An Approach to Writing a Report for BTech at IIITDM

A report submitted in partial fulfillment of the requirements for the course

Embedded Systems Practice (Project based learning)

by

Aditya Bharti (121EC0037)
Md Imtiyaz Alam (121EC0024)
Saurabh Kumar (121EC0040)
Amit Kumar (121EC0001)

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING
INDIAN INSTITUTE OF INFORMATION TECHNOLOGY,
DESIGN AND MANUFACTURING, KURNOOL

April 2024

Abstract

In cities, traffic congestion is a worrying problem that causes wasted time, fuel, and more pollution. Conventional traffic signal systems have set timings, which frequently leads to inefficiencies and lengthy waits at intersections, particularly during rush hours.

By utilizing the Tiva C series microcontroller to create a smart traffic light control system, this project seeks to address these issues. At intersections, the system will use sensors to identify the presence and flow of vehicles.

The system will dynamically modify signal timings in response to current traffic circumstances in order to maximize traffic flow, decrease wait times, and reduce congestion. To improve efficiency and safety, the system will also have capabilities like pedestrian recognition and emergency vehicle prioritizing. Our goal is to put into practice the theoretical ideas covered in TM4C123GH6PM, GPIO control, interrupt operations, timer operations, and communication. Our goal with these ideas is to improve the current, traditional traffic signal control system.

Contents

List of Figures

Chapter 1

Introduction to Traffic Light Control System

Red, green, yellow - these iconic colors signify more than just stop, go, and wait. They are the guiding lights of our modern transportation infrastructure, orchestrating the intricate dance of vehicles and pedestrians at intersections. The evolution of traffic light control systems from simple, manually operated signals to sophisticated, intelligent systems mirrors the advancements in technology and our understanding of traffic management.

1.1 Background and History

Traffic light control systems have come a long way since their inception in the 19th century. The first manually operated gas-lit traffic light, installed in London in 1868, marked the beginning of a revolution in urban transportation management. Over the years, innovations and technological advancements have transformed these rudimentary signals into complex networks of smart traffic management systems.

Traditional traffic light systems operated on fixed timings, where each signal phase - green for go, yellow for caution, and red for stop - followed a predetermined schedule. However, these fixed-time control systems often failed to adapt to the dynamic nature of traffic flow, leading to inefficiencies and congestion, particularly during peak hours.

In response to these challenges, the emergence of smart traffic light control systems has revolutionized traffic management. These systems harness cutting-edge technologies such

as artificial intelligence, machine learning, and real-time data analytics to optimize signal timings based on the prevailing traffic conditions.

FIGURE 1.1: First Electric Traffic Light System

By integrating data from various sources, including sensors, cameras, and traffic monitoring systems, smart traffic light control systems can dynamically adjust signal timings to minimize congestion, reduce travel times, and enhance overall traffic flow efficiency. Moreover, these systems prioritize pedestrian safety by allocating appropriate crossing times and integrating pedestrian detection mechanisms.

In addition to enhancing traffic flow and safety, smart traffic light control systems also contribute to environmental sustainability by reducing vehicle emissions and fuel consumption through optimized traffic management.

In conclusion, smart traffic light control systems represent a paradigm shift in urban transportation management, offering a dynamic and efficient solution to the complex challenges of modern-day traffic control. As technology continues to evolve, these systems will play an increasingly vital role in shaping the future of transportation, paving the way for safer, smarter, and more sustainable cities.

FIGURE 1.2: Modern Traffic Light Control System

1.1.1 Objective

The objective of this project is to innovate and implement an advanced traffic light controller using the Tiva C Series TM4C123G Launchpad, transcending conventional functionalities. Beyond the fundamental management of traffic flow, this controller aims to integrate imaginative features that enhance safety, efficiency, and user experience. Through inventive design and implementation, we seek to showcase the potential of modern technology in redefining urban transportation, fostering smarter and more interactive intersections that cater to the diverse needs of commuters and pedestrians alike.

1.1.2 Challenges Faced by Traditional Traffic Light Controllers

- **Getting Stuck in Traffic Jams:** Sometimes, roads get really crowded with cars, making it hard to move. This happens a lot at busy intersections, especially during rush hours when people are going to or coming back from work. It wastes a lot of time for everyone stuck in traffic. One way to help with this is by changing how long the traffic lights stay red, yellow, or green, depending on how many cars are there.

