Clock using Binary Logic in Arduino using AVR-GCC

EE24BTECH11001

March 24, 2025

Overview

Features

The main features of the calculator are,

- The entire code was written in AVR GCC.
- Instead of using **inc** command, k-maps were used.
- The use of one decoder for each 7-segment display was avoided (only one decoder was used).

Hardware

The hardware consists of:

Quantity	**Component**
6	7 Segment Display
1	Arduino Uno
-	Wires
1	Decoder (7447)

Table 1: Materials Required

- Decrease requirement of multiple decoders, by intelligentally connecting the circuit as shown in the schematic and using **software multiplexing** Figure **??**.

The schematic for connections is as shown below,

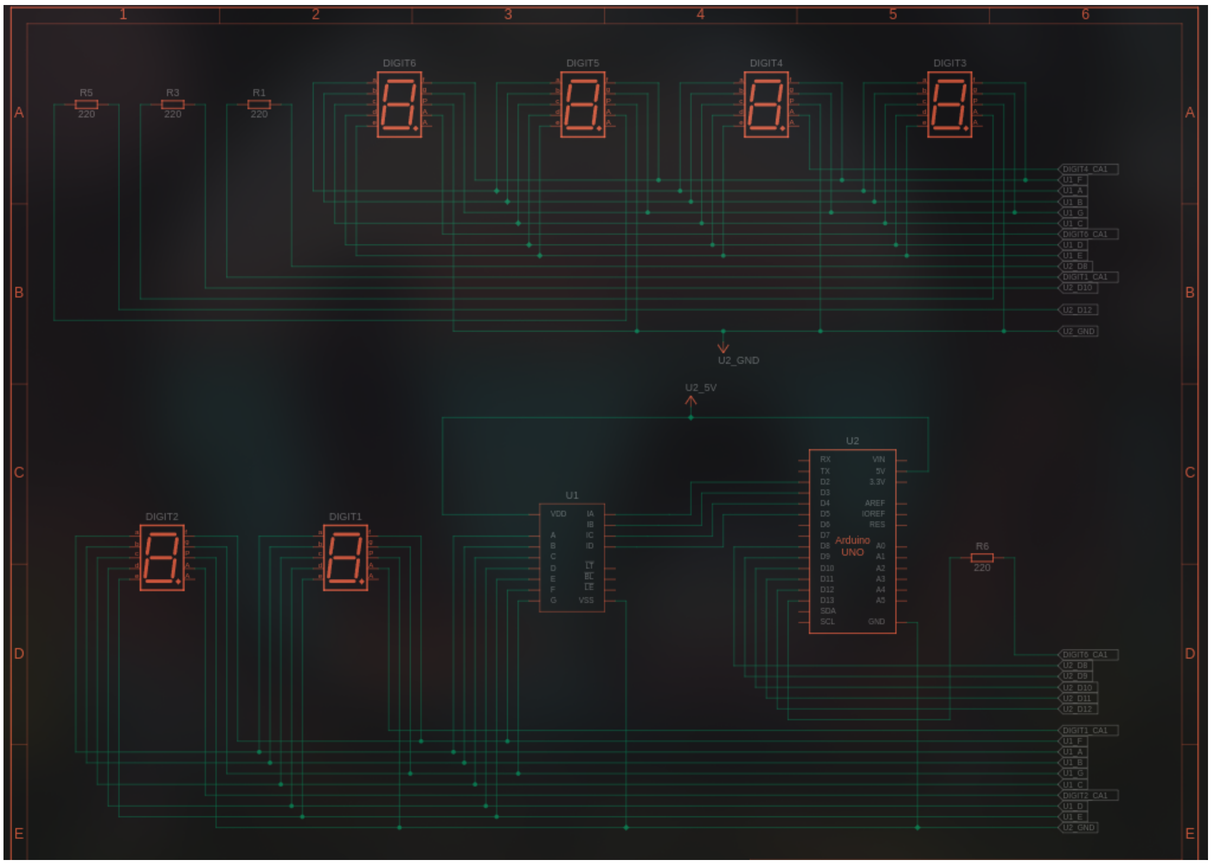


Figure 1: Schematic of Circuit.

