



AUTOMATION OF MATERIAL TRANSPORT IN APPAREL INDUSTRY

**A robot based system to autonomously transport
cut parts from fabric storage to sewing line loading
stations.**

A dissertation submitted in partial fulfilment of the
requirement for the award of degree in

Bachelor of Fashion Technology (Apparel Production)

UNDER THE GUIDANCE OF:

Dr. Deepak Panghal

National Institute of Fashion Technology, New Delhi

SUBMITTED BY:

Charuvi Ranjan, BFT/20/3

Shreyash Ojha, BFT/20/41

Department of Fashion Technology
National Institute of Fashion Technology, New Delhi
May, 2024



ABSTRACT

This report details the design, development, and testing of an Automated Guided Vehicle (AGV) system aimed at reducing raw material transport time and the labour required for it, within a sewing floor environment. The project addresses a critical bottleneck identified in the current workflow - **the manual movement of cut fabric bundles from storage racks to sewing line stations. This manual process introduces delays in material delivery, hindering sewing line efficiency and causing idle time for workers.**

The proposed solution utilizes an AGV equipped with navigation capabilities and wireless communication to automate material transport. Key features include:

- **RFID Tag-Based Navigation with Line following algorithm:** The AGV employs strategically placed RFID tags to determine its location and navigate along pre-defined paths, which have been marked with a simple line. This ensures that the AGV maintains a consistent path of movement within the floor.
- **Wireless Demand and Destination Management:** A single human operator interacts with the AGV through a user interface hosted by the robot itself via a wireless network. This interface facilitates the generation of material transport requests to the AGV, eliminating the need for physical intervention or complex scheduling systems.
- **Material Handling:** The AGV is designed to carry fabric bundles effectively and safely, ensuring efficient movement without compromising material integrity.
- **Easy Deployment and Cost-Effectiveness:** The chosen components and technologies prioritize ease of deployment and affordability, making the solution readily adaptable to existing sewing floor infrastructure.

The overall objective of this project is to demonstrate the feasibility and benefits of implementing an AGV system for WIP transport within a sewing floor environment.

Keywords: AGV, raw material transport, line following, RFID tagging



DECLARATION

“This is to certify that this Project Report titled, ‘**ACME: A robot based system to autonomously transport cut parts from fabric storage to sewing line loading stations**’, is based on our, **Charuvi Ranjan and Shreyash Ojha**, original research work, conducted under the guidance of **Dr. Deepak Panghal**, towards partial fulfilment of the requirement for the award of the Bachelor’s Degree in Fashion Technology (Apparel Production), of the National Institute of Fashion Technology, New Delhi.

No part of this work has been copied from any other source. Material, wherever borrowed, has been duly acknowledged.”

CHARUVI RANJAN

SHREYASH OJHA

Name and Signature of the Students

Signature of Faculty mentor



ACKNOWLEDGEMENT

This Graduation Project was a great learning opportunity and we would like to express our deepest gratitude to all the people involved directly or indirectly, in making it possible for us to complete it successfully. We are grateful to NIFT for providing us an opportunity to do research work on '**Automation of Material Transport in Apparel Industry: A robot based system to autonomously transport cut parts from fabric storage to sewing line loading stations**'.

We would like to convey our sincere gratitude to Dr. Deepak Panghal, our Faculty mentor, for his enriching inputs, constant encouragement and valuable support towards the completion of our project.

We would like to thank Mr. Rishikesh Ranjan (IFS, Mining Engineer), Mr. Ekansh Prasad (Automobile Engineer) and Miss Sakshi Bhargava (FC-6, NIFT Delhi) who spared their time and helped us with their views on the structure design and CAD modelling. We would also like to thank Mr. Pawan Kumar (Sr. IE Manager, Vamani Exports) for his views and improvements for the factory floor implementation along with Mr. Aman Dahiya who helped us source parts for the assembly of the robot. Lastly, we are also obliged to the faculty members of the Department of Fashion Technology, New Delhi for providing us with their suggestions and reviewing our work.

Charuvi Ranjan and Shreyash Ojha
Department of Fashion Technology
NIFT, New Delhi



CONTENTS

ABSTRACT.....	2
DECLARATION.....	3
ACKNOWLEDGEMENT.....	4
Introduction and Research Establishment.....	7
CHAPTER 1: Background.....	8
CHAPTER 2: Gap Analysis and Objectives.....	9
2.1 Industry Scenario.....	9
2.1.1 Project Focus.....	10
2.2 Current Factory Floor Scenario.....	10
Major Challenges:.....	11
Considerations for Implementation of new solutions:.....	11
2.3 Objective and Proposed Solution.....	12
2.3.1 Sub-Objectives:.....	12
2.3.2 Research Methodology.....	12
CHAPTER 3: Literature Review.....	13
3.1 Current Scenario of Material Handling Systems in Apparel Industry.....	13
3.1.1 Material Handling Systems in Apparel Manufacturing:.....	13
3.1.2 Case Studies:.....	13
3.1.3 Limitations and Research Gaps:.....	13
3.1.4 Further Research Directions:.....	13
3.2 AGV in Warehousing Setups.....	14
3.3 Types of Material Handling Systems currently in the Apparel Industry.....	15
3.3.1 Introduction:.....	15
3.3.2 Types of Material Handling Systems:.....	15
3.3.3 Emerging Technologies:.....	15
3.3.4 Future Trends:.....	15
3.3.5 Limitations and Research Gaps:.....	15
3.4 Competitive edge of AGVs through Automation (Case Studies).....	16
3.4.1 A study on the Sri Lankan apparel industry (A De Silva, S Gunathilake, H Munaweera, D Perera and L Gunathilake).....	16
3.4.2 Automation in Garment Manufacturing (Rajkishore Nayak and Rajiv Padhye)....	17
3.4.3 Automated transportation systems subject to interruptions in production and intralogistics – a survey and evaluation (Ireneus Wior, Stefan Jerenz and Alexander Fay).....	17
3.5 The Need and Benefits of AGVs in the Apparel Industry.....	18

3.5.1 Need for Automation in the Apparel Industry:.....	18
3.5.2 Benefits of AGV Technology in Various Industries:.....	18
3.5.3 AGV Applications for Optimization and Cost Reduction in Apparel Sewing Floors:..	19
3.6 Comparison of Material Handling Systems: Transport Mechanisms vs. AGVs.....	19
3.6.1 Traditional Transport Mechanisms:.....	19
3.3.2 Limitations of Traditional Systems:.....	20
3.3.3 Case Studies and Examples:.....	20
3.3.4 Research Support:.....	20
3.4 Conclusion.....	20
3.5 History.....	21
3.6 Designing process.....	21
3.7 Movement Modeling.....	22
3.7.1 System Configuration [10].....	22
3.7.2 Kinematic Computation [10].....	23
3.8 Body structure design.....	25
3.9 Reliability Study.....	26
3.9.1 Factors Affecting Reliability.....	26
3.9.2 Reliability Metrics.....	26
3.10 Reliability Calculation.....	27
3.10.1 Designing a Maintenance Schedule.....	28
Methodology: Design and Development of AGV.....	30
CHAPTER 1: Ground Work.....	31
1.1 Functionalities.....	31
1.1.1 Main goals and tasks that the AGV is expected to accomplish.....	31
1.1.2 Autonomous Navigation (RFID & PN532 Reader):.....	31
1.1.3 Payload Handling (20 kg Capacity):.....	31
1.1.4 Station Calling (Style Piece Requirement):.....	31
1.1.5 Home Base Docking (Charging Station):.....	32
1.1.6 Real-time User Interface (Responsive HTML):.....	32
1.1.7 Fixed Path Following (Line Sensor):.....	32
1.1.8 Object Avoidance (Ultrasonic Sensor & Servo Motor):.....	32
1.1.9 Work environment analysis:.....	33
1.2 Navigation Methods Study.....	33
1.2.1 Navigation methods suitable.....	33
1.2.2 Task Automation.....	35
1.3 Background Study.....	39
1.3.1 Objectives and Milestones:.....	39
1.3.2 Why SLAM would not be ideal for our AGV.....	41

1.3.3 Why Magnetic Guide Sensors would not be ideal for our AGV.....	42
1.3.4 Chosen Approach.....	42
1.4 Core logic development (Without SLAM).....	43
1.4.1 Ways to develop line following navigation system.....	43
1.4.2 Ways to feed markers in line following approach.....	45
1.4.3 Defining the route through RFID tags.....	47
1.4.4 Shortest Path Finding Algorithms.....	48
1.5 Controller and Language.....	49
CHAPTER 2: Hardware Design.....	51
2.1 System Configuration.....	51
2.2 Vehicle Specifications.....	51
2.2.1 Dimensions of the AGV.....	51
2.3 Calculation for Parts selection.....	53
2.3.1 Motors.....	53
2.3.2 Voltage requirement/ Battery.....	55
2.3.3 Calculating Total Power Consumption:.....	55
CHAPTER 3: Software Design.....	57
3.1 ESP Directory.....	59
3.2 Mega Directory.....	64
3.3 Data Flow in System:.....	66
3.4 Approach to System Design:.....	66
3.5 Navigation Logic.....	67
3.6 Interoperational Mechanism of both navigation files (path.h and path.cpp):.....	70
CHAPTER 4: AGV Development.....	71
4.1 Arduino Connections.....	71
4.2 ESP32 Connections.....	74
4.2.1 LCD connections (connected through I2C).....	74
4.2.2 PN532 NFC Reader Connections.....	74
4.2 Body Fabrication.....	74
4.2.1 Material Selection.....	74
4.2.2 CAD modelling.....	75
Implementation and Conclusion.....	76
CHAPTER 1: Operation and Implementation.....	77
1.1 Precautions taken.....	77
1.2 Implementation Instructions.....	77
AGV Training Manual: Implementation and Troubleshooting.....	77
CHAPTER 2: Conclusion.....	81
2.1 Limitations and Scalability.....	81
2.1.1 Limitations.....	81

2.1.2 Scalability.....	81
2.2 Reliability.....	81
Justification for reliability.....	81
2.3 Feasibility.....	82
CHAPTER 3: Future Scope.....	84
3.1 Modular wheel design.....	84
3.2 Battery.....	86
3.3 Virtual map feed for defining route.....	88
3.4 Dijkstra's Algorithm.....	90
3.4.1 Error Handling for Dijkstra's Algorithm and AGV Navigation.....	92
References.....	94

PART-I

Introduction and Research Establishment

CHAPTER 1: Background

Automation has become the core of modern manufacturing so much so that no company is able to survive in a competitive market without automating its operations. In fact the term automation basically refers to the use of computers and other automated machinery for the execution of tasks that human labour would otherwise perform. Automation is used to manage systems and to control processes, thus leading to reducing the necessity of human intervention.

Nowadays, manufacturers seek to implement methods of automation appropriate to their needs and purposes. Companies automate their activities for a variety of reasons. Increasing productivity is normally the main aim for companies desiring competitive advantages. Automation reduces human errors and improves the quality of output. Other reasons for automation include the presence of hazardous working environments and the high cost of human labour in such areas. The decision regarding automation is often associated with some economic and social considerations: Ravazzi and Villa [1], Chadwick and Jones [2].

The apparel industry is currently undergoing an automation push driven by rising labor costs, fast fashion demands, and growing needs for customization. While some automation exists (e.g., computer-aided design), there's a clear need for more to address challenges like shrinking margins, quick turnaround times, and rising labour costs. Increased automation can improve efficiency, ensure consistent quality, and enable faster production cycles, making the industry more competitive in the dynamic fashion landscape.

One of the key components of automation in a manufacturing process is the MHS. This system is responsible for loading, unloading, moving or generally transporting any type of materials (raw material, work in process, and finished goods) within and out of manufacturing cells such as warehouses, machines and assembly lines. MHS consists of different components (i.e. Conveyors, AGVs, Robots, Automated Storage and Retrieval system (AS/RS)). Utilisation of components, either individually or from a combination point of view, is determined by its application or pre-assigned flexibility.

In this study, we focus on AGV. **An AGV is a driverless transportation system used for horizontal movement of materials. It is an unmanned vehicle, controlled and driven by a host computer, to carry out the required material movement in a manufacturing floor.** AGVs can be used in both interior and exterior environments such as manufacturing, distribution, transshipment and (external) transportation areas. In manufacturing areas, AGVs are used to transport all types of materials related to the manufacturing process.

CHAPTER 2: Gap Analysis and Objectives

2.1 Industry Scenario

The apparel industry is undergoing a significant transformation, driven by several factors that necessitate the adoption of automation solutions.

1. Rising Labour Costs and Shrinking Margins:

- Globalisation has led to intense competition in the apparel industry, with manufacturers facing pressure to reduce production costs.
- Labour costs, particularly in developed economies, have steadily increased, squeezing profit margins [3].

2. Fast Fashion and Shorter Product Life Cycles:

- The rise of fast fashion trends necessitates quicker turnaround times and increased production flexibility.
- Traditional, labour-intensive manufacturing processes struggle to keep pace with the rapid changes in fashion demands [4].

3. Growing Demand for Customization and Mass Customization:

- Consumers are increasingly seeking personalised and customised clothing options.
- Automation solutions can enable efficient production of smaller batches and customised products, catering to this growing demand [5].

4. Labour Shortages and Skill Gaps:

- The apparel industry faces challenges in attracting and retaining skilled labour, particularly in developed countries.
- Automation can address these shortages by taking over repetitive tasks and freeing up workers for higher-value activities [6].

5. Quality Control and Consistency:

- Manual processes can be prone to human error, leading to inconsistencies in product quality.
- Automation solutions offer higher precision and repeatability, ensuring consistent product quality [7].

The apparel industry is at a crossroads, facing the need to balance cost-effectiveness with agility and customization. Automation solutions emerge as a critical tool to address these challenges, improve efficiency, and cater to the evolving needs of the market. By embracing

automation, apparel manufacturers can ensure their competitiveness and navigate the changing industry landscape successfully.

One of the key components of automation in any manufacturing process is the Material Handling System because of its repetitive nature and the fact that it is a non-value adding but necessary task [18]. This system is responsible for loading, unloading, moving or generally transporting any type of materials (raw material, work in process, and finished goods) within and out of manufacturing cells such as warehouses, machines and assembly lines. MHS consists of different components (i.e. Conveyors, AGVs, Robots, Automated Storage and Retrieval system (AS/RS)). [19]

2.1.1 Project Focus

In this study, we focus on AGV: Automated Guided Vehicles. **An AGV is a driverless transportation system used for horizontal movement of materials. It is an unmanned vehicle, controlled and driven by a host computer, to carry out the required material movement in a manufacturing floor.** AGVs can be used in both interior and exterior environments such as manufacturing, distribution, transshipment and (external) transportation areas. In manufacturing areas, AGVs are used to transport all types of materials related to the manufacturing process.[10] The reason why AGV was chosen among all the other transport systems is justified in Chapter 3.

AGVs are broadly implemented in warehouses where the major task is transportation of goods. Relying on this fact, we have developed a solution for sewing floor material transport, which is still done manually by unskilled labour.

Furthermore, for developing this AGV, we have chosen the initial material movement of the sewing floor, i.e., bringing cut parts from the rack/storage area to the sewing line input tables. We have not entered the intra-line movement for ease of development. This project is one of a kind since there are no examples of AGVs being deployed on the sewing floor to automate the material transportation process.

2.2 Current Factory Floor Scenario

To get a realistic idea of material transport on the sewing floor, we talked to a Senior IE Manager in XYZ Exports. Following are the observations noted:

- The cutting room is on the basement floor where packages of cut pieces are made and sent to the first floor through lifts where the sewing floor is located. Each package consists of around 100-150 pieces that consist of different bundles of panels in a particular garment.
- 2 helpers are involved in this process of getting packages from the cutting floor to the lift and eventually to the sewing line input tables. There are 16 sewing lines on the sewing floor and the distances of these lines from the lift are approximately 80 feet, 100 feet and 120 feet.

- The capacity of the sewing lines is 60 garment pieces per hour and thus the loading happens almost 4 times a day per sewing line. In total, the loading is happening 64 times a day and the 2 loaders travel this distance throughout the day with that load.
- These loaders move crates which have a maximum capacity of 150 pieces and weigh around 40 kgs in trolleys equipped with 4 wheels and a handle, weighing around 20 kgs. So the total load moved by them in each trip is 60 kgs. The approximate time of a loading is around 3 minutes.
- They deliver these packages to sewing line helpers, 2 on each line (32 helpers) in total; who unwrap the package, spread all the panels on the input table and put 1 component each in a basket for the sewing line operation.
- All of these helpers fall under the category of unskilled labour for the company and the CTC for each of them is Rs.15,000 per month for a shift time of 10 hours per day.

Major Challenges:

- The lift that the loaders use is a passenger lift, they do not have a material lift for the vertical transport. Sometimes, the lift is busy and thus the number of cycles for the lift increases along with the time in loading. Lift cycle time is the major problem in this setup.
- Since cutting packages cannot be carried by stairs due to their weight, any breakdown or lift occupancy for a longer time poses a big challenge as it delays loading, however these occurrences are rare.
- If the lift is non-usable at the moment and loading is necessary, 4 helpers from the line accompany the 2 loaders to help get cutting packages from the basement through stairs.

According to the manager, there is a major need for better solutions for MHS that can optimise the present arrangement.

Currently the unit has automation solutions for sewing operations like pocket creasing, pocket attachment and real-time data monitoring. However, there's no automation in MHS.

Considerations for Implementation of new solutions:

- Because of the need for competitive pricing and cutting margins everywhere to offer the best possible price, the scope of earning is very less. Business is very challenging and it is difficult to grab orders since all competitors have an equivalent chance with the knowledge of standard and improved procedures.
- Thus any kind of saving in either time or cost is welcome.
- Automation is useful when the ROI is within 1.5 to 2 years and the cost saved is between 5 to 10 lakhs in a year. The implementation and maintenance cost should also be within feasible limits.

2.3 Objective and Proposed Solution

Objective: To automate material transport on sewing floor in apparel industry by developing an AGV suited for the RMG (Ready-made garment manufacturing) environment

In other words, we strive to optimise production floor operations and reduce manual labour costs via development of an Automated Cut Parts Transport System- a robot based system which will autonomously transport cut parts from fabric storage to sewing line loading stations.

2.3.1 Sub-Objectives:

- To identify the automation requirements of fabric feeding system into sewing lines
- To develop an AGV according to the specifications that cater to this requirement
 - Fixing functionalities
 - Studying different navigation systems and choosing the most feasible method
 - Developing a feasible core logic
 - Parts selection
 - Designing the hardware and software for the AGV
- To develop guidelines for implementation of the AGV on the factory floor
- To evaluate the feasibility of this solution in RMG environment

2.3.2 Research Methodology

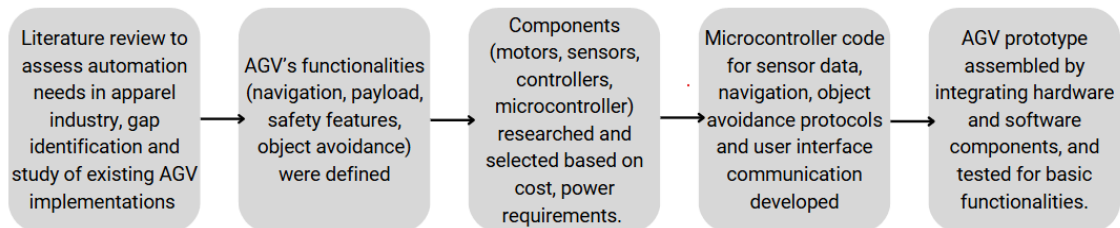


Fig 1: Methodology

CHAPTER 3: Literature Review

3.1 Current Scenario of Material Handling Systems in Apparel Industry

3.1.1 Material Handling Systems in Apparel Manufacturing:

- **Warehousing:** Automated Storage and Retrieval Systems (AS/RS), conveyor belts, and forklifts are commonly used for efficient storage, retrieval, and movement of raw materials like fabric rolls and trims [20].
- **Cutting Department:** Spreading machines automate fabric spreading for accurate cutting. Cutters (manual or automated) and conveyor belts facilitate material movement during the cutting process [21].
- **Sewing Floor:** While less common, some studies suggest the potential benefits of deploying Automated Guided Vehicles (AGVs) on sewing floors. AGVs can autonomously transport cut parts between workstations, reducing manual labor and improving efficiency [22].

