Hardware Assignment-1

EE24BTECH11003 - Akshara Sarma Chennubhatla

March 2025

Assignment:
To make a digital clock
which displays hours,
minutes and seconds
with modes for timer and stopwatch

Bachelor of Technology

Department of Electrical Engineering

Contents

1 Introduction

A digital clock is a very useful device used to display time in 24-hour format. This aim of the project is to make a digital clock using Arduino.

2 Components used

The following components were used:

- Arduino Uno
- Breadboard
- Six 7-segment displays
- A 7447 BCD to 7-segment decoders
- Push buttons for setting time
- Resistors(220Ω) and wiring
- Power source

3 Circuit Design

The 7-segment displays are connected to each other and then one of them is connected to the pins of the 7447 according to the table below

7447	$\bar{a}$	$\bar{b}$	$\bar{c}$	$\bar{d}$	$\bar{e}$	$\bar{f}$	$\bar{g}$
Display	a	b	c	d	e	f	g

The remaining pins of 7447 which are to be connected to the arduino are as follows

7447	D	C	B	A
Arduino	5	4	3	2

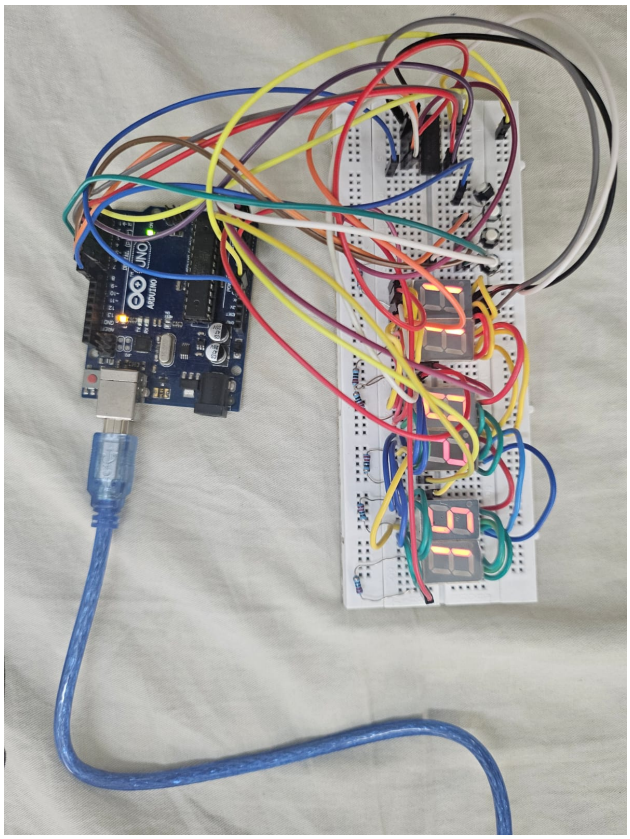
$5v$ pin of Arduino is connected to v_{cc} of 7447 while their grounds are connected to each other.

The COM pins of the 7-seg displays are connected to 220Ω which are connected to analogue pins on the Arduino.

There are also 4 push buttons which are connected to 6,7,8 and 9 pins in the Arduino . They have the following uses

- The first is used to adjust the hours of the clock by incrementing till 23 and then reset to zero.
- The second is used the adjust the minutes by incrementing till 59 then reset to zero.
- The third switches the clock between showing the time and being used as a stopwatch.
- The fourth is used to stop time or the stop watch.

The following is the picture of the fully functioning circuit



4 How does the clock work with just 1 7447 Decoder?

Since all the COMs of the seven segment displays are connected to the arduino pins, the logic used is that only one pin is given high voltage at a time when all the others are given low voltage. Through this the data sent to the 7447 decoder is only sent to one seven segment display at a time. Now we cycle through all the seven segment displays like this with a very small delay of around 10 milliseconds. We send the data to the individual seven segment displays such that time properly works. Due to the very small delay, the human eye cannot see the flickering of the displays which makes it look like a regular working clock which displays numbers continuously.

