TED UNIVERSITY

2023 Fall

CMPE 453 Embedded Systems

LAB REPORT # 6

Lab Name: Adjustable Clock Design By Using Timers

Name: Erkan Sancak

Section: 2

Instructor's Name: Prof. Saiful Islam

I. Tasks

In this lab, the objective was to design a real-time clock using an Arduino-Uno and various components. The initial task involved displaying the current time on an RGB LCD display using Timer1 of the AVR Microcontroller.

1.) Initialization of Components

- The RGB LCD display and other necessary components, including LEDs and a push button, were initialized in the setup phase.
- The LCD display was configured to have 16 columns and 2 *rows (lcd.begin(16, 2)).*
- The RGB color values for the LCD were set to achieve an initial color of (0, 50, 50) for Red, Green, and Blue channels, *(lcd.setRGB(colorR, colorG, colorB)).*

2.) Displaying Initial Time

- The initial time values, including the hour, minute, and seconds, were pre-set. The formatted time string *(String(hour) + ":" + String(minute) + ":" + String(seconds))* was then printed on the LCD display.

3.) Using Push Button for Hour and Minute Adjustment

- Utilizing a push button to adjust the hour and minute values dynamically. This was achieved by implementing logic that responded to button presses.

4.) AVR Listening to UART and Getting User Input

- Arduino Uno was configured to listen to UART communication for user input. The Serial communication was utilized to receive new hour and minute values from the user.

5.) Setting Hour or Minute Based on Button Presses

- The program was designed to set the hour or minute based on the number of button presses within a specified time frame. LED indicators were used to provide feedback during this process.

6.) Handling Invalid Inputs

- If the user entered hour or minute values outside the valid range, appropriate prompts were given

II. Hardware Implementation

To implement the push button functionality and facilitate user interaction for adjusting the real-time clock, the following hardware connections were established

Arduino Uno and Base Shield

The Arduino Uno board, serving as the core of the system, was utilized for processing and control. The Arduino Base Shield was connected to the Arduino Uno, simplifying the interfacing of additional components.

Breadboard Integration

The breadboard providing a platform for organizing and connecting various components. It facilitated the creation of an efficient circuit layout, enabling easy connections and troubleshooting.

Push Button Configuration

The push button serves as a user interface element to facilitate dynamic adjustments to the hour and minute values displayed on the RGB LCD. One leg of the push button was connected to the ground (GND) pin on the Arduino Uno, while the other leg was connected to DIGITAL PIN 2. This configuration allows the Arduino to detect changes in the button state.

LED Indicators

Two LEDs were utilized as visual indicators. One LED was connected to digital pin 8, and the other to digital pin 9. These LEDs served to signal whether the hour or minute values were being adjusted by the user.

Connecting Wires

Connecting wires were used on the breadboard and Arduino Uno to establish connections between components.

