(HTI)

Electrical Engineering & Computers Department

Final Project Report

(Traffic Control)

Microcontroller

(EEC 214)

Submitted by:

ابراهيم جمال عبدالموجود علي – 20211073
مصطفي احمد محمد عبدالسلام – 20210746
طه احمد محمد سيد – 20210753
اسلام خالد حسن – 20200845
عبدالرحمن معتمد محمد خيري – 20200489
عبدالرحمن اسامة محمد ابومهدي – 20200481

Submitted to:

Dr: Mohamed Torad

Nov 2024

Abstract

This project focuses on the development of a versatile digital clock using the ATmega32 microcontroller, an alarm function, and a user-friendly interface for time and date adjustments. The primary objectives include designing a reliable timekeeping system, implementing an intuitive user interface, and incorporating an alarm feature to enhance the clock's functionality.

The clock utilizes the ATmega32 microcontroller's internal timer for precise timekeeping. The real-time clock functionality ensures accurate time tracking, while an added alarm feature enhances user experience by allowing them to set personalized wake-up times or reminders. The user interface is designed for ease of interaction, employing push buttons to enable users to effortlessly adjust both time and date settings.

The implementation encompasses clock initialization, timekeeping logic, and display algorithms, detailing how the microcontroller manages and updates time data. The user interface facilitates seamless interaction through keypad, employing debouncing mechanisms to ensure reliable input. Additionally, the display logic is carefully designed to convert time and date information into a format suitable for the LCD.

Testing procedures have been rigorously applied to validate the accuracy and reliability of the digital clock, including functional tests for timekeeping, date adjustments, and alarm triggering. Challenges encountered during the development process have been addressed to ensure a robust and user-friendly final product.

In conclusion, this project successfully integrates a digital clock with advanced features, providing users with not only a reliable timekeeping device but also an alarm function and an intuitive interface for personalized time and date adjustments. Future enhancements could explore additional features such as different alarm tones, or connectivity options. The versatility of this digital clock makes it a valuable and adaptable solution for various applications, from personal use to professional settings.

Acknowledgment

At first, Thanks to **ALLAH** the most merciful the most gracious, for this moment has come and this work has been accomplished. Thanks to the **Higher Technological Institute of 10th Ramdan** for preparing me to be a successful Engineer and lifting me up to achieve this training in an environment that is full of encouragement and motivation.

Deepest gratitude is to be delivered to **Dr.Mohammed Torad**, my role model in engineering. He understood the nature of my thoughts and guided me step by step till this work was brought to light. Endless trust in my potential guided me till the end. Thank you.

Not to forget everyone who helped me, prayed for me, wished me luck, or pushed me forwards and bored me a lot to help this work come to life. Thanks to my colleagues, friends, laborers, technicians, and everyone else for everything they did.

Last but never forgotten, Thanks to my dear family, for being supportive and always by my side. No words can express my deepest and sincere gratitude towards the love and care you have granted me in my hardest times. May **ALLAH** fill your hearts with happiness when we share this success together.

Table of Contents

Contents

Table of Figures

Chapter 1

Introduction

The aim of this project is to design and implement a feature-rich digital clock utilizing the ATmega32 microcontroller. The clock incorporates an alarm function for enhanced utility, an LCD display for clear information presentation, and a keypad interface to facilitate user input.

This report describes that project in 7 chapters:

1. Introduction
2. Components Used
3. Circuit simulation
4. Implementation
5. Code
6. Conclusion & Future Enhancements
7. References

chapter 2

components

- AVR_USPASP
- ATmega32 Microcontroller
- LCD Display (16x2)
- LCD header pin
- Push Button
- Crystal Oscillator (32768 HZ)
- Resistors and Capacitors
- Power Supply
- Male to female jumpers
- Seven Segment
- BCD 7448