Software Overview

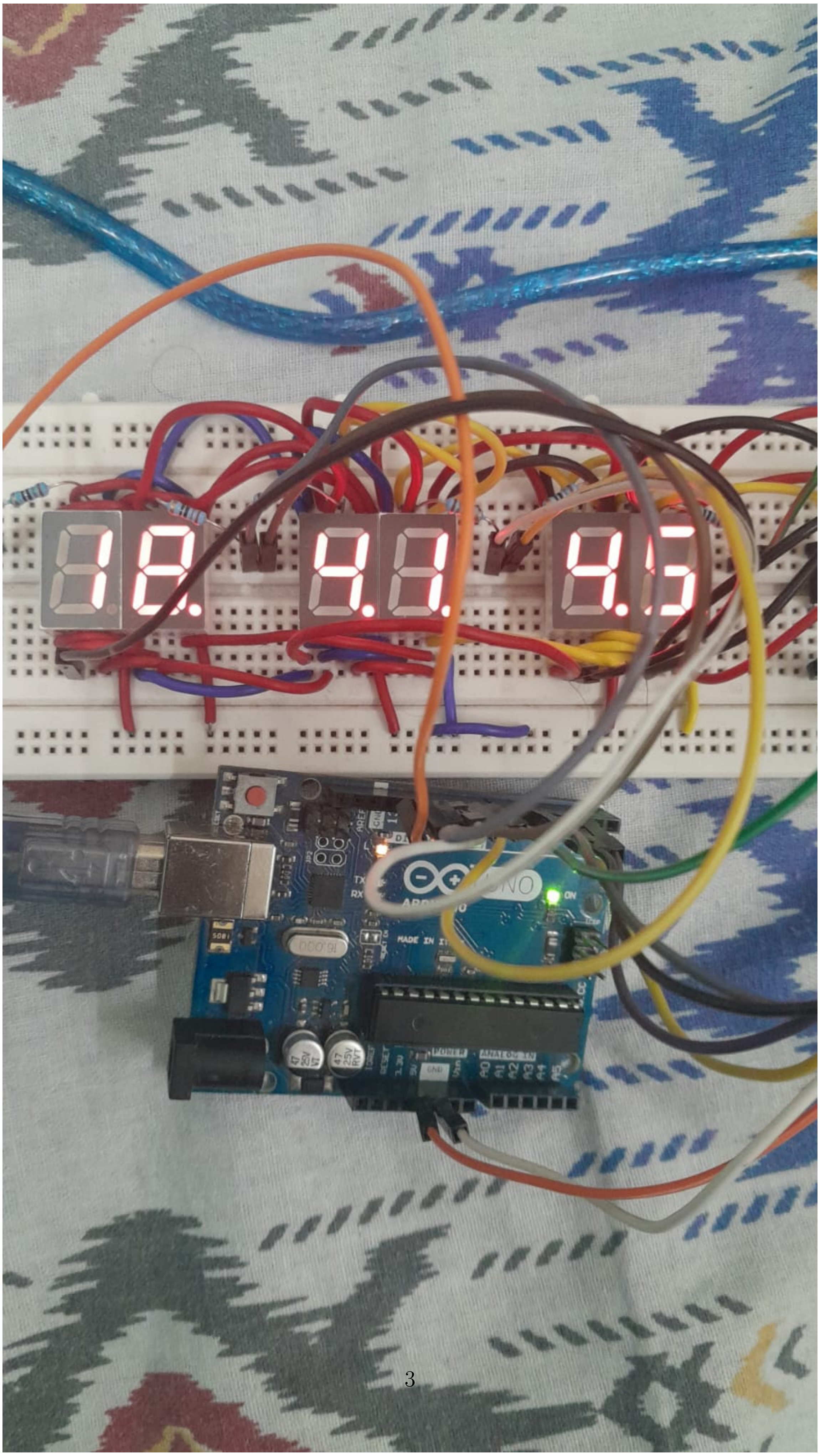
The software is implemented using AVR-GCC. The software trick used is called **software multiplexing**. The individual 7-segment-displays are connected to digital pins. The digital pins are cycled through at a very high rate (1 ms between two digitalWrite calls), turning only 1 on at a time. At an appropriate delay of 1 second (realized using the Arduino's Counter), all the values to be displayed are recalculated. This achieves the effect of a stationary time which updates exactly at 1 second.

Timer

Using multiple loops, a timer of one second was created. Within that one second, time is displayed (using the logic mentioned above), and at the end of one second, one's digit is incremented. The sub routine that does this has a piece of code at the end to increment ten's digit of seconds if incremented value of one's digit is 0. Similar the function to increment ten's digit has a line of code to increment one's digit of minutes if value after incrementing is 0. This is how time is incremented

Timer Initialization and Interrupt Handling

The clock relies on precise timing through Timer1 and interrupt service routines:



```
void init_timer() {
    // Set CTC mode (Clear Timer on Compare Match)
    TCCR1B |= (1 << WGM12);

    // Set prescaler to 1024
    TCCR1B |= (1 << CS12) | (1 << CS10);

    // Set compare match value for 1s intervals (16MHz/1024)
    OCR1A = 15624;

    // Enable Timer1 compare match interrupt
    TIMSK1 |= (1 << OCIE1A);

    // Enable global interrupts
    sei();
}

// Interrupt Service Routine triggered every second
ISR(TIMER1_COMPA_vect) {
    flag = 1;  // Set flag to update clock values
}
```

The timer operates in CTC mode with a 1024 prescaler, generating an interrupt every second. When the interrupt occurs, the ISR sets a flag that triggers the time update in the main loop, ensuring accurate timekeeping without blocking program execution.

This uses the Arduino's internal clock to achieve very minimal time losses.

Conclusion

We achieved a fully functioning 24-hour clock using minimal number of BCD decoders by using software multiplexing in AVR-GCC using Arduino microcontroller.