Digital Clock

John Bobby
ee24btech11032

1 Objective

This project involves the design of a digital clock using seven-segment displays, a single 7447 IC, and an Arduino. The 7447 IC is utilized to control all the displays efficiently, with the code implemented in embedded C.

2 Hardware Components

- Arduino
- 1 7447 IC
- 6 seven segment displays
- 6 220 Ω Resistors
- Jumper Wires
- Normal wires

.

3 Connections

Connections are made according to the below table:

Pin(Arduino)	**Connected To**
2,3,4,5	A,B,C,D of 7447 IC
6,7,8,9,10,11	COM of all the 6 displays
5V, GND	V_{cc} and GND of 7447 IC

Table 1: Connections

The seven-segment displays are connected with each other so that all the inputs for all the displays are the same.

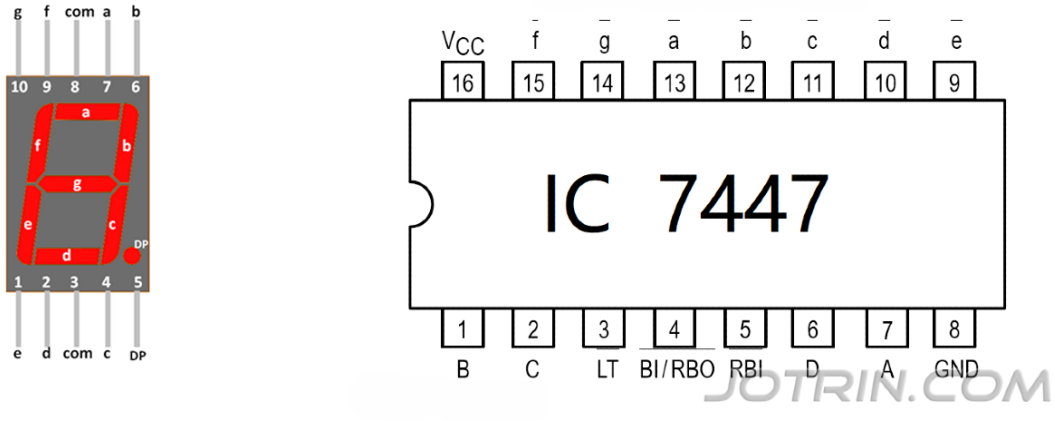

Figure 1: Caption

4 Logic

- We can turn the displays on or off by controlling the digital pins (6-11).
- To change a digit at a specific position, we turn on only the corresponding display while keeping the others off. We then apply the updated digit value to the display.
- This switching happens very rapidly, so each display is turned off for only a very short time. Due to persistence of vision, our eyes perceive all displays as continuously lit.

5 Multiplexing In Code

Below is the implementation of the multiplexing logic in Embedded C:

```
void displayTime() {
    uint8_t h1 = (hours >> 4) & 0x0F;  // Extract tens digit of
       hours
    uint8_t h2 = hours & 0x0F;         // Extract units digit of
       hours
    uint8_t m1 = (minutes >> 4) & 0x0F;
    uint8_t m2 = minutes & 0x0F;
    uint8_t s1 = (seconds >> 4) & 0x0F;
    uint8_t s2 = seconds & 0x0F;

    // Displaying each digit sequentially
    PORTD |= (1 << H1); displayDigit(h1); _delay_ms(5); PORTD &= ~(1
        << H1);
    PORTD |= (1 << H2); displayDigit(h2); _delay_ms(5); PORTD &= ~(1
        << H2);
    PORTB |= (1 << M1); displayDigit(m1); _delay_ms(5); PORTB &= ~(1
        << M1);
```

```
    PORTB |= (1 << M2); displayDigit(m2); _delay_ms(5); PORTB &= ~(1 << M2);
    PORTB |= (1 << S1); displayDigit(s1); _delay_ms(5); PORTB &= ~(1 << S1);
    PORTB |= (1 << S2); displayDigit(s2); _delay_ms(5); PORTB &= ~(1 << S2);
}
```

Listing 1: Multiplexing Logic for 7-Segment Display

- **Digit Extraction**: Each digit for hours, minutes, seconds is stored in respective variables.
- **Activation of Displays**: The display is activated using PORTD(an 8bit register used to control the output of digital pins).
- **Delay**:5ms delay is introduced.
- **Deactivation**: The display is deactivated.

$$T_{display} = 6 \times 5ms = 30ms$$

On calling the function above the time is displayed for 30ms.
This function is repeatedly called inside the loop all the time is updated by Timer1 interrupts.

