DC MOTOR CONTROL SYSTEM DESIGN WITH INTEGRATED SAFETY FEATURE

Nephat Gakuya Githua

Graduate Electrical and Electronic Engineer

Email: gakuyagithua.com@gmail.com

Phone: 0717776072

Role: Lead Designer & Developer
Date: February 2025
Version: 1.0

Contents

ABSTRACT.

This project details the design and implementation of a microcontroller-based DC motor control system with integrated fault protection. Using the STM32F411CEU6, motor speed is precisely regulated through hardware timer–driven Pulse Width Modulation (PWM), offering fine control of duty cycle and smooth dynamic response. A user interface is provided via a push-button start/stop mechanism, while system reliability is enhanced through real-time overcurrent monitoring with the ACS712 sensor. Upon detecting a fault condition, the controller immediately disables motor drive and activates a fault-indicator LED, ensuring hardware protection. The system was developed using STM32CubeIDE (HAL drivers in C) and validated through experimental testing. Results confirm stable PWM-driven speed control, accurate and fast fault detection, and robust user interaction. The integration of embedded safety mechanisms within the control loop demonstrates a scalable approach for industrial and consumer motor control applications requiring both performance and protection.

CHAPTER 1: INTRODUCTION

1.1 Background Information

DC motor control is essential for various industrial and consumer applications due to its simplicity, efficiency, and adaptability [1]. Precise control of motor operation enables improved system performance, energy savings, and enhanced safety. Pulse Width Modulation (PWM) is widely adopted for motor speed regulation, as it offers smooth performance and high efficiency by modulating the duty cycle of the drive signal [2]. Modern microcontrollers, such as the STM32 series, provide integrated peripherals that make them ideal for implementing PWM control, monitoring, and fault protection within a single system [3].

Motor control systems must also address operational safety. Fault detection mechanisms, such as overcurrent protection, are essential to prevent hardware damage, ensure user safety, and increase system reliability [4]. Integrating these safety features directly into the motor control loop enhances system resilience and extends the lifespan of both the controller and the motor.

1.2 Problem Statement

Conventional DC motor drivers often lack integrated safety mechanisms, leaving the system vulnerable to overcurrent faults, short circuits, or unexpected load variations. Without proper fault detection, such conditions can damage the motor, power electronics, or power supply, leading to increased costs and downtime. There is therefore a need for a microcontroller-based control system that combines efficient motor control with robust safety features.

1.3 Problem Justification

Developing a DC motor control system with integrated safety features demonstrates the practical application of embedded systems in real-world control problems. By leveraging the STM32 microcontroller's hardware timers and peripherals, the project achieves precise motor speed regulation while minimizing CPU overhead. The inclusion of fault detection mechanisms improves system reliability and user confidence, making the design suitable for industrial automation, robotics, and consumer electronics.

1.4 Objectives

1. Design and implement a DC motor control system using an STM32F411CEU6 microcontroller.
2. Control motor speed via a PWM signal.
3. Integrate a fault detection mechanism for overcurrent protection.
4. Incorporate a start/stop push-button for user interaction.
5. Use an LED to indicate the fault status.

CHAPTER 2: LITERATURE REVIEW

2.1 Existing Technologies

DC motor control has been extensively studied and implemented using different technologies, each optimized for specific applications. Two notable approaches are:

2.1.1 H-Bridge Motor Driver ICs

Integrated H-Bridge driver ICs, such as the L298N or L293D, are widely used for basic DC motor control. They provide bi-directional control and allow for PWM-based speed regulation. However, their current handling capacity is limited (typically up to 2A), and they often lack integrated safety features such as fault detection or thermal protection, making them less suitable for demanding industrial environments [5].

2.1.2 Dedicated Motor Control Units (MCUs)

Specialized motor control MCUs, such as Texas Instruments' C2000 series, offer advanced motor control capabilities including vector control, current regulation, and integrated fault protection. These devices are highly efficient but come at a higher cost and steeper learning curve, which may not be ideal for compact or low-budget projects [6].

