Automatic Guided vehicle (AGV) with Solar Power

[A.] A.JYOTHI [B.] B.Jaswanth, [C.] A.Vivek, [D.] A.Vamsi, [E.] G.Dayasagar, [F.] G.Ashok
[A] *Assistant Professor Wellfare Institute Of Science Technology & Management*
[B,C,D,E,F] *Wellfare Institute of Science Technology & Management*

***Abstract*-Line follower is a smart autonomous robot that detects or follows a visible line embedded in the ground and guides itself. The trail is +predetermined and can be selected with a high contrast color or with a black line visible on the trail surface. Infrared sensors are used to detect these lines. Typically speaking, the area unit of the infrared sensors is used to locate the path that the robot has to follow.**

***Index* – Introduction, Literature survey, Components used, Architecture, Result, Conclusion.**

1. INTRODUCTION

A Line Follower Robot is an autonomous mobile robot designed to follow a predefined path, typically marked by a contrasting line in the surface. The project demonstrates the integration of renewable energy with embedded systems to develop an integration and self-sustaining robotic system. Line follower robots have widespread applications in industrial automation, material handling, and smart transportation, making them an essential component of modern robotic technology.

1.1 AIM OF TNE PROJECT:

The aim of this project is to design and develop an autonomous Line Follower Robot that can efficiently navigate a predefined path using an IR sensor array, while incorporating renewable energy for sustainable operation.

1.2 SCOPE OF THE PROJECT:

1. The robot can follow a predefined path using an IR sensor array for real time line detection.

2. Data on robot performance and movement can be collected and analyzed for optimization.

3. A solar panel is incorporated to reduce reliance on external charging and promote renewable energy use.

2. LITERATURE SURVEY

The development of the Line Follower Robots has been widely explored in the fields of robotics, automation, and artificial intelligence. Several research papers and projects have contributed to the understanding and advancement of line following robots. This literature survey reviews previous works and technological advancements related to autonomous navigation, motor control, renewable energy integration, and IOT connectivity.

2.1 PROBLEM STATEMENT:

In industrial automation, logistics, and smart transportation, autonomous robots are increasingly being used to enhance efficiency and reduce human intervention. However, existing line follower robots face several limitations, including limited connectivity, energy inefficiency, and poor adaptability to complex environments.

Lack remote monitoring and control as they do not support IOT – based connectivity.

Consume excessive power due to reliance solely on battery-powered systems, leading to frequent recharging.

Struggle with advance path navigation, as they rely on basic IR sensor arrays without dynamic adjustments.

2.2 PROPOSED SYSTEM:

To overcome the limitations of existing line follower robots, the proposed system integrates IOT connectivity, energy efficiency, and optimized motor control to enhance autonomous navigation and sustainability. This system is designed to be smart, remotely accessible, and energy-efficient, making it suitable for industrial automation, warehouse logistics, and smart transportation

The overcome uses an IR sensor array to accurately detect and follow a predefined path.

The ESP8266 Node MCU microcontroller processes sensor data and controls motor movement.

The sensors provide real-time feedback to ensure precise movement, even on curved or interrupted paths.

Enables Wi-Fi-based remote monitoring and control, allowing real-time data transmission.

The L298 motor driver is used to control 1kg 60RPM motors, ensuring smooth and accurate navigation.

Supports speed control and directional movement, optimizing power consumption.

A solar panel is incorporated to supplement the power supply, reducing

Improves energy efficiency and extends operational time without frequent recharging.

The system is designed to be modular and scalable, allowing future upgrades such as obstacle detection.

3 .COMPONENTS USED

3.1 Hardware Details:

1. l298 motor driver

2.1kg 60 rmp

3. 12v battery

4. solar panel

5. esp8266 node mcu

6. IR sensor array

7. jumper wires

3.2 Software Details:

1. arduino ide

2. embdded 'c'

3.2.1 ARDUINO IDE:

The Arduino IDE (Integrated Development Environment) is a cross-platform software application designed for writing, compiling, and uploading code to Arduino microcontroller boards. It serves as the primary tool for developers, hobbyists, and students to create interactive electronics projects. With its intuitive interface and straightforward workflow, the Arduino IDE has become popular worldwide for prototyping and developing both simple and complex systems. The Arduino IDE stands out for its simplicity, flexibility, and extensive support network. It is ideal for beginners learning programming and electronics, as well as professionals developing advanced systems, making it a versatile tool in the world of embedded systems and IOT.

