE
(xuhongh2@illinois.edu)
YUYI AO
(yuyiao2@illinois.edu)
QIQIAN FU
(qiqianf2@illinois.edu)

TA: Haoran Cui, Yanbing Yang

May 18, 2025

Abstract

Cargo delivery in large factories faces challenges such as low efficiency and safety risks associated with manual labor. To address these issues, we design an automated guided vehicle (AGV) to reduce labor costs and improve safety in the delivery process. Our AGV system comprises the following components: a navigation system, an obstacle avoidance system, a communication subsystem, and a mechanical and electrical control system.

This project implements a prototype autonomous guided vehicle (AGV) capable of picking up and placing cargo within a mock warehouse environment while supporting real-time status updates. The system is built around a Jetsen Orin Nano running ROS, which operates motor control unit via UART. External commands are sent through an MQTT broker and AGV status is managed using a lightweight SQLite database.

For navigation, the AGV uses SLAM algorithm with LiDAR. A Dijkstra-based path planner and motor controller enable the AGV to follow paths smoothly. Obstacle detection is handled by ultrasonic sensors that trigger emergency stops when people walk around.

Contents

1 Introduction

1.1 Problem

Cargo delivery in large factories has long been a problem due to the low efficiency, restricted working period, and high safety risks caused by manual labor. Human-operated systems, such as forklifts, often results in delays, errors, and accidents, as evaluated by injuries linked to forklifts in the U.S[1]. Human-operated vehicles or manual transport methods often result in delays, errors, and inconsistent performance, especially in complex factory surroundings. Workers' working hours are limited and therefore cannot maintain a 24/7 operation. Additionally, heavy machinery, narrow transporting channels, and dynamic obstacles increase the risk of accidents and injuries. To solve these problems, factory owners and researchers want to design a kind of automated guided vehicle that performs better than manual labor. These challenges call for the rising demand of automated guided vehicles (AGVs), which will eliminate human error, reduce labor costs by 20–30% [2], and operate continuously day and night.

1.2 Solution

An automated guided vehicle needs to be designed and assembled. The vehicle needs to deliver cargo within a large factory. The vehicle needs to be equipped with a control/navigation and obstacle avoidance system. This system ensures that the vehicle can move to the ordered destination by itself safely. Moreover, the vehicle needs to lift goods weighted at least 10kg. This AGV frees workers from moving the goods from point to point, and the only job they need to do is to place the goods in the correct position when the AGV reaches its destination. To accomplish these functions, five parts will be built and assembled onto a manual forklift to transform the cart into an AGV. The main system is a Raspberry Pi, serving as a core processor of any signals. Users can access the AGV from a distance, and the order assigned by the user will be transferred to the main system through the instruction subsystem, which contains database to provide the user with the available positions the AGV can go to. Once the AGV is in motion, another two subsystems will start functioning. The navigation subsystem employs laser radar and SLAM reconstruction positioning to navigate the route, and the obstacle avoidance subsystem uses ultrasonic sensor to detect any possible obstacles and give instant signal to avoid them. These subsystems communicate with the main system, and after processing, the Raspberry Pi will send signals to the mechanical and electrical subsystems to adjust the power supply of each motion part, controlling the motion of the cart.

1.3 Visual Aid

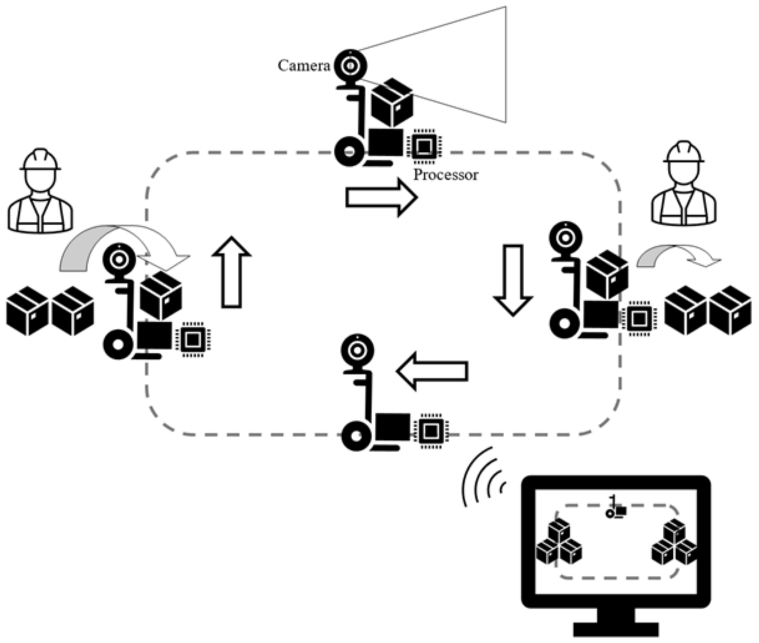


Figure 1: AGV System Overview

The application scenarios and functionality of our AGV are illustrated in Figure 1. In factory environments requiring cargo transportation, the AGV can be deployed to reduce manual labor and minimize workforce requirements. When a cargo transportation task is initiated, the computer sends a command to the AGV, prompting it to follow a route generated by the AGV's main system.

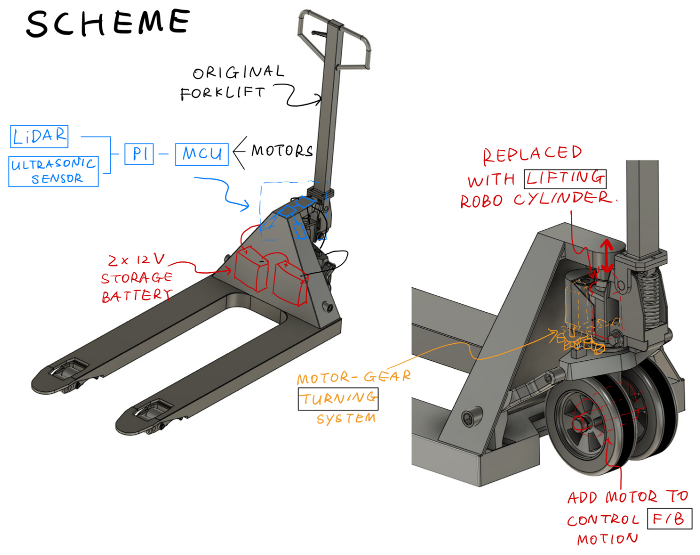

SCHEME
ORIGINAL FORKLIFT
LiDAR
ULTRASONIC SENSOR
PI
MCU
MOTORS
2x 12V STORAGE BATTERY
REPLACED WITH LIFTING ROBO CYLINDER.
MOTOR-GEAR TURNING SYSTEM
ADD MOTOR TO CONTROL F/B MOTION

Figure 2: AGV Scheme

The design schematic of our redesigned forklift is shown in Figure 2. For mechanical modifications, three motors were added to automate motions such as moving, turning, and lifting, which previously required manual operation. A gearbox was also installed adjacent to the turning shaft to transmit motion. All motors are powered by two 12V rechargeable batteries.

