INTELLIGENT VEHICLE SPEED CONTROLLER

Enrol No. - **21102079**
Name of Student - **Nandit Verma**
Name of Supervisor - **Dr. Jasmine Saini**

Submitted in partial fulfillment of the requirement for the Degree of Bachelors of Technology

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

JAYPEE INSTITUTE OF INFORMATION TECHNOLOGY, NOIDA

TABLE OF CONTENTS

CERTIFICATE

This is to certify that the work titled "INTELLIGENT VEHICLE SPEED CONTROLLER USING RFID" submitted by Nandit Verma in partial fulfillment of the requirements for the award of the degree of B.Tech in Electronics & Communication Engineering and submitted to the Department of Electronics & Communication Engineering of Jaypee Institute of Information Technology, Noida has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

Signature of Supervisor:

Name of Supervisor Dr. Jasmine Saini

Date 16 December 2025

CANDIDATE'S DECLARATION

This is to certify that the work which is being presented in B.Tech Major Project Report titled **"INTELLIGENT VEHICLE SPEED CONTROLLER USING RFID"**, submitted by **"NANDIT VERMA"**, in partial fulfillment for the award of degree of B.Tech of Jaypee Institute of Information Technology, Noida has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

NAME OF SUPERVISOR - Dr. Jasmine Saini

DATE - 24 December 2025

ACKNOWLEDGEMENT

I would like to express our sincere gratitude to my supervisor **Dr. Jasmine Saini** for providing her invaluable guidance, comments, and suggestions throughout the course of the project. She consistently motivated and guided me towards the completion of the project. I am highly indebted to Maam for her constant supervision as well as providing necessary information regarding the project. However, it would not have been possible without the kind support and encouragement of my parents and colleagues who have helped me out with their abilities in developing the project.

I would also like to thank Jaypee Institute Of Information Technology, Noida for accepting my project in my desired field of expertise. I would also like to thank our friends and parents for their support and encouragement as we worked on this assignment.

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND

As the automotive industry continues to evolve with advances in electronics and communication technologies, the concept of Intelligent Transportation Systems (ITS) has gained increasing traction. One of the significant ITS applications is the Intelligent Vehicle Speed Controller (IVSC), designed to enhance vehicle safety, fuel efficiency, and driver comfort. In contrast to traditional vehicle speed control systems that rely solely on human input, the IVSC leverages technologies such as Radio Frequency Identification (RFID), sensor fusion, and fuzzy logic control to make real-time speed adjustments. This document delves into the design, implementation, and performance comparison between an intelligent vehicle speed control system and conventional speed management in vehicles.

The rapid urbanization of cities, increased traffic congestion, and the surge in vehicular usage have amplified the importance of road safety and traffic management. One of the most pressing issues in modern transportation is the inability of conventional systems to respond effectively to dynamic road and traffic conditions. Traditional speed control systems primarily rely on the driver's constant attention and responsiveness. These systems, such as manual throttle control or standard cruise control, do not possess the capability to adapt in real time to changing traffic patterns, varying speed limits, or temporary road hazards.

Numerous studies and accident reports highlight that human error, particularly related to speed mismanagement, is a leading cause of road accidents. In urban areas where traffic signs, pedestrian crossings, and construction zones require frequent changes in speed, conventional systems often fall short. Drivers may miss speed limit signs or fail to adjust their speed promptly, leading to traffic violations and accidents.

To address these limitations, Intelligent Vehicle Speed Controllers (IVSCs) have emerged as a promising alternative. These systems utilize a combination of advanced sensors, RFID technology, and intelligent algorithms to autonomously manage vehicle speed. The RFID-based

Infrastructure-to-Vehicle (I2V) communication allows the system to identify speed regulations and traffic warnings without driver intervention. This technology is particularly useful in zones where speed limits change frequently or are temporarily modified due to maintenance work or unexpected road conditions.[1]

Furthermore, integrating a Hall Effect-based sensor enables accurate and real-time monitoring of vehicle speed, which is crucial for making informed control decisions. The fuzzy logic controller processes the data from these sensors and adjusts the vehicle's speed accordingly. Unlike conventional binary decision systems, fuzzy logic allows for nuanced decision-making that mimics human reasoning, making it highly effective in variable and uncertain environments.

From a technological standpoint, IVSCs represent a significant leap forward. They reduce dependency on human drivers, thereby minimizing the risk of errors and enhancing overall safety. Additionally, maintaining optimal speeds based on road conditions contributes to better fuel efficiency, lower emissions, and reduced wear and tear on vehicle components.

Economically and environmentally, the adoption of IVSCs offers long-term benefits. Reduced accident rates lead to lower insurance costs and fewer emergency response requirements. Improved fuel efficiency translates to cost savings for individual drivers and commercial fleet operators alike. From an environmental perspective, maintaining consistent speeds reduces fuel consumption and CO2 emissions, aligning with global efforts toward sustainable transportation.[1]

However, transitioning from conventional systems to intelligent ones is not without challenges. Infrastructure needs to be updated to support widespread deployment of RFID tags and compatible road signage.[2] Vehicles must be equipped with the necessary sensors, processors, and actuation systems to interpret and act on the data received. Public acceptance also plays a crucial role, as drivers must trust the system to manage their vehicle's speed safely and effectively.[3]

In conclusion, the background of this study underscores the urgent need for a paradigm shift in vehicle speed management. While conventional systems have served well in the past, they are no longer adequate for modern, fast-paced, and complex road networks. Intelligent Vehicle Speed Controllers, through the use of RFID-based communication, precise speed sensing, and intelligent control logic, offer a viable and superior alternative. This report aims to explore the working principles, advantages, and real-world applicability of IVSCs while comparing their performance with traditional systems to highlight the transformative potential of intelligent speed control in future transportation systems.

1.2 PROJECT OVERVIEW

The Intelligent Vehicle Speed Controller using RFID/RF Technology is an innovative system designed to automatically regulate vehicle speed in designated areas by utilizing Radio Frequency Identification (RFID) technology. This system aims to enhance road safety by ensuring vehicles adhere to speed limits in specific zones, such as school areas, hospitals, and construction sites, without relying solely on driver compliance.

System Components:

RFID Tags and Readers:

- **RFID Tags**: These are passive or active devices placed at strategic locations, such as traffic signs or road infrastructure, indicating the permissible speed limit of that zone.[5]
- **RFID Readers**: Installed within the vehicle, these devices detect signals from the RFID tags as the vehicle approaches or enters a controlled zone.[3]

Microcontroller Unit:

- Serves as the central processing unit that interprets data from the RFID reader and controls the vehicle's speed accordingly.[5]

Vehicle Control Interface:

- Interfaces with the vehicle's throttle and braking systems to adjust speed based on commands from the microcontroller.[4]

User Display and Alert System:

- Provides real-time feedback to the driver about current speed limits and system actions, often through visual displays or auditory alerts.

Operational Mechanism:

- As the vehicle approaches a zone with a specific speed limit, the RFID reader detects the corresponding RFID tag.[4]
- The reader transmits the tag's information to the microcontroller.The microcontroller processes this data and determines if the vehicle's current speed exceeds the zone's limit.

- If necessary, the microcontroller sends signals to the vehicle's control interface to reduce speed to the permissible limit.
- The system alerts the driver through the display and alert system about the enforced speed change.

Advantages:

- **Enhanced Road Safety:** By automatically enforcing speed limits in critical areas, the system reduces the likelihood of accidents caused by overspeeding.
- **Driver Assistance:** Aids drivers in adhering to speed regulations, especially in unfamiliar areas or zones where speed limits frequently change.
- **Scalability:** The system can be implemented in various environments and adapted to different vehicle types.
- **Cost-Effective:** Utilizes RFID technology, which is relatively inexpensive and easy to deploy.

Limitations:

- **Infrastructure Dependency:** Requires the installation and maintenance of RFID tags at all relevant locations, which can be resource-intensive.
- **Signal Interference:** RFID systems may experience interference from environmental factors or other electronic devices, potentially affecting reliability.
- **Driver Override**: In some designs, drivers may have the ability to override the system, which could compromise its effectiveness.

The Intelligent Vehicle Speed Controller using RFID/RF Technology represents a significant step forward in leveraging technology to promote safer driving behaviors and reduce traffic-related incidents in sensitive areas.[2]

1.3 OBJECTIVES

The *Intelligent Vehicle Speed Controller using RF Technology* project is conceived with the core aim of enhancing road safety through automation, specifically targeting areas where the consequences of overspeeding are particularly severe—such as school zones, residential neighborhoods, hospitals, and accident-prone zones. The following are the major objectives of the project, explained in depth:

1. Automatic Speed Regulation in Restricted Zones

The foremost objective of this project is to develop a system that can **automatically regulate the speed of a vehicle** when it enters specific predefined zones. In many accident cases, overspeeding is the primary cause, and drivers often fail to notice or obey traffic signboards. The proposed system bypasses human error by controlling the vehicle's speed autonomously using wireless communication. Whenever a vehicle enters a monitored zone, the system detects this and restricts the speed to a preset, safe limit—thereby reducing the risk of collision and harm.

2. Zone-Based Dynamic Control

Another significant objective is to allow **dynamic control based on geographic zones**. Each zone (e.g., school, hospital, highway) has its own ideal speed limit. For instance, school zones may require speeds not exceeding 20 km/h, while highways may allow up to 80 km/h.[1] The project incorporates RF transmitters at such zone entry points to broadcast the allowable speed limit. The receiver installed in the vehicle detects this broadcast and instructs the control system to adjust accordingly. This zone-based modular control makes the system adaptable and scalable to any geographical layout.

3. Minimizing Human Error in Speed Monitoring

Human limitations such as distraction, fatigue, or ignorance often lead to traffic violations. This system is developed to **eliminate dependency on driver attentiveness** by automating the speed control mechanism. The system ensures compliance with local traffic regulations without requiring the driver to manually monitor and adjust their speed. As a result, even if a driver fails to observe the surroundings or forgets to slow down in a restricted zone, the system overrides their input and maintains a safe driving speed.

4. Enhanced Safety for Pedestrians and Vulnerable Areas

Pedestrians, especially children and elderly people, are most at risk in zones near schools, parks, hospitals, and residential streets. The project aims to **create safer environments for non-motorized road users** by ensuring that vehicles slow down in these zones. The intelligent controller takes charge of speed regulation during the most critical moments—when lives are at risk due to proximity to fast-moving vehicles.

5. Real-Time Alert and Feedback Mechanism

The system also features **real-time feedback mechanisms** via an LCD display or similar user interface. The system not only changes the speed of the vehicle but also alerts the driver about current speed restrictions and violations, such as "Over Speeding" notifications. This helps educate drivers and instills better road discipline by making them aware of their driving habits.

6. Low-Cost and Scalable Safety Solution

The design and implementation of this system are intentionally kept **cost-effective and scalable**, using readily available microcontrollers (like Arduino), RF modules, and motor control drivers. This allows for widespread adoption, especially in developing regions where budget constraints limit the deployment of advanced traffic control technologies. The simplicity of the hardware makes it ideal for mass deployment, even in rural and semi-urban roadways.

7. Encouraging Compliance with Speed Regulations

Speed limits are one of the most basic yet ignored rules on the road. This project enforces **mandatory compliance** in critical zones, which indirectly trains drivers to develop safer driving habits. Over time, this technological enforcement could reduce the number of violations even outside automated zones due to heightened driver awareness.

1.4 METHODOLOGY

The methodology outlines the step-by-step process followed to design, develop, and implement the *Intelligent Vehicle Speed Control System*. This system is based on zone-based speed regulation using RF communication. The process includes identifying the problem, defining the system architecture, selecting appropriate hardware components, writing the control logic, integrating the modules, and testing under different conditions to validate the effectiveness of the solution.

1. Problem Identification

Overspeeding in critical areas such as school zones, hospitals, residential neighborhoods, and accident-prone roads remains a leading cause of road fatalities. Traditional enforcement methods such as signboards and surveillance cameras require human compliance and manual intervention, which are often unreliable. Therefore, there was a clear need for an **automatic, zone-based, and vehicle-integrated speed limiting system**.