AVR_USPASP

Figure 1 AVR_USPASP

ATmega32 Microcontroller

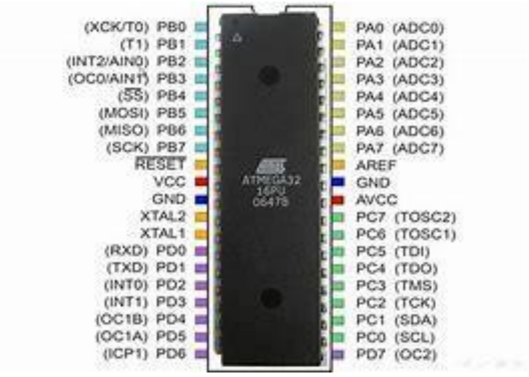


Figure 2 ATmega32 Microcontroller

LCD Display

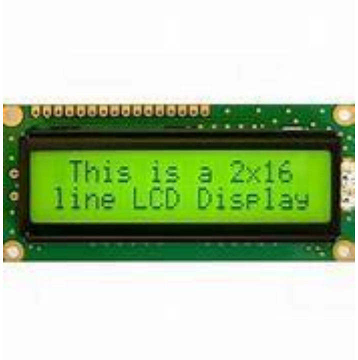


Figure 4 LCD Display

Seven Segment

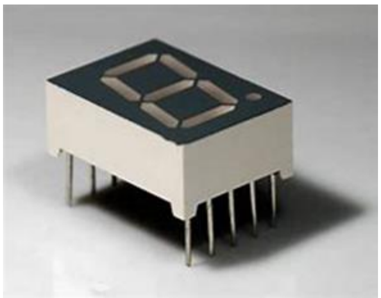


Figure 3 Seven Segment

LCD Header Pins

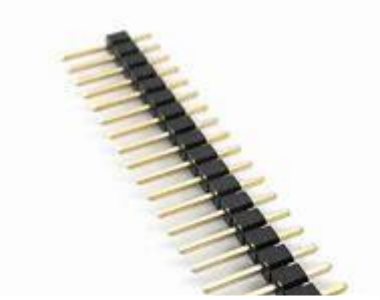


Figure 6 LCD Header Pins

Push Button

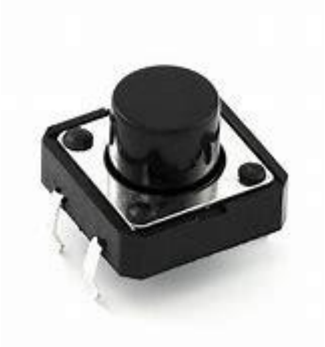


Figure 5 Push Button

Crystal Oscillator (32768 HZ)

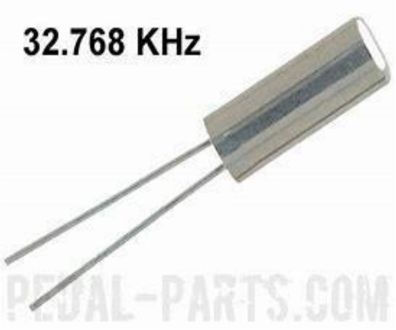


Figure 6 Crystal Oscillator (32768 HZ)

