CHAPTER 1

INTRODUCTION

In the present era, the role of automation in daily life has become increasingly significant. With the advancement of the Internet of Things (IoT), many conventional systems are being redesigned to function automatically, reducing human intervention and increasing efficiency. This project is based on the implementation of automation in two real-time scenarios using Arduino Uno and basic electronic components.

The first system focuses on the development of a prototype that simulates a radar-based detection mechanism. An ultrasonic sensor mounted on a servo motor is programmed to rotate across a defined range and measure the distance of any object that falls within its sensing range. The collected data is sent to the Processing IDE, where it is displayed visually in the form of a radar interface. This offers a basic yet effective method of object detection that could be expanded for use in surveillance, security monitoring, or autonomous navigation.

The second system demonstrates the automation of a railway gate. In many places, railway crossings are still operated manually, which can lead to human errors and accidents. To address this issue, the project integrates an ultrasonic sensor to detect the arrival of a train. Once the presence of a train is identified within a certain distance, the Arduino activates a servo motor that closes the gate. After the train passes, the gate reopens automatically. This kind of automation can enhance safety at crossings and reduce the dependence on human operators.

Both systems are designed using low-cost, widely available hardware, making them suitable for implementation in regions where resources are limited. The project showcases how microcontroller-based automation can be applied to improve safety and efficiency in everyday scenarios.

1.1 Problem Statement

The need for simple, cost-effective solutions that can perform basic object detection and provide automated control is evident. A system that can automatically detect an approaching train and operate the gate, as well as one that can scan and visualize surrounding objects using basic components, would be highly beneficial for safety and monitoring applications. The goal is to reduce human involvement in critical decision-making processes by introducing reliable automation that can function with minimal oversight.

1.2 Objectives and Scope

The main objective of this project is to implement IoT-based automation systems using Arduino Uno, ultrasonic sensors, and servo motors to enhance safety and efficiency in real-world scenarios. The project aims to build a prototype capable of detecting surrounding objects and visualizing them through a radar-style interface using Processing software. This offers a basic yet effective solution for real-time area monitoring and object detection, which can be further developed for use in surveillance, robotics, or industrial safety systems.

Another key objective is to develop an automatic railway gate control system that functions without human intervention. The system should be capable of identifying the arrival of a train using sensor input and controlling the opening and closing of the gate using a servo motor. This ensures timely and reliable operation of railway crossings, thereby minimizing the risk of accidents.

The project also focuses on achieving real-time response through efficient sensor-data communication and control logic. Furthermore, it aims to offer an affordable and scalable solution using open-source tools and low-cost hardware components. In doing so, the project demonstrates how embedded systems and IoT technologies can be applied to solve everyday problems in transportation and monitoring.

Ultimately, the project aims to reduce human error, improve operational safety, and contribute to the ongoing movement towards intelligent, automated infrastructure.

CHAPTER 2

LITERATURE SURVEY

2.1 Introduction

Several research studies and project implementations have been carried out in the field of automation using microcontrollers and sensor-based systems. The advancement of open-source platforms such as Arduino has enabled students, researchers, and developers to design and prototype automation systems with ease and affordability.

In the study titled *"Automatic Railway Gate Control Using Arduino and IR Sensors"* by S. Kumari et al., an Arduino-based system was proposed where IR sensors were used to detect the arrival of a train and control the gate automatically. The project aimed to prevent accidents at unmanned railway crossings by reducing the dependency on human gatekeepers. Although the system proved effective, it lacked distance-based sensing which could be influenced by ambient light and obstacles.

Another related work, *"Radar System Simulation Using Ultrasonic Sensors and Arduino"* by M. Rahman et al., implemented a basic radar interface using a servo motor and an ultrasonic sensor. The project successfully demonstrated object detection within a certain range and displayed the output in a radar-like format on the Processing IDE. This helped in understanding how real-time scanning and visual feedback can be achieved using basic electronics and coding.

In *"Automated Safety System for Railway Crossing using IoT"*, researchers used a combination of sensors and GSM modules to send alerts and control railway gates. The use of IoT made it possible to extend the functionality to remote monitoring, although the complexity and cost of the system increased significantly.

The current project integrates the strengths of these previous works by combining a radar-based object detection system with a distance-based automatic railway gate control system using ultrasonic sensors and servo motors. By utilizing Arduino Uno as the core controller and the Processing IDE for visualization, the project achieves real-time automation and display with minimal hardware requirements. The use of ultrasonic sensing overcomes many limitations of traditional IR-based systems, offering more precise and reliable detection.

This literature review indicates that while several isolated systems exist for radar simulation or gate automation, combining these concepts into a single, dual-purpose project offers greater learning and real-world application value.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 Introduction

Before designing and implementing any automation project, it is essential to perform a proper requirement analysis. This involves identifying the necessary hardware and software components, understanding the functional expectations of the system, and ensuring that the selected components are compatible with each other. Requirement analysis forms the foundation for the successful development and testing of the project.

For this project, the requirement analysis was carried out by studying the operational flow of both the radar detection system and the automatic railway gate mechanism. Each system demands accurate sensing, timely actuation, and reliable communication between the components. Therefore, the selection of sensors, actuators, microcontrollers, and interface software is made carefully to achieve desired performance with minimum complexity and cost.

The system also needs to operate in real-time, respond promptly to the detection of objects or trains, and display output either visually (in the case of radar scanning) or physically (in the case of gate movement). Based on these needs, the components were finalized to suit the scope and objectives of the project.

3.2 Existing System

In the current scenario, railway crossing gates in many regions, especially in rural and semi-urban areas, are operated manually by personnel who monitor train movements and control the gates accordingly. This manual operation system is highly dependent on human attention and efficiency. Any delay, negligence, or miscommunication can lead to serious accidents at the crossing, posing risks to both pedestrians and vehicles.
Additionally, in the domain of area monitoring and object detection, high-end radar systems are used for applications such as surveillance, security, and navigation. These systems, however, are often expensive, complex, and require advanced infrastructure, making them unsuitable for basic prototype development or small-scale practical use.
There have been some attempts to automate railway gate systems using IR sensors or simple relay-based controls. However, these systems are not very reliable in varying environmental conditions and often lack accuracy and responsiveness. Similarly, most ultrasonic-based object

detection systems are standalone and do not provide visual feedback or scanning capabilities in a radar-like format.