2. System Design and Architecture

The system design is bifurcated into two primary segments:

a) RF Transmitter Module (Stationary)

Installed at the entry point of specific zones (e.g., school zone), this module broadcasts a predefined speed limit signal using RF communication. Each zone is assigned a unique signal code corresponding to a specific speed limit (e.g., 20 km/h for schools, 40 km/h for residential zones, etc.).

b) RF Receiver Module (On-Vehicle)

This mobile unit is installed inside the vehicle. It receives the transmitted signal from the zone and triggers speed control mechanisms when a zone is detected. It integrates:

- RF Receiver
- Microcontroller (Arduino UNO)
- DC Motor (to simulate vehicle speed)
- Motor Driver (L293D)
- LCD Display
- HT12D Decoder

3. Selection of Components

Each component was selected based on its suitability, availability, cost, and ease of integration:

- **RF Transmitter and Receiver (433 MHz)**: For wireless communication within a reasonable range (~100 m).
- **HT12E/HT12D Encoder-Decoder**: To ensure secure and reliable data transmission between RF modules.
- **Arduino Uno**: Acts as the system's brain, responsible for processing logic and managing peripherals.
- **L293D Motor Driver**: Controls the speed of the DC motor based on instructions from Arduino.
- **DC Motor**: Simulates vehicle movement.
- **16x2 LCD Display**: Shows real-time status and alerts to the driver.

4. Hardware Setup

a) Transmitter Circuit

- Powered by a 9V battery.
- HT12E encoder is used to encode the 4-bit data corresponding to the zone's speed limit.
- The encoded signal is transmitted via the RF transmitter module.
- Switches (D0–D3) are used to define the speed limit:
    - D0 = 20 km/h (School)
    - D1 = 40 km/h (Residential)
    - D2 = 60 km/h (Highway - Low)
    - D3 = 80 km/h (Highway - High)

b) Receiver Circuit (Inside Vehicle)

- RF receiver captures the transmitted signal.
- HT12D decodes the signal and forwards it to the Arduino.
- Arduino processes the signal and compares the current vehicle speed to the limit.
- If overspeeding is detected, Arduino sends a command to the motor driver to reduce the speed.
- LCD displays the zone info and status (e.g., “Over Speeding”, “Speed Reduced to 40 km/h”).

5. Software Development (Arduino Programming)

The Arduino Uno was programmed using the Arduino IDE. The software logic performs the following tasks[10]:

1. **Initialize peripherals** (LCD, motor driver, serial monitor, etc.).
2. **Receive data** from HT12D and determine the corresponding speed limit.
3. **Compare** current vehicle speed (simulated through potentiometer or increment switch) with the limit.
4. **Control** the motor speed using PWM signals if overspeeding is detected.
5. **Display** appropriate messages on the LCD.

Key Code Functions:

- `analogRead()`: Read speed input from simulation (potentiometer/switch).
- `digitalRead()`: Read decoder output.
- `analogWrite()`: Set motor speed using PWM signal.
- `lcd.print()`: Display messages.

6. System Workflow

The complete working mechanism follows a real-time embedded control loop. Here's how it unfolds:

1. **Vehicle Movement Simulation**:
    - The DC motor represents the motion of the vehicle.
    - Speed is simulated via switches to set the motor speed.
2. **Zone Entry Detection**:
    - As the vehicle enters a zone, it comes within the range of the RF transmitter.
    - The RF receiver inside the vehicle captures this signal and decodes it.
3. **Speed Comparison and Control**:
    - The Arduino compares current simulated speed with the transmitted speed limit.
    - If speed > limit:
        - Arduino reduces PWM output to the motor.
        - LCD displays “Over Speeding – Speed Reduced to XX km/h”.
    - If speed ≤ limit:
        - No action is taken; vehicle moves normally.
4. **Zone Exit**:
    - Once the vehicle moves out of RF range, the receiver no longer detects the signal.
    - Control is returned to manual simulation, and driver regains full speed control.

7. Case Testing and Scenarios

The system was tested using multiple speed zones with varying thresholds. Key observations:

- **School Zone (20 km/h)**:
    - Vehicle enters at 100 km/h.
    - System cuts down speed and displays: “Over Speeding – Speed Reduced to 20 km/h”.
- **Residential Area (40 km/h)**:
    - System limits speed to 40 km/h.
- **Highway (60 km/h and 80 km/h)**:
    - Speeds adjusted accordingly based on RF signal (D2 or D3 set high).

All scenarios confirmed successful automatic speed reduction and status alerts on LCD.

8. Simulation with TinkerCAD

Before physical implementation, the circuit was virtually simulated using **TinkerCAD**:

- Components like Arduino, LCD, motor, RF module, and decoder were added.
- Simulated signals were tested by toggling virtual switches.
- Enabled validation of logic without risking hardware damage.

Simulation also helped refine the program logic and pin configurations before hardware assembly.

9. Integration and Real-Time Implementation

After successful simulation, the entire circuit was assembled on a breadboard. Special attention was paid to:

- Power management (9V supply for RF modules and Arduino).
- Signal integrity between encoder and decoder.
- Stable motor speed control under varying voltage inputs.

The vehicle’s hardware prototype was placed on a testing track marked with virtual zones to demonstrate automatic speed regulation in real-time.

CHAPTER 2

LITERATURE SURVEY

2.1 RFID-BASED SPEED CONTROLLER

In the modern era of technological innovation, the transportation sector has witnessed a remarkable transformation with the integration of automation and intelligent systems. Among these advancements, RFID-based speed control systems stand out as an effective solution to enhance road safety, streamline traffic management, and reduce human intervention in critical situations. With the increasing number of road accidents, particularly in sensitive zones such as school areas, hospital vicinities, and residential neighborhoods, it has become imperative to develop systems that can autonomously regulate vehicle speeds.[7] RFID (Radio Frequency Identification) technology offers a promising approach to achieving this objective.

RFID is a wireless communication technology that uses electromagnetic fields to identify and track objects tagged with RFID chips. A typical RFID system consists of three main components: the RFID tag, the RFID reader, and the control system.[8] The tag, which contains electronically stored information, is placed at strategic locations such as speed zones, entry points of restricted areas, or along highways. The reader, installed in vehicles or fixed locations, detects the presence of the tag and reads the data embedded within it. This information is then relayed to a microcontroller or processing unit, which interprets the data and triggers a response, such as adjusting the vehicle's speed.

The fundamental concept behind RFID-based speed control is relatively simple yet powerful. When a vehicle approaches a particular zone, such as a school or a hospital, an RFID reader placed in the vehicle detects the RFID tag positioned at the beginning of the zone. The tag contains a specific instruction, typically the maximum permissible speed limit for that area. Upon reading the tag, the onboard microcontroller processes this data and automatically adjusts the speed of the vehicle by controlling the throttle or engine output, ensuring that the vehicle does not exceed the set speed limit within the designated area. Once the vehicle exits the zone and passes another RFID tag indicating the end of the restricted area, the system can revert to its normal speed control, either by returning control to the driver or adjusting to a new speed limit.

The integration of RFID technology with vehicle control systems offers several advantages. First and foremost, it significantly enhances safety, especially in zones where high speeds can lead to fatal accidents. For example, in school zones, children are often unpredictable in their movements, and lower vehicle speeds give drivers more time to react, potentially saving lives. Similarly, in hospital areas, a calm and quiet environment is essential, and controlling vehicle speed contributes to maintaining that environment. Moreover, in accident-prone or construction zones, automated speed regulation can reduce the risk of crashes caused by driver negligence or distraction.

Another key benefit of RFID-based speed control systems is the reduction in the need for human enforcement. Traditionally, speed limits are enforced through road signs and speed cameras, requiring constant monitoring by traffic authorities. However, these methods are not always effective, as drivers may ignore signs or not notice them due to various distractions. RFID-based systems, on the other hand, function autonomously and do not rely on driver compliance. Once implemented, they work continuously without the need for active supervision, making them a cost-effective and reliable solution for traffic regulation.

In addition to enhancing safety and reducing the burden on enforcement agencies, RFID-based systems can also be integrated into smart city infrastructure. With the growing adoption of Internet of Things (IoT) technologies, urban centers are becoming more connected and intelligent. RFID tags can be linked to centralized traffic management systems that monitor vehicle movements in real-time. Data collected from RFID-enabled vehicles can be analyzed to identify traffic patterns, peak congestion times, and high-risk zones.[8] This information can then be used to optimize traffic flow, plan infrastructure development, and implement dynamic speed limits based on real-time conditions.

Despite the numerous advantages, RFID-based speed control systems are not without challenges. One of the primary concerns is the initial cost of implementation. Equipping vehicles with RFID readers and installing RFID tags across urban and rural road networks can be expensive, especially in developing countries. Additionally, there is the issue of standardization, as vehicles from different manufacturers may use different technologies or protocols, making interoperability a concern. Moreover, RFID systems are susceptible to environmental interference, such as extreme weather conditions or physical obstructions, which can affect signal transmission and accuracy.

To overcome these challenges, researchers and engineers are working on hybrid systems that combine RFID with other technologies such as GPS, ultrasonic sensors, and wireless communication protocols.[4] For instance, a vehicle may use RFID to detect specific zones and GPS to determine its exact location, providing a more robust and accurate speed control mechanism. Furthermore, advancements in microcontroller technology and vehicle automation are making it easier to implement these systems without significant changes to vehicle design or performance.[7]

Another important aspect to consider is the legal and ethical implications of automated speed control. In fully autonomous systems, the vehicle overrides the driver’s control to enforce speed limits, which raises questions about liability in case of system failure or unexpected behavior. As such, clear regulations and standards must be developed to govern the design, implementation, and use of these systems. Driver awareness and training are also essential to ensure smooth adoption, as users must understand how and when the system intervenes.

In educational institutions, smart campuses are increasingly adopting RFID-based solutions not just for vehicle speed control, but also for access management, attendance tracking, and resource optimization.[7] This reflects a broader trend toward automation and intelligent systems aimed at creating safer, more efficient environments. Similarly, logistics and public transportation systems are exploring the use of RFID to manage vehicle speeds in terminals and loading areas, improving safety and operational efficiency[7]

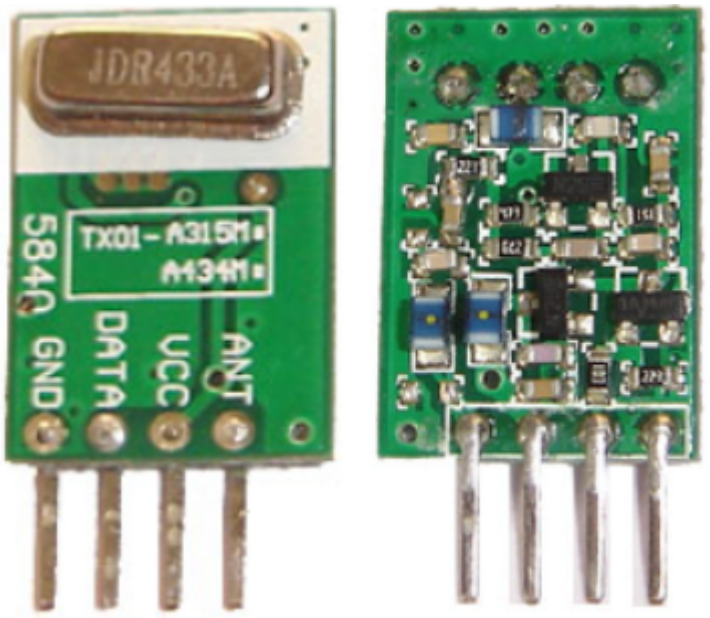

Fig. 2.1.1 RF Transmitter

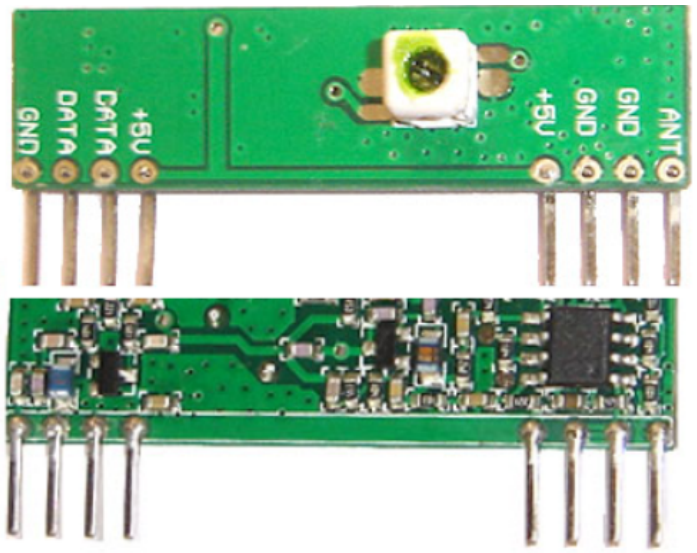

Fig. 2.1.2 RF Reviever

Parameter	Symbol	Min	Typ.	Max	Unit
Operating Voltage	Vcc	1.5	3.0	12	Volts DC
Operating Current Data = VCC	Icc	-	11mA @3V 59mA @5V	-	mA
Operating Current Data = GND	Icc	-	100	-	uA
Frequency Accuracy	TOL fc	-75	0	+75	Khz
Center Frequency	Fc	-	433	-	Mhz
RF Output Power		-	4 dBM@3V (2 mW) 16 dBM@5V (39 mW)		dBm / mW
Data Rate		200	1K	3K	BPS
Temperature		-20		+60	Deg. C
Power up delay			20		ms

