

Digital Clock using Arduino and 7-Segment Displays

Srihaas Gunda-EE24BTECH11026

Contents

1	Introduction	2
2	Components used	2
3	Circuit Design	2
4	Code	4
5	Working of circuit based on the code	9
6	Results	13
7	Conclusion	14

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

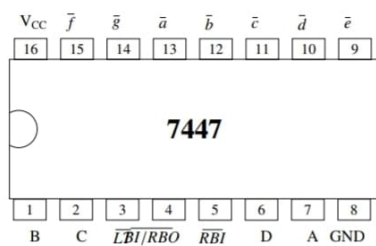


Fig. 3.1: 7447 IC

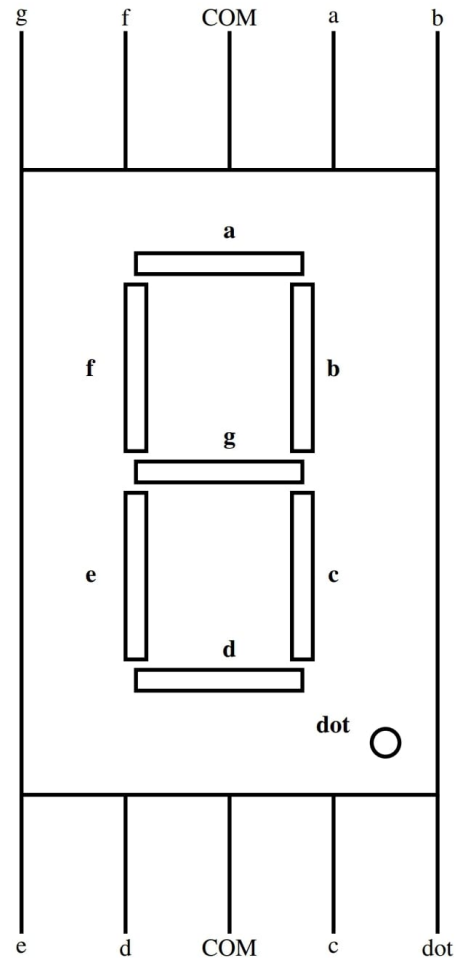


Fig. 2.2: Seven Segment pins

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.

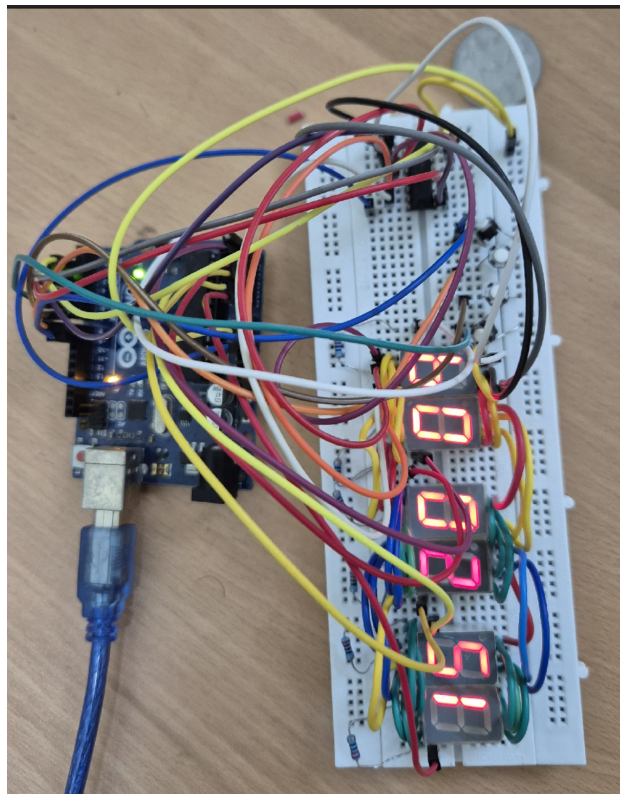
There are also 4 push buttons which are connected to 6,7,8 and 9 pins in the Arduino . They have the following uses