Therefore, the existing systems, though functional in certain capacities, fail to offer an integrated, low-cost, and reliable solution for automated gate control and real-time object detection with visual feedback. This project aims to overcome these limitations by implementing a combined system using basic sensors, Arduino Uno, and the Processing IDE for radar visualization, making the solution accessible, educational, and effective.

3.3 Proposed System

The proposed system introduces a dual-function IoT-based automation solution that integrates two independent modules: a radar detection system and an automatic railway gate control system. Both modules are developed using the Arduino Uno microcontroller, ultrasonic sensors, and servo motors to deliver accurate and real-time responses.

In the radar detection system, an ultrasonic sensor mounted on a servo motor scans a predefined range by rotating in degrees. As it detects objects in its path, the distance data is collected and sent to a computer via serial communication. This data is then visualized using the Processing IDE, creating a radar-style display that shows object position and distance in real-time. This simulation provides a basic but effective model of how radar detection works and can be used for educational and monitoring purposes.

For the automatic railway gate control system, another ultrasonic sensor is used to detect the approach of a train by sensing any large object within a specific range. When the train is detected, the Arduino triggers a servo motor to close the gate automatically. Once the object passes and the path is clear, the gate opens again. This mechanism eliminates the need for manual gate operation, thereby increasing safety and reducing the risk of human error.

The proposed system offers a simple, cost-effective, and scalable solution that utilizes minimal hardware while demonstrating the core concepts of automation, sensing, and real-time control. It is designed in such a way that both systems can function independently or be integrated as part of a broader safety and monitoring infrastructure.

3.4Hardware Componets

The hardware components required to build the system include:

1. **Arduino Uno:**

   A popular microcontroller board that acts as the brain of the system. It processes sensor data, controls components like motors, and communicates with a computer. It's beginner-friendly, programmable via USB, and supports various sensors and modules.

2. **Ultrasonic Sensor (HC-SR04):**

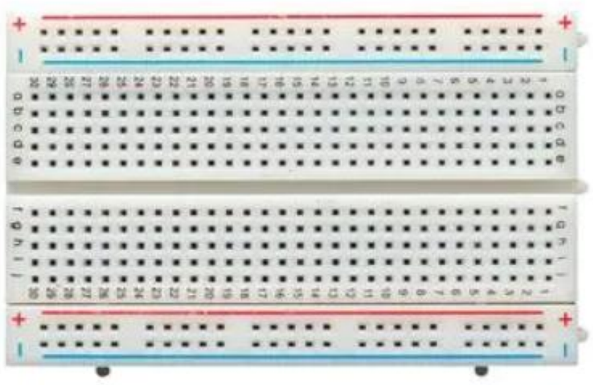
   A sensor that uses sound waves to measure the distance to an object. It emits ultrasonic pulses and calculates the distance based on the echo time, making it perfect for object detection and proximity sensing in the radar system.

3. **Micro Servo (9g:**

   A small motor with precise angular control, commonly used for rotating objects to specific positions. In the radar system, it rotates the ultrasonic sensor to scan the environment, enabling 180-degree coverage.

4. **Breadboard:**

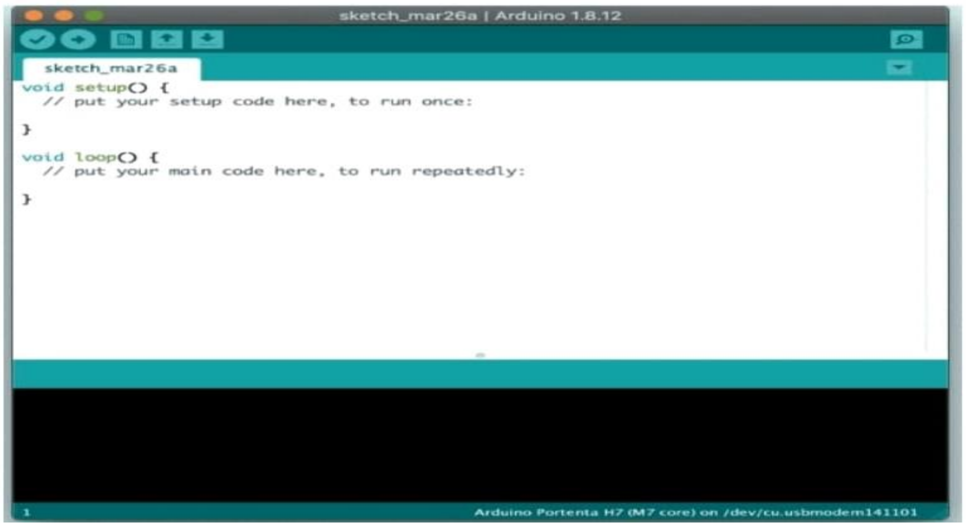
   A prototyping tool used to build and test circuits without soldering. It allows components to be connected and rearranged easily, making it ideal for temporary setups and experimenting with circuit designs.



5. **Jumper cables:**

   Wires used to make temporary electrical connections between components on a breadboard or directly with microcontroller pins. They come in male and female ends, allowing flexible connections for quick testing and debugging.

