****

**NUST Balochistan Campus (NBC)**

Department of Computer Science

2nd semester (Spring), 2023

DLD Semester Project

**“Traffic Light System for Emergency Vehicles”**

**Submitted to:** Engr Muhammad Abid Hussain

**Submitted by:**

* Muhammad Haroon
* Khadija
* Muhammad Anas
* Muhammad Ahsan
* Muhammad Ibtesam

**“**Traffic Light System for Emergency Vehicles”

**Abstract:**

Traffic congestion is a widespread issue impacting transportation systems. A traffic light controller system using RF transmission and Arduino Uno has been developed to address this. It restores normal traffic light sequences when emergency vehicles pass through intersections, reducing accidents. The prototype operates at 433 MHz and has the potential for real-time traffic management and improved traffic light technology.

**Introduction:**

This project aims to develop an efficient traffic light system to facilitate the smooth passage of emergency vehicles. Using RF transmission, the system grants priority to emergency vehicles by temporarily adjusting traffic light sequences. The implementation minimizes traffic complications and optimizes emergency response efforts. Future enhancements may include real-time traffic analysis and integration with other systems to further improve efficiency. Overall, the project contributes to society by enabling swift and effective responses to emergencies while minimizing traffic disruptions.

**Components:**

* 433 MHz RF transmitter
* Receiver
* Arduino Uno
* LEDs
* Wires
* Battery
* Switch

**Circuit diagram:**

The project has two parts:

1. Transmitter: (Emergency Vehicles)
2. Receiver

**Transmitter circuit:**

A computer screen shot of a circuit board

Description automatically generated

**Fig: 01**

**Receiver circuit:**

A computer screen shot of a circuit board

Description automatically generated

**Fig: 02**

**Hardware Description:**

Given are the description of the hardware used in the project:

* **433 MHz RF transmitter:**

A 433 MHz RF transmitter is a wireless communication device that operates at the frequency of 433 megahertz (MHz). It is commonly used for short-range communication in applications such as remote-control systems, wireless data transmission, and home automation. The transmitter generates an RF signal at 433 MHz, which is modulated with data using techniques like ASK or FSK. It amplifies the signal before transmitting it through an antenna, covering distances of a few hundred meters. These transmitters are cost-effective, simple to use, and suitable for low-power applications like remote controls, security systems, and IoT devices.

* **Receiver:**

The receiver for a 433 MHz RF transmitter is the complementary device that receives and decodes the transmitted signals. It is designed to work in tandem with the transmitter and is commonly used in various wireless communication applications.

The receiver consists of an antenna that captures the RF signals transmitted by the 433 MHz transmitter. The captured signals are then passed to a demodulation circuit, which extracts the original data from the modulated RF signal. Depending on the type of modulation used in the transmitter (e.g., ASK or FSK), the receiver employs the appropriate demodulation technique.

Once demodulated, the receiver processes the data and sends it to the intended destination or application. This could be a microcontroller, a computer, or any other device capable of interpreting the received data.

The range of the receiver also depends on factors such as the sensitivity of its components, the quality of the antenna, and the environment in which it operates. In open spaces, a well-designed 433 MHz RF receiver can achieve a range similar to that of the transmitter, which is typically a few hundred meters.

433 MHz RF receivers are widely used in applications where short-range wireless communication is required, such as remote controls, wireless sensors, telemetry systems, and home automation devices. They provide a convenient and efficient means of wirelessly transmitting data between devices over relatively short distances.

* **Arduino Uno:**

The Arduino Uno is a popular open-source microcontroller board based on the Atmel ATmega328P microcontroller. It is one of the fundamental boards in the Arduino family and is widely used in various electronic projects and prototyping.

The board features a compact design with input/output (I/O) pins, digital and analog pins, and other essential components that allow users to interact with external sensors, actuators, and electronic components. It can be powered either through a USB connection or an external power supply.