1.4 High-level requirements list

- The AGV should move along the route generated by itself from the start point to the destination.
- The communication system should be able to subscribe the navigation topic and transfer the high-level data to motor data and send it to the mechanical system.
- While moving, the AGV must continuously scan for obstacles and stop immediately if one is detected.
- The mechanical system should provide smooth motion, which means that the motion transmission and change should be continuously processed without interruption by the mechanism itself.
- When handling cargo, the AGV must properly align its fork before lifting or lowering the load.

2 Design

2.1 Block Diagram

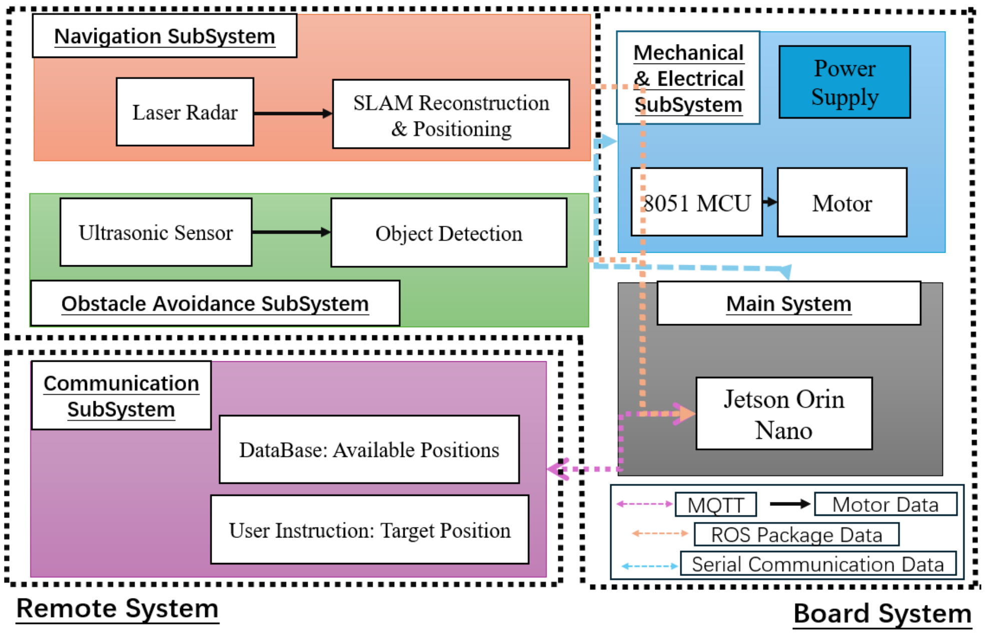


Figure 3: AGV System Block Diagram

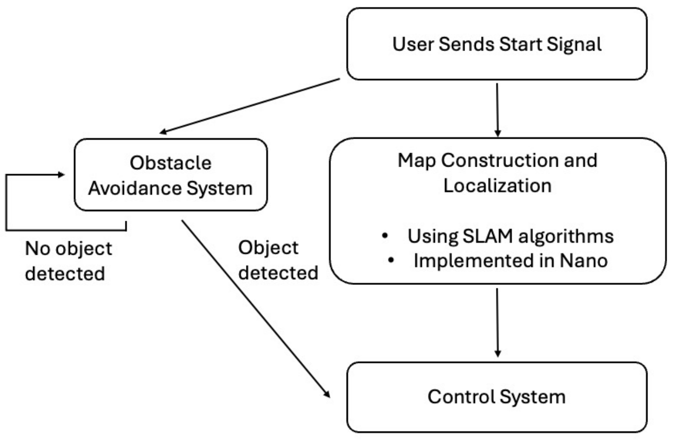


Figure 4: AGV System Overview

2.2 Subsystem Overview

The AGV consists of four modular interconnected subsystems (as shown in Figure 3): Navigation Subsystem, Obstacle Avoidance Subsystem, Communication Subsystem and

Mechanical & Electrical Subsystem. At the core of the whole system is the Jetson Nano, which serves as the primary computing platform responsible for high-level perception, decision-making, and communication coordination. All peripheral modules interface with the Jetson Nano, forming a tightly integrated control architecture.

A laser radar (LiDAR), connected via USB, is used for environment mapping and localization. Based on the processed map and positional information, the navigation stack generates motion commands, including target linear and angular velocities. These commands are refined through the obstacle avoidance subsystem, which utilizes ultrasonic sensors connected to the Jetson Nano through GPIO ports. The ultrasonic sensors provide close-range distance measurements to detect static or dynamic obstacles that may not be captured effectively by the LiDAR.

The finalized movement commands are transmitted through the UART protocol to a microcontroller unit (MCU), which acts as an intermediary between the high-level planner and the low-level actuator drivers. The MCU receives the commands via the UART interface and translates them into electrical signals to control the AGV's motors, thereby executing the desired motion. This layered architecture ensures modularity, real-time responsiveness, and robustness against sensing uncertainties in dynamic factory environments.

2.2.1 Navigation Subsystem

The SLAM (Simultaneous Localization and Mapping) [3] is key for constructing a real-time map of the factory environment while concurrently tracking the vehicle's position within it. By processing data from laser radar, SLAM continuously estimates the vehicle's position and refines the environmental map. This capability provides critical spatial information essential for path planning. This subsystem provides critical spatial information for path planning. Through precise localization and path planning, SLAM enables the forklift to operate autonomously within the factory. Furthermore, the vehicle is programmed to stop immediately once it reaches a predefined proximity (threshold) to its target point. The workflow is shown in Figure 5.

The robotic mapping and navigation include the process of recursive Bayesian estimation. It involves two main steps: prediction and update.

The prediction step updates the belief of the current state based on the previous state and the motion model:

$$\overline{bel}(x_k) = \int p(x_k|x_{k-1}, u_k) bel(x_{k-1}) dx_{k-1} \tag{1}$$

where: x_k is the state at time k; u_k is the control input at time k; $bel(x_{k-1})$ is the belief of state x_{k-1}; $\overline{bel}(x_k)$ is the predicted (prior) belief of state x_k; and $p(x_k|x_{k-1}, u_k)$ is the motion model.

The update step incorporates the latest sensor measurement to refine the predicted belief:

$$bel(x_k) = \eta \, p(z_k|x_k) \overline{bel}(x_k) \tag{2}$$

where: z_k is the measurement at time k; $p(z_k|x_k)$ is the measurement model; $\overline{bel}(x_k)$ is the predicted (prior) belief (from Eq. (1)); $bel(x_k)$ is the updated (posterior) belief of state x_k; and η is a normalization constant.

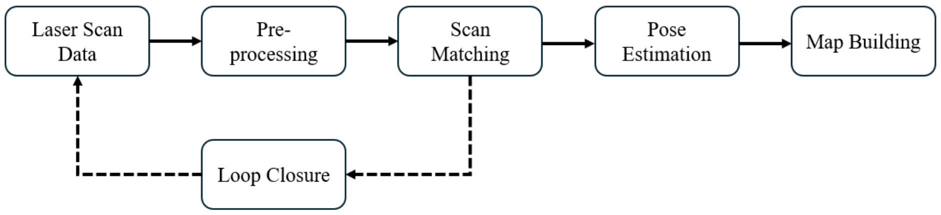


Figure 5: SLAM Scene Construction Block Diagram

2.2.2 Obstacle Avoidance Subsystem

The obstacle avoidance subsystem ensures that the AGV can safely navigate through the factory environment by detecting both static and dynamic obstacles in real time. It employs an ultrasonic sensors mounted at the front of the vehicle. The sensor continuously emit high-frequency sound waves and listen for echoes to determine whether there are nearby obstacles.