3.1.2 Case Studies:

- **Fast Retailing (Case Study):** This case study describes how the fashion retailer implemented conveyor belts and overhead monorail systems to improve material handling efficiency within their garment production facilities [23].

3.1.3 Limitations and Research Gaps:

- Limited research specifically addresses material handling systems deployed directly on sewing floors.
- Studies on the effectiveness and return on investment (ROI) of AGVs in apparel sewing environments are scarce.

3.1.4 Further Research Directions:

- Investigation of the implementation and impact of AGVs for cut parts transportation on sewing floor efficiency and worker productivity.
- Comparative analysis between traditional manual material handling and AGV-based systems in apparel production.
- Exploration of the integration of AGVs with existing sewing floor infrastructure and production management software.

3.2 AGV in Warehousing Setups

Automated Guided Vehicles (AGVs) are increasingly playing a vital role in warehouse operations by automating various material handling tasks. Following are the ways in which AGVs are commonly deployed in a warehouse setup.

1. Transportation of Goods:

- AGVs efficiently move pallets, totes, and other materials within the warehouse, freeing up human workers for more complex tasks. This reduces reliance on manual forklifts and improves overall throughput. [25]

2. Order Fulfillment:

- AGVs can be integrated into picking and packing processes, delivering items to pickers or transporting packed orders to dispatch areas. This streamlines order fulfillment and reduces order fulfillment times. [26]

3. Repetitive Tasks:

- AGVs excel at handling repetitive tasks like moving empty pallets or transporting waste materials. This automates mundane tasks, allowing workers to focus on value-added activities [27].

4. Increased Efficiency and Productivity:

- AGVs operate 24/7, unlike human workers who require breaks. They also navigate precisely and efficiently, minimizing travel time and maximizing productivity within the warehouse [28].

5. Improved Safety:

- By automating material handling, AGVs reduce the risk of workplace accidents associated with manual lifting and forklift operation [29].

Additional Considerations:

- The specific application of AGVs in a warehouse depends on factors like warehouse layout, product types, and order fulfillment processes.
- Different types of AGVs exist, such as line-guided, vision-guided, and laser-guided, each with its own advantages for specific tasks.

3.3 Types of Material Handling Systems currently in the Apparel Industry

3.3.1 Introduction:

Efficient material handling is crucial for optimizing production processes in the apparel industry. This review explores the various material handling systems used within different departments and highlights real-world examples and research findings.

3.3.2 Types of Material Handling Systems:

A) Warehousing:

- **Automated Storage and Retrieval Systems (AS/RS):** These computer-controlled systems store and retrieve materials vertically, maximizing space utilization [30].

B) Cutting Department:

- **Spreading Machines:** Automate fabric spreading for accurate cutting.
- **Cutters (Manual or Automated):** Cut fabric plies based on patterns.
- **Conveyor Belts:** Facilitate material movement during the cutting process [31].

C) Sewing Floor:

- **Carts and Trolleys:** Manual transportation of cut parts between workstations.
- **Overhead Monorail Systems:** Garments hang on hangers suspended from a track, facilitating movement [32].

3.3.3 Emerging Technologies:

- **Automated Guided Vehicles (AGVs):** Driverless vehicles navigate pre-programmed paths or utilize sensors for autonomous movement, transporting materials within factories [33].

3.3.4 Future Trends:

- Integration of material handling systems with production management software for real-time tracking and optimization.
- Exploration of collaborative robots (cobots) for tasks like material handling and assembly in sewing floors.

3.3.5 Limitations and Research Gaps:

- Limited research specifically addresses material handling systems deployed directly on sewing floors.
- Studies on the effectiveness and return on investment (ROI) of AGVs in apparel sewing environments are scarce.

3.4 Competitive edge of AGVs through Automation (Case Studies)

3.4.1 A study on the Sri Lankan apparel industry (A De Silva, S Gunathilake, H Munaweera, D Perera and L Gunathilake)

1. **Process automation:** Use of model-based approach for automatic inspection of fabric, use of laser knife or water jet in the cutting department, and use of pressing robots in the pressing process of apparels have been implemented in the apparel industry to gain a competitive advantage in the apparel industry. Through these technological advancements, vertical integration and horizontal integration could be achieved in both the apparel manufacturing sector and all parties in the value chain [34].
2. **Intelligent Manufacturing:** Through automating the apparel industry by implementing various techniques such as Robotic Process Automation, the business processes could be made more efficient as the employees will have more time to work on product innovation and its improvement. This will eventually lead the apparel production companies to distribute the apparel products faster than the competitors and achieve a competitive advantage in the market [35]. The robotic technology introduced in the apparel sector in the sewing department, pressing department and fabric handling department as stated, will prove that intelligent manufacturing will have a positive impact towards achieving competitive advantage in apparel companies [36].
3. **Customised production:** Customised production of apparels will refer to manufacturing apparels in standard designs where their sizes will be produced to fit the individuals and designs will comprise of different pockets, cuff shapes, fabrics and colours that are personally embellished to suit different consumer tastes [37]. The competitiveness of companies could be seen in the ability to react quickly to the rapid changes in customer requirements and to cope up with these changes various flexible automation tools such as CAD and CAM are used to design apparels [38].
4. **Relationship between Process Automation and Competitive Advantage:** According to Nayak & Padhye (2018), automating the processes of the garment industry will help to achieve the competitive advantage in the apparel industry while producing apparels of high quality at lower costs. Thus, it is clear that process automation will ultimately lead to achieving competitive advantage in the industry because challenges which arise through competition could be faced successfully with various technological advancements. Cost advantage will be mediating the relationship between process automation and competitive advantage because cost advantage could be achieved through process automation as low costs could be achieved in the apparel industry through automating all processes [39]. Thus, apparel companies could gain a

competitive advantage by being superior to their competitors due to the low cost that will be achieved through process automation.

3.4.2 Automation in Garment Manufacturing (Rajkishore Nayak and Rajiv Padhye)

There are several benefits of using automatic tools and equipment in garment manufacturing, which are mentioned below:

- **Increase in productivity:** Automation increases productivity by increasing the efficiency of the process. When the job is performed by a labour, there are chances of error, reduced efficiency due to fatigue, and the breaks taken by the worker. However, the automated process of performing the job eliminates these and increases productivity.
- **Increased inventory turnover:** As the productivity of the industry increases, the material turnover also increases. With manual operations the raw materials, cut components, and semi finished components have to wait longer to get converted into the final garment. Hence, with automation increased inventory turnover is achieved.
- **Improvement in quality:** As mentioned above, automation leads to reduced error of garments because human intervention is eliminated. This leads to the products with less defects, improved quality, and reduced rejection rates.
- **Replacement of repetitive and monotonous work:** Majority of the garment manufacturers use progressive bundle system (PBS) of production. In PBS one worker performs a specific job and passes to the other. Hence, the work becomes repetitive and monotonous for the worker. This can lead to fatigue and reduced efficiency. However, automation can help to avoid these as all these repetitive works are performed by the machine.
- **Reduction of variability among products and product batches:** As the involvement of labour is reduced, the variability of the products produced by different workers is also reduced. Similarly, the variability of the same product manufactured in different batches (manufactured over different times or in different industries) is also reduced.
- **Performing jobs beyond human capability:** Automation can perform some jobs, which needs high skills of the labour. As today's garments are moving toward the integration of electronic devices and other gadgets, high skill is needed many times to perform these operations. Automation can achieve these objectives much easily.

3.4.3 Automated transportation systems subject to interruptions in production and intralogistics – a survey and evaluation (Ireneus Wior, Stefan Jerenz and Alexander Fay)

This research paper supports the need for automation AGVs in the apparel industry by showcasing the benefits of AGV technology in various industries where it has been successfully implemented. Case studies referenced in the document, such as Liu et al. (2004) and Zhen et al. (2012), demonstrate the positive impact of AGVs in improving performance, reducing labor costs, and increasing capacity in container terminals. These successes are extrapolated to the apparel industry to show the potential benefits of AGV implementation.

3.5 The Need and Benefits of AGVs in the Apparel Industry

3.5.1 Need for Automation in the Apparel Industry:

The apparel industry faces intense competition with rising labor costs, shrinking profit margins, and the demand for faster production cycles [1]. Manual material handling processes are inefficient and prone to errors, hindering productivity and increasing costs [2]. Automation using technologies like AGVs can address these challenges. [40] [41].

3.5.2 Benefits of AGV Technology in Various Industries:

Several industries have successfully implemented AGVs, demonstrating their benefits:

- **Case Study 1: Automotive Industry:** BMW uses AGVs for material handling tasks within their assembly lines, achieving a 25% reduction in production lead time and improved worker safety [42].

Relevance to Apparel Industry:

The apparel industry also faces tight production deadlines and prioritizes worker safety. Similar to BMW's experience, AGVs in sewing floors can:

- **Reduce material handling time** by automatically transporting fabrics, trims, and garments between workstations, leading to faster production turnaround.
- **Minimize manual lifting** by workers, potentially reducing the risk of injuries associated with repetitive lifting and carrying of materials.

These case studies illustrate the potential of AGVs to improve efficiency, reduce costs, and enhance safety across various industries.

- **Case Study 2: Warehouse and Distribution Centers:** AGVs streamline material movement in warehouses, leading to increased storage capacity, faster order fulfillment, and reduced labour costs [43].

- **Benefits Achieved:**
 - **Reduced Lead Time:** BMW reports a 25% reduction in production lead time by streamlining material movement using AGVs. This translates to faster production cycles and potentially quicker product delivery to customers.
 - **Enhanced Worker Safety:** AGVs automated material handling tasks, potentially reducing manual lifting and carrying by workers, leading to a safer work environment.

3.5.3 AGV Applications for Optimization and Cost Reduction in Apparel Sewing

Floors:

The sewing floor presents several opportunities for AGV implementation:

- **Material Movement:** AGVs can transport fabrics, trims, and finished garments between cutting rooms, sewing stations, and storage areas, reducing manual handling and improving worker productivity [44].

In our current context, we aim to minimize the human input to the transportation of cut materials possibly from fixed racks to individual sewing lines.

- **Kitting and Order Picking:** AGVs can be integrated with picking systems to deliver pre-assembled kits of materials to sewing stations, minimizing setup times and production delays [45].
- **Waste Removal:** AGVs can efficiently collect and transport fabric scraps and other waste materials, minimizing clutter and improving workplace safety and cleanliness [46] [47].

3.6 Comparison of Material Handling Systems: Transport Mechanisms vs. AGVs

3.6.1 Traditional Transport Mechanisms:

- **Forklifts:** Versatile for transporting heavy loads but require skilled operators and pose safety risks.
- **Conveyor Belts:** Efficient for linear movement of materials but lack flexibility and can be limited in handling capacities.
- **Overhead Cranes:** Ideal for high-bay warehouses but require significant infrastructure investment and have limited horizontal movement capabilities.

3.3.2 Limitations of Traditional Systems:

- **Labor Intensive:** Often require manual operation, increasing labor costs and potential for human error.
- **Limited Flexibility:** Fixed layouts for conveyors and cranes restrict adaptability to changing production needs.
- **Safety Concerns:** Manual operation of forklifts and cranes poses safety risks for workers.

3.3.3 Case Studies and Examples:

- **Bosch Rexroth (Case Study):** The industrial automation leader implemented AGVs for transporting parts between machining centers, resulting in a 20% reduction in production lead times [48].
- **BMW Manufacturing Plant:** BMW utilizes AGVs within their production lines to transport car parts autonomously, improving overall production efficiency [49].
- **Apparel Industry Example:** While less common, AGVs have the potential to revolutionize material handling in sewing floors. They can autonomously transport cut parts between workstations, reducing manual labor and improving worker productivity [3].

3.3.4 Research Support:

- A study by Deloitte suggests that AGVs can lead to a 20-30% increase in warehouse productivity [50].
- Research by International Journal of Production Research found that AGVs significantly reduce material handling times and improve production line efficiency [51].

3.4 Conclusion

AGVs offer a compelling alternative to traditional transport mechanisms. Their flexibility, safety advantages, and potential for efficiency gains make them a valuable investment for various industries. While traditional systems have their place, AGVs present a future-proof solution for optimizing material handling needs [52].

3.5 History

Automatic Guided Vehicles (AGVs) have played a vital role in moving material and product for more than 50 years. The first AGV system was built and introduced in 1953. It was a modified towing tractor that was used to pull a trailer and follow an overhead wire in a grocery warehouse. By the late 1950's and early 1960's, towing AGVs were in operation in many types of factories and warehouses.

The first big development for the AGV industry was the introduction of a unit load vehicle in the mid 1970s. This unit load AGVs gained widespread acceptance in the material handling marketplace because of their ability to serve several functions; a work platform, a transportation device and a link in the control and information system for the factory.

Since then, AGVs have evolved into complex material handling transport vehicles ranging from mail handling AGVs to highly automated automatic trailer loading AGVs using laser and natural target navigation technologies.

In fact the improvement of AGVs over the last decade is deeply indebted to development of Scheduling, Algorithm and Steering methods. The problem of scheduling of AGVs and the other supporting equipment has been extensively studied by Basnet and Mize [8] and Rachamadugu and Stecke [9] currently providing the most up-to-date and comprehensive reviews in this area.

3.6 Designing process

The procedure of designing an AGV is a complicated process. Some issues which directly impact the design of the proposed AGV are listed and widely explained. These issues are not only hardware but also software issues. Software is not just constants in inputs but it is variable and outputs must be chosen to specify the design. Furthermore, these issues interact with each other so that each cannot be considered separately but all must be considered simultaneously. [10]

- Vehicle Hardware Design
 - Movement modelling
 - System Configuration
 - Kinematic Computation
 - Components of AGV
- Path and Guide-path Design
 - Interaction of path and sensors
 - Path's specifications
- Workstation Information
- AGV Scheduling
 - Idle-AGV positioning

- AGV moving
 - Stopping (deadlock resolution)
- Software design
 - Algorithm
 - Programming
 - Interface commander
- Battery management

3.7 Movement Modeling

One of the most challenging parts of designing the AGV is the movement modelling. Movement modelling highly depends on the size of the area, expected maneuvering ability, position of stations and allocated path between them. Furthermore, it becomes much more vital if the area is small with restricted moving space so that the vehicle should be designed to move and make U-turns, sharp turns, curve turns and of course handling deviations.

Typically, standard AGVs are distinguished by having an axis in one rotational degree of freedom at the back and the front axis in two rotational degrees of freedom to guide the vehicle on the arbitrary path. The most important advantages of such a model are the accuracy and simplicity which make the plan highly applicable. However, there is a big disadvantage which is the weakness in maneuvering on the sharp turns. In fact in such models the minimum radius of the turn curve is restricted.

Despite the advantage of implementing standard models, another model has to be used. It is because of the specifications of the place for which the AGV is proposed. Due to the need of an AGV with the ability of any type of movement and turns, the military movement model is chosen. In this model there are 2 motors installed on each side, connected to the main controller kit. This kit receives the commands and respectively controls and drives each motor independently. It means that the vehicle is able to perform a U-turn even on a point, which is the hardest and sharpest movement. Benefiting from this model, the vehicle is capable of performing all types of movements and turns. The following part will illustrate it with figures of all different capabilities of the vehicle. [10]

3.7.1 System Configuration [10]

The configuration of the vehicle forces is shown in figure 1. It is seen that the two sprockets in the middle of the AGV are driving wheels, which are actuated separately by two DC motors. So there are two trajectories of line and arc for this kind of AGV. The centre of linking line between two driving wheels is the kinematic origin of AGV.

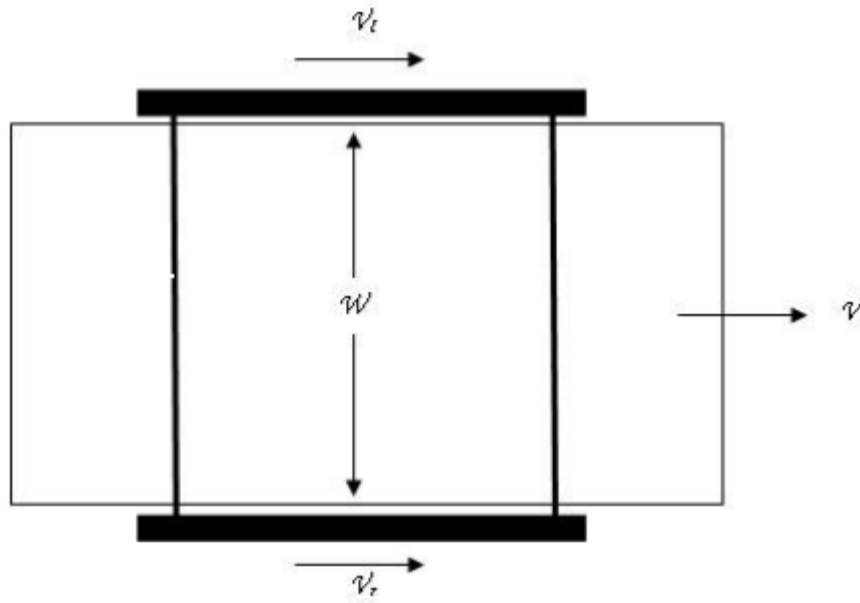


Figure 1: Vehicle Forces

V = linear AGV velocity (mm/s)

ω = angular AGV velocity (rad/s)

W = distance between two sprockets = 400mm

V_r = velocity of right wheel (mm/s)

V_l = velocity of left wheel (mm/s)

3.7.2 Kinematic Computation [10]

In order to be able to configure the kinematic computation of the AGV, it is assumed that all of the belt's wheel force is applied to a point which is at the centre of each wheel on each side.

Therefore for the linear movement, velocity is calculated as follows:

$$V = \frac{V_r + V_l}{2} \text{ (mm/s)} \quad (2.1)$$

And for the angular movement, velocity is calculated as:

$$\omega = \frac{V_r - V_l}{W} \text{ (rad/s)} \quad (2.2)$$

Figure 2 shows how opposite rotation in motors with the same speed enables the vehicle to rotate around its centre.