The Arduino Uno is programmed using the Arduino Integrated Development Environment (IDE), which offers a beginner-friendly platform for writing and uploading code to the board. The programming language is based on Wiring, a simplified variant of C++.

With its versatility, ease of use, and vast community of developers, the Arduino Uno has become a popular choice for hobbyists, students, and professionals alike. It is commonly used in various projects such as robotics, home automation, sensor monitoring, and interactive art installations, enabling users to bring their ideas to life with accessible and affordable hardware.

**Software:**

In this, project the software platform used is given below:

1. **Proteus 8 professional:**

Proteus is a simulation software used for electronic circuit design and microcontroller-based projects. It offers a virtual environment to design, simulate, and test analog, digital, and mixed-signal circuits. It is popular for simulating microcontroller behavior and includes a vast library of electronic components, aiding electronics engineers, hobbyists, and students in designing and validating projects.

1. **Arduino IDE:**

The Arduino IDE (Integrated Development Environment) is a user-friendly software used for programming Arduino microcontroller boards. It provides a simple interface for writing, compiling, and uploading code to Arduino boards, making it accessible for beginners and experienced developers alike. With a variety of built-in functions and libraries, the Arduino IDE enables users to create and control a wide range of electronic projects easily.

We used Arduino IDE to code for both receiver and transmitter part.

**Transmitter code:**

|  |
| --- |
| **#include <VirtualWire.h>**  **#define button 6**  **#define button1 7**  **#define button2 8**  **#define button3 9**  **char \*data;**  **int val, val1, val2, val3;**  **int value = 0;**  **int value1 = 0;**  **int value2 = 0;**  **int value3 = 0;**  **void setup()**  **{**  **vw\_set\_tx\_pin(12);**  **vw\_setup(2000);**  **pinMode(button, INPUT\_PULLUP);**  **pinMode(button1, INPUT\_PULLUP);**  **pinMode(button2, INPUT\_PULLUP);**  **pinMode(button3, INPUT\_PULLUP);**  **Serial.begin(9600);**  **}**  **void loop()**  **{**  **val = digitalRead(button);**  **val1 = digitalRead(button1);**  **val2 = digitalRead(button2);**  **val3 = digitalRead(button3);**  **if (val == LOW && value == 0)**  **{**  **data = "a";**  **vw\_send((uint8\_t \*)data, strlen(data));**  **vw\_wait\_tx();**  **value = 1;**  **Serial.println("Button 'a' pressed");**  **}**  **else if (val1 == LOW && value1 == 0)**  **{**  **data = "b";**  **vw\_send((uint8\_t \*)data, strlen(data));**  **vw\_wait\_tx();**  **value1 = 1;**  **Serial.println("Button 'b' pressed");**  **}**  **else if (val2 == LOW && value2 == 0)**  **{**  **data = "c";**  **vw\_send((uint8\_t \*)data, strlen(data));**  **vw\_wait\_tx();**  **value2 = 1;**  **Serial.println("Button 'c' pressed");**  **}**  **else if (val3 == LOW && value3 == 0)**  **{**  **data = "d";**  **vw\_send((uint8\_t \*)data, strlen(data));**  **vw\_wait\_tx();**  **value3 = 1;**  **Serial.println("Button 'd' pressed");**  **}**  **delay(100); // Add a small delay to debounce the buttons**  **}** |

