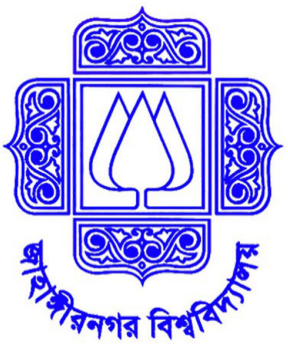

Project Report

Smart Expressway Management System

Course Title: IoT Laboratory

Course Code: CSE-460

4th Year 2nd Semester Examination 2023

Submitted by:

Fatima Binte Aziz(202179)

Md Soad Anam(202197)

Md Hasan Al Mamun(202211)

Submitted to:

Dr. Md. Ezharul Islam

Professor

Samsun Nahar Khandakar

Lecturer

Department of Computer Science and Engineering

Jahangirnagar University

Submitted on: May 22, 2025

Abstract

This report presents the design and implementation of the Smart Expressway Management System, an IoT-based solution for monitoring vehicle speed, issuing overspeed alerts, and identifying vehicles using RFID technology. The system employs an ESP32 microcontroller, IR sensors, an MFRC522 RFID reader, and a buzzer, integrated with the Blynk IoT platform for remote monitoring and control. The project calculates vehicle speed using two IR sensors, activates a buzzer for speeds exceeding 60 cm/s, and displays vehicle information via RFID tags. The Blynk app provides real-time speed display, buzzer control, and vehicle data visualization. This report details the hardware, software, methodology, testing, challenges, and future scope of the project, demonstrating its potential to enhance road safety and traffic management.

Contents

1 Introduction

1.1 Background

Modern expressways face challenges such as speeding, inefficient vehicle tracking, and lack of real-time monitoring. With the rise of IoT technologies, smart systems can address these issues by providing automated monitoring and control. The Smart Expressway Management System leverages IoT to enhance road safety and traffic efficiency.

1.2 Problem Statement

Speeding is a leading cause of accidents on expressways, and manual monitoring is labor-intensive and error-prone. Additionally, identifying vehicles for toll collection or security purposes requires efficient systems. Existing solutions often lack integration with user-friendly interfaces for remote management.

1.3 Objectives

The primary objectives of this project are:

- To develop a system for real-time vehicle speed monitoring using IR sensors.
- To implement an overspeed alert mechanism using a buzzer.
- To enable vehicle identification via RFID tags.
- To provide remote monitoring and control through the Blynk IoT platform.

1.4 Scope

This project focuses on a prototype for a single-lane expressway, using dummy vehicle data for RFID. It can be scaled to real-world applications with additional sensors and database integration.

2 System Overview

The Smart Expressway Management System is an IoT-based prototype that monitors vehicle speed, issues alerts for overspeeding, and identifies vehicles using RFID. The system comprises an ESP32 microcontroller, two IR sensors, an MFRC522 RFID reader, a buzzer, and an LED, integrated with the Blynk app for remote access.

2.1 Key Features

- **Speed Monitoring**: Calculates vehicle speed using IR sensors.
- **Overspeed Alerts**: Activates buzzer for speeds more thant 80 km/h.
- **Vehicle Identification**: Reads RFID tags and displays dummy vehicle data.
- **Remote Control**: Blynk app for speed display and buzzer control.

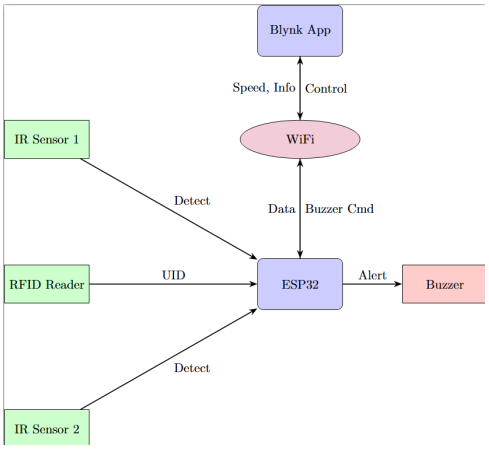

Figure 2.1: Block Diagram

3 Hardware Description

The system uses the following components, detailed in Table 3.1.

Component	Description
ESP32	Microcontroller with WiFi for processing and Blynk connectivity.
IR Sensors (2)	Detect vehicle movement (pins 33, 32, INPUT_PULLUP).
MFRC522 RFID Reader	Reads vehicle tags (pins: SS=5, RST=21, SPI: 14,12,13).
Buzzer	Alerts for overspeeding (pin 25).
LED	Indicates RFID detection (pin 2).