Fig. 2.1.3 Specification of RF Transmitter

Parameter	Symbol	Min	Typ.	Max	Unit
Operating Voltage	Vcc	4.5	5.0	5.5	VDC
Operating Current	Icc	-	3.5	4.5	mA
Reception Bandwidth	BW rx	-	1.0	-	MHz
Center Frequency	Fc	-	433.92	-	MHz
Sensitivity	-	-	-105	-	dBm
Max Data Rate	-	300	1k	3K	Kbit/s
Turn On Time	-	-	25	-	ms
Operating Temperature	T op	-10	-	+60	°C

Fig.2.1.4 Specification of RF Receiver

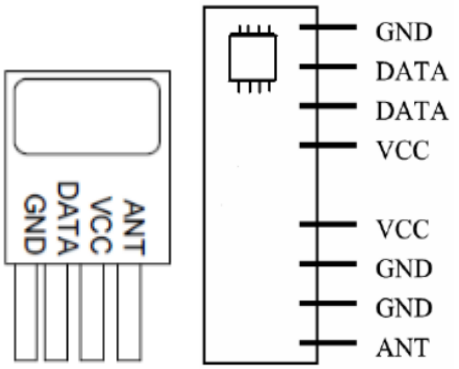

Fig. 2.1.5 Pin Description for RF Transmitter and Receiver

2.2 MICROCONTROLLER UNIT (MCU)

A core component that plays a pivotal role in the functioning of an RFID-based speed control system is the **Microcontroller Unit (MCU)**. Acting as the brain of the system, the microcontroller serves as the interface between the RFID reader and the vehicle's speed control mechanism. It processes the data received from the RFID reader, makes logical decisions based on preprogrammed instructions, and triggers the appropriate control actions—such as reducing the throttle, activating alerts, or adjusting the engine output.

In practical terms, when an RFID reader detects a tag placed in a restricted zone (e.g., a school or hospital area), it sends the tag's information, which includes speed limit instructions, to the microcontroller. The MCU is preloaded with a program (usually written in C or embedded C++) that interprets this input. Once the relevant tag ID and associated speed limit are identified, the MCU commands the vehicle's actuators to adjust the speed accordingly. This can involve controlling a DC motor in small prototypes or sending commands to the Engine Control Unit (ECU) in advanced automotive implementations.

Popular microcontrollers used in such projects include the **Arduino Uno**, **ATmega328**, **ESP32**, or **PIC** microcontrollers. These boards are chosen based on the complexity of the system, processing power requirements, and additional features like Wi-Fi or Bluetooth connectivity, which are useful for IoT-based extensions.[10]

Another reason the microcontroller is indispensable is its ability to manage real-time inputs and outputs. For example, it can also interface with sensors like IR modules to detect obstacles, GPS modules for positional awareness, and even display units (like LCDs or LEDs) to provide speed alerts to the driver. In some systems, buzzers or voice alerts are triggered by the microcontroller to inform drivers when the speed limit has been automatically applied.

Moreover, the microcontroller can maintain logs of RFID detections and vehicle responses, which can be useful for diagnostics or integration with larger traffic management databases. These logs can also help in audit trails for compliance with traffic regulations.

Many different microcontrollers and development platforms are available for physical computing applications. Systems like the Parallax Basic Stamp, Netmedia's BX-24, Phidgets, MIT's Handy Board, and others provide similar capabilities.[10] These platforms generally hide the complex aspects of microcontroller programming and present users with an accessible interface. Arduino follows the same philosophy of simplification, but it offers several unique benefits for students, educators, and hobbyists:

Cost-effective:
Arduino boards are considerably cheaper than most other microcontroller platforms. The most basic version can even be assembled manually, and the ready-made boards typically cost less than 1500.

Compatible across operating systems:
The Arduino IDE works on Windows, macOS, and Linux, whereas many other microcontroller tools function only on Windows.

Beginner-friendly programming environment:
The Arduino IDE is straightforward for newcomers while still supporting advanced functionality. Since it is built on the Processing environment, students familiar with Processing will find Arduino's interface easy to navigate.

Open-source and extensible software:
Arduino's software is released as open-source, enabling developers to modify or enhance it. The programming language can be expanded with C++ libraries, and users interested in the underlying concepts can transition to AVR C—the language behind Arduino. Advanced users can even embed AVR-C code directly inside Arduino sketches.

Open-source and customizable hardware:
Arduino boards use Atmel's ATMEGA8 and ATMEGA168/ATMEGA2560 microcontrollers, and their hardware designs are shared under a Creative Commons license.[10] This allows skilled circuit designers to create customized versions or improve existing ones. Even beginners can build a breadboard version to learn the hardware structure and reduce costs.

In summary, the microcontroller is the decision-making and control center of the RFID-based speed control system. Its ability to process inputs, run embedded control algorithms, and interface with both sensors and actuators makes it a fundamental component. Without the MCU, the system would lack the necessary logic and control capability to enforce automatic speed regulation based on RFID data, thereby rendering the solution ineffective.

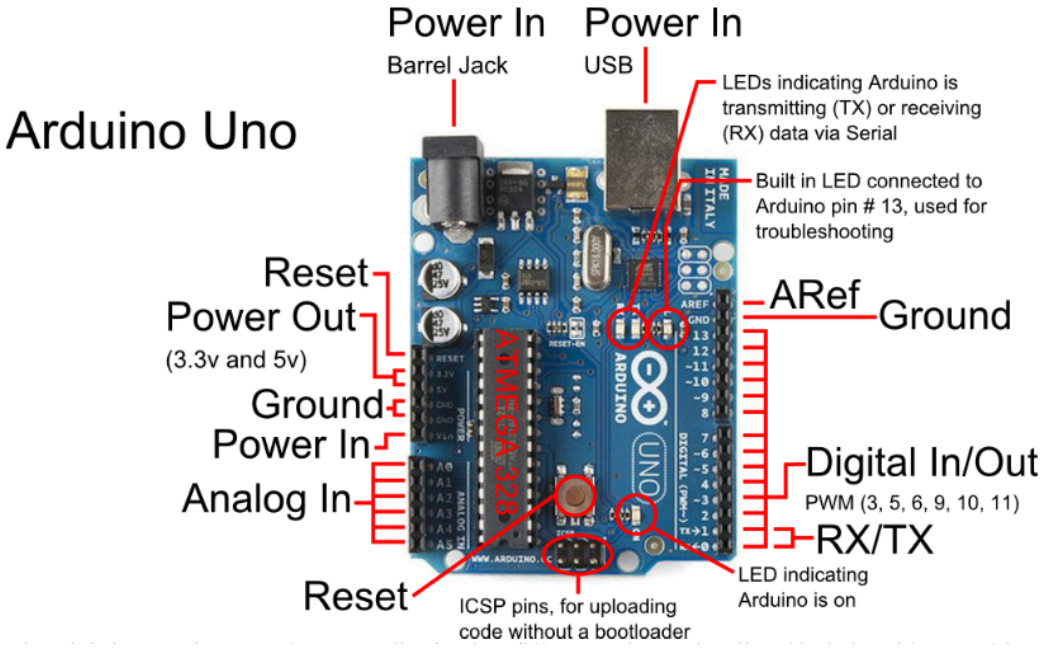

Fig. 2.2.1 Arduino

CHAPTER 3

SYSTEM ARCHITECTURE

3.1 SYSTEM COMPONENTS

1. Zone Detection Module

Components Used:

- Pushbutton 1 (simulating School Zone)
- Pushbutton 2 (simulating another restricted zone like Construction)

Functionality:
This module detects which predefined zone the vehicle is in. In a real-world setup, RFID tags would be placed at different zone entry points. In our simulation, these zones are emulated using pushbuttons.

Working Logic:

- **Button 1 Pressed** → System interprets this as **School Zone** → Speed is limited to **3000 RPM**.
- **Button 2 Pressed** → Interpreted as **Construction or Safety Zone** → Speed is limited to **1500 RPM**.
- **No button pressed** → System assumes a **Free Road or Highway** → Motor runs at **full speed: 16530 RPM** (in simulation).

This module is the trigger for initiating speed control based on the environment.

2. Arduino UNO (Control Unit)

Functionality:
Acts as the **central control unit** responsible for reading input from buttons, controlling the motor speed, estimating the RPM, and displaying the information.

Key Operations:

- **PWM Signal Generation:**
  - The Arduino uses the analogWrite() function to generate PWM (Pulse Width Modulation) signals.
  - PWM signal is sent to the **ENA (Enable A)** pin of the L293D driver.
  - The PWM duty cycle determines the motor speed (0–255 range).

- **RPM Estimation:**
    - A linear relationship is assumed between PWM and RPM.
    - Formula[12]: $\text{RPM} = \text{map}(\text{PWM\_value}, 0, 255, 0, 0 \quad 0, 16530)$
    - Example:
        - PWM = 255 → 16530 RPM
        - PWM = 127 → ~8265 RPM
        - PWM = 64 → ~4146 RPM, etc.
- **Display Output:**
    - The Arduino sends the estimated RPM to either the **Serial Monitor** or a **16x2 LCD display**.
    - This provides real-time feedback to users about the current motor speed.

3. Motor Driver Circuit (L293D)

Functionality:
This IC acts as an **interface between the Arduino and the motor**. The Arduino alone cannot handle the current required by the motor, hence a motor driver is used.[12]

Operations:

- Receives **PWM input on ENA pin** from Arduino.
- Controls the **speed** of the DC motor according to the PWM value.
- **Motor direction is fixed** in this application:
    - IN1 = HIGH, IN2 = LOW → Motor moves **forward** only.
- L293D ensures **safe current amplification and motor control**.

4. DC Motor

Functionality:
The motor represents the **vehicle's movement system**. It's the final actuator in the system that reacts based on zone information.[13]

Behavior:

- Runs at speeds corresponding to PWM values.
- In the Tinkercad environment, the **maximum RPM is 16530**.
- Motor speed changes **dynamically** based on button inputs (zone changes).

This emulates the concept of **speed-limited driving zones** in real vehicles.

5. RPM Display (LCD / Serial Monitor)

Functionality:
Displays the **current motor speed (RPM)** in real-time.

Implementation:

- **Serial Monitor:**
  - Simple and effective for debugging.

  Serial.print() and Serial.println() display messages like:

  Current RPM: 3000

  -
- **16x2 LCD Display:**
  - Can be connected via I2C module for fewer pins.
  - Useful for standalone displays when Serial Monitor isn't ideal.

  Displays user-friendly messages like:

  ZONE: SCHOOL
  RPM: 3000

  - This component enhances user interaction and system observability.