**Receiver Code:**

|  |
| --- |
| **#include <VirtualWire.h>**  **int lane1[]={1,2,3};**  **int lane2[]={4,5,6};**  **int lane3[]={7,8,9};**  **int lane4[]={10,11,13};**  **void setup()**  **{**  **// put your setup code here, to run once:**  **//reciver file**  **vw\_set\_rx\_pin(12);**  **vw\_setup(2000);**  **vw\_rx\_start();**  **//traffic lights**  **for(int i=0; i<3; i++)**  **{**  **pinMode(lane1[i],OUTPUT);**  **pinMode(lane2[i],OUTPUT);**  **pinMode(lane3[i],OUTPUT);**  **pinMode(lane4[i],OUTPUT);**  **}**  **for(int i=0; i<3; i++)**  **{**  **digitalWrite(lane1[i],LOW);**  **digitalWrite(lane2[i],LOW);**  **digitalWrite(lane4[i],LOW);**  **}**  **}  digitalWrite(lane3[i],LOW);**    **void loop()**  **{**  **// put your main code here, to run repeatedly:**  **uint8\_t buf[VW\_MAX\_MESSAGE\_LEN];**  **uint8\_t buflen = VW\_MAX\_MESSAGE\_LEN;**  **//l1 green high**  **digitalWrite(lane1[2],HIGH);**  **//lane 1**  **if (vw\_get\_message(buf, &buflen))**  **{**  **if (buflen >= 1)**  **{**  **switch (buf[0])**  **{**  **// case 'a':**    **//   break;**  **case 'b':**  **digitalWrite(lane1[2],LOW);**  **digitalWrite(lane1[1],HIGH);**  **digitalWrite(lane2[1],HIGH);**  **delay(2000);**  **digitalWrite(lane1[1],LOW);**  **digitalWrite(lane2[1],LOW);**  **digitalWrite(lane2[2],HIGH);**  **digitalWrite(lane1[0],HIGH);**  **delay(2000);**  **digitalWrite(lane2[2],LOW);**  **digitalWrite(lane2[1],HIGH);**  **digitalWrite(lane1[0],LOW);**  **digitalWrite(lane1[1],HIGH);**  **delay(2000);**  **digitalWrite(lane2[1],LOW);**  **digitalWrite(lane1[1],HIGH);**  **break;**  **case 'd':**  **digitalWrite(lane2[0], HIGH);**  **digitalWrite(lane1[2], LOW);**  **digitalWrite(lane1[1], HIGH);**  **digitalWrite(lane4[0], LOW);**  **digitalWrite(lane4[1], HIGH);**  **delay(2000);**    **digitalWrite(lane1[1], LOW);**  **digitalWrite(lane4[1], LOW);**  **digitalWrite(lane4[0], LOW);**  **digitalWrite(lane4[2], HIGH);**  **digitalWrite(lane1[0], HIGH);**  **delay(2000);**  **digitalWrite(lane4[2], LOW);**  **digitalWrite(lane1[0], LOW);**  **digitalWrite(lane1[1], HIGH);**  **break;**  **}**  **}**  **}**  **digitalWrite(lane2[0],HIGH);**  **digitalWrite(lane3[0],HIGH);**  **digitalWrite(lane4[0],HIGH);**  **delay(2000);**  **digitalWrite(lane1[2],LOW);**  **digitalWrite(lane3[0],LOW);**  **digitalWrite(lane1[1],HIGH);**  **digitalWrite(lane3[1],HIGH);**  **delay(2000);**  **digitalWrite(lane3[1],LOW);**  **//l3 green high**  **digitalWrite(lane3[2],HIGH);**  **//lane 3**  **if (vw\_get\_message(buf, &buflen))**  **{**  **if (buflen >= 1)**  **{**  **switch (buf[0])**  **{**  **// case 'a':**    **// break;**  **case 'a':**  **digitalWrite(lane3[2],LOW);**  **digitalWrite(lane3[1],HIGH);**  **digitalWrite(lane1[1],HIGH);**  **delay(2000);**  **digitalWrite(lane3[0],HIGH);**  **digitalWrite(lane3[1],LOW);**  **digitalWrite(lane1[1],LOW);**  **digitalWrite(lane1[2],HIGH);**    **delay(2000);**    **digitalWrite(lane1[2],LOW);**  **digitalWrite(lane1[1],HIGH);**  **digitalWrite(lane3[0],HIGH);**  **delay(2000);**  **digitalWrite(lane3[0],LOW);**  **digitalWrite(lane3[1],HIGH);**  **break;**  **case 'b':**  **digitalWrite(lane1[1],LOW);**  **digitalWrite(lane1[0],HIGH);**  **digitalWrite(lane3[2], LOW);**  **digitalWrite(lane3[1], HIGH);**  **delay(2000);**  **digitalWrite(lane2[1], LOW);**  **digitalWrite(lane2[1], HIGH);**  **digitalWrite(lane3[1], LOW);**  **digitalWrite(lane3[0], HIGH);**  **digitalWrite(lane2[0],LOW);**  **delay(2000);**  **digitalWrite(lane2[1], LOW);**  **digitalWrite(lane2[2], HIGH);**  **delay(2000);**  **digitalWrite(lane2[2], LOW);**  **digitalWrite(lane2[1],HIGH);**  **digitalWrite(lane3[1],HIGH);**  **digitalWrite(lane3[0], LOW);**  **delay(2000);**  **digitalWrite(lane2[1],LOW);**  **digitalWrite(lane2[0],HIGH);**  **break;**  **}**  **}**  **}**  **digitalWrite(lane1[1],LOW);**  **digitalWrite(lane1[1],LOW);**  **digitalWrite(lane1[0],HIGH);**  **delay(2000);**  **digitalWrite(lane3[2],LOW);**  **digitalWrite(lane4[0],LOW);**  **digitalWrite(lane3[1],HIGH);**  **digitalWrite(lane4[1],HIGH);**  **delay(2000);**  **//l4 green high**  **digitalWrite(lane4[2],HIGH);**  **//lane 4**  **if (vw\_get\_message(buf, &buflen))**  **{**  **if (buflen >= 1)**  **{**  **switch (buf[0])**  **{**  **// case 'a':**    **//   break;**  **case 'a':**  **digitalWrite(lane3[1],LOW);**  **digitalWrite(lane3[0],HIGH);**  **digitalWrite(lane4[2],LOW);**  **digitalWrite(lane4[1],HIGH);**  **digitalWrite(lane1[1],HIGH);**  **delay(2000);**  **digitalWrite(lane4[1],LOW);**  **digitalWrite(lane1[1],LOW);**  **digitalWrite(lane1[0],LOW);**  **digitalWrite(lane4[0],HIGH);**  **digitalWrite(lane1[2],HIGH);**  **delay(2000);**  **digitalWrite(lane1[2],LOW);**  **digitalWrite(lane4[0],LOW);**  **digitalWrite(lane1[1],HIGH);**  **delay(2000);**  **digitalWrite(lane1[1],LOW);**  **digitalWrite(lane1[0],HIGH);**  **digitalWrite(lane4[0],HIGH);**  **// delay(2000);**  **break;**  **case 'c':**  **digitalWrite(lane4[2], LOW);**  **digitalWrite(lane3[1], HIGH);**  **digitalWrite(lane4[1], HIGH);**  **delay(2000);**  **digitalWrite(lane4[1], LOW);**  **digitalWrite(lane3[1], LOW);**  **digitalWrite(lane4[0], HIGH);**  **digitalWrite(lane3[2], HIGH);**  **delay(2000);**  **digitalWrite(lane4[1], HIGH);**    **digitalWrite(lane3[2], LOW);**      **digitalWrite(lane3[1], HIGH);**    **delay(700);**  **digitalWrite(lane4[0], LOW);**  **break;**  **}**  **}**  **}**    **digitalWrite(lane3[1],LOW);**  **digitalWrite(lane4[1],LOW);**  **digitalWrite(lane3[0],HIGH);**  **delay(2000);**  **digitalWrite(lane4[2],LOW);**  **digitalWrite(lane2[0],LOW);**  **digitalWrite(lane4[1],HIGH);**  **digitalWrite(lane2[1],HIGH);**  **delay(2000);**  **//l2 green high**  **digitalWrite(lane2[2],HIGH);**  **//lane 2**  **if (vw\_get\_message(buf, &buflen))**  **{**  **if (buflen >= 1)**  **{**  **switch (buf[0])**  **{**  **// case 'a':**  **case 'c':**  **digitalWrite(lane4[1], LOW);**  **digitalWrite(lane4[0], HIGH);**  **digitalWrite(lane2[2], LOW);**  **digitalWrite(lane2[1], HIGH);**  **digitalWrite(lane3[0], LOW);**  **digitalWrite(lane3[1], HIGH);**  **delay(2000);**  **digitalWrite(lane2[1], LOW);**  **digitalWrite(lane3[1], LOW);**  **digitalWrite(lane2[0], HIGH);**  **digitalWrite(lane3[2], HIGH);**  **delay(2000);**  **digitalWrite(lane2[1], HIGH);**    **digitalWrite(lane2[1], LOW);**  **delay(2000);**  **digitalWrite(lane3[1], LOW);**  **// digitalWrite(lane3[0], HIGH);**    **digitalWrite(lane3[2], LOW);**  **// digitalWrite(lane3[0], LOW);**  **digitalWrite(lane3[1], HIGH);**  **// digitalWrite(lane3[1], LOW);**  **break;**  **case 'd':**  **digitalWrite(lane2[2], LOW);**  **digitalWrite(lane2[1], HIGH);**  **digitalWrite(lane4[1], HIGH);**  **delay(2000);**  **digitalWrite(lane2[1], LOW);**  **digitalWrite(lane4[1], LOW);**  **digitalWrite(lane2[0], HIGH);**  **digitalWrite(lane4[2], HIGH);**  **// digitalWrite(lane2[1], HIGH);**  **// digitalWrite(lane2[0], LOW);**  **delay(2000);**  **digitalWrite(lane4[2], LOW);**  **digitalWrite(lane2[0], LOW);**  **digitalWrite(lane2[1], HIGH);**  **delay(2000);**  **break;**  **}**  **}**  **}**  **digitalWrite(lane4[1],LOW);**  **digitalWrite(lane2[1],LOW);**  **digitalWrite(lane4[0],HIGH);**  **delay(2000);**  **digitalWrite(lane1[0],LOW);**  **digitalWrite(lane2[2],LOW);**  **digitalWrite(lane1[1],HIGH);**  **digitalWrite(lane2[1],HIGH);**  **delay(2000);**  **digitalWrite(lane2[1],LOW);**  **digitalWrite(lane1[1],LOW);**  **}** |