Table 3.1: Hardware components of the system.

3.1 ESP32 Microcontroller

The ESP32 is a powerful microcontroller with built-in WiFi, ideal for IoT applications. It processes sensor data, communicates with Blynk, and controls outputs.

3.2 IR Sensors

Two IR sensors, spaced 10 cm apart, detect vehicle movement by triggering when an object interrupts their beam. They use INPUT_PULLUP to ensure stable readings.

3.3 RFID Reader

The MFRC522 module reads RFID tags to simulate vehicle identification, outputting a unique UID for each tag.

3.4 Buzzer and LED

The buzzer alerts for speeds exceeding 60 cm/s or manual activation via Blynk. The LED indicates successful RFID tag detection.

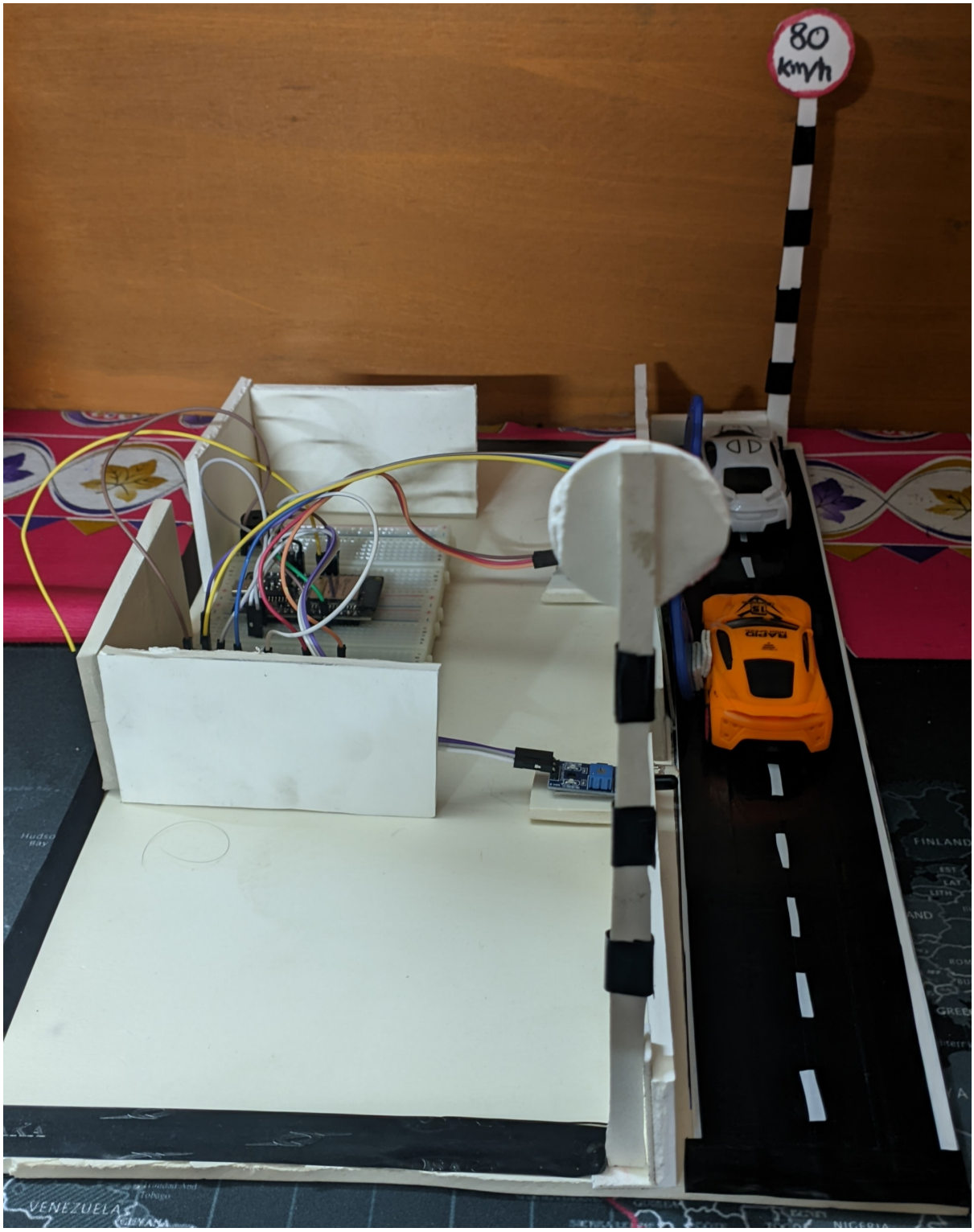

Figure 3.1: Assembled Project Prototype

4 Software Description

4.1 Arduino IDE

The Arduino IDE is used to program the ESP32, providing a user-friendly environment for coding and uploading firmware.

