**An Integrated System of Automated Guided Vehicle (AGV)**

Karam Haidar, Kerana Ngoie

**Abstract.** An automated guided vehicle (AGV) is a part of the broader field of robotics, which focuses on the design, development, and programming of robots to perform tasks autonomously or with minimal human intervention. The (AGV) is navigated along pre-defined guide paths using various guidance technologies such as floor-surface mounted magnetic tape or bars, lasers, optical sensors, and magnet/gyroscope based inertial guidance, as well as the software and hardware components to enable autonomous navigation, obstacle avoidance, and task execution, and this is what most research and algorithms in this field of robotics have included, but with separate researches, as they do not provide complete system design from A to Z, which is why this system has been designed and included the buildup procedures of hardware components and programming of (AGV) based on a 2D graphic search system, uniform grid map, static environment, real-time motion, path planning, and localization to reach the target, in contrast, the implementing a user-friendly web-based interface using Hypertext Markup Language (HTML), and creating, developing, integrating and then converting algorithms into code was one of the methodological challenges that was successfully implemented, along with a guidance system and vehicle type methodology designed for specific applications and environments. This project involves leveraging two infrared sensors (IR) sensors as the robot's eyes, interfacing with an ESP32 microcontroller to make real-time navigation decisions by using KHSFA algorithm. Additionally, the integration of a web-based control interface through (HTML) and Wi-Fi connectivity empowers users to dynamically select specific locations. An integrated system is created that includes installing hardware components and programming them with high standards and performance to carry out the tasks assigned to it via the Wi-Fi network. The novelty content creating an integrated AGV system includes installation of components and implementation of KHSFA algorithms for educational purposes. At the end of the thesis, the system interface can be accessed via smartphone or laptop to assign a task to AGV.

**Keywords:** Line tracking robot, Infrared sensors, Microcontroller, Navigation, Robotics, AGV

1. **Introduction**

The concept of line following robots dates back to the 1960s and has been used as a way to allow robots to move autonomously when processors at the time struggled to handle the data required for full autonomous movement. Collaborations between research institutions and companies specializing in robotics promise to accelerate the translation of theoretical advancements into practical, real-world solutions [1]. Line tracking robots can be used for educational purposes, allowing beginners in robotics to learn about both the software and hardware aspects of robotics. Overall, line tracking robots are versatile and widely used in different fields due to their ability to autonomously follow a line and perform specific tasks based on the line detected. In industrial settings, they excel in scenarios that demand precise navigation along predefined paths, like those in warehouses or assembly lines. Moreover, Line tracking robots have various practical applications, involving, industrial automation, educational robotics, and entertainment. These robots boast the capability to perceive the surroundings with remarkable precision, enabling them to make informed decisions in real-time. The applications of this capability span a spectrum of domains, from industrial automation [2] to healthcare [3]. A line tracking robot is an autonomous robot that detects and follows a line drawn on the floor. Line tracking robots typically use infrared sensors, ultrasonic sensors, or camera-based systems to detect and follow the line. The robot adjusts its speed and direction based on the line detected [4].

The steering angle of the robot's wheels is adjusted according to the black line detected, allowing the robot to have a smooth drive while The robot can make adjustments to its movement, such as turning left or right, based on the line detected [5]. The Navigation and Tracking Robots represent a significant advancement in technological leap, featuring robots equipped with sophisticated sensors and algorithms [6]. Through the years the technological developments (mainly in electronics and robotics) have offered AGVs several advantages over other material handling systems, such as, routing flexibility, reliability, low operating costs, unobstructed movement and easy integration with other systems. The variety in configurations of AGVs is endless and virtually any type of mobile material handling equipment can be converted to an AGV [7]. These smart cars can now make decisions that were previously made by central control units. This results in self-learning and adaptive systems, which are particularly suitable for complex, large-scale systems involving many vehicles and a high probability of vehicle interference [8].