**Working Methodology:**

**A diagram of a circuit board

Description automatically generated**

Traffic Light Control System with Sequential and Emergency Modes

Modes:

1. Sequential Mode:

- Traffic lights operate in a predetermined sequence.

- Only one traffic light is green at any given time, while the others display red or yellow signals.

- The sequence continues uninterrupted until an external signal is received.

2. Emergency Mode:

- Activated when an emergency vehicle approaches the junction.

- Each lane leading to the intersection is equipped with a corresponding button (a, b, c, d) for the emergency vehicle to trigger.

- When the emergency vehicle approaches from Lane 1 and presses Button "a," the system activates "Case a."

- The same principle applies to Lanes 2, 3, and 4 with buttons "b," "c," and "d," respectively.

- Each case is designed to interrupt the ongoing sequence of the traffic lights and prioritize the lane associated with the pressed button.

- As a result, the green light is immediately activated for the corresponding lane, allowing the emergency vehicle to pass through the intersection unimpeded.

Benefits and Features:

1. Efficient and Flexible Traffic Management:

- The system provides an efficient solution for managing traffic at intersections.

- Under normal circumstances, the sequential mode ensures a smooth flow of traffic.

- The predetermined sequence helps to optimize traffic movements and reduce congestion.

2. Prioritization of Emergency Vehicles:

- When an emergency vehicle triggers the corresponding button, the system swiftly transitions into emergency mode.

- The traffic light sequence is interrupted to prioritize the lane associated with the pressed button.

