Digital Clock

Eshan Ray
EE24BTECH11021

March 24, 2025

Abstract

This document presents the design and implementation of an Arduino Uno-based digital clock featuring a countdown timer mode. The project utilizes an Arduino Uno and a 7-segment display for time representation. The system supports real-time clock functionality, user-configurable time adjustments, and a timer mode with auto-stop when time reaches zero.

1 Introduction

The purpose of this project is to develop a digital clock using an Arduino Uno. The clock includes features such as time display, time adjustment via push buttons, and a timer mode. The clock operates based on a millisecond timer interrupt and follows the 24-hour format.

2 Hardware Components

- Arduino Uno
- 7-Segment Display (6 Digits)
- Push Buttons (Hour, Minute, Second Increment/Decrement, Timer Mode, Reset)
- Resistors and Capacitors
- Power Supply (5V)

3 Circuit Connections

The following table outlines the connections between the Arduino Uno and various components:

Component	Arduino Pin	Description
Hour Increment Button	PB0	Increments the hour count
Hour Decrement Button	PB1	Decrements the hour count
Minute Increment Button	PC5	Increments the minute count
Minute Decrement Button	PC3	Decrements the minute count
Second Increment Button	PD0	Increments the second count
Timer Mode Button	PC4	Toggles timer mode on/off
Reset Button	PC2	Resets or restores time settings
7-Segment Display	PD2, PD3, PD4, PD5	Used for digit selection (decoder)
Display Enable Lines	PB5, PB4, PC1, PC0, PB3, PD7	Selects active digit

Each button is connected to a specific pin on the Arduino Uno with internal pull-up resistors enabled. The display is multiplexed using a decoder circuit and selection lines to cycle through the six digits.

4 Software Implementation

The project is implemented in C using the AVR-GCC compiler. The system structure includes:

- Millisecond timer interrupt for real-time updates.
- Button handling with debounce and long-press detection.
- Time management functions for incrementing, decrementing, and resetting time.
- Display multiplexing for 7-segment display using a decoder.

5 Code Implementation

The main functionalities are defined in the following sections:

5.1 Timer Interrupt for Millis Counter

The system maintains a millisecond counter using the Timer0 Compare Match interrupt.

Listing 1: Millis Timer Interrupt

```
ISR(TIMER0\_COMPA\_vect) {
millis\_count++;
}

unsigned long currentMillis(void) {
unsigned long m;
```

```
cli();
m = millis\_count;
sei();
return m;
}
```

5.2 Button Handling

Each button is mapped to specific microcontroller pins. A debounce mechanism prevents false triggers.

Listing 2: Button Checking Function

```
void checkButtons(void) {
static unsigned long holdStart[NUM\_BUTTONS] = {0};
void (*actions[])() = {incrementHour, decrementHour, incrementMinute,
decrementMinute, toggleTimerMode, incrementSecond, resetClock};
// Process button states and handle actions
}
```

5.3 Time Update Functions

The system supports both time increment and decrement operations, ensuring proper rollover at 24-hour and 60-minute limits.

Listing 3: Increment Time

```
void incrementTime(void) {
seconds++;
if(seconds > 59) {
seconds = 0;
minutes++;
if(minutes > 59) {
minutes = 0;
hours++;
if(hours > 23) {
hours = 0;
}
}
}
}
```

5.4 Display Multiplexing

The 7-segment display is controlled using a decoder and port-based switching. Since the Arduino Uno has a limited number of I/O pins, the display multiplexing is done by enabling only one display at a time while sending the corresponding digit data.

Listing 4: Show Digit Function

```
void showDigit(uint8\_t digit, uint8\_t displayIndex) {
*decoderPort &= ~((1 << PD2) | (1 << PD5) | (1 << PD4) | (1 << PD3));
if(digit & 0x01) *decoderPort |= (1 << PD2);
if(digit & 0x02) *decoderPort |= (1 << PD5);
if(digit & 0x04) *decoderPort |= (1 << PD4);
if(digit & 0x08) *decoderPort |= (1 << PD3);
}
```

6 Results and Testing

The clock was successfully tested for accurate timekeeping and responsiveness to user inputs. The timer mode correctly halts at zero, and button presses reliably adjust time settings.

7 Conclusion

This project demonstrates an effective implementation of a digital clock with a countdown timer using an Arduino Uno. Future improvements could include an OLED display for enhanced readability.

8 Project Repository

The source code and additional details can be found at:
GitHub Repository