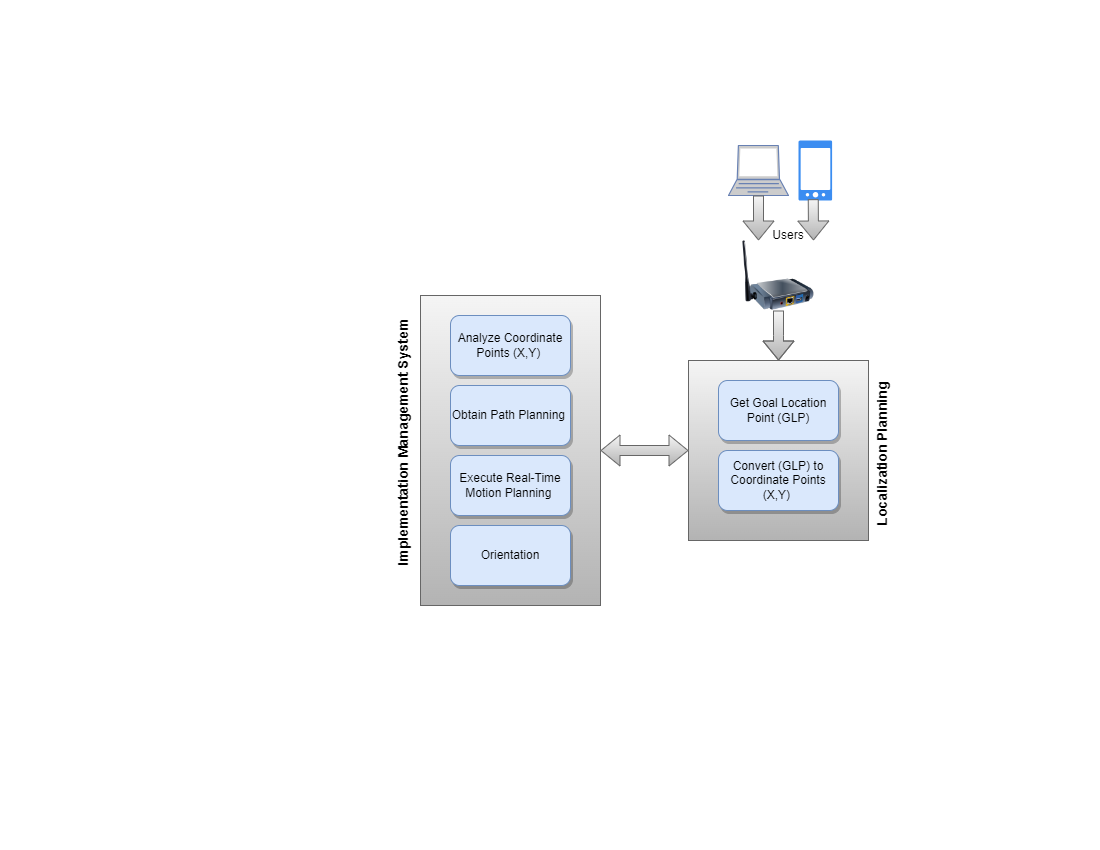
Ongoing research endeavours have focused on improving AGV decision-making algorithms, making them more efficient and versatile

The introduction of a web-based control interface, facilitated by (HTML) and wireless internet access, represents a significant advancement. Users can dynamically interact with the robot, selecting destinations on-the-fly. This not only adds a layer of sophistication to the technology but also highlights the adaptability of these robots. This represents a significant stride in the field, not only enhancing navigation accuracy but also providing users with unprecedented control and adaptability over the robot's movements [9]. Current research focus on improving the control algorithms of decision-making processes of (AGV) making them efficient and flexible.

(AGV) systems are composed of three crucial basic tasks, *localization* One of the most important things to think about in (AGV) control is localization that obtain the exact location in the map. This task, in contrast to the others, is mostly already decentralized because all the necessary software and hardware for localization are already in place. For additional control purposes, information regarding the location in the 2D map of the surroundings can be sent to nearby devices or a central computer. *Path Planning* is the generation of an obstacle free path from point A to B. Static planning involves computing a minimum collision-free path using known parameters. The known map containing obstacles is the only one used; no dynamic, time-dependent elements are taken into account. The robot can identify its open configuration space and the obstacle space by using this map. Thus, path planning can be defined as creating an obstacle-free path that connects the start point and the target point while accounting for the geometric requirements of planning algorithms, which need them to be complete. *Motion Planning* is a real-time modification planned path. Planning for static paths, in which a simple path is built with only stationary obstacles in mind. When an (AGV) is in a stalemate, it can no longer take any more actions. It is not able to travel backward or forward. Motion planning is the process of altering a static predefined path to prevent collisions and deadlocks [9].

[8]. The vast majority of the robotics work in industry, carrying out various duties like in the warehouse, and material transportation. Any sustainable approach that can lower the final cost of the product is welcome in a world where the economy is growing. This article presents researches on the Automated Guided Vehicle (AGV) control tasks for education purpose where a static environment is used, and the cells is predefined uniformly. Three techniques that determine the (AGV) behaviour are discussed: optical localization (obtaining the exact location map), path planning (generation of free path according to static environmental), and motion planning (real-time modification of a planned path). When a static environment with uniform cell mapping are used in any domain, the three techniques can be combined into a single algorithm called KHSFA.

This research is based on the effectiveness of the KHSFA algorithm, which is expected to guide the AGV to the target location at the user's command via a wireless connection and a web interface that can be accessed from any smartphone or laptop by entering an IP address. The KHSFA algorithms are also expected to give the closest path to the target and move the AGV along the path without deviation and ensure its arrival in a smooth and harmonious manner with the rest of the events. As well as responding to upcoming tasks that the user may assign to him.



*Fig.1. Workflow of Implementation System Management*

1. **Experimental/Theoretical section:**
   1. Hardware Design

The line-tracing robot consists of five basic components: two infrared (IR) sensors, a microcontroller (ESP32), four-wheel motors, a battery, and L298N motor driver. These components form the basis of a line tracking robot system as shown in the Fig. 2.

1. *Infrared Sensors (IR):*

IR sensors can detect the guidance line and determine the position of the robot as s shown in Fig. 2 (a). Their ability to perceive changes in reflected brightness allows the robot to recognize and follow the desired path with high accuracy [10]. The microcontroller can translate sensor data (IR) into information about direction, position movements, and displacement.