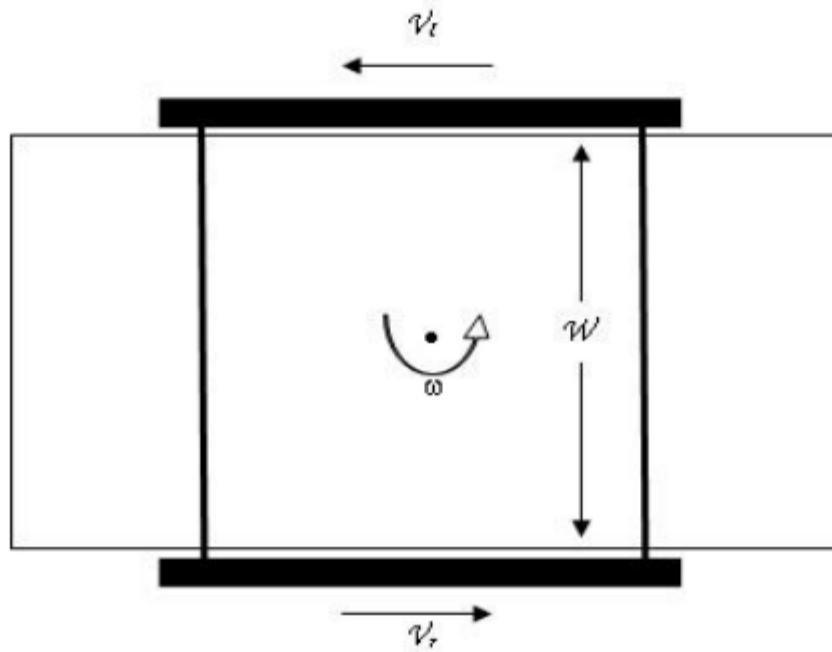


Figure 2: Rotation in Both Sides, Different Orientation, Same Speed
 Varying the velocities (both speed and direction) of the wheels, all kinds of movements can be performed.

3.8 Body structure design

Table 1. Design parameter table of AGV.

Projects	Parameter	Remarks
Overall dimensions (mm)	2500×1800×600	
Steering mode	Omnidirectional steering	Can realize in situ steering
Number of steering wheels	2	Horizontal steering wheel
Steering wheel power (Kw)	4	
Number of angle wheels	2	
Load mass (kg)	3000	
Readiness quality (kg)	1500	Including battery
Full load maximum speed (m/s)	0.5	
Control mode	Manual remote control, automatic	
Protection grades	IP54	
Suspension mode	Drum damper	
Battery	Lithium iron phosphate	
Voltage (v)	48	
Battery capacity (A.h)	270	
Duration (h)	8	

The chassis of the car body is shown in Figure 4.

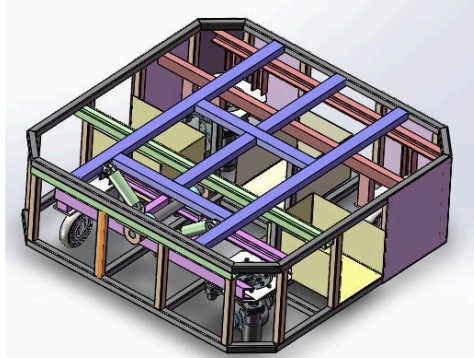


Figure 4. AGV body structure

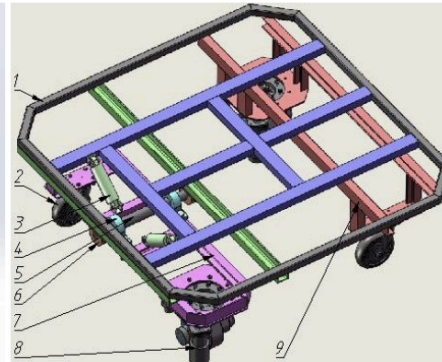


Figure 5. Exploded car body

(1. Load platform, 2. Driven wheel, 3. Shock absorber, 4. Bearing group 1, 5. Shaft, 6. Bearing group 2, 7. Front wheel connection group, 8. Steering wheel, 9. Rear wheel connection group)

The car body is the basic part of AGV, the installation foundation of other assembly components, and also one of the main components in motion. The main conditions considered in the design are:

(1) The strength and rigidity of the car body must be designed on the premise of meeting the requirements of the car when carrying and accelerating.

(2) The weight of the car body has to be reduced as much as possible so as to increase the effective bearing weight; The premise must be under the condition that the car body has enough rigidity.

(3) The centre of gravity of the car body has to be lowered so as to improve the antiroll-over capability of the whole car and ensure better and stable operation of the car. [11]

3.9 Reliability Study

Reliability is characteristic of equipment that allows it to consistently perform its intended function over a specific period. It measures the ability of equipment to operate without unexpected failures or disruptions. Reliability is of utmost significance in industrial operations as it directly impacts operational goals, customer satisfaction, and overall productivity. Reliable equipment ensures smooth production processes, minimises downtime, and instils confidence in customers who rely on consistent and uninterrupted service.

3.9.1 Factors Affecting Reliability

Design and Manufacturing:

Proper design practices, such as incorporating robust components, redundancy, and fault-tolerant features, enhance reliability by minimising the likelihood of failures. Moreover, a thorough understanding of the operating environment, material selection, and adherence to industry standards during manufacturing contribute to the overall reliability of the equipment.

Maintenance Practices:

By implementing proactive maintenance strategies, we can identify potential issues, address them promptly, and prevent failures that can compromise reliability. Regular inspections, lubrication, calibration, and timely repairs are all maintenance activities that contribute to enhancing equipment reliability.

3.9.2 Reliability Metrics

To quantify and measure reliability, certain metrics and indicators are used in industrial settings. Two commonly used metrics are:

Mean Time Between Failures (MTBF):

MTBF measures the average time between consecutive failures of a specific piece of equipment. It provides insights into the reliability of the equipment by indicating the expected interval between failures. A higher MTBF value implies greater reliability, as it suggests longer periods of uninterrupted operation.

Mean Time to Repair (MTTR):

MTTR represents the average time required to repair equipment after a failure occurs. It measures the efficiency of the maintenance process in terms of restoring functionality. A shorter MTTR indicates quicker response and repair times, minimizing downtime and maximizing equipment reliability.

In addition to MTBF and MTTR, other metrics like availability, reliability, and maintainability are also employed to assess and evaluate the performance of equipment. [12]

So, what can be done to ensure high availability of the machinery (reliability) and keep our manufacturing costs under control?

The best solution is a hybrid approach that incorporates elements of all the strategies above. This approach should also include the role of an experienced maintenance manager. It is incumbent upon them to perform an in-depth analysis of the current state of the machinery.

The analysis should include:

- the number of identical pieces of equipment
- criticality for the customer, i.e., for production (can the process be transferred to other lines in the event of a breakdown?)
- current technical condition of the equipment, its age, signed service contracts, level of know-how of local maintenance services, etc.
- availability of spare parts
- historical data on downtimes and failures [13]

3.10 Reliability Calculation

Calculating MTBF (Mean Time Between Failures):**1. Reliability Testing:**

- Conduction of rigorous reliability tests on the AGV prototype.
- Simulation of various operating conditions (load, speed, environment) our AGV might encounter in real-world use.
- During testing, recording the number of failures and the operating hours between each failure.
- MTBF can be estimated by averaging the operating hours between failures:

MTBF = Total Operating Hours / Number of Failures [14]

2. Parts Derating and Reliability Prediction:

- Analysis of the AGV's design and identification of the MTBF of each individual component (motors, batteries, sensors, etc.) based on manufacturer specifications.
- Deration of the component MTBFs by a certain percentage (typically 10-30%) to account for real-world stresses and potential variations.
- Usage of reliability prediction software or mathematical models to calculate the overall system MTBF considering component reliabilities and their interactions. [15]

Calculating MTTR (Mean Time To Repair):

- Analysis of the design and identification of the time it would take to replace or repair each critical component that might fail.
- Factors to be considered:
 - Accessibility of components for repair
 - Complexity of repair procedures
 - Availability of spare parts and tools
 - Required skill level of maintenance personnel
- Estimation the MTTR by averaging the repair times for critical components:

MTTR = Sum of Repair Times for Critical Components / Number of Critical Components [16]

3.10.1 Designing a Maintenance Schedule

An effective maintenance schedule considers both preventive and corrective maintenance:

- **Preventive maintenance:**
 - Based on component MTBFs and recommended maintenance intervals from component manufacturers, we schedule preventive maintenance tasks like:
 - Lubrication
 - Inspections (visual, thermal, etc.)
 - Filter replacements
 - Minor adjustments
 - The goal is to identify and address potential issues before they escalate into failures.
- **Corrective maintenance:**
 - This involves repairs after failures occur.
 - Use the MTTR data to estimate downtime associated with repairs.

- Continuously monitor maintenance data to identify recurring failures and adjust the preventive maintenance schedule accordingly. [17]

Additional Considerations:

- **Environmental factors:** If the AGV operates in harsh environments (dust, extreme temperatures), we need to consider more frequent preventive maintenance.
- **Operating conditions:** Heavy workloads or high speeds might warrant more frequent inspections or component replacements.
- **Data-driven approach:** As real-world operating data is collected from deployed AGVs, it should be used to refine the MTBF, MTTR estimates, and maintenance schedule for future models.

PART-II

Methodology: Design and Development of AGV

CHAPTER 1: Ground Work

1.1 Functionalities

1.1.1 Main goals and tasks that the AGV is expected to accomplish

1. Autonomous movement around the floor (without the need of human intervention) through RFID tags and PN532 NFC Reader
2. Carrying a payload of 20 kg from Point A to Point B
3. Can be called to different stations as per style pieces requirement
4. Can be called to home base (charging station)
5. Responsive to user interface in real time
6. Will work on a fixed path
7. Object avoidance protocol through ultrasonic sensor moved by SG90 servo motor
8. Presence of a kill-switch in case of emergency shut-downs

1.1.2 Autonomous Navigation (RFID & PN532 Reader):

- The AGV utilizes RFID tags embedded in the floor and a PN532 NFC reader onboard to navigate autonomously.
- As the AGV moves, the reader continuously detects nearby tags, enabling it to identify its current location and plan its path to the designated destination.
- The AGV resumes motion only in the case of detection of a black line, crossing out the chance of possible path deviations.
- This eliminates the need for human intervention in guiding the AGV, streamlining material transport workflows.

1.1.3 Payload Handling (20 kg Capacity):

- The AGV is equipped to carry a payload of up to 20 kg, making it suitable for transporting various materials within your facility.
- This capacity allows for efficient movement of a wide range of items, reducing manual handling tasks and improving overall productivity.

1.1.4 Station Calling (Style Piece Requirement):

- The AGV can be summoned to different stations based on real-time production needs.
- This functionality likely integrates with a production management system or a user interface to receive instructions on the required style pieces and their locations.
- By automatically delivering materials to designated workstations, the AGV minimizes production downtime and ensures a smooth flow of materials.

1.1.5 Home Base Docking (Charging Station):

- The AGV is programmed to return to its home base (charging station) when its battery level reaches a predefined threshold.
- This ensures continuous operation by automatically recharging the battery without manual intervention.

1.1.6 Real-time User Interface (Responsive HTML):

- The AGV can be monitored and controlled in real-time through a user interface developed using HTML.
- This interface, likely accessible through a computer or tablet, might provide functionalities like:
 - Viewing the AGV's current location and status (battery level, payload status).
 - Manually sending the AGV to specific stations.
 - Monitoring past travel history and performance data.

1.1.7 Fixed Path Following (Line Sensor):

- The AGV utilizes a line sensor to maintain its position and follow a predefined path on the floor.
- This sensor detects a specific line pattern (e.g., black line on white background) and adjusts the AGV's movement accordingly.
- The combination of line sensors and RFID tags provides a robust navigation system, ensuring the AGV stays on track even in areas with potential interference.

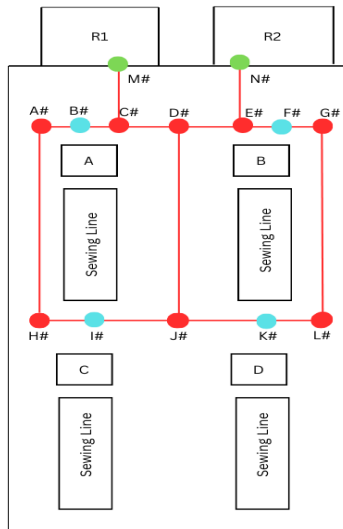
1.1.8 Object Avoidance (Ultrasonic Sensor & Servo Motor):

- The AGV employs an ultrasonic sensor mounted on a servo motor for obstacle detection and avoidance.
- The sensor emits and receives ultrasonic waves to detect objects in its path.
- The servo motor allows the sensor to rotate, providing a wider range of obstacle detection.
- Upon detecting an obstacle, the AGV's control system triggers appropriate maneuvers (e.g., stopping, changing direction) to avoid collisions, ensuring safe operation within your facility.
 - adaptability to future technological advancements or specific factory requirements.
 - **Power Management:** Efficient power management is vital for the robot's continuous operation. The home base should be equipped with a charging station, and

the system should have the capability to monitor and manage power levels effectively.

1.1.9 Work environment analysis:

Description of the layout and flooring of the facility.



- Typical apparel production floor setup with racks situated in front of the sewing lines to receive cut material from the cutting department and then distribute material to respective sewing lines according to the style specifications.

- Operator lines adjacent to each other with walking space of 10 inches width will be present, along with auxiliary machines present in the sides

Potential obstacles or constraints in the facility that can affect the AGV operation:

- Prevalent worker paths, possible waste disposal obstacles (like baskets)
 - Inability of the AGV to traverse facility vertically as it is designed to only operate on the floor it is deployed on, and rough terrain (if any) would be a problem
- Discarded cloth pieces pose a threat to get lodged in the tyres or the motor drive system
 - Crowded spaces

1.2 Navigation Methods Study

1.2.1 Navigation methods suitable: wire guidance, laser navigation, or vision-based navigation

1. First approach ideation: LiDAR-based Location Mapping with Gazebo and ROS:

Hardware required :

- Raspberry Pi 4
- Arduino Mega
- LiDAR Scanner (e.g., rplidar A3)
- Motor drivers and actuators for AGV movement
- ROS-compatible motor controllers (optional)

Software required :

- Raspbian OS for Raspberry Pi
- ROS (Robot Operating System)
- Gazebo Simulator

- SLAM algorithms (e.g., cartographer, gmapping)
- Navigation stack (e.g., move_base)

Steps:

1. Install ROS and Gazebo on Raspberry Pi: Follow official ROS installation guides for your platform.
2. Connect LiDAR to Raspberry Pi and configure ROS driver: Refer to the LiDAR manufacturer's instructions and ROS driver documentation.
3. Build a Gazebo model of your manufacturing floor: Use CAD software or existing blueprints to create a virtual environment resembling your factory layout.
4. Integrate the AGV model in Gazebo: Design a 3D model for your robot and add sensor and actuator configurations.
5. Run SLAM in Gazebo: Choose a suitable SLAM algorithm, configure parameters, and launch the simulation to build a map of the virtual environment.
6. Port SLAM and navigation to real-world robot: Deploy the ROS nodes and navigation stack onto the Raspberry Pi + Arduino Mega setup.
7. Calibrate sensors and motor movements: Ensure accurate sensor readings and robot control by fine-tuning calibration parameters.
8. Test and refine navigation: Run the robot in a controlled environment, observe its behavior, and adjust parameters as needed to achieve optimal performance.

Environment Mapping:

Hardware: Same as LiDAR approach.

Software: Navigation stack with pre-built map capabilities (e.g., move_base)

Steps:

1. Create a digital map of your factory floor: Use CAD software or online tools to build a detailed map with obstacles, pathways, and landmarks.
2. Upload the map to the robot's navigation system: Convert the map to a ROS-compatible format and import it into the navigation stack.
3. Configure navigation parameters: Specify start and goal locations, robot dimensions, and safety settings within the pre-built map.
4. Deploy the navigation stack on the robot: Transfer ROS nodes and navigation configuration to the Raspberry Pi + Arduino Mega setup.
5. Test and refine navigation: Similar to LiDAR approach, run the robot in a controlled environment and fine-tune settings for efficient and safe movement.

2. Line Following using Waveshare Line sensor and RFID Tags with object avoidance and home base protocol, and interactive user interface to choose/alter path: (Most feasible option)

Hardware:

- Raspberry Pi 4
- Arduino Mega

- Waveshare Line Following Sensor
- Motor drivers and actuators for AGV movement

Software:

- Arduino IDE
- Line following library for the Waveshare sensor

Steps:

1. Connect the line following sensor to the Arduino Mega: Follow the Waveshare sensor instructions for proper wiring and configuration.
2. Upload the line following code to Arduino: Develop or acquire Arduino code specific to the Waveshare sensor that calculates steering commands based on sensor readings.
3. Connect Arduino to Raspberry Pi for communication: Establish communication between the Arduino (reading sensor data) and Raspberry Pi (controlling navigation) using serial communication tools.
4. Develop navigation logic on Raspberry Pi using RFID tags: Implement code on the Raspberry Pi that translates steering commands from the Arduino into motor control signals for the AGV.
5. Test and refine line following: Test the robot's ability to follow lines in your environment, adjust code and control parameters for smooth and accurate movement within the aisles

1.2.2 Task Automation

(1) What specific tasks should be automated by the AGV (e.g., material transport, machine loading)?

- **Fabric roll movement:** Transportation of rolls from storage to cutting stations and then to sewing machines, optimising fabric flow and reducing manual lifting.
- **Garment rack transportation:** Move finished garments between workstations for finishing, ironing, and quality control, streamlining the post-production process.
- **Box handling:** Handle boxes with raw materials, accessories, or finished products between storage, production areas, and shipping docks, improving inventory management and logistics.
- **Dynamic routing:** Implement software for adapting to changes in production flow and re-routing the AGV based on real-time information from sensors or production schedules, minimising delays and optimising material movement.

(2) Will the AGV operate in a fixed or dynamic environment?

An apparel manufacturing floor is a hybrid environment, where the overall layout of the workstations, machinery are well defined and change infrequently, on the other hand the material movement, worker traffic and the production schedules are less predictable from moment to moment due to rotating styles.

(3) Integration with existing systems:

3.1 How will the AGV integrate with other systems, such as warehouse management or production planning software?

No intention of integration with any prevalent systems (if present) as such. Possible ways of integration would involve the use of specific APIs that allow external systems like the ROS in our AGV to access and update data.

(4) Safety features:

4.1 What safety features need to be incorporated, including obstacle detection, emergency stops, and collision avoidance?

Emergency Stops:

- **Physical button:** Easily reachable emergency stop button on the robot for manual intervention.
- **Remote stop:** a remote control system for stopping the AGV from a distance in case of emergencies.
- **Automatic sensors:** Program the robot to automatically stop based on sensor data indicating collisions or unsafe situations.

4.2 How will the AGV ensure compliance with safety regulations?

Contingent on personnel accustomization, training and clear labelling of pathways and components.

(5) Programming and Control

5.1 What control modes are suitable for the AGV (manual, semi-autonomous, or fully autonomous)?

Semi-Autonomous Control:

- **Advantages:** Offers flexibility with human oversight, adaptable to dynamic changes in production flow, suitable for complex tasks.
- **Implementation:**
 - Utilisation of pre-programmed routes for typical material movement between workstations and storage areas.
 - Integration with production planning software or central control system for dynamic task prioritisation based on real-time production needs.

- Maintaining a human supervisor to monitor the AGVs operation and intervene in case of unexpected situations, and adjust routes as needed.

Enhanced Manual Control:

- **Advantages:** Precise control by human operators, lower development cost, simpler to implement compared to full autonomy.
- **Implementation:**
 - Utilisation of guidance systems like magnetic tape lines or virtual tracks on the floor for assisted navigation.
 - Development of an intuitive user interface for operators to control the AGV's movement, loading, and unloading functions.
 - Implementation of safety features like emergency stop buttons and collision avoidance sensors to ensure safe operation.

Hybrid Approach:

- **Advantages:** Combines the benefits of both methods, maximizing efficiency while maintaining human oversight for safety and flexibility.
- **Implementation:**
 - Combination of pre-programmed routes for routine tasks with manual control options for handling deviations or unforeseen situations.
 - Utilisation of sensors for obstacle detection and automatic avoidance within predefined zones, while allowing manual control for navigating dynamic areas.