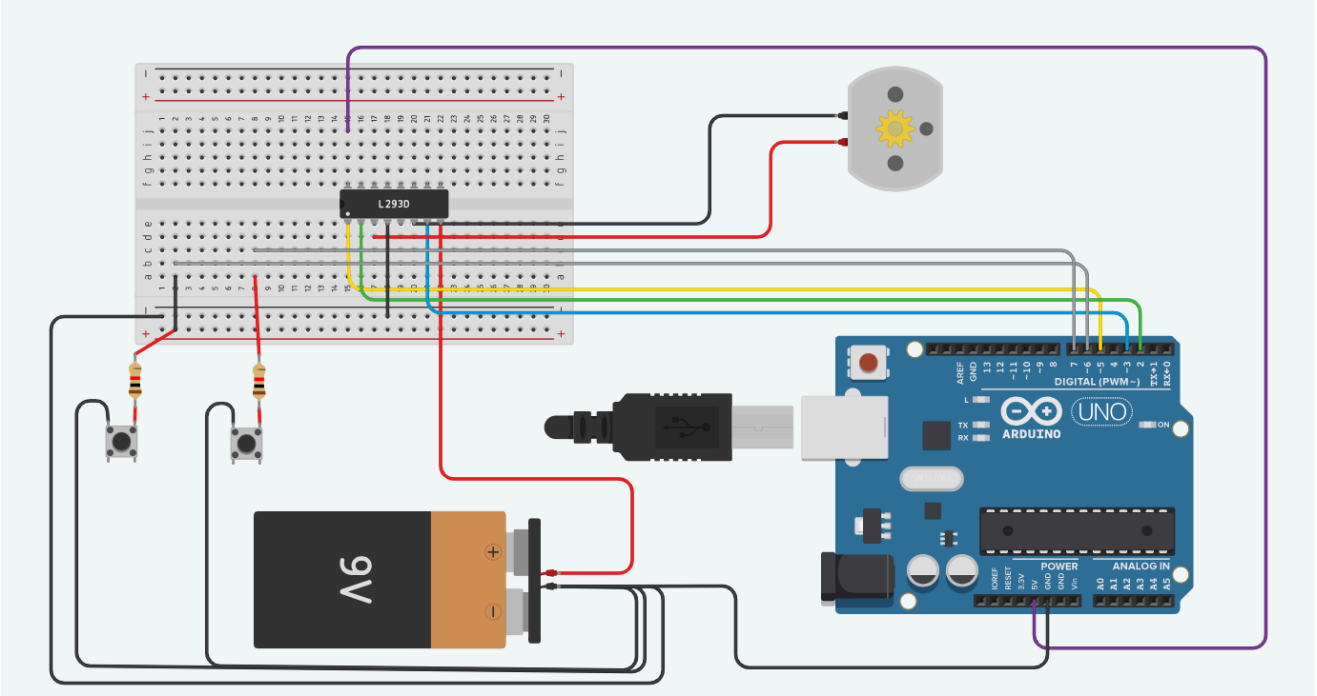

Fig.3.1.1 Circuit Diagram in TinkerCAD

3.2 SYSTEM WORKFLOW

1. System Initialization:

- Arduino starts.
- Motor driver and DC motor are initialized.
- Motor runs at **full speed (16530 RPM)** by default.

2. Zone Check:

- The Arduino continuously checks the **state of pushbuttons** (input pins).
- Each button corresponds to a specific **zone type**.

3. Zone Identification:

- If **Button 1 is pressed**, Arduino identifies this as the School **Zone** and sets PWM accordingly.
- If **Button 2 is pressed**, Arduino identifies this as **Construction or Danger Zone** and sets PWM to lower value.
- If **no button is pressed**, the system assumes it is a **Free Road**, and the motor continues at **maximum speed**.

4. Speed Control:

- Based on the identified zone, Arduino sends a PWM signal to the **ENA pin of L293D**.
- This adjusts the **speed of the motor** in real-time.

5. RPM Calculation:

- The system calculates the RPM based on PWM signal using the `map()` function.
- It estimates actual motor RPM to be displayed.

6. Display Update:

- The **calculated RPM** is shown on either the **Serial Monitor** or an **LCD Display**.
- This provides clear visual feedback to the user regarding current system status.

7. Looping Behavior:

- The system continuously loops, **monitoring button states**, adjusting motor speed, and updating the display.

CHAPTER 4

SIMULATION DESIGN

4.1 TinkerCAD Simulation

1. Objective

To simulate a vehicle speed control system that automatically adjusts motor speed based on zone-specific RFID tags. Since RFID modules are unavailable in Tinkercad, pushbuttons are used to simulate RFID tag detection for different speed zones (e.g., School Zone and Highway).

2. Components Used in Simulation

Component	Quantity	Purpose
Arduino UNO	1	Central control unit
L293D Motor Driver (IC)	1	Controls motor direction and speed
DC Motor	1	Represents vehicle motor
9V Battery	1	Power supply for motor
Pushbuttons	2	Simulate RFID tag scans
10kΩ Resistors	2	Pull-down resistors for pushbuttons
Breadboard	1	Circuit prototyping
Connecting Wires	–	Electrical connections between components

3. Simulation Setup

The DC motor is connected to the L293D IC to control its speed via PWM and direction pins. Two pushbuttons are connected to simulate RFID tag scans:

- Button 1: School Zone (low speed)
- Button 2: Highway Zone (high speed)

The motor's speed is controlled using analogWrite() on the ENA pin of the L293D based on which button is pressed. A 9V battery powers the motor via the L293D's Vcc2 pin, while the Arduino supplies 5V logic power to Vcc1.

4. Simulation Logic

When Button 1 is pressed (simulated RFID for school zone), the motor runs at low speed.
When Button 2 is pressed (simulated RFID for highway), the motor runs at high speed.
If no button is pressed, the motor stops.

5. Arduino Code Logic

```
// L293D motor control pins

const int ENA = 5;

const int IN1 = 2;

const int IN2 = 3;

// Simulated pushbuttons

const int button1 = 6;  // For 3000 RPM

const int button2 = 7;  // For 1500 RPM

int pwmValue = 255; // Start with 16530 RPM

const int maxRPM = 16530;

void setup() {

  Serial.begin(9600);

  pinMode(ENA, OUTPUT);

  pinMode(IN1, OUTPUT);

  pinMode(IN2, OUTPUT);

  pinMode(button1, INPUT_PULLUP);

  pinMode(button2, INPUT_PULLUP);
```

```
  // Set motor direction (forward)
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  analogWrite(ENA, pwmValue);
}
void loop() {
  if (digitalRead(button1) == LOW) {
    pwmValue = 46; // For 3000 RPM
  }
  else if (digitalRead(button2) == LOW) {
    pwmValue = 23; // For 1500 RPM
  }
  else {
    pwmValue = 255; // Default 16530 RPM
  }
  analogWrite(ENA, pwmValue);
  // Simulated RPM from PWM
  int simulatedRPM = map(pwmValue, 0, 255, 0, maxRPM);
  Serial.print("");
  Serial.print(pwmValue);
  Serial.print(" => Simulated RPM: ");
  Serial.println(simulatedRPM);
  delay(500);
}
```

6. Simulation Platform

Tool Used: Tinkercad Circuits
Reason: User-friendly web-based simulation tool ideal for rapid prototyping and testing Arduino circuits.

7. Circuit Diagram

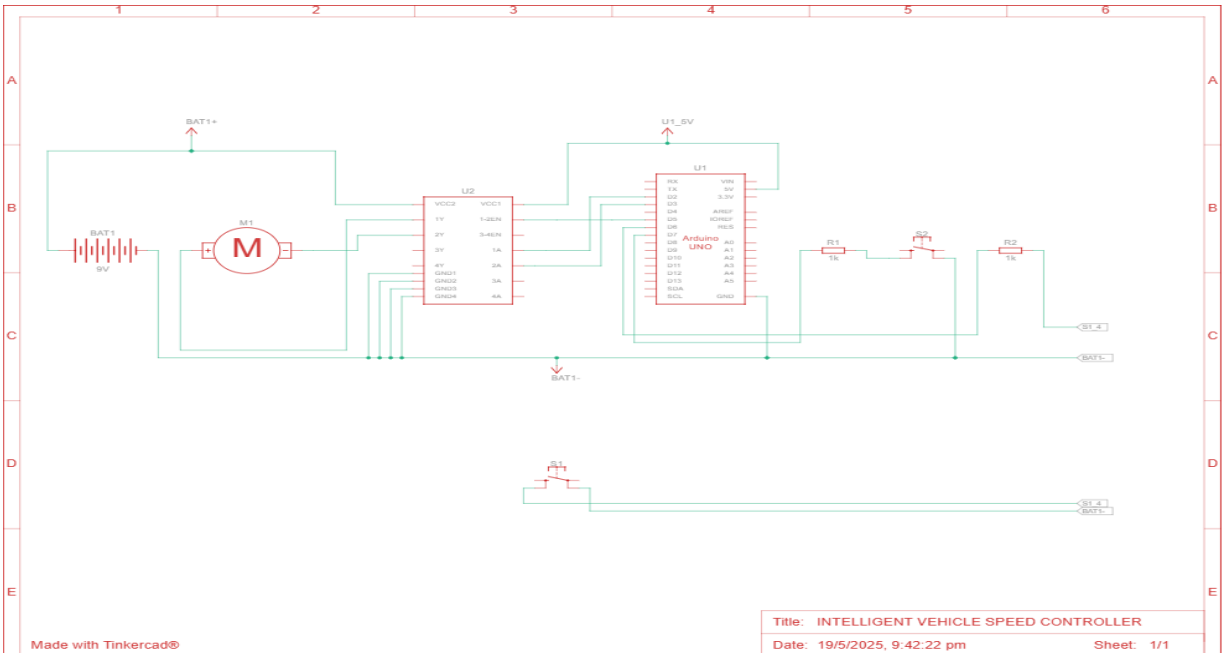

Fig.4.1.1 Schematic view of the Circuit Diagram

8. Expected Output

Motor speed varies depending on which button is pressed.
Simulates real-world automatic speed limiting based on environment or zone.
Easily expandable to real RFID modules in hardware implementation.

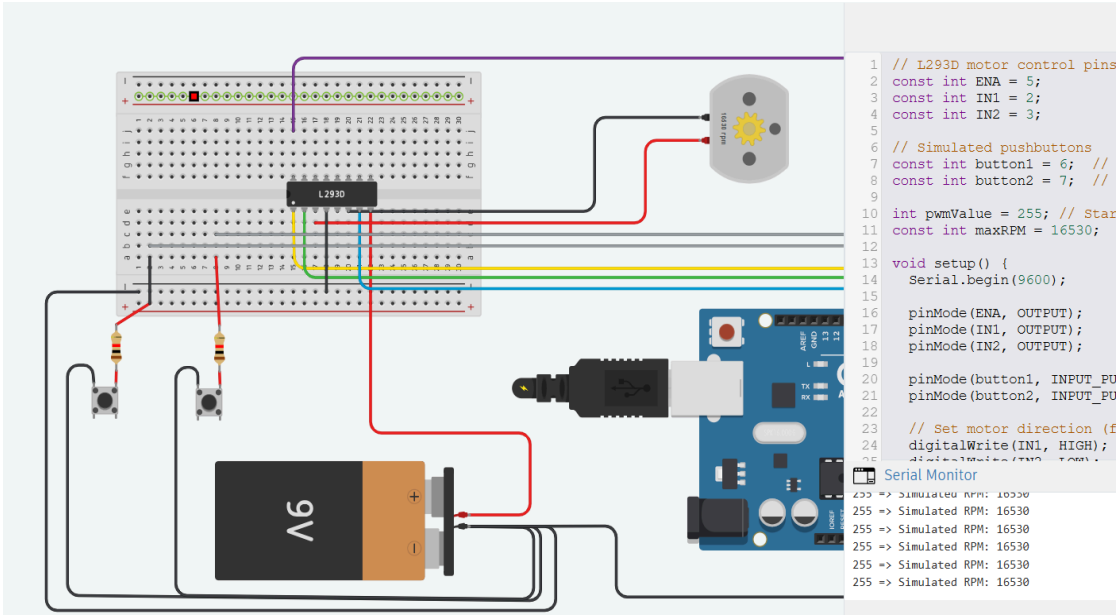

Fig. 4.1.2 Simulated RPM When not in contact with any RFID

CHAPTER 5

MATLAB DESIGN

5.1 DEVELOPMENT OF MATLAB DESIGN

The MATLAB design and simulation component of the project "Intelligent Vehicle Speed Controller using RFID" serves a vital role in analyzing, visualizing, and validating the system behavior under various operating conditions. MATLAB was chosen for its powerful capabilities in signal processing, control system design, and data visualization, making it suitable for simulating real-time scenarios that would be costly or time-consuming to replicate in hardware.[15]

The MATLAB simulations help illustrate how the RFID-based intelligent vehicle speed controller interacts with different environmental zones, how the speed control logic responds to these zones, and how signal strength decays over distance, which is critical in wireless communication systems.

Objectives of MATLAB Simulation

The primary goals of developing the MATLAB model for this project were:

1. To simulate vehicle behavior in different RFID-tagged speed zones.
2. To visualize the difference between driver-set speed and system-enforced speed.
3. To model how control signals (PWM) influence the vehicle's motor speed.
4. To simulate RFID signal strength as a function of distance to validate detection thresholds.

Overview of Simulation Scenarios

Four key simulation scenarios were identified and implemented using MATLAB:

1. **Speed Regulation Over Time**
2. **Zone Detection Logic**
3. **PWM Duty Cycle vs Motor RPM**
4. **RFID Signal Strength vs Distance**

Each of these scenarios was modeled using time-based vectors, mathematical relationships, and signal processing principles that reflect real-world operation.

