TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechanical Engineering

# Creating an Algorithm for AGV Control

## Master Thesis

| | |
|---|---|
| *Study programme:* | N2301 Mechanical Engineering |
| *Study branch:* | Manufacturing Systems and Processes |
| *Author:* | **Hari Shankar Venkataraman** |
| *Thesis Supervisors:* | Ing. Petr Keller, Ph.D.<br>Department of Manufacturing Systems and Automation |

Liberec 2022

TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechanical Engineering

**Master Thesis Assignment Form**

# Creating an Algorithm for AGV Control

*Name and surname:* **Hari Shankar Venkataraman**
*Identification number:* S19000477
*Study programme:* N2301 Mechanical Engineering
*Branch:* Manufacturing Systems and Processes
*Assigning department:* Department of Manufacturing Systems and Automation
*Academic year:* **2020/2021**

**Rules for Elaboration:**

The aim of the work is to create and test an algorithm for automatic control of inter-operational transport in the designed model of the manufacturing system based on the principles of Industry 4.0. Recommended methods of elaboration: 1. Become familiar with the existing components of manufacturing system. 2. Make research of existing previous AGV solutions, especially line following control 3. Design the vehicle control algorithm based on the system requirements, as are wireless communication with other peripherals, automatic transport of material to the place, automatic charging of batteries, but also reverse movement from serviced stations, etc. 4. Test the algorithm practically on a vehicle model controlled for example by an Arduino development board, etc. 5. Evaluate your work

*Scope of Graphic Work:* according to need
*Scope of Report:* approx. 60 pages
*Thesis Form:* printed/electronic
*Thesis Language:* English

**List of Specialised Literature:**

[1] BEQUETTE, B. *Process control: modeling, design, and simulation.* Upper Saddle River, N.J.: Prentice Hall PTR, 2003. ISBN 0133536408. [2] Arduino Learning: Getting Started with Arduino. In: Arduino [online]. 2014 [cit. 2020-09-15]. available from: http://arduino.cc/en/Guide [3] ULLRICH, G. *Automated Guided Vehicle Systems.* New York: Springer, 2016. ISBN 9783662448137. [4] BASL, J. Pilot Study of Readiness of Czech Companies to Implement the Principles of Industry 4.0. Management and Production Engineering Review [online]. 2017, **8**(2), 3-8 [cit. 2020-09-15]. ISSN 2082-1344. Available from: doi:10.1515/mper-2017-0012

*Thesis Supervisors:* Ing. Petr Keller, Ph.D.
Department of Manufacturing Systems and Automation

*Date of Thesis Assignment:* November 19, 2020
*Date of Thesis Submission:* May 19, 2022

L.S.

prof. Dr. Ing. Petr Lenfeld
Dean

Ing. Petr Zelený, Ph.D.
Head of Department

Liberec March 5, 2020

# Declaration

I hereby certify, I, myself, have written my master thesis as an original and primary work using the literature listed below and consulting it with my thesis supervisor and my thesis counsellor.

I acknowledge that my master thesis is fully governed by Act No. 121/2000 Coll., the Copyright Act, in particular Article 60 – School Work.

I acknowledge that the Technical University of Liberec does not infringe my copyrights by using my master thesis for internal purposes of the Technical University of Liberec.

I am aware of my obligation to inform the Technical University of Liberec on having used or granted license to use the results of my master thesis; in such a case the Technical University of Liberec may require reimbursement of the costs incurred for creating the result up to their actual amount.

At the same time, I honestly declare that the text of the printed version of my master thesis is identical with the text of the electronic version uploaded into the IS/STAG.

I acknowledge that the Technical University of Liberec will make my master thesis public in accordance with paragraph 47b of Act No. 111/1998 Coll., on Higher Education Institutions and on Amendment to Other Acts (the Higher Education Act), as amended.

I am aware of the consequences which may under the Higher Education Act result from a breach of this declaration.

May 11, 2022 Hari Shankar Venkataraman

## ACKNOWLEDGEMENT

I recognize the support of my university for providing me with this fantastic opportunity to further my engineering skills, and I want to thank everyone who helped me finish this diploma thesis with great pleasure and gratitude. It is really appreciated that your excellent guidance for my research was vital in its success.

I owes a great debt of gratitude to **Ing. Petr Keller, Ph.D**. He persuaded and motivated me to be professional and do the right thing even when the going got tough, and he pushed me to my limits. The goal of this thesis was achieved thanks to his constant coaching.

Also, I'd like to thank **Ing. Petr Zeleny, Ph.D.**, Head of the Department, Department of Manufacturing Systems and Automation, for his unconditional support throughout the study.

Also I like to express my deepest gratitude to The Department of Manufacturing Systems and Process at the Technical university of Liberec for supporting me to complete my studies successfully and helps me to gain knowledge throughout my studies.

My deepest gratitude also goes out to my family and friends for their unwavering support and love during all of my good and bad times.

This work was supported by the Student Grant Competition of the Technical University of Liberec under the project Optimization of manufacturing systems, 3D technologies and automation No. SGS-2019-5011.

## ABSTRAKT

Tato diplomová práce se zabývá automaticky vedeným vozíkem sledujícím čáru, používané v průmyslu pro manipulaci s materiálem. Jejím hlavním cílem je vytvoření řídicího algoritmu pro provoz tohoto vozíku po definované dráze pro následné řízení pomocí Arduina. Řízení zabezpečuje i couvání vozíku ze stanice zpět na hlavní dráhu. Kromě toho je v systému řízení vozíku implementován bezdrátový centralizovaný komunikační systém s využitím modulů IoT. Celý systém byl úspěšně otestován. Navržený algoritmus přispěje k snadnější implementaci konceptu Průmysl 4.0, který je v současné době na vzestupu.

Klíčová slova: AGV, Průmysl 4.0, Arduino nano, Arduino Mega2560, infračervený senzor, internet věcí (IoT)

## ABSTRACT

This thesis work depends on the line follower AGV which is used in the industries for the material handling process. In this main aim is to develop a control algorithm to operate this AGV along a defined path using the Arduino programming. The control also ensures the reverse movement of the vehicle from the station back to the main path. In addition, with this the wireless communication system is implemented as a centralized communication system using the IoT modules and has been tested. The proposed algorithm will contribute to the easier implementation of the Industry 4.0 concept which is currently the booming trends in industries.

Keywords: AGV, Industry 4.0, Arduino nano, Arduino Mega2560, IR sensor, IoT

**Table of Contents**

**Table of figures**

**List of Tables**

## 1. Introduction

The diploma thesis " Creating an Algorithm for AGV control " in which the AGV controlling algorithm needs to be created and it should be tested. Nowadays, the automation is being implemented in all the fields of industry. This includes even the carrying of materials or packing of materials uses automation. This AGV is a vehicle which is used to carry the materials from one place to another. It helps to do the process or workflow efficient ways. This AGV is being used in many industries as per their needs in the industries. This AGV can be used for logistics, material handling, etc.

In this thesis the AGV is created as the line following robot which follows the black line and moves in the path. This line follower consists of various components and these components are explained and the components that are used are explained in this. The working of the line following robot is studied and implemented in this.

Before the creation of this protype model the current trends and the line following robots that are used in the industries are learned and mentioned in the below chapters.

After this the steps that are followed to create the protype of the AGV is explained and the Arduino is used for this controlling methods and these codes are also explained about the working of the codes.

In this the Industrial 4.0 concepts are learned and explained in this and the concept that is being used in this is also discussed and the future implementation are also discussed in the following chapters.

## 2. Aim of the Thesis

The aim of the thesis is to create an algorithm for automatic control which is used for Inter-operational transport in the manufacturing systems based on the requirements in the system based on the industry 4.0. This AGV is to be designed as the line following control and need to move in the reverse direction when it is required. These are the requirements that are need to be done on the practical model and the algorithm should be tested.

## 3. Automated Guided Vehicle

In 1953, the first period of AGVS began in America, followed by Europe a few years later. It lasted over two decades. The initial machines were distinguished technologically by simple track-guided systems and tactile "sensors" such as bumpers and emergency arrest handles with mechanical switches[1]. An American innovator conceived the concept of employing automation to replace the drivers of a tractor trailer for transporting products in the early 1950s.

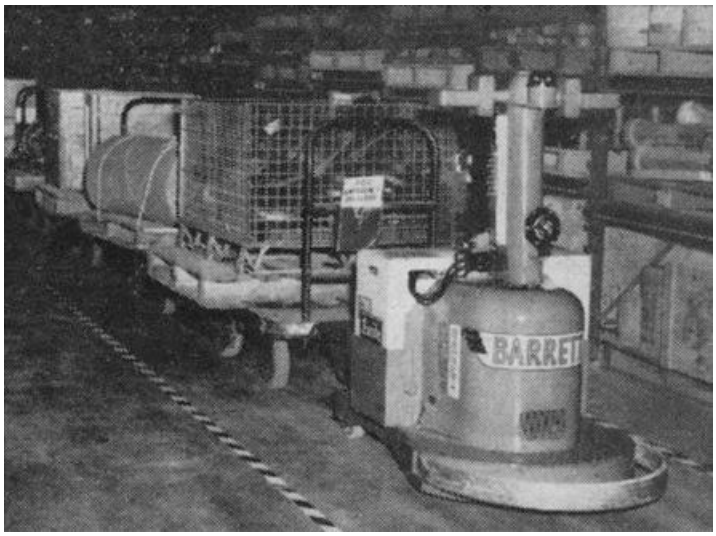


*Figure 1: One of the first American AGVS, built starting in 1954 as tractor for five trailers*[1]

The above figure 1 was the first implemented AGV in the tractor-trailer in 1954 Mercury Motor Freight Company in Columbia, South Carolina, for long-distance round-trip consignment shipping.

### 3.1 AGV Working

The AGV are self-operated vehicles which will be working under the guidance of software and sensors. These AGV's mostly works under the defined pathways. The working also consists of various types of operations that are being carried out for the movement. The operations that are taken under consideration to setup the AGV are explained below.

#### 3.1.1. Navigation

In the AGV design the navigation plays an important role which is used for the movement of the vehicle. If there is no proper navigation system in the industries the operation of Agv will be a huge problem. So selection of proper navigation system in the industry is the necessary step. The navigation technologies that are used in the industries are given in the table 1 below.

| Function | Type of Sensor | Application |
|---|---|---|
| **Safety Sensors** | Safe 2D Lidar | Detects whether it is safe or not automatically |
| | Bumper | Stops when there is contact |
| | Encoder | Detects speed and steering of the vehicle |
| **Environment Perception** | 2D & 3D Lidar, Ultrasonic, Camera, Radar | The contact with the objects can be avoided |
| **Navigation and Localization** | 2D & 3D Lidar, Ultrasonic, Camera, Radar | Mapping, Localization and Navigation |
| | Line sensors (magentic, inductive, optic sensors) | Navigation |
| **Load Handling** | Cameras, 2D or 3D | Pallet pocket detection |

| | | |
|---|---|---|
| | LiDAR, Ultrasonic | |
| | Optical distance sensors or Wire draw encoders | Fork Height Sensors |
| | Photocells, ultrasonic, inductive | Ensure the right load positioning |
| **Identification** | RFID, Laser or Image based bar code scanners | Transported material identification |

*Table 1: Various sensors used for navigation*[2]

### 3.1.2. Steering control

The steering control is used to navigate the AGV to move forward and backward movements. This can be achieved by following methods.

- Differential speed control: AGVs most commonly use this form of steering control. Two independent driving wheels are used in differential speed control. To turn, each drive wheel is turned at a separate speed. The two drives are driven at the same speed to go forward or backward. This type of movement is not used in towing applications.

- Steered wheel control: This method is similar to the steering control in cars or truck. In this the movement will be smooth and it offers precise turning. This type of movement is used in towing applications and in some operator handling vehicles.
- Combination steering: This method is the combination of the above two methods.[3]

Based on this research the Differential speed control is used in this project as similar to the AGVs.

### 3.1.3. Path Decision

AGVs must make decisions on which course to take. This is accomplished using a variety of methods, including

- frequency select mode (wired navigation only)
- path select mode (wireless navigation only) and

- a magnetic tape on the floor, which is used not only to direct the AGV but also to issue steering and speed commands.

As per the requirement the path select mode is used and the vehicle is tested on the path.

### 3.1.3. Traffic control

In the Flexible manufacturing systems there will be more AGVs will be in the operation process and they will be moving as per the requirement in the industry. So there should be proper traffic control in the system. This could be done based on the following types.

- Zone control
- Collision avoidance
- Combination control

## 3.2 Types of AGV'S

Automated guided vehicles come in a variety of forms. Many AGVs are meant to run without direct human interaction or guidance, comparable to other human-operated vehicles. They are listed below.

i. Fork lift AGV: The common type of AGV is forklift automatic guided vehicles, They're meant to transfer pallets in the same way that a human-operated forklift does, but without the necessity for a human operator.

*Figure 2: Fork lift AGV*[4]

ii. Towing AGV: Towing vehicles, also known as tugger automatic guided vehicles, are vehicles that tow one or more non-powered, load-carrying vehicles in a train-like pattern behind them. Powered towing vehicles, sometimes known as driverless trains, run on wheels.

*Figure 3: Towing AGV*[5]

iii. Unit Load Handlers: Unit load handlers transport single loads containing many things, such as a pallet or pail.


*Figure 4: Unit load handlers*[6]

iv. Heavy Burden Carriers: Heavy weight carriers are a type of AGV utilized in applications such as big assembly, casting, and coil and plate transfer for the heaviest loads. Self-loading capabilities and conventional, pivot, or omni-directional steering are available on some big freight carriers.

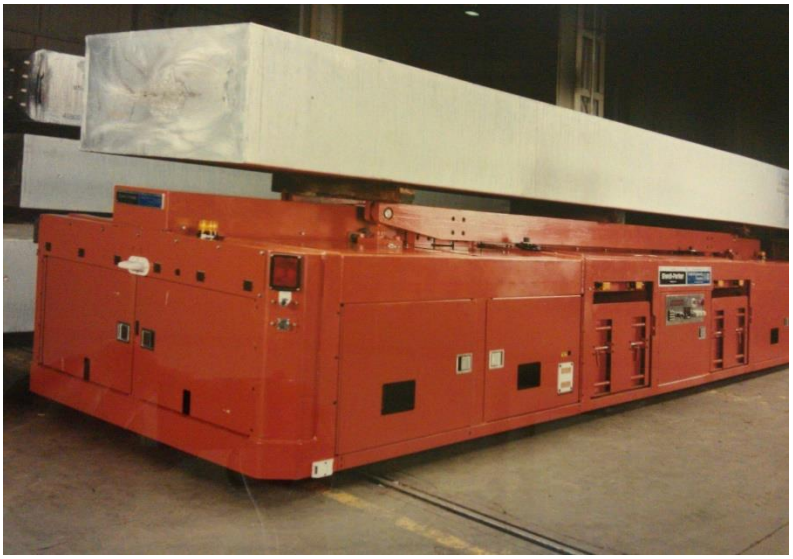

*Figure 5: Coil handler Agv*[7]

### 3.3 AGV technology used in Manufacturing systems

Inside a factory, these vehicles are designed for processing and transporting materials. They are automated and rely on guidance systems and its way of control. The AGVs are now experiencing rapid growth and has a very dynamic market. The future expansion of AGV systems will be fueled by

(i) the rise of flexible manufacturing systems,

(ii) the increased demand for customized AGVs, and

(iii) the adoption of industrial automation by SMEs, according to the report.