(6) Communication Protocols:

6.1 What communication protocols will the AGV employ to integrate with other systems and devices?

Proposed approaches:

1. REST API:

- **Pros:** Familiar format for many developers, widely supported across different languages and platforms, easy to scale and extend in the future.
- **Cons:** Might require slightly more complex backend development compared to simpler protocols like MQTT.

2. MQTT (Message Queuing Telemetry Transport):

- **Pros:** Lightweight and efficient, suitable for resource-constrained devices like AGVs, ideal for real-time data exchange and updates.
- **Cons:** Less familiar than REST for some developers, might require specific libraries or frameworks for implementation.

3. WebSockets:

- **Pros:** Bidirectional communication for real-time interaction and control, lower latency compared to HTTP requests, suitable for interactive user interfaces.
- **Cons:** Requires server-side support for WebSockets, might not be as widely supported as REST or MQTT.

4. gRPC (Remote Procedure Calls):

- **Pros:** High performance and efficiency, optimized for data serialization and transport, good choice for streaming large amounts of data.
- **Cons:** Requires language-specific code generation tools, not as widely used as other options, additional dependencies and configuration might be needed.

The easiest way to achieve cross platform communication is to run pre-built software solutions, like moveo, ROS2 for Raspberry PI etc.

(7) Maintenance and Support:

7.1 What are the maintenance requirements for the AGV, including routine checks and software updates?

Changing of the battery, maintenance of connections within the processing unit and motor connections, cleaning of LiDAR sensors, remapping in case of layout changes, wheel and suspension maintenance, LiDAR calibration, battery health, motor and drive system, software updates if applicable

7.2 How will support be provided to address issues promptly?

Implementation and personnel training of maintenance schedules wrt the agv, which would include LiDAR cleaning, maintenance of motors etc, along with manual distribution which will contain all commonly faced problems and how to rectify them, including quick recalibration of sensors.

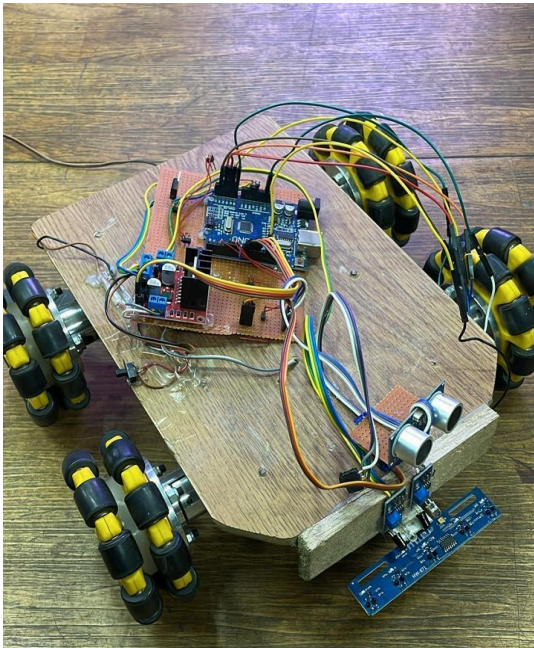
7.3 What is the testing plan to validate AGV functionality?

- We would start off with component testing, which means individually testing each sensor for accuracy and calibration, we verify motor control and drive system functionality, and assess the communication channels between the components (Raspberry PI 4 and Arduino Mega)
- Then, the next step would be navigation testing. We start off with testing basic movement commands. We would validate path planning and path following capabilities using pre-programmed routes. We would also evaluate obstacle avoidance and prevention mechanisms at this stage.
- The next step would be integration testing., We test interaction of the AGV with the user interface, verifying data exchange and visualisation of map, agv position and planned path. We validate emergency stop and recall homebase functionalities.
- Interaction of the physical body under required stress and weight would then be tested, defining intricacies and nuances such as the inability to put the RFID sensor on the metal base plate.
- System testing would consist of testing the algorithm in the deployment site. We evaluate the battery life and performance under realistic operating conditions.

1.3 Background Study

The initial weeks were dedicated to crucial decisions, navigation method exploration, LiDAR research, and prototype enhancements.

1.3.1 Objectives and Milestones:



(1) Product Finalization

Explored the choice between Automated Guided Vehicles (AGVs) and Autonomous Mobile Robots (AMRs) for the material transport system. Considered factors like payload capacity, flexibility, and cost-effectiveness.

(2) AGV Navigation Method Research

Conducted an in-depth study of various navigation methods suitable for industrial AGVs. Explored options such as magnetic tape guidance, inertial navigation, and vision-based systems. Considered factors like precision, adaptability to dynamic environments, and cost implications.

(3) AGV LiDAR VSLAM Research

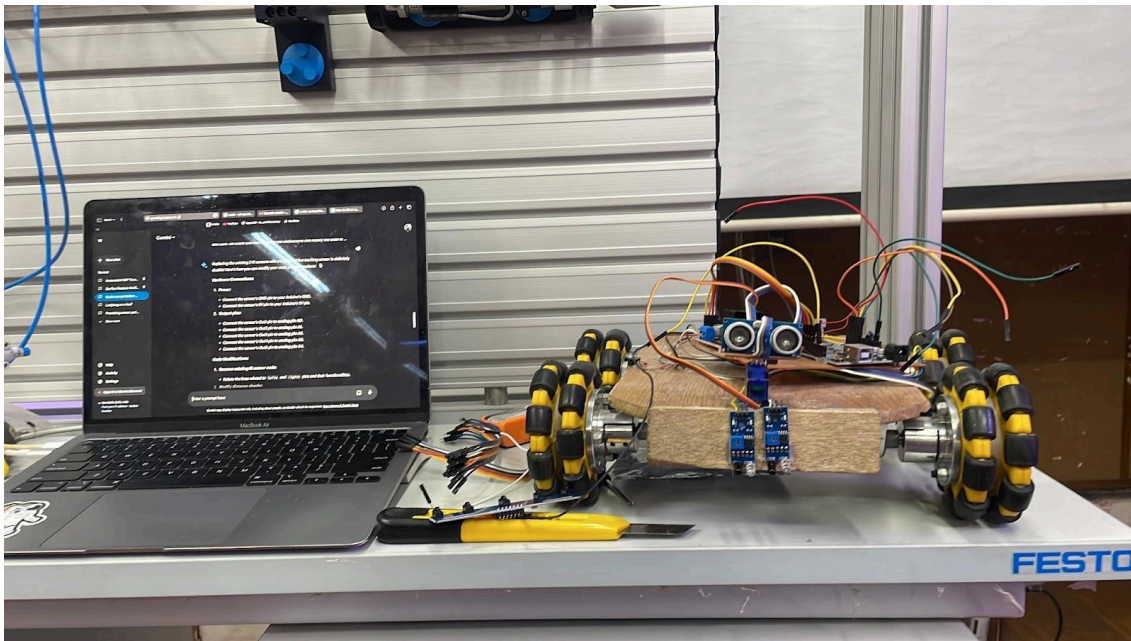
Focused on LiDAR-based Visual Simultaneous Localization and Mapping (VSLAM) for precise navigation. Explored the principles of LiDAR technology, assessing its compatibility with AGV navigation and potential advantages in real-time mapping.

(4) Functionality Finalization

Synthesized findings from navigation method and LiDAR research to finalize the overall functionality requirements for the AGV. Defined essential features such as real-time mapping, simplification of localisation and the navigation scope.

(5) Previous Prototype Repair and Rewiring and Addition of 5-channel IR sensor

Performed a comprehensive overhaul of the existing prototype to enhance stability. Addressed issues with wiring and general wear and tear. Added a 5-channel Infrared (IR) sensor for improved line following capabilities:



(6) Coding and Floor Setup Finalization

Implemented coding to achieve line following and object avoidance functionalities. Optimized the code for efficient performance by tweaking proportional KP gain value.. Finalized the physical setup of the industrial floor for testing, considering factors like layout and lighting conditions.

(7) RFID-Based Navigation Algorithm Research

Explored RFID-based navigation algorithms for potential integration into the AGV system. Investigated the RFID RC522 sensor's capabilities and limitations. Analyzed existing literature

on RFID-based navigation solutions.

(8) Coding for RFID Sensor and Arduino Extender Research

Initiated coding to reprogram the RFID sensor for recognizing different tags as unique IDs. Researched the addition of an Arduino extender or a different model to accommodate a WiFi shield, addressing the challenge of limited digital pins. Testing feasibility of RFID responsiveness and delay times for effective maneuvering.

(9) LiDAR VSLAM Research:

Investigated the principles of LiDAR and its application in VSLAM for real-time mapping and localization. Assessed compatibility with AGV navigation and noted the need for further testing in the specific industrial context. Researched on the feasibility of VSLAM mapping using ROS and Gazebo to attain real-time environmental awareness

During our dissertation, we explored various advanced navigation techniques beyond the chosen approach of RFID tags and path lookup for our AGV navigation system design, such as:

9.1 SLAM (Simultaneous Localization and Mapping):

- SLAM algorithms allow a robot to build a map of its environment while simultaneously keeping track of its location within that map. This is particularly useful for robots navigating unknown or dynamic environments.
- There are different flavors of SLAM, including:
 - **LiDAR SLAM (LA-SLAM):** This technique uses LiDAR data to create a 3D point cloud map of the environment. It's computationally expensive and requires a powerful processor.
 - **Visual SLAM (vSLAM):** This technique uses cameras to capture visual data and build a map of the environment. It can be computationally intensive depending on the chosen algorithms and processing power.
 - **ORB-SLAM (Odometry, Feature Tracking, and Loop Detection):** This is a vSLAM technique that uses features extracted from camera images and odometry data from wheel encoders to build a map. It's considered computationally less demanding compared to other vSLAM approaches.

1.3.2 Why SLAM would not be ideal for our AGV:

- (1) **Computational Complexity:** While ORB-SLAM offers reduced complexity compared to other vSLAM approaches, it still requires more processing power than the lookup table-based approach used in your system. This could be a concern for resource-constrained embedded systems like Arduinos often used in AGVs.
- (2) **Dynamic Environments:** SLAM is well-suited for situations where the environment can change. However, for a sewing floor environment, the layout remains static. The

simpler approach using RFID tags provides sufficient accuracy and reliability for this scenario.

- (3) **Cost:** LiDAR sensors can be significantly more expensive compared to RFID tags. For a structured environment with a pre-defined path, the cost-effectiveness of RFID tags outweighs the benefits of LiDAR in this scenario.
- (4) **Complexity:** While LiDAR data can be used for SLAM, it also requires additional processing power and software libraries to interpret the data and extract meaningful information for navigation. To run a complete ML model through real time LiDAR imaging, we require processing power beyond the scope of this project. The simpler approach using RFID tags avoids this complexity.

1.3.3 Why Magnetic Guide Sensors would not be ideal for our AGV:

- (1) **More complex design:** Requires specialized magnetic field sensors which can be pricier.
- (2) **Higher cost:** Sensors and potentially creating the magnetic field track can be more expensive.
- (3) **Setup complexity:** Needs a pre-established magnetic field, which can be trickier to implement.

Benefits of RFID Tags over Magnetic Guide Sensors:

- (1) **Flexibility and ease of re-routing:** With RFID tags, you can easily modify the path the AGV takes by simply placing or removing tags at strategic locations. This offers significantly more flexibility when compared to a fixed magnetic strip layout.
- (2) **Reduced Installation Complexity:** Installing RFID tags at specific points is much simpler and less disruptive than laying down a long magnetic strip across the entire path.

1.3.4 Chosen Approach:

- **RFID Tags and Path Lookup:** The final design leverages RFID tags placed at decision points (junctions, stops) on the sewing floor. The AGV uses an RFID sensor to read these tags and identify its location. A pre-defined lookup table or decision tree (implemented in `findPath.cpp`) on a central server uses this location information (current RFID tag) and the destination tag (received from client input) to determine the appropriate movement command (straight, left, right, stop). This approach offers a simpler and more reliable solution for the structured environment of a sewing floor.

1.4 Core logic development (Without SLAM)

1.4.1 Ways to develop line following navigation system

There are several ways to develop a navigation system without LIDAR or SLAM, each with its own advantages and disadvantages:

1. Line following with zone identification:

Concept: Use line sensors to follow a dedicated path throughout the floor. Divide the floor into zones using additional lines or markers. Based on the zone it enters, the AGV identifies the appropriate sewing line it needs to deliver to.

Implementation:

- **Line sensors:** Mount multiple line sensors on the front of your AGV, ensuring they can reliably detect the designated path marking (e.g., black electrical tape on a white floor).
- **Zone markers:** Divide your sewing floor into zones using perpendicular lines or markers (e.g., colored tape squares) placed strategically. Each zone should correspond to a specific sewing line.
- **Control logic:** Program the Arduino controller to:
 - Follow the designated line using the sensor readings.
 - Identify zone transitions based on encountering the perpendicular markers.
 - Trigger specific actions (e.g., stop, turn) upon entering a zone associated with the target sewing line.

Advantages:

- Simple and cost-effective, requiring only basic Arduino programming and readily available sensors.
- Reliable for fixed layouts as long as the line markings are maintained.

Disadvantages:

- Requires careful planning and placement of line markings and zone indicators.
- Inflexible for layout changes, as any adjustments would necessitate modifying the line markings.
- Accuracy depends on the quality of line detection and zone marker visibility.

2. RFID tags and readers:

Concept: Place RFID tags at specific points on the floor (e.g., loading stations, sewing lines). Equip the AGV with an RFID reader to detect these tags. Program the AGV to navigate based on the sequence of tag detections.

Implementation:

- **RFID tags:** Attach unique RFID tags at designated points on the floor, such as loading stations and sewing line starting points. Choose tags with appropriate reading range and durability for your environment.
- **RFID reader:** Mount an RFID reader module on your AGV, ensuring it can detect tags within the desired range.
- **Control logic:** Program your Arduino controller to:
 - Continuously scan for RFID tags as the AGV moves.
 - Identify the specific tag corresponding to the target loading station or sewing line based on its unique ID.
 - Trigger actions based on the detected tag (e.g., stop, turn) to reach the designated point.

Advantages:

- More flexible than line markings, allowing for easier layout changes by simply moving the RFID tags.
- Potentially higher accuracy if the tags are placed strategically and the reader has good range.

Disadvantages:

- Requires additional investment in RFID tags and reader module.
- Tag placement and maintenance become crucial for system functionality.
- Metal objects or other interfering signals might affect tag detection reliability.

3. Dead reckoning:

Concept: Track the AGV's movements based on odometry data from its wheels or encoders. Program specific movements (e.g., turn left X degrees, move forward Y distance) to reach designated points.

Implementation:

- **Encoders:** Equipping each wheel of our AGV with encoders to track its rotations. Alternatively, usage of a motor driver module that provides odometry data can also be done.
- **Control logic:** Program our Arduino controller to:

- Calculate the AGV's movement based on encoder data (distance and direction).
- Store and track its position on a virtual map of our sewing floor (predefined within the program).
- Plan and execute movement commands (e.g., move forward X cm, turn left Y degrees) to reach the target point based on the virtual map.

Advantages:

- No additional infrastructure needed beyond encoders or odometry data.
- Potentially low-cost solution depending on your existing setup.

Disadvantages:

- Prone to errors due to wheel slippage, uneven terrain, or encoder inaccuracies.
- Requires precise calibration and tuning to achieve acceptable accuracy.
- Errors accumulate over time, making long-distance navigation unreliable.

4. Vision-based navigation:

- **Concept:** Equip the AGV with a camera and use image processing to identify visual markers placed strategically on the floor. Program the AGV to navigate based on the recognition of these markers.
- **Advantages:** More flexible than fixed lines or tags, potentially high accuracy if implemented well.
- **Disadvantages:** Requires significant computational power and image processing expertise, might be sensitive to lighting changes.

Since this method won't work on Arduino, we did not explore this further.

1.4.2 Ways to feed markers in line following approach

1. Pre-programmed sequences:

- **Concept:** Define specific sequences of line segments and turns for each route between zones. Store these sequences within your Arduino program.
- **Implementation:**
 - Map your sewing floor and identify desired paths between zones.
 - Break down each path into a series of straight line segments and turns (e.g., "move forward 2m, turn right 90 degrees, move forward 3m").
 - Store these sequences for each route in your Arduino program.
 - When instructed to move between zones, the AGV retrieves the corresponding sequence and executes it step-by-step following the line and performing turns.

Advantages:

- Simple and predictable behavior.
- No need for additional infrastructure beyond line markings.

Disadvantages:

- Inflexible for changes in layout or additional routes.
- Requires manual programming of each route, which can be time-consuming for complex layouts.

2. Decision points with markers:

- **Concept:** Use strategically placed markers at key decision points where the AGV chooses its next direction based on the target zone.
- **Implementation:**
 - Identify key intersections or decision points on your sewing floor.
 - Place unique markers (e.g., different colored squares) at each decision point, corresponding to the possible directions leading to different zones.
 - Program the AGV to recognize these markers using line sensors or color sensors.
 - Based on the target zone and the marker encountered, the AGV chooses the appropriate next line segment (pre-programmed within the Arduino) to follow.

Advantages:

- More flexible than fixed sequences, allowing for some changes in layout without reprogramming.
- Requires fewer markers compared to full path marking.

Disadvantages:

- Requires careful marker placement and reliable sensor detection.
- Complexity increases with more decision points and possible routes.

3. Hybrid approach:

- **Concept:** Combine pre-programmed sequences with decision points for a balance between flexibility and simplicity.
- **Implementation:**
 - Use pre-programmed sequences for major sections of the path between zones.

- At key decision points, rely on markers to choose the correct direction based on the target zone.

Advantages:

- Offers more flexibility than fixed sequences without requiring extensive marker placement.
- Reduces programming complexity compared to fully marker-based navigation.

Disadvantages:

- Still requires some pre-programming and careful marker placement.
- Might need adjustments if layout changes significantly.

1.4.3 Defining the route through RFID tags

1. Pre-programmed route mapping:

- **Concept:** Define a map of your sewing floor and the connections between points (loading stations, sewing lines) where RFID tags are placed. Store this map within your Arduino program.
- **Implementation:**
 - Map your sewing floor and identify desired paths between points marked by RFID tags.
 - Encode this map within your Arduino program, specifying connections between tags (e.g., "tag X leads to tag Y and Z").
 - When instructed to move from point A (tag A) to point B (tag B), the AGV retrieves the corresponding route information from the map.
 - It follows the path by moving to the next tag specified in the map, reading its ID, and repeating until reaching tag B.

Advantages:

- Simple and predictable behavior.
- No need for additional infrastructure beyond RFID tags.

Disadvantages:

- Inflexible for changes in layout or additional routes.
- Requires manual programming of the entire map, which can be time-consuming for complex layouts.

2. Centralized control system:

- **Concept:** Use a separate computer or microcontroller as a central control system that receives tag information from the AGV and sends navigation instructions back.
- **Implementation:**
 - Set up a central system connected to the AGV that can communicate (e.g., WiFi, Bluetooth).
 - When instructed to move between points A and B, the AGV communicates its current location (tag ID) to the central system.
 - The central system retrieves the optimal route from a pre-defined map stored within itself.
 - It sends step-by-step instructions to the AGV, specifying which tag to read next.

Advantages:

- More flexible for changes in layout or adding new routes by updating the central map.
- Reduces programming complexity on the AGV itself.

Disadvantages:

- Requires additional hardware and software for the central system.
- Introduces potential communication delays and complexity.

3. Hybrid approach:

- **Concept:** Combine pre-programmed routes on the AGV with centralized control for specific decision points.
- **Implementation:**
 - Program the AGV with basic routes between frequently used points using tag IDs.
 - For complex intersections or less frequent destinations, rely on the central system to provide guidance based on the current tag and target destination.

Advantages:

- Offers more flexibility than fixed routes without requiring extensive central control.
- Reduces overall programming complexity.

