IoT BASED SMART TRAFFIC CONTROL AND EMERGENCY MANAGEMENT SYSTEM

A PROJECT REPORT

Submitted by

THARUNIKA K

ANUSRI S

In partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

ELECTRONICS AND COMMUNICATION ENGINEERING

ANNA UNIVERSITY REGIONAL CAMPUS, COIMBATORE

ANNA UNIVERSITY: CHENNAI – 600 025

ABSTRACT

Urban traffic congestion delays ambulances, risking patient outcomes. This paper examines a smart traffic light system with ambulance detection technology to mitigate these issues. Utilizing advanced sensors, IoT, the system detects ambulances and adjusts signals for a clear path.

Key components include transmitter and receiver technology, V2I communication, and adaptive signal control, facilitating smooth ambulance passage through intersections.

Simulations and trials show improved emergency response times and traffic flow. The system's adaptive nature enhances safety and mobility, offering a scalable solution to improve emergency response and reduce urban congestion.

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION:

Urban traffic congestion poses significant challenges to emergency response services, delaying ambulances and affecting patient care. As cities grow, innovative solutions are needed to ensure timely responses.

Smart traffic light systems with ambulance detection offer a promising solution. By utilizing advanced sensors and IoT, these systems manage traffic dynamically. They detect ambulances and adjust signals in real-time, creating clear paths to reduce delays.

This paper explores the design and implementation of such systems, focusing on key components like transmitter and receiver technology, vehicle-to-infrastructure (V2I) communication, and adaptive signal control. Simulation and real-world trials show these systems enhance response times and traffic flow, providing a scalable solution for modern cities. This paper addresses technical challenges and benefits, contributing to more efficient emergency services.

CHAPTER 2

LITERATURE SURVEY

2.1 INTRODUCTION

The increasing traffic congestion in urban areas necessitates innovative solutions for managing traffic flow and prioritizing emergency vehicles such as ambulances. Smart traffic control systems that integrate advanced sensor technologies and communication protocols are being actively researched to address these challenges. This literature survey reviews key studies and developments in the field of smart traffic systems, focusing on the use of ultrasonic sensors, transmitters, receivers, ESP8266, and Arduino Uno.

2.2 USING ULTRASONIC SENSOR

The application of ultrasonic sensors in traffic management systems is well-documented. Ultrasonic sensors are used to measure the density of vehicles at intersections by emitting sound waves and detecting the return signal. This method allows for accurate real-time monitoring of traffic conditions. Gandhi and Trivedi (2007) demonstrated the effectiveness of ultrasonic sensors in estimating traffic density and improving signal timing to optimize flow.

2.3 USING TRANSMITTERS AND RECEIVERS

The integration of transmitters and receivers for emergency vehicle prioritization is a significant advancement in traffic management systems. Transmitters installed in ambulances can communicate with receivers at intersections to trigger signal changes. Kumar and Verma (2020) developed a system where ambulance drivers activate a transmitter, which sends signals to matched receivers at traffic lights, ensuring a clear path for emergency vehicles.

2.4 ESP12-E

The ESP8266 microcontroller, with its integrated Wi-Fi and Bluetooth capabilities, is ideal for IoT applications in smart traffic systems. It enables seamless communication between transmitters, receivers, and central traffic management systems. Al-Sultan et al. (2014) discussed how IoT devices like ESP8266 can facilitate real-time data exchange and control, enhancing the responsiveness and efficiency of traffic systems.

2.5. ARDUINO UNO FOR POWER MANAGEMENT

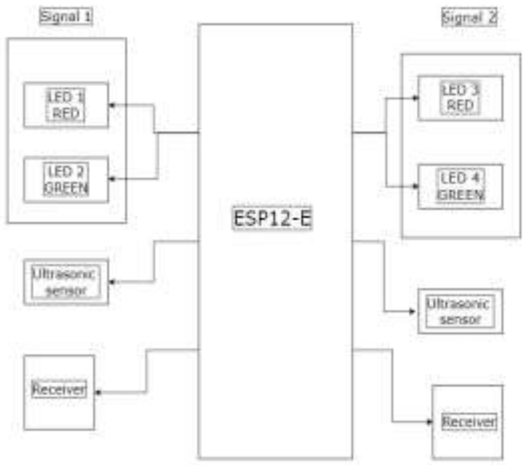
The Arduino Uno microcontroller is often used in traffic systems for managing power supply and interfacing with sensors and other components. Its simplicity and compatibility with various sensors make it a popular choice for prototyping and deploying smart traffic solutions. Research by Zhao et al. (2019) highlights the use of Arduino platforms in controlling traffic signal hardware and integrating with IoT systems.