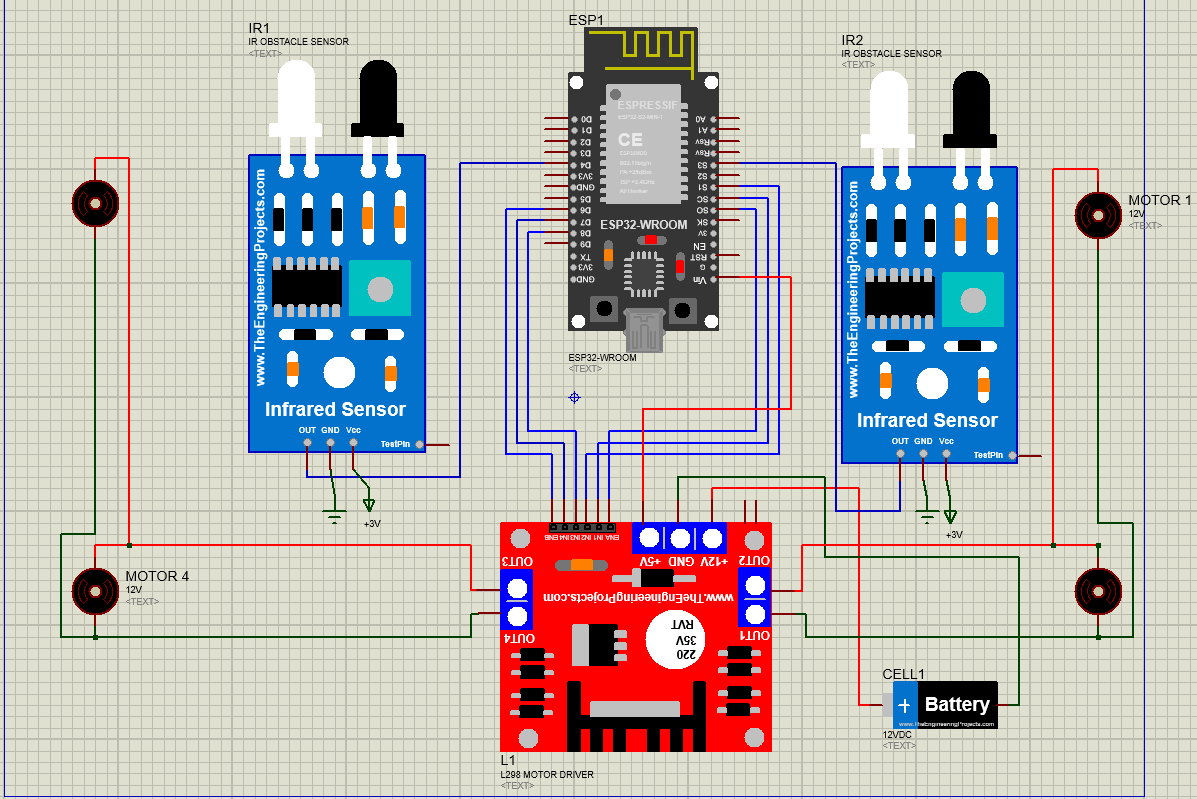
1. *Microcontroller ESP32*:

It is the core of the system, and the decision maker for the robot. As shown in Fig. 2 (b), it consists of a microprocessor chip, memory, I/O ports, and integrated Bluetooth/Wi-Fi devices [11]. The designed program can be downloaded from the PC to the console via the appropriate connection cable (USB cable) which is saved in memory as long as no other program is loaded. It processes data collected through its input ports and makes intelligent decisions based on infrared sensor data, web server data, and the robot's current location. Next, the microcontroller sends control signals to the L298N motor driver through its output ports to implement what it has decided [12].

1. *Motors and L298N Driver:*

The robot's kinetic energy is driven by four synchronous wheel motors Fig. 2 (c). The microcontroller is an important intermediate with the L298N motor driver Fig. 2 (d) which can pass high amperes to control the speed of DC motors. The movement of these actuators in the direction specified by the user places the designed robot model in the specified direction whether to move forward, stop, or perform a left or right turn [13].

|  |  |
| --- | --- |
|  |  |
| *(a)* | *(b)* |
|  |  |
| *(c)* | *(d)* |
| *Fig.2. Components of Line Tracking Robot, a) IR Sensors, b) ESP32, c) Wheel motors, and d) Driver* | |
| The above-mentioned components that are installed on the vehicle’s chassis and connected to each other via electrical wires, bear in mind the voltage and maximum amperage of each component, as shown in fig.3. Two infrared sensors are mounted under the vehicle to serve as the vehicle's eyes, they are responsible for navigating the vehicle whenever it deviates from the path, also through them the target path is determined by calculating the intersections.  The four wheels are connected in an appropriate way with the number of motor's driver inputs, as it has only two inputs, so the front and rear wheels of the right side of the vehicle are connected in parallel, as well as the two wheels of the left side, keep in view the polarity when connecting wheels in parallel. A breadboard is used to install the components on it and facilitate electrical connections, although it is preferable to use a PCB card to avoid possible errors from the breadboard and loss of the signal when the vehicle is moving and turning. A 12-volt, 2250 mA lithium battery is used via a control switch to activate/deactivate the system, care must be taken to connect the battery polarity correctly.  Calibration of commercial wheel motors:  To obtain a uniform speed that ensures the robot moves in a straight line, the wheel motors must be calibrated by adjusting the speed values for each wheel motor fed from the L298N driver. It is also necessary to improve the accuracy of the robot's rotation by adjusting the time delay when the rotation occurs, ensuring good consistency of right, left and reverse rotation. Take in to consideration about the state of charging the battery, as it may lead to unsatisfactory results. | |



*Fig.3. Circuit Diagram of Line Tracking Robot*

* 1. Software Design

The behavior of the robot is determined by the readings from the sensors, referred to the left and right sensors. Here's a breakdown of the robot's actions based on the IR conditions [14].