3.5 Software Requirements


sketch_mar26a | Arduino 1.8.12
sketch_mar26a
Arduino Portenta H7 (M7 core) on /dev/cu.usbmodem141101

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Fig 3.6 Software Requirements

David Mellis developed Arduino IDE software at the Ivrea Interaction Design Institute, an easy tool for fast prototyping. Arduino IDE is an open-source cross-platform application. It supports C, and C++ languages in Windows, Mac, and Linux operating systems. It is mainly used to write and upload programs to Arduino or Genuine hardware. It contains a text editor to write code, a message area, a text console, a toolbar with buttons for common functions, and a series of menus. Here programs are called Sketches. These sketches are written in the editor area and then saved with the. Ion extension. Sketches are then converted into Hexfiles that can be uploaded to hardware. If the program has an error in it, then it will show an error in that line. Output can be seen in the serial monitor. This displays the serial sent from the Arduino board over a USB or serial connector. To send data to the board, enter text and click on the "send" button or press enter. Choose the baud rate from the drop-down menu that matches the rate passed to Serial. Begin in your sketch. The source code for the Arduino IDE is released under the General Public License version 2. It has 3 sections: Menu bar, Text editor, and Output pane. It is an open-source Arduino software that makes it easy to write code and upload it to the board. This software can be used with any Arduino board. It is a cross-platform application that is written in functions from C and C++.

CHAPTER 4:
SOFTWARE REQUIREMENT SPECIFICATION

4.1 Introduction

Software plays a vital role in the development and execution of embedded and IoT-based systems. In this project, the software components are responsible for handling sensor data, processing commands, controlling hardware actions, and providing real-time visual feedback to the user. The integration of software with hardware ensures the system operates efficiently, accurately, and as intended. For the Radar Detection and Automatic Railway Gate Control project, the Arduino IDE is used for writing and uploading code to the Arduino Uno microcontroller, enabling it to read sensor inputs and control servo motors accordingly. Additionally, the Processing IDE is employed to create a visual radar interface that graphically displays the distance and angle of detected objects. These software tools provide an easy-to-use, flexible platform that supports the overall automation goals of the system.

4.1.1 Purpose

The purpose of the software in this project is to control and coordinate the actions of the hardware components while ensuring accurate detection, decision-making, and user interaction. The Arduino IDE is used to program the microcontroller, allowing it to process sensor inputs and control servo motor movements effectively for both radar scanning and automatic gate operations. Additionally, the Processing IDE is employed to visualize radar data, offering a clear and interactive display of detected objects in real time. This combination of software ensures the system functions smoothly, enhances automation, and provides users with meaningful feedback, making the overall setup intelligent, responsive, and easy to monitor.

4.1.2 Project Scope

The scope of this project encompasses the development of an intelligent and automated system that combines radar-based object detection and automatic railway gate control using Arduino technology. The system aims to demonstrate how low-cost, easily available components like ultrasonic sensors, servo motors, and microcontrollers can be integrated with open-source software to build functional prototypes for real-time monitoring and safety automation. The radar detection module provides a visual representation of nearby objects, simulating radar scanning, while the railway gate module enhances public safety by eliminating the need for manual gate operation. This project is scalable, customizable, and can be extended to suit real-world transportation systems, security applications, and educational demonstrations in embedded systems and IoT technologies.

4.2 Tools/Environment Used and Hardware Requirements

- Arduino Uno
- Ultrasonic Sensors (HC-SR04)
- Servo Motors (SG90 or MG90S)
- Jumper Wires and Breadboard
- USB Cable
- Laptop/PC

4.3 Software System Requirements

- Operating System: Windows 10/11
- Arduino IDE

4.4 Specific Requirements

4.4.1 External Interface Requirements

External interface requirements specify hardware, software or database elements with which System or component must interface

4.5 Functional Requirements

The functional requirements of the **Radar Detection and Automatic Railway Gate Control** system using Arduino Uno are designed to ensure intelligent monitoring, automation, and safety in railway and obstacle detection environments. The system is composed of two core modules—**Radar Detection** and **Railway Gate Control**—working in synchronization to achieve efficient real-time performance.

Firstly, the radar module must be capable of detecting nearby objects using an ultrasonic sensor mounted on a servo motor, which rotates across a defined angular range. It should accurately measure distances and angles, transmitting this data through serial communication to a PC. The system should visualize this information using the Processing IDE in the form of a real-time radar sweep interface, indicating the presence and position of objects.

The railway gate module must autonomously monitor train movement using ultrasonic sensors. When a train is detected within a specified threshold, the Arduino must trigger the servo motor to close the railway gate. After the train passes and the path is confirmed clear, the gate should reopen automatically. The operation must be precise, responsive, and continuous to ensure public safety.

The system should also include features for real-time control, automatic decision-making based on sensor inputs, and support for low-latency response. Integration with alarm/buzzer modules for alerting in case of emergency or obstruction detection is encouraged for improved security.

Additionally, the project can be extended to support wireless alerts, cloud-based data logging, and camera integration for visual monitoring. These enhancements would allow authorities to track train passages, collect incident data, and optimize gate operation schedules.

Key Functional Requirements:

- **Object Detection**: The radar system must scan the environment and detect any object within a certain range using ultrasonic sensors.
- **Angle and Distance Calculation**: The system must measure and transmit object distance and angle to a visual interface in real time.
- **Gate Automation**: The railway gate must automatically open/close based on detected distance from the ultrasonic sensor.
- **Servo Motor Control**: The servo motors must rotate precisely according to commands for both radar sweep and gate movement.
- **Real-Time Visualization**: Processing IDE must visually represent radar data as a live scanning interface.
- **Emergency Response**: The system should be capable of triggering alerts when a train or object is dangerously close.
- **Data Logging and Expansion (Optional)**: Capability to log sensor readings and integrate future modules like GSM/GPS for alerts and tracking.

These functional requirements aim to develop a **smart and reliable railway automation system** that enhances safety, reduces human intervention, and integrates modern technology for real-time operation and monitoring.