The distance to an object is calculated using the time-of-flight principle and the speed of sound at room temperature (assumed to be 20°C). The following formula is used to compute the distance:

$$d = \frac{v \cdot \Delta t}{2} \tag{3}$$

where $v = 331.5 + 0.6T$ (in m/s) is the speed of sound at temperature $T = 20°C$, and Δt is the measured round-trip time. Substituting $T = 20$, we obtain $v = 343\,\text{m/s}$ or equivalently $0.0343\,\text{cm}/\mu s$, resulting in:

$$d \approx 0.0343 \cdot \Delta t \,(\text{in cm}) \tag{4}$$

The detection logic considers a threshold: 200 cm. If any reading falls below this threshold, the system immediately sends a stop signal to the AGV controller. This mechanism allows the vehicle to halt before a collision occurs.

A known limitation of ultrasonic sensors is their dependence on receiving a reflected echo. In certain conditions, such as irregular surfaces or high-absorption materials, the sensor may not receive a valid echo. To handle such cases, the system includes a timeout mechanism. If no echo is detected within a predefined time window, the sensor returns a 'None' value, and the system conservatively assumes the path is clear. This design choice favors avoiding false positives that could disrupt AGV movement due to phantom obstacles.

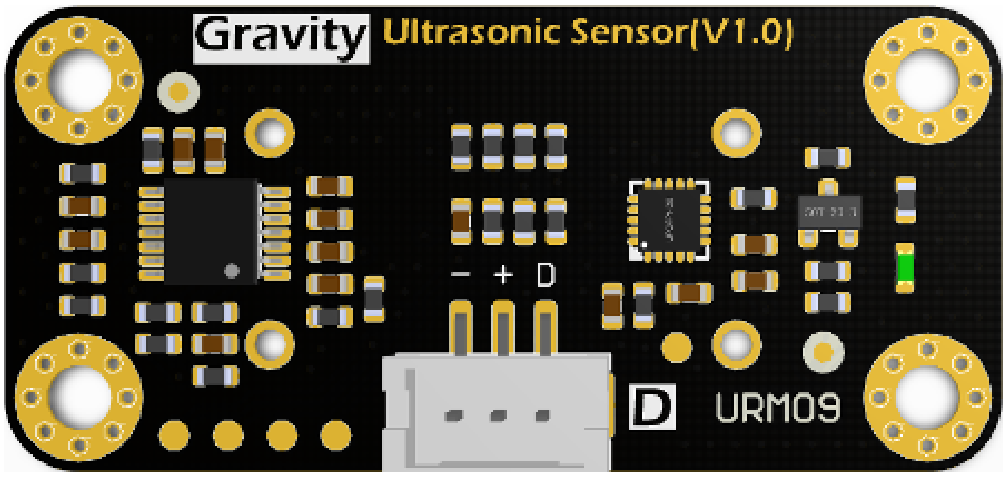


Figure 6: Ultrasonic sensor

2.2.3 Communication Subsystem

The Communication Subsystem handles both external and internal communication tasks required for AGV operation. It allows the AGV to receive commands from the user and send information back to the control center. The overall design is shown in Figure 8.

For external communication, the system uses the Message Queuing Telemetry Transport (MQTT) protocol [4], a lightweight publish-subscribe method in which a central broker forwards messages from publishers to subscribers. The control server, which connects to the broker, stores data such as shelf status in an SQLite database. When a user issues a delivery command, the server publishes a message to the broker, which is received by the AGV to initiate movement.

To quantify communication latency, we model the end-to-end message delay as:

$$T_{\text{msg}} = T_{\text{net}} + T_{\text{proc}} + T_{\text{ack}} \tag{5}$$

where T_{net} is the network delay, T_{proc} is client/server processing time, and T_{ack} is the optional acknowledgment delay. Since both sides run on the same local network and use QoS 0, T_{ack} is negligible. Measurements show:

$$T_{\text{msg}} < 200\,\text{ms} \tag{6}$$

which satisfies the low-latency requirements of real-time AGV control.

The internal communication part connects the AGV's modules. The main control program runs at 100 Hz, receiving velocity updates from the navigation system and parsing them into motion commands—forward, left, right, or stop. These commands are encoded in hexadecimal and sent via the ttyTHS0 serial port to the motor controller.

This subsystem enables reliable, real-time response and is designed to support future upgrades such as more advanced user interfaces, brokers, or distributed databases.

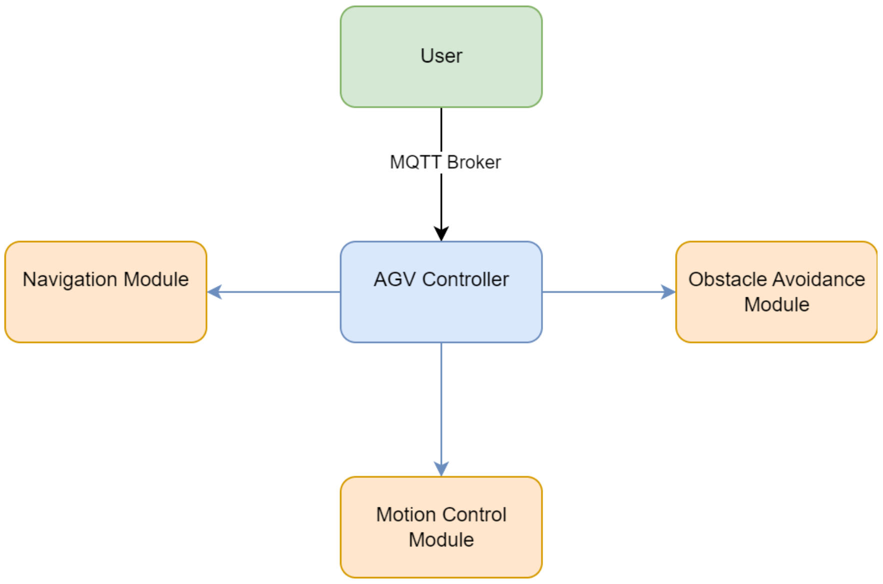


Figure 7: AGV Communication System

2.2.4 Mechanical & Electrical Subsystem

The mechanical and electrical subsystem incorporates three electrical motors, controlling the motion of lifting, turning, and accelerating. The three motors are connected to an STC89C52-RC microcontroller, sharing signals from the main system. Digital and analog signals are transferred and assigned to each of these motors, controlling the motion of the AGV to accomplish the order given by the user or the algorithm in the processor. Detailed schematic diagram is shown in Figure 8.

