Arduino digital clock

Jashwanth Medamoni

March 2025

1 Introduction

The purpose of this project is to design and implement a digital clock using an Arduino and seven-segment displays.The clock tracks hours, minutes, and seconds, starting from a defined initial time. The Arduino handles timekeeping through internal programming, updating the display in real time. The circuit consists of multiple seven-segment displays controlled by the Arduino to show the current time accurately. This project demonstrates key concepts in digital electronics, timing mechanisms, and microcontroller based display control.

2 Components Used

- Arduino Uno
- Seven-segment displays - 6
- Resistors - 6
- Push buttons
- Connecting wires
- Breadboard

3 Circuit Design

The circuit consists of six 7-segment displays connected to a single 7447 decoder, with common anodes controlled by Arduino through multiplexing. The BCD inputs (A, B, C, D) of the 7447 are connected to the Arduino, and the decoder output drives the corresponding segments of the displays. Common anodes are activated sequentially using Arduino digital output pins.

Each digit of the clock (hours, minutes, and seconds) is displayed using two 7-segment displays, forming a total of six displays. To efficiently control all six displays while minimizing the number of required I/O pins, the circuit employs a multiplexing

technique. The Arduino quickly switches between activating each digit and updating the corresponding BCD values.

4 Working Principle

The 7447 BCD-to-7-segment decoder simplifies the control mechanism by converting 4-bit BCD inputs from Arduino into the respective segment activations needed to display decimal digits (0-9). Since the 7447 is a common-anode decoder, the segment outputs are active-low, meaning that the segments light up when a low signal is received.

Each 7-segment display has its common anode connected to an Arduino digital pin, which is toggled high one at a time while updating the BCD values. Resistors are placed in series with each segment to limit current flow and prevent damage to display components.

- The Arduino tracks time in hours, minutes, and seconds using the millis() function.
- A multiplexing technique is used to cycle through the six 7-segment displays rapidly, making them appear continuously lit to the human eye.

5 Code explanation

This project involves the development of a digital clock using an Arduino and seven-segment displays. The primary objective is to create a clock that counts the hours, minutes, and seconds from a predefined starting time. The timekeeping mechanism is implemented entirely through software-based timing using micro-controller interrupts.

The clock utilizes six seven-segment displays to show time in HH:MM:SS format. To minimize the number of required micro-controller pins, the project employs a multiplexing technique that allows control of all six digits with fewer connections.

5.1 Display Multiplexing and Updating Mechanism

Since the micro-controller has a limited number of output pins, multiplexing is used to control all six displays efficiently. Rather than lighting up all six digits simultaneously, each digit is turned on one at a time in a rapid cycle, which occurs so fast that the human eye perceives them as always being on.

The steps for updating the display are as follows:

- The hours tens digit is activated, and its corresponding value is sent to the decoder.
- This process is repeated for minutes tens, minutes units, seconds tens, and seconds units.
- The cycle continues indefinitely, giving the illusion of a fully illuminated display.

6 Results and Observations

6.1 Clock Performance

The implemented digital clock successfully tracks time in the HH:MM:SS format. The use of timer interrupts ensures that the clock maintains a stable and accurate count. The multiplexed display updates efficiently, making the time continuously visible without noticeable flickering.

6.2 Accuracy and Stability

The clock was tested for an extended period to assess its precision. The timer based approach effectively maintained synchronization with real-world time, with minimal deviation. The primary source of inaccuracy would be minor drift caused by micro-controller clock variations, but these deviations are negligible for short-term applications.

6.3 Power Consumption

Since the display remains active at all times, power consumption is a factor to consider. Using low-power modes or optimizing the display refresh rate could help reduce energy usage. However, in its current form, the power consumption remains within acceptable limits for typical micro-controller based applications.

6.4 Observations

- The clock displays the correct time and increments each second without noticeable lag.
- The multiplexing method effectively reduces the number of micro-controller pins required.
- The display appears stable, with no visible flickering.
- Minor time drift could occur over long periods due to the internal clock variations of the microcontroller.

7 Conclusion

This project successfully implements a digital clock using an Arduino and seven-segment displays. The clock accurately tracks time in HH:MM:SS format, utilizing binary-coded decimal (BCD) storage and a multiplexing display technique to optimize pin usage. The system employs timer interrupts to ensure precise second-by-second timekeeping, updating the display dynamically.

Overall, this project demonstrates the integration of hardware and software-based timing mechanisms to develop a functional digital clock. The structured approach used in display multiplexing, BCD-based storage, and interrupt-driven updates ensures an efficient and scalable design. Future improvements could include additional features such as manual time setting, an alarm system, or low-power optimizations to enhance usability.