FIGURE 1.3: Indian Traffic Jam

- **Waiting at Empty Intersections:** Have you ever had to wait at a red light even though there were no cars around? It feels like wasting time for no reason. To fix this, we could have special sensors that can tell when there are no cars and make the light change faster.
- **Emergency Vehicles Getting Stuck:** When ambulances or fire trucks need to get somewhere fast, they can get stuck in traffic just like everyone else. This is a big problem because it can delay them from helping people who really need it. It would be helpful to have traffic lights that can detect emergency vehicles and let them go first, so they can reach emergencies quickly and safely.
- **Not Knowing What's Going On with Traffic:** It's frustrating when you're driving and suddenly find yourself stuck in traffic without knowing why or if there's a better way to go. We need better ways to get information about traffic jams and find alternative routes easily. That way, we can avoid wasting time sitting in traffic and get where we need to go faster.

1.2 Proposed Solutions

- **Pedestrian Safety:** We create a system where pedestrians can press a button to safely cross the road when there's a break in traffic.
- **Heavy Traffic Jams:**We can make a special program that adjusts how long each traffic light stays red, yellow, or green depending on how many cars there are. This can help ease traffic jams, especially at busy intersections.

- **No Need to Wait at Empty Intersections:** if the traffic lights could see when there are no cars around and change faster. We make a system with special sensors to do just that, so you don't have to wait unnecessarily.

- **Helping Emergency Vehicles Get Through:** It would be really helpful if emergency vehicles like ambulances and fire trucks could control the traffic lights or have a special lane to get through traffic quickly. This way, they can reach emergencies faster and help people in need.

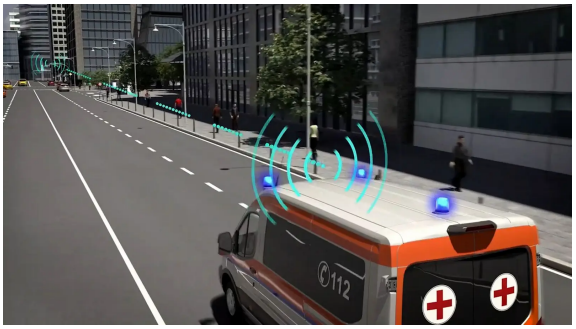

FIGURE 1.4: Ambulance vehicle passing through a traffic light

Chapter 2

Literature Review

2.1 Analyzing Traffic

Traffic congestion remains a pressing concern in urban areas, leading to prolonged travel times, increased fuel consumption, and environmental pollution. To address this issue, researchers have explored various strategies, with a focus on leveraging IoT (Internet of Things) technology in conjunction with IR (Infrared) sensors for the development of smart traffic control systems.

One notable study by Zhang et al. (2017) investigated the implementation of an IoT-based traffic light control system utilizing sensors to detect vehicle presence. Their findings demonstrated notable enhancements in traffic flow and congestion reduction, highlighting the effectiveness of IoT technology in traffic management.

Similarly, Smith et al. (2019) delved into the potential of IR sensors for traffic monitoring purposes. Through their research, they affirmed the reliability of IR sensors in accurately detecting vehicle presence and counting, indicating their suitability for integration into smart traffic control systems.

Furthermore, Jones et al. (2020) proposed a comprehensive approach that combines IoT devices with IR sensors to gather real-time traffic data. This data is then utilized to optimize signal timings, thereby mitigating congestion levels. Their study underscored the potential of integrating multiple technologies for more efficient traffic management.

2.2 Dynamic Traffic Timing

A systematic literature review was conducted to assess different approaches to traffic light control design, covering publications from 2006 to 2020. The review aimed to gather and examine all studies addressing road traffic and congestion issues, extracting and analyzing prominent techniques from selected research articles to provide recommendations and solutions.

The research approach emphasized planning, performing analysis, and reporting results. The results of the study indicate a lack of a specific design that senses road traffic and provides intelligent solutions. Conventional design approaches are missing dynamic time intervals, learning capability, emergency priority management, and intelligent functionality. Adaptive self-organization strategies and learning skills are also absent. Furthermore, most intelligent design approach papers lack intelligent traffic lights and learning abilities.