4.2 Libraries

The following libraries are used:

- `WiFi`: For ESP32 WiFi connectivity.
- `BlynkSimpleEsp32`: For communication with the Blynk platform.
- `SPI`: For RFID communication.
- `MFRC522`: For controlling the RFID reader.

4.3 Blynk IoT Platform

Blynk enables remote monitoring and control via a mobile app. The system uses:

- **V0 (Double)**: Displays speed in cm/s.
- **V1 (Integer)**: Controls buzzer (0=OFF, 1=ON).
- **V2 (String)**: Displays dummy vehicle info.

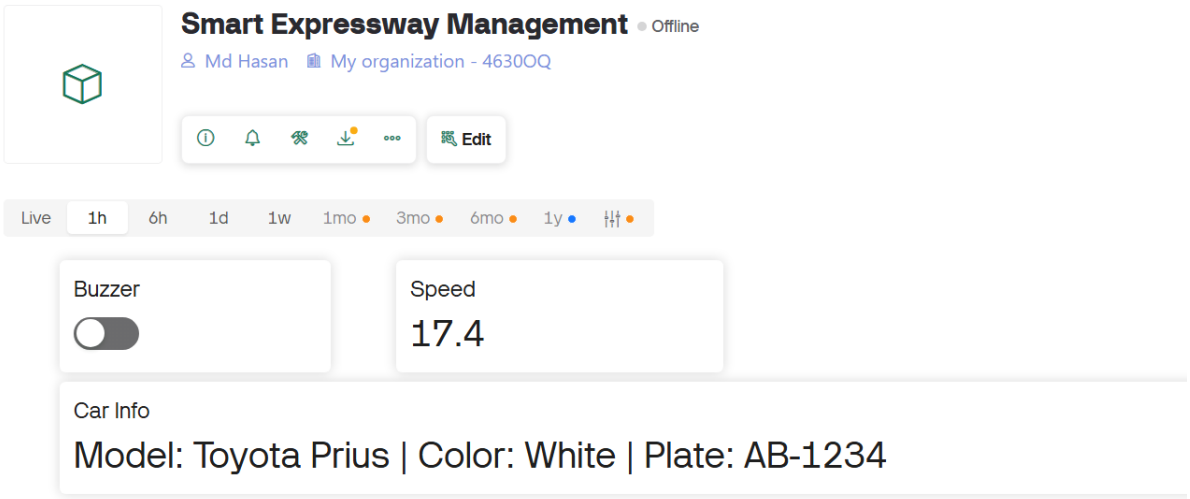

Figure 4.1: Blynk app dashboard showing speed, buzzer control, and car info.

5 Methodology

5.1 Speed Calculation

Two IR sensors measure the time a vehicle takes to travel 10 cm:

$$\text{Speed} = \frac{\text{Distance}}{\text{Time Difference}} = \frac{10}{\Delta t} \text{ cm/s} \tag{5.1}$$

where Δt is the time between sensor triggers in seconds.

5.2 Buzzer Control

The buzzer activates automatically if speed is greater than 80 km/ or manually via Blynk (V1). The logic ensures overspeed alerts take precedence.

5.3 RFID Vehicle Identification

The MFRC522 reader detects RFID tags and sends dummy vehicle data (e.g., "Model: Toyota Prius") to Blynk (V2), alternating between two entries.

5.4 Blynk Integration

The ESP32 connects to Blynk via WiFi (SSID: A35, Password: subarna1234), sending speed to V0 and vehicle data to V2, and receiving buzzer commands from V1.