1. *Forward Motion:*

The robot moves forward when the two infrared sensors detect a bright condition. Synchronized action of equal speed motors drives the robot forward along the specified path.

1. *Left Turn:*

If the right IR sensor senses a bright condition while the left IR senses a dark condition, then the ESP-32 will execute a left turn function to correct the path as shown in Fig. 4 (a).

1. *Right Turns:*

Conversely, if the right IR sensor senses a dark condition and the left IR senses a bright condition, then the ESP-32 will execute a right turn function to correct the path as shown in Fig. 4 (b).

1. *Stop:*

When the two infrared sensors detect a dark condition simultaneously and the target location is reached, the ESP-32 will perform an off function to stop the robot as shown in Fig. 4 (c).

|  |  |
| --- | --- |
|  |  |
| *a)* | *b)* |
|  | |
| *c)* | |
| *Fig.4. Robot Movement Strategy, a) Should be turned Left, b) Should be turned Right and c) Stopping,* | |

Arduino software (IDE) based on C++ language program are implemented to write and upload system functions and procedure codes. The necessary libraries are downloaded to identify and manipulate the hardware components, then the codes for the functions related to guide the robot are written and uploaded, as a first step to enable the components to be checked and calibrated as mentioned previously in paragraph 2.1. These functions are as follows:

* + 1. Forward Movement Function

In the algorithm below, the inertia () function is used to optimizes motion initiation and is executed instantaneously only if the motion is started from rest, the variable (mask2) indicates that. The inertia () is identical to the Forward function codes with high speed values. The assigned speed values (35) ​​depend on the type and capacity of the battery used, as well as the accuracy of the calibration.

Void forward ()

{

if (mask2==0)

{

inertia ();

mask2=1;

}

analogWrite ( ena, 35 );

analogWrite ( enb, 35 );

digitalWrite (in1, LOW);

digitalWrite (in2, HIGH);

digitalWrite (in3, LOW);

digitalWrite (in4, HIGH);

}

* + 1. Stop Function:

Void off ()

{

mask2=0; // to trigger inertia function

digitalWrite (in1, LOW);

digitalWrite (in2, LOW);

digitalWrite (in3, LOW);

digitalWrite (in4, LOW);

}

* + 1. Turn Left Function:

In the algorithm below, the assigned speed values (250, 110) ​​and the time delay (315) depend on the type and capacity of the battery used, as well as the accuracy of the calibration.

Void Tleft90 ()

{

analogWrite (ena, 250);

analogWrite(enb, 110);

digitalWrite (in1, LOW);

digitalWrite (in2, HIGH);

digitalWrite (in3, HIGH);

digitalWrite (in4, LOW);

delay ( 315);

}

* + 1. Turn Right Function:

In the algorithm below, the assigned speed values (140, 240) ​​and the time delay (270) depend on the type and capacity of the battery used, as well as the accuracy of the calibration. When the installation and calibration of the previously mentioned initial codes are successfully completed, the line tracking codes are developed and implemented, and the related function codes are as follows:

Void Tright90 ()

{

analogWrite (ena, 140);

analogWrite (enb, 240);

digitalWrite (in1, HIGH);

digitalWrite (in2, LOW);

digitalWrite (in3, LOW);

digitalWrite (in4, HIGH);

delay ( 270);

}

* + 1. Line Tracking Function:

The line tracking function ensures that the vehicle does not deviate from its designated path, and here the optical technology is implemented, which is determining the path on the ground by drawing straight lines with a width of 2 cm or in proportion to the distance between the two IR sensors [3]. The line tracking algorithm shown below has been modified to count the number of intersections that the vehicle passes through, whether they are forward or backward. When the vehicle is moving forward, the counter increases, and when it is moving backward, it decreases.

Void compare ()

{

if (l==0 && r==0) // means IR sensors in white line

{

forward();

}

if (l==1 && r==0) // left IR in black, Right IR in white= turn left

{

Tleft();

}

else if (l==0 && r==1) // Right IR in black, left IR in white= turn Right

{

Tright();

}

if (l==1 && r==1 && mask==0) // Right IR in black, left IR in black= stop

{

delay(300);

off();

mask=1; // to stop increment (pos++), will not enter again

if (pos\_direction == 1)

{

pos--;