6 Using Timer-1

Below is the intialisation of Timer1

```
void displayTime() {
    uint8_t h1 = (hours >> 4) & 0x0F;  // Extract tens digit of hours
    uint8_t h2 = hours & 0x0F;         // Extract units digit of hours
    uint8_t m1 = (minutes >> 4) & 0x0F;
    uint8_t m2 = minutes & 0x0F;
    uint8_t s1 = (seconds >> 4) & 0x0F;
    uint8_t s2 = seconds & 0x0F;

    // Displaying each digit sequentially
    PORTD |= (1 << H1); displayDigit(h1); _delay_ms(5); PORTD &= ~(1 << H1);
    PORTD |= (1 << H2); displayDigit(h2); _delay_ms(5); PORTD &= ~(1 << H2);
    PORTB |= (1 << M1); displayDigit(m1); _delay_ms(5); PORTB &= ~(1 << M1);
```

```
    PORTB |= (1 << M2); displayDigit(m2); _delay_ms(5); PORTB &= ~(1
        << M2);
    PORTB |= (1 << S1); displayDigit(s1); _delay_ms(5); PORTB &= ~(1
        << S1);
    PORTB |= (1 << S2); displayDigit(s2); _delay_ms(5); PORTB &= ~(1
        << S2);
}
```

Listing 2: Multiplexing Logic for 7-Segment Display

- Timer1 generates an interrupt every second for updating the display.
- Ensures precise timekeeping.

7 Conclusion

7.1 Advantages of Multiplexing

- **Reduces Hardware Requirements:** Using only one 7447 IC instead of separate decoder ICs for each display minimizes the number of components.
- **Lower Power Consumption:** At any given moment, only one display is actively drawing current, leading to lower overall power consumption.
- **Cost-Effective:** Since fewer components are required, the overall cost of the circuit is reduced.
- **Efficient Use of Microcontroller Pins:** Instead of dedicating multiple I/O pins to each display, we only use a few pins for selecting displays, making it ideal for resource-constrained microcontrollers.

7.2 Disadvantages of Multiplexing

- **Processing Overhead:** The microcontroller continuously executes the multiplexing logic, consuming processing time that could otherwise be used for other tasks.
- **Perceived Brightness Reduction:** Since each display is turned on only for a fraction of the time, the brightness appears lower compared to direct driving methods.

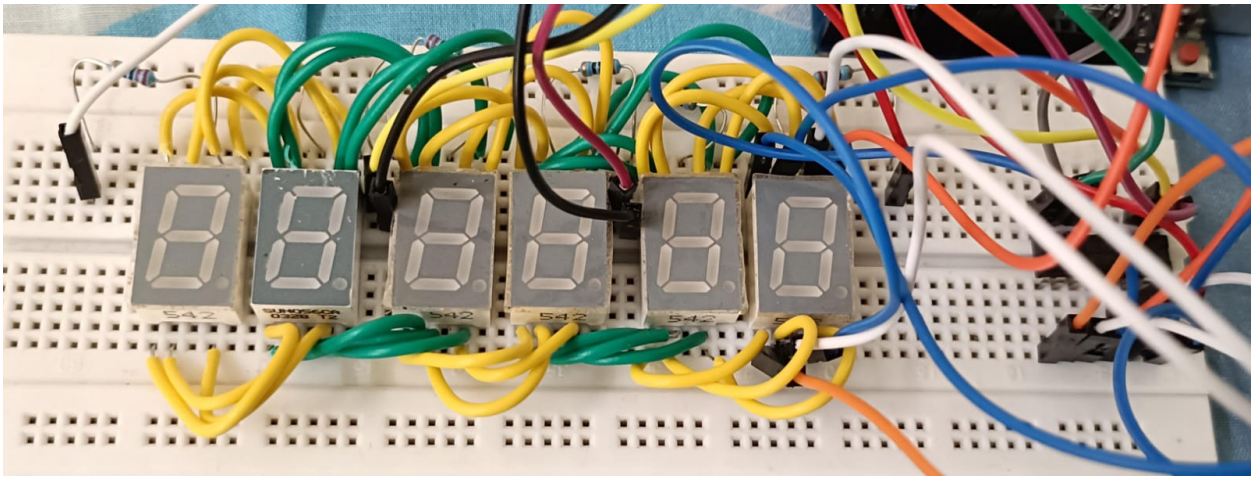

Figure 2: Clock