AGV systems are well-known and widely used in industries including as manufacturing, medical, and logistics.

The industries uses two types of system (i) Centralized architecture system and (ii) decentralized architecture system. Now the decentralized architecture system is booming in the industries because of its flexibility.[8]

#### 3.3.1. AGV based on Industrial 4.0

The industrial 4.0 concept is being used in many work places which helps the process flow to more efficient. In this industrial 4.0 there is no longer any need to rely on central structures to direct parts in the production or logistics ecosystem because data may flow freely between them. Members in a more complicated system may communicate directly with their other and with their local environment on multiple levels as industry 4.0 models migrate from the usual automation pyramid ISA 95 to RAMI 4.0.[8]

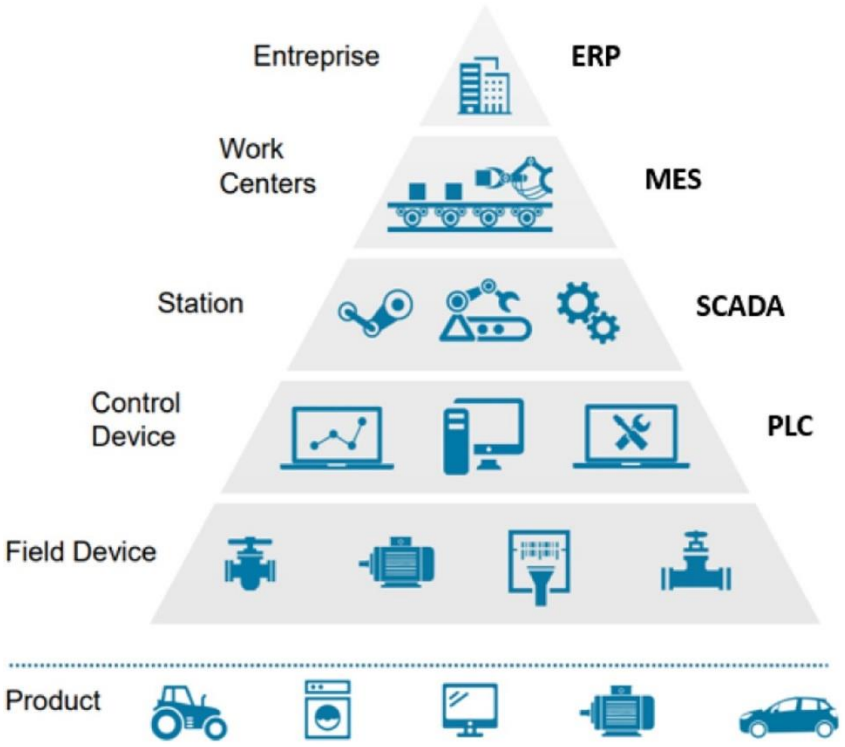


*Figure 6: ISA 95 model*[9]

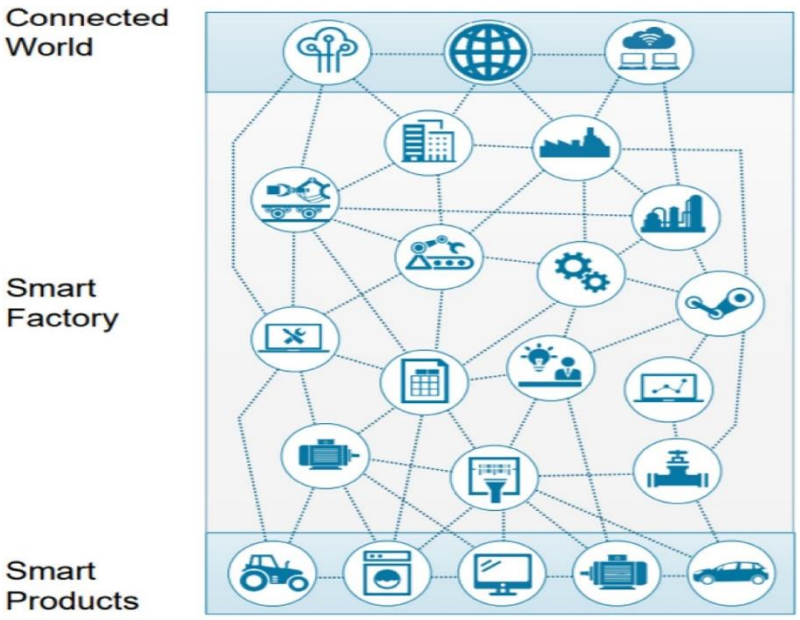


*Figure 7: RAMI 4.0 model*[10]

The above images 6 and 7 shows the evolution that happened in the industries. The RAMI 4.0 is not being used by all the industries but these are future technologies that can be implemented all over the world. Based on this the function of AGV should be more efficient than that are being currently used in the Industries.

### 3.4 Line Following AGV

The line following AGV is a vehicle which follows the line or a black path that is created as per the requirement in the system of operations. This line follower uses the IR sensor which tracks the line in an efficient way. This line follower AGV is a feedback system which is working under the guidance of the feedbacks from the sensor. In this the AGV tracks the black line, if the vehicle is over the some different surface the vehicle will stop.

The working of this function is shown in the figure 8 which shows the response of the vehicle in a general way.

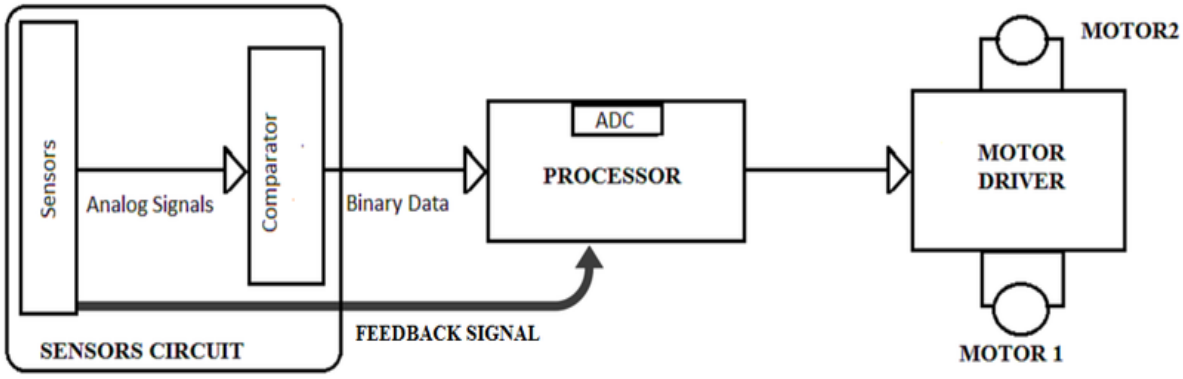


*Figure 8: Working function of line following AGV*

## 4. Components of AGV

The AGV has various components that are being used for various operations. The components that are required for the line following AGV are as follows.

- Base
- DC motor
- Arduino
- Bread board
- Infrared Sensor
- NRF2L01 Module
- H- Bridge(TB6612FNG)
- Batteries
- Wires

### 4.1 Base

For each and every vehicles in the world the base is the most important thing which supports every parts in it. If the base is not stable it could be the massive collapse in the movement and there will be a failure in implementing the operations. But as per the task the prototype model is only required for the operation so the normal 3D printed base is used which can hold the remaining components in it.

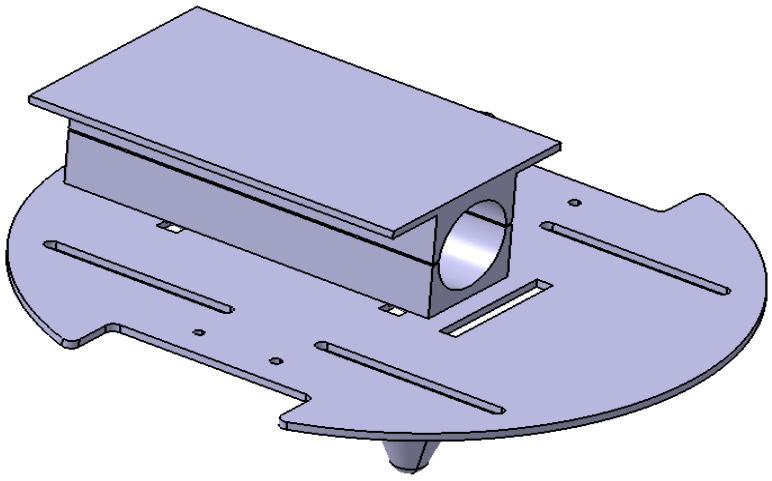

*Figure 9: Base design*

### 4.2 DC motor

The DC motor is a device that works on the principle of Flemings Left hand rule. This is used here for the transmission of the vehicle to be achieved. Generally, the motors generates the mechanical energy when the electrical energy is applied. There will be a basic components in the system which is

- Rotor: The rotor is a rotating component in which the rotating arm is which is bounded by the coils.
- Stator: The stator is a magnet.

There are two types of motors are available in the market. They are

- Brushed motors
- Brushless motors

The major difference between these two motors is the magnetic position. These two motors have the different advantages over one another.

*Figure 10: DC Motor*

### 4.3 Arduino

The Arduino is the main component of this project. This is the main mode of communication and the control system of the AGV. In this project the two types of Arduino boards are used. The one is Arduino nano and the other one is Arduino Mega. These two boards have different types of peripheral connections in them.

The Arduino nano is same as the Arduino uno the perfect board for beginners who want to learn about electronics and programming. The nano is the mostly used board for the beginners working with the platform. The Arduino nano is the most popular and documented board in the Arduino family. The Arduino nano is a microcontroller board that uses the ATmega328P microcontroller. There are

- 14 digital I/O pins (6 of which can be used as PWM outputs)
- Six analog inputs
- A 16 MHz ceramic resonator
- A USB connection, a power jack
- An ICSP header
- A reset button on the board.
- SPI communication

It comes with everything which is need to get started with the microcontroller; directly connected into the computer using a USB cable or power it with an AC to DC adapter or battery. In this nano board the prototypes are experimented without any fear and it can be tested several times if anything goes wrong it is economical to replace it.

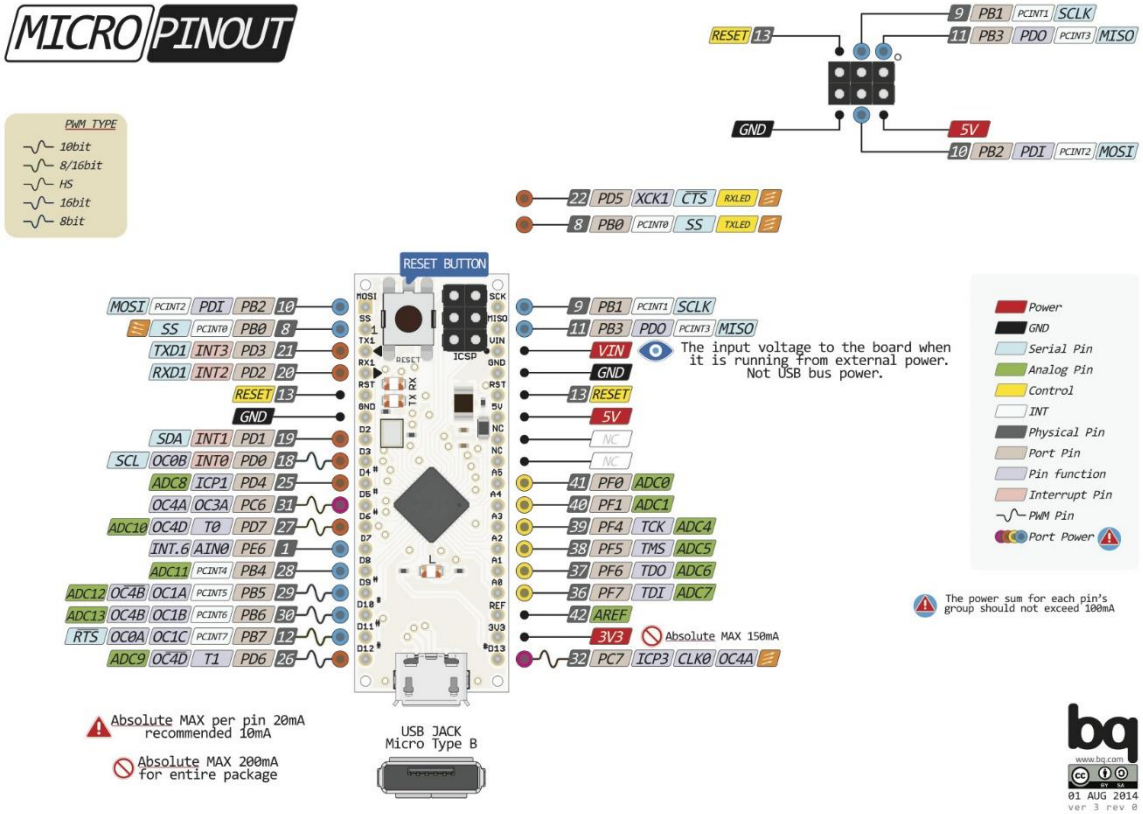


*Figure 11: Arduino Pinout* [11]

The Arduino MEGA 2560 is a board with extra I/O lines, sketch memory, and RAM for projects that require it. It is the suggested board for 3D printers and robotics applications since it has 54 digital I/O pins, 16 analog inputs, and more room for your sketch. This gives your projects lots of freedom and potential while keeping the Arduino platform's simplicity and effectiveness. The layout of this board is given below in the figure 12.