4.5.1 Software System Attributes

1. **Reliability:** The system must be highly reliable, ensuring accurate object detection in the radar module and consistent operation of the automatic railway gate mechanism, especially in critical real-time scenarios.
2. **Availability:** The radar and railway gate system must function continuously without interruption to monitor trains and objects effectively, ensuring 24/7 operational capability.
3. **Scalability:** The system should support scalability, allowing additional sensors, modules (such as GPS, GSM, or camera systems), or railway lines to be integrated for enhanced performance and wider coverage.
4. **Security:** The system must incorporate basic security measures to avoid external disruptions, particularly if wireless communication (e.g., IoT or GSM) is used in future expansions.
5. **Usability**: The graphical radar interface (via Processing IDE) should be user-friendly, providing a clear and intuitive visual representation of the environment and detected objects, allowing for easy interpretation.
6. **Performance:** The system must perform with minimal latency, ensuring real-time object detection, distance measurement, and timely gate control to avoid accidents.
7. **Maintainability:** The system should be easy to maintain and update, especially in terms of sensor calibration, firmware updates for Arduino, and modifications to the radar interface or control logic.
8. **Interoperability:** The solution should allow potential integration with other safety systems, such as traffic management, remote alerts, or centralized railway control platforms
9. **Portability:** The project should be portable and adaptable to different platforms (e.g., different Arduino boards or microcontrollers), allowing easy deployment across diverse locations.

10. **Fault Tolerance:** The system must be capable of handling minor errors (such as temporary sensor failure or power fluctuations) and continue functioning without causing safety hazards.

11. **Compliance:** The project should aim to follow basic safety and electronic standards, especially in terms of railway automation and public safety systems, ensuring its deployment doesn't interfere with existing infrastructure.

4.6 Non-Functional Requirements

The Radar Detection System and Automatic Railway Gate Control project using Arduino Uno demands careful attention to non-functional requirements to ensure the system meets real-world performance, safety, and usability standards. These requirements are essential for enhancing the system's efficiency, responsiveness, and adaptability in dynamic railway and object monitoring environments.

The performance of the system must emphasize real-time responsiveness, ensuring minimal latency during object detection and immediate action in gate control. The radar system should scan smoothly across defined angles and return accurate measurements with high precision. The railway gate should respond instantly when a train is detected to prevent accidents.

Usability plays a key role; the Processing IDE radar interface should be simple, visually clear, and user-friendly, enabling easy understanding of the environment. The Arduino setup should be straightforward to operate and require minimal training.

Reliability is critical, as the system operates in real-time where delays or failures could pose safety risks. Components must be durable and capable of functioning under various environmental conditions. Redundant checks in logic must ensure smooth operation in case of minor sensor faults or fluctuations.

Security is essential, especially when wireless modules (like GSM or IoT) are integrated in future expansions. The system must protect against unauthorized access and interference that could compromise safety.

Scalability must be ensured so that additional gates, sensors, or modules can be added for covering larger railway areas or integrating more advanced technologies, such as GPS or camera-based surveillance.

Interoperability allows the system to communicate with external monitoring or alert systems, such as railway control centers, mobile apps, or smart traffic systems, enabling broader integration.

Maintainability is necessary for smooth updates to software logic or hardware components, ensuring minimal downtime and easy troubleshooting.

Accessibility considerations should be made for operators or authorities using the interface, ensuring it is clearly presented and adaptable to different screen resolutions and devices.

Compliance with basic safety, hardware interfacing, and embedded system standards ensures that the system is eligible for real-world deployment in railway safety automation.

Collectively, addressing these non-functional requirements guarantees that the radar detection and automatic gate system is robust, responsive, and suitable for real-time field application, improving railway safety and minimizing human dependency.

CHAPTER 5

SYSTEM ANALYSIS AND DESIGN

5.1 Introduction

The Radar Detection and Automatic Railway Gate Control System is designed through a structured and systematic approach, focusing on integrating both hardware and software components to ensure efficient and automated railway crossing management. The system design involves defining key modules such as the ultrasonic sensors for object detection, servo motors for gate control, and microcontrollers for processing and communication. It ensures that each component functions seamlessly within the architecture to meet real-time operational needs. By analyzing the system's requirements, the design process establishes a reliable and responsive framework that minimizes manual intervention, enhances safety, and provides a scalable solution adaptable to various railway environments.

5.2 System Architecture

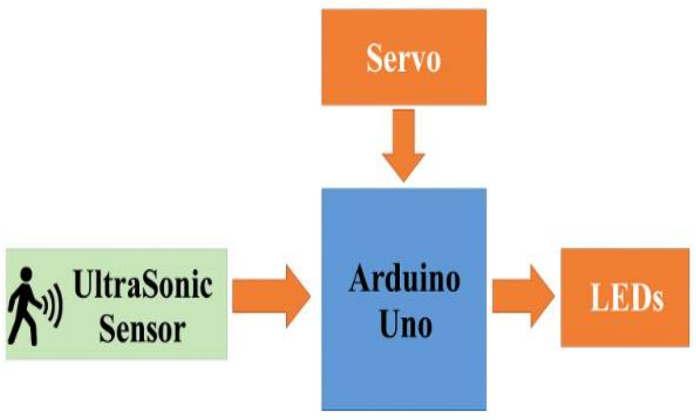


Fig 5.1 System Architecture

5.3 System Analysis

The system analysis for the Radar Detection and Automatic Railway Gate Control project focuses on identifying current issues in manual railway gate operations and providing an automated, sensor-based solution. It studies how IR and radar sensors interact with the Arduino Uno to detect approaching trains and obstacles. The analysis defines the key functions, including automatic gate control and warning signal activation, ensuring accurate, timely responses to improve safety and reduce human error.