2.6 SUMMARY

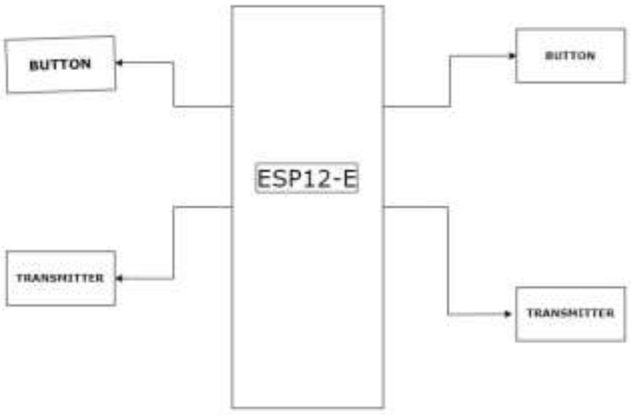
In conclusion, the literature suggests that smart traffic control systems employing ultrasonic sensors, ESP8266, Arduino Uno, and transmitter-receiver technology can significantly improve traffic management and emergency response times. Continued research and development are essential to address current challenges and maximize the potential of these technologies.

CHAPTER 3

3.1 BLOCK DIAGRAM



TRAFFIC CONTROL SYSTEM WITH RECEIVER



TRANSMITTER IN AMBULANCE

CHAPTER 4

PROPOSED SYSTEM

4.1 PROPOSED SYSTEM

The proposed smart traffic control system is designed to alleviate traffic congestion and enhance emergency response efficiency in urban environments. By integrating advanced technologies such as ultrasonic sensors, transmitters, receivers, ESP8266, and Arduino Uno, the system offers a comprehensive solution for managing traffic flow and prioritizing emergency vehicles like ambulances. The key components and functionalities of the proposed system are as follows:

1.TRAFFIC DENSITY MONITORING

Ultrasonic sensors are deployed at intersections to continuously monitor the density of vehicles. These sensors emit high-frequency sound waves and measure the time it takes for the echoes to return, allowing for accurate detection of the number and proximity of vehicles. This real-time data is used to assess traffic density in each direction, enabling dynamic adjustment of traffic signal timings.

2.ADAPTIVE SIGNAL CONTROL

The system utilizes an adaptive signal control mechanism to optimize traffic flow based on real-time density data. When traffic density reaches a critical level in all directions, the system implements a time-sharing strategy, changing traffic signals every 10 seconds. This approach ensures that no single direction experiences prolonged congestion and helps maintain an equitable flow of traffic through intersections.

3.EMERGENCY VEHICLE DETECTION AND PRIORITIZATION

For ambulance prioritization, the system employs a transmitter-receiver setup. Each ambulance is equipped with a transmitter controlled by the driver. In emergency situations, the driver activates the transmitter, which is programmed with unique codes corresponding to receivers installed at traffic intersections.

Transmitter and Receiver Functionality: When the transmitter is activated, it sends a signal to the receivers at upcoming intersections. The matched receiver identifies the signal and immediately triggers the traffic light to change, providing the ambulance with a clear path through the intersection. This reduces delays and ensures a swift passage for emergency vehicles.

4.COMMUNICATION AND CONTROL INFRASTRUCTURE

The system employs ESP8266 microcontrollers for communication and control tasks. One ESP8266 is used in the transmitter module within the ambulance, while another is integrated into the receiver at intersections. The ESP8266 facilitates real-time communication between the ambulance and traffic signals, ensuring timely responses to emergency signals.

5.POWER MANAGEMENT AND CONTROL

Arduino Uno microcontrollers are utilized for managing power supply to the system components and interfacing with sensors and other modules. The Arduino Uno acts as a central hub for processing sensor data, executing control logic, and ensuring the reliable operation of the traffic signal system.

6.SYSTEM IMPLEMENTATION AND EVALUATION

The proposed system will be implemented in a simulated environment to evaluate its effectiveness in improving traffic flow and reducing emergency response times. The simulation will consider various traffic scenarios, including peak congestion periods and emergency situations, to assess the system's performance under different conditions. Real-world trials will also be conducted to validate the system's functionality and benefits.

7.SCALABILITY AND INTEGRATION

The proposed system is designed to be scalable and adaptable to different urban environments. Its modular architecture allows for easy integration with existing traffic infrastructure and expansion to accommodate additional intersections or advanced functionalities as needed. Future developments may include the incorporation of artificial intelligence algorithms to further enhance traffic prediction and management capabilities.