Resistors and Capacitors

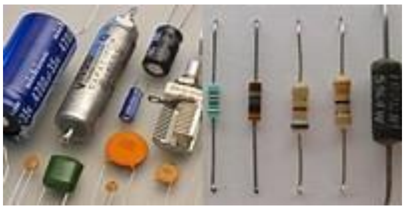


Figure 7 Resistors and Capacitor

Chapter 3

Circuit simulation

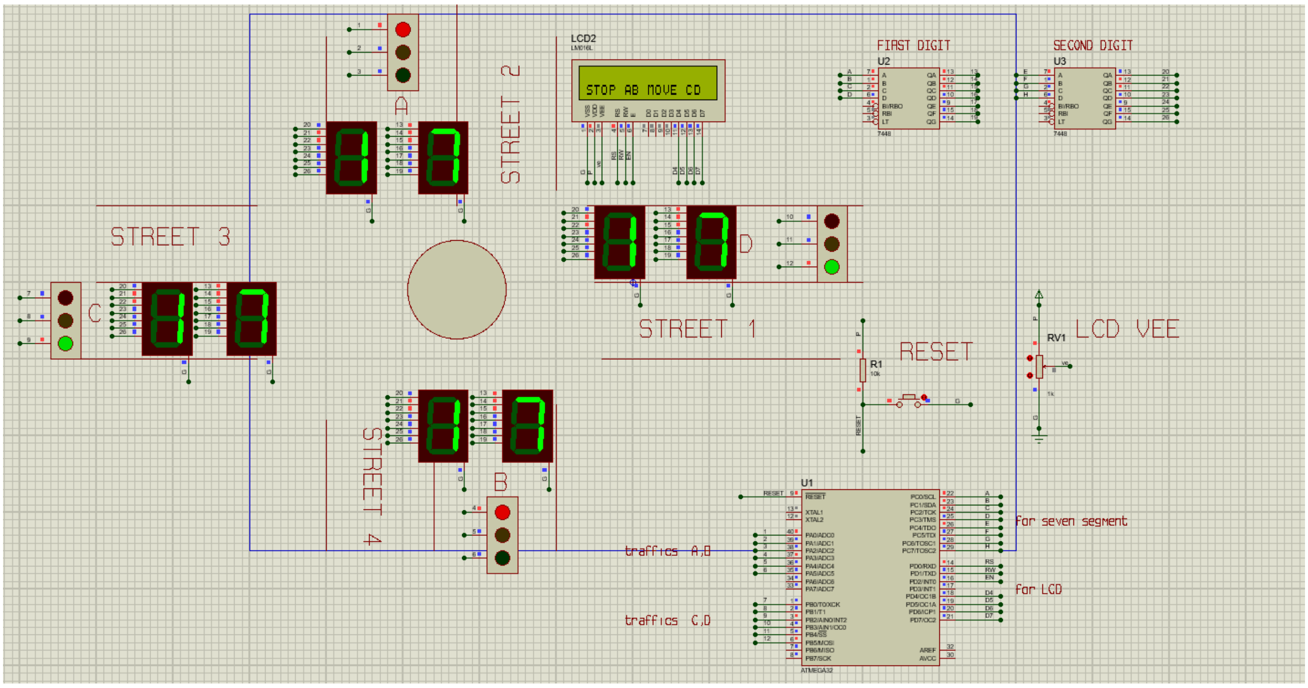


Figure 7 Circuit Simulation

- This simulation is done on a proteus program.
- It includes all the components of the circuit to be done as well as hardware.
  - On the top middle is the LCD 2*16 which is data output for the user
  - On the right is the push button which receive the input data from user
  - On the top right is the BCD 7448
  - On the right the Crystal Oscillator (32768 HZ)
  - On the bottom right we found the ATmega32.
  - On the left we found a traffic light control modules

- We used the Atmel studio to create the files that needed to be burned on that controller from our code.

❖ The Pin Configurations of the controller

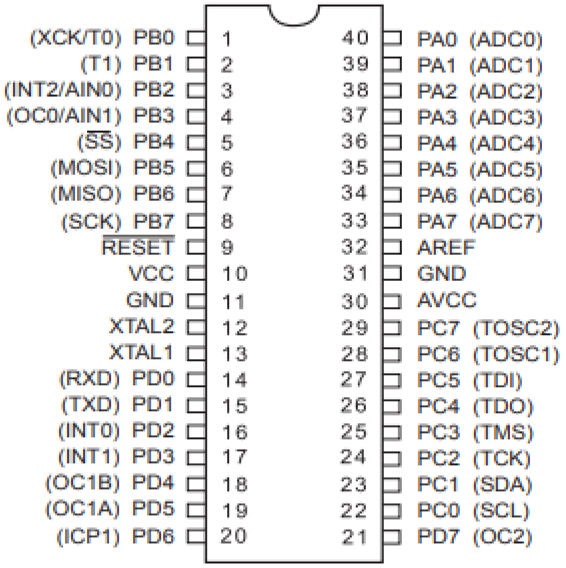


Figure 8 The Pin Configurations of the Controller

Chapter 4

Implementation

Traffic Light Control:

