digital clock

Jadhav Rajesh-EE24BTECH11028

1 Objective

The objective of this experiment is to design and implement a digital clock using an **Arduino** and the **7447 BCD to 7-segment decoder/driver IC**. The clock displays time in the **HH:MM:SS** format on six 7-segment displays. The Arduino generates the BCD (Binary-Coded Decimal) output, which is decoded by the 7447 IC to drive the 7-segment displays.

2 Components Required

- Arduino Uno (or compatible board).
- Six 7-segment displays (common cathode or common anode).
- Six 7447 BCD to 7-segment decoder/driver ICs.
- Resistors (220Ω for current limiting in 7-segment displays).
- Breadboard and connecting wires.
- Power supply (5V DC from Arduino or external source).
- Push buttons (optional, for manual time setting).

3 Circuit Design

The digital clock is designed using the following components and connections:

3.1 Arduino to 7447 IC

- Connect the 4 BCD output pins of the Arduino (e.g., D2, D3, D4, D5) to the BCD input pins (A, B, C, D) of the 7447 IC.
- Connect the GND pin of the Arduino to the GND pin of the 7447 IC.

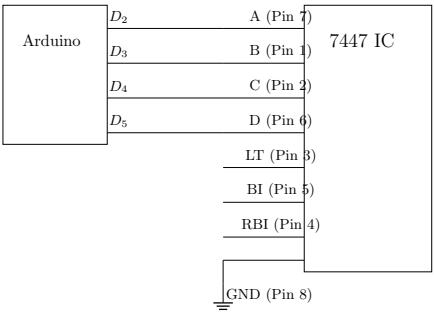
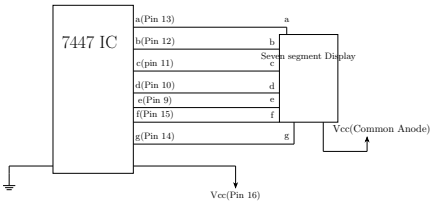


Figure 1: Connections between Arduino and 7447 BCD to 7-Segment Decoder

Fig. 1: Circuit Diagram

3.2 7447 IC to 7-Segment Displays

- Connect the output pins (a, b, c, d, e, f, g) of the 7447 IC to the corresponding segments of the 7-segment display.
- Use current-limiting resistors 200Ω between the 7447 outputs and the 7-segment display segments.
- Connect the common cathode (or anode) of the 7-segment display to GND (or VCC) depending on the type of display.



1

Fig. 2: Circuit Diagram

3.3 Multiplexing

- Use six 7-segment displays for hours, minutes, and seconds.
- Use additional 7447 ICs for each digit (total of six 7447 ICs).
- Connect the BCD outputs of the Arduino to each 7447 IC in parallel, but use separate Arduino pins to control which digit is active at a time (multiplexing).
- Use Arduino digital pins (e.g., D6, D7, D8, D9, D10, D11) to control the common cathode (or anode) of each 7-segment display.

4 Experimental Procedure

1) **Circuit Connections**:

- Connect the Arduino to the 7447 IC and 7-segment displays as described in the circuit design.
- Ensure proper current-limiting resistors are used for the 7-segment displays.

2) **Arduino Code**:
   - Write and upload the provided Arduino code to implement the digital clock.
3) **Testing**:
   - Power on the circuit and verify that the displays show the initial time (12:00:00).
   - Observe the seconds incrementing every second.
   - Check that the minutes and hours increment correctly.
4) **Optional Enhancements**:
   - Add push buttons to manually set the hours, minutes, and seconds.
   - Add an AM/PM indicator for a 12-hour format.

TABLE I: Arduino to 7-Segment Display Connections

Arduino Pin	7-Segment Display Pin
2	a
3	b
4	c
5	d
6	e
7	f
8	g
9	COM (1st Display)
10	COM (2nd Display)
11	COM (3rd Display)
12	COM (4th Display)
A0	COM (5th Display)
A1	COM (6th Display)

5 Arduino Code

Below is the Arduino code used to implement the digital clock:

```
#include <stdint.h>
#include <stdbool.h>
#include <avr/io.h> // For AVR microcontrollers
#include <util/delay.h> // For delay functions

// Define BCD output pins
#define BCD_A 2
#define BCD_B 3
```

```
#define BCD_C 4
#define BCD_D 5

// Define digit control pins
const uint8_t digitPins[6] = {6, 7, 8, 9, 10, 11};

// Variables to store time
uint8_t hours = 12;
uint8_t minutes = 0;
uint8_t seconds = 0;

// Function to set a specific pin HIGH or LOW
void setPin(uint8_t pin, bool state) {
    if (state) {
        PORTB |= (1 << pin); // Set pin HIGH
    } else {
        PORTB &= ~(1 << pin); // Set pin LOW
    }
}

// Function to display a digit on the 7-segment display
void displayDigit(uint8_t digit, uint8_t value) {
    // Set the BCD output based on the value
    setPin(BCD_A, value & 0x01);
    setPin(BCD_B, (value >> 1) & 0x01);
    setPin(BCD_C, (value >> 2) & 0x01);
    setPin(BCD_D, (value >> 3) & 0x01);

    // Turn on the corresponding digit
    setPin(digitPins[digit], true);
    _delay_ms(5); // Display for a short time
    setPin(digitPins[digit], false); // Turn off the digit
}

// Function to update the time
void updateTime() {
    seconds++;
    if (seconds >= 60) {
        seconds = 0;
        minutes++;
        if (minutes >= 60) {
            minutes = 0;
            hours++;
            if (hours >= 24) {
                hours = 0;
            }
        }
    }
}

int main(void) {
    // Set BCD pins as output
    DDRB |= (1 << BCD_A) | (1 << BCD_B) | (1 << BCD_C) | (1 << BCD_D);
```

```
62
63      // Set digit control pins as output
64      for (uint8_t i = 0; i < 6; i++) {
65          DDRB |= (1 << digitPins[i]);
66          setPin(digitPins[i], false); // Turn off all digits initially
67      }
68
69      // Main loop
70      while (1) {
71          // Update time every second
72          static uint32_t lastTime = 0;
73          uint32_t currentTime = millis(); // Implement millis() for your
                platform
74          if (currentTime - lastTime >= 1000) {
75              lastTime = currentTime;
76              updateTime();
77          }
78
79          // Display hours, minutes, and seconds
80          displayDigit(0, hours / 10); // Tens place of hours
81          displayDigit(1, hours % 10); // Units place of hours
82          displayDigit(2, minutes / 10); // Tens place of minutes
83          displayDigit(3, minutes % 10); // Units place of minutes
84          displayDigit(4, seconds / 10); // Tens place of seconds
85          displayDigit(5, seconds % 10); // Units place of seconds
86      }
87
88      return 0;
89  }
```

Listing 1: Arduino Code for Digital Clock

6 Observations and Results

- The Arduino generated the correct BCD output for each digit.
- The 7447 IC successfully decoded the BCD signals and drove the 7-segment displays.
- The digital clock displayed the time accurately in the HH:MM:SS format.
- Multiplexing ensured that all digits were displayed clearly without flickering.

7 Conclusion

The digital clock was successfully implemented using an Arduino and the 7447 BCD to 7-segment decoder/driver IC. The circuit accurately displayed time in the HH:MM:SS format. Future enhancements could include adding an AM/PM indicator, alarm functionality, and manual time-setting buttons.