- By activating the green light for the emergency vehicle's lane, it can pass through the intersection without delay.

3. Seamless Mode Transition:

- The system seamlessly transitions between sequential and emergency modes as required.

- When an external signal is received from the emergency vehicle, the system automatically switches to emergency mode.

- Once the emergency vehicle passes or the emergency situation is resolved, the system returns to the sequential mode.

4. Precise and Reliable Control:

- Each button press corresponds to a specific case (a, b, c, d) designed to handle the associated lane.

- This approach ensures precise control over the traffic light sequence and reliable prioritization of emergency vehicles.

- The system guarantees accurate responses and minimizes the risk of errors or confusion during emergency situations.

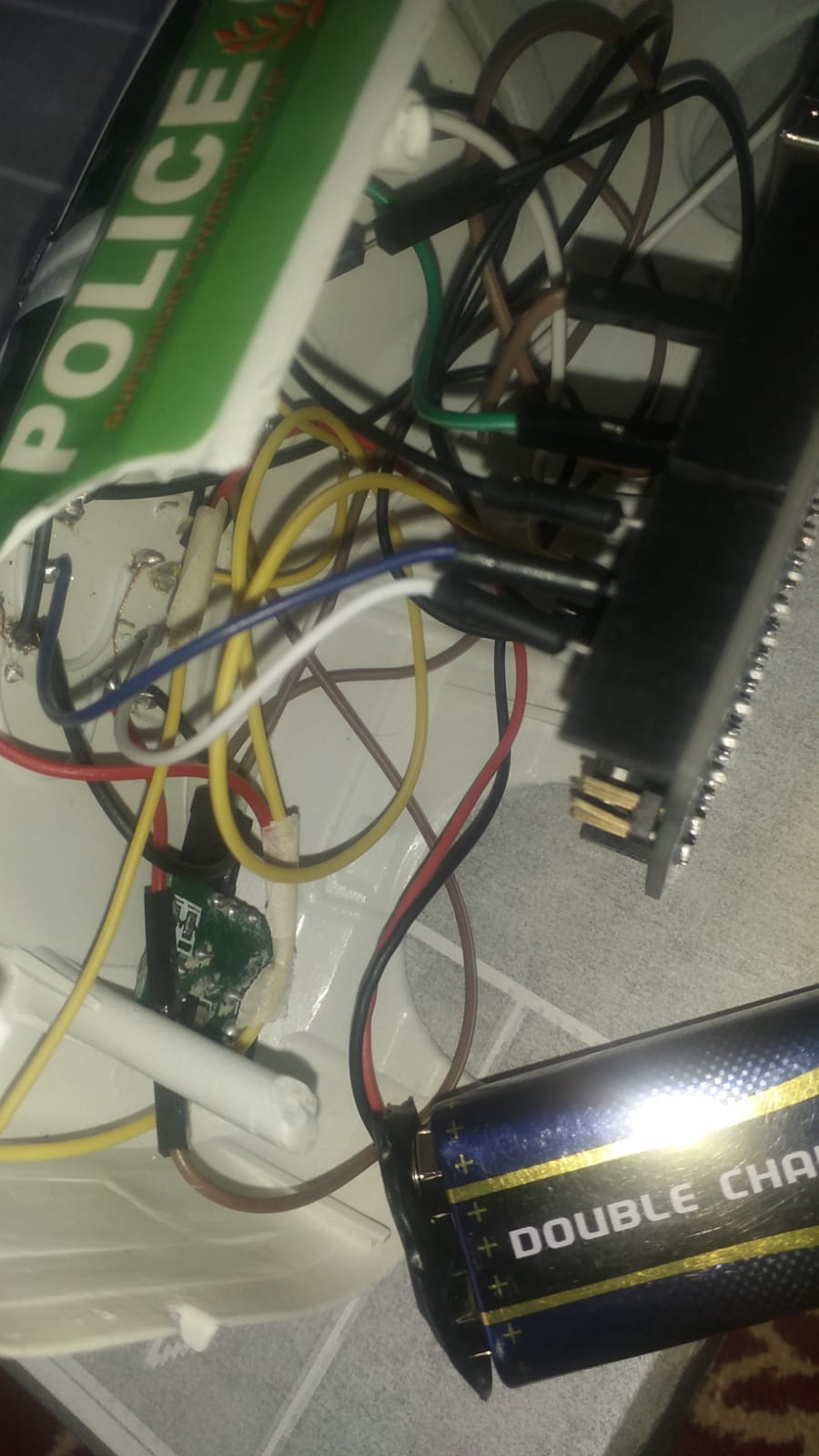
By implementing this Traffic Light Control System, we can effectively manage traffic at intersections, ensuring efficient movement of vehicles while prioritizing the passage of emergency vehicles during critical situations. The system's ability to seamlessly transition between sequential and emergency modes, along with precise control over the traffic light sequence, provides a reliable solution for catering to both regular traffic flow and emergency scenarios.

