DC Motor Control Using Controller Area Network (CAN) Communication

Design and Implementation of a Closed-Loop Control System with Real-Time Monitoring

[1]Shravani Milind Devare, [2]Vaishnavi Sunil Patil, [3]Ishwari Basavraj Saddalgi, [4]Dr Sunita S. Lokhande

[1], [2], [3]Student and, [4]Professor

[1], [2], [3], [4], Department of Electronics and Telecommunication Engineering

[1], [2], [3], [4], Sinhgad College of Engineering, Pune, Maharashtra, India

[1]shravanidevare.scoe.entc@gmail.com, [2]vaishnavipatil.scoe.entc@gmail.com,

[3]ishwarisaddalgi.scoe.entc@gmail.com, [4]sslokhande.scoe@sinhgad.edu

***Abstract*: DC motors are commonly used in industrial automation due to their simplicity, reliability, and rapid control. As the demand for remote monitoring and precise motor control in distributed systems is increasing, communication-based control designs have become vital. This paper presents the design and implementation of a GUI-based closed-loop DC motor control system using Controller Area Network (CAN) communication. The proposed system allows real-time speed and direction control of a DC motor through a graphical user interface while continuously monitoring motor speed using an encoder feedback mechanism. A proportional-integral-derivative (PID) control algorithm is implemented to maintain the desired motor speed. The CAN protocol confirms reliable, noise-resistant data transmission between control nodes, making the system suitable for industrial environments. The results show that the motor reaches the required speed, runs steadily, and can be controlled from the GUI successfully.**

Index Terms: DC motor control, CAN communication, closed-loop control, PID controller, encoder feedback, graphical user interface, real-time monitoring, embedded systems.

I. Introduction

In modern industrial automation systems, accurate control of motor speed and direction plays a vital role. It is highly required in applications like robotics, material handling, conveyor systems, and machine tools. Traditional motor control methods require manual adjustments and local monitoring. This limits flexibility and efficiency in automated environments.

With the advancement of embedded systems and industrial communication protocols, remote monitoring and motor controlling have become essential for improving system reliability and operational convenience. Controller Area Network (CAN) communication is widely used in industrial applications due to its high reliability, noise immunity, and real-time data transfer capability.

This project presents a GUI-based closed-loop DC motor control system using CAN communication. The motor speed and direction are controlled remotely through a graphical user interface while real-time speed feedback is obtained using an encoder. A PID-based control algorithm is implemented to minimize speed error and maintain stable motor performance.

The proposed system provides an efficient, reliable, and user-friendly solution for motor control in industrial automation environments.

II. Objective Of The Study

The objective of this study is to design and implement a DC motor control system using Controller Area Network (CAN) communication and a graphical user interface (GUI). The system aims to achieve accurate control over the motor speed and direction through encoder feedback and a closed-loop control mechanism.

Another objective of this work is to develop a reliable communication framework which allows motor control commands to be transmitted through the CAN network while enabling real-time monitoring of motor performance through the GUI. The PID-based control algorithm is intended to maintain the desired motor speed and improve system stability.

The study also aims to show the effectiveness of integrating embedded controllers, CAN communication, and GUI-based monitoring for automation and industrial control applications.

III. Scope Of Study

This study focuses on the development and evaluation of a DC motor control system integrated with CAN communication and a graphical user interface. The system allows users to control motor speed and direction while monitoring actual speed using encoder feedback.

The work is limited to the controlling and monitoring of a single DC motor operating under laboratory conditions. The study evaluates the performance of the PID control algorithm in maintaining stable motor speed and ensuring accurate response to user commands provided through the GUI.

The implementation is restricted to a wired CAN communication network and does not consider wireless communication systems or multi-motor control architectures here. Future work may extend this system to support multiple motor nodes, advanced control algorithms, and large-scale industrial automation systems.