Compared to these solutions, general-purpose microcontrollers such as the STM32 series provide a middle ground: they combine affordability and flexibility with precise PWM generation and customizable safety mechanisms, making them suitable for scalable DC motor control applications.

2.2 Project Components

1 **STM32F411CEU6 Microcontroller** – Serves as the central controller, handling PWM generation, ADC current sensing, and user input/output.
2 **DC Motor** – The actuator whose speed is controlled via the PWM signal.
3 **IRF540N N-Channel MOSFET** – Functions as a power switch to drive the motor based on PWM output.
4 **ACS712 Current Sensor** – Provides real-time current measurement for overcurrent detection.
5 **Push-Button Switch** – Enables manual start/stop control.
6 **LED Indicator** – Provides fault status indication.
7 **12V DC Power Supply** – Powers the motor and control system.
8 **Resistors (330 Ω, 220 Ω)** – Used for current limiting and gate drive control.
9 **0.1 µF Capacitor** – Used for noise filtering and decoupling.

These components were chosen for their availability, cost-effectiveness, and suitability for real-time embedded motor control.

2.3 System Overview

The proposed DC motor control system integrates speed regulation, user interaction, and safety monitoring into a single embedded platform. The STM32F411CEU6 generates a PWM signal using its internal hardware timers, which is applied to the gate of the IRF540N MOSFET to regulate the motor's supply voltage. The duty cycle of the PWM determines the motor's effective speed.

User interaction is achieved through a push-button that toggles motor operation between start and stop states. The ACS712 current sensor continuously monitors motor current and feeds the data to the STM32's ADC. If the sensed current exceeds a present threshold, the microcontroller immediately halts the PWM signal and activates the LED indicator to notify the user of the fault.

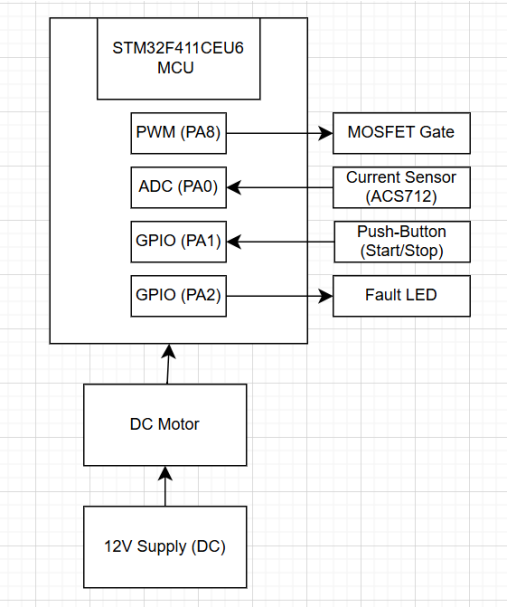
Key features of the system include:

- **Precise PWM-based speed control** with minimal CPU load.
- **Integrated overcurrent protection** for enhanced reliability.
- **User-friendly operation** via push-button input.
- **Fault notification** using a visual LED indicator.
- **Scalable design** suitable for future enhancements (e.g., wireless control, feedback-based regulation).

This architecture demonstrates a balance between simplicity, safety, and flexibility, making it suitable for both academic prototyping and real-world applications.

CHAPTER 3: METHODOLOGY

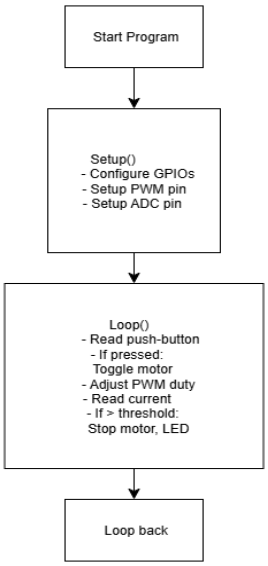
3.1 Block Diagram



This block diagram illustrates how the **STM32 microcontroller** acts as the central unit, generating PWM, monitoring current, and responding to user input.