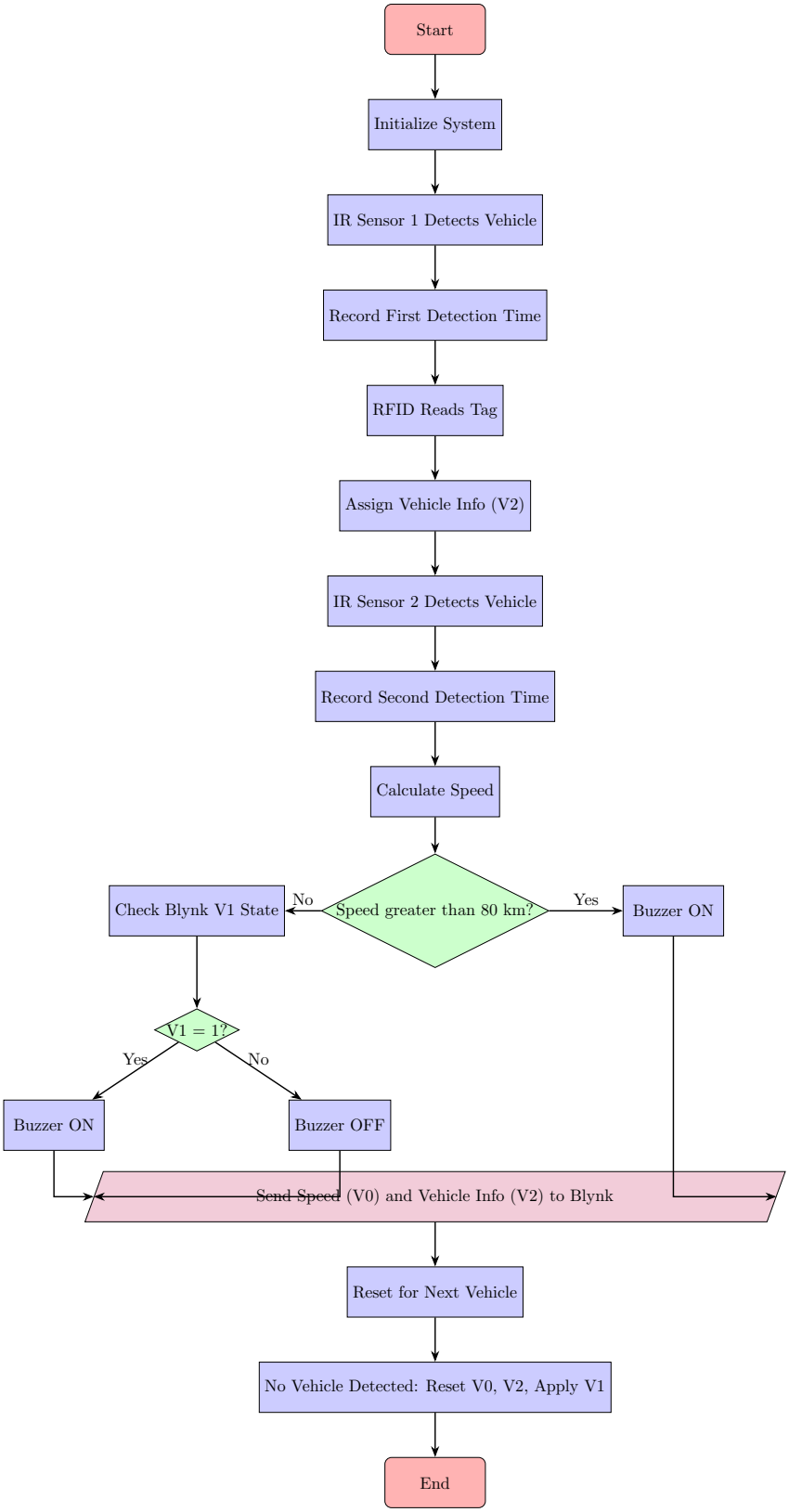

Figure 5.1: System Flowchart

6 Implementation

6.1 Code Structure

The code is written in C++ for the Arduino IDE. Key sections include:

- **Setup**: Initializes serial communication, SPI, pins, and Blynk.
- **Loop**: Handles IR sensor readings, speed calculation, RFID detection, and Blynk updates.
- **Blynk Handler**: Processes V1 button inputs.