4.2 HARDWARE REQUIREMENTS

ARDUINO UNO

ESP8266

LEDS

RESISTOR

ULTRASONIC SENSOR

SWITCH

TRANSMITTER AND RECEIVER

JUMPER WIRES

4.2.1 ULTRASONIC SENSORS:

Ultrasonic sensors are deployed at strategic locations at intersections to continuously monitor vehicle density. These sensors operate by emitting high-frequency sound waves that reflect off vehicles and return to the sensor. By measuring the time it takes for the echo to return, the system calculates the distance to vehicles, allowing for accurate detection of traffic density in each lane. This real-time data is crucial for assessing congestion levels and optimizing traffic signal timings.

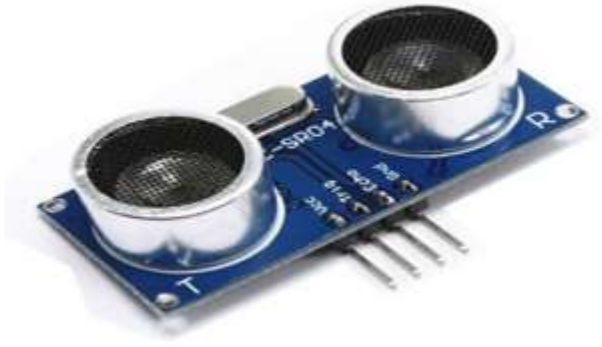


FIG 1 ULTRASONIC SENSOR

4.2.2 ARDUINO UNO:

The Arduino Uno manages the power supply for connected components, ensuring stable operation of the sensors, and traffic lights. Its energy-efficient design makes it suitable for continuous operation in urban environments.

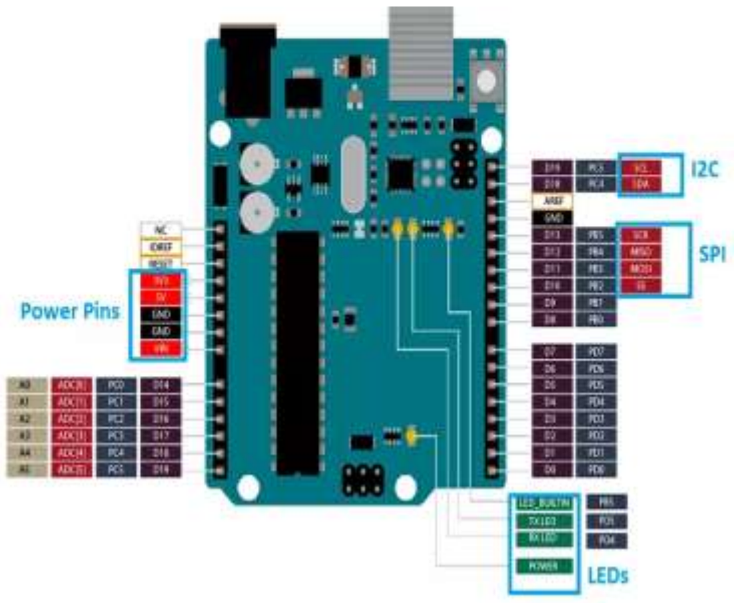


FIG 2 PIN CONFIGURATION OF ARDUINO UNO

4.2.3 TRANSMITTER AND RECEIVER:

TRANSMITTER FUNCTION

Placement and Activation: The transmitter is installed in emergency vehicles, such as ambulances, and is under the control of the vehicle's driver. When an emergency arises, the driver activates the transmitter using a designated button.

Signal Transmission: Upon activation, the transmitter sends out a unique coded signal. This signal contains specific identifiers that correspond to receivers located at traffic intersections along the ambulance's route.

Communication Protocol: The transmitter is equipped with an ESP8266 microcontroller, which facilitates the wireless transmission of signals. It ensures that the signal reaches the intended receivers effectively and without interference.

RECEIVER FUNCTION

Placement and Operation: receivers are strategically installed at traffic intersections to detect incoming signals from emergency vehicle transmitters. each receiver is programmed to recognize and respond to signals from authorized transmitters only.

Signal reception and processing: when a receiver detects a signal from an approaching ambulance, it immediately processes the signal using its esp8266 microcontroller. the receiver verifies the signal's authenticity and identifies the specific intersection for which the signal is intended.