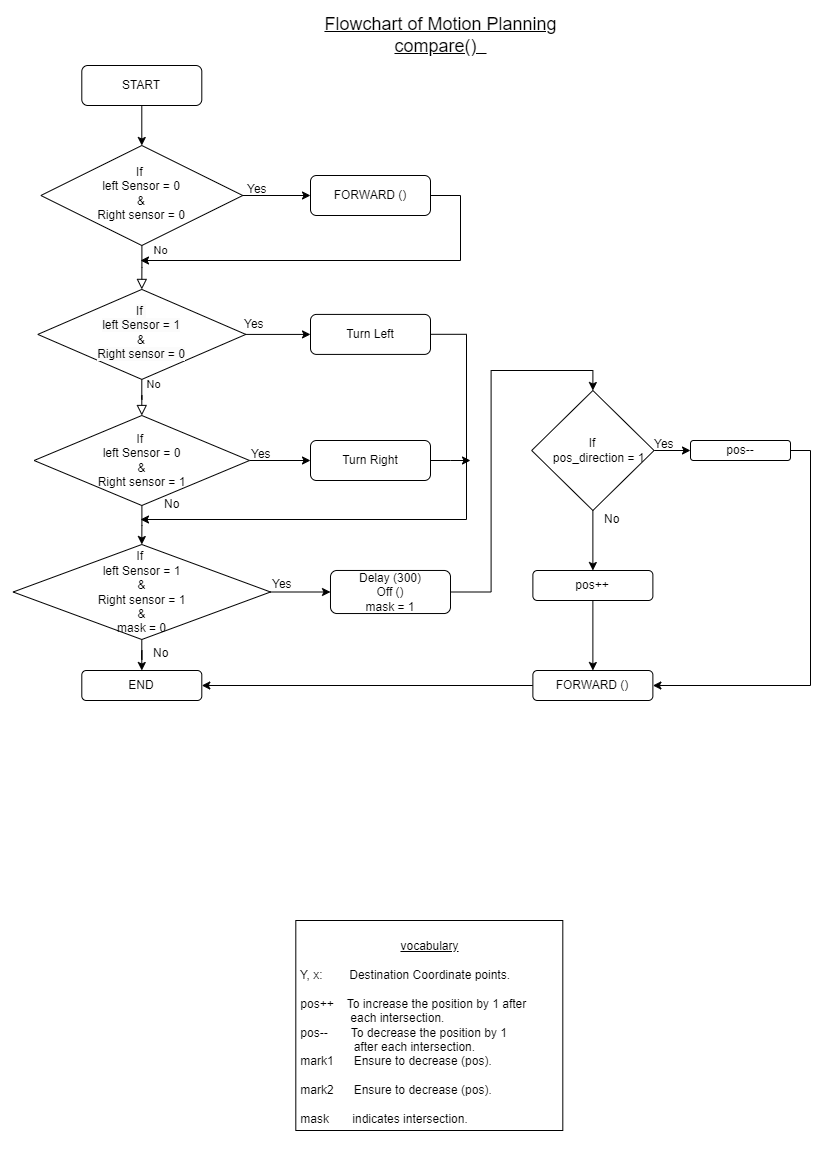
}

else pos++;

forward();

}

}



*Fig.5. Sub-KHSFA Line Tracking Algorithm Flowchart*

* + 1. Motion Planning Function:

The algorithm of function below guides the vehicle to follow line track and ensure that it reaches the target from its location, by passing and implementing the values of the coordinate points X or Y to the function via the variable (Z), where the variable (potr) indicates the direction of the vehicle.

Void exy (int Z, int potr)

{

pos\_direction = 0;

mask = 0;

loop7:

if (potr==1)

{

if (Z > pos)

{

getvalues();

compare();

}

else

{

off();

pos=0;

goto loop8;

}

}

else

{

if (Z < pos)

{

getvalues();

compare();

}

else

{

off();

pos=0;

goto loop8;