6.2 Code Snippet: Speed Calculation

```
float timeDifference = (secondDetectionTime - firstDetectionTime)
    / 1000.0;
if (timeDifference > 0.05 && timeDifference < 10.0) {
  speed = distanceBetweenSensors / timeDifference;
  Blynk.virtualWrite(V0, speed);
}
```

6.3 Code Snippet: RFID and Car Info

```
if (mfrc522.PICC_IsNewCardPresent() && mfrc522.
   PICC_ReadCardSerial()) {
  if (toggleCar) {
    Blynk.virtualWrite(V2, carInfo1);
  } else {
    Blynk.virtualWrite(V2, carInfo2);
  }
  toggleCar = !toggleCar;
}
```

6.4 Setup Process

1. Connect hardware (ESP32, IR sensors, RFID, buzzer, LED).
2. Configure Blynk datastreams (V0: Double, V1: Integer, V2: String).
3. Upload code with correct WiFi credentials and Blynk auth token.
4. Test with objects for IR sensors and RFID tags.

7 Testing and Results

7.1 Test Procedure

- **Speed**: Passed objects between IR sensors to simulate vehicles.
- **Buzzer**: Tested automatic activation (greater than 80 km/h) and manual control via Blynk.
- **RFID**: Scanned tags to verify UID detection and car info display.
- **Blynk**: Confirmed real-time updates on V0 and V2, and V1 control.

7.2 Results

- **Speed**: Recorded speeds of 50–100 cm/s, displayed on Blynk V0.
- **Buzzer**: Activated correctly for overspeeding and manual toggling.
- **RFID**: Alternated between "Toyota Prius" and "Honda Civic" on V2.
- **Serial Output**: Confirmed detections (e.g., "Speed: 66.67 cm/s", "UID: 4A 2B 1C 3D").

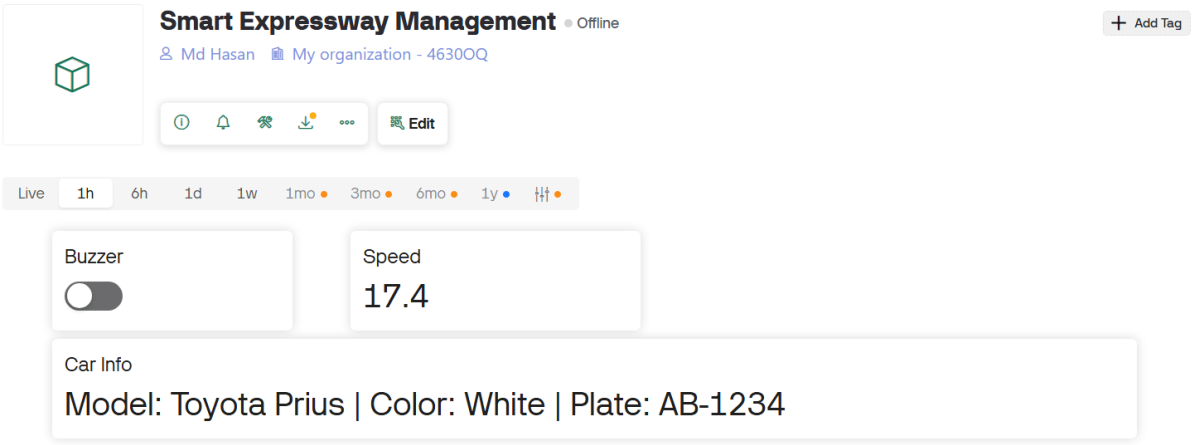

Figure 7.1: Sample test setup with IR sensors and RFID reader.

8 Challenges and Solutions

- **Challenge**: IR sensor timing sensitivity led to inaccurate speeds.
  - **Solution**: Implemented time validation (0.05–10s) to filter invalid triggers.
- **Challenge**: Configuring Blynk datastreams for float and string data.
  - **Solution**: Used Double for V0 and String for V2, ensuring correct widget mapping.
- **Challenge**: WiFi connectivity issues with ESP32.
  - **Solution**: Verified credentials and added connection status debugging.

9 Future Scope

The system can be enhanced by:

- Mapping RFID UIDs to a real vehicle database.
- Adding cloud storage for speed and vehicle logs.
- Integrating a camera for license plate recognition.
- Supporting multiple lanes with additional sensors.
- Implementing real-time alerts via SMS or email.

10 Conclusion

The Smart Expressway Management System successfully demonstrates real-time vehicle speed monitoring, overspeed alerts, and vehicle identification using IoT. The integration of ESP32, IR sensors, RFID, and Blynk provides a user-friendly solution for traffic management. The prototype achieves accurate speed detection, reliable buzzer control, and vehicle data display, with potential for real-world deployment in smart highways.

Bibliography

[1] Blynk IoT Platform Documentation, `https://docs.blynk.io`.