Traffic signal control: upon successful verification, the receiver triggers an immediate change in the traffic lights, granting the ambulance a clear path through the intersection. this rapid response minimizes delays and enhances the efficiency of emergency response services.

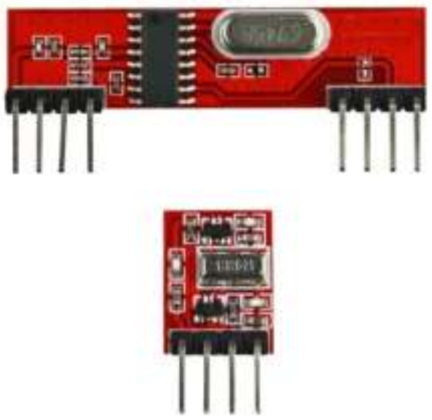


FIG 3 RF TRANSMITTER AND RECEIVER WIRELESS MODULE

4.2.4 NODEMCU ESP8266

The ESP8266 microcontroller is an integral part of the proposed smart traffic control system, providing essential communication and control functions for both normal traffic management and emergency situations. Its roles include:

COMMUNICATION INTERFACE

The ESP8266 is used for wireless communication between the ambulance's transmitter and the receivers at traffic intersections. One ESP32 module is installed in the ambulance as part of the transmitter system, and another is located at each intersection as part of the receiver system. This setup enables fast and reliable data exchange, ensuring that emergency signals are promptly recognized and acted upon.

EMERGENCY SIGNAL PROCESSING

Upon receiving a signal from the ambulance's transmitter, the ESP8266 in the receiver processes the information and triggers an immediate change in the traffic lights. This capability ensures that emergency vehicles can pass through intersections without delay, significantly improving response times.

REAL-TIME DATA HANDLING

In addition to emergency vehicle prioritization, the ESP8266 handles real-time data from the ultrasonic sensors to assess traffic density and adjust signal timings accordingly. This dual functionality allows the system to maintain efficient traffic flow while being ready to prioritize emergency vehicles when needed.

SYSTEM COORDINATION

The ESP8266 acts as a coordinator between various components of the system, including the Arduino Uno, sensors, and traffic signals. It ensures seamless integration and operation, allowing for dynamic adjustments based on current traffic conditions and emergency requirements.

SCALABILITY AND FLEXIBILITY

The ESP8266's robust processing capabilities and flexible programming environment make it ideal for scalable and adaptable traffic management solutions. It can easily accommodate additional features or be integrated with other smart city applications, enhancing the overall functionality of the traffic control system.

FIG 4 NODEMCU ESP8266

4.3 COST OF THE PROJECT:

ARDUINO UNO – Rs 770

ESP8266 – Rs 450

ULTRASONIC SENSOR – Rs 120

RF TRANSMITTER RECEIVER – Rs 120

LED - Rs 2

JUMPER WIRES – Rs 30

BREAD BOARD – Rs 60

RESISTOR (220 ohms) – Rs 9

TOTAL ESTIMATE = Rs 2278

4.4 SOFTWARE DESCRIPTION

4.4.1 INTRODUCTION

This chapter deals with the description of the software used for hardware working. Here the system uses Arduino IDE as programming software. Arduino is an integrated development environment for Embedded C that is designed for beginners. It supports different ways of stepping through the code, step- by-step expression evaluation, detailed visualization of the call stack and a mode for explaining the concepts of references and heap

4.4.2 ARDUINO IDE

Arduino IDE (Integrated Development Environment) is a software tool used for programming and development of Arduino boards. It is an open-source platform, available for free, and is compatible with multiple operating systems including Windows, Mac OS, and Linux. The main features of the Arduino IDE include:

- **CODE EDITOR:** The code editor is the main interface of the Arduino IDE, where you can write, edit and upload code to the Arduino board. It includes features such as syntax highlighting, auto-completion, and code snippets to make programming easier.

- **SKETCHES:** Arduino programs are referred to as "sketches" and can be easily created and saved within the IDE. The sketch contains two main functions: the setup () function, which is called once at the start of the program, and the loop () function, which is called repeatedly as long as the program is running.

- **LIBRARY MANAGER:** The Library Manager allows users to easily install and manage libraries for their Arduino projects. It includes a collection of pre- built libraries that can be used to add functionality to your projects. Users can also create their own libraries and add them to the IDE.

- **SERIAL MONITOR:** The Serial Monitor allows users to communicate with the Arduino board and monitor the data being sent and received through the serial port. This is particularly useful for debugging and troubleshooting.