5 Code

The following is the code implemented

```
#define F_CPU 16000000UL
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>

#define BCD_PORT PORTD
#define BCD_DDR DDRD
#define BCD_MASK 0b00111100  // PD2 to PD5

#define COMMON_PORT PORTC
#define COMMON_DDR DDRC

#define MODE_BUTTON PB0 // Switch between Clock, Timer, and Stopwatch
#define STOPWATCH_BUTTON PB1 // Start/Stop Stopwatch and Timer

volatile int seconds = 0, minutes = 0, hours = 15;
volatile int timer_seconds = 0, timer_minutes = 0, timer_hours = 0;
volatile int stopwatch_seconds = 0, stopwatch_minutes = 0, stopwatch_hours = 0;
volatile int mode = 0; // 0 = Clock, 1 = Timer, 2 = Stopwatch
volatile int stopwatch_running = 0; // 1 = Running, 0 = Stopped

void setup() {
    // Set BCD display pins (PD2-PD5) as output
```

```
    BCD_DDR |= BCD_MASK;
    BCD_PORT &= ~BCD_MASK;

    // Set digit selector pins (PORTC) as output
    COMMON_DDR = 0xFF;
    COMMON_PORT = 0x00;

    // Enable pull-up resistors for buttons
    PORTD |= (1 << PD6) | (1 << PD7);
    PORTB |= (1 << MODE_BUTTON) | (1 << STOPWATCH_BUTTON);

    // Timer1 Setup: CTC Mode, 1-second interval
    TCCR1B |= (1 << WGM12) | (1 << CS12) | (1 << CS10);
    OCR1A = 15625; // 1-second interrupt
    TIMSK1 |= (1 << OCIE1A);

    // Debug LED on PC7 (Bit 7 of PORTC) to check if ISR is
       running
    DDRC |= (1 << 7);  // Set PC7 as output
    PORTC &= ~(1 << 7); // Initially turn it off

    sei(); // Enable global interrupts
}

ISR(TIMER1_COMPA_vect) {
    PORTC ^= (1 << 7); // Toggle PC7 to check ISR is running

    // Clock Mode Updates
    if (mode == 0) {
        seconds++;
        if (seconds == 60) {
            seconds = 0;
            minutes++;
            if (minutes == 60) {
                minutes = 0;
                hours = (hours + 1) % 24;
            }
        }
    }

    // Timer Countdown (only when running)
    if (mode == 1 && stopwatch_running) {
        if (timer_seconds > 0 || timer_minutes > 0 ||
           timer_hours > 0) {
            if (timer_seconds == 0) {
                if (timer_minutes > 0) {
                    timer_minutes--;
                    timer_seconds = 59;
                } else if (timer_hours > 0) {
```

```
                    timer_hours--;
                    timer_minutes = 59;
                    timer_seconds = 59;
                }
            } else {
                timer_seconds--;
            }
        }
    }

    // Stopwatch Increment
    if (mode == 2 && stopwatch_running) {
        stopwatch_seconds++;
        if (stopwatch_seconds == 60) {
            stopwatch_seconds = 0;
            stopwatch_minutes++;
            if (stopwatch_minutes == 60) {
                stopwatch_minutes = 0;
                stopwatch_hours = (stopwatch_hours + 1) % 24;
            }
        }
    }
}

void displayTime();
void setBCD(int value);
void checkButtons();

int main() {
    setup();
    while (1) {
        checkButtons();
        displayTime();
    }
}

// Function to display time on a 6-digit 7-segment display
void displayTime() {
    int digits[6];

    if (mode == 0) { // Clock Mode
        digits[0] = hours / 10;
        digits[1] = hours % 10;
        digits[2] = minutes / 10;
        digits[3] = minutes % 10;
        digits[4] = seconds / 10;
        digits[5] = seconds % 10;
    } else if (mode == 1) { // Timer Mode
        digits[0] = timer_hours / 10;
```

```
        digits[1] = timer_hours % 10;
        digits[2] = timer_minutes / 10;
        digits[3] = timer_minutes % 10;
        digits[4] = timer_seconds / 10;
        digits[5] = timer_seconds % 10;
    } else { // Stopwatch Mode
        digits[0] = stopwatch_hours / 10;
        digits[1] = stopwatch_hours % 10;
        digits[2] = stopwatch_minutes / 10;
        digits[3] = stopwatch_minutes % 10;
        digits[4] = stopwatch_seconds / 10;
        digits[5] = stopwatch_seconds % 10;
    }

    // Multiplex 7-segment display
    for (int i = 0; i < 6; i++) {
        setBCD(digits[i]); // Send the BCD value first
        COMMON_PORT = (1 << i); // Enable the corresponding
            digit
        _delay_us(500); // Short delay for smooth display
    }
}

// Function to set BCD output for 7-segment display
void setBCD(int value) {
    BCD_PORT = (BCD_PORT & ~BCD_MASK) | ((value << 2) &
        BCD_MASK);
}

// Function to check button inputs and update mode/settings
void checkButtons() {
    if (!(PIND & (1 << PD6))) {
        _delay_ms(50);
        if (!(PIND & (1 << PD6))) {
            if (mode == 0) {
                hours = (hours + 1) % 24;
                seconds = 0;
            } else if (mode == 1) {
                timer_hours = (timer_hours + 1) % 24;
                seconds = 0;
            }
            while (!(PIND & (1 << PD6))); // Wait for release
        }
    }

    if (!(PIND & (1 << PD7))) {
        _delay_ms(50);
        if (!(PIND & (1 << PD7))) {
            if (mode == 0) {
```

```
                minutes = (minutes + 1) % 60;
                seconds = 0;
            } else if (mode == 1) {
                timer_minutes = (timer_minutes + 1) % 60;
                seconds = 0;
            }
            while (!(PIND & (1 << PD7))); // Wait for release
        }
    }

    if (!(PINB & (1 << MODE_BUTTON))) {
        _delay_ms(50);
        if (!(PINB & (1 << MODE_BUTTON))) {
            mode = (mode + 1) % 3; // Cycle through Clock,
                Timer, and Stopwatch
            while (!(PINB & (1 << MODE_BUTTON))); // Wait for
                release
        }
    }

    // Modified section: Stopwatch button controls both Timer
        and Stopwatch
    if (!(PINB & (1 << STOPWATCH_BUTTON))) {
        _delay_ms(50);
        if (!(PINB & (1 << STOPWATCH_BUTTON))) {
            if (mode == 2) {  // Toggle Stopwatch running
                stopwatch_running = !stopwatch_running;
            } else if (mode == 1) {  // Toggle Timer running
                stopwatch_running = !stopwatch_running;  //
                    Reuse the same flag
            }
            while (!(PINB & (1 << STOPWATCH_BUTTON))) {
                _delay_ms(10);
            }
        }
    }
}
```