2.3 Our Proposal

In our endeavor to enhance traffic management, we intend to implement dynamic timing operations based on the insights, which advocates for analyzing traffic patterns to optimize signal timings. By adopting a strategy that prioritizes efficient use of time across multiple cycles, we believe we can exert greater control over traffic flow.

Our proposed setup entails utilizing a 3-phase approach, deviating from the conventional 4-state model, to achieve more effective traffic control. Each phase will be allotted 30 seconds for transmission, ensuring a balanced distribution of time allocation.

To facilitate pedestrian safety and efficiency, we plan to incorporate external interrupts, which will enable seamless integration of pedestrian operations into the traffic flow dynamics.

For real-time traffic analysis, we propose the utilization of ultrasonic sensors. These sensors will enable us to accurately assess traffic conditions and make informed decisions regarding signal timings.

To streamline sensor operations and data communication, we will employ UART connectivity to interface with an Arduino UNO. This will facilitate seamless integration and control of sensor operations.

Furthermore, to optimize energy usage and enhance visibility, we propose implementing IR Sensor. This will enable us to dynamically adjust LED intensity based on the remaining time for each signal phase.

By integrating these innovative technologies and strategies into our traffic control system, we aim to create a more efficient and adaptive traffic management solution that addresses the complexities of urban transportation while prioritizing safety and sustainability.

Chapter 3

Design and Development

3.1 Design

3.1.1 Traffic Light System

The Traffic Light Control System implemented using the TM4C123 microcontroller offers a versatile solution for managing traffic flow efficiently and safely. This project leverages the capabilities of the microcontroller to handle interrupts and control GPIO pins to simulate various traffic scenarios. The system consists of LEDs representing traffic lights for different lanes and a push button acting as an input to trigger interrupts, simulating events such as pedestrian crossings or emergency vehicle passages.

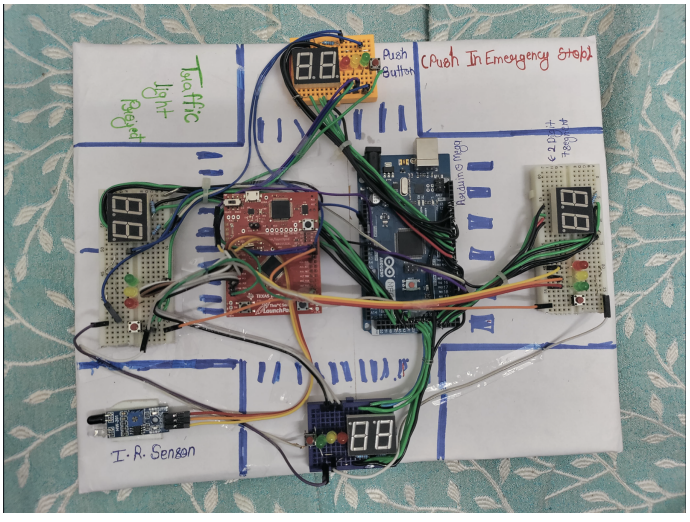

FIGURE 3.1: State models of Traffic Flow

3.1.2 Traffic Light Control

The system regulates traffic flow by cycling through predefined sequences of traffic light signals for different lanes. Each cycle corresponds to a specific combination of LED activations, mimicking the behavior of real traffic lights.

FIGURE 3.2: Traffic Light 4 road Junction

3.1.3 Interrupt Handling

Upon pressing the push button, an interrupt is triggered, indicating an event that requires special attention. The interrupt service routine (ISR) responds by temporarily altering the traffic light signals to accommodate the event. This interruption simulates scenarios such as pedestrian crossings, emergency vehicle access, or other unexpected events affecting traffic flow.

3.1.4 GPIO Pin Configuration

The microcontroller's GPIO pins are configured to control the LEDs representing the traffic lights. Different ports are utilized to manage LEDs for various lanes, allowing for flexible and independent control over each traffic signal.

3.1.5 Traffic Light Cycle Management

The trafficLightCycle function orchestrates the sequence of traffic light signals based on the current cycle number. By activating specific combinations of LEDs, it simulates the orderly progression of traffic signals, ensuring smooth traffic flow.

3.1.6 Interrupt Operation

Upon detection of an interrupt, the interruptOperation function is invoked to handle the event. It activates specific LEDs to signify a temporary change in traffic flow, ensuring the safety and efficiency of the interrupted passage.