1.4.4 Shortest Path Finding Algorithms

1. Map Representation:

- **Graph Data Structure:** Represent your sewing floor layout as a graph, where stations are nodes and connections between them are edges. Each edge can optionally have a weight representing distance or travel time.

- **RFID Node Mapping:** Assign unique IDs to each station and store their corresponding node IDs in the graph data structure. This simplifies tag identification and avoids complex logic based on specific tag readings.

2. Navigation Logic:

- **Pathfinding Algorithm:** Choose an appropriate algorithm to calculate the shortest path between source and destination stations based on the graph data.
- **Line Following Integration:** Once the shortest path is calculated, develop logic to guide the AGV along the path using line sensors. This might involve breaking down the path into segments and using line following for each segment, adjusting for turns and intersections based on the graph data.

Standard Shortest Path Algorithms:

- **Dijkstra's Algorithm:** As mentioned earlier, this is a popular choice for non-negative edge weights and efficiently finds the shortest path from a single source to all other nodes.
- *A Search:** This algorithm excels in finding the shortest path while considering estimated distances to the destination, making it faster for certain scenarios.
- **Bellman-Ford Algorithm:** This handles negative edge weights but might be slower than Dijkstra's for non-negative weights.
- **Floyd-Warshall Algorithm:** This pre-computes all shortest paths between all pairs of nodes in the graph, allowing for fast retrieval but requires more memory storage.

Recommendation:

For this specific case, **Dijkstra's Algorithm** is a good starting point. If your floor layout is relatively simple and the edge weights directly represent distances, Dijkstra's might be sufficient. Since we dropped this method due to its complexity, more on the algorithm is elaborated in Part III : Chapter 4.

1.5 Controller and Language

This section details the hardware platform and software development environment chosen for the Automated Guided Vehicle (AGV) controller. The selection of these components plays a crucial role in the overall functionality, performance, and ease of development for the AGV system.

Hardware Selection:

The central processing unit (CPU) for the AGV controller is the Arduino Mega 2560. This microcontroller provides a robust platform with ample processing power and numerous input/output (I/O) pins, enabling the connection of various sensors and actuators required for

the AGV's operation. In our earlier iteration of this model, the main challenge that we faced with the microcontroller at the time (Arduino Uno) was the lack of I/O pins and the limited computing power, which facilitated our move towards the Mega 2560. The specific choice of additional modules is elaborated upon in a separate section.

Development Language:

C++ was selected as the programming language for the AGV controller. While a subset of the full C++ language is typically used in Arduino development environments, it offers a well-established balance between power and ease of use for embedded systems programming. This language familiarity allows for efficient code development and leverages existing knowledge of C++ syntax and programming paradigms.

Development Environment:

The chosen Integrated Development Environment (IDE) for the AGV project is Visual Studio Code (VS Code) with the PlatformIO extension installed. VS Code offers a versatile and customizable platform for writing and editing C++ code, providing features like syntax highlighting, code completion, and debugging support. PlatformIO, a powerful extension specifically designed for embedded systems development within VS Code, further simplifies the development process by streamlining various tasks, including:

- **Board Management:** PlatformIO facilitates the selection and installation of specific Arduino board definitions and libraries required for the AGV project.
- **Library Management:** Installing and managing external libraries for specific sensors or functionalities becomes a straightforward process.
- **Build Automation:** PlatformIO automates the code compilation process, ensuring efficient translation of the C++ code into instructions the Arduino can understand.
- **Debugging:** PlatformIO integrates debugging functionalities within VS Code, allowing for step-by-step code execution and identification of potential errors.
- **Flashing:** Uploading the compiled code to the Arduino board is simplified with PlatformIO's flashing options, eliminating the need for manual steps typically involved in traditional Arduino IDE workflows.
- **Monitoring:** PlatformIO offers tools to monitor the serial output from the Arduino while it's running, allowing for real-time observation of sensor data and debugging of program behaviour.

CHAPTER 2: Hardware Design

2.1 System Configuration

As studied in Part I: Section 3.3 above, we have followed the same system configuration for our final model. However, instead of one wheel on each side of the vehicle, there are two. For understanding, it can be considered that we have divided each wheel from our reference into two wheels. Same goes for the motors, instead of two in the reference, we have four.

2.2 Vehicle Specifications

2.2.1 Dimensions of the AGV

The payload that we decided while establishing our functionalities was 20 kg. The size of a 20 kg carton full of fabric cut pieces depends on several factors:

- **Fabric type:** Different fabrics have varying densities (weight per unit volume). Lighter fabrics like chiffon will occupy a larger volume than heavier fabrics like denim for the same weight.
- **Piece size and shape:** Smaller, flat fabric pieces will pack more densely than larger or irregularly shaped pieces, leaving less empty space in the carton.
- **Packing efficiency:** How tightly the pieces are packed will significantly impact the overall volume of the carton. Tight packing leaves less air space and reduces the overall size.

Here's a rough estimation based on some assumptions:

1. Fabric Density and Packing Efficiency:

- **Fabric density:** We'll consider a range of 100 GSM to 150 GSM, representing typical cotton shirting fabrics.
- **Packing efficiency:** Assuming reasonably good packing with some air space, let's use a 70% packing efficiency.

2. Estimating Fabric Volume: We'll need to convert GSM (fabric weight per unit area) to a density value (fabric weight per unit volume) for volume calculations.

Density conversion (assuming an average of 125 GSM):

$125 \text{ GSM} = 0.125 \text{ kg/m}^2$ (divide GSM by 1000 to get kg/m^2)

Fabric volume calculation:

Fabric volume = Total weight / (Density x Packing efficiency)

Volume = 20 kg / (0.125 kg/m³ x 0.7)

Volume ≈ 22.22 m³ (assuming 125 GSM)

Volume variation based on density:

Since the fabric density is a range (100-150 GSM), the fabric volume will also vary. Here's an adjusted volume range:

- Lower density (100 GSM): Volume = 20 kg / (0.1 kg/m³ x 0.7) ≈ 28.57 m³
- Higher density (150 GSM): Volume = 20 kg / (0.15 kg/m³ x 0.7) ≈ 18.67 m³

3. Estimating Carton Size:

- **Lower density (100 GSM):** Volume ≈ 28.57 m³

Possible carton dimensions: Length: 0.7 m, Width: 0.8 m, Height: 0.5 m

- **Higher density (150 GSM):** Volume ≈ 18.67 m³

Possible carton dimensions: Length: 0.6 m, Width: 0.7 m, Height: 0.4 m

Using these possible carton dimensions, we finalize the size of the AGV:

0.5 m in length and 0.4 m in breadth, i.e. 50cm x 40cm.

Thus, the vehicle specifications come out to be as noted in the table below:

Sr.no.	Parameter	Value	Reason
1.	Payload	20 kgs	Fixed in functionalities
2.	Estimated weight of AGV	5 kgs	Fixed for calculation
3.	Maximum speed	1 m/s	Approximate or desired
4.	Static friction coefficient	0.7	Value taken from Google for industry surfaces
5.	Dynamic friction coefficient	0.5	Value taken from Google for industry surfaces
6.	Acceleration	1 m/s ²	Approximate or desired
7.	Wheel radius	5 cm	Decided by availability, strength and ground clearance

2.3 Calculation for Parts selection

2.3.1 Motors

Stall torque vs Rated torque: The stall torque is the maximum torque a motor can produce at zero speed. However, most motors are not designed to operate at stall torque for extended periods. The rated torque is a more realistic value for continuous operation.

The relationship between power and torque of a DC motor is described by the following equation:

$$\text{Power (P)} = \text{Torque (T)} \times \text{Angular Velocity (\omega)}$$

Where:

- **P** is the power in watts (W)
- **T** is the torque in newton-metres (Nm)
- **ω** is the angular velocity in radians per second (rad/s)

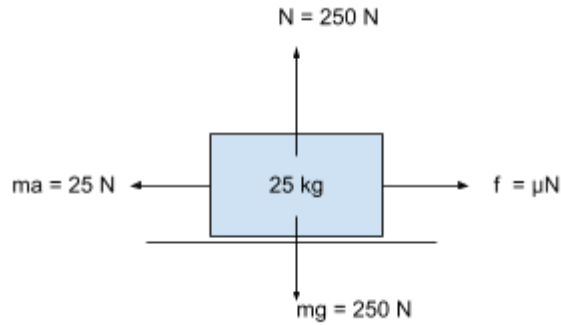
Here's a breakdown of what this means:

- **Higher Torque, Lower Speed:** If a DC motor produces a high amount of torque, it can exert a strong rotational force. However, this comes at the expense of its rotational speed (angular velocity). In this case, even though the torque is high, the overall power output (product of torque and speed) might not be very high if the speed is low.
- **Lower Torque, Higher Speed:** Conversely, a DC motor can achieve a high rotational speed with a lower torque output. This can be useful for applications where speed is a priority. In this scenario, the motor might have a lower torque, but if the speed is high enough, the overall power output can still be significant.

In the context of our Line Follower AGV:

We prioritised a motor that offers a **higher torque and lower speed**. It is needed to overcome friction and move the AGV with its payload and self weight, the speed need not be very high.

For the precise torque calculation, observe the free body diagram below:



Here,

- Total weight = 25 kgs
- Speed = 1 m/s
- Static friction coefficient (μ_s) = 0.7
- Dynamic friction coefficient (μ_d) = 0.5
- Acceleration (a) = 1 m/s²
- Wheel radius = 0.05 m

$f = \mu N$, where N is 250 Newtons

$$f_s = \mu_s N = 0.7 \times 250 = 175 \text{ N}$$

$$f_d = \mu_d N = 0.5 \times 250 = 125 \text{ N}$$

$$\text{Total force to start motion} = f_s + ma = 175 + 25 \text{ N} = 200 \text{ N}$$

$$\tau = rF \sin \theta \text{ (here, } \sin \theta = 1 \text{ as the angle is } 90^\circ \text{)}$$

$$\tau_{\text{stall}} = 0.05 \times 200 = 10 \text{ Nm}$$

$$\text{Force to continue motion} = f_d + ma = 125 + 25 \text{ N} = 150 \text{ N}$$

$$\tau_{\text{rated}} = 0.05 \times 150 = 7.5 \text{ Nm}$$

We have considered a total mass of 25 kgs and a single wheel for simplified calculations. So among the four wheels the weight will actually divide, which means the final torque values divided by 4 will be the torque rating of each motor.

Therefore, for each motor, the torque rating should be:

$$\tau_{\text{stall}} = 2.5 \text{ Nm}$$

$$\tau_{\text{rated}} = 1.875 \text{ Nm}$$

Considering that there will be losses, we will select a motor with a slightly higher torque rating than this. Also note that the maximum speed does not play any role in the calculation and motors have their maximum rpm fixed. So we will select a motor that gives us a decent speed but the torque rating has to be higher than the calculated torque.

2.3.2 Voltage requirement/ Battery

Power Hungry Components:

- **Motors:** The main source of power consumption are the DC motors driving the AGV wheels. The number of motors, their individual voltage ratings, and their operating current are considered here.
- **Control System:** The electronic components responsible for controlling the AGV (microcontroller, motor drivers) will also have a power consumption, typically specified at a certain voltage level, however this will be very less compared to the power consumed by the motors.

For ease of finding parts and making the full circuit, we will fix our maximum potential difference at 12V. **Thus the motors and battery selected will have a rating of 12 V.**

2.3.3 Calculating Total Power Consumption:

1. Power Consumption:

- **Motors:** The main power consumers will be the DC motors driving the wheels.
 - Number of motors = 4
 - Voltage rating per motor = 12 V
 - Current drawn per motor at operating speed = 0.3 A (at one wheel load, this will increase when the weight is higher, thus a buffer will be taken here)
- **Control System:** This is much lower than the motors and will be easily accommodate in the buffer.

2. Total AGV Power:

Total Power (P) = (Number of Motors x Voltage per Motor x Current per Motor)

$$= (4 \times 12 \times 0.3)$$

$$= 14.4$$

Since the current demand will increase as the weight increases, we will add a buffer of ± 5 W. **Therefore the maximum total power consumption will be around 20 W.**

3. Battery Voltage:

We have already decided that to be **12V** in accordance with the motors.

4. Operating Time and Buffer:

- **Operating Time:** For the prototype, let's consider 2- 3 hours as the operating time for the AGV on a single battery charge.
- **Buffer:** The time is between 2- 3 hours considering a 20-30% buffer to the total power to account for inefficiencies, peak loads during acceleration, and other losses.

5. Calculating Battery Capacity:

Battery Capacity (Ah) = (Total Power x Operating Time) / Battery Voltage = (20 x 3) / 12 V

Therefore the total battery capacity required is 5 Ah.

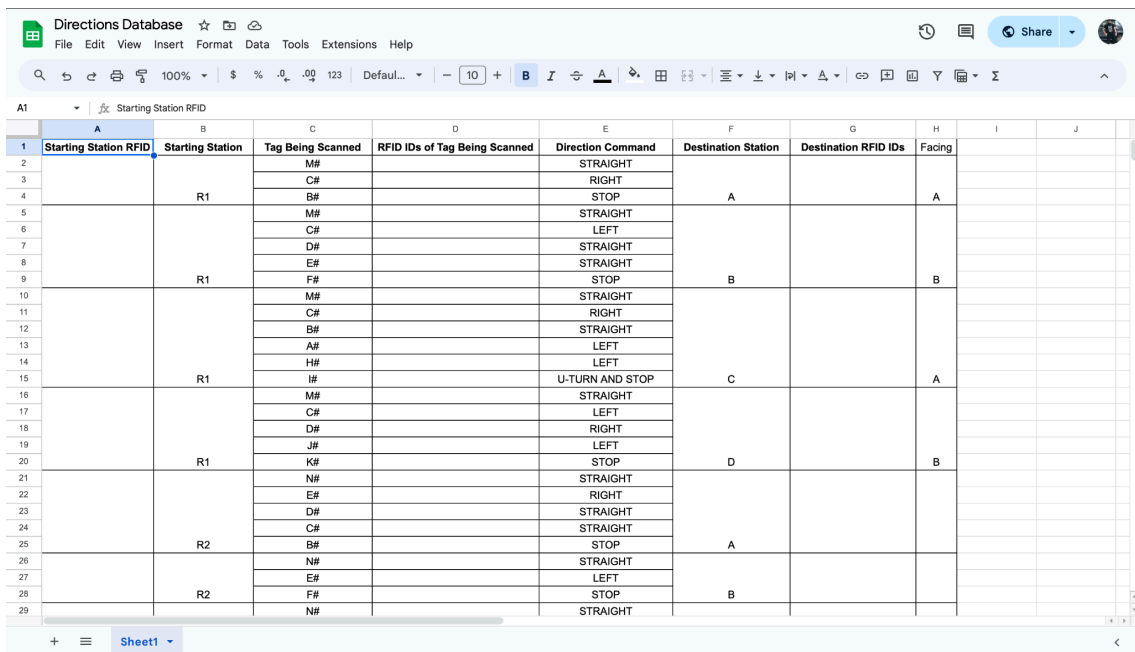
CHAPTER 3: Software Design

Platform for code development is Visual Studio Code through the help of platformIO extension for easy maintenance, debugging and deployment.

We started off with recognising the main problem: **how do we make a robot navigate its way through a floor without any real-time spatial awareness?**

The solution we came to was the deployment of a predefined path around stations with RFID tags serving as unique ID markers for the AGV to recognise and make decisions upon sensing. We then created a mockup layout of a floor and created a directions database for the AGV to aid us during further creation of a tree map of the direction decisions that it has to make to go from any point A to point B.

Directions Database:



1	Starting Station RFID	Starting Station	Tag Being Scanned	RFID IDs of Tag Being Scanned	Direction Command	Destination Station	Destination RFID IDs	Facing
2			M#		STRAIGHT			
3			C#		RIGHT			
4		R1	B#		STOP	A		A
5			M#		STRAIGHT			
6			C#		LEFT			
7			D#		STRAIGHT			
8			E#		STRAIGHT			
9		R1	F#		STOP	B		B
10			M#		STRAIGHT			
11			C#		RIGHT			
12			B#		STRAIGHT			
13			A#		LEFT			
14			H#		LEFT			
15		R1	I#		U-TURN AND STOP	C		A
16			M#		STRAIGHT			
17			C#		LEFT			
18			D#		RIGHT			
19			J#		LEFT			
20		R1	K#		STOP	D		B
21			N#		STRAIGHT			
22			E#		RIGHT			
23			D#		STRAIGHT			
24			C#		STRAIGHT			
25		R2	B#		STOP	A		
26			N#		STRAIGHT			
27			E#		LEFT			
28		R2	F#		STOP	B		
29			N#		STRAIGHT			

This database recognised markers around the predefined path and allowed us to visualise a virtual map from rack to station, station to rack, station to station and so forth.

One particular consideration that we had to make while designing the road-map for the AGV was the direction in which the AGV was facing while stopping at every station. Since multiple facing directions would have taken longer to code, we decided to have common facing directions for stations that are on the same side of the map, effectively allowing us to simplify the code further.

We then moved on to the coding process, which started off by fetching the unique IDs from every RFID tag that we used to mark junctions and stops. This was done using a simple fetch-ID code using the RC522 sensor:

```
#include <SPI.h>
#include <MFRC522.h>
#define RST_PIN 5
#define SCK_PIN 52
#define MOSI_PIN 51
#define MISO_PIN 50
#define SS_PIN 53

MFRC522 mfrc522(SCK_PIN, MISO_PIN, MOSI_PIN, RST_PIN, SS_PIN);

void setup() {
  Serial.begin(9600);
  SPI.begin();
  mfrc522.PCD_Init();
}

void loop() {
  if ( ! mfrc522.PICC_IsNewCardPresent()) {
    return;
  }

  if ( ! mfrc522.PICC_ReadCardSerial()) {
    Serial.println("Reading card failed!");
    return;
  }

  Serial.print("Card UID: ");
  for (byte i = 0; i < mfrc522.uid.byteSize; i++) {
    Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? "0" : "");
    Serial.print(mfrc522.uid.uidByte[i], HEX);
  }
  Serial.println();

  // Wait for removal of the card
  while (mfrc522.PICC_IsNewCardPresent());
}
```

The AGV works by combining two directories, namely **ESP (Controlling the ESP32 module)** and **Mega (Controlling the Arduino Mega 2560)**. The system comprises of two main parts:

- **Client-side:** A web user interface (`app.html`) for users to interact with the AGV.
- **Server-side :** An Arduino program (`main.cpp`) running on the AGV's microcontroller that controls movement and communicates with the client. An additional file, `localWifi.cpp` is used in a different configuration for web server functionality hosted on the ESP8266 module connected to the Arduino Mega 2560.

3.1 ESP Directory

File 1 (`localWifi.h`)

This file contains definitions for functions used to connect the robot to WiFi and create a web server.

- `localWifi::localWifi()`: This function initializes the WiFi module on the robot.
- `localWifi::connectWifi(String ssid, String password, void (*function)())`: This function connects the robot to a WiFi network specified by the `ssid` and `password` arguments. The `function` argument is a callback function that gets executed while connecting.
- `localWifi::runServer()`: This function keeps the web server running.
- `localWifi::sensorServer(String apName, String apPassword, String pageName, String *sensorData, String *data, void (*function)())`: This function creates a web server that serves a webpage (`pageName`) and allows communication between the robot and a client (potentially a phone or computer).
 - Clients can send data to the server through a POST request to the `/data` path.
 - Clients can receive sensor data from the robot by requesting the `/data` path with a GET request.
 - The `function` argument is a callback function that gets executed when new data is received from the client.