6 Working of circuit based on the code

- **Clock Mode:** Displays and updates the current time in a 24-hour format.
- **Timer Mode:** Allows countdown from a set time.
- **Stopwatch Mode:** Tracks elapsed time when running.

A six-digit 7-segment display is used for output, and push buttons provide user interaction to switch modes and start/stop timing operations.

Pin Configuration and Display Control

The system uses Binary-Coded Decimal (BCD) representation for displaying digits on the 7-segment display. The display is multiplexed, meaning only one digit is active at a time, and the microcontroller rapidly cycles through them to create a persistent visual effect.

BCD and Digit Selection

```
#define BCD_PORT PORTD
#define BCD_DDR DDRD
#define BCD_MASK 0b00111100  // PD2 to PD5

#define COMMON_PORT PORTC
#define COMMON_DDR DDRC
```

- **BCD_PORT (PORTD, PD2-PD5)** controls the segment encoding.
- **COMMON_PORT (PORTC, PC0-PC5)** selects which digit to activate.

All these pins are set as outputs:

```
BCD_DDR |= BCD_MASK;
COMMON_DDR = 0xFF;
```

Button Configuration

Two push buttons are used for user input:

- **Mode Button (PB0)** - Switches between Clock, Timer, and Stopwatch modes.
- **Start/Stop Button (PB1)** - Starts and stops the stopwatch or timer.