IV. System Architecture

A. Block Diagram of the Proposed System

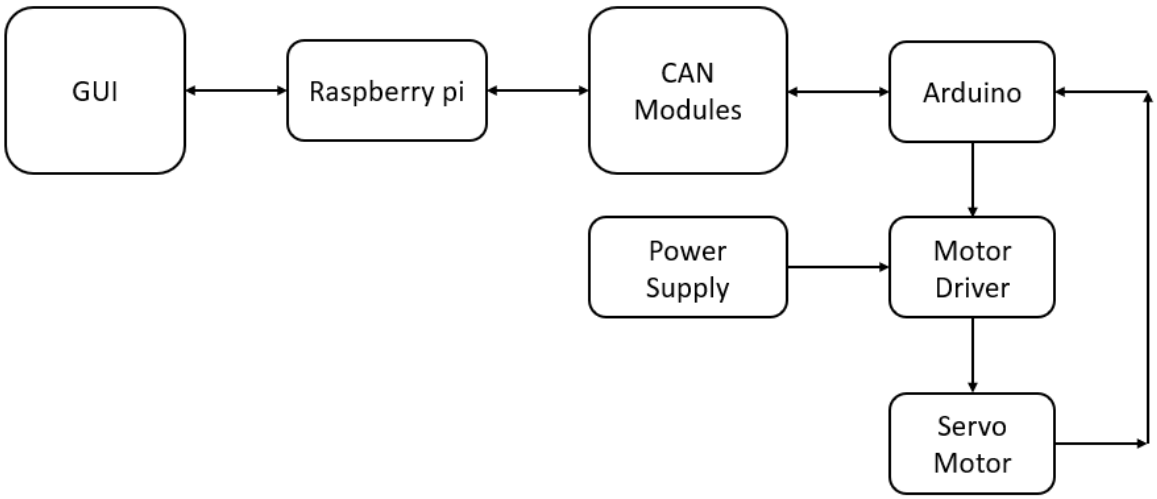

Figure 1 Proposed system of the block diagram

The Arduino UNO receives the CAN messages via the MCP2515 module and processes the received speed command. The desired speed is compared with the actual motor speed obtained from the encoder feedback. Based on the speed error, a PID control algorithm is executed in the Arduino to adjust the PWM signal applied to the L298N motor driver. This closed-loop control mechanism ensures that the motor reaches and maintains the required speed.

B. Hardware Components

1. Raspberry Pi 3 Model B+

The Raspberry Pi acts as the host system responsible for running the graphical user interface and communicating with the CAN network. It sends motor control commands such as desired speed and direction to the Arduino controller via the MCP2515 CAN module.

2. MCP2515 CAN Controller

The MCP2515 module is used to implement CAN communication between the Raspberry Pi and the Arduino controller. It interfaces with the controllers using the SPI protocol and enables reliable communication between system nodes.

3. Arduino UNO R3

The Arduino UNO functions as the main motor control unit. It receives speed and direction commands through CAN communication and generates PWM signals to control the motor driver. It also reads encoder pulses to calculate the motor speed.

4. L298N Motor Driver

The L298N motor driver acts as an interface between the Arduino controller and the DC motor. It allows the Arduino to control motor speed using PWM signals and motor direction using digital control signals.

5. Rhino GB37 DC Geared Encoder Motor

The Rhino GB37 motor is a 12V DC geared motor with an integrated quadrature encoder. The encoder generates pulse signals that are used to determine the motor speed and direction. These pulses are processed by the Arduino to calculate the rotational speed in RPM.

C. System Working

The system begins by initializing the GUI and communication interfaces. The user sets the desired motor speed using the GUI slider and selects the direction of rotation. The command is transmitted through the CAN communication network to the Arduino controller.

The Arduino receives the command and generates a PWM signal to control the L298N motor driver. The motor driver regulates the voltage applied to the DC motor. Therefore, controls its speed and direction.

Encoder pulses generated by the motor are read by the Arduino to calculate the motor speed. The calculated speed is transmitted back to the GUI for real-time monitoring.