[2] Arduino IDE Documentation, `https://www.arduino.cc`.

[3] MFRC522 Library, `https://github.com/miguelbalboa/rfid`.

A Complete Code

```
#define BLYNK_TEMPLATE_ID "TMPL6c5ml3ZsE"
#define BLYNK_TEMPLATE_NAME "Smart Expressway Management"
#define BLYNK_AUTH_TOKEN "OcNUrpKTsJ3SOiFsWoKm1W4q78GeNaoZ"

#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>
#include <SPI.h>
#include <MFRC522.h>

#define SS_PIN    5
#define RST_PIN   21
#define LED_PIN   2
MFRC522 mfrc522(SS_PIN, RST_PIN);

char ssid[] = "A35";
char pass[] = "subarna1234";

#define irSensor1 33
#define irSensor2 32
#define buzzerPin 25

float distanceBetweenSensors = 10.0;
unsigned long firstDetectionTime = 0;
unsigned long secondDetectionTime = 0;
bool firstDetected = false;
bool secondDetected = false;
float speed = 0;
bool buzzerManual = false;

String carInfo1 = "Model: Toyota Prius    |\nColor: White    |\
   nPlate: AB-1234";
```

```
String carInfo2 = "Model: Honda Civic    |\nColor: Black    |\
   nPlate: XY-5678";
bool toggleCar = false;

void setup() {
  Serial.begin(115200);
  SPI.begin(14, 12, 13, SS_PIN);
  pinMode(LED_PIN, OUTPUT);
  digitalWrite(RST_PIN, LOW);
  delay(100);
  digitalWrite(RST_PIN, HIGH);
  delay(100);
  mfrc522.PCD_Init();
  mfrc522.PCD_SetRegisterBitMask(mfrc522.RFCfgReg, (0x07 << 4));
  Serial.println("RFID Ready...");
  pinMode(irSensor1, INPUT_PULLUP);
  pinMode(irSensor2, INPUT_PULLUP);
  pinMode(buzzerPin, OUTPUT);
  Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass);
}

BLYNK_WRITE(V1) {
  int buttonState = param.asInt();
  buzzerManual = buttonState;
}

void loop() {
  Blynk.run();
  if (!firstDetected && digitalRead(irSensor1) == LOW) {
    firstDetectionTime = millis();
    firstDetected = true;
    Serial.println("First IR detected");
    delay(50);
  }
  if (firstDetected && !secondDetected && digitalRead(irSensor2)
     == LOW) {
    secondDetectionTime = millis();
    secondDetected = true;
    Serial.println("Second IR detected");
    float timeDifference = (secondDetectionTime -
       firstDetectionTime) / 1000.0;
```

```
70      if (timeDifference > 0.05 && timeDifference < 10.0) {
71        speed = distanceBetweenSensors / timeDifference;
72        Serial.print("Speed: ");
73        Serial.print(speed);
74        Serial.println(" cm/s");
75        Blynk.virtualWrite(V0, speed);
76        if (speed > 60) {
77          digitalWrite(buzzerPin, HIGH);
78        } else {
79          digitalWrite(buzzerPin, buzzerManual ? HIGH : LOW);
80        }
81      } else {
82        Serial.println("Invalid timing detected.");
83      }
84      firstDetected = false;
85      secondDetected = false;
86      firstDetectionTime = 0;
87      secondDetectionTime = 0;
88    }
89    if (mfrc522.PICC_IsNewCardPresent() && mfrc522.
          PICC_ReadCardSerial()) {
90      digitalWrite(LED_PIN, HIGH);
91      Serial.print("UID:");
92      for (byte i = 0; i < mfrc522.uid.size; i++) {
93        Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");
94        Serial.print(mfrc522.uid.uidByte[i], HEX);
95      }
96      Serial.println();
97      if (toggleCar) {
98        Blynk.virtualWrite(V2, carInfo1);
99      } else {
100       Blynk.virtualWrite(V2, carInfo2);
101     }
102     toggleCar = !toggleCar;
103     digitalWrite(LED_PIN, LOW);
104     delay(200);
105     mfrc522.PICC_HaltA();
106     mfrc522.PCD_StopCrypto1();
107   }
108   if (!firstDetected && !secondDetected && speed <= 60) {
109     digitalWrite(buzzerPin, buzzerManual ? HIGH : LOW);
```

```
110     }
111 }
```