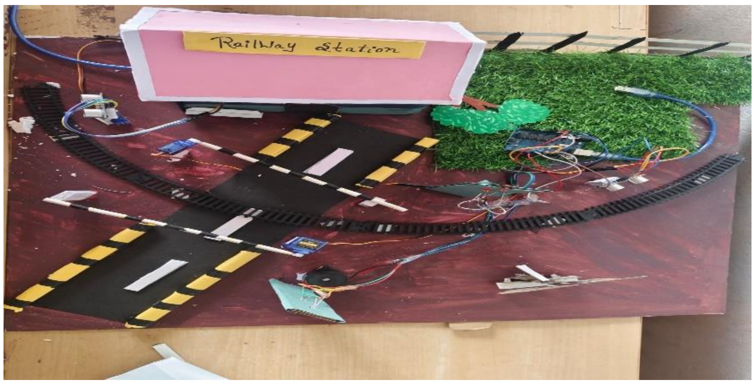
5.4 System Design

The system design for the Radar Detection and Automatic Railway Gate Control project aims to provide a clear and structured framework for implementing an automated safety solution. The design outlines the interaction between ultrasonic or IR sensors, the Arduino Uno microcontroller, and the servo motor for automatic gate control. The goal is to ensure modularity and simplicity, allowing easy modifications and future upgrades. A cohesive system with minimal interdependencies ensures efficient operation, ease of debugging, and maintainability. The design also includes intuitive interfaces that manage sensor input and motor control, offering smooth and reliable system functionality.

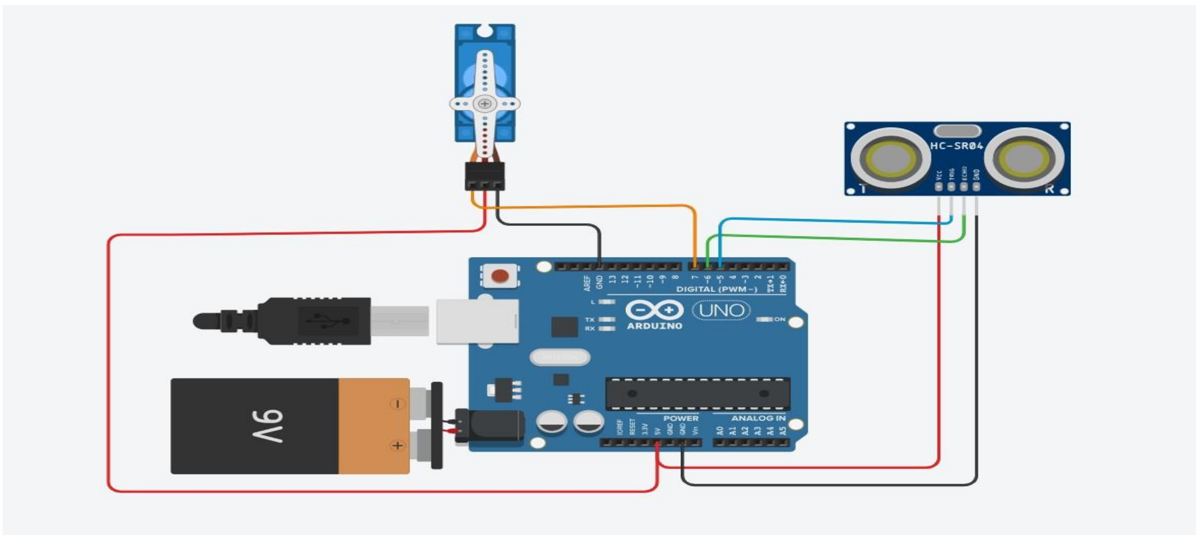
5.5 Detailed Design

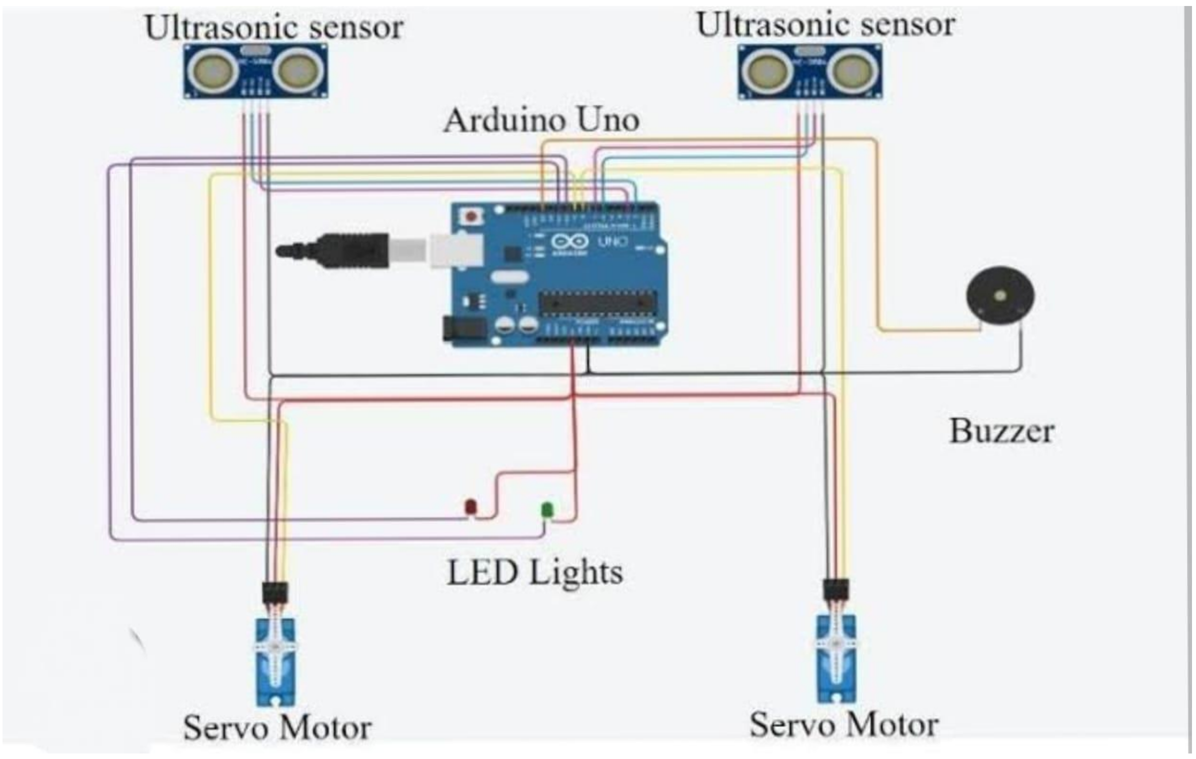
The detailed design of the Radar Detection and Automatic Railway Gate Control system involves specifying the internal structure and behavior of each module within the system. It translates the functional requirements into precise hardware and software configurations. This includes the setup of ultrasonic or IR sensors for detecting approaching trains or obstacles, interfacing them with the Arduino Uno, and programming the servo motor for automatic gate movement. The design also considers real-time data processing, appropriate signal delays, and safety measures. This phase ensures that each component interacts seamlessly, laying a solid foundation for implementation, testing, and future enhancements.