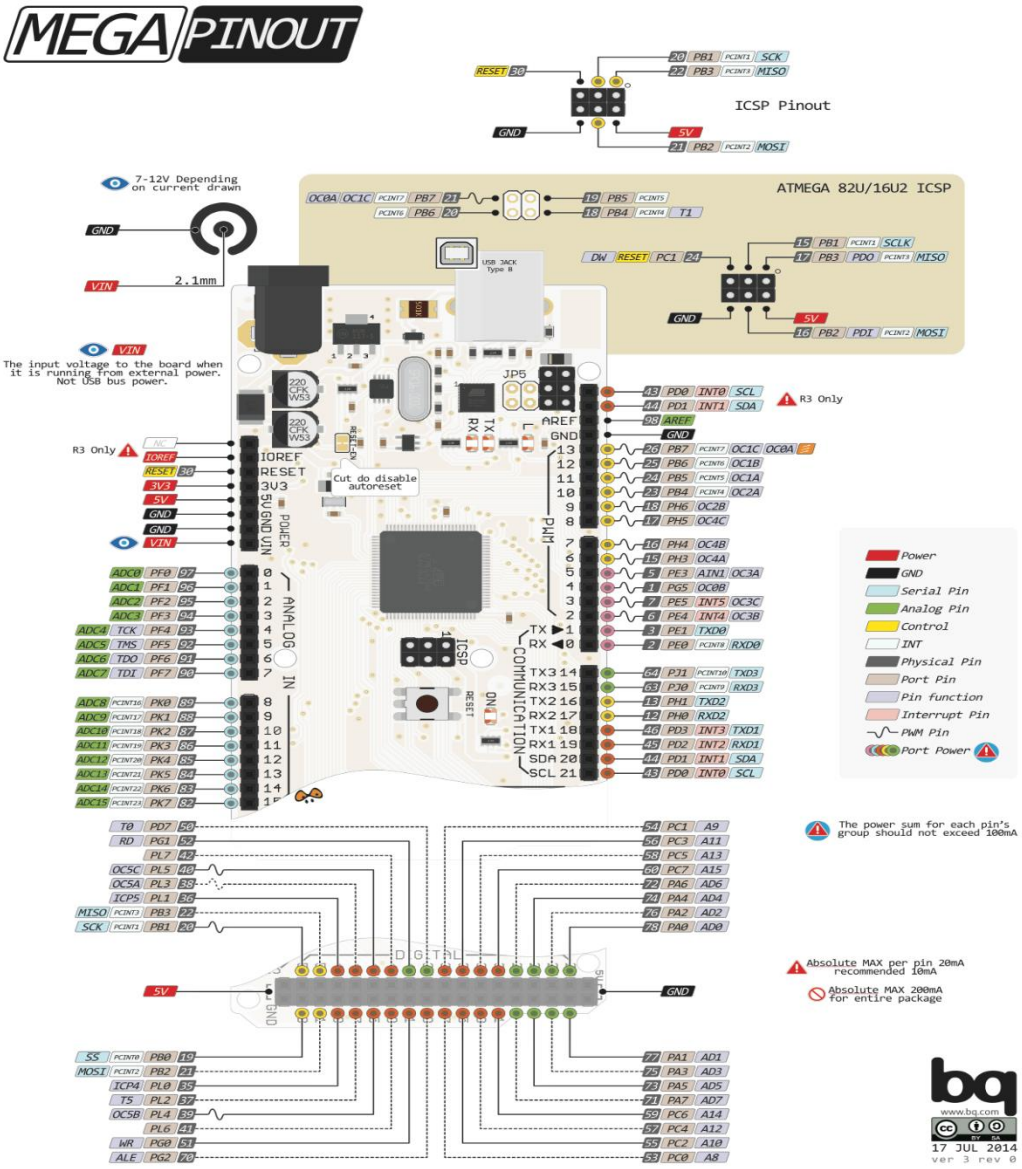


*Figure 12: Pinout diagram of Arduino Mega2560*[12]

This Arduino nano is used in this process for the motor movement and for the wireless communication in which it receives the data from the Arduino Mega which is used as a transmitter.

### 4.4 Bread board

When learning how to build circuits, breadboards are one of the most important components. A solderless breadboard is referred to as an electronics breadboard (as opposed to the type used to make sandwiches). These are useful for prototyping and building temporary circuits because they don't require any soldering.

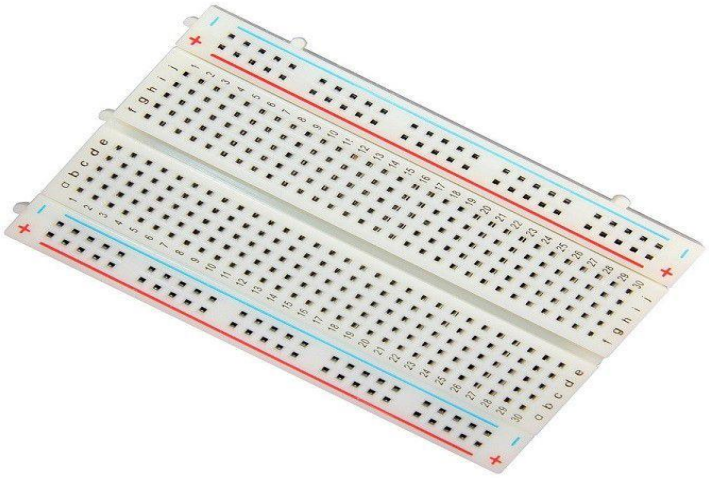

*Figure 13: Bread Board*[13]

### 4.5 Infrared Sensor

In this project the Infrared sensor is taken for tracking the line following purpose. The line following AGV is mainly depends on the light. For this the IR sensors will be the best solution. The IR sensor basically works on the principle of light tracking. Basically the IR sensor has IR transmitter and the IR receiver which is a photodiode. In this the IR transmitter transmits the light to the surface and the photodiode will be waiting for the feedback from the surface.

In this the black line is used for the path in which the vehicle needs to travel. The black surface is used because, when the light falls on the black surface the surface absorbs the light into it. Whereas, the white area or other surface will emit the light back.

The IR sensor will receive the signal back so the vehicle will not move as the receiver detects the light. When there is no reflection of light is received from the surface to the receiver the vehicle starts to move on that path.

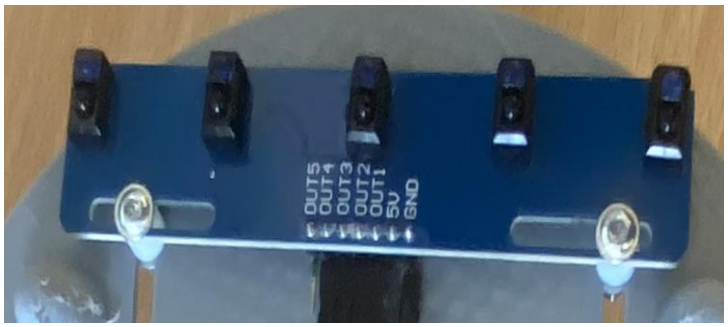


*Figure 14: IR Sensor*

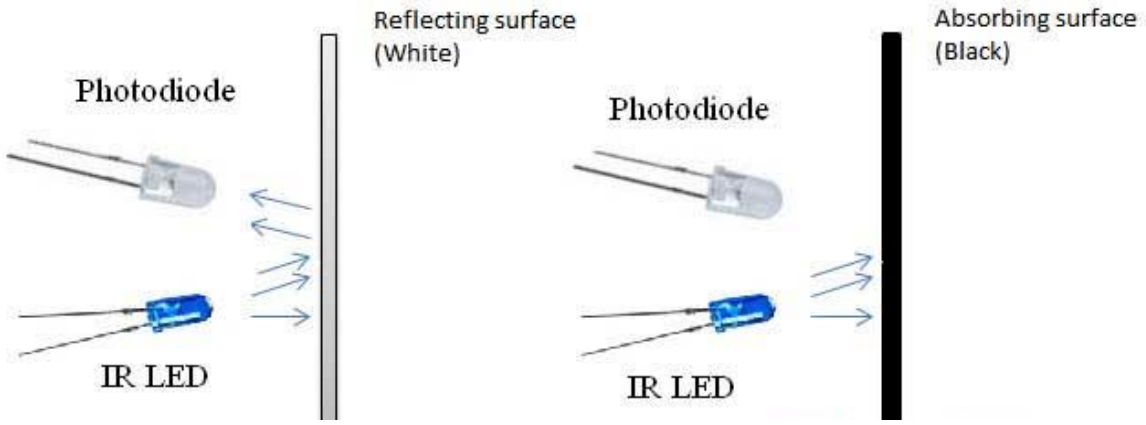


*Figure 15: Working of IR sensor*[14]

**4.6 NRF24L01**

The NRF24L01 is a transceiver module which is used to achieve the wireless communication. This is the very famous choice of communication while making the prototypes. This NRF24L01 uses 2.4 GHz band and it can operate with baud rates from 250 kbps up to 2 Mbps. This will function upto 100 m in an open space with lower baud rates.

| | |
|---|---|
| Frequency | 2.4 – 2.5GHz ISM band |
| Maximum output power | 0dBm |
| Working range | 100m (open space) |
| Data rates | 250Kbps / 1Mbps / 2Mbps |
| Input power | 5V |
| Operating Voltage | 1.9 – 3.6V |
| Standby Current | 22µA |

*Table 2: Specifications of NRF24L01*

This helps us to communicate between the two module even the displaying the necessary details can also be done. There will be three SPI pins in each of the transceiver modules. These pins should be connected to the SPI pin in the Arduino boards. After the connection of the pins these modules can be used as the transmitter or receiver as per the requirement function.

In this project one of the NRF24L01 model is connected with the Arduino Mega and the other one is connected to the Arduino nano. The module connected with Mega board is used as transmitter and other is a receiver which is connected with the vehicle.

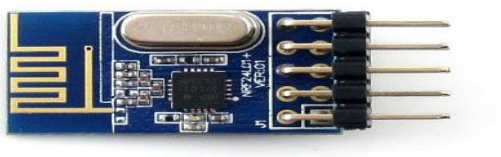

*Figure 16: NRF24L01 Module*

### 4.7 H-bridge (TB6612FNG)

The module TB6612FNG is a dual H-bridge which is used as dual motor drive system to control the movements of the motor. Generally, the H-bridge is a setup which consists of transistors in an unique way. This allows to switch the direction of current in the motor when it is connected. By changing the direction of current the direction of rotation of the motor can be changed and with the PWM input the speed of the motors also can be controlled.

In this project, this module is used which has two H-bridge which helps us to control the movement of AGV as per the requirement of motion.

There is also another type of module which is L298N. The L298N module is also the dual H-bridge module which also can able to control the motor like same as T6612FNG but when comparing these two TB6612FNG is better than L298N module based on the following criteria.

- Efficiency
- Peak current
- Size of the module
- Battery usage
- Power supply to the motor

To make this prototype in an efficient model this type of module is selected.

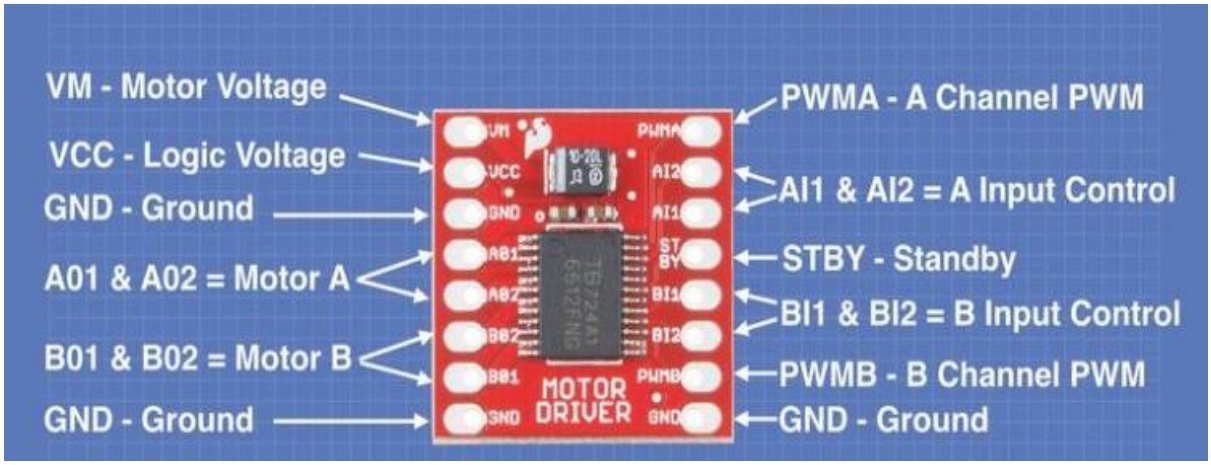


*Figure 17: TB6612 Module*[15]

### 4.8 Batteries

The power source is one of the main source for the vehicle to operate in an efficient way. In this prototype model the rechargeable battery bank is used as a power supply which can be charged as per the working time of the Agv.

### 4.9 Wires

The wires plays a vital role which is used for the connection between every modules to the Arduino to supply the power and to get the outputs. In this many jump wires are used for the connection.

## 5. Connection of AGV

There are some ways to connect the necessary components to make the protype works in an efficient way. The connections of each component is discussed below.

### 5.1 Motor Connection

The DC motor is controlled with the help of the TB6612FNG module which are connected to the Arduino board. Each motor has three control pins in which the two pins controls the direction of the motor and the another one pin is for speed.

In the Arduino the motor is connected as per the table 3 and the circuit diagram is shown in the figure 18.

| Motor A | Motor B | Function |
|---|---|---|
| PWMA = 3 | PWMB = 6 | Speed Control |
| AIN1 = 2 | BIN1 = 5 | Direction |
| AIN2 = 4 | BIN1 = 7 | Direction |

*Table 3: Connection of pins and function*

In the above table the PWM represents the speed control of the motor which can be done by the help of the Arduino by which we can able to do the necessary movement of the motor. These are only the inputs the output can be framed as per the requirement of the functions.

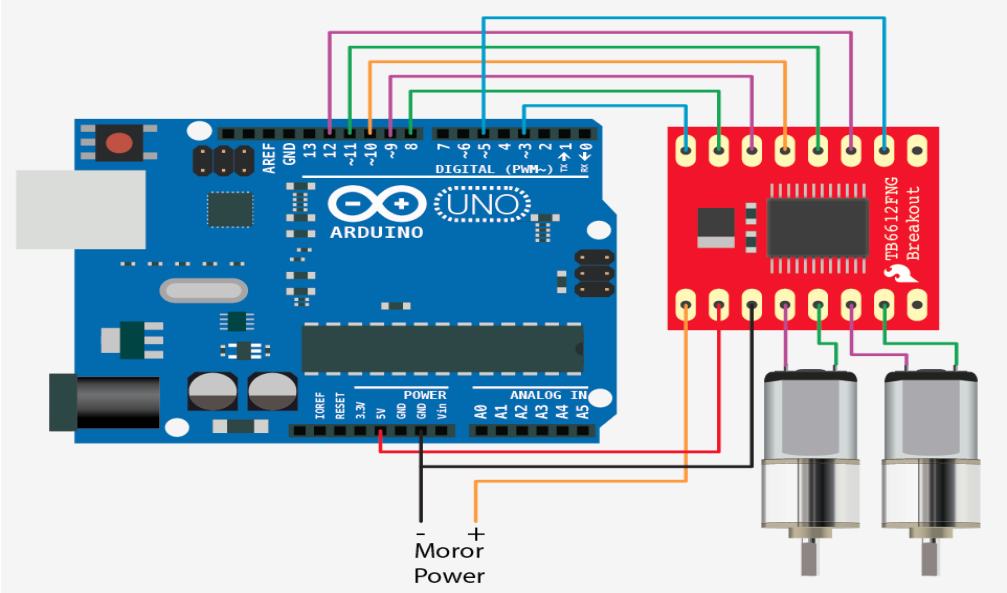


*Figure 18: DC motor connection*[16]

### 5.2 IR Sensor Connection

In this project the IR sensor module with 5 sensors is used for the line following purpose. In this the sensors are named as DL, SL, SC, SR and DR which represents the position and for the decision making purpose. The IR sensors are connected to the input ports 16, 17, 18, 19 and 20 respectively. In this the DR sensor is not used for any operation.