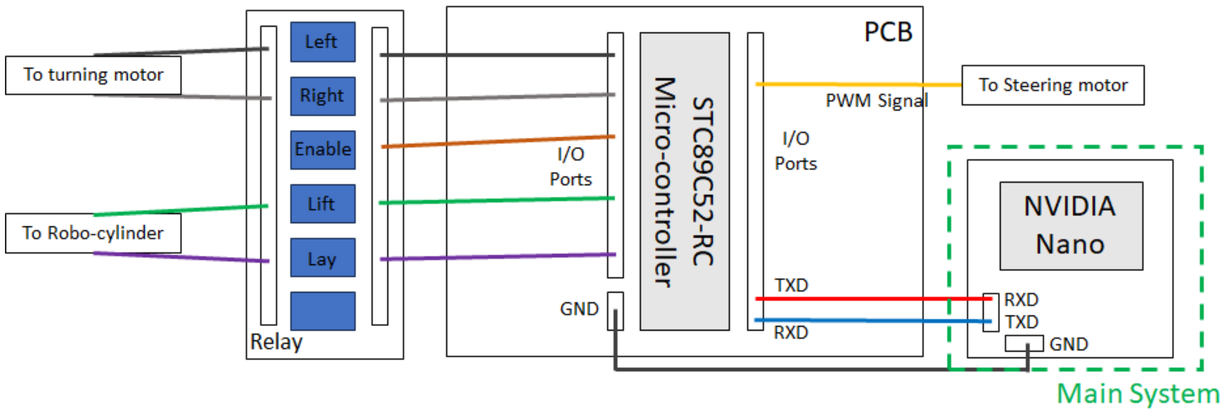


Figure 8: Electric control subsystem scheme

2.3 Subsystem Requirements and Verification

2.3.1 Navigation SubSystem

The navigation subsystem must generate a real-time map with an accuracy of ±10 cm and maintain localization drift under 2% over a 10-meter trajectory. It must process data from laser radar with a refresh rate of at least 10 Hz to ensure efficient responsiveness and store the data locally. The subsystem will provide continuous position updates and map data for route planning. If the system fails to maintain accurate localization and map updates, the vehicle will not be able to navigate autonomously within the factory.

Requirement	**Verification Method**
The SLAM system must be able to build a map of a 12 m × 15 m indoor environment within 5 minutes, with visualization tool detecting the location of the vehicle.	A. Place the vehicle in a known 12 m × 15 m indoor area with measured reference points. B. Start the SLAM system and allow it to build the map. C. Export the map and compare key landmarks to ground truth. D. Move the vehicle to 5 reference points and compare estimated position.
The vehicle must complete autonomous navigation to several random target points on the pre-built map with a 90% success rate, and must detect and avoid static obstacles with 90% success rate.	A. Set 5 navigation goals and let the vehicle follow planned paths; log success/failure and arrival accuracy for 10 total runs. B. Place static obstacles along the path in at least 10 trials. C. Verify that successful arrivals 90% .
Translate high-level velocity commands into motor control actions	Verify the mapping by commanding turns for specific durations and measuring the actual angular change, ensuring it matches the predicted values within an acceptable tolerance.

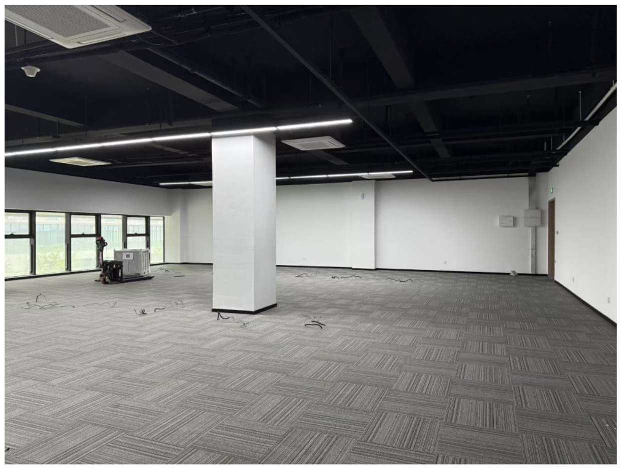


Figure 9: 12m × 15m test environment

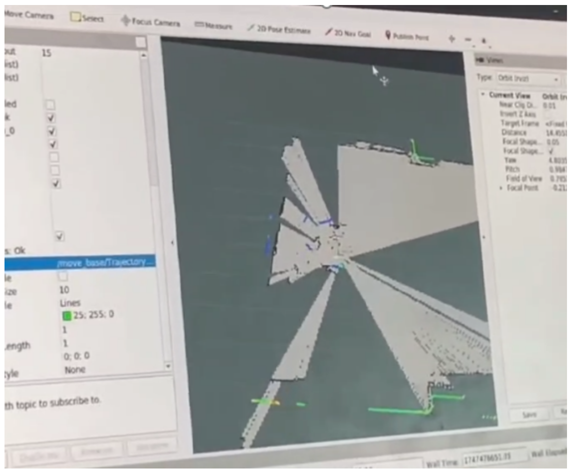


Figure 10: 12m × 15m map construction on Simulator

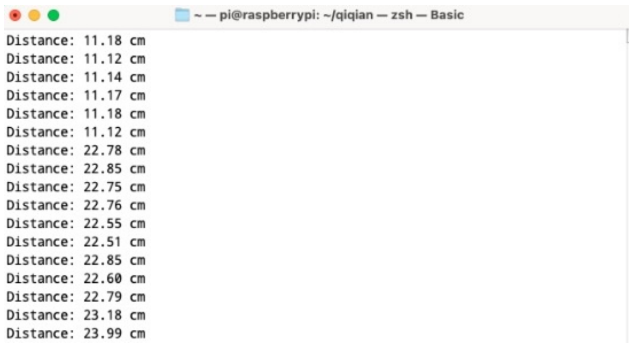
First, the SLAM system is evaluated in a known environment(Figure 9): the vehicle is placed in an indoor area with measured reference points, the SLAM system builds a map(Figure 10), and this map, along with the vehicle's estimated position at several reference points, is compared to ground truth to assess accuracy.

Next, the robot's autonomous navigation performance is tested: multiple navigation goals are set, the vehicle plans and follows paths, and its success rate and arrival accuracy are logged. Static obstacles are then placed along the path in multiple trials to further verify its obstacle avoidance and navigation reliability, with a target success rate of 90

2.3.2 Obstacle Avoidance Subsystem

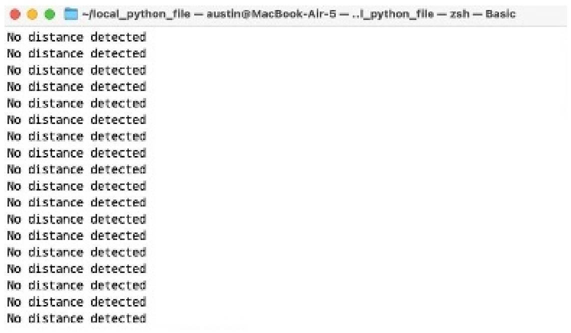
The Obstacle Avoidance Subsystem is responsible for detecting and avoiding both static and dynamic obstacles to ensure safe AGV operation. It continuously scans the surroundings using multiple ultrasonic sensors, which emit high-frequency sound waves and measure the time taken for echoes to return, allowing precise distance estimation. This data is processed to identify obstacles and determine their location relative to the AGV.