5.6 Structure of Project



5.7 Connectivity Diagram





5.8 Flow Chart

A flow chart is a graphical representation of a process or system that uses various symbols and arrows to illustrate the flow of information or activities. It is commonly used in industries such as manufacturing, engineering, software development, and project management to help visualize complex workflows and identify potential areas of improvement. The basic elements of a flow chart include shapes that represent different types of actions or decisions, and arrows that connect the shapes to show the sequence of events or steps in the process.

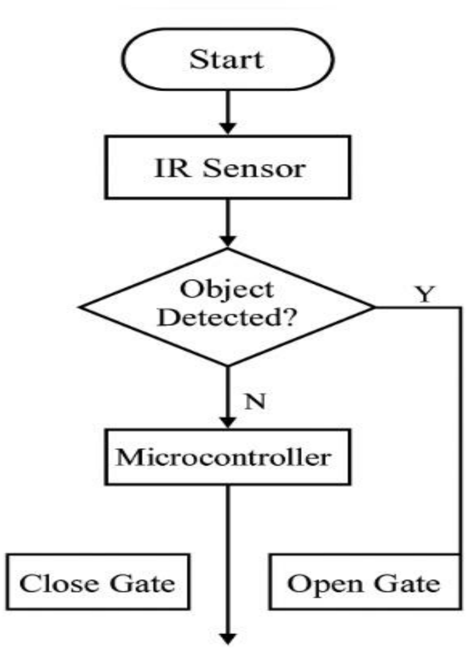


Fig 5.4 Flow Chart

The following describes the process of radar-based vehicle detection using IR sensors and LDR for adaptive lighting control. In this system, the LDR sensor output is connected to analog pin A4 of the Arduino Uno board to measure ambient light levels. The IR sensor used for vehicle detection is connected to analog pin A0, providing input signals to the Arduino. The ground pins of both the LDR and IR sensors are connected to the GND port on the Arduino. The output signal that controls the LEDs is connected to digital pins 6 and 7, which activate or dim the lights based on sensor input. All LED negative terminals are linked to the GND port. The Arduino is powered with a regulated 7–12V input to ensure stable operation of the system.

Chapter 6:

IMPLEMENTATION

6.1 System Architecture

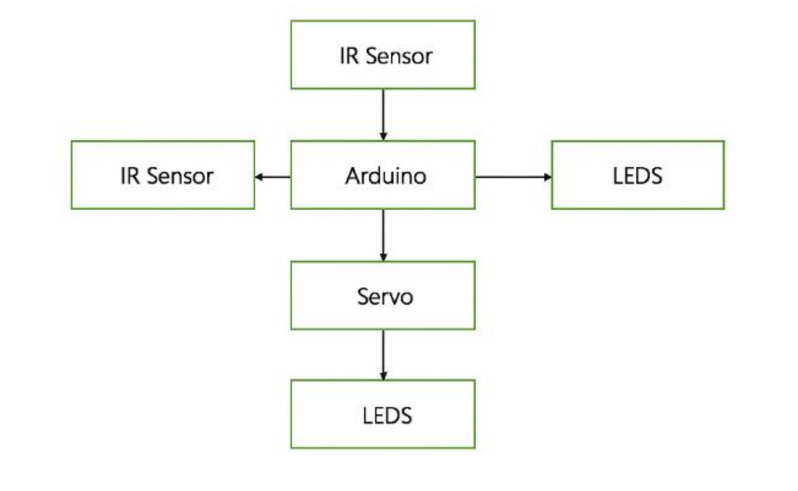


Fig 6.1 System Architecture

To implement a radar-based automatic railway gate system using Arduino Uno, you'll need several hardware components, including ultrasonic sensors (used as radar for train detection), a servo motor (for gate operation), LEDs (for visual warnings), a buzzer (for audio alerts), a power supply, connecting wires, and a breadboard or PCB for assembling the components. The ultrasonic sensors are used to continuously monitor the distance of an approaching train from a fixed point. The trigger and echo pins of the sensors are connected to the digital input/output pins of the Arduino Uno. The servo motor and output indicators (LEDs and buzzer) are connected to digital output pins.

In this system, the ultrasonic sensors detect when a train approaches within a predefined range. When the train is detected, the Arduino code activates the buzzer and LEDs to alert vehicles and pedestrians. At the same time, the servo motor is triggered to automatically close the railway gate. After the train has passed and is no longer detected within the threshold distance, the system deactivates the buzzer and LEDs, and the servo motor opens the gate automatically, restoring safe crossing conditions.

Additionally, in the provided Arduino code, the setup() function is used to initialize all sensor and actuator pins. The loop() function continuously reads distance values from the ultrasonic sensors and performs real-time decisions to either activate or deactivate the alert and gate mechanisms based on train proximity.

For further customization, you can add enhancements like real-time data logging, a GSM module for alerting railway control centers, or solar power for energy efficiency in remote locations. Adjusting the distance threshold and calibrating the sensors ensures accurate and responsive operation of the system.