- **BOARD MANAGER:** The Board Manager lets users to select the type of Arduino board they are using, configure settings, and install the required drivers. This is important because different Arduino boards may have different specifications and require different drivers.

- **UPLOAD:** The Upload feature allow users to upload their sketches to the Arduino board and begin executing the program. Users can select the correct board and serial port before uploading the sketch.

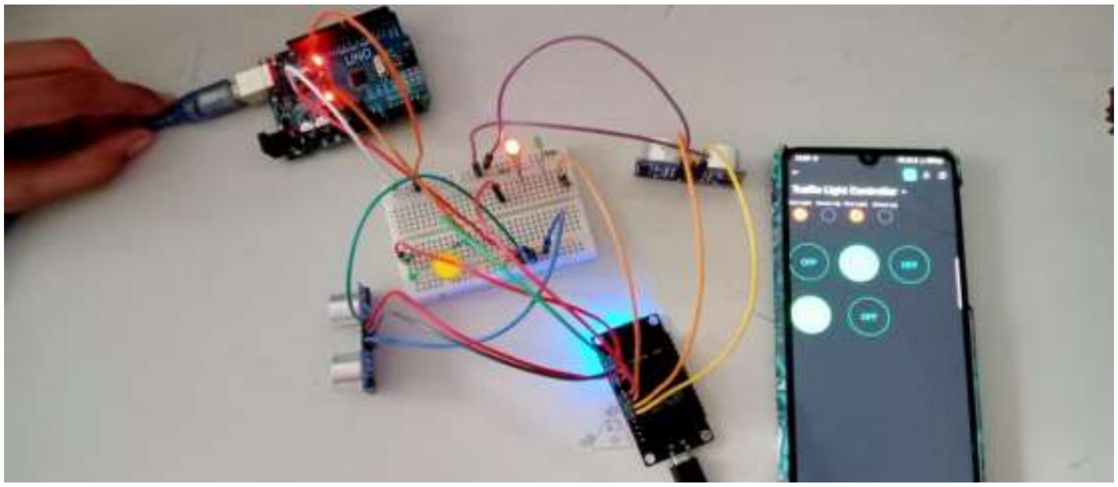
- **TOOLS:** The Tools menu includes a range of options for configuring and customizing the IDE. This includes options for setting the board type, serial port, programmer, and other settings.

Overall, the Arduino IDE is a user-friendly software tool that simplifies the programming process for beginners and experienced users alike. It is compatible with a wide range of Arduino boards and shields, making it a versatile tool for a variety of applications. With its many features and community support, the Arduino IDE is an essential tool for anyone interested in electronics and programming.

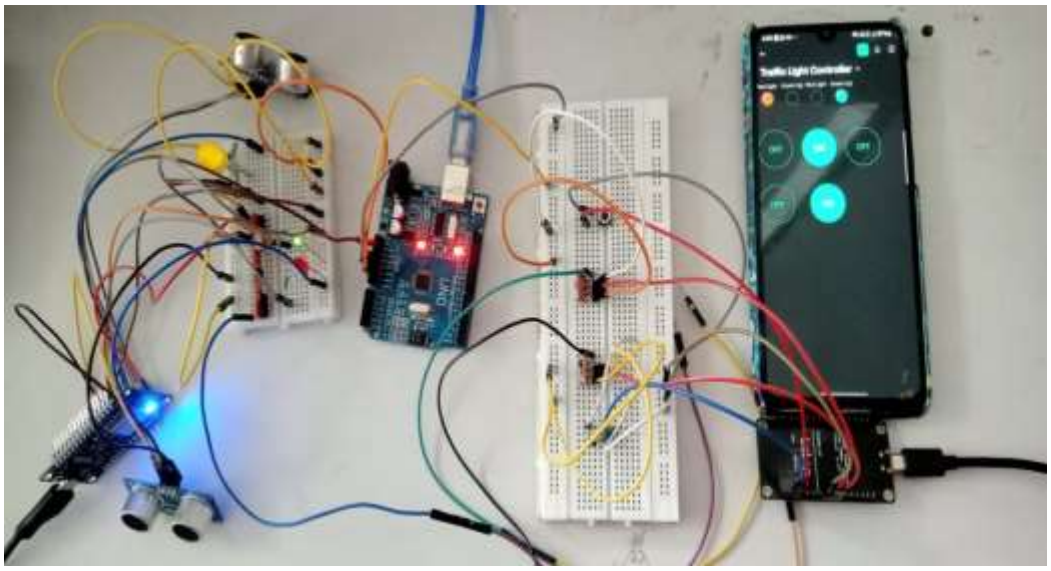
In addition to the basic features listed above, the Arduino IDE also supports advanced features such as debugging and profiling tools, version control integration, and multiple file editing. The IDE can also be extended through plugins and add-ons, allowing users to customize the tool to their specific needs. Additionally, the Arduino community provides a wealth of resources and tutorials to help users get started and troubleshoot any issues they may encounter.