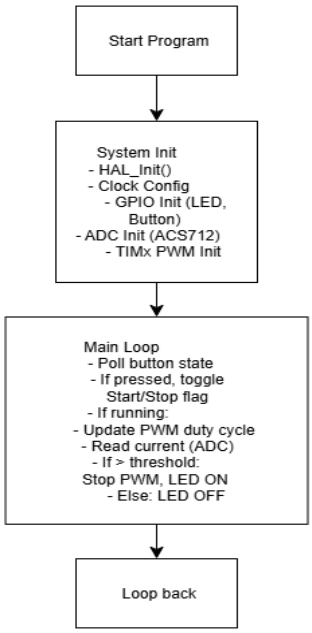
3.2 Software Flowcharts

3.2.1 Code1- C++ Flowchart



This flow relies on **Arduino libraries** (analogRead, digitalWrite, analogWrite) for hardware access.

3.2.2 Code 2- C HAL Drivers Flowchart



This flow emphasizes **peripheral initialization** and direct register-level control through HAL drivers, offering more precision than Arduino.

3.3 Code Structure & Logic

The implementation follows these steps:

1. **Peripheral Initialization**

- GPIO pins configured for button input (with pull-up), LED output, PWM output, and ADC input.
- TIM2 configured for PWM mode on PA8.
- ADC initialized for continuous current sensor readings.

2. **PWM Motor Control**

- Duty cycle determines motor speed.
- Controlled via HAL function (__HAL_TIM_SET_COMPARE) in STM32, or analogWrite() in Arduino IDE.

3. **Start/Stop Control**

- Push-button press toggles motor state (software flag).
- Debouncing ensures stable input.

4. **Fault Detection**

- ADC continuously samples motor current via ACS712.
- If measured current > threshold → PWM disabled, LED ON.

5. **Fault Indication**

- LED lights up to indicate overcurrent fault.

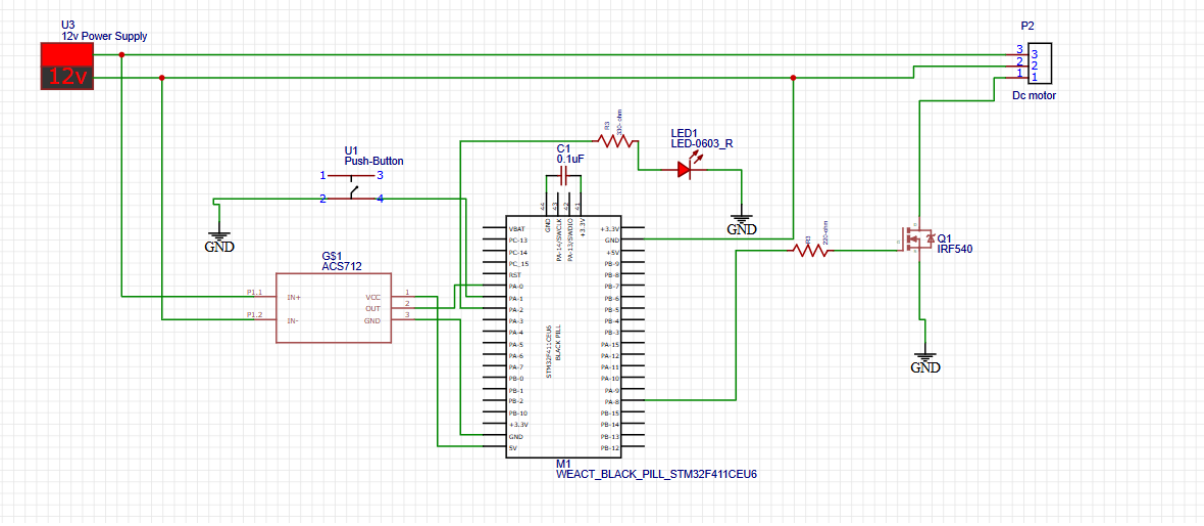
3.4 Circuit Design

Microcontroller Connections:

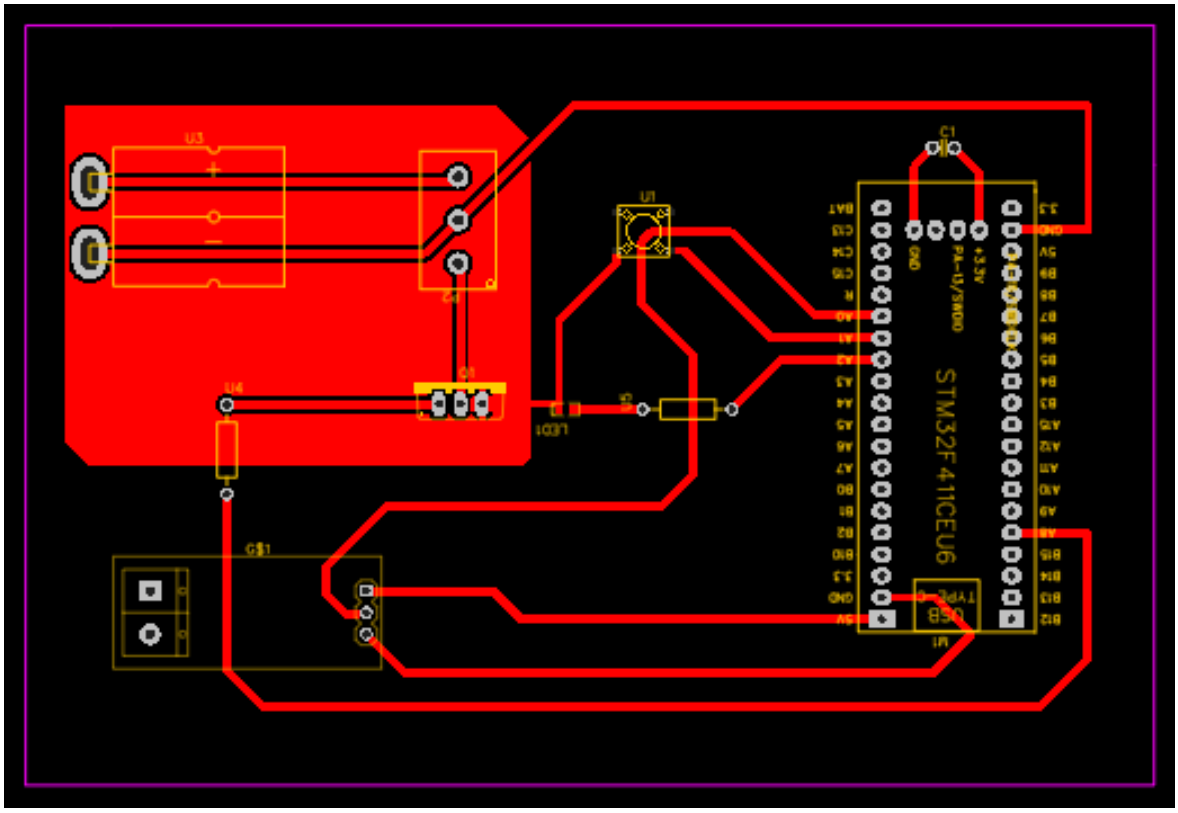
- **DC Motor:** Connect one terminal of the DC motor to the drain of the IRF540N MOSFET. Connect the motor terminal to the IN+ pin of the ACS712 current sensor. Connect the IN- pin of the ACS712 to the +12V DC supply.
  The source of the MOSFET connects to ground.
- **PWM Output:** Connect PA8 (pin labelled A8 on Black Pill) to the gate of the MOSFET through a 220-ohm resistor.
- **Current Sensor:** Connect the VCC and GND of the ACS712 to 5V and ground, respectively. Connect the output of the current sensor to PA0 (pin A0 on Black Pill) for ADC input.
- **Push-Button:** Connect one terminal of the push-button to ground and the other terminal to PA1 (pin A1 on Black Pill). Enable the internal pull-up resistor.
- **LED:** Connect the anode of the LED to PA2 (pin A2 on Black Pill) through a 330-ohm resistor. Connect the cathode to ground.
- **Power Supply:** Connect the 12V motor power supply to the motor and ensure that its ground is connected to the common ground shared with the STM32 microcontroller.
- **Decoupling Capacitors:** Place a 0.1uF ceramic capacitor near the microcontroller's power input pins (between 3.3V and GND pins) for noise suppression.