The buttons are connected with internal pull-up resistors to avoid floating states:

```
PORTB |= (1 << MODE_BUTTON) | (1 << STOPWATCH_BUTTON);
```

Time Tracking Variables

Three sets of time variables store hours, minutes, and seconds for each mode:

```
volatile int seconds = 0, minutes = 0, hours = 15;
volatile int timer_seconds = 0, timer_minutes = 0,
    timer_hours = 0;
volatile int stopwatch_seconds = 0, stopwatch_minutes = 0,
    stopwatch_hours = 0;
```

- **Clock mode** starts with an initial time (e.g., 15:00:00).
- **Timer mode** counts down when started.
- **Stopwatch mode** increments when running.

Interrupt-Driven Time Updates

A hardware timer (Timer1) generates an interrupt every second to update time values.

Timer1 Configuration

```
TCCR1B |= (1 << WGM12) | (1 << CS12) | (1 << CS10);
OCR1A = 15625;
TIMSK1 |= (1 << OCIE1A);
```

- Configures Timer1 in **CTC mode** (Clear Timer on Compare Match).
- Uses a prescaler of 1024 to achieve a 1-second interval.
- Triggers an interrupt when the timer reaches 15625 counts.

Interrupt Service Routine (ISR)

Every second, the ISR updates the appropriate time variables based on the active mode.

Clock Mode

```
if (mode == 0) {
    seconds++;
    if (seconds == 60) {
        seconds = 0;
        minutes++;
```

```
        if (minutes == 60) {
            minutes = 0;
            hours = (hours + 1) % 24;
        }
    }
}
```

- Increments seconds.
- Rolls over to minutes and hours when necessary.

Timer Mode

```
if (mode == 1 && stopwatch_running) {
    if (timer_seconds == 0) {
        if (timer_minutes > 0) {
            timer_minutes--;
            timer_seconds = 59;
        } else if (timer_hours > 0) {
            timer_hours--;
            timer_minutes = 59;
            timer_seconds = 59;
        } } else {
        timer_seconds--;
    }
}
```

Stopwatch Mode

```
if (mode == 2 && stopwatch_running) {
    stopwatch_seconds++;
    if (stopwatch_seconds == 60) {
        stopwatch_seconds = 0;
        stopwatch_minutes++;
        if (stopwatch_minutes == 60) {
            stopwatch_minutes = 0;
            stopwatch_hours = (stopwatch_hours + 1) % 24;
            }
    }
}
```

- Increments time while running.
- Rolls over when needed.

7-Segment Display Multiplexing

The function `displayTime()` handles time display:

```
void displayTime() {
    int digits[6];

    if (mode == 0) {
        digits[0] = hours / 10;
        digits[1] = hours % 10;
        digits[2] = minutes / 10;
        digits[3] = minutes % 10;
        digits[4] = seconds / 10;
        digits[5] = seconds % 10;
    }
    ...
    for (int i = 0; i < 6; i++) {
        setBCD(digits[i]);
        COMMON_PORT = (1 << i);
        _delay_us(500);
    }
}
```

- Converts the current mode's time into six digits.
- Updates one digit at a time using BCD.
- Uses a short delay to ensure smooth display.

Button Handling

The function `checkButtons()` reads button inputs and updates settings accordingly.

```
if (!(PINB & (1 << MODE_BUTTON))) {
    _delay_ms(50);
    if (!(PINB & (1 << MODE_BUTTON))) {
        mode = (mode + 1) % 3;
        while (!(PINB & (1 << MODE_BUTTON)));
    }
}
```

- Checks if the mode button is pressed.
- Debounces input using a delay.

- Cycles through Clock, Timer, and Stopwatch.

The start/stop button toggles operation:

```
if (!(PINB & (1 << STOPWATCH_BUTTON))) {
    _delay_ms(50);
    if (!(PINB & (1 << STOPWATCH_BUTTON))) {
        stopwatch_running = !stopwatch_running;
        while (!(PINB & (1 << STOPWATCH_BUTTON))) {
            _delay_ms(10);
        }
    }
}
```

7 Results

- The clock is successfully able to display time.
- Stopwatch is also functioning correctly.
- Countdown timer also works perfectly.
- The push helps in easy controlling and adjustment of time in clock as well as the others

8 Conclusion

This project implements a functional digital clock with help from Arduino. Although its simple, easy and functioning but there is still room for improvement.