The code controls the traffic lights using **PORTA** and **PORTB** for 4 lights:

- **state == 0**: Lights 1 and 2 (AB) are green, Lights 3 and 4 (CD) are red.
- **state == 1**: All lights are yellow (transition phase).
- **state == 2**: Lights 3 and 4 (CD) are green, Lights 1 and 2 (AB) are red.
- **state == 3**: All lights are yellow again.

LCD Display:

The **LCD** displays messages indicating the traffic light status, such as "MOVE AB STOP CD" or "DON'T MOVE AB". It is updated based on the current state.

7-Segment Display:

The **7-segment display** shows the countdown for the remaining time in each phase of the traffic light, such as the countdown for green or yellow lights.

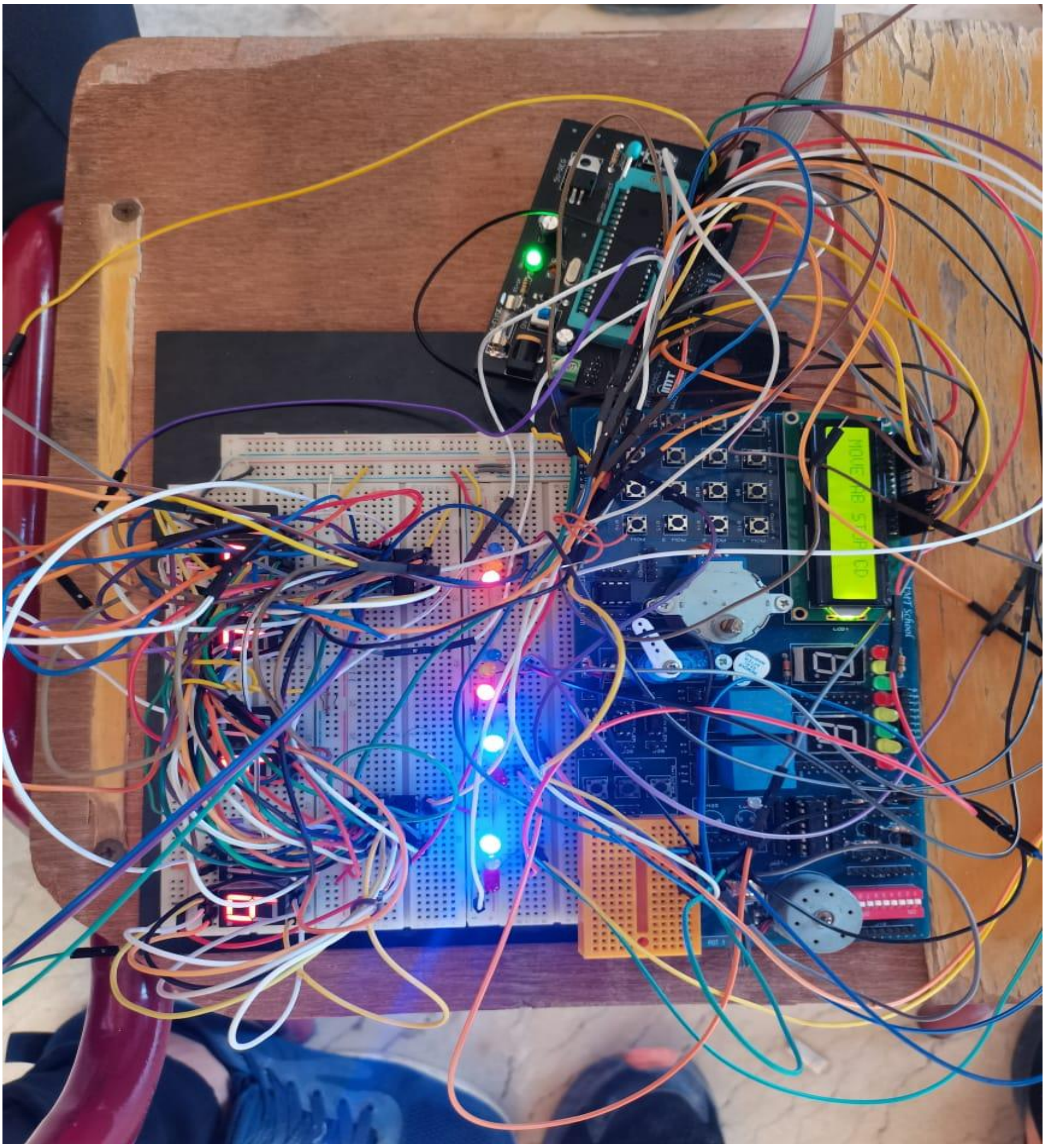
Timing and Delay:

_delay_ms() and **_delay_us()** are used to introduce delays between traffic light changes and the countdown. Everything updates every second.

State Transitions:

The program cycles through the states (Green, Yellow, Red) in a continuous loop while updating the countdown on both the 7-segment and LCD displays to indicate the current traffic light phase.

Hardware:



Chapter 5

Code

```
#include <avr/io.h>
#include <util/delay.h>

#define F_CPU 8000000UL // Define clock speed

// LCD control pins connected to PORTD
#define RS PD0
#define RW PD1
#define E  PD2
#define D4 PD4
#define D5 PD5
#define D6 PD6
#define D7 PD7

// Function to send command to LCD
void lcd_command(uint8_t command) {
        PORTD = (PORTD & 0x0F) | (command & 0xF0);  // Send the high nibble
        PORTD &= ~(1 << RS);                       // RS = 0 for command
        PORTD &= ~(1 << RW);                        // RW = 0 for write
        PORTD |= (1 << E);                        // Enable high
        _delay_us(1);
        PORTD &= ~(1 << E);                         // Enable low
        _delay_us(200);

        PORTD = (PORTD & 0x0F) | (command << 4);    // Send the low nibble
        PORTD |= (1 << E);                        // Enable high
        _delay_us(1);
        PORTD &= ~(1 << E);                         // Enable low
        _delay_us(200);
}

// Function to send data to LCD
void lcd_data(uint8_t data) {
        PORTD = (PORTD & 0x0F) | (data & 0xF0);     // Send the high nibble
        PORTD |= (1 << RS);                       // RS = 1 for data
        PORTD &= ~(1 << RW);                        // RW = 0 for write
        PORTD |= (1 << E);                        // Enable high
        _delay_us(1);
        PORTD &= ~(1 << E);                         // Enable low
        _delay_us(200);

        PORTD = (PORTD & 0x0F) | (data << 4);       // Send the low nibble
        PORTD |= (1 << E);                        // Enable high
        _delay_us(1);
        PORTD &= ~(1 << E);                         // Enable low
        _delay_us(200);
}

// Function to initialize LCD
void lcd_init() {
        // Set PORTD pins as output
        DDRD |= (1 << RS) | (1 << RW) | (1 << E) | (1 << D4) | (1 << D5) | (1 << D6) | (1 << D7);

        // Initialize LCD in 4-bit mode
        _delay_ms(15);        // Wait for more than 15 ms after Vcc is applied
        lcd_command(0x33);    // Initialize LCD in 4-bit mode
        lcd_command(0x32);    // Initialize LCD in 4-bit mode
        lcd_command(0x28);    // 4-bit mode, 2-line, 5x7 matrix
        lcd_command(0x0C);    // Display ON, Cursor OFF
        lcd_command(0x06);    // Increment cursor (shift to right)
        lcd_command(0x01);    // Clear screen
        _delay_ms(2);         // Wait for the command to process
}
// Function to set the cursor position on LCD
void lcd_gotoxy(uint8_t x, uint8_t y) {
        uint8_t pos;
        if (y == 0) {
            pos = 0x80 + x;  // First row
            } else {
            pos = 0xC0 + x;  // Second row
        }
        lcd_command(pos);   // Send position command
}

// Function to display string on LCD
void lcd_puts(const char *str) {
        while (*str) {
            lcd_data(*str);   // Display each character
            str++;
        }
}

// Function to display a number on two 7-segment displays
void display_bcd_number(uint8_t number) {
        uint8_t tens = number / 10;  // Extract tens place
        uint8_t units = number % 10; // Extract units place

        // Display tens place on first seven-segment
        PORTC = (tens << 4) | (units & 0x0F); // High nibble for tens, low nibble for units
        _delay_ms(5);
}

// Function to control traffic lights and display text on LCD
void control_traffic_lights(uint8_t state) {
        if (state == 0) {
            // Traffic Lights 1 & 2: Green; Traffic Lights 3 & 4: Red