}

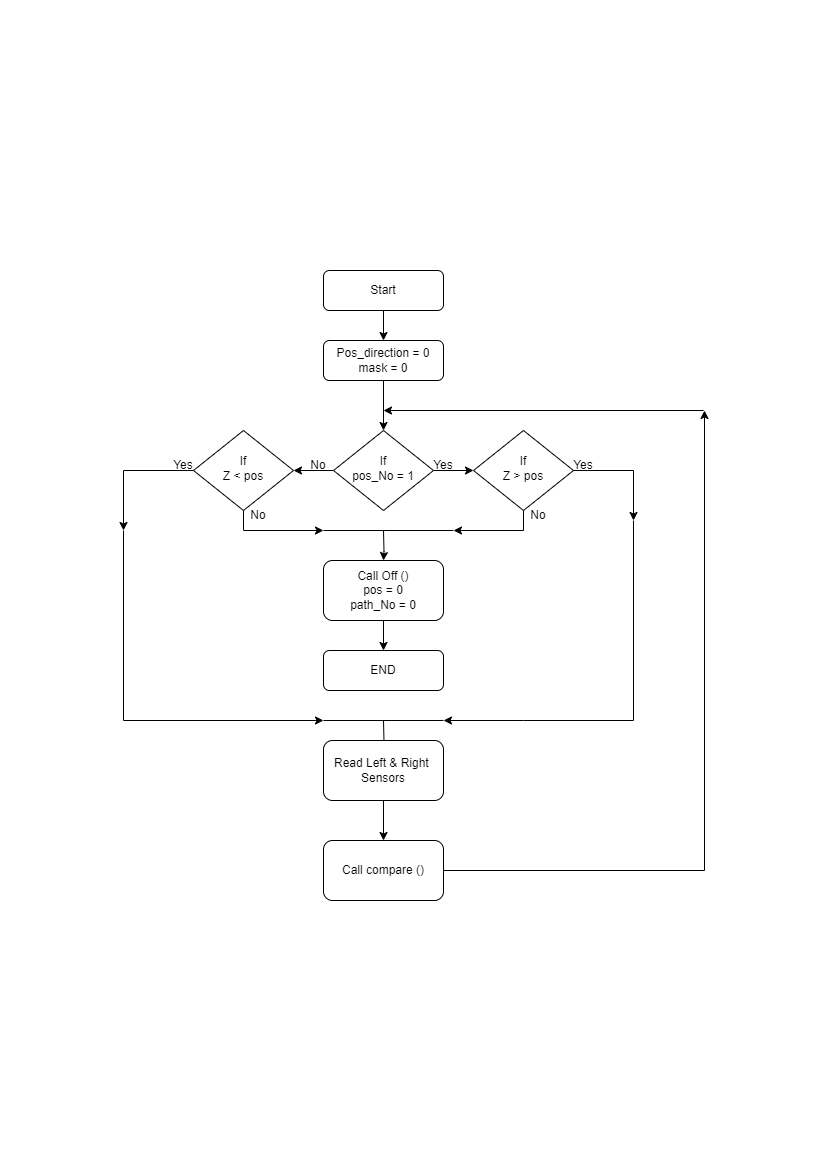
}

goto loop7;

loop8:

pos=0;

}

*Fig.6 Sub-KHSFA Motion Planning* 

* + 1. Localization and navigation Function:

The below KHSFA function determines the location of the target and the strategy for reaching it. It is important to mention one of the strategies used to complete the system, which is to direct the vehicle to a specific position when the assigned task is completed, and this position is taken as a reference for the next task. The following paragraphs provide an illustrative explanation of all the procedures followed in the algorithms.

1. *Initialization:*

- Check the server's status.

- Establish initial coordinates for the robot's position.

- Receive the desired location from the server.

1. *Localization:*

- Obtain the goal location by subtracting final location from initial location

1. Path Planning:

- Analyze coordinate points of the goal location

- When the destination coordinate points are (x > 0 and y > 0):

- Move forward on the Y-axis.

- turn right.

- Move forward on the X-axis.

- Set the orientation to reference position.

- When the destination coordinate points are (x < 0 and y < 0)

- Move backward in Y-axis by reversing the orientation.

- Set orientation to turn right.

- Move forward on the X-axis.

- Set orientation to reference position.

- When the destination coordinate points are (x == 0 and y > 0):

- Move forward on the Y-axis.

- When the destination coordinate points are (x == 0 and y < 0):

- Move backward in the Y-axis by reversing the orientation.

- Set the orientation to reference position.

- When the destination coordinate points are (y == 0 and x > 0):

- Turn right.

- Move forward on the X-axis.

- Set the orientation to the reference position.

- When the destination coordinate points are (y == 0 and x < 0):

- Turn left.

- Move forward on the X-axis.

- Set the orientation to the reference position.

- When the destination coordinate points are (x > 0 and y < 0)

- Turn right.

- Move forward on the X-axis.

- Turn right.

- Move forward on the Y-axis.

- Set the orientation with the reference position.

- When the destination coordinate points are (x < 0 and y > 0):

- Move forward on the Y-axis.

- Turn left.

- Move forward on the X-axis.

- Set the orientation with the reference position.

1. Updating Previous Location:

- Update the previous location of the robot.

1. Repeat the procedures:

- Repeat from step #2 procedures.

All the movements take consideration of the left and right sensors, either left and right sensors in black or white condition. If both left and right sensors are on the white condition, robot will continue moving forward, until both of them sense black condition, robot will stop and the intersections counter will be increased or decreased by 1 (pos) depending on the direction of the motion as shown in the flowchart Fig.5. If one of the sensor detect black line, while the other sensor in white region, robot will turn right or left based on right and left IR sensors as explained in section (2.2).

Void calc ()

{

x= nx-ox; //nx & ny denote the selected location

y= ny-oy; // ox & oy are the previous location

if (x>0 && y>0)

{

exy(y, 1); // Execute y-motion (Move forward on Y-axis)

orientation(3);

exy(x,1); // Execute x-motion (Move forward on X-axis)

orientation(4);

no=0;

}

else if (x<0 && y<0)

{

getvalues(); // To read left and right IR sensors

orientation(2); // Move in revers way

exy(y,0);

orientation(3); //Turn right

exy(x,0);

orientation(3); //Turn right

no=0;

}

else if (x==0 && y>0)

{

getvalues();

exy(y,1);

no=0;

}

else if (x==0 && y<0)

{

getvalues();

orientation (2);

exy(y,0);

orientation (2);

no=0;

}

else if (y==0 && x>0)

{

getvalues();

orientation(3);

exy (x,1);

orientation(4); //Turn Left

no=0;

}

else if (y==0 && x<0)

{

getvalues();

orientation(4);

exy(x,0);

orientation(3);

no=0;

}

else if (x>0 && y<0)

{

getvalues();

orientation(3);

exy(x,1);

orientation(3);

exy(y,0);

orientation(2);

no=0;

}

else if (x<0 && y>0)

{

getvalues();

exy(y,1);

orientation(4);

exy(x,0);