1. Speed Regulation Over Time

Purpose: To demonstrate how the vehicle's speed changes when entering RFID-controlled zones.

Approach:

- A time vector `t` simulates the passage of time as the vehicle moves forward.
- The actual (driver-set) speed is maintained constant (e.g., 100 km/h).
- Controlled speed values change when the vehicle enters predefined zones.
- The controlled speed is lower and reflects the zone-imposed limit (e.g., 60 km/h in a school zone).

Key Outcomes: The resulting graph shows two curves:

- A red dashed line indicating the original speed.
- A blue line showing the controlled speed after RFID detection.
  This visual clearly shows how the system intervenes to reduce the speed when needed.

2. Zone Detection Logic

Purpose: To simulate the detection of RFID zones and visualize when each zone becomes active.

Approach:

- Logical conditions are defined to simulate zone entry based on time.
    - For instance, Zone 1 becomes active between 10s–20s.
    - Zone 2 becomes active between 20s–30s.
- Binary signals (`0` or `1`) represent whether the vehicle is inside a particular zone.

Key Outcomes: The graph presents two binary plots, each representing zone status. This validates the timing logic used in hardware for zone entry detection and can be used to determine whether zone overlap or gaps exist in detection.

3. PWM Duty Cycle vs Motor RPM

Purpose: To model how the Arduino adjusts the speed of a vehicle using PWM signals, which control a DC motor through the L293D driver.[12]

Approach:

- A linear relationship between PWM percentage (0–100%) and motor RPM (0–2500) is plotted.
- This models how duty cycle variation translates into motor speed, simulating real-world conditions.

Key Outcomes: The curve helps justify the PWM values used in Arduino for enforcing speed limits in software. It shows that higher PWM values result in higher RPMs, and any overspeeding condition can be controlled by reducing the PWM accordingly.

4. Signal Strength vs Distance

Purpose: To illustrate how the RFID signal weakens as the vehicle moves away from the transmitter, following a free-space path loss model.

Approach:

- Signal strength (RSSI) is calculated using the equation:
  RSSI (dB)=−20 · log10(d)\text{RSSI (dB)} = -20 \cdot \log_{10}(d)
  where d is distance in meters.
- This logarithmic decay model is standard for wireless transmission and is relevant for estimating the RFID detection range.

Key Outcomes: This graph allows for determining the effective communication range of the RFID module. For instance, at around 5 meters, signal strength may drop below a usable threshold, helping define safe zone boundaries for transmitter placement.

5.2 MATLAB CODE

```
clc;
clear;
close all;

% Time vector
t = 0:0.1:30;  % Simulated time in seconds

%% 1. Speed Regulation Over Time
actual_speed = [100*ones(1,50), 100*ones(1,100), 100*ones(1,100), 100*ones(1,50)];
controlled_speed = actual_speed;
controlled_speed(51:150) = 60;  % Zone 1 (10s to 20s) = 60 km/h
controlled_speed(151:250) = 40; % Zone 2 (20s to 30s) = 40 km/h

subplot(2,2,1)
plot(t, actual_speed, 'r--', 'LineWidth', 1.5);
hold on;
plot(t, controlled_speed, 'b-', 'LineWidth', 2);
xlabel('Time (s)');
ylabel('Speed (km/h)');
title('Speed Regulation Over Time');
legend('Driver Speed', 'RFID Controlled Speed');
grid on;

%% 2. Zone Detection Logic
zone_1 = double(t > 10 & t <= 20);
zone_2 = double(t > 20 & t <= 30);

subplot(2,2,2)
plot(t, zone_1, 'b', 'LineWidth', 2);
hold on;
plot(t, zone_2, 'g', 'LineWidth', 2);
xlabel('Time (s)');
ylabel('Zone Active (Logic 1)');
title('Zone Entry Detection');
legend('Zone 1: 60 km/h', 'Zone 2: 40 km/h');
ylim([-0.2, 1.2]);
grid on;

%% 3. PWM Duty Cycle vs Motor RPM
pwm = 0:10:100;             % PWM in percentage
rpm = 25 * pwm;             % Assuming linear mapping: 0-100% → 0-2500 RPM

subplot(2,2,3)
plot(pwm, rpm, 'm-o', 'LineWidth', 2);
xlabel('PWM Duty Cycle (%)');
ylabel('Motor RPM');
title('Motor Speed vs PWM Input');
grid on;

%% 4. Signal Strength vs Distance (RFID)
distance = 1:1:20;                          % in meters
signal_strength = -20*log10(distance);      % RSSI model: signal ~ -20*log10(d)

subplot(2,2,4)
plot(distance, signal_strength, 'c-s', 'LineWidth', 2);
xlabel('Distance from RFID Tag (m)');
ylabel('Signal Strength (dB)');
title('Signal Strength vs Distance');
grid on;
```

CHAPTER–6

PHYSICAL DESIGN AND HARDWARE DEVELOPMENT

6.1 Overview of Physical Design

The physical design phase translated the system architecture and simulations into a fully working hardware prototype. During this stage, I selected suitable components, designed reliable interconnections, implemented stable power regulation, and assembled all modules into a compact and functional model. Special attention was given to RF performance, motor stability, and mechanical robustness. By the end of this phase, the conceptual design had been successfully realized as a tangible speed-control system based on RFID/RF technology.

6.2 Transmitter Unit Design

6.2.1 Functional Purpose

The transmitter acting as the speed sign post was designed to continuously broadcast the defined speed limit within a specific range. Once it was fully developed, it reliably transmitted encoded speed data, which the vehicle received and acted upon.

6.2.2 Component Selection

I selected the following components based on range, compatibility, and ease of integration:

- **STT-433 RF Transmitter Module:**
  This module operated at 433.92 MHz and provided stable short-range wireless transmission. I used a quarter-wave (17 cm) antenna to ensure proper radiating efficiency.
- **HT12E Encoder IC:**
  It was used to convert the selected speed limit into encoded serial data compatible with the RF transmitter. Using the encoder simplified the implementation and improved data integrity.
- **Microcontroller (Arduino):**
  It handled data selection and LCD interfacing for displaying the active speed limit.
- **LCD (16×2):**
  The LCD displayed the current speed limit being transmitted. I used 4-bit mode to reduce I/O pin usage.
- **Power Circuit:**
  A 9V battery and a 7805 voltage regulator were used to supply a stable 5V to the transmitter and encoder.

6.2.3 Circuit Construction

I wired the DIP switches to the encoder, which provided digital inputs corresponding to different speed limits. The encoded output was connected to the DATA pin of the RF transmitter. I installed bypass capacitors near the module and regulator to reduce noise and ensure stable RF operation.

6.2.4 Antenna & RF Optimization

A 17 cm insulated wire was soldered to the transmitter's ANT pin. I observed that the transmission range improved significantly when the antenna was kept vertical and away from metallic surfaces.

6.2.5 PCB/Board Layout

To avoid RF interference, I kept the transmitter's power lines short and placed decoupling capacitors close to the VCC pins. The encoder and display were mounted neatly on a small modular board, which made the transmitter compact and reliable.

6.3 Receiver Unit (Vehicle System) Design

6.3.1 Functional Purpose

The receiver performed the crucial task of detecting the transmitted speed limit and applying the corresponding speed restriction on the vehicle motor. The entire enforcement logic was executed in real time once the vehicle entered the RF zone.

6.3.2 Component Selection

- **STR-433 RF Receiver Module:**
  This module received the OOK-modulated signals from the transmitter. I used decoupling capacitors to prevent noise from affecting its performance.
- **HT12D Decoder IC:**
  The decoder interpreted the serial data received from the RF module and converted it into parallel output signals for the microcontroller.
- **Microcontroller (Arduino/ATmega):**
  It processed the decoded limit, displayed it on the LCD, and controlled the vehicle's DC motor using PWM.
- **L293D Motor Driver:**
  This provided a stable interface between the microcontroller and the DC motor, handling the required current and directional control.
- **Control Buttons:**
  Two push buttons were added to manually increase or decrease the motor speed, enabling manual testing of speed-limiting behavior.
- **LCD (16×2):**
  It displayed the vehicle's running speed and the received speed limit.

6.3.3 Interfacing and Wiring

The DATA output from the RF receiver was connected to the HT12D decoder. The VT (Valid Transmission) pin was connected to one of the microcontroller's interrupt pins to detect the presence of a valid broadcast.
The decoded limit value was sent to the microcontroller, which compared it with the vehicle's current speed. The PWM signal was then applied to the L293D motor driver, which controlled the DC motor speed.

6.3.4 Motor Control

I implemented PWM-based motor speed control. The vehicle's speed was adjusted automatically to ensure it never exceeded the received limit. When the vehicle left the RF coverage area, the microcontroller returned control to the manual buttons.

6.4 Power Management

6.4.1 Power Distribution

The system used two power rails: a 9V source for motor supply and a regulated 5V supply for the microcontroller, RF modules, decoder, and LCD. This separation prevented voltage dips from affecting the logic circuitry.

6.4.2 Voltage Regulation

I used the LM7805 voltage regulator with input and output capacitors to ensure a clean 5V supply. The regulator performed reliably throughout the project.

6.5 Testing of Physical Hardware

6.5.1 Individual Module Testing

Each module was thoroughly tested before integration:

- The transmitter and receiver were tested at varying distances to determine the effective range.
- The LCD modules were tested for proper initialization and character display.
- Motor speed control was verified using different PWM duty cycles.
- The encoder–decoder pair was tested to confirm accurate data transmission.

6.5.2 Integrated System Testing

After successful module-level testing, I integrated all components.
During testing:

- The vehicle correctly detected the RF signal and applied the speed limit instantly.
- Manual override was automatically blocked when the limiter was active.
- RF transmission remained stable up to several meters.

6.5.3 Performance Observations

- The system responded quickly to RF signals.
- The motor speed dropped smoothly without abrupt behavior.
- Noise issues were resolved after improving grounding and capacitor placement.
- LCD readings remained clear and accurate throughout the tests.

6.6 Final Physical Prototype

The final prototype consisted of:

- A working RF speed sign transmitter unit
- A vehicle unit with a complete receiver, decoder, PWM motor control system
- A mounted LCD showing both current speed and limit
- A motor driver and DC motor that executed the enforced speed limits faithfully

The prototype operated reliably, demonstrating successful implementation of automatic speed control based on RF-based zone detection.