Requirement	**Verification Method**
1. Ultrasonic Sensor must detect obstacles within a minimum range of 3 meters . 2. Ultrasonic Sensor can communicate with the control system properly.	1. Place known obstacles at distances of 10, 50, 100, ..., 600 cm and compare sensor readings. 2. Connect the ultrasonic sensor to GPIO pins of the Raspberry Pi and verify signal and echo reception.



```
~ — pi@raspberrypi: ~/qiqian — zsh — Basic
Distance: 11.18 cm
Distance: 11.12 cm
Distance: 11.14 cm
Distance: 11.17 cm
Distance: 11.18 cm
Distance: 11.12 cm
Distance: 22.78 cm
Distance: 22.85 cm
Distance: 22.75 cm
Distance: 22.76 cm
Distance: 22.55 cm
Distance: 22.51 cm
Distance: 22.85 cm
Distance: 22.60 cm
Distance: 22.79 cm
Distance: 23.18 cm
Distance: 23.99 cm
```

Figure 11: Close-range detection performance



```
~/local_python_file — austin@MacBook-Air-5 — ..l_python_file — zsh — Basic
No distance detected
No distance detected
No distance detected
No distance detected
No distance detected
No distance detected
No distance detected
No distance detected
No distance detected
No distance detected
No distance detected
No distance detected
No distance detected
No distance detected
No distance detected
No distance detected
No distance detected
No distance detected
```

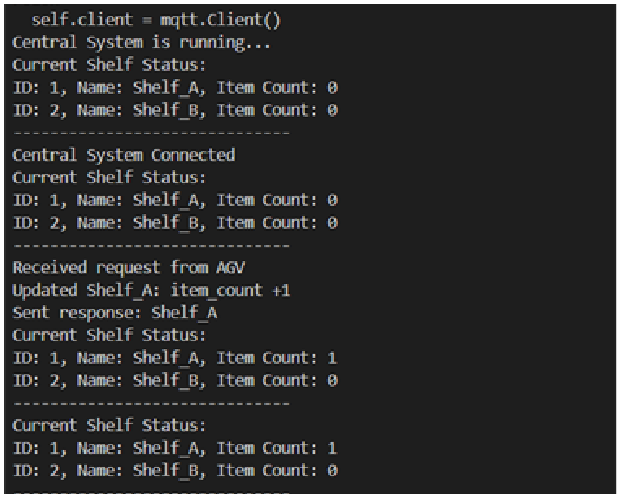
Figure 12: Long-range detection (approx. 6–7 meters)

As shown in Figure 11, the ultrasonic sensor provides accurate and reliable distance measurements at short ranges. At longer distances, specifically beyond 6 meters, loss of echo signals may occur due to signal attenuation or dispersion. However, this limitation does not impact the system's performance, as the obstacle avoidance logic only requires detection within a 2-meter range. Therefore, echo loss at longer ranges is considered acceptable and does not affect operational safety.

2.3.3 Communication SubSystem

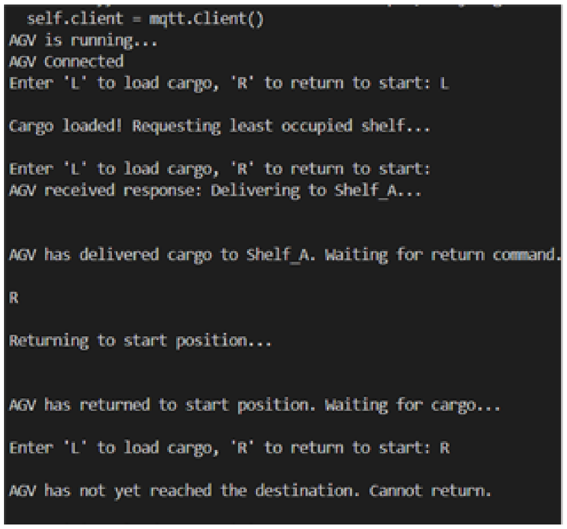
The communication subsystem ensures the data exchange between the AGV and the control center. When a cargo needs to be transferred to its final destination, the control center sends an instruction to the AGV to ensure that it first moves to the starting point, and then proceeds to the final destination and the correct cargo shelf. MQTT is used as the communication protocol between the AGV and the control center. It is a lightweight and low-consumption protocol. Its Quality of Service guarantees reliable message delivery, making it suitable for the AGV system.

Requirement	Verification Method
The communication system should enable the information flow between the vehicle and the central control. This module makes use of an MQTT broker to transfer information.	1. Input a command in the central control system on the computer; the vehicle should receive the start signal and print the information out, which demonstrates its communication ability. 2. The main control logic of the vehicle should receive information from the SLAM algorithm and send a control signal to the control unit. This can be verified by checking whether the received values are correctly printed out in the main control logic.



```
  self.client = mqtt.Client()
Central System is running...
Current Shelf Status:
ID: 1, Name: Shelf_A, Item Count: 0
ID: 2, Name: Shelf_B, Item Count: 0
-----------------------------
Central System Connected
Current Shelf Status:
ID: 1, Name: Shelf_A, Item Count: 0
ID: 2, Name: Shelf_B, Item Count: 0
-----------------------------
Received request from AGV
Updated Shelf_A: item_count +1
Sent response: Shelf_A
Current Shelf Status:
ID: 1, Name: Shelf_A, Item Count: 1
ID: 2, Name: Shelf_B, Item Count: 0
-----------------------------
Current Shelf Status:
ID: 1, Name: Shelf_A, Item Count: 1
ID: 2, Name: Shelf_B, Item Count: 0
```

Figure 13: Update in user side



```
  self.client = mqtt.Client()
AGV is running...
AGV Connected
Enter 'L' to load cargo, 'R' to return to start: L

Cargo loaded! Requesting least occupied shelf...

Enter 'L' to load cargo, 'R' to return to start:
AGV received response: Delivering to Shelf_A...


AGV has delivered cargo to Shelf_A. Waiting for return command.

R

Returning to start position...


AGV has returned to start position. Waiting for cargo...

Enter 'L' to load cargo, 'R' to return to start: R

AGV has not yet reached the destination. Cannot return.
```

Figure 14: High level operations for AGV

Figure 13 and Figure 14 demonstrate the successful communication between the user and the AGV via MQTT protocol. In Figure 13, the user correctly receives a request from the AGV and updates the corresponding database entry (i.e., increasing the item count on the selected shelf). This confirms that the control center can process incoming messages and perform correct actions based on them.

In Figure 14, the AGV responds to high-level commands correctly. The AGV receives instructions from the user and prints out status updates, confirming bidirectional communication. These logs verify the design goal that MQTT reliably enables information flow between the AGV and user in real time.

2.3.4 Mechanical & Electrical SubSystem

The lifting motion's electrical motor, as well as the lifting plate's strength, should be able to provide a force that can at least lift goods of 10 kilograms. The turning part is designed to make a 90-degree turn within a 3 meters radius. The accelerating motor should provide a minimum operating speed of 1 m/s when unloaded, and 0.5 m/s when fully loaded.