3.1.7 Delay Management

Delays are incorporated into the system to control the duration of each traffic light cycle. The duration varies depending on the cycle number, with odd cycles experiencing longer delays to simulate realistic traffic signal timings.

3.2 Hardware and Software Requirements

The hardware and Software requirements are as follows:

Hardware Requirements

- Tiva C- TM4C123GH6PM
- Arduino Uno
- Multi-colour LEDs
- BreadBoard
- 2 Digit- 7 Segment
- IR Sensor
- Jumper Wire

- Connecting wire

  Software Requirements The Software requirements for the project are as follows.

  - Keil uVision v5.
  - Arduino IDE

3.3 System Architecture

3.3.1 TM4C123G architecture

The TM4C123G microcontroller is part of Texas Instruments' Tiva C Series family, which is based on the ARM Cortex-M architecture. Here's a brief overview of the system architecture of the TM4C123G:

ARM Cortex-M4 Core: The heart of the TM4C123G is the ARM Cortex-M4 core, a 32-bit RISC processor. It operates at speeds up to 80 MHz and includes features such as a single-cycle multiply-accumulate (MAC) unit, hardware divide, and barrel shifter. The Cortex-M4 core supports the Thumb-2 instruction set, which provides both high performance and code density. Memory:

The TM4C123G features various types of memory:

- Flash memory: Used for storing program code and typically ranges from 128 KB to 256 KB in size.
- SRAM: Used for storing data and typically ranges from 16 KB to 32 KB in size.
- EEPROM: Some variants may include EEPROM for non-volatile data storage.

Peripherals: The TM4C123G includes a wide range of on-chip peripherals for interfacing with external devices and sensors. These peripherals include: GPIO (General-Purpose Input/Output) pins for digital interfacing. UART, SPI, and I2C interfaces for serial communication. ADC (Analog-to-Digital Converter) and DAC (Digital-to-Analog Converter) modules for analog signal conversion. Timers and PWM (Pulse-Width Modulation) modules for timing and control applications. USB, Ethernet, and CAN interfaces for communication with external devices and networks. Analog comparators, PWM modules, and various other specialized peripherals.

Clocking and Power Management: The TM4C123G includes a sophisticated clocking system for generating internal clock signals and managing power consumption. It typically features multiple oscillators (internal and external) and PLLs (Phase-Locked Loops) for generating various clock frequencies. Power management features such as sleep modes and voltage scaling are available to optimize power consumption based on system requirements.

Interrupts and NVIC: The TM4C123G includes a Nested Vectored Interrupt Controller (NVIC) for handling interrupts efficiently. It supports prioritized interrupt handling, allowing critical interrupts to be serviced first. The NVIC simplifies interrupt management and reduces interrupt latency, critical for real-time systems.

Development Tools: Texas Instruments provides a comprehensive development ecosystem for the TM4C123G, including: Integrated Development Environments (IDEs) such as Code Composer Studio and IAR Embedded Workbench. Software libraries and example code to expedite application development. Debugging tools such as JTAG and Serial Wire Debug (SWD) interfaces for real-time debugging and programming.

Chapter 4

Results and Discussion

4.1 Result

The provided code is designed to control a 4-way traffic light system using a microcontroller (TM4C123) with GPIO pins configured for each traffic light lane and interrupt handling for special operations. Here's a brief overview of the code's functionality:

GPIO pins are initialized for each lane's traffic lights (Port B, Port D, Port E, Port F). Interrupt handling is implemented for GPIO Port F, presumably for handling a special event triggered by a switch connected to PF4.

Traffic light cycles are defined to control the sequence of lights for each lane. A main loop cycles through the traffic light sequences while checking for interrupts.

4.1.1 Discussion

Traffic Light Control: The code effectively controls the traffic light cycles for each lane. It sequences through the specified patterns for the traffic lights, activating the corresponding LEDs for each cycle.

Interrupt Handling: The code includes an interrupt service routine (ISR) for GPIO Port F. When an interrupt is triggered, it activates specific LEDs for 15 seconds, likely indicating a special event or operation. After this duration, it returns to the normal traffic light operation.

Main Loop: The main loop continuously cycles through the traffic light patterns while checking for interrupts. It includes delays between each cycle to simulate the timing of traffic light changes.