6.2 Source Code

```
#include <Servo.h>.

// Defines Tirg and Echo pins of the Ultrasonic Sensor

const int trigPin = 10;

const int echoPin = 11;

// Variables for the duration and the distance

long duration;

int distance;

Servo myServo; // Creates a servo object for controlling the servo motor

void setup() {

  pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output

  pinMode(echoPin, INPUT); // Sets the echoPin as an Input

  Serial.begin(9600);

  myServo.attach(12); // Defines on which pin is the servo motor attached

}

void loop() {

  // rotates the servo motor from 15 to 165 degrees

  for(int i=15;i<=165;i++){
```

```
  myServo.write(i);
  delay(30);
  distance = calculateDistance();// Calls a function for calculating the distance measured by the
Ultrasonic sensor for each degree


  Serial.print(i); // Sends the current degree into the Serial Port
  Serial.print(","); // Sends addition character right next to the previous value needed later in the
Processing IDE for indexing
  Serial.print(distance); // Sends the distance value into the Serial Port
  Serial.print("."); // Sends addition character right next to the previous value needed later in the
Processing IDE for indexing
  }
  // Repeats the previous lines from 165 to 15 degrees
  for(int i=165;i>15;i--){
  myServo.write(i);
  delay(30);
  distance = calculateDistance();
  Serial.print(i);
  Serial.print(",");
  Serial.print(distance);
  Serial.print(".");
  }
}
// Function for calculating the distance measured by the Ultrasonic sensor
int calculateDistance(){
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  // Sets the trigPin on HIGH state for 10 micro seconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH); // Reads the echoPin, returns the sound wave travel time
in microseconds
  distance= duration*0.034/2;
  return distance;
}
//automatic railway gate
#include <Servo.h>
Servo servoMain; // Define our Servo
int trigpin = 10;
int echopin = 11;
int distance;
float duration;
float cm;
void setup()
{
   servoMain.attach(9); // servo on digital pin 10
    pinMode(trigpin, OUTPUT);
    pinMode(echopin, INPUT);
}
void loop()
{
  digitalWrite(trigpin, LOW);
  delay(1);
  digitalWrite(trigpin, HIGH);
  delayMicroseconds(50);
  digitalWrite(trigpin, LOW);
  duration = pulseIn(echopin, HIGH);
  cm = (duration/68.82);
  distance = cm;
  if(distance<10)
  {
    servoMain.write(90);  // Turn Servo back to center position (90 degrees)
   delay(4000);
  }
   else{
     servoMain.write(0);
     delay(20);
   }
}
```

6.3 Functions and Explanation

In the provided Arduino sketch for the Radar-Based Detection and Automatic Railway Gate Control System, there are no import statements for external libraries. The code includes:

#include <Arduino.h>

This preprocessor directive loads the core Arduino library, which provides essential functions for working with Arduino boards. The sketch does not use any additional libraries such as for LCD or GSM modules, thereby maintaining a simple, cost-effective design.

The Arduino sketch primarily consists of two core functions: setup() and loop().

- The setup() function executes only once when the board is powered on or reset. It configures all I/O pins connected to the ultrasonic sensors (used for radar detection), the buzzer, LEDs, and the servo motor controlling the railway gate using the pinMode() function.
- The loop() function runs continuously after the setup. It constantly reads the distance data from ultrasonic sensors using trigger and echo pins. Based on these readings, the code determines if a train is approaching. If a train is detected within the predefined threshold distance, the Arduino activates the alert system (LEDs and buzzer) and closes the railway gate using the servo motor. Once the train passes, the system reopens the gate and deactivates alerts.

1.Variables Initialization: The code starts by initializing various variables to represent the trigger and echo pins of ultrasonic sensors, pins for the LED, buzzer, and the servo control, as well as float or long variables to store the measured distance and duration of echo pulses.
2.Setup Function: Inside setup(), the pinMode() function is used to:

Set the **ultrasonic sensor trigger pins** as OUTPUT

- Set the **echo pins** as INPUT
- Set **LED and buzzer pins** as OUTPUT
- Initialize the **servo motor** and move it to the default open position (e.g., angle 0 degrees)

3. **Loop Function:** The loop() function constantly checks sensor data:

- Sends a trigger pulse to the ultrasonic sensor

- Receives the echo pulse and calculates distance using the formula:

distance = (duration * 0.034) / 2;

- If the distance is **below the threshold**, the system:
    - Turns on the LED and buzzer to warn of the approaching train
    - Closes the gate using servo.write(90)
- Once the train has passed and distance increases:
    - Deactivates the LED and buzzer
    - Reopens the gate using servo.write(0)

4.**Sensor Reading:** The ultrasonic sensor readings are taken using digitalWrite() to send a trigger and pulseIn() to measure the return time.

5.**Conditional Statements (if-else):** The sketch uses conditional logic to decide the state of the gate and alert system:

- if conditions check for a train within range
- else conditions handle the case when the track is clear

6. **Servo Motor Control:** Based on the detection:

- servo.write(90) closes the gate
- servo.write(0) opens the gate

7. **LED and Buzzer Control:** The LED and buzzer are turned on using digitalWrite(HIGH) when a train is detected, and turned off using digitalWrite(LOW) when no train is present.

CHAPTER 7

TESTING

7.1 Introduction

Testing in this project is defined as the process of evaluating the hardware and software components of the Radar-Based Detection and Automatic Railway Gate Control System to identify any deviations between expected and actual outcomes, and to assess the overall functionality and reliability of the system.

In this context, testing involves running the Arduino-based setup—including ultrasonic sensors, LED indicators, buzzer, and the servo motor for gate control—under various simulated real-world conditions. These include both normal scenarios, such as the absence of a train, and abnormal or critical scenarios, like a train being detected at close range.

The objective is to ensure that the system responds correctly in each situation: detecting approaching trains using radar (ultrasonic) sensors, activating alerts via LEDs and buzzer, and controlling the railway gate automatically. The testing process is crucial for validating that the logic implemented in the Arduino sketch is accurate and that the sensors and actuators behave as expected.

This testing phase is deliberately designed to trigger edge cases and error conditions to evaluate how the system handles unexpected or challenging inputs. By doing so, the system's reliability, responsiveness, and safety performance are verified, making it suitable for deployment in real-world railway crossings and hilly areas where visibility is low.