In this when the sensor detects the black line and gives the binary output as 0 and when it detects the light and receives the signal it gives the binary output as 1. Based on these outputs the AGV is controlled by the using the binary code equation as,

**4 SL + 2 SC + SR**

The above equation is framed based on the number of possibilities whereas the DL is used as Decision sensor where it detects only when there is any decision marking on the path.

### 5.3 Transceiver Connection

In this connection two NRF24L01 modules are used in which one is used as transmitter module which is connected with the Mega board and the other is used as receiver module which is connected to the vehicle.

The input of the receiver module is connected to the pins 14 and 15 which are used as the communication between the two modules. In this the RF24 library is used for the defining the pins.

The input of the transmitter module is connected to the pins 7 and 8 and the remaining dCLK, dMOSI, dCS, dDC, dRES are connected to the pins 13, 11, 4, 3, 2. These are defined with the U8glib. By using this library the displaying function can be achieved with the help of LCD or LED display. In this the Pull out function is used as the push button command in which the three jump wires are connected to the 25, 27 and 29 pins.

These are the connections that are used in this prototype model. Based on the connection the Arduino codes are created and tested. The whole connection of the motor is shown in the figure 19 and 20 below.

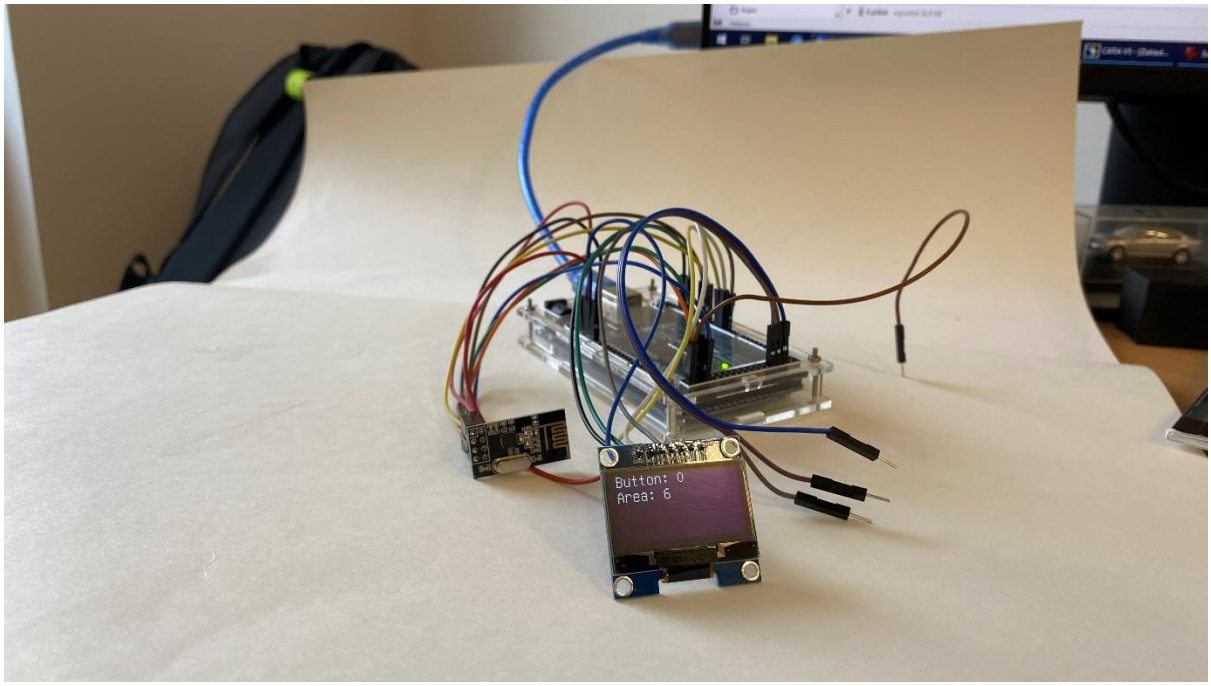


*Figure 19: Transmitter connection*

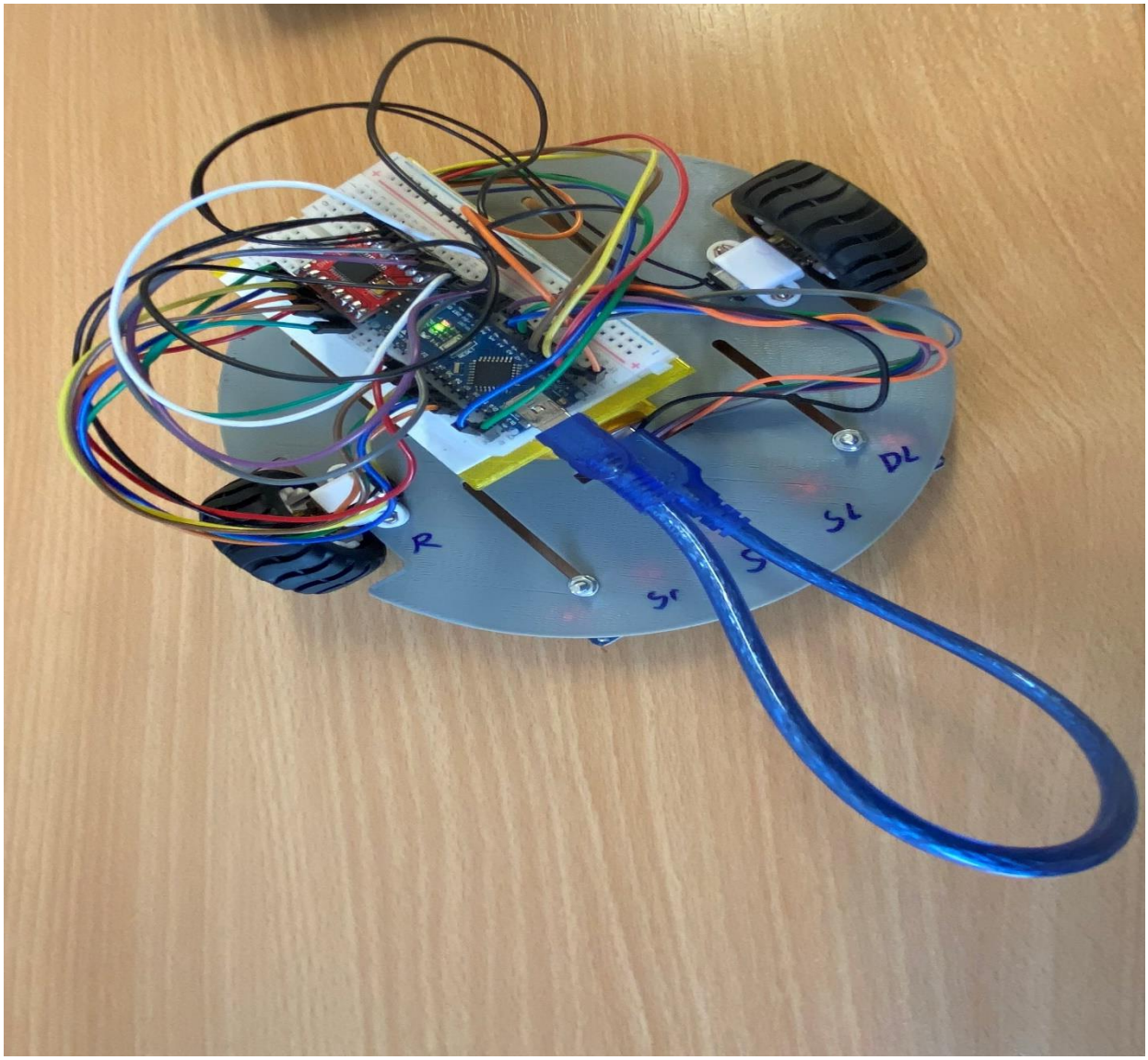


*Figure 20: Vehicle connection*

## 6. Practical Work

In this Arduino codes created and the testing results are discussed. There are lot of problems that are faced on the coding process and finally the codes are executed as per the requirement.

### 6.1 Testing path

The basic test path is created to check whether the forward and backward movement can be achieved without any error.

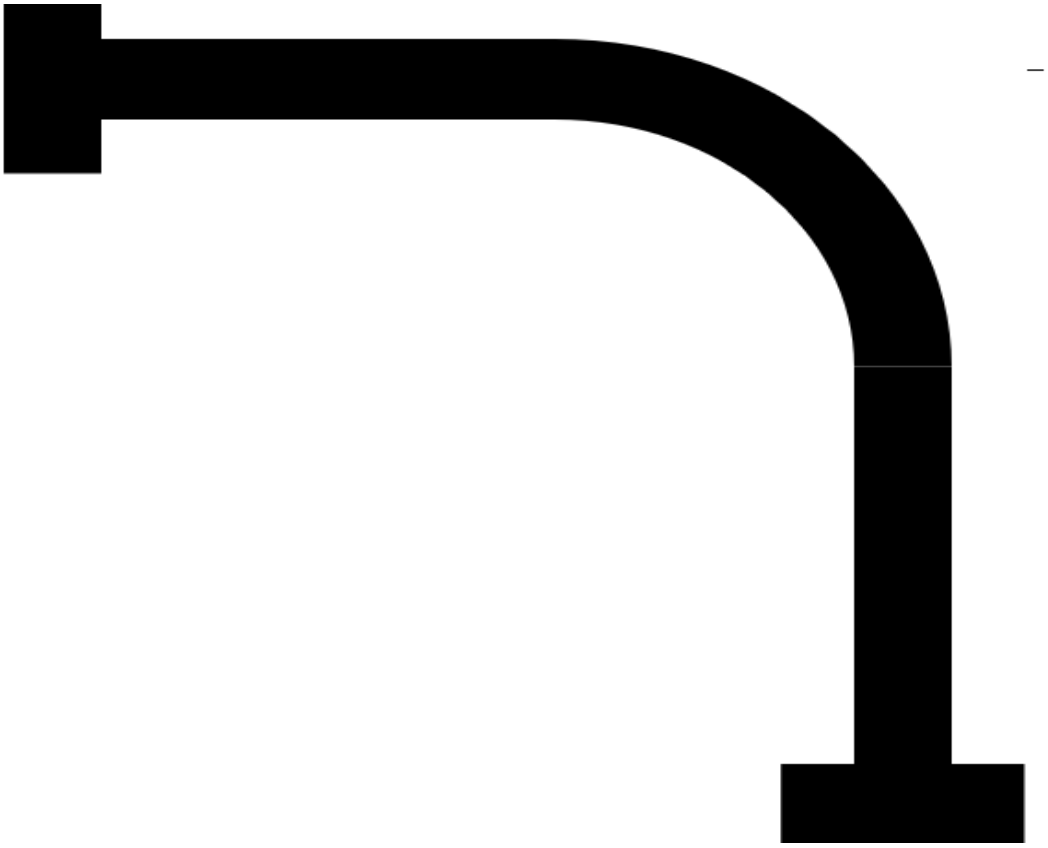

*Figure 21: Basic Test path*

The above figure 21 is the first path which is used to test the basic code whether the code that has been created working or not. This test path is used in the two stages of operation (i.e) for forward and backward movement.

In this the first horizontal area is the start area and the vehicle travels along the straight path in the forward motion and turns right and again stops in the stop area.

The movement is guided by the IR sensor based on the feedbacks that are received by the sensor. The IR sensor feedback possibilities are given below in the table 4.

| SL | SC | SR | Dir |
|---|---|---|---|
| 0 | 0 | 0 | Stop |
| 1 | 1 | 0 | Turn Right |
| 1 | 0 | 0 | Slight Right |
| 1 | 0 | 1 | Straight |
| 0 | 1 | 1 | Turn Left |
| 0 | 0 | 1 | Slight left |
| 1 | 1 | 1 | Error |

*Table 4: Movement based on the sensor Feedback*

The first forward movement has been achieved successfully but the problems started when coding the next following steps. In each steps there are lot of trail and errors are used and many possible solution are discussed.

1. At first when the reverse movement in the path is tried to achieve which is one of the necessary movement in the path. The problems starts while testing the codes. In this prototype one set of IR sensor is used to achieve the line tracking but while using one set of sensor the whole set of sensors are going out of the line while going back in the path during the turns. This results in the stoppage of the AGV and it shows error. This situation is shown in the below figure 22.

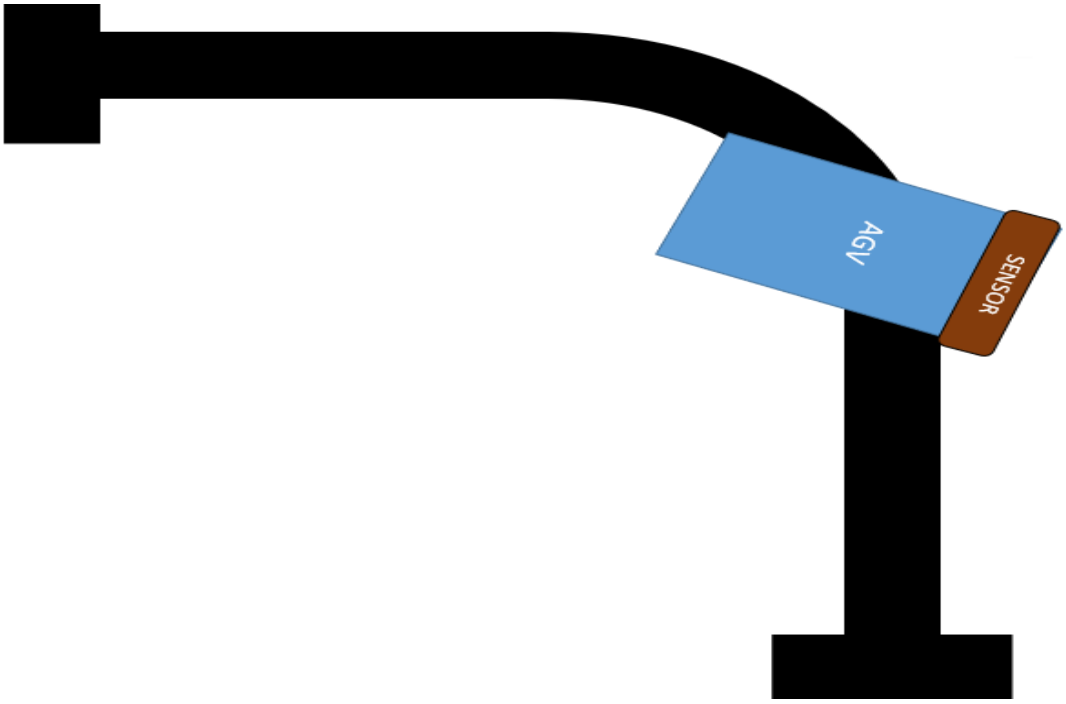


*Figure 22: Position of the AGV while going backwards*

2. Based on the above scenario as shown in the figure 22 the another solution is tried to achieve whether this satisfies the requirement. The solution is instead of using one pair of sensors the two set of two pair of sensor is used but for the line tracking. In this one pair of sensor is used in front and the another set of sensor is used in the back. In this the sensors in the front detects the forward motion and the back sensor is used for the reverse motion. But while using this solution the same problem as the above case the back sensor detects the line whereas the front sensor went out of the line because of this vehicle shows error and it will stop. This case is shown below in the figure 23.