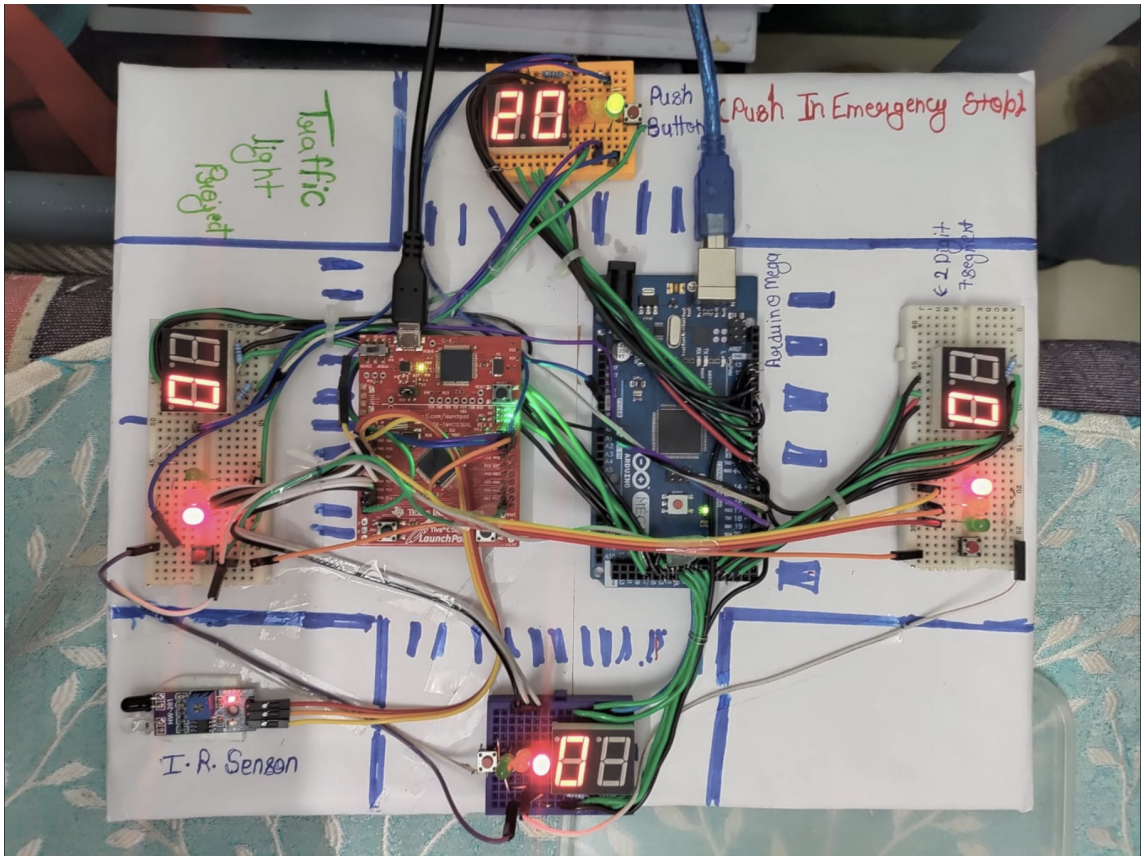

FIGURE 4.1: Real Model

4.2 Unexpected Outcomes

Potential Issue with Interrupt Handling: The interrupt handling mechanism seems straightforward, but potential issues might arise depending on the nature of the interrupt source (e.g., switch bounce, multiple triggering). It's crucial to ensure the ISR is robust enough to handle such scenarios without causing unintended behavior.

Traffic Light Timing: The delays between traffic light cycles are hardcoded based on specific time durations. However, real-world traffic light timing may vary depending on factors like traffic density, pedestrian crossings, and synchronization with adjacent traffic lights. Implementing a more dynamic timing mechanism could enhance realism and efficiency.

4.3 Analysis of Results:

Overall, the code provides a functional implementation of a 4-way traffic light controller using a microcontroller. However, there are areas for improvement, such as refining the interrupt handling mechanism and introducing dynamic timing for traffic light cycles. Additionally, thorough testing is essential to ensure the system operates reliably under various conditions, considering factors like input variations and environmental influences.