- 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 Code

The following is the code implemented

```
1 #define F_CPU 16000000UL
2 #include <avr/io.h>
3 #include <util/delay.h>
4 #include <avr/interrupt.h>
5
6 #define BCD_PORT PORTD
7 #define BCD_DDR DDRD
8 #define BCD_MASK 0b00111100 // PD2 to PD5
```

```

9
10 #define COMMON_PORT PORTC
11 #define COMMON_DDR DDRC
12
13 #define MODE_BUTTON PBO // Switch between Clock, Timer, and
    Stopwatch
14 #define STOPWATCH_BUTTON PB1 // Start/Stop Stopwatch and
    Timer
15
16 volatile int seconds = 0, minutes = 0, hours = 15;
17 volatile int timer_seconds = 0, timer_minutes = 0,
    timer_hours = 0;
18 volatile int stopwatch_seconds = 0, stopwatch_minutes = 0,
    stopwatch_hours = 0;
19 volatile int mode = 0; // 0 = Clock, 1 = Timer, 2 = Stopwatch
20 volatile int stopwatch_running = 0; // 1 = Running, 0 =
    Stopped
21
22 void setup() {
23     // Set BCD display pins (PD2-PD5) as output
24     BCD_DDR |= BCD_MASK;
25     BCD_PORT &= ~BCD_MASK;
26
27     // Set digit selector pins (PORTC) as output
28     COMMON_DDR = 0xFF;
29     COMMON_PORT = 0x00;
30
31     // Enable pull-up resistors for buttons
32     PORTD |= (1 << PD6) | (1 << PD7);
33     PORTB |= (1 << MODE_BUTTON) | (1 << STOPWATCH_BUTTON);
34
35     // Timer1 Setup: CTC Mode, 1-second interval
36     TCCR1B |= (1 << WGM12) | (1 << CS12) | (1 << CS10);
37     OCR1A = 15625; // 1-second interrupt
38     TIMSK1 |= (1 << OCIE1A);
39
40     // Debug LED on PC7 (Bit 7 of PORTC) to check if ISR is
        running
41     DDRC |= (1 << 7); // Set PC7 as output
42     PORTC &= ~(1 << 7); // Initially turn it off
43
44     sei(); // Enable global interrupts
45 }
46
47 ISR(TIMER1_COMPA_vect) {
48     PORTC ^= (1 << 7); // Toggle PC7 to check ISR is running
49
50     // Clock Mode Updates
51     if (mode == 0) {

```

```

52         seconds++;
53         if (seconds == 60) {
54             seconds = 0;
55             minutes++;
56             if (minutes == 60) {
57                 minutes = 0;
58                 hours = (hours + 1) % 24;
59             }
60         }
61     }
62
63     // Timer Countdown (only when running)
64     if (mode == 1 && stopwatch_running) {
65         if (timer_seconds > 0 || timer_minutes > 0 ||
66             timer_hours > 0) {
67             if (timer_seconds == 0) {
68                 if (timer_minutes > 0) {
69                     timer_minutes--;
70                     timer_seconds = 59;
71                 } else if (timer_hours > 0) {
72                     timer_hours--;
73                     timer_minutes = 59;
74                     timer_seconds = 59;
75                 }
76             } else {
77                 timer_seconds--;
78             }
79         }
80
81         // Stopwatch Increment
82         if (mode == 2 && stopwatch_running) {
83             stopwatch_seconds++;
84             if (stopwatch_seconds == 60) {
85                 stopwatch_seconds = 0;
86                 stopwatch_minutes++;
87                 if (stopwatch_minutes == 60) {
88                     stopwatch_minutes = 0;
89                     stopwatch_hours = (stopwatch_hours + 1) % 24;
90                 }
91             }
92         }
93     }
94
95     void displayTime();
96     void setBCD(int value);
97     void checkButtons();
98
99     int main() {

```

```

100     setup();
101     while (1) {
102         checkButtons();
103         displayTime();
104     }
105 }
106
107 // Function to display time on a 6-digit 7-segment display
108 void displayTime() {
109     int digits[6];
110
111     if (mode == 0) { // Clock Mode
112         digits[0] = hours / 10;
113         digits[1] = hours % 10;
114         digits[2] = minutes / 10;
115         digits[3] = minutes % 10;
116         digits[4] = seconds / 10;
117         digits[5] = seconds % 10;
118     } else if (mode == 1) { // Timer Mode
119         digits[0] = timer_hours / 10;
120         digits[1] = timer_hours % 10;
121         digits[2] = timer_minutes / 10;
122         digits[3] = timer_minutes % 10;
123         digits[4] = timer_seconds / 10;
124         digits[5] = timer_seconds % 10;
125     } else { // Stopwatch Mode
126         digits[0] = stopwatch_hours / 10;
127         digits[1] = stopwatch_hours % 10;
128         digits[2] = stopwatch_minutes / 10;
129         digits[3] = stopwatch_minutes % 10;
130         digits[4] = stopwatch_seconds / 10;
131         digits[5] = stopwatch_seconds % 10;
132     }
133
134     // Multiplex 7-segment display
135     for (int i = 0; i < 6; i++) {
136         setBCD(digits[i]); // Send the BCD value first
137         COMMON_PORT = (1 << i); // Enable the corresponding
            digit
138         _delay_us(500); // Short delay for smooth display
139     }
140 }
141
142 // Function to set BCD output for 7-segment display
143 void setBCD(int value) {
144     BCD_PORT = (BCD_PORT & ~BCD_MASK) | ((value << 2) &
        BCD_MASK);
145 }
146