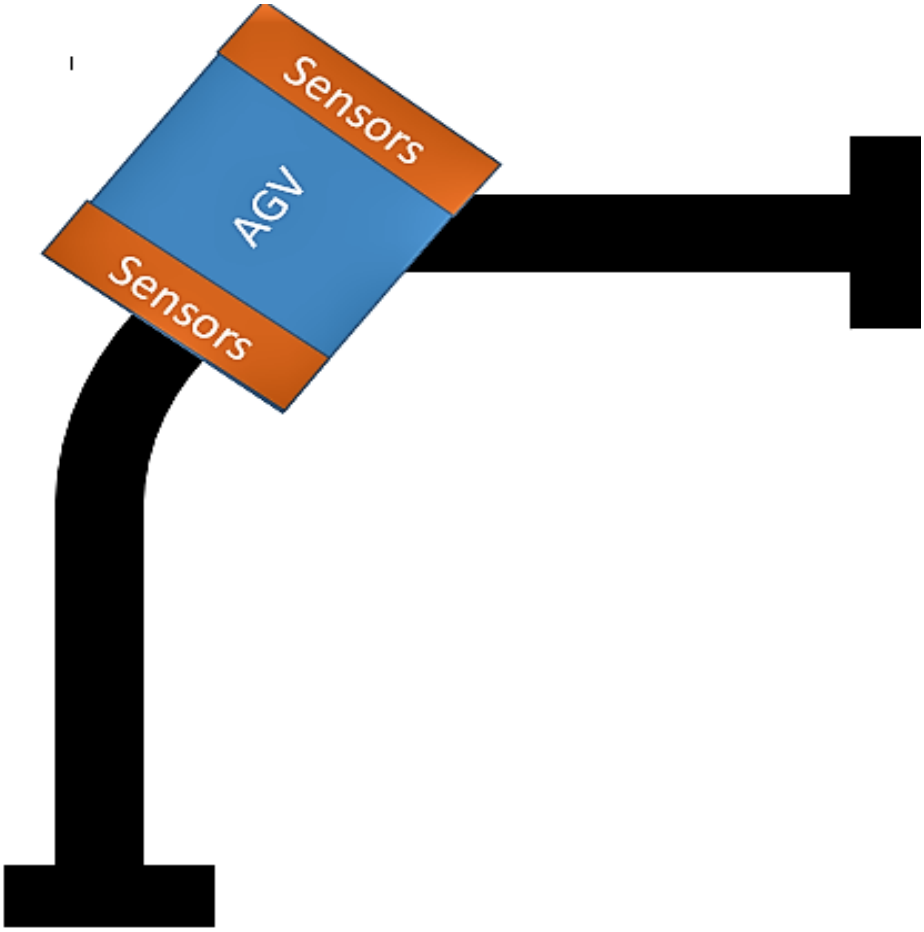


*Figure 23: Position of AGV while moving backwards using two sensors*

3. The next solution to achieve the reverse motion is using some change in the path. In this the path is slightly changed in which without moving in turns the addition path is defined straight in the as same as the straight path after the turn. But in this solution another problem arises when the AGV moves in a path the vehicle there will be multiple paths so it need to judge the path which to choose. But this solution can be achieved but when compared to the reality this situation can be some problem. For example when it is implemented in an industry scenario there may be unnecessary movement will be which can cause some effects in the production process and there will be additional power consumption to achieve this solution.

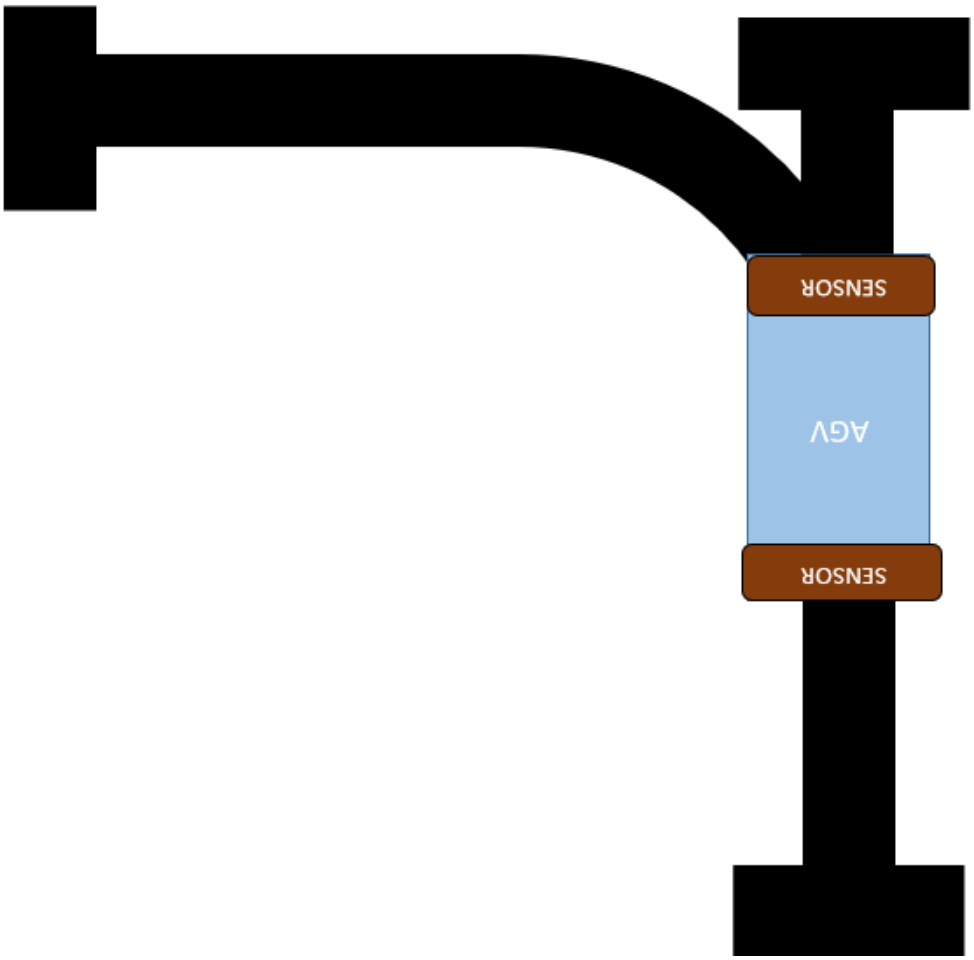


*Figure 24: New path for the AGV*

4. As per the lot of possibilities finally one solution is framed and achieved to make the movement of the AGV to work in an efficient way and works without any stoppage or error occurring. In this only one pair of sensor is used in the front of the vehicle and to achieve the reverse movement the successfully a path decision marking is placed in one side of the path and making one extra sensor to detect the marking and made to move the vehicle in an perfect way in the path.

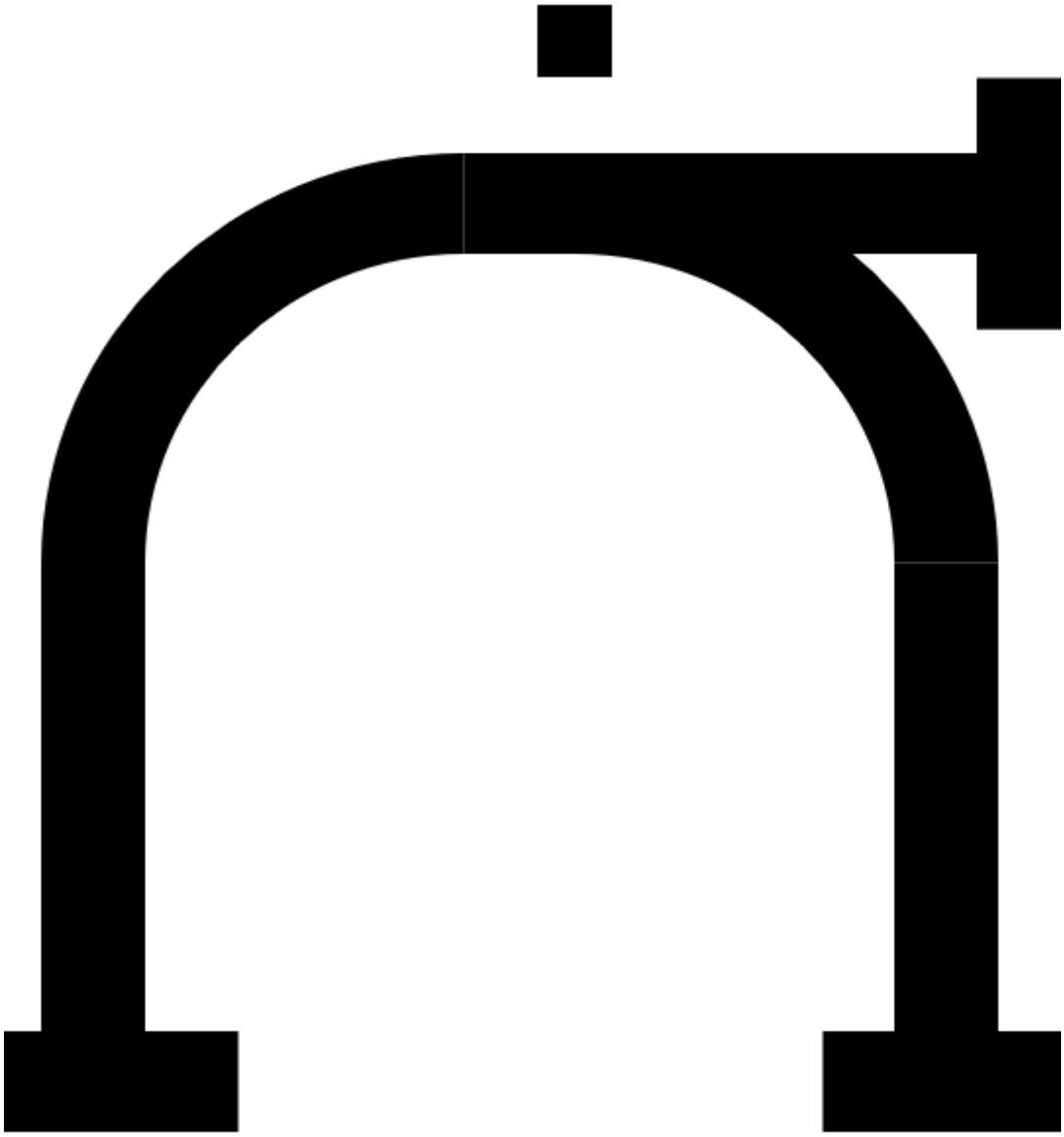

*Figure 25: Path with the decision marking*

Based on this much trial and error methods the final path is decided and using the Arduino codes a loop is created and the path is tested which works efficiently in the designed path. The designed path is given shown below in the figure 26.

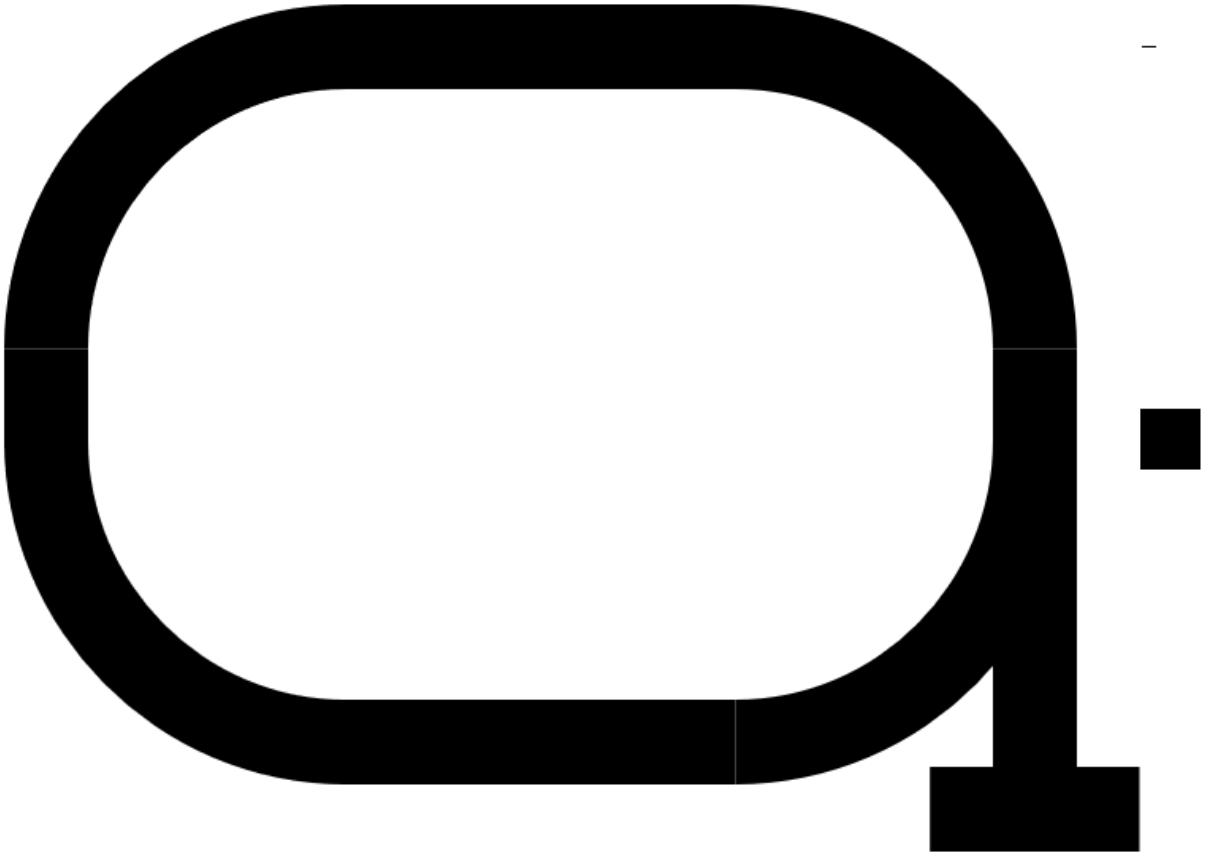

*Figure 26: Testing Path*

The codes for this path and the automation that implemented in the path are discussed and the final decided path in the following steps.