3.2.1.1 KEY COMPONENTS:

1. Code Editor:

The IDe3 provides a text editor where users write programs, known as "sketches," using a simplified version of the C++programming language.

It features syntax highlighting, auto-indentation, and bracket matching, enhancing readability and reducing coding errors.

2.Compiler:

The built-in compiler translates human-readable code into machine language, ensuring compatibility with the microcontroller on the Arduino board.

3.Uploder:

Using a USB connection, the

uploader transfers the complied code from the computer to the Arduino board, enabling the microcontroller to execute the programmed instructions.

4. Serial Monitor and Plotter:

The Serial Monitor allows users to communicate with the Arduino board in real time, facilitating debugging and data exchange.

The Serial plotter visualizes data from the board, making it easier to analyze sensor readings and other variables.

5. Library Manager:

A comprehensive library manager provides access to pre-written code libraries that simplify the use of sensors, displays, motors, and other hardware components.

3.2.1.2 FEATURES:

Cross-Platform Compatibility: The Ardunio IDE runs on windows, macOS, and Linux.

Open Source: As open-source software, it encourages collaboration, customization, and community contributions.

User-Friendly Interface: Its straightforward design makes it accessible to beginners while still offering advanced features for experienced developers.

Extensive Community Support: A vast online community shares code, tutorials, and troubleshooting tips, making it easier to learn and troubleshoot.

3.2.1.3 WORKING:

Writing Code: Users write sketches using the code editor, often starting with built-in examples.

Compiling Code: The IDE compiles the sketch, checking for syntax errors and ensuring compatibility with the selected Arduino board.

Upload Code: With a single click, the compiled code is uploaded to the Arduino board, enabling it to perform the desired tasks.

Testing and Debugging: The serial Monitor and plotter help test the code, observe output data, and identify issues.

3.2.1.3 USAGE:

Ideal for prototyping electronics projects, robotics, IoT devices, and automation systems.

Widely used in education, hobbyist projects, and professional prototyping due to its ease of use and versatility.

3.2.1.4 Applications of Aruino IDE:

Education: Widely used in schools and universities to teach programming and electronics.

Prototyping: Essential for developing prototypes of IoT devices, robots, and home automation systems.

Hobbyist Projects: Popular among makers for creating interactive art, DIY gadgets, and wearable technology.

3.2.2 EMBEDDED C

Embedded C is a specialized extension of the C programming language designed for programming embedded systems. Embedded systems are microcontroller-based devices that perform dedicated functions, such as medical devices, automotive systems, home appliances, and IoT devices. Unlike general-purpose computers, embedded systems operate with limited resources, making efficiency and reliability crucial. Embedded C helps developers write efficient and optimized code that directly interacts with hardware components. Embedded C plays a critical role in the development of embedded systems by combining the simplicity of C with hardware-specific functionalities. Its efficiency, low-level hardware control, and widespread industry adoption make it essential for designing devices that are reliable, responsive, and optimized for specific tasks.

3.2.2.1 CHARECTRISTICS:

Hardware-Oriented: Embedded C allows direct manipulation of hardware components such as input/output ports, timers, and memory.

Efficiency and Speed: Programs are optimized to use minimal memory and processing power, essential for devices with limited resources.

Real-Time Operation: Embedded systems often require real-time performance, where tasks must be executed within strict time limits.

Portability: Embedded C code can be adapted to different microcontrollers with minimal changes, increasing its flexibility.

Low-Level Access: The language provides low-level access to system memory and peripheral registers, allowing precise control over hardware.

3.2.2.2 STRUCTURE:

Preprocessor Directives: Include header files specific to the microcontroller, such as <avr/io.h> for AVR microcontrollers.

Global Variables: Store data that needs to be accessed throughout the program.

Main Function: The main () function contains the core logic of the program and often runs in an infinite loop to maintain continuous operation.

Peripheral Initialization: Configure hardware peripherals like timers, communication interfaces, and input/output ports.

Interrupt Service Routines (ISRs): Handle hardware interrupts that respond to external events.