4.5 HARDWARE SETUP

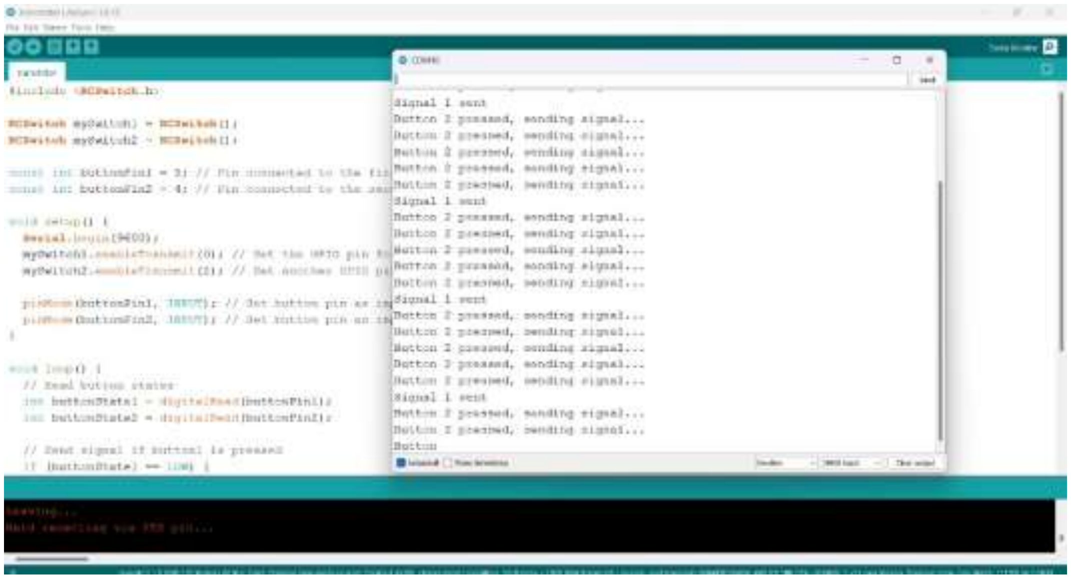
SMART TRAFFIC SIGNAL SYSTEM



TRAFFIC SIGNAL WITH EMERGENCY DETECTION SYSTEM



4.6 RESULTS

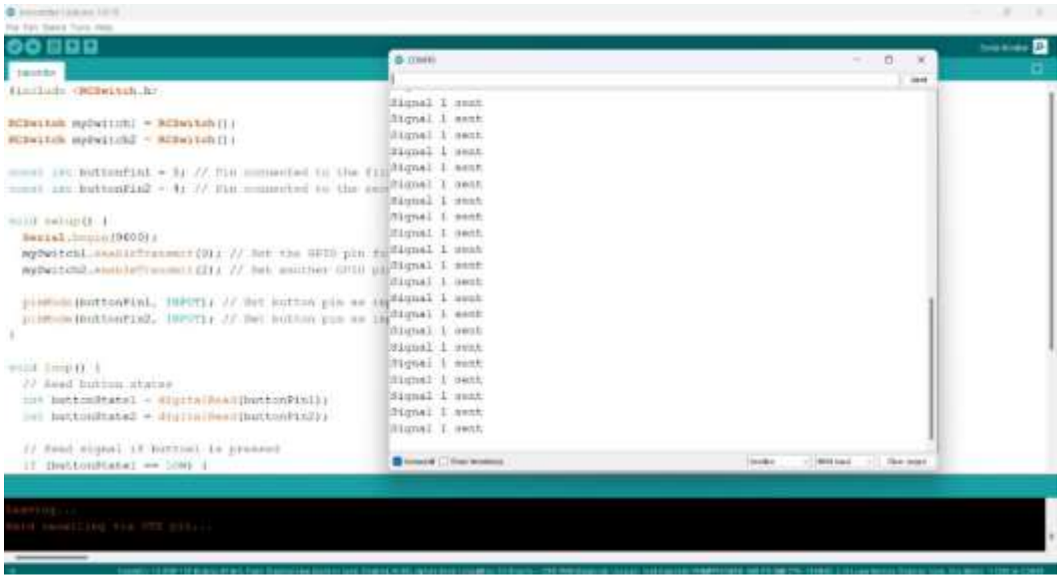


```
#include <RCSwitch.h>

RCSwitch mySwitch1 = RCSwitch();
RCSwitch mySwitch2 = RCSwitch();

const int buttonPin1 = 3; // Pin connected to the fi
const int buttonPin2 = 4; // Pin connected to the sec

void setup() {
  Serial.begin(9600);
  mySwitch1.enableTransmit(0); // Set the GPIO pin fo
  mySwitch2.enableTransmit(2); // Set another GPIO pi

  pinMode(buttonPin1, INPUT); // Set button pin as in
  pinMode(buttonPin2, INPUT); // Set button pin as in
}

void loop() {
  // Read button states
  int buttonState1 = digitalRead(buttonPin1);
  int buttonState2 = digitalRead(buttonPin2);

  // Send signal if button1 is pressed
  if (buttonState1 == LOW) {
```

```
Signal 1 sent
Button 2 pressed, sending signal...
Button 2 pressed, sending signal...
Button 2 pressed, sending signal...
Button 2 pressed, sending signal...
Button 2 pressed, sending signal...
Signal 1 sent
Button 2 pressed, sending signal...
Button 2 pressed, sending signal...
Button 2 pressed, sending signal...
Button 2 pressed, sending signal...
Button 2 pressed, sending signal...
Signal 1 sent
Button 2 pressed, sending signal...
Button 2 pressed, sending signal...