## 6.2 Arduino code

To create an AGV the control algorithm is required for the working in an efficient way. This can be achieved by Arduino codes. These Arduino coding becomes simple by the help of Integrated Developing Environment (IDE). The framework environment of the Arduino coding program is shown in the figure 27.

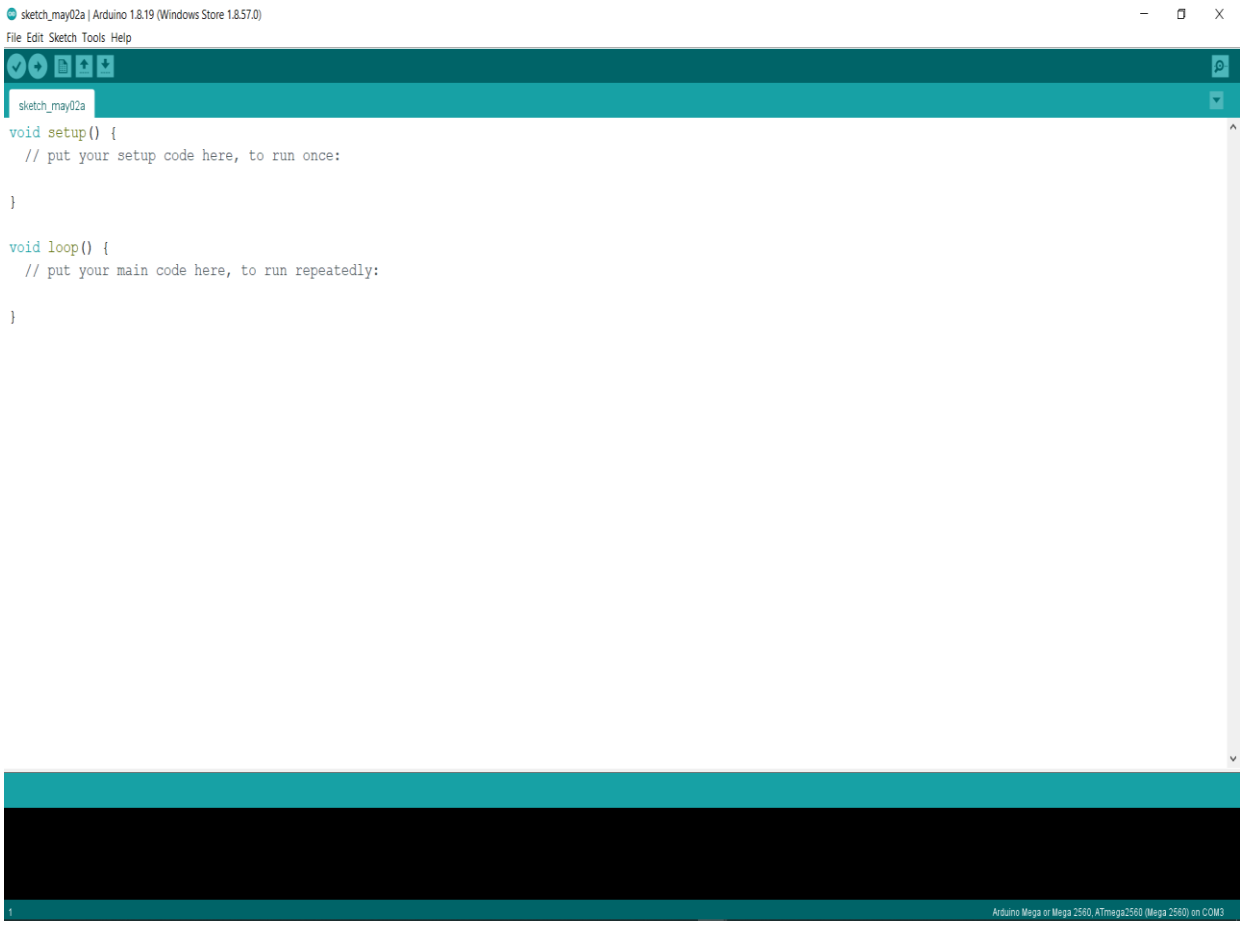


*Figure 27: Framework of Arduino IDE*

In this at the top of the layout there are five function which are verify, upload, new, open, save.

I. Verify: This function is used to compile the codes which helps to identify whether there is any syntax error has been in the codes that we have created.
II. Upload: This function is used to upload the codes to the necessary boards using the USB port.
III. New: This function is used to create the new set of codes.
IV. Open: This function is used to open the codes that are previously saved in the system.
V. Save: This function helps to save the coded file in the system.

The following steps represents the setup of the motor which is used to control the drive system. In this the global variables represents the starting area which represents the path of the motor and the speed of the motor then the byte Dir and byte Dl represents the data of the sensors.

The Void setup() is the function which is used to setup the necessary things as per the requirements of the task. In the below image the motor pins are connected to the output pins in the Arduino which will send the necessary outputs to the motor and controls as per our codes. The pins of the sensors are connected to the Input pins of the Arduino which collects the output of the sensors as the input and gives these inputs as the output to the motor. These are the things that are shown below in the codes.

```
// Global variables
int Area = 1;
int PrevSt = 255;
byte Dir;
byte Dl;

void setup()
{
pinMode(STBY, OUTPUT);

pinMode(PWMA, OUTPUT);
pinMode(AIN1, OUTPUT);
pinMode(AIN2, OUTPUT);

pinMode(PWMB, OUTPUT);
pinMode(BIN1, OUTPUT);
pinMode(BIN2, OUTPUT);

pinMode(DL, INPUT);
pinMode(SL, INPUT);
pinMode(SC, INPUT);
pinMode(SR, INPUT);
//  Serial.begin(57600);
}
```

- Motor Codes:

```
void move(int motor, int speed, int direction)
{
  // Move specific motor at speed and direction
  // motor: 0 for B 1 for A
  // speed: 0 is off, and 255 is full speed
  // direction: 0 clockwise, 1 counter-clockwise

  digitalWrite(STBY, HIGH); //disable standby

  boolean inPin1 = LOW;
  boolean inPin2 = HIGH;

  if(direction == 1)
  {
    inPin1 = HIGH;
    inPin2 = LOW;
  }

  if(motor == MotLeft)
  {
    digitalWrite(AIN1, inPin1);
    digitalWrite(AIN2, inPin2);
    analogWrite(PWMA, speed);
  }
  else
  {
    digitalWrite(BIN1, inPin1);
    digitalWrite(BIN2, inPin2);
    analogWrite(PWMB, speed);
  }
}

void stop()
{
  //enable standby
  digitalWrite(STBY, LOW);
}
```

In the above codes the setup for the movement of the motor is shown. In this function the 0 represents the Motor B and 1 represents the Motor A. While for the direction of rotation 0 is for anti-clockwise rotation and 1 is for clockwise rotation and for speed 0 is off and 255 is

full speed. The digitalWrite function helps to start the motor when it is at the high speed and stops the motor when it detects the 0.

The Boolean function is used to check the given statements value is true or false. In this the inpin1 and inpin2 is given as low and high which will be waiting for the necessary input. If the direction or the motor input is changed as per the void loop that has been created it will helps the motor to move as per the codes.

After the motor setup the loop is created for the movement of the motor. In the Arduino programming the void loop() is used to create the functions that need to be work in a continuous loop. At first in this loop the sensors are set to read the status of the line. As per the inputs from the sensors the codes are created and the switch function is used to switch the areas as per the readings received from the sensors. This is done in different cases which is explained in the following steps.

- Case 1: In this case the vehicle moves in the forward direction in the path and it turns right in the path and it continue in the same straight motion until the DL sensor detects the decision marking in which the vehicle stops and changes the area from 1 to 2 if the AGV wants to stop or from 1 to 4 if AGV wants to continue to next stop. The output possibilities from the sensor are as follows and it has been coded in this case. This would be the line following motion in this case.

| SL | SC | SR | Dir |
|---|---|---|---|
| 1 | 1 | 0 | Turn Right |
| 1 | 0 | 0 | Slight Right |
| 1 | 0 | 1 | Straight |
| 0 | 1 | 1 | Turn Left |
| 0 | 0 | 1 | Slight left |
| 1 | 1 | 1 | Rotate right (while Transceiver is used) |
| 0 | 0 | 0 | Stop (while Transceiver is used) |

*Table 5: Movement possibilities for case 1*

```
case 1:  // area 1
 {
   if (Dl == 0)
   {
     DlStop++;
     while (Dl == 0)
     {
       Dl = digitalRead(DL);
       delay(50);
     }
     stop();
     if (TargetStop == DlStop)
       Area = 2;
     else
       Area = 4;
     if (DlStop >= MAX_STOPS)
       DlStop = 0;
     while (!nRF.write(&Area, sizeof(Area)));
     break;
   }
   switch (Dir)
   {
     case 0:  // stop
     {
       move(MotLeft, 128, 1); // mot.1, half speed, forward
       move(MotRight,128, 1); // mot.2, half speed, forward
       while (Dir == 0)
       {
         readsensors();
         delay(50);
       }
       stop();
       DlStop = 0;
       Area = 6;
       while (!nRF.write(&Area, sizeof(Area)));
       // switch back for receiving
       nRF.startListening();
       break;
     }
     case 1:  // forward slightly left
     {
       move(MotLeft,128, 1);  // mot.1, half speed, forward
       move(MotRight,255, 1); // mot.2, full speed, forward
       break;
```

```
      }
      case 3:  // forward strictly left
      {
        move(MotLeft,  0, 1); // mot.1, stop, forward
        move(MotRight,255, 1); // mot.2, full speed, forward
        break;
      }
      case 4:  // forward slightly right
      {
        move(MotLeft,255, 1);  // mot.1, full speed, forward
        move(MotRight,128, 1); // mot.2, half speed, forward
        break;
      }
      case 5:  // forward
      {
        move(MotLeft, 255, 1); // mot.1, full speed, forward
        move(MotRight,255, 1); // mot.2, full speed, forward
        break;
      }
      case 6:  // forward strictly right
      {
        move(MotLeft, 255, 1); // mot.1, full speed, forward
        move(MotRight, 0, 1); // mot.2, stop, forward
        break;
      }
      case 7:  // turn right
      {
        move(MotLeft, 128, 1); // mot.1, half speed, forward
        move(MotRight,128,0); // mot.2, half speed, backward
        break;
      }
    }
    break;
  }
```

- Case 2: After changing the area the vehicle continues in the forward motion and it reaches the stop area where all the sensors are over the line and it will delay for 2 seconds (to simulate the loading or unloading the material) and the vehicle will change its area from 2 to 3. From where the vehicle need to achieve the reverse motion. The codes of area 2 is shown below and the sensor possibilities are also shown in the table below.

```
case 2:  // area 2 - straight, after DL marking
    {
      Dir = Dir & 6;
      switch (Dir)
      {
        case 0:  // stop, channge area
        {
          stop();
          delay(2000);
          Dl = digitalRead(DL);
          if (Dl == 0)
          {
             move(MotLeft,128,0);//mot.1,half speed, back
             move(MotRight,128,0);//mot.2,half speed, back
             while (Dl == 0)
             {
               Dl = digitalRead(DL);
               delay(50);
             }
          }
          Area = 3;
          while (!nRF.write(&Area, sizeof(Area)));
          break;
        }
        case 2:  // turn left
        {
          move(MotLeft,128,0);// mot.1, half speed, backward
          move(MotRight,128,1);// mot.2, half speed, forward
          break;
        }
        case 4:  // forward
        {
          move(MotLeft,255,1);// mot.1, full speed, forward
          move(MotRight,255,1);// mot.2, full speed, forward
          break;
        }
        case 6:  // forward slightly right
        {
          move(MotLeft,255,1);// mot.1, full speed, forward
          move(MotRight,128,1);// mot.2, half speed, forward
          break;
        }
        default:
        {
```

```
            stop();
            break;
          }
        }
        break;
      }
```

| SL | SC | SR | Dir |
|---|---|---|---|
| 1 | 1 | X | Slight Right |
| 1 | 0 | X | Straight |
| 0 | 1 | X | Slight left |
| 0 | 0 | X | Stop (Change Area) |

*Table 6: Movement possibilities for case 2*

- Case 3: In this case the reverse movement of the vehicle is made where the vehicle after reaching the stop area it will start to move in the reverse direction. In this case the vehicle will travel until the Decision marking sensor detects the decision marking. After it detects the marking the vehicle starts to change the area from 3 to 4. To make the sensors to detect the line perfectly the vehicle is made to move little forward and made to stop after that it will change the area. This codes of this function is shown below in the codes and table 7 shows the sensor detection. This uses the same sensor detection as case 2 but only the motion is reversed.

| SL | SC | SR | Dir |
|---|---|---|---|
| 0 | 1 | X | Slight Right |
| 1 | 0 | X | Straight |
| 1 | 1 | X | Slight left |
| 0 | 0 | X | Backward slowly |

*Table 7: Movement possibilities for case 3*

The reverse motion is achieved by setting the motor movement from 1 to 0, where 0 gives the anticlockwise direction. In this the vehicle is made to move in the half speed.

```
case 3:  // area 3 - bacward movement
    {
      if (Dl == 0)
      {
        move(MotLeft,128, 1);  // mot.1, half speed, forward
        move(MotRight,128, 1); // mot.2, half speed, forward

        while (Dl == 0)
        {
          Dl = digitalRead(DL);
          delay(50);
        }
        stop();
        Area = 4;
        while (!nRF.write(&Area, sizeof(Area)));
        break;
      }
      Dir = Dir & 6;
      switch (Dir)
      {
        case 0:  // backward slowly
        {
          move(MotLeft,128,0);  // mot.1, half speed, backward
          move(MotRight,128,0); // mot.2, half speed, backward
          break;
        }
        case 2:  // turn left
        {
          move(MotLeft,128,0);  // mot.1, half speed, backward
          move(MotRight,128,1); // mot.2, half speed, forward
          break;
        }
        case 4:  // backward
        {
          move(MotLeft,128,0); // mot.1, half speed, backward
          move(MotRight,128,0); // mot.2, half speed, backward
          break;
        }
        case 6:  // turn right
        {
          move(MotLeft,128,1); // mot.1, half speed, forward
          move(MotRight,128,0); // mot.2, half speed, backward
          break;
        }
```

```
      default:
      {
        stop();
        break;
      }
    }
    break;
  }
```

- Case 4: In this case the vehicle continue to move in the forward direction. But in this case again a problem arises to continue in the path. So again the decision making sensor need to be used to decide the path to continue because in this path there will be junction so the sensor will detect the another path and it will stops. So in this case again the vehicle is made to decide the area from 4 to 5 until the decision sensor detects the path it will stops and it changes to area 5. In this the left sensor is ignored. The code for this case is shown below.