3.2.3 WORKING:

The working of Embedded C involves writing code that directly interacts with hardware components of embedded systems, such as microcontrollers, sensors, and actuators. Unlike standard C programming, Embedded C is designed to run on hardware with limited resources, ensuring efficiency, reliability, and real-time performance. The process typically follows these steps:

1. Writing the Code:
Developers write the program (called firmware) using Embedded C, which includes specific libraries and syntax for hardware interaction.
The code is usually structured with initialization routines, an infinite loop for continuous operation, and interrupt service routines (ISRs) to handle external events.

2. Compiling the Code:
The written code is compiled using a cross-compiler, which translates the human-readable C code into machine code (binary format) that the microcontroller understands.
The compiler is specific to the target microcontroller architecture (e.g., AVR-GCC for AVR microcontrollers or Keil C51 for 8051 microcontrollers).

3. Linking and Optimization:
The linker combines different code modules into a single executable file, optimizing the code to minimize memory usage and maximize performance.
Optimization is crucial since embedded systems often have limited memory and processing power.

4. Generating the Hex File:
The final output of the compilation process is a HEX file, which contains the machine code in hexadecimal format.
This file is ready to be uploaded to the microcontroller.

5. Uploading the Code to the Microcontroller:
The HEX file is transferred to the microcontroller's flash memory using a programmer device or a USB connection.
For example, in Arduino, this process is handled by the Arduino IDE using a built-in boot loader.

6. Executing the Program:
Once uploaded, the microcontroller reads the machine code from its memory and executes the instructions continuously.
The main function typically runs in an infinite loop to ensure the system responds continuously to inputs and performs the desired tasks.

7. Real-Time Interaction with Hardware:
The program directly interacts with hardware components using peripheral registers, input/output ports, and communication interfaces (e.g., SPI, I2C, UART).
Interrupts allow the system to respond immediately to external events, ensuring real-time performance.

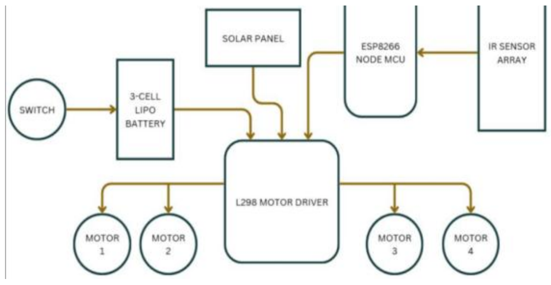
3.2.2.4 Real-Time Execution and Interrupts:
Embedded systems often need to respond to external events within strict time limits.
Interrupts allow the microcontroller to pause the main program and execute specific code when an external event occurs (e.g., a button press or sensor signal).

3.2.5 ADVANTAGES:
Simplicity and Readability: Easy to learn and read, especially for those familiar with standard C.
Efficiency: Generates compact, fast-executing code suitable for resource-limited systems.
Portability: Code can be reused across different microcontrollers with minimal modifications.
Direct Hardware Control: Provides precise control over hardware components, enabling efficient system design.
Wide Industry Adoption: Used extensively in automotive, healthcare, consumer electronics, and IOT applications.

3.2.6 APPLICATIONS:
Automotive Systems: Engine control units (ECUs), anti-lock braking systems (ABS), and airbags.
Medical Devices: Pacemakers, blood pressure monitors, and ventilators.
Consumer Electronics: Microwave ovens, washing machines, and smart home devices.
Industrial Automation: Motor control, process monitoring, and robotics.
IOT Devices: Sensors, actuators, and wireless communication modules.

4. ARCHITECTURE

BLOCK DIAGRAM:



4.2 CIRCUIT DIAGRAM:

4.3 WORKING:

The Line Follower Robot follows a predefined path using an IR sensor array, controls its movement via an L298 motor driver, and operates on a renewable power source (solar panel and Li-Po battery).

Power Supply & Management:

The system is powered by a 3-cell Li-Po battery and a solar panel.

The solar panel provides additional energy to charge the battery, making the robot energy-efficient.

A switch is included to manually turn the system on or off.

Sensor-Based Line Detection:

The IR sensor array consists of multiple IR sensors that detect the contrast between the path (black line) and the surrounding surface (white background).

Each IR sensor outputs a high (1) or low (0) signal based on whether it detects a line.