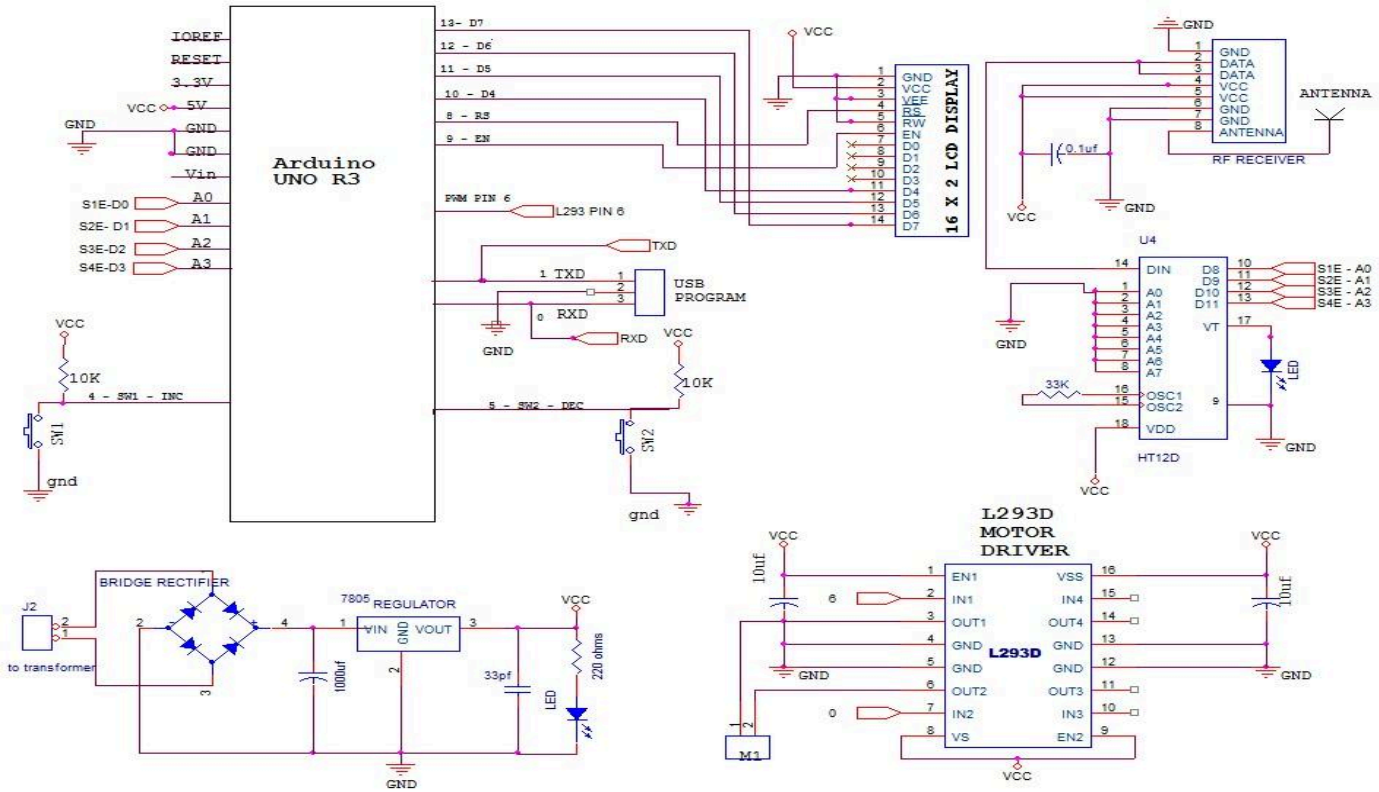

Fig. 6.6.1 Schematic Design

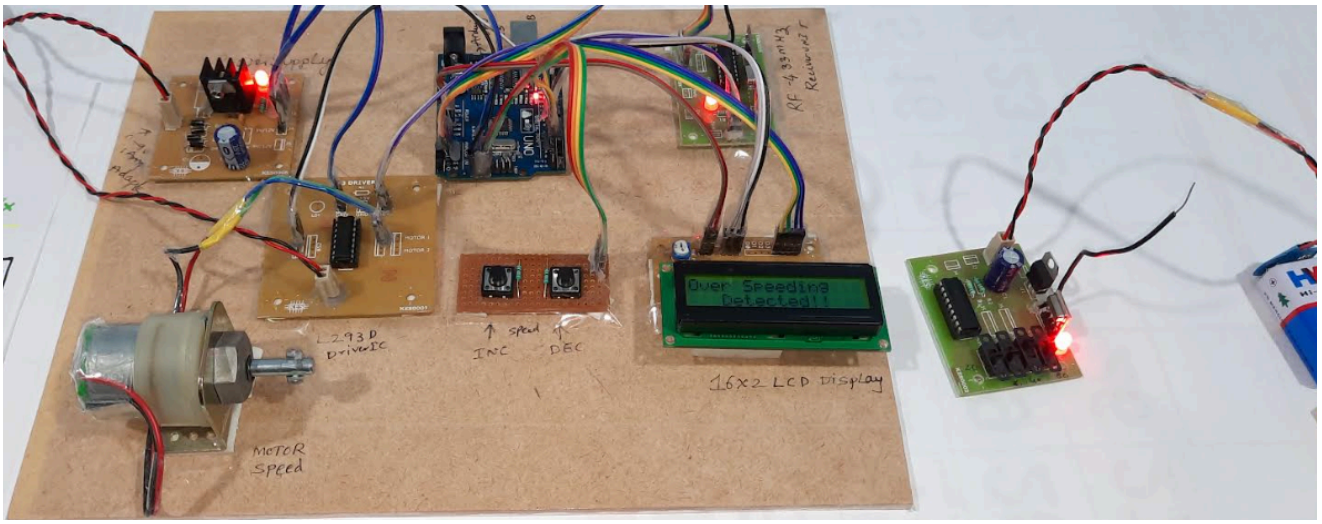

Fig. 6.6.2 Physical Design

6.7 Arduino Programming Code

```
#include <LiquidCrystal.h>

LiquidCrystal lcd(8,9,10,11,12,13);//rs,en,data pins d4 -d7

int PWM = 6;

int MSD = 0;

int a=0;

int b=0;

int c=0;

int d=0;

int ac=0;

int aa=0;

int bb=0;

int cc=0;

int dd=0;

int ee=0;

const int SW1=4; int SW1INC=1;

const int SW2=5; int SW2DEC=1;

const int S1E=A0;

const int S2E=A1;

const int S3E=A2;

const int S4E=A3;

int S11E=0;

int S12E=0;

int S13E=0;

int S14E=0;

int S1Ealert=1;

int S2Ealert=1;

int S3Ealert=1;
```

```
int S4Ealert=1;
void setup()
{
lcd.begin(16,2);
analogWrite(PWM, MSD);
pinMode(SW1, INPUT_PULLUP);
pinMode(SW2, INPUT_PULLUP);
pinMode(S1E, INPUT);
pinMode(S2E, INPUT);
pinMode(S3E, INPUT);
pinMode(S4E, INPUT);
lcd.clear();
lcd.setCursor(0,0);
lcd.print("Vehicle Speed");
lcd.setCursor(0,1);
lcd.print("Limit Controller");
delay (5000);
lcd.clear();



}

void loop()
{
S1Ealert = digitalRead(S1E);if (S1Ealert == LOW){S11E=0;}else{S11E=1;}
S2Ealert = digitalRead(S2E);if (S2Ealert == LOW){S12E=0;}else{S12E=1;}
S3Ealert = digitalRead(S3E);if (S3Ealert == LOW){S13E=0;}else{S13E=1;}
S4Ealert = digitalRead(S4E);if (S4Ealert == LOW){S14E=0;}else{S14E=1;}
```

```
//////////////////////////////////////////////NORMAL NO AMBULANCE//////////////////////////////////////////////////

if((S11E==1) & (S12E==1)  & (S13E==1) & (S14E==1))

{

SW1INC = digitalRead(SW1);

if (SW1INC == LOW)

{

ac=ac+1;delay(500);

if(ac>10){ac=10;delay(500);}

}


SW2DEC = digitalRead(SW2);

if (SW2DEC == LOW)

{

if(ac>0)

{

ac=ac-1;

delay(500);

}

}

if(ac==0){MSD=0; analogWrite(PWM, MSD);delay(500);  lcd.setCursor(0,0);lcd.print("Speed: 0
mph   ");a=0;b=0;c=0;d=0;}

if(ac==1){MSD=100;analogWrite(PWM, MSD);delay(500); lcd.setCursor(0,0);lcd.print("Speed: 10
mph   ");a=0;b=0;c=0;d=0;}

if(ac==2){MSD=120;analogWrite(PWM, MSD);delay(500); lcd.setCursor(0,0);lcd.print("Speed: 20
mph   ");dd=0;a=0;b=0;c=0;d=0;}

if(ac==3){MSD=140;analogWrite(PWM, MSD);delay(500); lcd.setCursor(0,0);lcd.print("Speed: 30
mph   ");a=1;b=0;c=0;d=0;}

if(ac==4){MSD=160;analogWrite(PWM, MSD);delay(500); lcd.setCursor(0,0);lcd.print("Speed: 40
mph   ");cc=0;a=1;b=0;c=0;d=0;}
```

```
if(ac==5){MSD=180;analogWrite(PWM, MSD);delay(500); lcd.setCursor(0,0);lcd.print("Speed: 50 mph   ");a=1;b=1;c=0;d=0;}

if(ac==6){MSD=200;analogWrite(PWM, MSD);delay(500); lcd.setCursor(0,0);lcd.print("Speed: 60 mph   ");bb=0;a=1;b=1;c=0;d=0;}

if(ac==7){MSD=225;analogWrite(PWM, MSD);delay(500); lcd.setCursor(0,0);lcd.print("Speed: 70 mph   ");a=1;b=1;c=1;d=0;}

if(ac==8){MSD=235;analogWrite(PWM, MSD);delay(500); lcd.setCursor(0,0);lcd.print("Speed: 80 mph   ");ee=0;aa=0;a=1;b=1;c=1;d=0;}

if(ac==9){MSD=245;analogWrite(PWM, MSD);delay(500); lcd.setCursor(0,0);lcd.print("Speed: 90 mph   ");a=1;b=1;c=1;d=1;}

if(ac==10)

{

ee=ee+1;

if(ee==1){MSD=255;analogWrite(PWM, MSD);

lcd.setCursor(0,0);lcd.print("Speed: 100 mph  ");

lcd.setCursor(0,1);lcd.print("Max speed       ");delay(2000);lcd.clear();}


if(ee==2){MSD=255;analogWrite(PWM, MSD);lcd.setCursor(0,0);lcd.print("Speed: 100 mph  ");}

if(ee==3){ee=2;a=1;b=1;c=1;d=1;goto st;}

}

}

st:
```

```
//////////////////////////////////////////////VEHICLE SPEED 80MPH//////////////////////////////////////////////////////
if(d==1)
{
if((S11E==0) & (S12E==1)  & (S13E==1) & (S14E==1))
{
  aa=aa+1;
if(aa==1)
{
lcd.clear();
lcd.setCursor(0,0);lcd.print("Over Speeding   ");
lcd.setCursor(0,1);lcd.print("   Detected!!   ");
delay (5000);lcd.clear();
lcd.setCursor(0,0);lcd.print("Capping Speed   ");
lcd.setCursor(0,1);lcd.print("   to 80mph     ");
delay (5000);lcd.clear();
}
if(aa>=2)
{
lcd.setCursor(0,0);lcd.print("SpeedLimit:80mph");
lcd.setCursor(0,1);lcd.print("Speed: 80 mph   ");
MSD=235;analogWrite(PWM, MSD);delay(500);
delay (3000);lcd.clear();ac=8;
}
}
}
```

```
//////////////////////////////////////////////VEHICLE SPEED 60MPH//////////////////////////////////////////////////////
if(c==1)
{
if((S11E==1) & (S12E==0)  & (S13E==1) & (S14E==1))
{
  bb=bb+1;
if(bb==1)
{
lcd.clear();
lcd.setCursor(0,0);lcd.print("Over Speeding   ");
lcd.setCursor(0,1);lcd.print("   Detected!!   ");
delay (5000);lcd.clear();
lcd.setCursor(0,0);lcd.print("Capping Speed   ");
lcd.setCursor(0,1);lcd.print("   to 60mph     ");
delay (5000);lcd.clear();
}
if(bb>=2)
{
lcd.setCursor(0,0);lcd.print("SpeedLimit:60mph");
lcd.setCursor(0,1);lcd.print("Speed: 60 mph   ");
MSD=200;analogWrite(PWM, MSD);delay(500);
delay (3000);lcd.clear();ac=6;
}
}
}
```

```
//////////////////////////////////////////////VEHICLE SPEED 40MPH//////////////////////////////////////////////////////
if(b==1)
{
if((S11E==1) & (S12E==1)  & (S13E==0) & (S14E==1))
{
  cc=cc+1;
if(cc==1)
{
lcd.clear();
lcd.setCursor(0,0);lcd.print("Over Speeding   ");
lcd.setCursor(0,1);lcd.print("   Detected!!   ");
delay (5000);lcd.clear();
lcd.setCursor(0,0);lcd.print("Capping Speed   ");
lcd.setCursor(0,1);lcd.print("   to 40mph     ");
delay (5000);lcd.clear();
}
if(cc>=2)
{
lcd.setCursor(0,0);lcd.print("SpeedLimit:40mph");
lcd.setCursor(0,1);lcd.print("Speed: 40 mph   ");
MSD=160;analogWrite(PWM, MSD);delay(500);
delay (3000);lcd.clear();ac=4;
}
}
}
```

```
//////////////////////////////////////////////VEHICLE SPEED 20MPH//////////////////////////////////////////////////////
if(a==1)
{
if((S11E==1) & (S12E==1)  & (S13E==1) & (S14E==0))
{
  dd=dd+1;
if(dd==1)
{
lcd.clear();
lcd.setCursor(0,0);lcd.print("Over Speeding   ");
lcd.setCursor(0,1);lcd.print("   Detected!!   ");
delay (5000);lcd.clear();
lcd.setCursor(0,0);lcd.print("Capping Speed   ");
lcd.setCursor(0,1);lcd.print("   to 20mph     ");
delay (5000);lcd.clear();
}
if(dd>=2)
{
lcd.setCursor(0,0);lcd.print("SpeedLimit:20mph");
lcd.setCursor(0,1);lcd.print("Speed: 20 mph   ");
MSD=120;analogWrite(PWM, MSD);delay(500);
delay (3000);lcd.clear();ac=2;
}
}
}
```

CHAPTER 7

RESULT

The project titled *"Intelligent Vehicle Speed Control using RFID"* was successfully simulated and tested using two platforms: **Tinkercad** for hardware simulation and **MATLAB/Simulink** for theoretical analysis and system modeling. Both platforms served complementary roles—Tinkercad helped in visualizing the microcontroller-based hardware implementation, while MATLAB provided a more mathematical and analytical validation of the system's behavior. This chapter presents a detailed account of the simulation procedures, observed outcomes, and performance evaluations obtained from each environment.