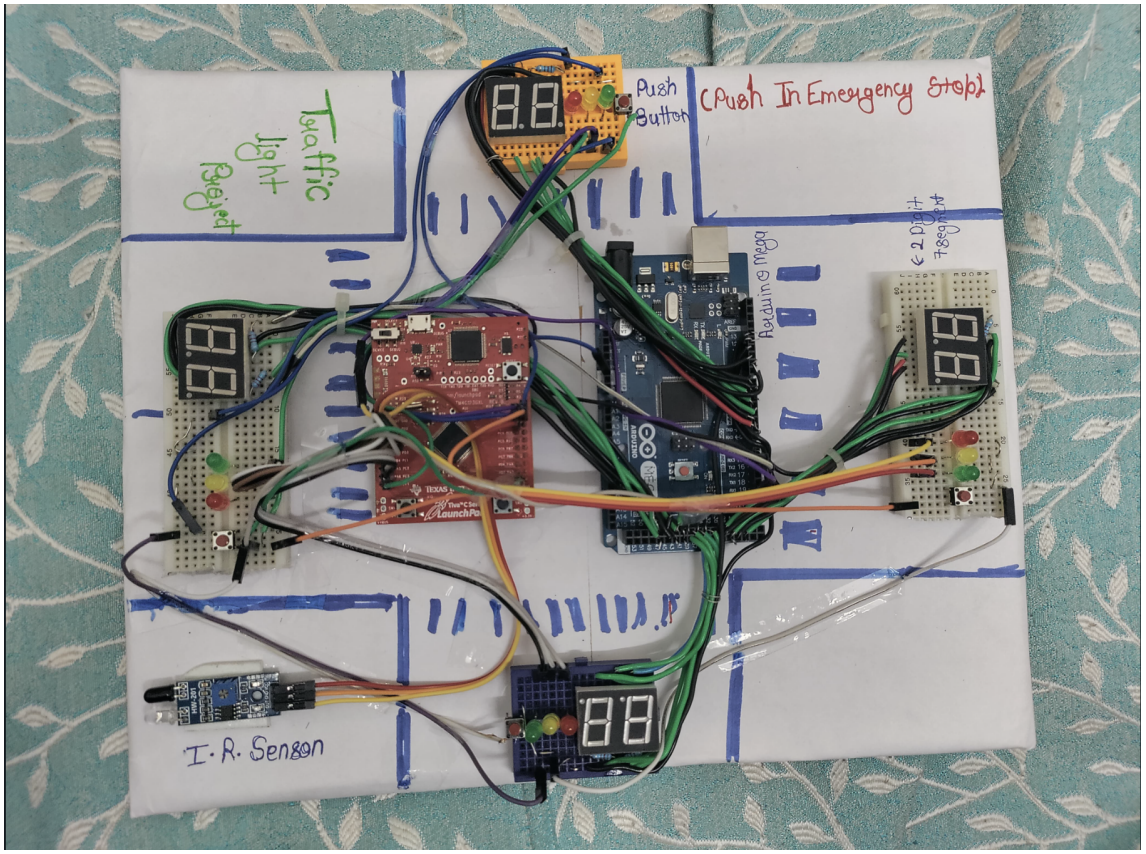

FIGURE 4.2: 4 Lane Traffic light Controller

Chapter 5

Conclusion

5.1 Conclusion

In conclusion, the provided code lays the groundwork for a functional 4-way traffic light controller using a microcontroller. It effectively manages traffic light cycles and includes interrupt handling for special events. However, further refinement is needed, particularly in interrupt handling robustness and dynamic timing algorithms. Thorough testing and validation will be crucial to ensure reliability under various conditions. Future work could focus on enhancing interrupt handling, implementing dynamic timing algorithms, integrating user interfaces for monitoring and configuration, optimizing energy efficiency, and ensuring scalability for expansion. Overall, with continued development, the traffic light controller project holds promise for improving traffic management and urban mobility.

5.2 Key Takeaways

Functionality: The code successfully implements the basic functionality of a traffic light controller, managing the sequencing of lights for each lane and handling interrupts for special operations.

Interrupt Handling: While the interrupt handling mechanism is implemented, further refinement may be necessary to ensure robustness and reliability, especially in scenarios involving switch bounce or multiple triggering events.