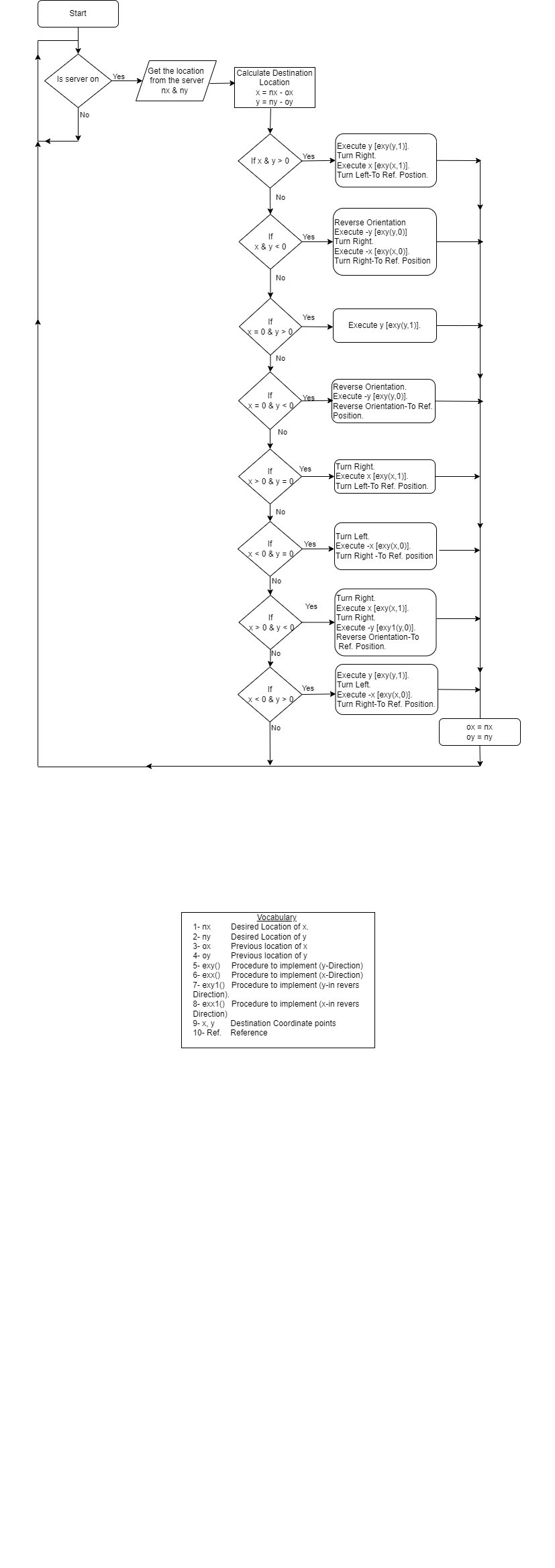
orientation(3);

no=0;

}

Ox=nx;

Oy=ny;

 }

*Fig.7. Flowchart of Localization and navigation algorithm using KHSFA*

1. Line Tracking Robot control system using KHSFA

The line tracking robot with a web-based control system provides a versatile and interactive solution for precise navigation. The web interface is designed by HTML, allowing users to control the robot's movements over Wi-Fi [15].

The KHSFA algorithm has been created based on static environments and fixed cells, target selection algorithms shown in (Fig. 5), rely on calculating the target's coordinates relative to the robot's current location (which are the paths the robot will take), while specifying an indicator of the robot's direction when each target is reached.

The workflow initiates by checking the server's status and establishing initial coordinates for the robot's position. Upon receiving the desired location from the server, the system calculates the discrepancies between the new and previous locations. A series of conditional checks then guide the robot's movements, determining whether to move forward, turn, or reverse based on the calculated differences. The sequence involves executing procedures for both the x and y directions, adjusting the orientation as needed. Subsequently, the robot updates its previous location, and the cycle continues by obtaining the next set of coordinates from the server. This iterative process, governed by precise conditions, orchestrates the robot's navigation to specified points, ensuring a dynamic and responsive movement strategy.

1. Results and Discussions

The results of the experiments show the smoothness in the movement of the robot’s among the specified locations, and its arrival at the target accurately. It can also be observed that it remains firmly on the track while moving. It is worth noting that its response to commands sent from the web interface is very fast, and no system downtime was observed during the testing procedures. The system may be simple and lack some safety measures such as deadlock and avoiding obstacles, and it does not discuss the path planning problems, which will be worked on in future research. However, it is considered an integrated and useful system for use in certain environments and may help in educational purposes, where these fields are lacking to a simple and integrated AGV system.

The project's innovation lies in the successful resolution of challenges related to fine-tuning commercial motors, implementing a user-friendly web-based control system, and overcoming algorithmic complexities by combining robot technique tasks into a single algorithm for target determination based on a uniform environment and fixed cell mapping. By achieving uniform motor calibration, precise rotation control, and integrating a dynamic web-based control interface, the robot stands out for its capability to respond to user-specified destinations.