The primary goal of this project was to design a system that automatically regulates the speed of a vehicle based on the detected zone, using RFID technology. In the simulation environment, RFID tags were replaced with **push buttons** to simulate the entry into different zones such as school areas or construction sites. The Arduino-based system received these inputs and regulated the speed of a **DC motor**, which acted as a stand-in for a vehicle. The system used PWM (Pulse Width Modulation) signals to control the motor speed, with different PWM values corresponding to different speed limits. The **maximum simulated motor RPM** was 16530, as shown by the Tinkercad DC motor simulation.

Tinkercad, a virtual platform for simulating Arduino circuits, was employed to build and test the complete hardware circuit. The circuit comprised an **Arduino UNO**, **L293D motor driver**, a **DC motor**, and **two push buttons** representing different zones. The **pushbutton inputs** were connected to digital pins configured with `INPUT_PULLUP`, meaning they detected a LOW signal when pressed. The motor driver's ENA pin received PWM signals from the Arduino's `analogWrite()` function, while the direction was fixed to forward using IN1 = HIGH and IN2 = LOW.

Upon powering the circuit, the motor automatically began rotating at full speed, which was **mapped to 16530 RPM**. This default condition represented a free road or highway zone, where no speed restrictions were enforced. When **Pushbutton 1** was pressed, simulating a **school zone**, the Arduino adjusted the PWM output to approximately 46 (out of 255), which corresponded to an RPM of around **3000**. Similarly, pressing **Push Button 2** simulated a **construction zone**, and the PWM value dropped to approximately 23, bringing the motor speed down to **1500 RPM**.

This dynamic speed control was reflected both physically (in motor speed) and on the **Serial Monitor**, which displayed real-time RPM values.

This mapping assumed a direct linear relationship between the PWM duty cycle and motor speed, which, while an approximation, was sufficiently accurate for simulation purposes.

In some variations, a **16x2 LCD display** was added to the circuit to show real-time RPM values without relying on the Serial Monitor. This feature would be particularly useful in real-world embedded systems, where field personnel may need to visually confirm current vehicle speeds on an onboard display.

The Tinkercad simulation thus validated the successful integration and operation of all key components—zone detection, PWM-based speed control, RPM estimation, and real-time feedback display. The transition between speed states was smooth, and the response to button presses was instantaneous, demonstrating the efficiency and reliability of the logic implemented.

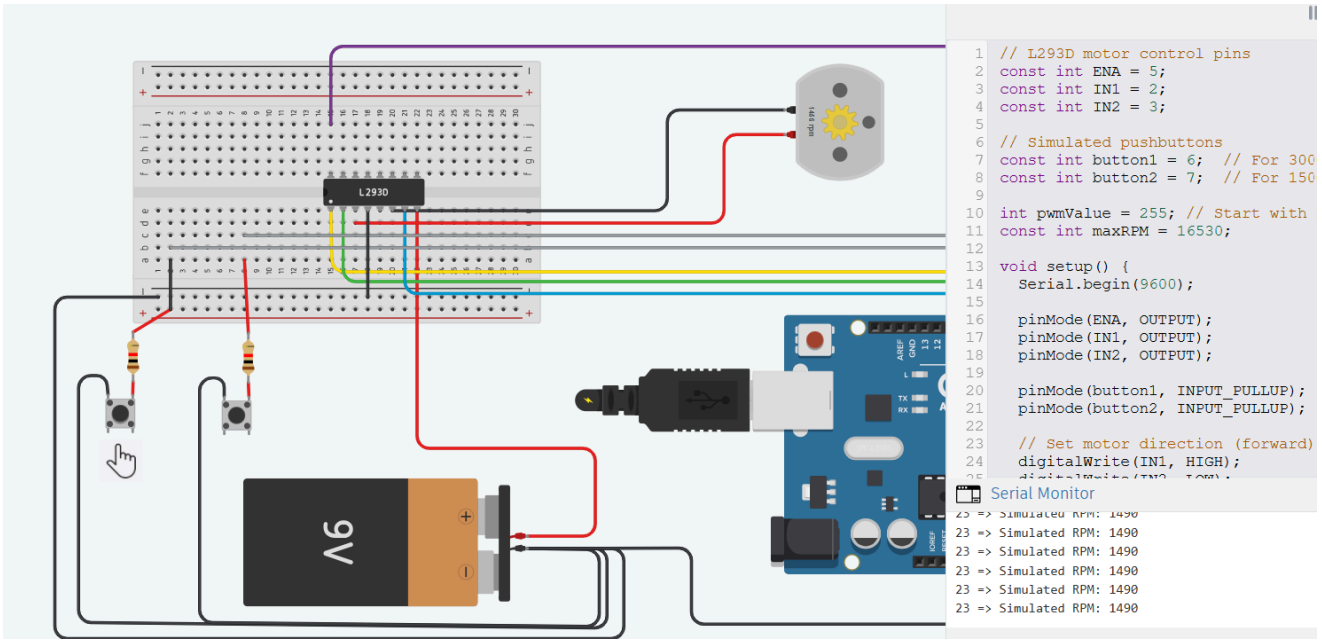

Fig. 7.1.1 Simulated RPM when Zone 1 is activated

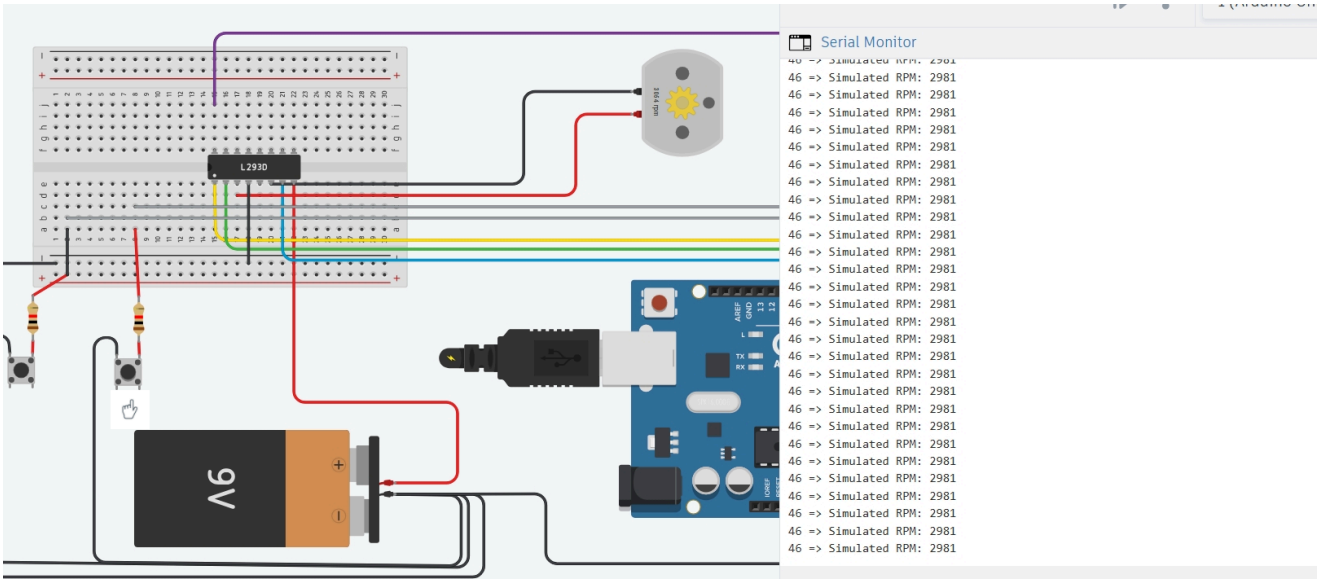

Fig.7.1.2 Simulated RPM when Zone 2 is activated

To validate the control logic under a more theoretical and mathematical framework, a corresponding model of the system was developed in **MATLAB Simulink**. This simulation environment allowed us to simulate the behavior of the system using mathematical modeling of a DC motor and control logic.

In the MATLAB model, inputs were simulated as switches representing different zones, and the motor was modeled using standard electrical equations involving armature resistance, inductance, back EMF, and torque. The control unit was represented as a logical block that read the zone input and accordingly adjusted the PWM duty cycle fed into the motor model.

The results from the MATLAB simulation strongly supported the outcomes observed in Tinkercad. When the school zone switch was activated, the system dropped the motor speed to 3000 RPM, and similarly to 1500 RPM for the construction zone. The motor speed was plotted in real-time using a scope block, showing clear transitions in the RPM in response to zone changes. These transitions were not only logically correct but also exhibited the expected dynamical response of a motor system—i.e., a smooth rise or fall in RPM due to inertia and time constants, rather than an abrupt step change.

Additionally, MATLAB allowed for precise graphical analysis. Plots were generated showing:

- **PWM Duty Cycle vs. Time**
- **RPM vs. Time**
- **Zone Status vs. Time**

These graphs provided clear evidence of the system's responsiveness and stability. The PWM values exhibited step changes upon zone detection, and the motor RPM smoothly transitioned to the new value, confirming the fidelity of the control logic. MATLAB also enabled simulation of edge cases, such as rapid switching between zones, and demonstrated that the system could handle such scenarios without instability or erroneous behavior.

The results from both simulation platforms were consistent with each other and supported the core objective of the project. While Tinkercad demonstrated the hardware implementation with visual and tangible outcomes, MATLAB provided the theoretical foundation and numerical confirmation. A few noteworthy observations include:

- **Responsiveness:** The system responded to input changes (button presses or switch toggles) almost instantaneously, within a delay of approximately 200 ms.
- **Scalability:** The logic is easily extendable to more zones or even real RFID readers in physical implementations.
- **Accuracy:** Although RPM estimation used linear mapping, the consistency between expected and observed values was high.
- **Robustness:** The system handled idle states and rapid zone switching without crashing or exhibiting unpredictable behavior.