            PORTA = (1 << PA2) | (1 << PA5); // Green for Traffic Lights 1 & 2
            PORTB = (1 << PB0) | (1 << PB3); // Red for Traffic Lights 3 & 4
            lcd_gotoxy(0, 0); // First row, first column
            lcd_puts("MOVE AB STOP CD   "); // Show "MOVE AB STOP CD"
            } else if (state == 1) {
            // All Yellow Lights for transition
            PORTA = (1 << PA1) | (1 << PA4); // Yellow for Traffic Lights 1 & 2
            PORTB = (1 << PB1) | (1 << PB4); // Yellow for Traffic Lights 3 & 4
            lcd_gotoxy(0, 0);
            lcd_puts("DON'T MOVE AB"); // Show "don't move FOR YELLOW"
            lcd_gotoxy(0, 1);
            lcd_puts("READY CD ");
            } else if (state == 2) {
            // Traffic Lights 3 & 4: Green; Traffic Lights 1 & 2: Red
            PORTA = (1 << PA0) | (1 << PA3); // Red for Traffic Lights 1 & 2
            PORTB = (1 << PB2) | (1 << PB5); // Green for Traffic Lights 3 & 4
            lcd_gotoxy(0, 1);
            lcd_puts("STOP AB MOVE CD   "); // Show "STOP AB MOVE CD"
        }
        else if (state == 3) {
            // All Yellow Lights for transition
            PORTA = (1 << PA1) | (1 << PA4); // Yellow for Traffic Lights 1 & 2
            PORTB = (1 << PB1) | (1 << PB4); // Yellow for Traffic Lights 3 & 4
        lcd_gotoxy(0, 0);
            lcd_puts("READY AB"); // FOR YELLOW"
            lcd_gotoxy(0, 1);
            lcd_puts("DON'T MOVE CD ");
        }
}

int main() {
        // Configure ports
        DDRA = 0xFF; // Set PORTA as output (Traffic Lights 1 & 2)
        DDRB = 0xFF; // Set PORTB as output (Traffic Lights 3 & 4)
        DDRC = 0xFF; // Set PORTC as output for BCD signals
        DDRD = 0xFF; // Set PORTD as output for LCD

        // Initialize ports
        PORTA = 0x00; // All lights off
        PORTB = 0x00; // All lights off
        PORTC = 0x00; // Clear BCD output
        PORTD = 0x00; // Clear LCD output

        // Initialize LCD
        lcd_init(); // Initialize LCD in 4-bit mode

        uint8_t state = 0; // Initial state (Traffic Lights 1 & 2 Green, 3 & 4 Red)

        while (1) {
            // Green/Red Phase: Countdown from 20 to 0
            for (uint8_t i = 20; i > 0; i--) {
                control_traffic_lights(state); // Set traffic light states
                display_bcd_number(i);      // Display countdown on 7-segment
                _delay_ms(1000);            // 1-second delay
            }
            lcd_command(0x01); // Clear LCD screen after display
            _delay_ms(1000);   // Delay after clearing

            // Yellow Phase: Run for 5 seconds
            for (uint8_t i = 4; i > 0; i--) {
                control_traffic_lights(1); // Yellow lights
                display_bcd_number(i);    // Display countdown on 7-segment
                _delay_ms(1000);          // 1-second delay
               }
```

```
            lcd_command(0x01); // Clear LCD screen after display
            _delay_ms(1000);   // Delay after clearing
            // Green/Red Phase: Countdown from 20 to 0
            for (uint8_t i = 20; i > 0; i--) {
                control_traffic_lights(2); // Set traffic light states
                display_bcd_number(i);        // Display countdown on 7-segment
                _delay_ms(1000);              // 1-second delay
            }
            lcd_command(0x01); // Clear LCD screen after display
            _delay_ms(1000);   // Delay after clearing
            // Yellow Phase: Run for 5 seconds
            for (uint8_t i = 4; i > 0; i--) {
                control_traffic_lights(3); // Yellow lights
                display_bcd_number(i);     // Display countdown on 7-segment
                _delay_ms(1000);           // 1-second delay
            }
            lcd_command(0x01); // Clear LCD screen after display
            _delay_ms(1000);   // Delay after clearing
        }
        return 0;
    }
```