7.1 Unit Testing

In this Radar-Based Detection and Automatic Railway Gate Control System using Arduino Uno, unit testing is carried out to verify the correct functioning of individual components such as ultrasonic sensors, LEDs, buzzers, and the servo motor. Each component is tested separately to ensure it performs its intended function—for example, the ultrasonic sensor must accurately detect the train's distance, the buzzer should activate when a train is detected, and the servo motor must correctly open or close the gate. The Arduino code's setup and loop functions are also tested to ensure proper initialization and continuous monitoring. This type of testing helps identify faults early and ensures each unit of the system works reliably before full system integration

7.1.2 Integration Testing

In the Radar-Based Detection and Automatic Railway Gate Control System, integration testing ensures that all hardware components and software logic work together seamlessly. This includes verifying the interaction between ultrasonic sensors, Arduino Uno, servo motor, LEDs, and buzzer. The system is tested as a whole to confirm that when the radar detects an approaching train, the Arduino processes the signal correctly, triggers the buzzer and LEDs for alert, and activates the servo motor to close or open the gate. Both normal and failure scenarios are tested to validate the system's response, ensuring reliable operation across different conditions.

7.1.3 System Testing

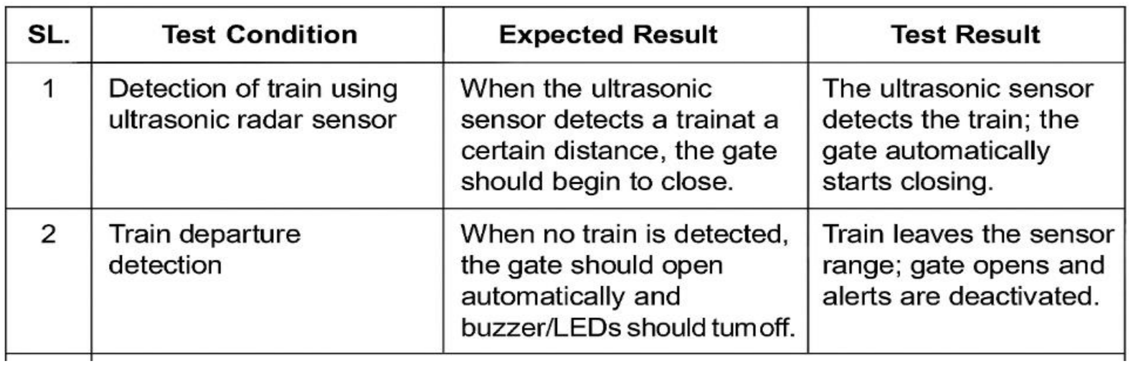
A system testing of software or hardware is testing conducted on a complete, integrated system to evaluate system's compliance with its specified requirements. System testing falls within the scope of black box testing, and such as, should require no knowledge of the inner design of the integrated software components.

7.1.4 Validation Testing

Validation Testing ensures that the product actually meets the client's needs. It can also be defined as demonstrating that the product fulfils its intended use when deployed on appropriate environment

7.2 Test Cases

7.2.1 Sensor Testing



SL.	Test Condition	Expected Result	Test Result
1	Detection of train using ultrasonic radar sensor	When the ultrasonic sensor detects a trainat a certain distance, the gate should begin to close.	The ultrasonic sensor detects the train; the gate automatically starts closing.
2	Train departure detection	When no train is detected, the gate should open automatically and buzzer/LEDs should tumoff.	Train leaves the sensor range; gate opens and alerts are deactivated.

Table 7.1 : Sensor Testing

CHAPTER 8

CONCLUSION AND FUTURE ENHANCEMENT

8.1 Conclusion

In conclusion, the Radar-Based Automatic Railway Gate System marks a significant advancement in enhancing railway crossing safety through automation. By integrating ultrasonic radar sensors and microcontroller-based controls, this system effectively detects approaching trains and automates gate operations without human intervention. It reduces the chances of human error, prevents accidents, and ensures the smooth flow of both rail and road traffic. The use of buzzers and LEDs further enhances alert mechanisms, improving overall safety. This project demonstrates the potential of embedded systems and sensor integration in solving real-world safety challenges, and it lays the foundation for future enhancements such as real-time monitoring and wireless communication for centralized railway management.

8.2 Future Enhancement

The Radar-Based Automatic Railway Gate System presents a strong foundation for improving safety at railway crossings, and there is ample scope for future enhancement. Integration of advanced technologies such as IoT (Internet of Things), real-time data analytics, and cloud-based monitoring can significantly upgrade the system's efficiency and capabilities. In future versions, GPS modules can be added to track train locations more precisely, and wireless communication can allow centralized control of multiple crossings. AI-based decision-making can be incorporated to handle complex scenarios such as multiple train detection or emergency vehicle prioritization. Additionally, solar-powered operation could enhance sustainability, while camera modules could enable video surveillance and incident recording. These enhancements will not only make the system more intelligent and reliable but also contribute to safer and smarter transportation infrastructure.

BIBLIOGRAPHY

Text Book Referred:

[1] S. Sharma, "Radar-Based Railway Gate Automation System," *International Journal of Advanced Research in Electronics and Communication Engineering*, Vol. 4, Issue 3, March 2015.

[2] Dr. Ramesh Gaonkar, *Microcontroller Programming: Principles and Applications*, Pearson Education, 2012.

[3] R. S. Kaler, "Intelligent Railway Gate Control using Sensor Networks," *International Journal of Computer Applications*, Vol. 56, No. 12, October 2012.

[4] M. M. Raj, "Automatic Railway Gate Control Using Arduino," *International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering*, 2016.

[1] https://www.hackster.io/abhilashpatel121/automatic-railway-gate-control-using-arduino-e28ae3

[2] https://circuitdigest.com/microcontroller-projects/arduino-based-railway-gate-controller-system

[3] https://www.instructables.com/Automatic-Railway-Gate-Control-Using-Ultrasonic-Se/

[4] https://www.youtube.com/watch?v=bmYPwvK44r4