Requirement	**Verification Method**
The motion motors should be controlled by the main control system, where signals from the upper computer (Nano) are received.	An STC89C52RC MCU is mounted on the AGV. It receives signals from the upper computer through serial communication, and then an 8-digit binary number will be received and used to determine which switch should be turned on.
The automated guided vehicle should be powered by a stable power source that supports both the motion of motors and the power supply of MCUs.	Two storage batteries are connected in series to provide a 24 V DC voltage for both motors and MCUs. The large capacity of the storage batteries ensures constant and smooth operation for a long time.

The turning motion is achieved by hacking into the original main shaft with a brushless motor added beside it, represented in Figure 15. Since the gear couple was added beside the main shaft to transfer the rotating motion, we tested its stability while functioning, by recording the time-angle relation curve, shown in Figure **??** and Figure 17. We will use the test to determine parameters like max turning time and angle-time relations.

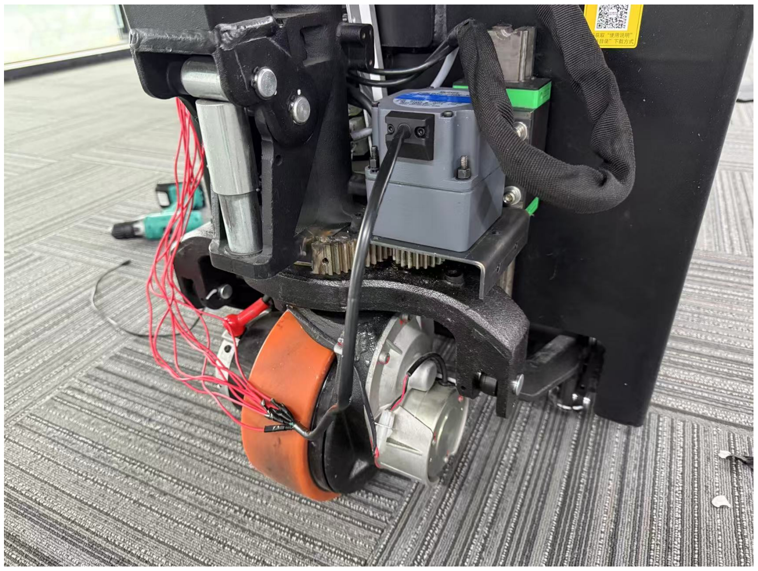


Figure 15: Side motor and gear couple installation

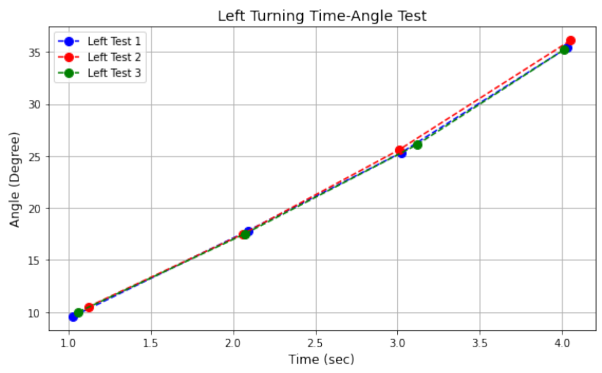


Figure 16: Time-Angle plot for left turning motion

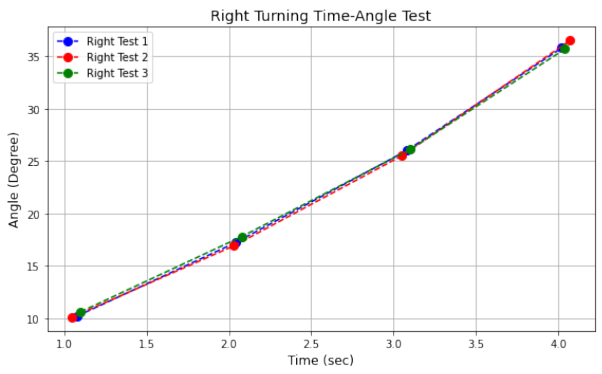


Figure 17: Time-Angle plot for right turning motion

Conclusion can be made from the tests that both left and right turning motion can be regarded at a same rate of angular velocity. The equation fitting these two lines will be

$$\theta = 8.67t + 1.33 \tag{7}$$

where the unit of t is second and θ is degree. We find that there is a instant high angular velocity appears at the starting point. It is because that there is some potential energy stored between the gear couples when the last turning motion stopped, and these potential energy is immediately released when the next turning motion starts, causing a sudden high angular speed.

2.4 Design Alternatives

2.4.1 Communication System

For communication between the AGV and external users (e.g., operator interfaces or central control systems), several protocols were considered, including TCP sockets, HTTP and MQTT. Each has trade-offs in terms of reliability, latency and implementation complexity. Among all the protocols, MQTT best fits our work.

TCP provides reliable and byte-stream-based data transfer, but it requires manual management of message framing, connection handling and retransmission logic. HTTP follows a client–server request–response model and is not suitable for real-time bidirectional messaging.

MQTT was chosen for its lightweight publish–subscribe architecture and robust support for asynchronous communication. It is well-suited for decoupled systems where multiple clients can subscribe to different topics. MQTT's simplicity and its mature client libraries for Python make it ideal for the AGV's real-time task management. Additionally, its compatibility with open-source brokers (such as Mosquitto) made integration and testing easy.

2.4.2 Obstacle Avoidance System

An alternative approach to obstacle detection involves using LiDAR in conjunction with deep learning-based object detection algorithms to identify humans and obstacles in the environment. This method can potentially provide richer semantic understanding and a broader sensing range compared to ultrasonic sensors. However, in the current system, this approach is not adopted due to two key limitations. First, the LiDAR unit employed lacks sufficient resolution and point density to support reliable object classification. Second, deep learning models would require dedicated training on factory-specific obstacle categories to ensure acceptable accuracy, which is not feasible within the current project scope. Nonetheless, this approach remains a promising direction for future enhancement, particularly in scenarios requiring human-aware navigation or higher-level semantic perception.

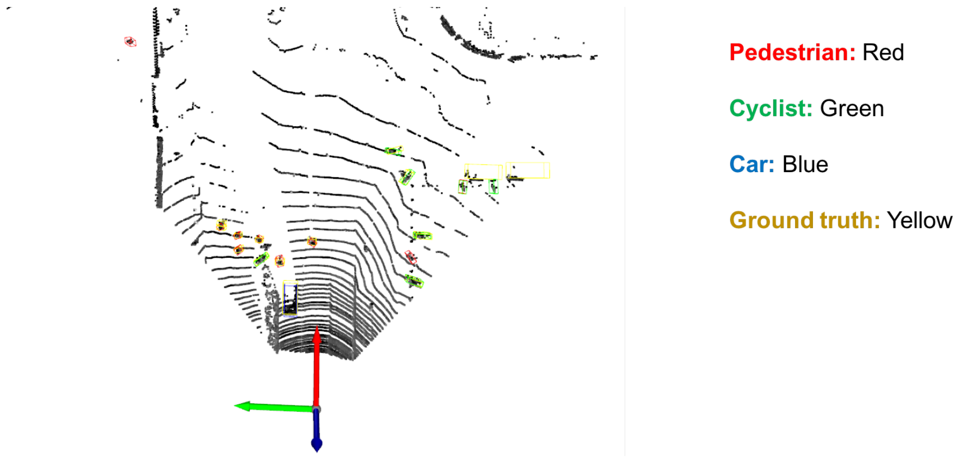


Figure 18: 3D Lidar Detection

2.4.3 Mechanical System

Our solution of fixing the large semi-ring gear to the main shaft is welding. However, a general approach to this fixing issue is use screw fix, which is easy and cheap, and is convenient for us to depart the gear from the shaft while testing. Actually, screw fix is the way we initially used to combine the gear to the shaft, but it turns out to be unstable while the turning motor operating. The semi-ring gear will deviate from its original position, shown in Figure 19, caused by the weak fixing strength provided by a single screw. Due to the geometry of the original forklift, it is impossible to add another screw in the opposite direction of the current one, as a result, the unbalanced screw forces causes the semi-ring gear to slip. Finally, although welding can cost more money and time, we had to choose this way to fix the gear to the main shaft.