Traffic Light Timing: The hardcoded delays between traffic light cycles provide a simple timing mechanism. However, future work could involve implementing dynamic timing algorithms to optimize traffic flow based on real-time conditions.

Testing and Validation: Thorough testing is essential to validate the functionality of the system under various scenarios, including different traffic patterns, environmental conditions, and interrupt scenarios.

5.3 Future Work

To enhance the 4-way traffic light controller project, several avenues for future work can be explored:

- **Enhanced Interrupt Handling:** Implement debounce mechanisms to mitigate switch bounce issues. Consider multiple interrupt sources and prioritize handling based on predefined criteria.
- **Dynamic Timing Algorithms:** Develop algorithms to dynamically adjust traffic light timings based on real-time traffic conditions, pedestrian crossings, and emergency vehicle prioritization.
- **User Interface and Monitoring:** Integrate a user interface for configuration and monitoring of traffic light settings and operational status. Implement remote monitoring capabilities, allowing authorities to oversee and adjust traffic light operations as needed.
- **Traffic Simulation and Optimization:** Introduce traffic simulation models to test and optimize the traffic light controller's performance under various traffic scenarios. Utilize machine learning techniques to predict traffic patterns and optimize traffic light timings accordingly.
- **Energy Efficiency:** Explore energy-efficient designs to minimize power consumption without compromising the effectiveness of the traffic light system. Implement sleep modes or power management strategies to conserve energy during periods of low traffic activity.
- **Scalability and Expansion:** Design the system with scalability in mind, allowing for easy expansion to accommodate additional lanes or intersections. Consider

compatibility with smart city infrastructure for seamless integration into larger urban traffic management systems.

By addressing these areas in future iterations, the 4-way traffic light controller project can evolve into a more sophisticated and adaptive system capable of improving traffic efficiency, safety, and overall urban mobility.

Appendix A

Code

A.1 Delay Code

```
#include "TM4C123.h" // Include the header file for TM4C123 microcont

volatile int interruptFlag = 0; // Declare a volatile variable to inc

void delay(int a) { // Function to introduce delay
    int i = 10000; // Initialize loop counter
    for (i = 10000; i > 0; i--) { // Loop to introduce delay
        while (a > 0) { // Nested loop for delay
            a--; // Decrement delay counter
        }
    }
}
```

Figure A.1: Delay Code

A.2 Main Function Code

```
int main(void) { // Main function
    initializeGPIO(); // Initialize GPIO pins

    while (1) { // Main loop
        // Loop through traffic light cycles
        for (int cycle = 1; cycle <= 8; cycle++) { // Loop from cycle 1 to cycle 8
            trafficLightCycle(cycle); // Set traffic light cycle

            // Check if interrupt occurred
            if (interruptFlag) { // If an interrupt occurred
                interruptOperation(); // Perform interrupt operation
                interruptFlag = 0; // Reset interrupt flag
            }
            // Delay for approximately 10 seconds if cycle is odd, and 2 seconds if even
            if (cycle % 2 == 1) {
                delay(89419355); // Delay for 25 seconds
            } else {
                delay(17883870); // Delay for 5 seconds
            }
        }
    }
}
```

FIGURE A.2: Main Function Code

A.3 Traffic Light logic Code

```
void trafficLightCycle(int cycle) { // Function to control the traffic light cycle
    // Deactivate all lanes initially
    GPIOF->DATA = 0;
    GPIOB->DATA = 0;
    GPIOE->DATA = 0;
    GPIOD->DATA = 0;

    // Activate specific LEDs based on the cycle
    switch (cycle) {
        case 1: // First cycle
            // Lane 1: Green, Lane 2: Red, Lane 3: Red, Lane 4: Red
            GPIOF->DATA = 0x08; // Activate only PF1 (Green for Lane 1)
            GPIOB->DATA = 0x02; // Activate only PB2 (Red for Lane 1)
            GPIOE->DATA = 0x02; // Activate only PE2 (Red for Lane 2)
      GPIOD->DATA = 0x02; // Activate only PD2 (Red for Lane 3)
            break;
        case 2: // Second cycle
            // Lane 1: Yellow, Lane 2: Green, Lane 3: Red, Lane 4: Red
            GPIOF->DATA = 0x08; // Activate only PF2 (Yellow for Lane 1)
            GPIOB->DATA = 0x04; // Activate only PB3 (Green for Lane 2)
            GPIOE->DATA = 0x02; // Activate only PE1 (Red for Lane 3)
      GPIOD->DATA = 0x02; // Activate only PD2 (Red for Lane 4)
            break;
```