3.5 Diagrams

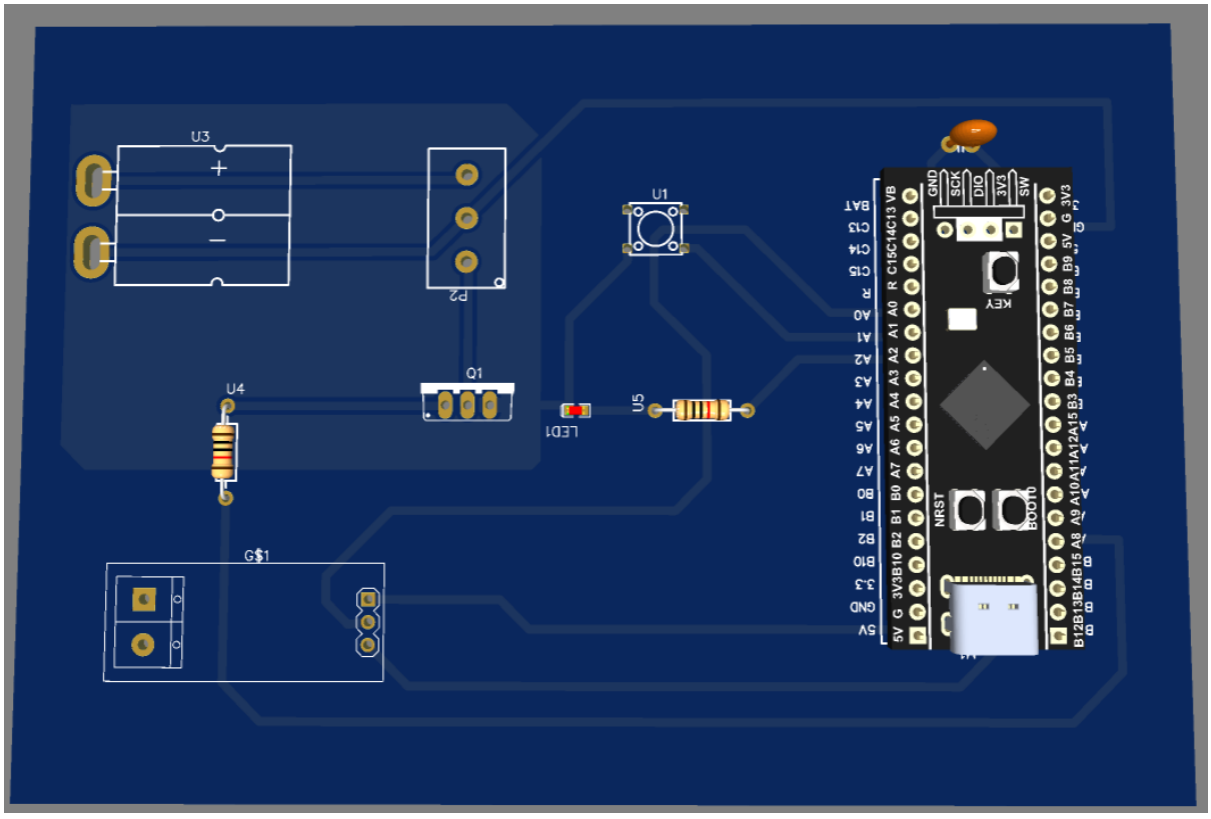
3.5.1 Schematic



3.5.2 PCB Layout



3.5.3 3D View



3.5.4 CODE:

GitHub Link:

https://github.com/NGGithua1/DC_MOTOR_CONTROL_SYSTEM_DESIGN_WITH_INTEGRATED_SAFETY_FEATURE/tree/main/Code