| SL | SC | SR | Dir |
|---|---|---|---|
| X | 0 | 0 | Slight Right |
| X | 0 | 1 | Straight |
| X | 1 | 0 | Right |
| X | 1 | 1 | Slight Left |

*Table 8: Movement possibilities for case 4 & 5*

```
case 4:  // area 4 - after crossway - ignore left sensor
 {
   if (Dl == 0)
   {
     stop();
     Area = 5;
     while (!nRF.write(&Area, sizeof(Area)));
     break;
   }
   Dir = Dir & 3;
   switch (Dir)
   {
     case 0:  // forward slightly right
```

```
      {
        move(MotLeft,128,1); // mot.1, full speed, forward
        move(MotRight,0,1); // mot.2, stop, forward
        break;
      }
      case 1:  // forward
      {
        move(MotLeft,128,1); // mot.1, full speed, forward
        move(MotRight,128,1); // mot.2, full speed, forward
        break;
      }
      case 2:  // turn right
      {
        move(MotLeft,128,1);  // mot.1, half speed, forward
        move(MotRight,128,0); // mot.2, half speed, backward
        break;
      }
      case 3:  // turn left
      {
        move(MotLeft,128,0);  // mot31, half speed, backward
        move(MotRight,128,1); // mot.2, full speed, forward
        break;
      }
      default:
      {
        stop();
        break;
      }
    }
    break;
  }
```

- Case 5: In this case the operations of the vehicle are same as in the case 4. But the only addition in this case is when the DL = 1 then the area is switched to 1 and continue the same process in a loop continuously. The codes for this case is shown below.

```
case 5:  // area 5 - same as 4, except Dl is over the second
crossway path
    {
      if (Dl == 1)
      {
        stop();
```

```
        Area = 1;
        while (!nRF.write(&Area, sizeof(Area)));
        break;
      }
      Dir = Dir & 3;
      switch (Dir)
      {
        case 0:  // forward slightly right
        {
          move(MotLeft,128,1); // mot.1, full speed, forward
          move(MotRight,0,1);  // mot.2, stop, forward
          break;
        }
        case 1:  // forward
        {
          move(MotLeft,128,1); // mot.1, full speed, forward
          move(MotRight,128,1);// mot.2, full speed, forward
          break;
        }
        case 2:  // turn right
        {
          move(MotLeft,128,1); // mot.1, half speed, forward
          move(MotRight,128,0);// mot.2,half speed, backward
          break;
        }
        case 3:  // turn left
        {
          move(MotLeft,128,0);// mot.1, half speed, backward
          move(MotRight,128,1);// mot.2, full speed, forward
          break;
        }
        default:
        {
          stop();
          break;
        }
      }
      break;
    }
```

This are the codes that are created for testing the line follower. To implement the wireless communication in the prototype the new path is created to implement this function. The below figure 28 shows the final path for the operation.

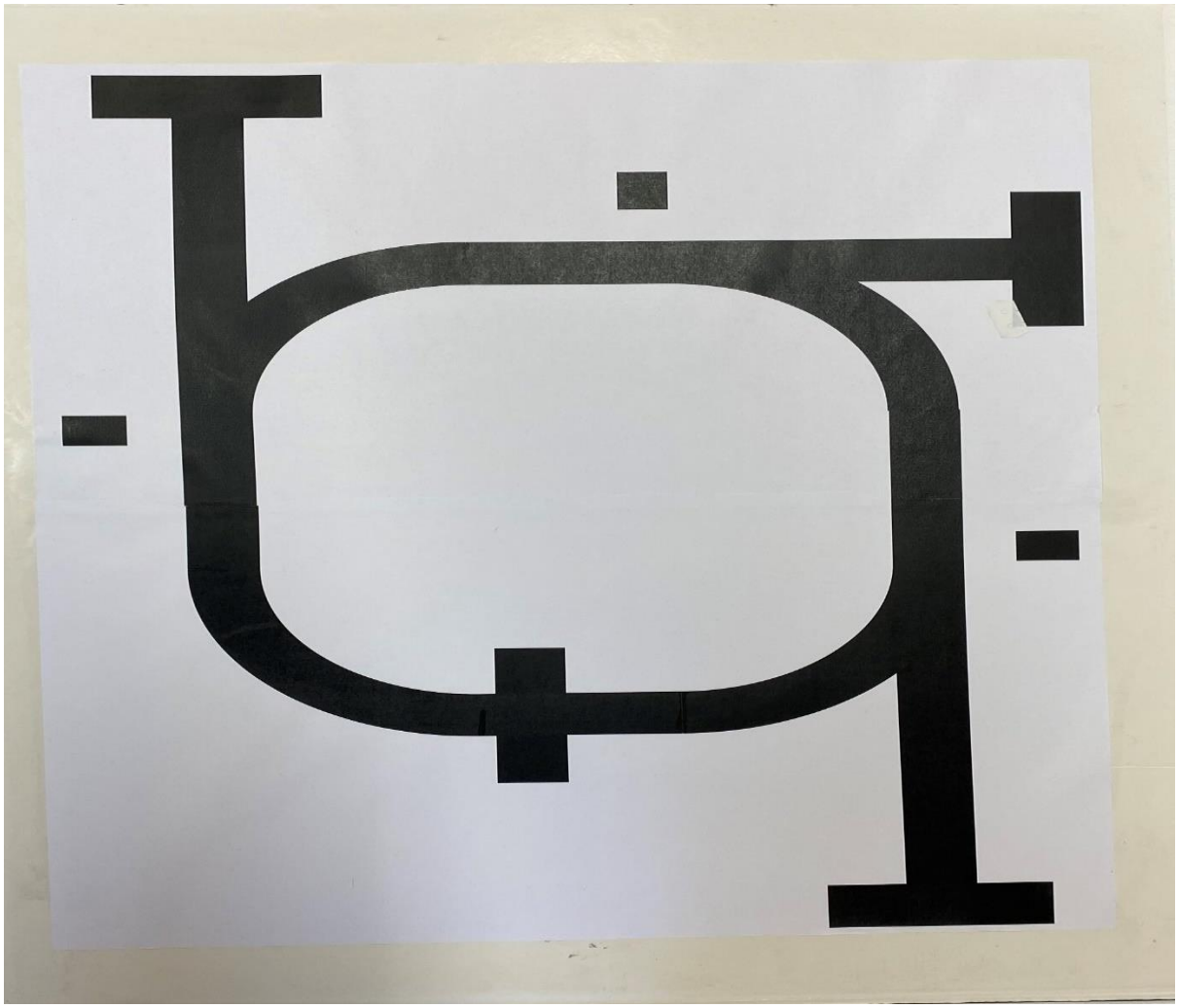

*Figure 28: Final Test Path*

In this path extra three stops has been created and in the middle of the path there is a waiting area where the vehicles will be waiting for the signal to receive from the transmitter. In this the transmitter sends the signal by the help of buttons in which each stops are given with separate buttons. This type of system is the centralized system where the computer is the master and the vehicle which is the slave. The communication will be taking place between these two systems.

- Transmitter code: In this system first the system is set for transmitting the signal response and it will also receive the response from the receiver which is used to display the current situation of the vehicle. In this the system will show the error function or it will not transmit the inputs to the vehicle when all the three buttons or any two buttons are pressed. While doing this will show the error. It is designed to

function when any one of the button is pressed the vehicle will move as per the input button that is being pressed.

- Receiver code: This receiver codes are coded in the same vehicle codes to execute the functions of the robot as per the selection of button. Only the little changes are made in the codes which are explained previously. In this the button inputs are added and in the waiting area there is a problem arises that after stopping in the waiting area the sensors reading stopped. To overcome this the vehicle is made to move little forward until the sensor detects the line and it is made to stop for receiving the signal. This is designed as that when the vehicle moves to one stop then again it will move to the waiting area.

| Transmitter | Receiver | Function |
|---|---|---|
| 1 | Response = 1 | Move to stop 1 |
| 2 | Response = 2 | Move to stop 2 |
| 3 | Response = 3 | Move to stop 3 |
| All buttons are pressed | Error | Error |

*Table 9: Operation of Transceiver*

- Working Loop For Receiver:

```
case 6:
    {
      // variables for recieved data and response
      byte recData;
      byte response;
      // if nRF is connected and detect recieving data
      // start listening
      if( nRF.available())
      {
        // wait for data
```

```
        while (nRF.available())
        {
          // in case of received data the content will be in
variable recData
          nRF.read(&recData, sizeof(recData) );
        }
        // encoding of recieved data
        switch(recData)
        {
          case 1:
            response = 1;
            TargetStop = 1;
            Area = 1;
            break;
          case 2:
            response = 2;
            TargetStop = 2;
            Area = 1;
            break;
          case 3:
            response = 3;
            TargetStop = 3;
            Area = 1;
            break;
          // if error, send back 0
          default:
            response = 0;
            break;
        }
        // end of recevining data
        nRF.stopListening();
        // waiting for switching between receiving and
transmitting
        delayMicroseconds(100);
        // sending the result
        while (!nRF.write( &response, sizeof(response) ));

      }
      break;
    }
  }
}
```

- Working Loop For Transmitter:

```
void loop()
{
  // variable for storing received data
  unsigned long recievedData;
  do
  {
    // loop for reading buttons
    byte Butt1 = digitalRead(Button1);
    byte Butt2 = digitalRead(Button2);
    byte Butt3 = digitalRead(Button3);
    Res = Butt3 * 4 + Butt2 * 2 + Butt1;
    if (Res == 7) Res = 0;
    if (Res == 6) Res = 1;
    if (Res == 5) Res = 2;
    oled.firstPage();
    do
    {
      oled.setFont(u8g_font_unifont);
      oled.setPrintPos(0, 10);
      oled.print("Button: ");
      oled.print(Res);
      oled.setPrintPos(0, 25);
      oled.print("Area: ");
      oled.print(recievedData);
    }
    while ( oled.nextPage() );
  }
  while (not((Res == 1) || (Res == 2) || (Res == 3)));

  // termination of data reception
  nRF.stopListening();
```

The stages of the vehicle working as per the code is tested and the following pictures shows the response as per the input.

- Stage 1: When the first button is given as the input the vehicle will start to move and stops in the target stop 1 and it will return back to the waiting area.

