Arduino-Based Digital Clock Using Seven-Segment Displays and 7447 Decoder

Niketh Achanta - EE24BTECH11047

Abstract

This report presents the design and implementation of an Arduino-based digital clock using six seven-segment displays and a 7447 BCD-to-seven-segment decoder. The clock functions using multiplexing to efficiently control the displays and maintain accurate timekeeping without using an external RTC module.

1 Introduction

Digital clocks are an essential part of embedded systems and electronic circuits. This project aims to design a functional digital clock using an **Arduino**, **six seven-segment displays**, and a **7447 BCD-to-seven-segment decoder**. The clock displays hours, minutes, and seconds while ensuring efficient power usage and reducing the number of I/O pins required. Instead of directly controlling all six displays, multiplexing is used to rapidly switch between them.

2 Components Used

The main components required for this project are listed in Table 1.

Component	**Description**
Arduino Uno	Microcontroller used for controlling the clock
6x Seven-Segment Display	Displays hours, minutes, and seconds
7447 BCD to 7-Segment Decoder	Converts BCD values to 7-segment signals
Resistors (220Ω)	Current limiting resistors for display protection
Wires & Breadboard	Circuit assembly and connections

Table 1: Components used in the project

3 Multiplexing Concept

Since an Arduino does not have enough I/O pins to control all six displays independently, **multiplexing** is used. This technique involves activating one display at a time while rapidly cycling through all six, creating the illusion of a continuous display due to **persistence of vision**.

3.1 How Multiplexing Works

- The Arduino outputs a **BCD** value corresponding to each digit.
- The 7447 decoder converts BCD to the seven-segment format.
- Control signals determine which display is active at any given moment.
- The Arduino cycles through all six displays multiple times per second.
- Each display is active for a short period (typically 5ms).
- At a refresh rate above 60Hz (entire cycle less than 16.7ms), no flickering is visible.

4 Circuit Diagram

Figure 1 illustrates the seven-segment display pin configuration, and Figure 2 shows the 7447 BCD decoder.

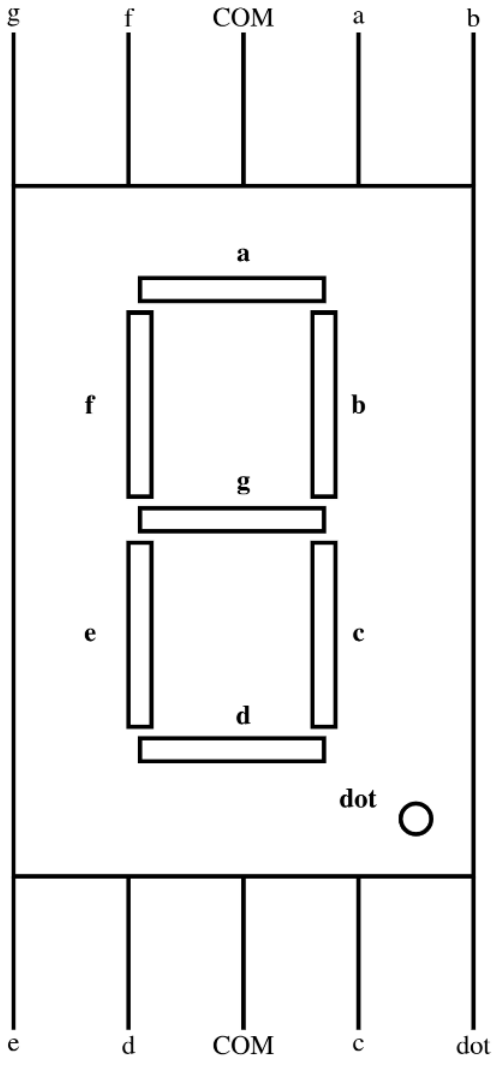

Figure 1: Common Cathode Seven-Segment Display Pinout

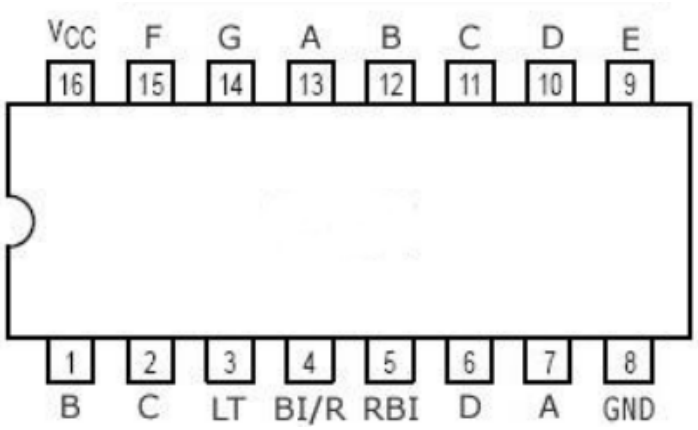

Figure 2: 7447 BCD-to-Seven-Segment Decoder Pinout

5 Pinout diagrams

The circuit consists of three main elements:

5.1 Display Circuit

Each seven-segment display is connected to the 7447 decoder through current-limiting resistors. The common cathode of each display is connected to ground through a transistor that acts as a switch.

5.2 Decoder Circuit

The 7447 decoder receives 4-bit BCD inputs from the Arduino and converts them to the appropriate seven-segment patterns. All six displays share the same decoder outputs, but only one display is active at any given time.

5.3 Control Circuit

The Arduino controls which display is active by turning on the corresponding transistor. Six output pins (PD6, PD7, PB0, PB1, PB2, PB3) are used to select which display is active, while four pins (PD2, PD3, PD4, PD5) provide the BCD data.

6 Code Implementation

The Arduino code is structured as follows:

- Initializes all required pins for **BCD input** and **display control**.
- Uses a **timer interrupt** to increment the clock every second.
- Implements **multiplexing** to update the display efficiently.

6.1 Code Explanation

6.1.1 BCD Time Representation

Time values are stored in Binary-Coded Decimal (BCD) format, where each decimal digit is represented by its 4-bit binary equivalent. For example, 12:34:56 is stored as:

- hours = 0001 0010 (0x12)
- minutes = 0011 0100 (0x34)
- seconds = 0101 0110 (0x56)

6.1.2 Timer Configuration

The timer is configured to generate an interrupt every second:

- Timer1 operates in CTC (Clear Timer on Compare) mode
- With a 16MHz clock and 1024 prescaler, Timer1 increments at 15,625Hz
- The compare match value (OCR1A) is set to 15,624, resulting in an interrupt frequency of 1Hz

6.1.3 Multiplexing Implementation

The displayTime() function implements multiplexing by:

1. Extracting individual digits from the BCD time variables
2. Activating one display at a time using its control pin
3. Setting the BCD output pins to show the correct digit
4. Keeping each display on for 5ms before moving to the next
5. Cycling through all six displays continuously

7 Conclusion

This project successfully implemented a digital clock using an Arduino, multiplexed seven-segment displays, and a 7447 decoder. The multiplexing approach significantly reduces the required I/O pins while maintaining accurate timekeeping. The design uses timer interrupts for precise one-second timing and efficiently manages six displays using just ten output pins from the Arduino.

The implementation demonstrates fundamental concepts in digital electronics including:

- Multiplexing techniques for display control
- BCD to seven-segment conversion
- Timer configuration for precise timing
- Interrupt-based timekeeping