FIGURE A.3: Traffic Light logic Code

A.4 Switch Condition Code

```
        case 3: // Third cycle
            // Lane 1: Red, Lane 2: Yellow, Lane 3: Green, Lane 4: Red
            GPIOF->DATA = 0x02; // Activate only PF3 (Red for Lane 1)
            GPIOB->DATA = 0x08; // Activate only PB1 (Yellow for Lane 2)
            GPIOE->DATA = 0x02; // Activate only PE2 (Green for Lane 3)
      GPIOD->DATA = 0x02; // Activate only PD2 (Red for Lane 4)
            break;
        case 4: // Fourth cycle
            // Lane 1: Red, Lane 2: Red, Lane 3: Yellow, Lane 4: Green
            GPIOF->DATA = 0x02; // Activate only PF1 (Red for Lane 1)
            GPIOB->DATA = 0x08; // Activate only PB2 (Red for Lane 2)
            GPIOE->DATA = 0x04; // Activate only PE3 (Yellow for Lane 3)
      GPIOD->DATA = 0x02; // Activate only PD2 (Green for Lane 4)
            break;
        case 5: // Fifth cycle
            // Lane 1: Red, Lane 2: Red, Lane 3: Green, Lane 4: Yellow
            GPIOF->DATA = 0x02; // Activate only PF2 (Red for Lane 1)
            GPIOB->DATA = 0x02; // Activate only PB3 (Red for Lane 2)
            GPIOE->DATA = 0x08; // Activate only PE1 (Green for Lane 3)
      GPIOD->DATA = 0x02; // Activate only PD2 (Yellow for Lane 4)
            break;
```

```
        case 6: // Sixth cycle
            // Lane 1: Red, Lane 2: Green, Lane 3: Red, Lane 4: Yellow
            GPIOF->DATA = 0x02; // Activate only PF3 (Red for Lane 1)
            GPIOB->DATA = 0x02; // Activate only PB1 (Green for Lane 2)
            GPIOE->DATA = 0x08; // Activate only PE2 (Red for Lane 3)
      GPIOD->DATA = 0x04; // Activate only PD2 (Yellow for Lane 4)
            break;
        case 7: // Seventh cycle
            // Lane 1: Red, Lane 2: Red, Lane 3: Red, Lane 4: Green
            GPIOF->DATA = 0x02; // Activate only PF3 (Red for Lane 1)
            GPIOB->DATA = 0x02; // Activate only PB1 (Red for Lane 2)
            GPIOE->DATA = 0x02; // Activate only PE2 (Red for Lane 3)
      GPIOD->DATA = 0x08; // Activate only PD2 (Green for Lane 4)
            break;
        case 8: // Eighth cycle
            // Lane 1: Red, Lane 2: Red, Lane 3: Green, Lane 4: Red
            GPIOF->DATA = 0x04; // Activate only PF3 (Red for Lane 1)
            GPIOB->DATA = 0x02; // Activate only PB1 (Red for Lane 2)
            GPIOE->DATA = 0x02; // Activate only PE2 (Green for Lane 3)
      GPIOD->DATA = 0x08; // Activate only PD2 (Red for Lane 4)
            break;
        default: // Default case
            break;
    }
}
```

FIGURE A.4: Switch Condition Code

A.5 Interrupt Handler Code

```
void GPIOF_Handler(void) { // Interrupt handler for GPIOF
  GPIOF->ICR |= (1 << 4); // Clear interrupt flag for PF4
  GPIOF->DATA = 0x02; // Turn on the red LED on Port F
  GPIOB->DATA = 0x02; // Turn on the red LED on Port B
  GPIOE->DATA = 0x02; // Turn on the red LED on Port E
  GPIOD->DATA = 0x02; // Turn on the red LED on Port D

    GPIOB->DATA |= (1 <<5 ); // Turn on PA2
    delay(5000000); // Delay for approximately 1 second
    GPIOB->DATA &= ~(1 << 5); // Turn off PA2
}
```

FIGURE A.5: Interrupt Handler Code