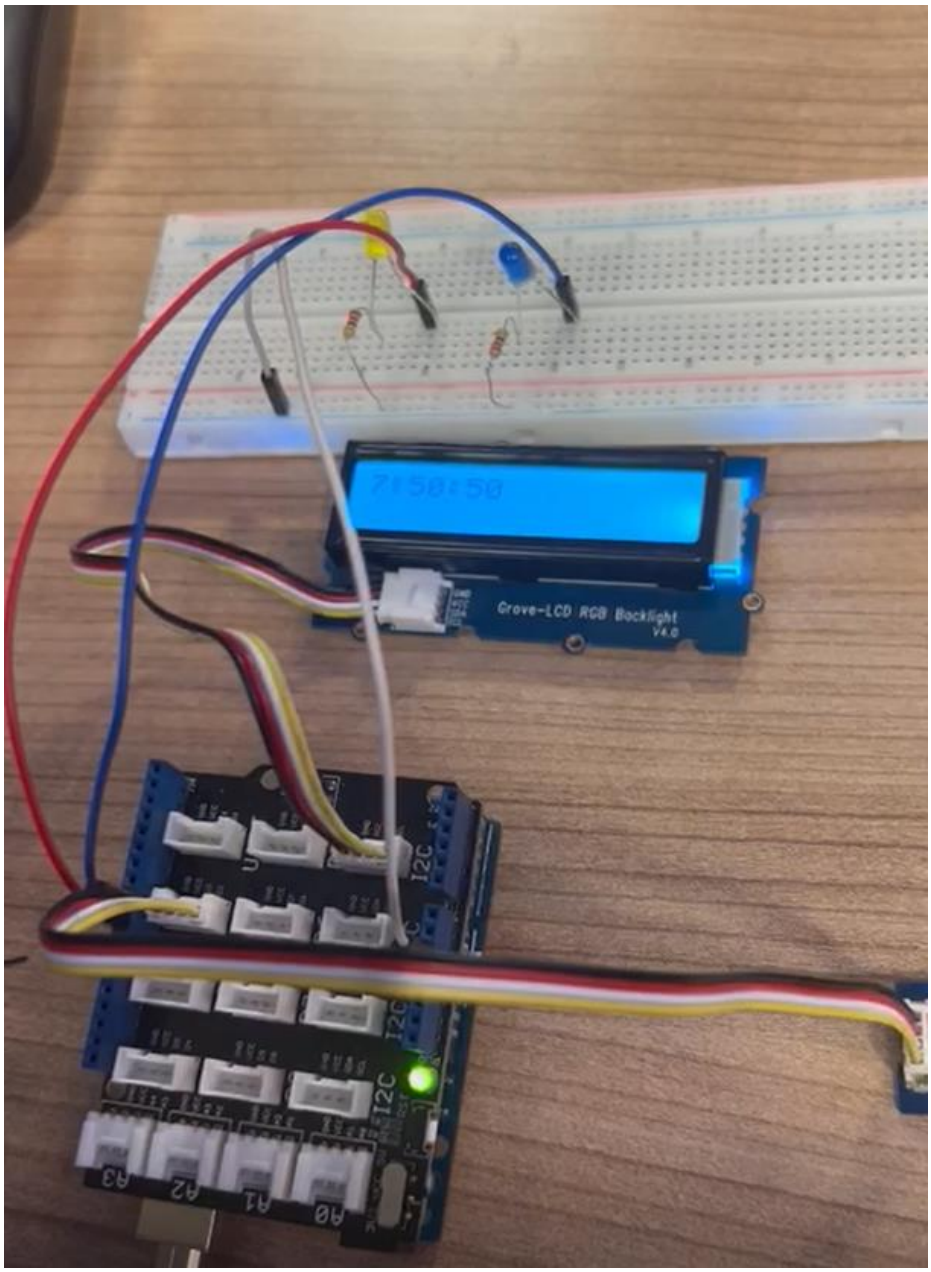

Figure 1 – Arduino, Arduino base shield, Push Button, Breadboard, LCD Display and LEDs with Resistors with the necessary connections provided and working as desired.(December 11, 2023 Lab-6)

III. Code

Initialization

The code begins by initializing critical components. This includes setting up the RGB LCD display for output and establishing Serial communication for interaction with the user. The ***setup ()*** function is responsible for the initial setup of the Arduino. It initializes the RGB LCD display, establishes Serial communication, sets the initial color for the display, and prints the initial time on the LCD.

Button Press and User Input

The code continuously monitors the state of a push button. The code checks for button presses using ***digitalRead(2).*** If a button press is detected, the period_counter is incremented.

If the time since the last button press ***(millis() - current)*** exceeds 2000 milliseconds (2 seconds), the code enters a block to handle user input. Depending on the value of period_counter, the program adjusts either the hour or minute values based on the input received through Serial communication (single press for hour, double press for minutes).

LED Indication

LED indicators (connected to pins 8 and 9) are illuminated (for 1 second) to provide visual feedback, indicating whether the hour or minute values are being adjusted.

Updating the Time

The real-time clock is updated every second by incrementing the seconds with using ***((millis() - current) % 1000 == 0).*** If seconds exceed 59, the minute value is increased, and similarly, if the minute exceeds 59, the hour is incremented. Then, the LCD display is cleared, and the updated time is printed, reflecting any changes made by the user through button presses and UART input. The display reflects the modified time in the format HH:MM:SS.

IV. Critical Analysis / Conclusion

To sum up, this lab successfully built and tested an interactive real-time clock adjustment system. The hardware components, including Arduino Uno, Base Shield, push button, LEDs, and RGB LCD display, seamlessly integrated with the software code. This lab provided a practical hands-on experience in embedded systems, emphasizing fundamental design principles and user interaction.