Button 2 pressed, sending signal...
Button 2 pressed, sending signal...
Button 2 pressed, sending signal...
Signal 1 sent
Button 2 pressed, sending signal...
Button 2 pressed, sending signal...
Button
```

```
Leaving...
Hard resetting via RTS pin...
```

SIGNAL SENT FROM AMBULANCE

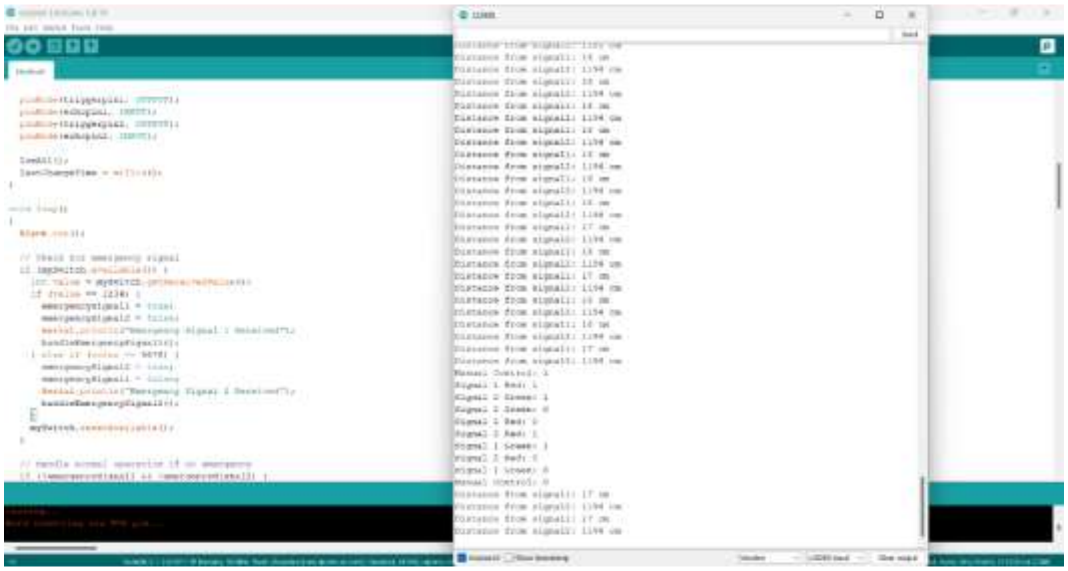


```
#include <RCSwitch.h>

RCSwitch mySwitch1 = RCSwitch();
RCSwitch mySwitch2 = RCSwitch();

const int buttonPin1 = 3; // Pin connected to the fi
const int buttonPin2 = 4; // Pin connected to the sec

void setup() {
  Serial.begin(9600);
  mySwitch1.enableTransmit(0); // Set the GPIO pin fo
  mySwitch2.enableTransmit(2); // Set another GPIO pi

  pinMode(buttonPin1, INPUT); // Set button pin as in
  pinMode(buttonPin2, INPUT); // Set button pin as in
}

void loop() {
  // Read button states
  int buttonState1 = digitalRead(buttonPin1);
  int buttonState2 = digitalRead(buttonPin2);

  // Send signal if button1 is pressed
  if (buttonState1 == LOW) {
```

```
Signal 1 sent
Signal 1 sent
Signal 1 sent
Signal 1 sent
Signal 1 sent
Signal 1 sent
Signal 1 sent
Signal 1 sent
Signal 1 sent
Signal 1 sent
Signal 1 sent
Signal 1 sent
Signal 1 sent
Signal 1 sent
Signal 1 sent
Signal 1 sent
Signal 1 sent
Signal 1 sent
Signal 1 sent
Signal 1 sent
Signal 1 sent
```

```
Leaving...
Hard resetting via RTS pin...
```