asm

```
; تعريف الثوابت
.equ F_CPU = 8000000
.equ RS = PD0
.equ RW = PD1
.equ E  = PD2
.equ D4 = PD4
.equ D5 = PD5
.equ D6 = PD6
.equ D7 = PD7

; تهيئة المنافذ
.org 0x00
rjmp main

main:
  ; تهيئة PORTA و PORTB و PORTC و PORTD كخرج
  ldi r16, 0xFF
  out DDRA, r16    ; PORTA كخرج
  out DDRB, r16    ; PORTB كخرج
  out DDRC, r16    ; PORTC كخرج
  out DDRD, r16    ; PORTD كخرج

  ; تهيئة كل المنافذ إلى صفر
  ldi r16, 0x00
  out PORTA, r16
```

```
    out PORTB, r16
    out PORTC, r16
    out PORTD, r16

    ; تهيئة LCD
    call lcd_init

    ; حالة المرور الابتدائية
    ldi r20, 0      ; الحالة الابتدائية

loop:
    ; العد التنازلي من 20 إلى 0
    ldi r21, 20     ; بدء العد التنازلي
countdown_green:
    call control_traffic_lights
    call display_bcd_number
    dec r21
    breq yellow_phase
    call delay_1s
    rjmp countdown_green

yellow_phase:
    ; تشغيل الأضواء الصفراء
    ldi r21, 4
countdown_yellow:
    ldi r20, 1      ; حالة الأضواء الصفراء
    call control_traffic_lights
    call display_bcd_number
    dec r21
    breq green_phase
    call delay_1s
    rjmp countdown_yellow

green_phase:
    ; العد التنازلي من 20 إلى 0
    ldi r21, 20
countdown_red:
    ldi r20, 2      ; حالة الأضواء الحمراء
    call control_traffic_lights
    call display_bcd_number
    dec r21
    breq yellow_phase2
    call delay_1s
    rjmp countdown_red

yellow_phase2:
    ldi r21, 4
countdown_yellow2:
    ldi r20, 3      ; حالة الأضواء الصفراء
    call control_traffic_lights
    call display_bcd_number
    dec r21
    breq loop
    call delay_1s
    rjmp countdown_yellow2

; وظيفة تهيئة وحدة LCD
lcd_init:
    ; تأخير لتهيئة الطاقة
    ldi r18, 15
    call delay_ms

    ; إرسال الأوامر اللازمة
    ldi r16, 0x33
    call lcd_command
    ldi r16, 0x32
    call lcd_command
    ldi r16, 0x28
    call lcd_command
    ldi r16, 0x0C
    call lcd_command
    ldi r16, 0x06
    call lcd_command
    ldi r16, 0x01
    call lcd_command
    ret

; وظيفة إرسال الأوامر إلى LCD
lcd_command:
    ; إرسال الجزء العالي
    out PORTD, r16
    cbi PORTD, RS     ; RS = 0
    cbi PORTD, RW     ; RW = 0
    sbi PORTD, E      ; E = 1
    call delay_us
    cbi PORTD, E      ; E = 0
    call delay_us

    ; إرسال الجزء المنخفض
    lsr r16
    lsr r16
    out PORTD, r16
    sbi PORTD, E      ; E = 1
    call delay_us
    cbi PORTD, E      ; E = 0
    call delay_us
    ret

; وظيفة لعرض الأرقام على شاشات سبع قطع
display_bcd_number:
    ; استخراج الآحاد والعشرات
    mov r22, r21