File 2 (`localwifi.cpp`)

This code defines the implementation for the `localWifi` class, which manages WiFi connectivity and a local web server for the robot controller. Here's a breakdown of the key functions:

Constructor (`localWifi::localWifi()`):

- Initializes the SPIFFS filesystem (used for storing web server files) for ESP8266 boards.

- There's a commented-out section for ESP32 boards (might require different initialization).

connectWifi(String ssid, String password, void (*function)()):

- This function connects to a WiFi network specified by **ssid** and **password**.
- It checks the current WiFi mode and sets it to **WIFI_AP_STA** (combined access point and station mode) if necessary.
- It establishes a connection using **WiFi.begin** and includes a placeholder for an optional callback function (**function**) that can be used during the connection process.

runServer():

- This function handles incoming client requests for the web server for a short duration (100 milliseconds).
- This prevents the server from blocking the main loop for extended periods.

sensorServer(String apName, String apPassword, String pageName, String *sensorData, String *data, void (*function)()):

- This function sets up a local web server with an access point named **apName** and password **apPassword**.
- It defines three web server handlers:
 - **/**: This handler serves the web page content from the file specified by **pageName**. It includes some JavaScript code for sending data to the server using a POST request.
 - **/data** (POST): This handler receives data sent from the web page and stores it in the ***data** variable. It also calls the optional callback function (**function**).
 - **/data** (GET): This handler returns the current value of the ***sensorData** variable.
- This function demonstrates how the server can send and receive data with a web page.

setServer(String apName, String apPassword):

- This function sets up the WiFi access point with the specified name and password.
- It configures WiFi mode, sleep behavior, and sets a static IP address for the access point.
- Some commented-out lines suggest additional configuration options (like transmit power and long-range mode) that might be specific to certain development boards.

File 3 (main.cpp)

This file is the main program that runs on the robot.

- It includes the necessary libraries (`localWifi.h`, `path.h`, `MFRC522.h`, and `SoftwareSerial.h`).
- It defines several variables:
 - `sendDatatoBot`: This variable is a serial object used to send commands to a motor controller or another device that controls the robot's movement.
 - `bot`: This variable is an instance of the `findPath` class (defined in file 4). It's used to determine the path the robot needs to take.
 - `app`: This variable is an instance of the `localWifi` class (defined in file 1). It's used to manage the WiFi connection and web server.
 - Several string variables are used to store data like starting and final tags, received data from the server, etc.
- The `setup()` function:
 - Initializes serial communication for debugging purposes.
 - Initializes SPI communication for the RFID reader.
 - Configures the WiFi access point and starts the web server using the `localWifi` library functions.
- The `loop()` function:
 - Runs the web server using the `localWifi` library function.
 - Periodically checks for new data from the server. If new data is available, it updates the `finalTag` and sends a movement command to the motor controller based on the current tag, starting tag, and final tag using the `findPath` object.
 - Reads data from the RFID reader and updates the `currentTag` if a tag is detected. If a new tag is read, it sends a movement command to the motor controller based on the current tag, starting tag, and final tag using the `findPath` object.

File 4 (path.h)

Preprocessor Directives:

- `#ifndef path_h`: This line checks if a macro named `path_h` has already been defined.
- `#define path_h`: If `path_h` is not defined, this line defines it, preventing the file from being included multiple times in the compilation process. This is a common practice to avoid including header files twice, which can lead to errors.
- `#include <Arduino.h>`: This line includes the Arduino library, which provides essential functions for interacting with Arduino boards.

Constant Definitions:

- Several lines define constants using the `#define` directive. These constants represent tag IDs (e.g., `r1Tag`, `aTag`, `T1`), movement commands (e.g., `moveStraight`, `moveLeft`), and other relevant values for the path planning logic.

`findPath` Class:

- The code defines a class named `findPath`.
- This class implements the logic for determining the path the robot needs to take based on its current location (represented by `currentTag`) and the destination tag (`finalTag`).
- The class has several public member functions:
 - `pathA(String *currentTag, String *finalTag)`: This function calculates the path for route A based on the current and final tags. It takes pointers to string variables containing the current and final tag values. It returns a string representing the movement command (e.g., "straight", "left") for route A.
 - Similar functions are defined for paths B, C, D, R1, R2, R3, and R4. Each function calculates the specific movement commands for its corresponding route.

File 5 (path.cpp)

This file defines the member functions of the `findPath` class that calculate the movement commands needed for the robot to reach a destination tag from a starting tag location.

Functions:

- `pathA(String *currentTag, String *finalTag)`: This function calculates the path for route A based on the current and final tags. It takes pointers to string variables containing the current and final tag values. It checks the `finalTag` and compares it with various predefined tags (`r1Tag`, `r2Tag`, etc.) and the `currentTag`. Based on the combination, it returns a string representing the movement command ("straight", "left", "right", "uturn", or "stop"). Similar logic is implemented in all the other path functions (`pathB`, `pathC`, and so on).
- `pathR1(String *currentTag, String *finalTag)` to `pathR4(String *currentTag, String *finalTag)`: These functions seem to be for "Return" paths, possibly used when the robot needs to return to specific locations (like starting points) based on the current and final tags. They follow a similar logic as the `pathA` function but handle different final tag destinations.

File 6 (app.html)



HTML Structure:

- The code defines an HTML document with three sections using `<div>` elements.
- Each section contains four buttons representing potential destinations or actions for the AGV.
- Buttons have a class "buttonClass" for styling and individual IDs like "btnR1" for easy referencing.
- A basic CSS section defines styles for the buttons (size, font size) and a class "black-and-white" that can potentially be used to grayscale the interface (not currently applied).

JavaScript Functionality:

- Two JavaScript objects (`mapToSensor` and `sensorToMap`) act as dictionaries mapping between user interface button labels (like "R1") and actual sensor data (like "52-43-FC-1B"). This represents the unique identifiers of the RFID tags.
- The `sData(data)` function takes a button label (`data`) as input.
 - It calls a function `sendData(mapToSensor[data])` sends the corresponding unique RFID ID (e.g., "52-43-FC-1B") to the AGV control system.
 - It logs the button label (`data`) to the console for debugging purposes.
 - It iterates through all buttons using `document.querySelectorAll('.buttonClass')` and resets their background color to a neutral gray ("rgb(92, 92, 92)").
 - It sets the background color of the clicked button (identified by "btn" + `data`) to green ("rgb(0, 255, 0)") to visually indicate selection.

- It updates a variable `lastClickedButtonId` to store the ID of the most recently clicked button.
- An `setInterval` function is used to periodically (every 2 seconds) retrieve sensor data from the AGV (using a function `getData()`).
 - The retrieved sensor data is logged to the console.
 - The function uses the `sensorToMap` dictionary to map the sensor data back to the corresponding button label (e.g., "R1" for "52-43-FC-1B").
 - Similar to the `sData` function, it resets all button background colors and then highlights the button corresponding to the received sensor data (red - "rgb(255, 0, 0)") and keeps the previously clicked button green.

Overall Functionality:

This code provides a user interface for sending commands to the AGV control system. Users can click buttons labeled with destinations or actions (represented by "R", "T", "A", etc.). Clicking a button sends the corresponding sensor data (likely RFID tag identifiers) to the AGV. The system also periodically retrieves sensor data from the AGV and visually indicates on the interface which tag was most recently detected. This creates a two-way communication between the user and the AGV.

3.2 Mega Directory

File 1 (main.cpp)

This code (`main.cpp`) controls a mobile robot using movement commands received over a serial connection. Here's a breakdown of the key components:

Includes:

- `<Arduino.h>`: Provides core functionality for Arduino boards.
- `<SoftwareSerial.h>`: Enables software serial communication using digital pins.

Pin Definitions:

- `A8, A9`: Pins used for software serial communication to receive data.
- `leftmotorforward, leftmotorback, rightmotorforward, rightmotorback`: Digital pins connected to the robot's motor controls.

Movement Macros:

- `moveStraight`, `moveLeft`, `moveRight`, `moveBack`, `moveStop`: String constants representing movement commands.

Software Serial Object:

- `getData`: A `SoftwareSerial` object configured on pins A8 and A9 for serial communication at 9600 baud.

Motor Control Functions:

- `forward()`: Sets motor pins to move the robot forward.
- `backward()`: Sets motor pins to move the robot backward. (Not used in the current loop)
- `leftward()`: Sets motor pins to turn the robot left and then move forward for 3 seconds (`delay(3000)`) before going straight.
- `rightward()`: Sets motor pins to turn the robot right and then move forward for 3 seconds (`delay(3000)`) before going straight.
- `stop()`: Sets motor pins to stop the robot.

Setup Function (`setup()`):

- Initializes serial communication for both the Arduino board (9600 baud) and the software serial object (`getData`).
- Sets pins connected to the robot motors (8-11) as outputs.

Loop Function (`loop()`):

- Continuously checks if data is available on the software serial port (`getData`).
- If data is available:
 - Reads the data string using `readStringUntil('\r')`.
 - Prints the received data string to the Arduino serial monitor (`Serial.println(data)`)
 - Compares the received data with the movement macros:
 - If `moveStraight`, calls the `forward()` function to move forward.
 - If `moveLeft`, calls the `leftward()` function to turn left, move forward for 3 seconds, and then go straight.
 - If `moveRight`, calls the `rightward()` function to turn right, move forward for 3 seconds, and then go straight.
 - If `moveStop`, stops the robot using the `stop()` function after a 10-second delay (`delay(10000)`).
 - Prints a single "s" to the Arduino serial monitor (`Serial.println("s")`) for debugging purposes (presumably when moving straight).

3.3 Data Flow in System:

1. The robot encounters an RFID tag and reads its identifier (e.g., T1, T2, etc.).
2. This identifier (`currentTag`) is transmitted wirelessly by the WiFi network hosted on the ESP8266 Module
3. The server also receives the destination tag identifier (`finalTag`) from the HTML user interface.
4. The server-side `findPath` class uses the `currentTag` and `finalTag` to determine the movement commands (e.g., `moveStraight`, `moveLeft`) required to reach the destination.
5. These movement commands are sent over the serial connection to the Arduino board.
6. The Arduino board receives the commands through software serial communication.
7. The `main.cpp (fashioncar)` code on the Arduino decodes the received commands.
8. Based on the decoded commands, the Arduino controls the robot's motors to perform the desired movement (straight, turn left, turn right, stop).

3.4 Approach to System Design:

Environment Modeling and Path Planning (`findPath.cpp`):

- A map of the sewing floor is created, similar to a occupancy grid map used in robotic exploration.
- Key decision points (junctions, stops) are identified on this map, analogous to waypoints in path planning.
- Unique RFID tags are assigned to each decision point, acting as landmarks for the robot's localization.

This approach aligns with the concept of symbolic representation of the environment, where key locations are identified and assigned unique identifiers. The `findPath.cpp` code implements a pre-defined lookup table or decision tree that we made. It takes the robot's current location (read RFID tag - `currentTag`) and the destination tag (`finalTag`) as inputs and retrieves the corresponding movement command (straight, left, right, stop) based on the pre-defined map and path network.

Localization and Navigation (`main2.cpp`, path files):

- The robot employs an RFID sensor (RC522) to detect and read RFID tags placed at decision points.
- The `main.cpp` code (in `fashioncar` directory) continuously monitors the software serial communication for movement commands sent from the server (presumably containing path planning results from `findPath.cpp`).

- These commands (moveStraight, moveLeft, moveRight, moveStop) correspond to the robot's actions at each decision point based on the RFID tag read and the destination received.

The system leverages a hybrid localization approach. The line following sensor ensures the robot stays on the designated path, while the RFID tags provide precise location identification at decision points. This two-tier approach enhances the system's robustness – deviations from the line can be corrected using the line following sensor, while RFID tags provide unambiguous identification of decision points for path planning.

The multiple path files (pathA.cpp, pathB.cpp, etc.) in `path.cpp` inform us of separate pre-defined paths the robot can take based on its starting location (`pathA`, `pathB`, etc.). Each function implements a lookup table or decision tree based on the `currentTag` and `finalTag` to determine the movement commands for a specific path.

Communication and Control:

- A software serial communication link is established between the server and the Arduino board on the robot for transmitting movement commands.
- The Arduino code (main2.cpp) decodes the received commands and controls the robot's motors using specific pin configurations to achieve the desired movement.

This communication architecture allows for centralized path planning on the server and real-time control on the robot's control unit (Arduino).

3.5 Navigation Logic

1. Lookup table (path.cpp):

The path.cpp file encompasses a series of function definitions tailored for implementation within a navigation framework. The efficacy of the system is facilitated by the deployment of Radio Frequency Identification (RFID) tags strategically positioned throughout the operational domain, coupled with a predetermined lookup mechanism aimed at delineating the requisite movement directives essential for traversing from an initial point to a designated destination.

Function Definitions:

The code delineates eight distinct functions, each denoted by the nomenclature "pathX," wherein the symbol 'X' symbolizes a capital letter (e.g., A, B, C, D, R1, R2, R3, R4). These functions are presumed to correspond to diverse routes or endpoints within the AGV's operational milieu.

Function Arguments:

Each function is structured to accept two arguments, both manifesting as pointers to String objects:

- **currentTag:** This pointer references a string encapsulating the identification code of the RFID tag being presently interrogated by the AGV.
- **finalTag:** This pointer indicates a string encapsulating the identification code of the target RFID tag signifying the desired destination.

Function Logic:

Each function operationalizes a succession of conditional constructs, primarily manifested as if-else statements, to juxtapose the values assigned to `currentTag` and `finalTag`. Consequent upon the evaluation outcomes, the function yields a String instance embodying the pertinent movement directive requisite for the AGV's progression:

- **moveStraight:** Denoting linear forward locomotion.
- **moveLeft:** Signifying a ninety-degree counterclockwise rotational adjustment.
- **moveRight:** Indicating a ninety-degree clockwise rotational realignment.
- **moveStop:** Commanding cessation of motion. In instances where none of the conditions within the if-else structures are met (i.e., an unregistered RFID tag is detected), the function returns the string "unregistered Tag."

Code Breakdown Example - pathA Function:

Within the pathA function, characterized by C++ syntax:

- The code interrogates whether the `finalTag` corresponds to "r1Tag" (i.e., the destination RFID tag).
- If affirmative, a subsequent examination is undertaken to ascertain if the `currentTag` matches "T1" (i.e., the originating RFID tag).
- Should this condition hold true, the function outputs "moveStraight" to propel the AGV forward.
- Conversely, if the condition is not met, the function issues a directive to halt movement.
- Analogous procedural sequences are iteratively pursued for alternative `finalTag` values (e.g., `r2Tag`, `r3Tag`, etc.), alongside their corresponding `currentTag` designations.
- In the absence of any matched conditions, the function returns "unregistered Tag," signaling the detection of an invalid RFID tag.

2. Lookup table (path.h):

This header file (`path.h`) defines constants and function prototypes for an AGV (Automated Guided Vehicle) navigation system. Let's break down the code line by line:

1. Preprocessor Directives:

- **#ifndef path_h:** This line checks if a header guard named path_h has already been defined. If not, it proceeds with the following code to prevent multiple inclusions of the same header file.
- **#define path_h:** This defines the header guard path_h. Any subsequent files including path.h will encounter this definition and skip the code within the #ifndef block.

2. Include Directive:

- **#include <Arduino.h>:** This line includes the Arduino library, which provides functionalities for interacting with Arduino boards.

3. Constant Definitions (RFID Tags & Movement Commands):

- A series of #define directives define constants for:
 - RFID tag IDs for starting locations (e.g., r1Tag = "52-43-FC-1B")
 - RFID tag IDs for destination locations (e.g., aTag = "62-15-5B-1B")
 - Movement commands for the AGV (e.g., moveStraight = "straight")

4. Waypoint Definitions (T1, T2, T3, T4):

- More #define directives define constants named T1, T2, T3, and T4. These likely represent specific waypoints within the navigation map. Their exact meaning depends on the specific implementation but could be intermediate locations or starting positions relative to the starting locations (rTags).

5. Class Definition (findPath):

- **class findPath:** This line declares a class named findPath. Classes are a way to group related functions and data in object-oriented programming.

6. Public Member Functions:

- Inside the class definition, several member functions are declared using the public keyword. These functions are accessible from outside the class. Each function has the following structure:
 - **String (return type):** This indicates the function returns a string value.
 - **pathX(String *currentTag, String *finalTag)** (function name and arguments):
 - **pathX:** The function name (e.g., pathA, pathB, etc.). The "X" refers to a specific route or starting location (A, B, C, D).
 - **String *currentTag:** This is a pointer to a string variable that will hold the ID of the RFID tag representing the AGV's current location.
 - **String *finalTag:** This is a pointer to a string variable that will hold the ID of the RFID tag representing the desired destination.

7. Header Guard Closing:

- **#endif:** This line marks the end of the code block protected by the header guard path_h.

3.6 Interoperational Mechanism of both navigation files (path.h and path.cpp):

The operational sequence initiates with the Automated Guided Vehicle (AGV) retrieving spatial coordinates denoting its current position from an RFID tag, encapsulating this information within a designated string variable referred to as 'currentTag.' Simultaneously, the destination coordinates are recorded within a distinct string variable designated as 'finalTag.'

Subsequently, the sequence of programs carried out is as follows:

1. Invocation of an appropriate 'pathX' function, where 'X' signifies a specific destination (e.g., 'pathA' for destination 'A'), is undertaken within the primary program framework. This invocation involves the provisioning of 'currentTag' and 'finalTag' as function arguments, thereby facilitating the contextual assessment of the AGV's present and intended locations.
2. Within the 'pathX' function, a methodical consultation of the predefined lookup table ensues, characterized by a series of if-else constructs integrated into the function's logic. This discerning process plays a pivotal role in ascertaining the optimal trajectory for AGV navigation.
3. Leveraging the acquired current and final coordinates as contextual parameters, the 'pathX' function orchestrates the discernment of the most judicious movement directive. Subsequently, a movement command string is judiciously constructed and returned, furnishing precise instructions to the AGV delineating the requisite directional adjustment (e.g., 'straight,' 'left,' 'right,' or 'stop').
4. Within the overarching program architecture, the transmitted movement command is received from the 'pathX' function. This command, in turn, serves as the authoritative directive guiding the AGV's operational manoeuvre, thereby effectuating the execution of the prescribed action.

CHAPTER 4: AGV Development

4.1 Arduino Connections

4.1.1 Motor Connections

We have assigned specific digital pins to control the direction of each pair of motors connected parallelly. A high (1) signal on the designated pin enables the motor's forward motion, while a low (0) signal activates its backward movement. All pins mentioned below are for Arduino Mega 2560.

Arduino Pins	Function	Module Pins	Description
8 (Digital)	Left Motor Forward	IN1	Controls the forward direction of the left set of motors
9 (Digital)	Left Motor Backward	IN2	Controls the backward direction of the left set of motors
10 (Digital)	Right Motor Forward	IN3	Controls the forward direction of the right set of motors
11 (Digital)	Right Motor Backward	IN4	Controls the backward direction of the right set of motors

4.1.2 5-Channel IR Array Inputs

The five-channel IR sensor array is assigned analog pins as it has separate IR sensors for detecting black line paths, and each channel provides an analog voltage output based on the received infrared intensity in that direction. This allows for more precise line tracking capabilities.