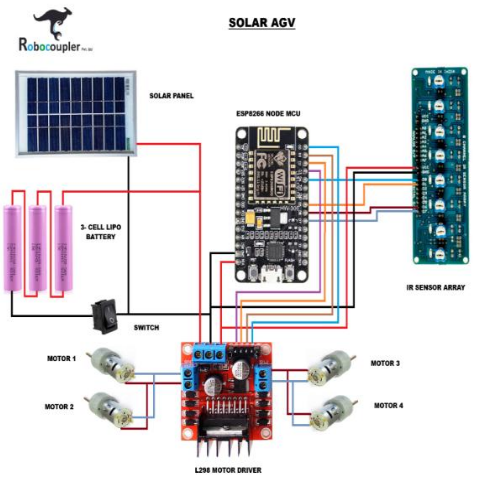
The ESP8266 Node MCU processes these signals to determine the robot's position relative to the line.

ESP8266 Node MCU Control Unit:

The ESP8266 Node MCU acts as the main processing unit.

It takes input from the IR sensor array and decides how the motors should move.

It sends control signals to the L298 motor driver to regulate the movement of the four motors.



Motor Control Using L298 Driver:

The L298 motor driver receives commands from the ESP8266 Node MCU to drive the motors.

It controls four motors (M1, M2, M3, M4) by adjusting speed and direction.

The robot moves forward, turns left, or turns right depending on the sensor input.

Robot Movement Logic:

When the line is centered:

The IR sensors detect the black line in the middle.

The ESP8266 sends a signal to move the motors forward.

When the robot drifts left:

The right-side sensors detect the line.

The ESP8266 adjusts the left motors to slow down and the right motors to speed up, making the robot turn right to correct its path.

When the robot drifts right:

The left-side sensors detect the line.

The ESP8266 slows down the right motors and speeds up the left motors, making the robot turn left.

If the line is lost:

The robot stops or moves in a predefined search pattern to find the line again.

IOT & Wireless Monitoring (ESP8266):

The ESP8266 Node MCU can send real-time data via Wi-Fi, allowing remote monitoring.

It can be connected to a cloud or mobile app for performance tracking and troubleshooting.

5.RESULTS

1. The IR sensor array accurately detected and followed the predefined path.
2. The robot adjusted its speed and direction smoothly when making turns or correcting its position.
3. The ESP8266 NodeMCU processed sensor data efficiently, ensuring real-time navigation.
4. The L298 motor driver effectively controlled the 1kg 60RPM motors, ensuring precise movement.

5. The motors responded well to directional commands, providing smooth acceleration and deceleration.
6. The solar panel provided additional power to the 12V Li-Po battery, extending operational time.
7. The robot operated for a longer duration without frequent battery recharging.
8. Power consumption was optimized, making the system more sustainable.
9. The ESP8266 NodeMCU successfully connected to a Wi-Fi network, enabling real-time monitoring.
10. The robot's status and sensor data were transmitted wirelessly, allowing remote tracking and troubleshooting.

6.CONCLUSION

The line follower robot successfully demonstrated efficient autonomous navigation using an IR sensor array, ESP8266 NodeMCU, and L298 motor driver. The integration of a solar panel with a 12V Li-Po battery significantly improved energy efficiency, making the system more sustainable and self-sufficient. The ESP8266 NodeMCU enabled IoT-based remote monitoring and control, allowing real-time tracking and potential automation in various industries.

The results confirmed that the robot is accurate, reliable, and adaptable for industrial automation, logistics, and educational purposes. The use of renewable energy and wireless communication makes it a cost-effective and scalable solution for modern automation needs.

With further enhancements such as AI-based navigation, obstacle detection, and GPS integration, this robot can be deployed in smart warehouses, automated transportation, and industrial material handling. The project showcases the potential of IoT-driven, solar-powered robotics in shaping the future of automation and sustainability.

7.REFERENCES

[1] Conceptual Details of Track Follower: Advances in Robotics: FIRA RoboWorld Congress 2009, Incheon,-Page 69 Jong-Hwan Kim, Shuzhi Sam Ge, Prahlad Vadakkepat -2009 -322 pages.
[2] Concept of Line follower Robot: Evolutionary swarm robotics: evolving self-organisingbehaviours -Page 164Vito Triennia-2008 -189 pages
[3] www.projectdesignline.com/howto/207800773
[4] http://en.wikipedia.org/wiki/Resistor
[5] http://en.wikipedia.org/wiki/Capacitor
[6] Simple Line Follower.www.societyofrobots.com/member_tutorials/node/62
[7] http://www.datasheetarchive.com/
[8] http://en.wikipedia.org/wiki/Dcmotor