CAN command format – Hexadecimal values
ID = 0x101 (Pi to Motor)
ID = 0x201 (Arduino to GUI)

Table 1 CAN message format

Byte	Meaning
0	For direction: 0-Stop, 1-Forward (clockwise rotation), 2-Reverse (anti-clockwise rotation)
1-2	Target RPM
3-7	Unused

D. Motor Control Strategy

The motor control strategy of the proposed system is based on pulse width modulation (PWM), encoder feedback, and a PID control algorithm. The Arduino microcontroller generates PWM signals to regulate the voltage supplied to the motor through the L298N motor driver, thereby controlling the motor speed.

The encoder attached to the motor continuously produces pulses corresponding to the rotation of the motor shaft. These pulses are counted by the Arduino to determine the actual rotational speed of the motor. The motor speed is calculated using the encoder pulse count and the encoder resolution.

Motor speed is calculated using encoder pulses in revolutions per minute using:

RPM = (Number of pulses × 60) / CPR……. where CPR represents counts per revolution of the encoder.

CPR = 975 (w.r.t datasheet of motor)

For this project, the mode used is single channel A, single edge for speed,

Therefore, as the encoder is quadrature type, divide by 4,

CPR = 975/4

CPR = 244

The calculated motor speed is compared with the desired speed received from the graphical user interface through CAN communication.

Speed error = desired speed – measured speed,

A PID control algorithm processes this error and adjusts the PWM duty cycle accordingly.

Proportional (P) – reacts to the present error and provides immediate correction.

Integral (I) – compensates accumulated past errors to eliminate steady-state error.

Derivative (D) – predicts future error trends and improves system stability by reducing overshoot.

Tune kp, ki and kd values step-by-step.

By continuously adjusting the PWM signal based on these terms, the controller minimizes the speed error and maintains stable motor operation.

In this system, the encoder resolution after the gearbox corresponds to the effective counts per revolution of the output shaft. The Arduino counts encoder pulses using hardware interrupts and calculates the motor speed at regular time intervals. This measured speed is then used as feedback for the PID controller to regulate the motor speed accurately.

This closed-loop control mechanism ensures accurate speed regulation and improves the dynamic response of the motor system.

E. System Workflow Algorithm

1. Start system
2. GUI sends target speed
3. CAN message transmitted
4. Arduino receives command
5. Read encoder pulses
6. Calculate RPM
7. Compute PID error
8. Update PWM
9. Drive motor
10. Repeat loop