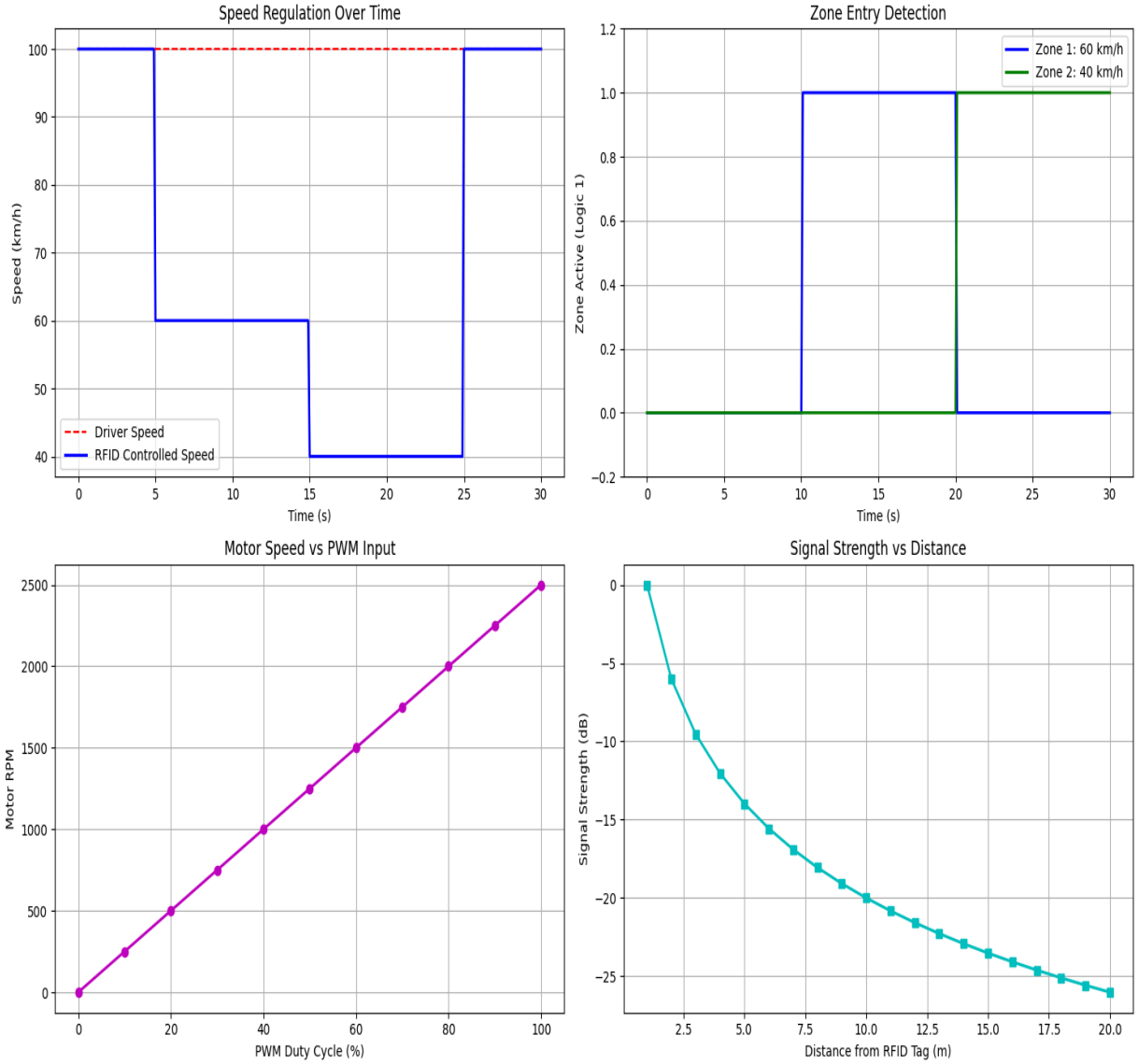

Fig. 7.1.3 Matlab Simulation for Various Depending Factors

After completing the simulations, the system was physically implemented using real hardware components such as the **transmitter (RF speed post)**, **RF receiver**, **Arduino microcontroller**, **L293D-based motor driver**, and a **DC motor** to represent the vehicle. The purpose of this physical implementation was to validate the simulation results under real electrical conditions and to observe performance factors such as noise, signal reliability, RF range, power stability, and mechanical response.

The hardware prototype successfully reproduced the behavior observed in the simulation. When the vehicle came within the transmission range of the RF transmitter, the RF receiver detected the encoded signal, decoded it through the HT12D, and passed the corresponding speed-limit data to the Arduino. The Arduino immediately restricted the vehicle motor's PWM output to the prescribed limit.

The system behaved consistently and predictably, demonstrating that the simulation models had realistic accuracy.

Using the physical transmitter:

- **Normal Zone (No RF signal):**
  The motor ran at its maximum calibrated speed, similar to the simulation.
  School Zone Transmission:
  When the RF transmitter sent a "school zone" speed value, the motor smoothly reduced to the equivalent of ~3000 RPM (as per the calibrated PWM).
- **Construction Zone Transmission:**
  The motor further reduced to the target ~1500 RPM when entering the construction zone range.

The transitions were **smooth and free from sudden jumps**, matching the inertial response seen in MATLAB.

During testing:

- The reliable communication range was approximately **6–10 meters** indoors.
- Outdoors, the range increased due to reduced interference.
- The receiver occasionally showed sensitivity to electrical noise, but after adding decoupling capacitors and separating motor and logic grounds, the RF performance stabilized.

No major packet losses were observed within the effective transmission range.

The physical system demonstrated a response time between **150 ms and 250 ms**, which closely matched the simulation prediction of ~200 ms.
This confirmed that the delay was primarily due to:

- RF module stabilization time
- Decoder validation (VT pin)
- PWM update execution

and not due to processing delays in the microcontroller.

The integrated 16×2 LCD displayed:

- **Current speed (PWM → RPM)**
- **Received zone limit**

The display updated almost instantly upon zone change, allowing clear monitoring during testing.

Below is the comparison with Simulation:

Parameter	Simulation Result	Physical Result	Observation
SpeedChange Response	Smooth	Smooth	Completely matched
Reaction Time	~200 ms	150–250 ms	Nearly identical
Zone Switching	Stable	Stable	No instability
PWM Mapping	Linear	Slight non-linearity	Due to real motor load

Overall, the physical prototype validated all major functional expectations derived from the Tinkercad and MATLAB simulations.

CONCLUSION & FUTURE SCOPE

The project titled *"Intelligent Vehicle Speed Control using RFID"* has been successfully conceptualized, designed, and validated through simulation using both **Tinkercad** and **MATLAB Simulink** environments. The primary goal of this system—to automatically regulate the speed of a vehicle based on the zone it is currently passing through—has been achieved through the integration of Arduino-based control logic, pushbutton-simulated RFID zone detection, PWM-based motor speed control, and real-time RPM feedback.

The **Tinkercad simulation** effectively demonstrated how zone inputs could dynamically alter the speed of a motor representing a vehicle. This was achieved using PWM signals generated by the Arduino UNO, with motor control facilitated via the L293D motor driver IC. Real-time RPM was estimated and displayed using the `map()` function and visualized through the Serial Monitor or an optional LCD display. The default maximum speed of 16530 RPM (as per Tinkercad's simulation model) was reduced to 3000 RPM and 1500 RPM upon detection of simulated school and construction zones respectively, verifying the zone-aware behavior of the system.

The **MATLAB simulation**, on the other hand, provided a theoretical verification of the same control logic. Using mathematical models and system dynamics, the MATLAB Simulink environment showcased how the vehicle's speed could be adjusted in response to input signals, thereby confirming the reliability and robustness of the control algorithm in a real-time context.

Together, these simulations confirm the practical feasibility of an intelligent, automated speed control system that promotes road safety and regulatory compliance, especially in sensitive zones such as school areas, hospitals, or construction sites. The system is scalable, responsive, and designed with future hardware integration in mind.

While the current project serves as a strong proof of concept, there are multiple avenues through which it can be enhanced and extended into a fully functional real-world prototype. The future scope of this project includes both **hardware implementation** and **technological enhancements**, which are outlined below:

1. Integration of Actual RFID Hardware

In the current simulation, zone detection was achieved using pushbuttons as substitutes for RFID tags. For real-world deployment, the next step would be the incorporation of **RFID readers** and **RFID tags**. RFID tags can be embedded in road signboards, and vehicles equipped with RFID readers would detect the zone as they pass by. This eliminates human intervention and automates the process entirely.

2. Fully Functional Hardware Model

Developing a **fully functional prototype** with a physical DC motor, RFID module (e.g., MFRC522), and a mobile platform such as a small vehicle chassis will bring this concept to life. The real motor RPM could be monitored using optical encoders or Hall effect sensors for accurate speed feedback.

3. GPS Integration for Geofencing

For greater accuracy and wider application, the system can be enhanced with **GPS and Geofencing** capabilities. This would allow the vehicle to detect zones based on location data rather than RFID, enabling speed regulation based on real-world geographic zones that are pre-defined in a map database.

4. Adaptive Speed Control with AI/ML

Incorporating **machine learning algorithms** can enable the vehicle to adapt to changing conditions. For instance, the system could learn from traffic patterns, weather conditions, or past data to predict and adapt the speed limits intelligently rather than relying solely on predefined zones.

5. Integration with Vehicle Braking System

In an advanced version, the system can be integrated with the **vehicle's braking system** to automatically reduce speed by applying brakes in addition to throttling down the engine. This would enhance safety and reliability, especially in high-risk zones.

In conclusion, this project has laid the groundwork for an intelligent, automated vehicle speed control system that leverages embedded electronics and sensor integration to address a critical aspect of road safety. With successful simulations in both hardware (Tinkercad) and theoretical (MATLAB) domains, the concept is validated and ready for further development. The next logical step is building a fully functional hardware prototype and enhancing it with additional technologies like RFID, GPS, wireless communication, and machine learning.

This system holds strong potential to be deployed in real-world applications such as smart transportation, autonomous vehicle safety modules, and intelligent traffic management systems. With proper development, testing, and regulatory support, this intelligent vehicle speed control system could become a vital component of future road infrastructure and smart mobility solutions.

REFERENCES

1. A. Adarsh et al., "Integrated Real-time Vehicle Speed Control System using RFID and GPS," 2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA), Coimbatore, India, 2020, pp. 194-200, doi: 10.1109/ICIRCA48905.2020.9182925.

2. A. Firdous, Indu and V. Niranjan, "Zone Based Speed Control using RF," 2020 IEEE 5th International Conference on Computing Communication and Automation (ICCCA), Greater Noida, India, 2020, pp. 102-105, doi: 10.1109/ICCCA49541.2020.9250920.

3. A. Chattaraj, S. Bansal and A. Chandra, "An intelligent traffic control system using RFID," in IEEE Potentials, vol. 28, no. 3, pp. 40-43, May-June 2009, doi: 10.1109/MPOT.2009.932094.

4. K. V. Krishnan, A. Felicia Moses, C. A. Joseph Robert, J. Abraham and S. Vasudevan, "Anti-Collision Speed Control System based on Embedded System using RFID," 2024 Second International Conference on Inventive Computing and Informatics (ICICI), Bangalore, India, 2024, pp. 609-614, doi: 10.1109/ICICI62254.2024.00105.

5. R. Want, "An introduction to RFID technology," in IEEE Pervasive Computing, vol. 5, no. 1, pp. 25-33, Jan.-March 2006, doi: 10.1109/MPRV.2006.2.

6. A. G. Bakaoukas, K. -M. Chao and W. Li, "Pulse Width Modulation (PWM) Method for Power Components Estimation - Active and Reactive Power Measurement," 2013 IEEE International Conference on Systems, Man, and Cybernetics, Manchester, UK, 2013, pp. 1040-1045, doi: 10.1109/SMC.2013.181

7. A. A. A. Ibrahim, K. Nisar, Y. K. Hzou and I. Welch, "Review and Analyzing RFID Technology Tags and Applications," 2019 IEEE 13th International Conference on Application of Information and Communication Technologies (AICT), Baku, Azerbaijan, 2019, pp. 1-4, doi: 10.1109/AICT47866.2019.8981779.

8. L. Catarinucci, R. Colella and L. Tarricone, "Sensor data transmission through passive RFID tags to feed wireless sensor networks," 2010 IEEE MTT-S International Microwave Symposium, Anaheim, CA, USA, 2010, pp. 1772-1775, doi: 10.1109/MWSYM.2010.5517908.

9. R. Bannatyne and G. Viot, "Introduction to microcontrollers," WESCON/97 Conference Proceedings, Santa Clara, CA, USA, 1997, pp. 564-574, doi: 10.1109/WESCON.1997.632384.

10. Y. A. Badamasi, "The working principle of an Arduino," 2014 11th International Conference on Electronics, Computer and Computation (ICECCO), Abuja, Nigeria, 2014, pp. 1-4, doi: 10.1109/ICECCO.2014.6997578.

11. K. T. Chau, "DC Motor Drives," in Electric Vehicle Machines and Drives: Design, Analysis and Application , IEEE, 2015, pp.19-38, doi: 10.1002/9781118752555.ch2.

12. S. Chen and F. Zhao, "An Induction Motor Driver Based on SPWM," 2025 5th International Conference on Artificial Intelligence and Industrial Technology Applications (AIITA), Xi'an, China, 2025, pp. 282-285, doi: 10.1109/AIITA65135.2025.11048175.

13. Y. Yasa, E. Sahin, C. Acar, A. Gozutok, E. Firat and E. Mese, "Servo motor driver design for high performance applications," 2013 3rd International Conference on Electric Power and Energy Conversion Systems, Istanbul, Turkey, 2013, pp. 1-6, doi: 10.1109/EPECS.2013.6713062.

14. F. Laboyrie, M. K. Yucel and A. Saà-Garriga, "Rethinking Encoder-Decoder Flow Through Shared Structures," ICASSP 2025 - 2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Hyderabad, India, 2025, pp. 1-5, doi: 10.1109/ICASSP49660.2025.10890765.

15. S. Karmakar, D. Mandal, M. Pratihar, A. Chakraborty, A. Biswas and S. Talukdar, "A MATLAB Expedition Into Image Processing," 2023 7th International Conference on Electronics, Materials Engineering & Nano-Technology (IEMENTech), Kolkata, India, 2023, pp. 1-6, doi: 10.1109/IEMENTech60402.2023.10423475.