SIGNAL RECEIVED BY RECEIVER



```
  pinMode(trigPin1, OUTPUT);
  pinMode(echoPin1, INPUT);
  pinMode(trigPin2, OUTPUT);
  pinMode(echoPin2, INPUT);

  [illegible]();
  lastChangeTime = millis();
}

void loop()
{
  Blynk.run();

  // Check for emergency signal
  if (mySwitch.available()) {
    int value = mySwitch.getReceivedValue();
    if (value == 1234) {
      emergencySignal1 = true;
      emergencySignal2 = false;
      Serial.println("Emergency Signal 1 Received");
      handleEmergencySignal1();
    } else if (value == 5678) {
      emergencySignal2 = true;
      emergencySignal1 = false;
      Serial.println("Emergency Signal 2 Received");
      handleEmergencySignal2();
    }
    mySwitch.resetAvailable();
  }

  // Handle normal operation if no emergency
  if (!emergencySignal1 && !emergencySignal2) {
```

```
Distance from signal1: 1194 cm
Distance from signal1: 18 cm
Distance from signal2: 1194 cm
Distance from signal1: 18 cm
Distance from signal2: 1194 cm
Distance from signal1: 18 cm
Distance from signal2: 1194 cm
Distance from signal1: 18 cm
Distance from signal2: 1194 cm
Distance from signal1: 18 cm
Distance from signal2: 1194 cm
Distance from signal1: 18 cm
Distance from signal2: 1194 cm
Distance from signal1: 18 cm
Distance from signal2: 1194 cm
Distance from signal1: 17 cm
Distance from signal2: 1194 cm
Distance from signal1: 18 cm
Distance from signal2: 1194 cm
Distance from signal1: 17 cm
Distance from signal2: 1194 cm
Distance from signal1: 18 cm
Distance from signal2: 1194 cm
Distance from signal1: 18 cm
Distance from signal2: 1194 cm
Distance from signal1: 17 cm
Distance from signal2: 1194 cm
Manual Control: 1
Signal 1 Red: 1
Signal 2 Green: 1
Signal 2 Green: 0
Signal 1 Red: 0
Signal 2 Red: 1
Signal 1 Green: 1
Signal 2 Red: 0
Signal 1 Green: 0
Manual Control: 0
Distance from signal1: 17 cm
Distance from signal2: 1194 cm
Distance from signal1: 17 cm
Distance from signal2: 1194 cm
```

TRAFFIC CONTROL SYSTEM

CHAPTER 5

CONCLUSION & FUTURE WORK

5.1 CONCLUSION

The smart traffic control system integrates ultrasonic sensors and RF communication to enhance urban traffic management. By measuring vehicle density in real-time, it optimizes signal timings to reduce congestion. The use of RF modules prioritizes emergency vehicles, improving response times and safety. Implemented using NODEMCU and monitored via the Blynk app, this system demonstrates effective traffic flow management and improved emergency response. Its scalable design offers a valuable solution for modern cities, with potential for further refinement and expansion.

5.2 FUTURE ENHANCEMENT

Future enhancements for the smart traffic light system with emergency ambulance integration include integrating GPS modules to track ambulance locations and prioritize traffic signals based on proximity. Advanced traffic management algorithms and real-time traffic monitoring can further optimize signal timings by considering traffic density and historical data. Integrating with city traffic management systems and IoT platforms can enhance coordination and remote control. Implementing machine learning for traffic prediction and user interface improvements will offer better traffic flow and user interaction. Additional features such as automatic emergency vehicle detection, battery backup, and security measures can ensure reliable operation and safeguard against threats. Integration with public transportation systems and environmental sensors will provide comprehensive traffic management and adapt to varying conditions.