**Hardware part:**

A toy police car on a road

Description automatically generated

**Transmitter:**



**Receiver:**

A circuit board with wires and numbers

Description automatically generated

**Result:**

The implementation of the Traffic Light Control System with Sequential and Emergency Modes has resulted in significant improvements in traffic management and emergency vehicle prioritization at intersections. The following outcomes have been observed:

1. Efficient Traffic Flow: The Sequential Mode of the system ensures smooth traffic flow during regular circumstances. The predetermined sequence optimizes vehicle movements, reducing congestion and improving overall traffic efficiency.

2. Effective Emergency Response: The Emergency Mode of the system has proven to be highly effective in prioritizing emergency vehicles. When an emergency vehicle approaches the intersection and triggers the corresponding button, the system swiftly interrupts the ongoing sequence and grants immediate passage to the emergency vehicle. This feature has significantly reduced response times for emergency services, potentially saving lives and minimizing damage during critical situations.

3. Seamless Mode Transition: The system's ability to seamlessly transition between Sequential and Emergency Modes has been instrumental in maintaining the balance between regular traffic flow and emergency vehicle prioritization. The system automatically switches to Emergency Mode upon receiving the external signal from an emergency vehicle and smoothly returns to Sequential Mode once the emergency vehicle has passed or the emergency situation has been resolved.

4. Precise and Reliable Control: The utilization of distinct cases for each button press (a, b, c, d) ensures precise and reliable control over the traffic light sequence. This methodology eliminates ambiguity and guarantees accurate responses to emergency situations. The system's reliability enhances overall traffic management and emergency vehicle prioritization.

Overall, the implementation of this working methodology has resulted in improved traffic management efficiency and effective prioritization of emergency vehicles. By seamlessly transitioning between Sequential and Emergency Modes and providing precise control over the traffic light sequence, the system offers a flexible solution that caters to the needs of both regular traffic and emergency situations. This methodology contributes to safer and more efficient traffic flow, ultimately enhancing the overall transportation infrastructure of the area.

**Conclusion:**

In conclusion, this project presents an efficient traffic light system designed to prioritize emergency vehicles' passage. By incorporating advanced technologies, such as RF transmitters and receivers, the system ensures swift and effective responses to critical situations. During emergencies, the traffic lights grant priority to the emergency vehicle, optimizing traffic flow and minimizing disruptions. The project's successful implementation contributes to the community's well-being by enhancing emergency response efforts and mitigating the challenges posed by traffic congestion. Future iterations may further enhance the system's functionality and integrate with other traffic management technologies for even greater efficiency.

**References:**

Research article (<https://www.researchgate.net/profile/Nik-Mohd-Zarifie-Hashim/publication/307801912_Traffic_Light_Control_System_for_Emergency_Vehicles_Using_Radio_Frequency/links/629967916886635d5cb9b697/Traffic-Light-Control-System-for-Emergency-Vehicles-Using-Radio-Frequency.pdf>)