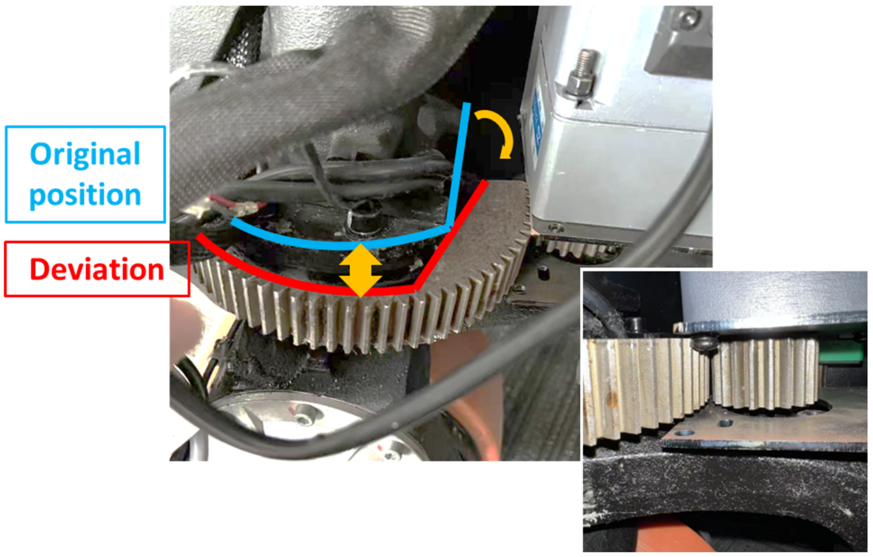


Figure 19: Deviation of the semi-ring gear

2.5 Tolerance Analysis

A critical risk in using ultrasonic sensors for obstacle detection is measurement accuracy and response time, particularly in a dynamic factory environment where reflective surfaces, environmental noise, and sensor blind spots could affect performance. The system relies on three ultrasonic sensors positioned at the front, front-left, and front-right of the AGV to detect obstacles in multiple directions. This configuration allows the AGV to identify obstacles in its path and react accordingly by stopping when an object is detected within a 50 cm range. However, ultrasonic sensors may suffer from signal reflection and absorption issues, especially in environments with metallic surfaces or irregularly shaped objects. Reflections from highly smooth surfaces can cause false detections, while soft materials may absorb the sound waves, leading to undetected obstacles. Additionally, cross-interference between the three ultrasonic sensors could result in inaccurate readings, especially if they operate at similar frequencies.

To mitigate these risks:

- The sensors should operate at slightly different frequencies or incorporate sequential triggering to minimize interference.
- Signal filtering techniques, such as median filtering, should be implemented to discard outlier readings caused by reflections.
- The system should integrate redundant checks where an obstacle is only confirmed if detected by at least two consecutive reads within 50 ms to reduce false positives.

3 Cost and Schedule

3.1 Cost analysis

3.1.1 Labor Cost

For this project, we assume each team member is compensated at a rate of **$35/hour**, which aligns with typical internship or early career salaries for ECE graduates at the University of Illinois.

Each member works **15 hours/week** over **14 weeks**, and a multiplier of **2.5** is applied to account for meetings, documentation, testing, and project management.

$$\text{Cost}_{\text{individual}} = \$35\ /\text{hr} \times 15\ \text{hrs/week} \times 14\ \text{weeks} \times 2.5 = \textbf{\$18,375}$$

There are four members in the team, so:

$$\text{Total Labor Cost} = 18,375 \times 4 = \textbf{\$73,500}$$

Table 1: Personnel Cost Table

Name	**$/Hour**	**Hours/Week**	**Weeks #**	**Multiplier**	**Cost ($)**
Xuhong	35	15	14	2.5	18375
Qiqian	35	15	14	2.5	18375
Yuyi	35	15	14	2.5	18375
Zhengjie	35	15	14	2.5	18375
Total	35	60	14	2.5	**73500**

3.1.2 Manufacturing Prototype Costs

The following parts are used in our project and are critical to implementing the AGV control system, SLAM-based localization communication system and sensor feedback systems:

Table 2: Equipment Cost Table

Name	**Description**	**Quantity**	**Price ($)**	**Cost ($)**
NVidia Jetson Orin Nano	The CPU that is used for the whole system	1	273.97	273.97
RPLIDAR-C1	The laser radar that can be used for SLAM[3] algorithm	1	58.90	58.90
URM09-Trig	The ultrasonic sensors that can be used for obstacle detection	3	8.22	24.66
STC89C52 MCU development board	The MCU used in the main control system	2	10.96	21.92
DS3230 steering engine	The engine used to control the forward/backward motion	1	14.80	14.80

3.2 Schedule

As shown in Tables 3 and 4, we provide a detailed timetable showing when each step of the design will be completed by week. It also includes information about how the tasks are shared among team members.

Table 3: Weekly Schedule: Xuhong He and Qiqian Fu

Date	**Xuhong He**	**Qiqian Fu**
2/24/2025	Study ROS and set up the execution environment	Study the concept of ROS
3/3/2025	Study SLAM[3] algorithm and corresponding package	Study deep learning
3/10/2025	Set up the workspace of the whole project	Study concept of cloud point and relevant models
3/17/2025	Start programming on scene construction using SLAM[3]	Testing how well the laser radar perform using Point Pillar model
3/24/2025	Finish programming on scene construction part using the simulator.	Thinking of other ways to implement obstacle avoidance

3/31/2025	Finish programming on vehicle localization part using the simulator.	Study how ultrasonic sensor works
4/7/2025	Finish programming on route planning part using the simulator.	Writing the code for using ultrasonic sensor to test distance
4/14/2025	Use laser radar to test the scene construction functionality in the real scenario.	Test ultrasonic sensor on Raspberry Pi
4/21/2025	Test the functionality of localization in a real-world scenario.	Study how to run ROS code
4/28/2025	Integrate the whole SLAM system into the main function.	Integrate the obstacle avoidance logit to the whole system
5/5/2025	Test AGV system in real scenario	Mapping the relationship between angular velocity and turning time
5/12/2025	Prepare final demo	Prepare final demo