Arduino Pin	Function	Description
A0 (Analog)	Sensor 1 Input	Input for the first IR sensor in array

A1 (Analog)	Sensor 2 Input	Input for the second IR sensor in array
A2 (Analog)	Sensor 3 Input	Input for the third IR sensor in array
A3 (Analog)	Sensor 4 Input	Input for the fourth IR sensor in array
A4 (Analog)	Sensor 5 Input	Input for the fifth IR sensor in array

4.1.3 ESP32 Wi-Fi Module connections

The ESP32 Wi-Fi module uses digital pins for communication with our microcontroller. Digital pins operate at specific voltage levels, typically 3.3V for high and 0V for low on the ESP8266. This allows for clear distinction between data bits (1 or 0) being transmitted. Analog pins, designed for variable voltage readings, wouldn't provide this clear distinction. Since our project also relies on hosting a Wi-Fi network to communicate with the AGV on the ESP module itself, sending and receiving data packets on digital pins ensures accurate transmission and reception of these data packets.

The function of these digital pins specifically in the case of the Wi-Fi module is software serial communication (Rx/Tx). The ESP8266's limited hardware serial ports (often used for debugging) necessitate alternative solutions for multiple communication channels. Software serial libraries leverage digital pins, offering flexibility in defining communication channels when hardware ports are unavailable. For example, in our AGV we needed the hardware port to stay connected to its own source of power while hosting a LAN network and facilitating communication between the AGV and the user interface. This is where software serial proves itself as the solution.

Arduino Pins	Module Pins	Description
RX1/TX1	RX2/TX2	Used for receiving data through software serial communication

4.1.4 HC-SR04 Ultrasonic Sensor connections

The pins assigned to the ultrasonic sensor on the arduino are digital. This is because an ultrasonic sensor works by sending a sound wave and receiving its echo to calculate distance.

In doing so, it does not provide a continuous analog voltage output based on the distance. One pin triggers the sensor to emit a sound wave (often a digital high pulse), and the other receives a digital signal (high or low) indicating the echo reception.

Some ultrasonic sensors do have an analog voltage output for raw data, but for the purposes of our project we deployed a simpler model that utilises digital communication for triggering and echo detection.

Pin	Function	Description
6 (Digital)	Trigger Pin	Initiates a sound wave emission from the ultrasonic sensor
7 (Digital)	Echo Pin	Receives the echo from the ultrasonic sensor to determine distance

4.1.5 SG-90 Servo Motor connections

The SG90 servo motor requires PWM (Pulse Width Modulation) pins on the Arduino Mega 2560 for precise control of its angular position. The PWM pins allow control over the servo motor via pulse width, which helps the servo motor interpret the duty cycle of the PWM signal on its control pin as an instruction. A specific duty cycle range corresponds to specific angular positions of the servo shaft.

Pin	Function	Description
3 (PWM)	Servo Motor Control	Controls the position of the servo motor for obstacle detection

4.1.5 Buck Converter connections

We installed a buck converter to provide 5V to the Arduino mega 2560 from the 12V battery, while also providing 12v DC supply to the motor driver module. This buck converter is basically a step-down converter that converts high input voltage to a lower output voltage.

Buck Converter Wires	Arduino Pins
----------------------	--------------

Brown (positive)	VIN
Black (negative)	GND

4.2 ESP32 Connections

4.2.1 LCD connections (connected through I2C)

The LCD display is connected to the ESP32 using an I2C, through only 4 pins.

ESP32 Pins	I2C Pins
VIN	VCC
GND	GND
SCL	D22
SDA	D21

4.2.2 PN532 NFC Reader Connections

ESP32 Pins	I2C Pins
SDA	SDA
SCL	SCL

4.2 Body Fabrication

4.2.1 Material Selection

Two materials will be used: the inner core structure will be of metal and the outer casing will be of plastic.

Depending on the availability, general knowledge, strength, look and load calculations, the following metals have been chosen for the plates:

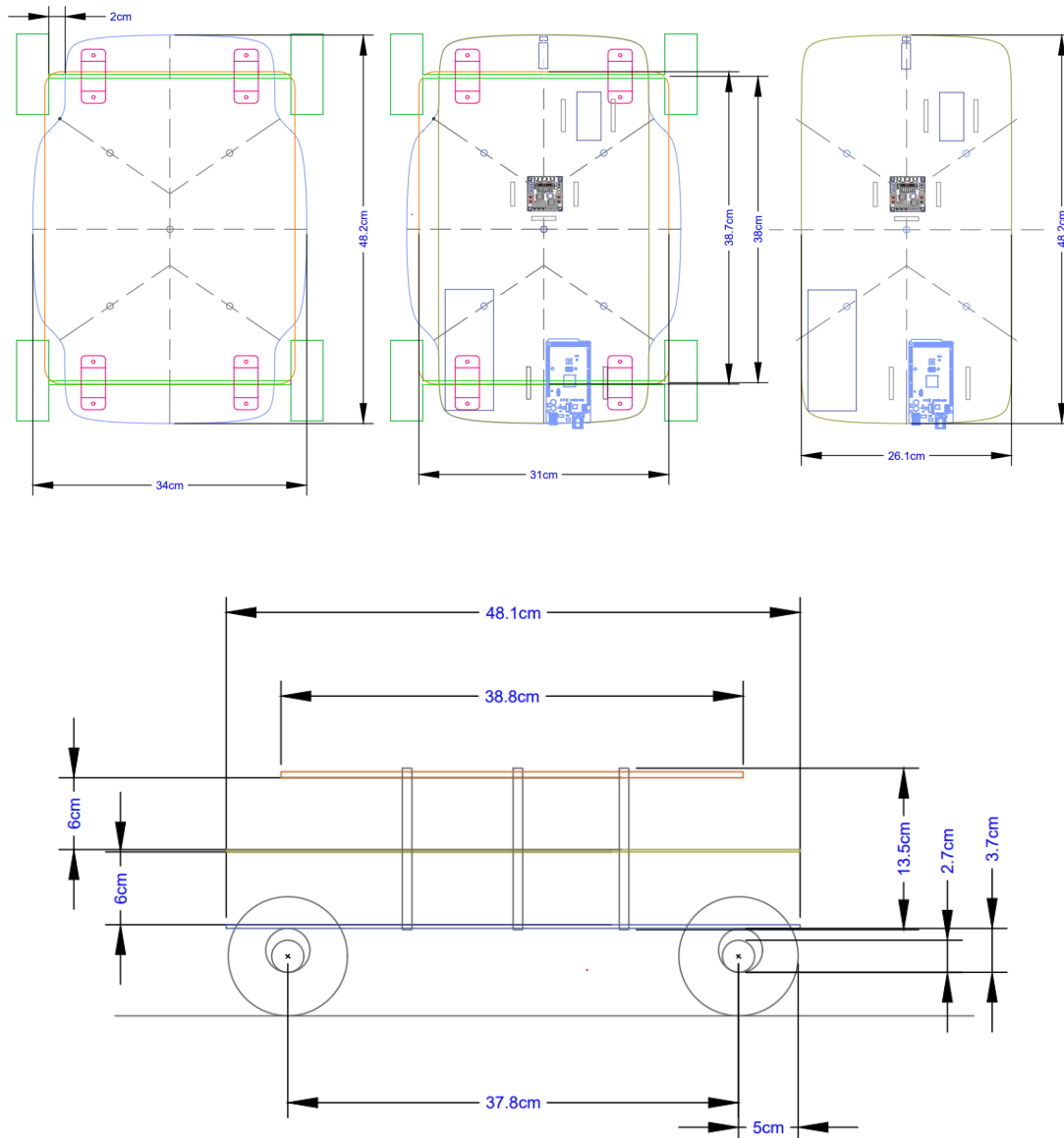
- Wheel plate: 4mm aluminium
- Electronics plate: 1mm stainless steel

- Load bearing platform: two 4mm aluminium plates; 8 mm aluminium

The metal plates have been cut using a CNC laser cutting machine. The nuts and bolts used are M8 for the main support and M2 for all other nut-bolt assemblies.

4.2.2 CAD modelling

The digital dimensional drafting of the base plate, sensor plate and load-bearing plate has been done through the means of AutoCAD.



PART- III

Implementation and Conclusion

CHAPTER 1: Operation and Implementation

1.1 Precautions taken

1. Since we have used mostly DC components, the wires have been kept short and the components close since DC current drops when length of wire increases.
2. While fixing the geared motors to the plate, the alignment of the gearbox and motorbox plays a crucial role as any misalignment can result in the motor shaft breaking due to excess transversal load on the shaft of the motor. In the testing phase, 8 motors which are **2 sets of 4 motors got damaged because of this reason**. We added metal clamp supports to the motor box and gear box individually, which caused the central shaft to buckle and warp under weight. This further caused the coils in the motor box to burn, as it tried with full force to move a warped rod. To shift the weight from the motor shafts, we have added a support wheel at the centre. Another method to eliminate this is to use ball bearings between the motor shafts and the wheel hubs, however our structure did not have enough space for it. Additionally, a wheel axle for the vehicle can be developed which will be discussed further in Chapter 4.
3. All the grounds of all parts should be connected to have a common low potential line which will help in creating the potential difference across the circuit.
4. The RFID module has to be placed away from the aluminium plates in such a way that there is no interference in signals. Otherwise, it would not be able to read the RFID tags. This is because metals have been known historically to detune and reflect RFID signals, which in turn causes poor tag read, unreliable reads, or no read whatsoever.

1.2 Implementation Instructions

Training Manual: Implementation and Troubleshooting

This manual provides instructions for implementing and troubleshooting your AGV. It covers basic setup, operation, and common issues you might encounter.

Section 1: Implementation

1.1 Unboxing and Assembly:

- Carefully unpack the AGV, ensuring all components are present and undamaged.

1.2 Hardware Setup:

- Connect the Arduino Mega, ESP module, motor driver board, and other components according to the wiring diagram provided.
- Double-check all connections to ensure secure and proper orientation.

1.3 Software Installation:

- Download and install the necessary software for the Arduino Mega and ESP module following the manufacturer's instructions.
- Upload the provided AGV control code to the Arduino Mega.
- Configure the ESP module according to the desired WiFi network settings.

1.4 Initial Testing:

- Power on the AGV and verify that the Arduino Mega LED lights up.
- Check the LCD display for any error messages or abnormal behavior.
- Test motor functionality using the control software or manual controls (if available).

Section 2: Troubleshooting

2.1 Common Issues:

Issue: No display on LCD **Possible Causes:**

- Faulty LCD screen
- Loose or damaged LCD wiring
- Software issue with LCD communication

Troubleshooting Steps:

- Visually inspect the LCD for damage.
- Check the LCD connection to the Arduino Mega for secure and proper orientation.
- Refer to the software documentation for troubleshooting LCD communication errors.

Issue: Motor not working **Possible Causes:**

- Faulty motor
- Loose or damaged motor connections
- Issue with motor driver board
- Software issue with motor control

Troubleshooting Steps:

- Visually inspect the motor and connections for damage.
- Check the connection between the motor and the motor driver board.
- Verify if the motor driver board is receiving power and control signals from the Arduino Mega.
- Refer to the software documentation for troubleshooting motor control issues.

Issue: No WiFi network discovered **Possible Causes:**

- Faulty ESP module
- Loose or damaged ESP module connections
- Incorrect WiFi network configuration

Troubleshooting Steps:

- Verify the ESP module is powered on and functioning.
- Check the ESP module connection to the Arduino Mega for secure and proper orientation.
- Ensure the WiFi network name and password are correctly configured on the ESP module.

Issue: No light on Arduino Mega **Possible Causes:**

- Faulty Arduino Mega
- Insufficient power supply
- Issue with buck converter (if used)

Troubleshooting Steps:

- Visually inspect the Arduino Mega for damage.
- Verify that the power supply voltage is within the recommended range for the Arduino Mega.
- Check the buck converter wiring and adjust its settings if necessary (refer to the buck converter documentation).

Additional Tips:

- Always refer to the manufacturer's manuals for specific troubleshooting steps and component specifications.
- Maintain a clean work area and keep the AGV free of debris to prevent malfunctions.
- Regularly inspect the AGV for loose connections or damaged components.
- Update the AGV software periodically if new versions are available.

Section 3: Safety Precautions

- Always power off the AGV before performing any maintenance or troubleshooting.

- Keep hands and loose clothing away from moving parts while the AGV is operational.
- Operate the AGV in a clear and unobstructed environment to avoid collisions.
- Be aware of your surroundings while operating the AGV manually.

CHAPTER 2: Conclusion

2.1 Limitations and Scalability

2.1.1 Limitations

- The locations where the RFID tags are placed, have to be kept free of obstacles so that the RFID module is able to detect tags and the AGV movement is not hampered repeatedly.
- Better support structure is required if the dimensions and the payload capacity are to be increased dramatically. In this prototype, there's one caster wheel at the centre that supports the AGV payload and takes away the weight from the motors and their shafts. For a higher load, this structure needs to be improved with support rods or wheel axle support.
- Currently, our core logic cannot show which component of the AGV is malfunctioning. In case of a breakdown, the functionality that is failing has to be identified and the components aiding that functionality have to be replaced.
- We could not develop a maintenance schedule since it was out of the timeframe of the project, however the motors are the only parts that can have a maintenance schedule. All of the other electronic components have to be replaced if they start malfunctioning. It's advisable to have a spare parts inventory for the same.

2.1.2 Scalability

- The RFID navigation system is not fixed to location, so it can be moved anywhere. Eg., if this system is to be deployed in the fabric warehouse, only the RFID tags have to be placed at the loading, unloading and intersection points, rest everything remains the same.
- AGV is easily scalable to adapt to increased loads and higher RPM according to each factory or location, just the motors have to be changed.
- If the platform size or the dimension has to increase but the payload is not increasing dramatically (as mentioned above in 2.1.1) then a simple scaling up of the structure is enough to meet the demand.

2.2 Reliability

Justification for reliability

As per study done in Part I: Section 3.10, the testing and calculation of MTBF and MTTR are paramount for determining the reliability of the AGV and designing a maintenance schedule. This procedure requires at least 3 months of active operation and testing which exceeds the timeframe of the project. However, a justification for higher reliability can be given due to the following facts:

- The components used in the development of the AGV are standard components and are easily available.
- The cost of procuring the components is quite low since they can be found in any electronics market and they are fairly cheap. (Please refer the BOM for the same)
- There is no complexity in maintenance of the AGV since motors are the only parts that require maintenance. All other components have to be simply replaced if they break down or malfunction.
- The Implementation Instructions are quite detailed and simple for any maintenance manager to understand with basic training, so no special personnel or process is required for maintenance management.

2.3 Feasibility

Current Scenario:

- Two helpers transport packages (150 pieces, 40 kg) from the lift to sewing lines (80-120 ft away).
- Loading happens 4 times/day/line (64 times/day total).
- Helpers travel throughout the day with a total load of 60 kg per trip.
- Loading time per trip: 3 minutes.
- Helper cost (unskilled labor): Rs. 30,000/month (10-hour shift)

Benefits of AGV Implementation:

- Eliminates the work of 2 loaders (potential labor cost savings).
- Improves efficiency by potentially reducing travel and loading time.

Costs of AGV Implementation:

- Initial investment cost of the AGV (Rs. 15,000)
- Maintenance costs of the AGV (Rs. 5,000/year) (assuming monthly cost = Rs. 5000/year / 12 months = Rs. 416.67)

Feasibility Analysis:

Labor Cost Savings:

- Monthly salary per helper: Rs. 15,000
- Total cost for 2 helpers: Rs. 30,000/ month

AGV Performance:

The payload limitation (20 kg) of the AGV compared to the package weight (40 kg) needs to be addressed. According to whatever is feasible, either the payload capacity could be increased

or the number of AGVs deployed. So how many total loadings happen in a day depends on this.

3. Travel Time & Loading Time Analysis:

- Round trip distance (considering average distance) = $(80 \text{ ft} + 100 \text{ ft} + 120 \text{ ft}) / 3 * 2 \approx 200 \text{ ft}$ (one-way) = 61 meters
- Speed: Currently, the speed of the AGV is 40 RPM with load which is roughly 0.2 m/s
- Travel Time per trip = Distance / Speed = 61 meters / 0.2 m/s = 305 seconds

We cannot estimate the loading time since the loading is manual for the prototype developed.

The total time saved can only be estimated after carefully calculating the total time for a delivery and comparing it to the time taken by the loaders to do the same. This will also give an idea of how much cost is saved in total along with the increase in production. The ROI period can also be calculated after this.

CHAPTER 3: Future Scope

3.1 Modular wheel design

A modular wheel design for your AGV means the wheels are designed to be easily swapped or replaced with minimal modification to the main body. This offers several benefits during the development and operation of your line follower AGV.

Benefits of a Modular Wheel Design:

- **Testing Different Wheel Sizes:** As discussed earlier, finding the optimal wheel size for maneuverability and ground clearance might require some experimentation. A modular design allows you to quickly swap between different wheel sizes without needing to rebuild the entire wheel attachment system.
- **Easy Wheel Replacement:** During operation, wheels are prone to wear and tear. A modular design allows for quick and easy replacement of individual wheels without disassembling the entire AGV drivetrain.
- **Future Modifications:** Your AGV's needs might evolve in the future. A modular design allows you to potentially swap the entire wheel module (including motor) for a different type if needed, increasing the adaptability of your AGV.

There are several ways to achieve a modular wheel design, here are a couple of common approaches:

1. **Swappable Wheel Hubs:** Design the wheel hub (the part that connects the wheel to the axle) to be detachable from the main body of the AGV. This allows you to have different wheels with interchangeable hubs that can be easily attached and detached.
2. **Modular Wheel Pods:** Design a complete module that includes the motor, gearbox (if used), and wheel assembly. This entire pod can be bolted or secured to the main body of the AGV. This approach offers a more complete and pre-assembled unit for easy swapping.

Considerations for Modular Design:

- **Strength and Stability:** The connection between the modular element (wheel hub or pod) and the main body should be strong and stable to handle the weight and potential vibrations of the AGV.

- **Alignment:** Ensure the modular element can be precisely aligned with the drivetrain for proper operation.
- **Standardization:** If you plan to use multiple wheel sizes, consider designing the modular interface to be standardised across all the different wheel options. This simplifies swapping and ensures compatibility.

By implementing a modular wheel design, you gain flexibility and efficiency during the development and operation of your line follower AGV.

Approach 1: Swappable Wheel Hubs

1. **Design the Wheel Hub:**
 - The wheel hub is the part that connects the wheel to the axle of the AGV.
 - Design the hub to have a central bearing seat for the axle and an outer attachment mechanism for the wheel itself.
2. **Outer Attachment Mechanism:** There are several options here:
 - **Press-fit:** The hub can have a slightly tapered fitting surface where the wheel is press-fit for a secure connection. This requires ensuring proper tolerances for a tight fit.
 - **Bolts/Screws:** The hub can have threaded holes and the wheel can have corresponding holes for bolting them together. This allows for easier disassembly and reassembly.
 - **Locking Mechanism:** A more complex option would be to design a locking mechanism that allows for quick attachment and detachment of the wheel. This might involve levers or clamps.
3. **Wheel Design:**
 - Design the wheels to have an inner surface that securely connects to the chosen attachment mechanism on the wheel hub.
 - Ensure the wheels have a central hole that fits the axle and bearings.

Approach 2: Modular Wheel Pods

1. **Design the Wheel Pod:**
 - The wheel pod is a complete module that includes the motor, gearbox (if used), and the wheel assembly.
 - Design a housing for the motor and integrate the axle with the wheel.
2. **Mounting Interface:**
 - Design a mounting plate or interface on the main body of the AGV where the entire wheel pod can be attached.
 - This interface can involve:
 - **Slide-in and Lock:** The pod can slide into a designated slot on the AGV body and then be secured with a locking mechanism.

- **Bolted Connection:** The pod can be bolted directly onto the main body using threaded holes on both components.
3. **Electrical Connections:**
- Design a connector system for the motor wires to connect to the main AGV power supply. This connection should be part of the mounting interface to ensure proper connection when the pod is attached.