The project is successfully implemented by using KHSFA algorithm. The robot's movement is based on real-time decisions made by the microcontroller using data from IR sensors, allowing it to detect and follow lines drawn on the floor, and localized as per predefined location coordinate points. The line tracking robot project overcame some challenges in design and implementation, such as: Calibration of commercial motors to obtain a uniform speed that ensures the robot's movement in a straight line, by calibrating and adjusting the speed values ​​for each motor fed from the ESP32 and L298N Driver. Improve the robot's rotation accuracy by adjusting the time delay when turning, ensuring good consistency for right, left and reverse turns. Taking into account the battery charge state. The algorithm for determining the target shown in Fig.5 (The location of the robot’s arrival) chosen by the user was one of the challenges in designing the program and may not be the best, but it is the most appropriate and simplest given the available capabilities. The microcontroller interprets the selected point, calculates the required movements in terms of X and Y coordinates, the robot performs forward movements, turns, and stops based on the calculated values to direct itself to the destination location.

There are few review papers on AGV systems. However, they concentrate on only limited parts of the problem or are not up to date. Moreover, they ignore some areas such as idle-vehicle positioning and battery management

1. Conclusions

A rule-based (AGV) was implemented, and employed a navigation algorithm using KHSFA. The robot is represented by two IR sensors, interact with the ESP32 microcontroller, facilitating real-time decision-making based on detected line conditions to incorporate forward movement, right, left, and reverse turns, as well as a complete stop. To enhance user interaction, a web-based control interface using (HTML) over Wi-Fi connectivity allowed users to dynamically select specific locations for the robot to navigate, showcasing its adaptability. Furthermore, the advantage of this system is that it is reprogrammable, so any upgrade can be performed simply by uploading the update-software, and it is also suitable for innumerable other applications. The other advantages of this algorithm is quick-responsive and implementation, and less memory consumption. The research on (AGV) control tasks for intelligent warehouses or supermarkets with specified and fixed cells in a static environment is presented in this article. The three methods for determining the behavior of (AGVs): motion planning, path planning, and optical localization are combined into a single KHSFA algorithm. Any domain that uses a static environment with fixed cell mapping can combine the three methods into a single algorithm known as KHSFA. When compared to other algorithms employed in this sector, the findings demonstrate that KHSFA has more reasonable community numbers, higher modularity, and superior accuracy.

References

1. Agostini, A., C. Torras, and F. Wörgötter, *Efficient interactive decision-making framework for robotic applications.* Artificial Intelligence, 2017. **247**: p. 187-212.

2. Ma’arif, A. and A.A. Nuryono. *Vision-based line following robot in webots*. in *2020 FORTEI-International Conference on Electrical Engineering (FORTEI-ICEE)*. 2020. IEEE.

3. Antony, M., et al. *Design and implementation of automatic guided vehicle for hospital application*. in *2020 5th International Conference on Communication and Electronics Systems (ICCES)*. 2020. IEEE.

4. Baumgartner, E.T. and S.B. Skaar, *An autonomous vision-based mobile robot.* IEEE Transactions on Automatic Control, 1994. **39**(3): p. 493-502.

5. Punetha, D., N. Kumar, and V. Mehta, *Development and applications of line following robot based health care management system.* International Journal of Advanced Research in Computer Engineering & Technology (IJARCET), 2013. **2**(8): p. 2446-2450.

6. Mugure, M.A., *DESIGN AND FABRICATION OF AN AUTONOMOUS LINE FOLLOWER ROBOT CAPABLE OF PICKING AND DROPPING OBJECTS FROM ONE POINT TO ANOTHER*. 2019, Doctoral dissertation, Kenyatta University.

7. Kaloutsakis, G., N. Tsourveloudis, and P. Spanoudakis. *Design and development of an automated guided vehicle*. in *IEEE International Conference on Industrial Technology, 2003*. 2003. IEEE.

8. Le-Anh, T. and M. De Koster, *A review of design and control of automated guided vehicle systems.* European Journal of Operational Research, 2006. **171**(1): p. 1-23.

9. De Ryck, M., M. Versteyhe, and F. Debrouwere, *Automated guided vehicle systems, state-of-the-art control algorithms and techniques.* Journal of Manufacturing Systems, 2020. **54**: p. 152-173.

10. Hasan, K.M., et al. *Sensor based autonomous color line follower robot with obstacle avoidance*. in *2013 IEEE Business Engineering and Industrial Applications Colloquium (BEIAC)*. 2013. IEEE.

11. Ahmed, A.S., H.A. Marzog, and L.A. Abdul-Rahaim, *Design and implement of robotic arm and control of moving via IoT with Arduino ESP32.* International Journal of Electrical & Computer Engineering (2088-8708), 2021. **11**(5).

12. Alvarado, I., et al. *Learn control building your own line follower robot at home*. in *2022 European Control Conference (ECC)*. 2022. IEEE.

13. Ariyansyah, Q. and A. Ma'arif, *DC Motor Speed Control with Proportional Integral Derivative (PID) Control on the Prototype of a Mini-Submarine.* Journal of Fuzzy Systems and Control, 2023. **1**(1): p. 18-24.

14. Shuqair, M., et al., *LINE TRACKING ROBOT.* 2007.

15. Aroca, R.V., R.Q. Gardiman, and L.M.G. Gonçalves. *Web-based robot programming environment and control architecture*. in *2012 Brazilian Robotics Symposium and Latin American Robotics Symposium*. 2012. IEEE.