V. Experimental Results and Discussion

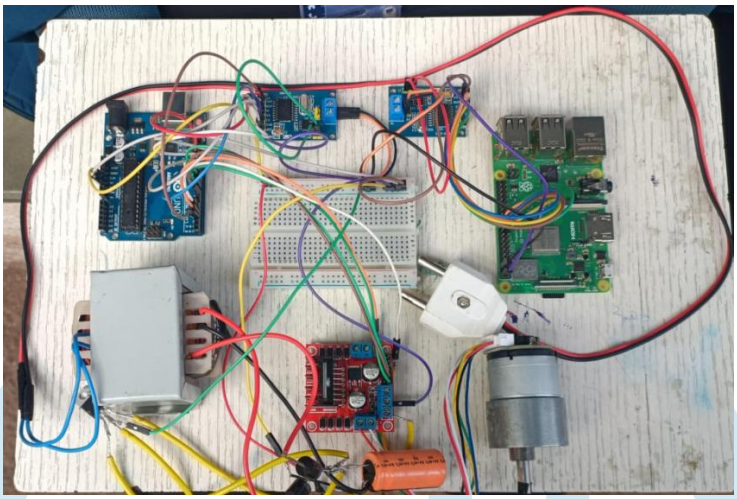

Figure 2 Experimental set-up

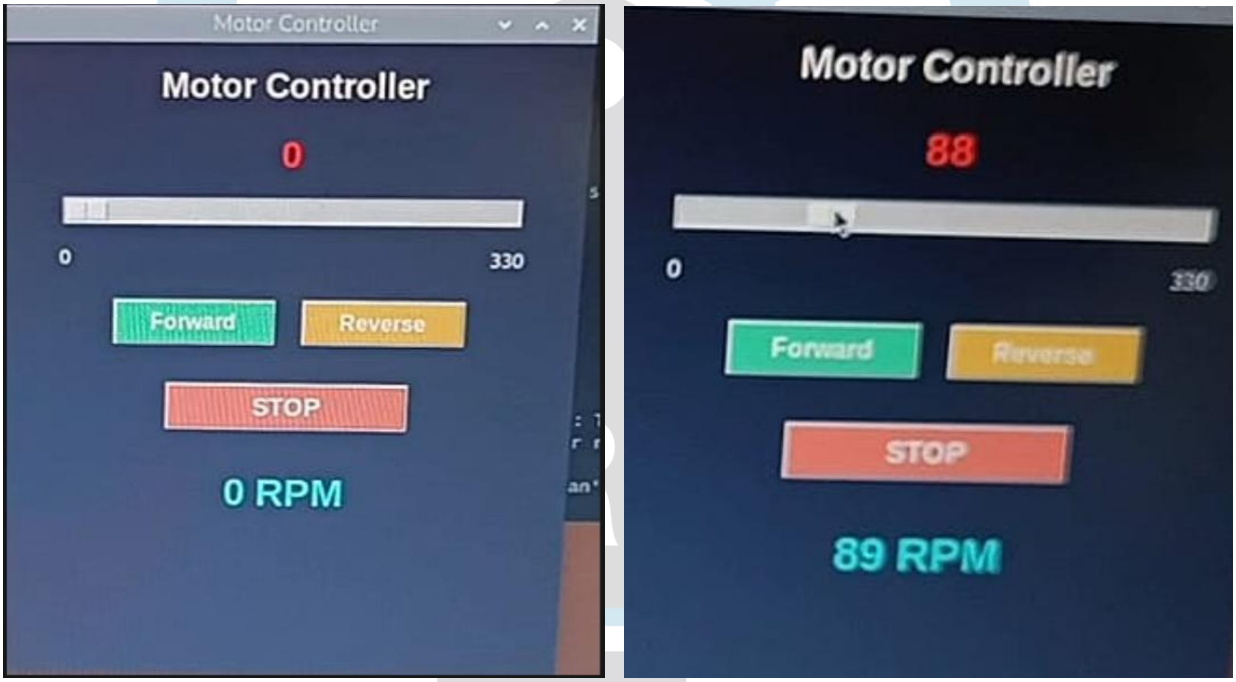

Figure 3 Graphical User Interface

The experiment setup successfully controls the speed and direction of dc motor with encoder through GUI. The 12v dc power supply provides power to the motor driver. Raspberry Pi manages the GUI and acts as the master node. The communication between Raspberry Pi and Arduino is CAN-based controllers. The Arduino acts as the slave node. It is responsible for the speed and direction control of DC motor. It receives the feedback from the encoder which is passed back to the Raspberry Pi and GUI via CAN protocol.

The tolerance of RPM is approximately plus-minus 2 RPM. The minimum rpm required for the motor to start rotation visibly is ~30 rpm. At 0 rpm, the motor does not rotate. The STOP button stops the rotating motor whenever clicked. The maximum rpm of the motor is 330 rpm. But, w.r.t motor specifications at no load condition it runs at 300 rpm. So, even when set speed is 330 rpm, the actual motor runs at 300 rpm and it it displayed on the GUI as well.

VI. Conclusion

The proposed system successfully demonstrates a GUI-based closed-loop DC motor control using Controller Area Network (CAN) communication. The integration of Raspberry Pi, Arduino, MCP2515 CAN module, and encoder feedback enables reliable and real-time motor control.

The implementation of the PID control algorithm ensures accurate speed regulation with minimal steady-state error and improved stability. The system is capable of maintaining the desired motor speed with a tolerance of approximately ±2 RPM, which confirms the effectiveness of the control strategy.

The graphical user interface provides a user-friendly platform for controlling motor speed and direction while simultaneously monitoring real-time performance. The use of CAN communication enhances system reliability and noise immunity, making it suitable for industrial applications.

Overall, the developed system proves to be an efficient, scalable, and practical solution for modern automation systems requiring remote monitoring and precise motor control.