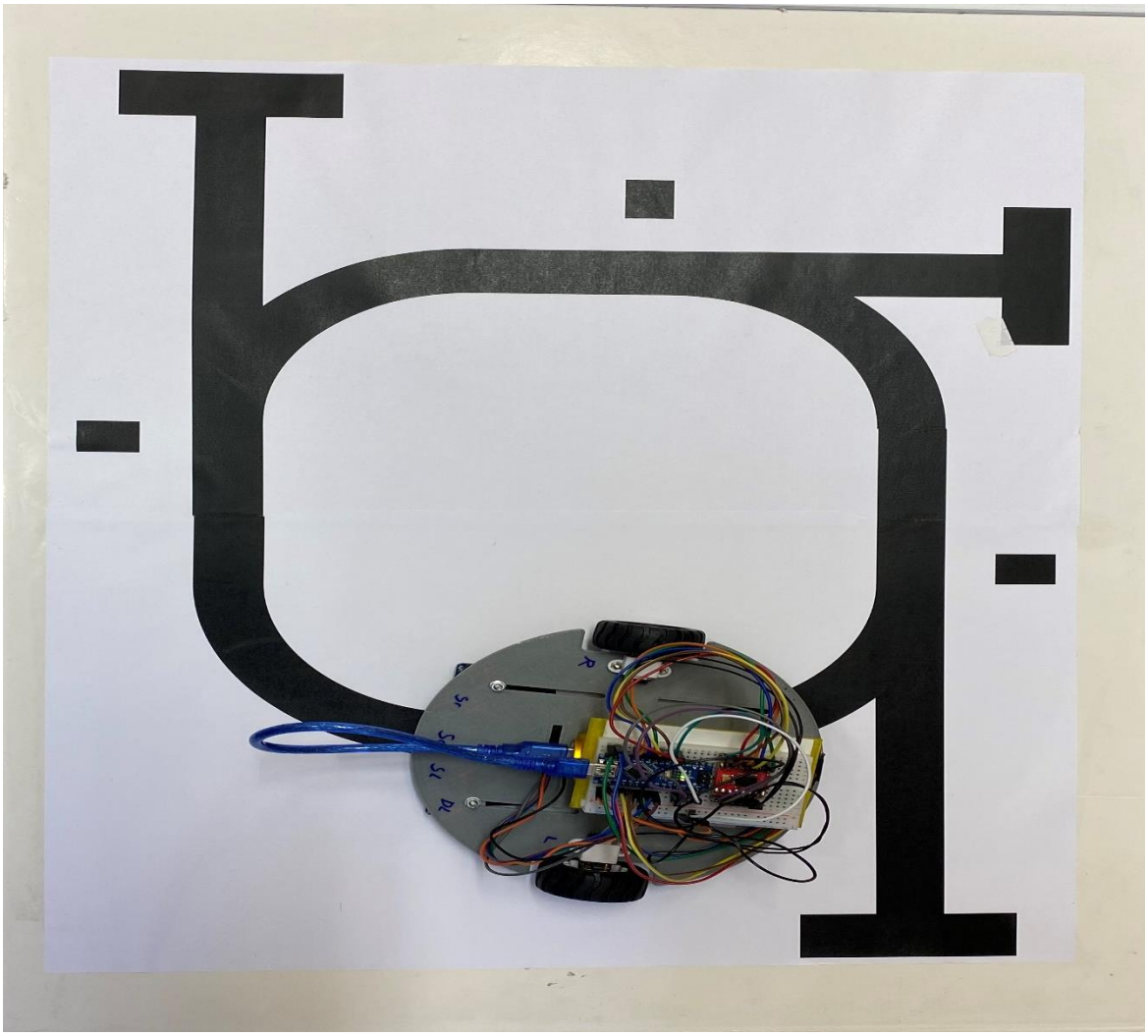

*Figure 29: Vehicle at Waiting area*

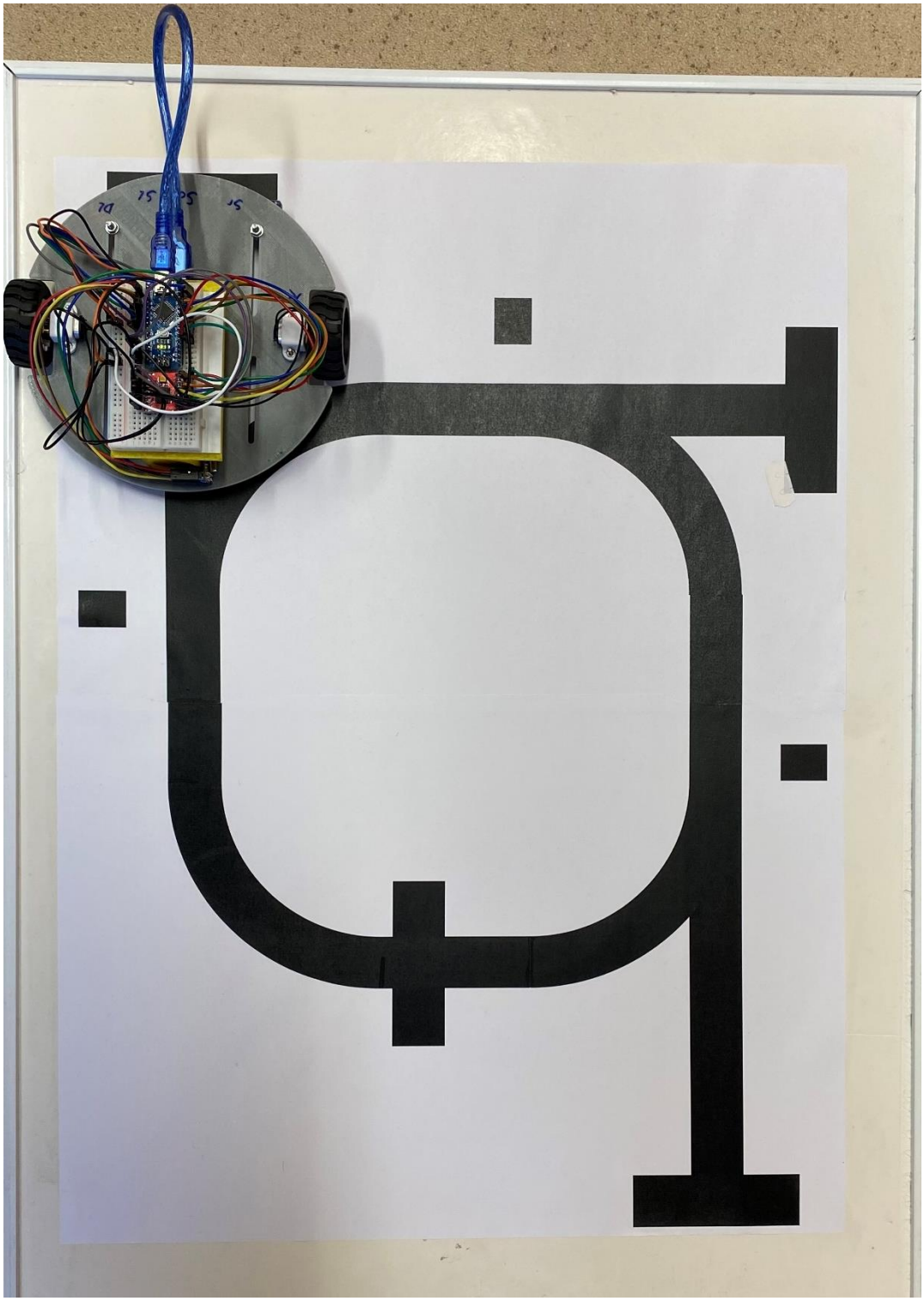

*Figure 30: Vehicle at target stop 1*

- Stage 2: When the Second button is given as the input the vehicle will start to move and stops in the target stop 2 and it will return back to the waiting area.

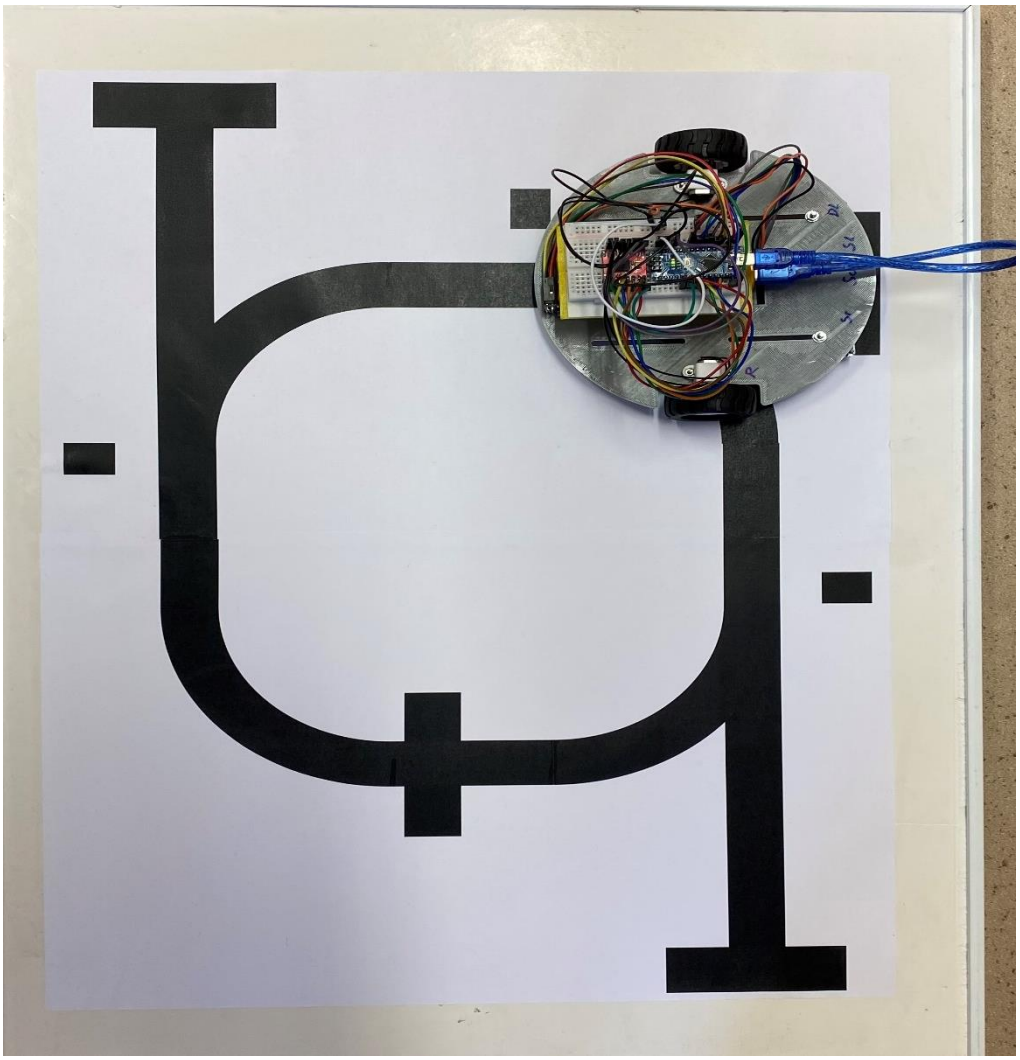

*Figure 31: Vehicle at target stop 2*

- Stage 3: When the third button is given as the input the vehicle will start to move and stops in the target stop 3 and it will return back to the waiting area.

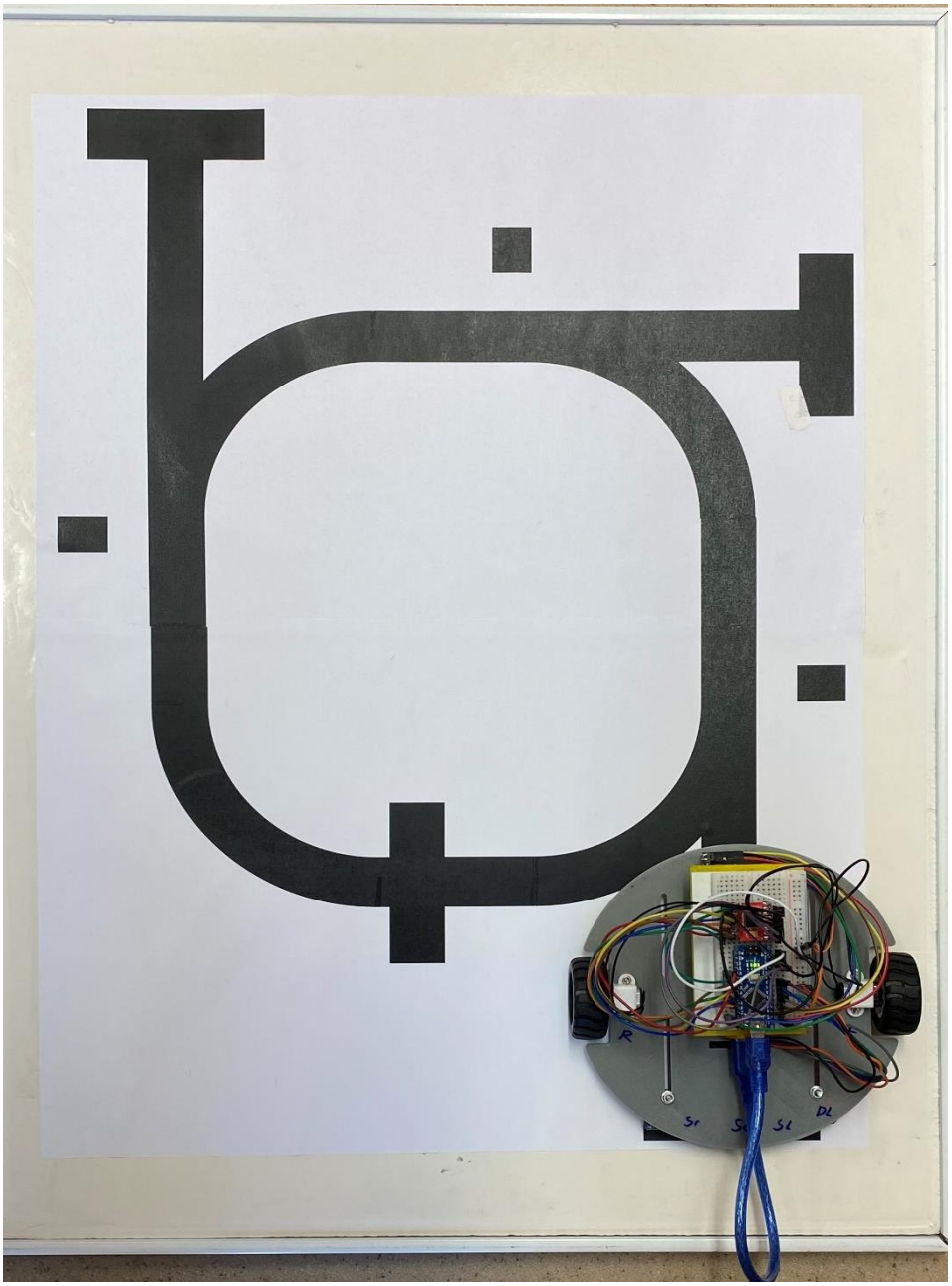

*Figure 32: Vehicle at target stop 3*

These are the operating codes and testing results of the AGV vehicle. The vehicle is working as per the codes that are created while testing.

## 7. Conclusion

In this thesis, an algorithm is created and tested successfully for a test path using the Arduino coding. With addition to this the wireless communication also implemented and tested for the different path. In this the wireless communication is designed for the centralized system type and need to change into industry 4.0. This thesis gives the knowledge about the purpose of AGV that are used in the industries and the working possibilities of AGV are learned and implemented.

In this the testing of the vehicle is done with different types of path because to check whether the vehicle follows the path as per the codes. This trial and error method helps to improve the creation of codes based on the increasing of the paths and for the implementation of the wireless communication using NRF24L01. In this the implementation of the backward movement is done based on the scenario of considering that there is no possibility of turning of vehicle in all the direction because of the less space. So, the vehicle should follows the path there only in the backward motion.

There are lot of problems that are faced during the testing process of this vehicle as mentioned above and this helps to learn about the problem solving ability from this. Still it is a basic prototyped AGV using the centralized system the improvements for this system is implementing the decentralized system which can be the advanced AGV with the communication between the vehicles which can deliver the materials to the work station in an efficient way and faster way than this type of model. This includes the communication system between the vehicles so there may require different paths and may be two vehicles can be used for the operation testing for the prototype.

## 8. References

[1] G. Ullrich, *Automated Guided Vehicle Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015. doi: 10.1007/978-3-662-44814-4.

[2] S. User, "AGV Sensors – The eyes and ears of mobile robots," *www.agvnetwork.com*. https://www.agvnetwork.com/mobile-robot-sensors-agv-amr.

[3] "Automated Guided Vehicles (AGV) Meaning & Types," *6 River Systems*, Oct. 23, 2019. https://6river.com/what-are-automated-guided-vehicles.

[4] "Forklift AGVs (Automated Guided Vehicles)," *Asseco CEIT*. https://www.asseco-ceit.com/en/agv-systems/forklift-agvs/.

[5] "Figure 1: Set with AGV towing vehicle within the system MilkRun," *ResearchGate*. https://www.researchgate.net/figure/Set-with-AGV-towing-vehicle-within-the-system-MilkRun_fig1_337876934.

[6] "JBT Conveyor Deck - AGV Solutions - JBT," *Automated Systems*. https://www.jbtc.com/automated-systems/products-and-applications/products/unit-load-agvs/conveyor-deck.

[7] "AGV Coil Handlers," *Elwell-Parker*. https://elwellparker.com/legacy-equipment/coil-handling-equipment/agv-coil-handlers.

[8] "Automated guided vehicle systems, state-of-the-art control algorithms and techniques | Elsevier Enhanced Reader." https://reader.elsevier.com/reader/sd/pii/S0278612519301177?token=7FD4ECCA4296D8F5DE8BBECD8D06CA04BE99B922E7AD05337F9EF59CD19CCAA766E80241ACF2884CAE4FC30E35EA88B8&originRegion=eu-west-1&originCreation=20220426235034.

[9] "1-s2.0-S0278612519301177-gr2_lrg.jpg (1333×1156)." https://ars.els-cdn.com/content/image/1-s2.0-S0278612519301177-gr2_lrg.jpg.

[10] "1-s2.0-S0278612519301177-gr3_lrg.jpg (1333×1124)." https://ars.els-cdn.com/content/image/1-s2.0-S0278612519301177-gr3_lrg.jpg.

[11] "Unofficial Arduino Micro Pinout Diagram - Using Arduino / Project Guidance," *Arduino Forum*, Feb. 13, 2013. https://forum.arduino.cc/t/unofficial-arduino-micro-pinout-diagram/145230.

[12] "Arduino-mega-pinout.png (1240×1753)." https://upload.wikimedia.org/wikipedia/commons/1/1c/Arduino-mega-pinout.png.

[13] “Breadboard 8.5x5.5cm (400 Holes),” *Cytron Technologies*. https://www.cytron.io/p-breadboard-8.5x5.5cm-400-holes.

[14] “Advance Line Follower robot.” http://playwithrobots.com/advance-line-follower-robot.

[15] D. Workshop, “TB6612FNG H-Bridge with Arduino - Better Than L298N?,” *DroneBot Workshop*, Dec. 15, 2019. https://dronebotworkshop.com/tb6612fng-h-bridge.

# ATTACHMENTS

Zip file of the code

Arduino.zip