General Considerations for Both Approaches:

- **Strength and Stability:** The connection between the modular element (wheel hub or pod) and the main body should be strong and stable to handle the weight and potential vibrations of the AGV.
- **Alignment:** Ensure the modular element can be precisely aligned with the drivetrain for proper operation of the wheels.
- **Standardization:** If you plan to use multiple wheel sizes, consider designing the modular interface to be standardized across all the different wheel options. This simplifies swapping and ensures compatibility.

Additional Tips:

- Use lightweight materials for the wheel pods or hubs to minimize the overall weight of the AGV.
- Design the modular components for easy access to bearings or motor components for future maintenance.
- Consider incorporating visual cues or alignment markers to aid in proper positioning of the modular wheels during assembly.

By implementing a well-designed modular wheel system, you gain flexibility in testing different wheel configurations and streamlining future maintenance of your line follower AGV.

3.2 Battery

Regenerative Braking: If your AGV incorporates regenerative braking (returning energy to the battery during stopping), this can slightly improve overall efficiency and extend operating time.

Also, BMS board can be added with a rechargeable battery system.

A Battery Management System (BMS) acts as the brain and bodyguard of your rechargeable battery pack. It plays a crucial role in ensuring optimal performance, safety, and longevity of your battery. Here's a breakdown of how a BMS typically functions:

Core Functions:

- **Cell Monitoring:** The BMS constantly monitors the voltage, current, and temperature of each individual cell within the battery pack. This allows for early detection of any imbalances or abnormalities in a single cell.
- **Cell Balancing:** If the BMS detects a voltage difference between cells, it can take corrective actions to even out the charge distribution. This process, called cell balancing, helps prevent overcharging or undercharging of individual cells, extending the overall battery life.
- **Protection Circuits:** The BMS incorporates various protection circuits to safeguard the battery from harmful conditions. These include:
 - **Over-charge Protection:** Prevents the battery voltage from exceeding a safe limit, which can damage the cells.
 - **Over-discharge Protection:** Disconnects the load from the battery when the voltage drops too low, preventing damage from deep discharge.
 - **Over-current Protection:** Limits the current flow into or out of the battery to prevent overheating or damage.
 - **Thermal Protection:** Monitors battery temperature and takes action (like reducing charging current) if it exceeds safe limits.
- **State of Charge (SOC) Estimation:** The BMS tracks the battery's charge and discharge cycles to estimate the remaining capacity (SOC). This is similar to a fuel gauge for your AGV, providing valuable information about how much power is left before needing a recharge.
- **State of Health (SOH) Monitoring:** Over time, battery performance can degrade. The BMS can monitor changes in cell parameters to estimate the overall health (SOH) of the battery, indicating its remaining useful life.
- **Communication Interface:** Some BMS offer communication interfaces that allow them to transmit data (voltages, currents, temperatures, SOC, SOH) to an external device like a monitor or control system. This enables real-time monitoring and potential control adjustments for the battery.

Benefits of a BMS:

- **Extends Battery Life:** By preventing overcharging, undercharging, and other harmful conditions, a BMS helps maintain battery health and prolong its lifespan.
- **Improves Safety:** Protection circuits safeguard the battery from potential damage or fire hazards.
- **Optimizes Performance:** Cell balancing ensures all cells contribute equally, leading to efficient battery utilization.

- **Provides Valuable Data:** The BMS offers insights into battery health and remaining capacity, facilitating informed decisions about charging and maintenance.

In conclusion, a Battery Management System plays a vital role in ensuring the safe, reliable, and efficient operation of your AGV's battery pack. It's an essential component for maximizing battery life and performance.

3.3 Virtual map feed for defining route

1. Simplified map representation:

- **Concept:** Instead of storing a full detailed map, define the layout using basic geometric shapes and waypoints.
- **Implementation:**
 - Represent your sewing floor as a series of lines, points, and angles. For example, define loading stations and sewing lines as points, and paths as connecting lines.
 - Store these map elements within the Arduino memory as simple coordinates and angles.
 - Calculate movement commands based on your current position (estimated from encoders) and the target point coordinates relative to your position. This calculation involves basic trigonometry and geometry.

Advantages:

- Easier to implement and store on Arduino memory compared to a full map.
- Requires less processing power for navigation.

Disadvantages:

- Less accurate, especially for complex layouts with curves or obstacles.
- Limited to predefined waypoints and paths, making dynamic navigation challenging.

2. Offload processing to an external device:

- **Concept:** Use a more powerful device like a Raspberry Pi or a computer to handle complex map processing and send simplified instructions to the Arduino.
- **Implementation:**
 - Create a detailed map of your sewing floor on the external device using software tools.
 - Implement pathfinding algorithms on the external device that calculate the optimal route based on the map and the AGV's current position (received from Arduino).

- The external device sends simplified instructions to the Arduino, such as "move forward X distance, turn left Y degrees".

Advantages:

- More accurate navigation possible with a detailed map and pathfinding algorithms.
- Flexible for changes in layout by updating the map on the external device.

Disadvantages:

- Requires additional hardware and software development effort.
- Introduces communication delays and potential complexity between the devices.

The method for feeding a map into a controller depends on the complexity of the map and the capabilities of the chosen controller.

Arduino:

- **Simple Maps:**
 - **Store Coordinates:** If your map is fairly simple and consists of basic shapes and waypoints, you can directly store the coordinates of these elements in Arduino's memory as arrays or variables. This approach is suitable for small layouts with defined paths.
 - **Pre-processed Instructions:** Alternatively, pre-process your map beforehand on a computer and convert it into a sequence of movement instructions (e.g., move forward X cm, turn left Y degrees). Store these instructions on the Arduino and execute them sequentially.
- **Complex Maps:**
 - **External Processing:** For complex maps with curves, obstacles, or dynamic navigation needs, the limited processing power of an Arduino becomes a challenge. Consider offloading map processing to an external device like a Raspberry Pi.

Raspberry Pi:

- **Image-based Maps:**
 - **Store image:** You can store the map directly as an image file on the Raspberry Pi's SD card. Use image processing libraries to analyze the image and extract relevant information for navigation (e.g., landmarks, paths).
- **Vector-based Maps:**
 - **Store data:** Represent the map using a vector format that stores elements like lines, points, and polygons. This approach offers more flexibility for

manipulation and pathfinding algorithms. Libraries like OpenMapTiles or Mapbox Vector Tile can be used.

- **Pre-processed Instructions:**

- **Store instructions:** Similar to Arduino, you can pre-process the map on a computer and store simplified instructions on the Raspberry Pi. This requires less processing power on the Pi itself.

Feeding Methods:

- **SD Card:** Both Arduino and Raspberry Pi can access data stored on an SD card. This is a common method for storing large maps or pre-processed instructions.
- **Serial Communication:** You can send map data from a computer to the controller via serial communication (e.g., USB cable). This allows for dynamic updates or more complex data transfer.
- **WiFi Module (Raspberry Pi):** For wireless communication and map updates, consider using a WiFi module and a server to send map data to the Raspberry Pi.

3.4 Dijkstra's Algorithm

Dijkstra's algorithm is a well-known algorithm for finding the shortest paths between nodes in a weighted graph, which perfectly suits the scenario of using RFID tags and line following for AGV navigation.

Understanding Dijkstra's Algorithm:

1. **Weighted Graph:** It works on a weighted graph where nodes represent stations and edges represent connections between them. Each edge has a weight, typically representing the distance or travel time between stations.
2. **Unvisited and Visited Nodes:** The algorithm starts by considering all nodes as "unvisited" and initializes a "distance" variable for each node. Initially, the distance to all nodes except the starting station (source) is set to infinity.
3. **Iterative Process:** The algorithm iterates through the following steps until all nodes are visited:
 - It identifies the unvisited node with the **smallest** current distance from the source. This node becomes the "current node."
 - For all **unvisited neighbors** of the current node, the algorithm calculates the tentative distance to reach them. This involves the current distance to the current node plus the weight of the edge connecting them.

- If the tentative distance is **smaller** than the neighbor's current distance, the algorithm updates the neighbor's distance with the new lower value. This essentially explores shorter paths through the current node.
 - The current node is then marked as "visited" as it's been fully explored.
4. **Shortest Path:** After all nodes are visited, the "distance" variable of each node represents the shortest distance from the source node to that particular node.

Line Following Integration: Once you have the shortest path (a list of nodes), develop logic to guide the AGV along the path using line sensors. This might involve breaking down the path into segments and using line following for each segment, adjusting for turns and intersections based on the path sequence.

Dijkstra's algorithm typically calculates the shortest path only once per request, not continuously throughout the AGV's operation. The breakdown of its operation mode is as follows:

Single Calculation per Request:

1. **Demand Generation:** When a demand arises from a specific loading station (source), you provide this information to the program.
2. **Dijkstra's Algorithm:** The algorithm utilizes the pre-defined graph data structure representing your entire floor layout, including all stations and connections. It calculates the shortest path specifically from the **source station** (loading station with the current demand) to the **destination station** (defined by you or pre-programmed).
3. **Path Execution:** Once the shortest path is identified (a sequence of nodes), your program uses this path to guide the AGV via line following, intersection handling, and reaching the destination.

Alternative Approach (Limited Use Case):

In rare scenarios, if your floor layout is **extremely simple** and **static** with only one loading station and a single destination, a pre-calculated path from the loading station to the destination might be sufficient. However, this approach lacks flexibility and won't work for multiple loading stations or changing destinations.

Recommendation:

For most practical AGV navigation scenarios, the **single calculation per request** approach using Dijkstra's algorithm is the recommended approach. This balances efficiency with flexibility, allowing the AGV to navigate efficiently based on dynamic demands while avoiding unnecessary continuous calculations.

Additional Considerations:

- **Dynamic updates:** If your floor layout changes or new stations are added, you'll need to update the graph data structure and potentially re-calculate the shortest path accordingly.
- **Error handling:** Implement mechanisms in your program to handle situations where unexpected events occur during navigation, such as missed connections or obstacles, and potentially recalculate the path if necessary.

3.4.1 Error Handling for Dijkstra's Algorithm and AGV Navigation

1. Error Sources:

While implementing Dijkstra's algorithm and using it for AGV navigation, you might encounter various errors. Here are some common examples:

- **Missing connections:** The AGV might miss an expected connection due to sensor limitations or environmental factors.
- **Unexpected obstacles:** Obstacles not present in the map data could block the planned path.
- **Incorrect sensor readings:** Faulty sensors might provide inaccurate information about the environment.
- **Communication issues:** Communication errors between Raspberry Pi and the AGV controller could disrupt navigation instructions.
- **Software bugs:** Bugs in your code could lead to unexpected behavior or incorrect path calculations.

2. Error Handling Mechanisms:

Here are some mechanisms you can implement to handle these errors:

- **Sensor data validation:** Implement checks on sensor readings to identify outliers or inconsistencies that might indicate issues.
- **Real-time path replanning:** If the AGV encounters an obstacle or missed connection, use Dijkstra's algorithm again with the updated real-time location as the source to recalculate the shortest path to the destination.
- **LiDAR integration (optional):** Consider using LiDAR sensors for obstacle detection and incorporate real-time obstacle information into the graph data structure for dynamic path adjustments.
- **Communication error handling:** Implement mechanisms to detect and recover from communication failures between the Raspberry Pi and the AGV controller. This might involve re-sending critical data or using alternative communication channels.
- **Robust coding practices:** Employ good coding practices like error checking, exception handling, and clear documentation to minimize software bugs.

3. Feeding the Graph Data Structure:

Here are several ways to feed the graph data structure for Dijkstra's algorithm to start its calculation:

- **Pre-defined data:** If your floor layout is static, define the graph data structure directly in your code, including nodes (stations) and edges (connections) with their weights (distances).
- **External file:** Store the graph data in a structured format like JSON or CSV and load it into your program when needed. This allows for easier updates without modifying the code itself.
- **Database:** For complex layouts or frequent updates, consider storing the graph data in a database like SQLite or MySQL. This enables centralized management and access from your program.

References

- [1] Ravazzi, P., Villa, A., Economic Aspects of Automation. (2009). Springer Handbook of Automatin, Part A, PP. 93-116
- [2] Chadwick-Jones, J. K., Automation behaviour: A social psychological, (1969), study'Wiley-Interscience, (London and New York), (ISBN0471143006) xi, PP. 168.
- [3] Kumar, S., & Rahman, N. (2016). *Made in Bangladesh: Challenges and opportunities in the RMG industry*. Springer International Publishing.
- [4] McKinsey & Company. (2017, January 10). *Fashion's digital revolution: Capturing the future of retail through omnichannel strategies*. McKinsey & Company.
<https://www.mckinsey.com/~media/McKinsey/Industries/Retail/Our%20Insights/Fashions%20digital%20transformation%20Now%20or%20never/Fashions-digital-transformation-now-or-never-VF.pdf>
- [5] Søndergaard, M., Petersen, E., & Hansen, E. T. (2018). *Mass customization: The next frontier in business model innovation*. Business Horizons, 61(5), 673-681.
- [6] Tucker, R., & Bennett, H. (2016). *The future of fashion: Rethinking production and consumption*. Earthscan.
- [7] Hrishna, S., & Raj, R. M. (2019). *Impact of automation in textile and apparel industry*. Journal of Textile Engineering & Apparel Science, 6(2), 113-118.
- [8] Basnet, C., Mize, J.H. Scheduling and control of manufacturing systems: a critical review, (1994), International Journal of Computer Integrated Manufacturing, Vol.7, PP. 340±355.
- [9] Rachamadugu, R., Stecke, K.E., Classification and review of FMS scheduling procedures, (1994), Production Planning and Control, Vol.5, PP. 2±20.
- [10] Design and Development of an Automated Guided Vehicle for Educational Purposes Khosro Bijanrostami Submitted to the Institute of Graduate Studies and Research in partial fulfillment of the requirements for the Degree of Master of Science in Mechanical Engineering Eastern Mediterranean University September 2011 Gazimağusa, North Cyprus
- [11] Li Song et al 2020 J. Phys.: Conf. Ser. 1575 012095
- [12]<https://sensemore.io/what-is-the-difference-between-maintenance-and-reliability/>
- [13]<https://www.milliken.com/en-us/businesses/performance-solutions-by-milliken/blogs/maintenance-and-reliability>

- [14] <https://asq.org/>
- [15] Blanchard, B. S., Verma, D., DeVlieg, K. L., & Lubas, P. (2015). *System Engineering Analysis, Design, and Development* (Eighth Edition). John Wiley & Sons, Inc. (Chapter 12: Reliability)
- [16] <https://smrp.org/>
- [17] Campbell, J. D. (2013). *Uptime: Strategies for Excellence in Maintenance Management* (Third Edition). Productivity Press. (Chapter 4: Planning and Scheduling Maintenance)
- [18] Operations Management" by S.M. Sodhi and R.K. Abhishek (2021)
- [19] Logistics Management" by A. Rudolph and J. Kruse (2019)
- [20] Yusuf, Y., Kaynak, H., & Sari, I. C. (2004). Material handling systems and automation in apparel manufacturing. *Fibers and Textiles in Eastern Europe*, 12(1(43)), 63-68
- [21] Şahin, I. (2007). An application of bottleneck detection and elimination methods in a garment manufacturing company. *The International Journal of Clothing Science and Technology*, 19(2), 119-128
- [22] Oncu, S., & Kara, S. (2013). A review and a new classification of flexible manufacturing systems. *Computers & Industrial Engineering*, 64(1), 199-222
- [23] A Materials Handling System for Fashion Retailing. (n.d.). Retrieved from <https://www.everythingsupplychain.com/automated-storage-and-retrieval-system-asrs/>
- [24] (Sodhi, M. S., & Abhishek, R. K., 2021)
- [25] Sodhi, M. S., & Abhishek, R. K. (2021). *Operations Management*. New Age International
- [26] Boysen, N., & Fliedner, M. (2019). *Warehouse Management*. Springer Gabler.
- [27] U.S. Department of Labor – Occupational Safety and Health Administration (OSHA). (2023, January 11). Material Handling and Storage. OSHA website: <https://www.osha.gov/sites/default/files/publications/osha2236.pdf>
- [28] Rudolph, A., & Kruse, J. (2019). *Logistics Management*. Springer Gabler.
- [29] Boysen, N., & Fliedner, M. (2019). *Warehouse Management*. Springer Gabler
- [30] <https://www.everythingsupplychain.com/automated-storage-and-retrieval-system-asrs/>

- [31] Şahin, I. (2007). An application of bottleneck detection and elimination methods in a garment manufacturing company. *The International Journal of Clothing Science and Technology*, 19(2), 119-128.
- [32]
[\(https://www.everythingsupplychain.com/automated-storage-and-retrieval-system-asrs/\)](https://www.everythingsupplychain.com/automated-storage-and-retrieval-system-asrs/)
- [33] a study by Oncu, S., & Kara, S. (2013). A review and a new classification of flexible manufacturing systems. *Computers & Industrial Engineering*, 64(1), 199-222
- [34] (Jayatilake & Withanaarachchi, 2019)
- [35] (Jayatilake & Withanaarachchi, 2019)
- [36] (Ahmad, et al., 2020)
- [37] (Saravanan, 2009)
- [38] Kim & Culler, 2014).
- [39] (Singh & Prasad, 2009)
- [40] Qian, F., Yu, H., & Zhou, Y. (2019). Smart manufacturing for apparel industry: opportunities and challenges. *Journal of Fashion Marketing and Management*, 23(4), 430-442
- [41] Kara, S., Kaehne, A., & Sutherland, M. (2019). Facilitating agility in apparel supply chains: A maturity model for production planning and control. *International Journal of Production Research*, 57(1), 189-210
- [42] Automated Guided Vehicles (AGVs) for the Automotive Industry [White Paper]. Swisslog.
<https://www.swisslog.com/en-us/products-systems-solutions/transport/agv-automated-guided-vehicles>
- [43] How AGVs Improve Warehouse Efficiency: A Guide to Benefits and Applications [White Paper]. Geek+.
<https://www.bevindustry.com/articles/84877-warehouses-automate-with-agvs>
- [44] Boysen, N., & Bernhardt, G. (2020). *Fashion 4.0: An introduction to digital fashion*. Springer Nature.
- [45] Wu, D., Duerstock, D., Wambsganss, T., & Greif, G. (2016). Emerging logistics technologies and their impact on supply chain performance: A comprehensive review. *European Journal of Operational Research*, 253(3), 673-695. <https://doi.org/10.1016/j.ejor.2016.02.013>

- [46] Christopher, M. (2016). Logistics & supply chain management. Pearson
<https://www.rokin.tech/post/3-reasons-why-you-should-consider-implementing-agvs-on-your-factory-floor>
- [47] International Journal of Production Research - Special Issue on Smart Manufacturing in the Fashion Industry: <https://www.sciencedirect.com/journal/journal-of-manufacturing-systems/special-issues>
- [48] [Case Study] Bosch Rexroth Cuts Lead Times 20% with Automated Guided Vehicles. <https://www.boschrexroth.com/en/pl/products/product-groups/mobile-robotics/>
- [49] AGVs in BMW Manufacturing Plant Source: Industry publication
- [50] Oncu, S., & Kara, S. (2013). A review and a new classification of flexible manufacturing systems. Computers & Industrial Engineering, 64(1), 199-222.
- [51] Deloitte. (2019). The Future of Manufacturing: Automation and Jobs. <https://www2.deloitte.com/content/dam/Deloitte/us/Documents/energy-resources/us-2023-outlook-manufacturing.pdf>
- [52] Boysen, N., & Fliedner, A. (2000).** Impact of automated guided vehicles on production logistics**. International Journal of Production Research, 38(18), 4421-4432.