Abbreviations and Acronyms

The following abbreviations and acronyms are used throughout this paper:

1. CAN – Controller Area Network
2. PID – Proportional–Integral–Derivative
3. RPM – Revolutions Per Minute
4. GUI – Graphical User Interface
5. SPI – Serial Peripheral Interface
6. MCP2515 – CAN controller module
7. PWM – Pulse Width Modulation

Units

The system primarily uses SI units for all measurements and calculations.

- Motor speed is measured in revolutions per minute (RPM).
- Electrical quantities such as voltage and current are expressed in volts (V) and amperes (A) respectively.
- Time is represented in seconds (s).
- Angular velocity calculations, where applicable, are expressed in radians per second (rad/s).

All numerical values are represented with a leading zero before decimal points (e.g., 0.25 instead of .25), ensuring clarity and consistency. Mixed unit systems are avoided to maintain dimensional correctness in equations and calculations.

Equations

The motor speed is calculated using encoder pulses as:

$$RPM = \frac{N \times 60}{CPR}$$

(1) where:

- N= number of encoder pulses
- CPR= counts per revolution

The error used in the control system is given by:

$$e(t) = RPM_{target} - RPM_{actual}$$

(2) The PID controller output is expressed as:

$$u(t) = K_p e(t) + K_i \int e(t)dt + K_d \frac{de(t)}{dt}$$

(3) where:

- K_p= proportional gain
- K_i= integral gain
- K_d= derivative gain

These equations enable accurate speed control and stable motor operation.

VII. ACKNOWLEDGMENT

The authors would like to express their sincere gratitude to Dr. S. S. Lokhande for her invaluable guidance, continuous support, and encouragement throughout the development of this project. Her insightful suggestions and technical expertise greatly contributed to the successful completion of this work.

The authors are also thankful to Prof. C. R. Kuwar, Project Coordinator, Department of Electronics and Telecommunication Engineering, for his constant motivation, valuable advice, and support during the course of this project.

Special thanks are extended to Dr. R. S. Kawaitkar, Head of the Department of Electronics and Telecommunication Engineering, for providing the necessary facilities and a conducive environment for carrying out this work successfully. The authors also express their gratitude to Dr. S. D. Lokhande, Principal of Sinhgad College of Engineering, for his encouragement and institutional support.

The authors would like to acknowledge the support provided by the faculty members and technical staff of the department for their assistance and cooperation during the implementation of the project.

Finally, the authors express their heartfelt appreciation to their families and friends for their constant encouragement, understanding, and support, which helped in the timely completion of this work.

REFERENCES

[1] Daffa M. Zhefara, 2024 (*Operational Failure Analysis* (originally in Indonesian language))

[2] Nguyen Van Thuyen, 2023 (*Optimize the Parameters of the Barrette Grab Bucket...*)

[3] Ilya Tsipurskiy, 2018 (*Simulation Curves of Scooping and Digging Grab*)

[4] Raspberry Pi 4 Model B Datasheet: https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-datasheet.pdf

[5] CAN Controller –
MCP2515 Tx:
https://ww1.microchip.com/downloads/en/DeviceDoc/MCP2515-Stand-Alone-CAN-Controller-with-SPI-20001801J.pdf
TJA1050 Rx:
https://www.nxp.com/docs/en/data-sheet/TJA1050.pdf

[6] A. Ray and K. R. Srivastava, "PID Controller Design for DC Motor Speed Control System," International Journal of Engineering Research and Applications, vol. 3, no. 3, pp. 120–125, 2013.

[7] S. B. Patil and R. S. Deore, "Speed Control of DC Motor Using PID Controller," International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, vol. 4, no. 4, pp. 2345–2352, 2015.

[8] A. Banerjee, "Implementation of CAN Bus Communication in Embedded Systems," International Journal of Computer Applications, vol. 98, no. 7, pp. 1–5, 2014.

[9] Arduino, "Arduino UNO Technical Specifications," 2020.

[10] STMicroelectronics, "L298N Dual Full Bridge Driver Datasheet," 2016.

[11] Microchip Technology Inc., "MCP2515 Stand-Alone CAN Controller Datasheet," 2018.