CHAPTER 4: RESULTS AND DISCUSSION

The designed DC motor control system achieved the set objectives and demonstrated reliable performance during testing. The motor was successfully driven using PWM (Pulse Width Modulation) signals, which enabled smooth speed variation with minimal electrical noise and stable mechanical response. The start/stop push-button interface offered intuitive user control and operated consistently under repeated trials.

The fault detection mechanism, implemented through current sensing with the ACS712 sensor, accurately identified overcurrent conditions. Upon fault detection, the motor was immediately stopped and the fault LED indicator was activated, confirming the safety response. This ensured protection of both the motor and the control circuitry from potential damage due to overload conditions.

Key Observations:

- PWM-based motor control provided stable operation with responsive changes in speed.
- Overcurrent detection worked reliably, preventing system stress and demonstrating effective hardware–software integration.
- The STM32CubeIDE environment facilitated precise hardware configuration, while Arduino IDE testing simplified initial validation of control logic.
- The system design and PCB layout minimized noise issues through the use of decoupling capacitors and proper grounding.
- Comparative testing between the Arduino IDE code and the STM32 HAL-based code highlighted differences in abstraction: the Arduino implementation was more straightforward, while the STM32 implementation offered greater control and efficiency.

Overall, the experimental results validate that the system is robust, responsive, and scalable for small-scale motor control applications.

CHAPTER 5: CONCLUSION AND RECOMMENDATIONS

The project successfully designed and implemented a DC motor control system with integrated safety mechanisms. The system demonstrated reliable performance in motor control, user interaction, and fault protection. By leveraging the STM32 microcontroller, the design achieved precision, flexibility, and efficient execution**,** proving suitable for real-world embedded control applications.

5.1 Conclusions:

- PWM control of the DC motor was achieved with smooth and stable operation**.**
- The implemented current sensing fault detection effectively safeguarded the system.
- The modular software design allows for future expansion and adaptability**.**
- Both Arduino IDE (prototype-friendly) and STM32CubeIDE (low-level, efficient) workflows proved valuable during development.

5.2 Recommendations:

Enhanced Safety Features

- Integrate temperature sensing for both motor and MOSFET protection.
- Implement emergency stop functionality with hardware interrupts.

Advanced Control Methods

- Upgrade to closed-loop speed control using PID algorithms and encoder feedback.
- Implement soft-start/soft-stop routines for smoother motor handling.

Connectivity & Remote Monitoring

- Add wireless control interfaces (Bluetooth, Wi-Fi, or LoRa) for remote operation.
- Integrate with IoT dashboards for live monitoring of motor current, temperature, and status.

System Scalability

- Extend design for multi-motor control with synchronized operation.
- Optimize PCB for smaller footprint and higher current handling.

REFERENCES

[1] R. e. a. Kumar, " Advanced Motor Control Systems.," *IEEE Transactions on Industrial Electronics.,* 2018.

[2] T. e. a. Lee, "PWM Techniques for DC Motor Efficiency Optimization.," *IEEE Engineering Journal.,* 2019.

[3] Smith, J., & Jones, L., , "Embedded Systems for Motor Control.," *IEEE Transactions on Embedded Systems.,* 2021.

[4] Brown, P., et al.,, "Fault Detection in Electronic Systems.," *IEEE Reliability Journal.,* 2020.

[5] A. Patel, "Design and Implementation of H-Bridge Motor Drivers for Low-Power Applications," *IEEE Transactions on Circuits and Systems,* 2017.

[6] S. Johnson, "High-Performance Motor Control Using Dedicated MCUs: A Case Study of the TI C2000 Family,," *IEEE Industrial Electronics Magazine,* 2019.