    ldi r23, 10
    div r22, r23      ; r22 = الآحاد
    mov r24, r22      ; r24 = العشرات
    ; عرض الأرقام على PORTC
    ldi r16, 0
    or r16, r24
    or r16, r21
    out PORTC, r16
    ret

; وظيفة التحكم في إشارات المرور
control_traffic_lights:
    ; تحديد حالة إشارات المرور بناءً على r20
    cpi r20, 0
    breq state_green_red
    cpi r20, 1
    breq state_yellow
    cpi r20, 2
    breq state_red
    cpi r20, 3
    breq state_yellow

state_green_red:
    ; الأضواء الخضراء
    sbi PORTA, PA2
    sbi PORTA, PA5
    cbi PORTB, PB0
    cbi PORTB, PB3
    ; عرض النص على LCD
    ldi r16, "MOVE AB STOP CD"
    call lcd_puts
    ret

state_yellow:
    ; الأضواء الصفراء
    sbi PORTA, PA1
    sbi PORTA, PA4
    sbi PORTB, PB1
    sbi PORTB, PB4
    ; عرض النص على LCD
    ldi r16, "DON'T MOVE AB"
    call lcd_puts
    ret

state_red:
    ; الأضواء الحمراء
    cbi PORTA, PA0
    cbi PORTA, PA3
    sbi PORTB, PB2
    sbi PORTB, PB5
    ; عرض النص على LCD
    ldi r16, "STOP AB MOVE CD"
    call lcd_puts
    ret

; وظيفة تأخير زمني بالمللي ثانية
delay_ms:
    ; تنفيذ تأخير زمني (تقديري)
    ret

; وظيفة تأخير زمني بالميكرو ثانية
delay_us:
    ; تنفيذ تأخير زمني (تقديري)
    ret

; وظيفة لعرض النص على LCD
lcd_puts:
    ; عرض كل حرف في النص على LCD
    ret
```

Chapter 6

Conclusion & Future Enhancements

Conclusion

This project successfully implements a traffic light control system with an LCD display and 7-segment countdown. The integration of the LCD allows for clear display of traffic light status, while the 7-segment display provides a visual countdown. The program effectively cycles through green, yellow, and red light phases, ensuring proper traffic management. Overall, it provides a functional and intuitive system for traffic light control.

Future Enhancements

Future improvements could include:

- Adding a real-time clock for scheduling light transitions.
- Implementing more advanced traffic control algorithms, such as adaptive traffic lights based on traffic flow.
- Integrating wireless communication for remote monitoring and control of the system.
- Adding more LED displays to show real-time traffic data or accidents on the road.

References

- Atmega32A-DataSheet-Complete-DS40002072A.pdf
- The AVR Microcontroller and Embedded Systems: Using Assembly and C Muhammad Ali Mazidi | Sarmad Naimi Sepehr Naim
- Interfacing LCD with AVR Microcontrollers
- Seven Segment Display with AVR Microcontroller
- Traffic Light Control System Using AVR