Table 4: Weekly Schedule: Yuyi Ao and Zhengjie Wang

Date	**Yuyi Ao**	**Zhengjie Wang**
2/24/2025	Study basic knowledge about ROS	Studying the mechanism of original forklift
3/3/2025	Set up workspace on Raspberry Pi	Set forklift 3D model in Autodesk Fusion
3/10/2025	Test MQTT system on computer	Prepare for motion connection remodel
3/17/2025	Connect Raspberry Pi to computer with SSH and start testing MQTT.	Set motion connection remodel to the forklift and test
3/24/2025	Finish programming the server-side and vehicle-side code for the communication system	Adjust motion connection remodel to the forklift and test
3/31/2025	Testing communication between the vehicle and computer	Install the MCU to the forklift and test basic functions
4/7/2025	Coding the main logic function framework of AGV.	Set the main control program to the MCU
4/14/2025	Coding for the function for information communication inside AGV: Gathering movement information in main logic function from SLAM algorithm.	Adjust the main control program of the MCU, with the adjustment of connection remodel part
4/21/2025	Coding for the function: Sending control signal in main logic function to control unit.	Combine the MCU with upper computer to test the control
4/28/2025	Integrate the whole communication system into the main function.	Integrate the whole communication system into the main function.
5/5/2025	Test AGV system in real scenario	Test AGV system in real scenario
5/12/2025	Prepare final demo	Prepare final demo

4 Ethics and Safety

The development and deployment of AGVs for industrial use may raise ethical and safety considerations.

4.1 Ethical Considerations

4.1.1 Public Welfare and Safety

Human safety has a higher importance than AGVs, especially in shared workspaces. This is most important during the whole development process, and should follow the IEEE Code of Ethics, and ACM Code of Ethics and Professional Conduct [5][6]. A breach could lead to accidents or harm due to system failures or inadequate collision avoidance. To avoid this, it is possible to implement safety mechanisms (e.g., LiDAR, emergency stop systems) and evaluate them through limit tests like accelerated scenario-based evaluation.

4.1.2 Data Privacy

AGVs' motion is partly based on collecting operational data (e.g., worker movements), and may have risks in violating privacy if mishandled. A possible solution can be aligning with the client and employer best interests principle [7] and to mitigate such risks by encrypting sensitive data and complying with GDPR or equivalent regulations.

4.2 Safety Standards and Regulatory Compliance

During development, it is significant to note the restrictions published by the local government. Some relative regulations are listed here. ISO 3691-4 specifies AGV safety requirements, including obstacle detection and emergency braking[8]. The National Electrical Safety Code (NESC) ensures safe electrical infrastructure for AGVs, particularly in high-risk environments [9]. ISO 34502 outlines scenario-based safety evaluations for autonomous systems[10].

4.3 Potential Safety Risks

Potential safety risks in this AGV development are mostly about the risk of physical harm, system failures, and environmental hazards. For instance, in the deployment of an autonomous mobile device, there is a significant risk of collisions with humans, particularly vulnerable people such as children and the old, leading to injuries. Similarly, in industrial environments, the use of AGVs may also cause collision risks due to sensor failures or algorithmic errors. Additionally, in high-temperature factory environments, such as those involving heat-treating furnaces, the lack of proper protective gear like heat-resistant gloves can lead to burns or other injuries.

4.4 Implementation Verification

All safety systems undergo validation through accelerated scenario testing that simulates edge cases including sensor failures, high-traffic environments, and extreme operating conditions. Compliance documentation includes ISO standard checklists, encryption audit logs, and training records that collectively demonstrate adherence to the highest safety standards throughout the product lifecycle. This comprehensive safety framework not only protects users and developers but also establishes a model for ethical AGV deployment in complex industrial settings.

5 Conclusion

The development and deployment of AGVs for industrial use may raise ethical and safety considerations.

5.1 Results

This work presents a successfully developed and implemented autonomous mapping and navigation system tailored for factory scenarios. Utilizing advanced SLAM techniques, the system demonstrates efficient real-time environmental map construction—for instance, mapping a **12m x 15m** area typically within **2 minutes**—and achieves precise vehicle localization with less than **3%** drifting distance. Rigorous quantitative evaluations have confirmed its high operational reliability, highlighted by a navigation success rate to arbitrary target points exceeding **90%** within the designated environment. These results establish a robust foundation for dependable autonomous vehicle deployment and significantly enhance operational efficiency.

The AGV prototype also demonstrates a reliable and responsive communication system. External MQTT-based commands show end-to-end latency within **200 ms** over a **5-meter** indoor Wi-Fi connection, with no message loss during over **30 minutes** of continuous operation. Internal UART communication between the main controller and motor MCU runs at **100 Hz**, with each command processed in under **5 ms**. These results confirm that the system meets real-time requirements for low latency and high reliability.

5.2 Future Work

5.2.1 User Interface

We plan to design a user-friendly interface to help workers operate the AGV more easily. The UI will display a visual map of the warehouse, allowing users to select target locations with simple clicks. Task status and the AGV's position will be shown in real time to improve clarity. A web-based interface is a likely option, enabling easy access from tablets or PCs. This addition will enhance overall system usability.

5.2.2 Visual Assistance for Precise Alignment

To enhance the AGV's ability to align with shelves during pickup, future work will introduce visual assistance using cameras. One approach is to mount a camera on each AGV to provide local visual feedback, improving alignment accuracy but increasing hardware cost. Alternatively, fixed cameras can be installed in the warehouse to track AGV movement and send adjustment commands. This reduces cost but requires additional infrastructure. Both methods can improve positioning precision, and the choice depends on budget and deployment needs.

References

[1] OSHA. "Forklift Accident Statistics." (2023), [Online]. Available: https://www.safetymanualosha.com/forklift-fatalities/ (visited on 02/21/2025).

[2] Deloitte. "The Future of Smart Manufacturing." (2021), [Online]. Available: https://www.deloitte.com/global/en/Industries/industrial-construction/perspectives/Digital-transformation-in-the-fom.html (visited on 03/11/2025).

[3] J. A. Placed, J. Strader, H. Carrillo, N. Atanasov, V. Indelman, and L. Carlone, "A survey on active simultaneous localization and mapping: State of the art and new frontiers," *IEEE Transactions on Robotics*, vol. 39, no. 5, pp. 1234–1250, 2023.

[4] OASIS. "MQTT Version 5.0." (2019), [Online]. Available: https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html (visited on 03/18/2025).

[5] ACM. "ACM Code of Ethics." (2018), [Online]. Available: https://www.acm.org/code-of-ethics (visited on 04/12/2025).

[6] IEEE. "IEEE Code of Ethics." (2020), [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html (visited on 04/12/2025).

[7] ACM. "Software Engineering Code of Ethics." (1997), [Online]. Available: https://ethics.acm.org/code-of-ethics/software-engineering-code/ (visited on 04/12/2025).

[8] ISO. "ISO Standard 83545." (2023), [Online]. Available: https://www.iso.org/standard/83545.html (visited on 04/12/2025).

[9] IEEE. "IEEE NESC Standards." (2025), [Online]. Available: https://standards.ieee.org/products-programs/nesc/ (visited on 04/14/2025).

[10] GlobalSpec. "Standard 34502." (2022), [Online]. Available: https://standards.globalspec.com/std/14572778/34502 (visited on 04/14/2025).