```

```

147 // Function to check button inputs and update mode/settings
148 void checkButtons() {
149     if (!(PIND & (1 << PD6))) {
150         _delay_ms(50);
151         if (!(PIND & (1 << PD6))) {
152             if (mode == 0) {
153                 hours = (hours + 1) % 24;
154                 seconds = 0;
155             } else if (mode == 1) {
156                 timer_hours = (timer_hours + 1) % 24;
157                 seconds = 0;
158             }
159             while (!(PIND & (1 << PD6))); // Wait for release
160         }
161     }
162
163     if (!(PIND & (1 << PD7))) {
164         _delay_ms(50);
165         if (!(PIND & (1 << PD7))) {
166             if (mode == 0) {
167                 minutes = (minutes + 1) % 60;
168                 seconds = 0;
169             } else if (mode == 1) {
170                 timer_minutes = (timer_minutes + 1) % 60;
171                 seconds = 0;
172             }
173             while (!(PIND & (1 << PD7))); // Wait for release
174         }
175     }
176
177     if (!(PINB & (1 << MODE_BUTTON))) {
178         _delay_ms(50);
179         if (!(PINB & (1 << MODE_BUTTON))) {
180             mode = (mode + 1) % 3; // Cycle through Clock,
181                                     Timer, and Stopwatch
182             while (!(PINB & (1 << MODE_BUTTON))); // Wait for
183                                     release
184         }
185     }
186
187     // Modified section: Stopwatch button controls both Timer
188     // and Stopwatch
189     if (!(PINB & (1 << STOPWATCH_BUTTON))) {
190         _delay_ms(50);
191         if (!(PINB & (1 << STOPWATCH_BUTTON))) {
192             if (mode == 2) { // Toggle Stopwatch running
193                 stopwatch_running = !stopwatch_running;
194             } else if (mode == 1) { // Toggle Timer running
195                 stopwatch_running = !stopwatch_running; //

```



```

193         }
194         while (!(PINB & (1 << STOPWATCH_BUTTON))) {
195             _delay_ms(10);
196         }
197     }
198 }
199 }

```

5 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

```

1 #define BCD_PORT PORTD
2 #define BCD_DDR DDRD
3 #define BCD_MASK 0b00111100 // PD2 to PD5
4
5 #define COMMON_PORT PORTC
6 #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:

```
1 BCD_DDR |= BCD_MASK;
2 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:

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

Time Tracking Variables

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

```
1 volatile int seconds = 0, minutes = 0, hours = 15;
2 volatile int timer_seconds = 0, timer_minutes = 0,
  timer_hours = 0;
3 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

```
1 TCCR1B |= (1 << WGM12) | (1 << CS12) | (1 << CS10);
2 OCR1A = 15625;
3 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

```
1 if (mode == 0) {
2     seconds++;
3     if (seconds == 60) {
4         seconds = 0;
5         minutes++;
6         if (minutes == 60) {
7             minutes = 0;
8             hours = (hours + 1) % 24;
9         }
10    }
11 }
```

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

Timer Mode

```
1 if (mode == 1 && stopwatch_running) {
2     if (timer_seconds == 0) {
3         if (timer_minutes > 0) {
4             timer_minutes--;
5             timer_seconds = 59;
6         } else if (timer_hours > 0) {
7             timer_hours--;
8             timer_minutes = 59;
9             timer_seconds = 59;
10        } } else {
```

```

11         timer_seconds--;
12     }
13 }

```

Stopwatch Mode

```

1  if (mode == 2 && stopwatch_running) {
2      stopwatch_seconds++;
3      if (stopwatch_seconds == 60) {
4          stopwatch_seconds = 0;
5          stopwatch_minutes++;
6          if (stopwatch_minutes == 60) {
7              stopwatch_minutes = 0;
8              stopwatch_hours = (stopwatch_hours + 1) % 24;
9          }
10     }
11 }

```

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

7-Segment Display Multiplexing

The function `displayTime()` handles time display:

```

1  void displayTime() {
2      int digits[6];
3
4      if (mode == 0) {
5          digits[0] = hours / 10;
6          digits[1] = hours % 10;
7          digits[2] = minutes / 10;
8          digits[3] = minutes % 10;
9          digits[4] = seconds / 10;
10         digits[5] = seconds % 10;
11     }
12     ...
13     for (int i = 0; i < 6; i++) {
14         setBCD(digits[i]);
15         COMMON_PORT = (1 << i);
16         _delay_us(500);
17     }
18 }

```

- 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.

```

1  if (!(PINB & (1 << MODE_BUTTON))) {
2      _delay_ms(50);
3      if (!(PINB & (1 << MODE_BUTTON))) {
4          mode = (mode + 1) % 3;
5          while (!(PINB & (1 << MODE_BUTTON)));
6      }
7  }

```

- Checks if the mode button is pressed.
- Debounces input using a delay.
- Cycles through Clock, Timer, and Stopwatch.

The start/stop button toggles operation:

```

1  if (!(PINB & (1 << STOPWATCH_BUTTON))) {
2      _delay_ms(50);
3      if (!(PINB & (1 << STOPWATCH_BUTTON))) {
4          stopwatch_running = !stopwatch_running;
5          while (!(PINB & (1 << STOPWATCH_BUTTON))) {
6              _delay_ms(10);
7          }
8      }
9  }

```

6 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

7 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.