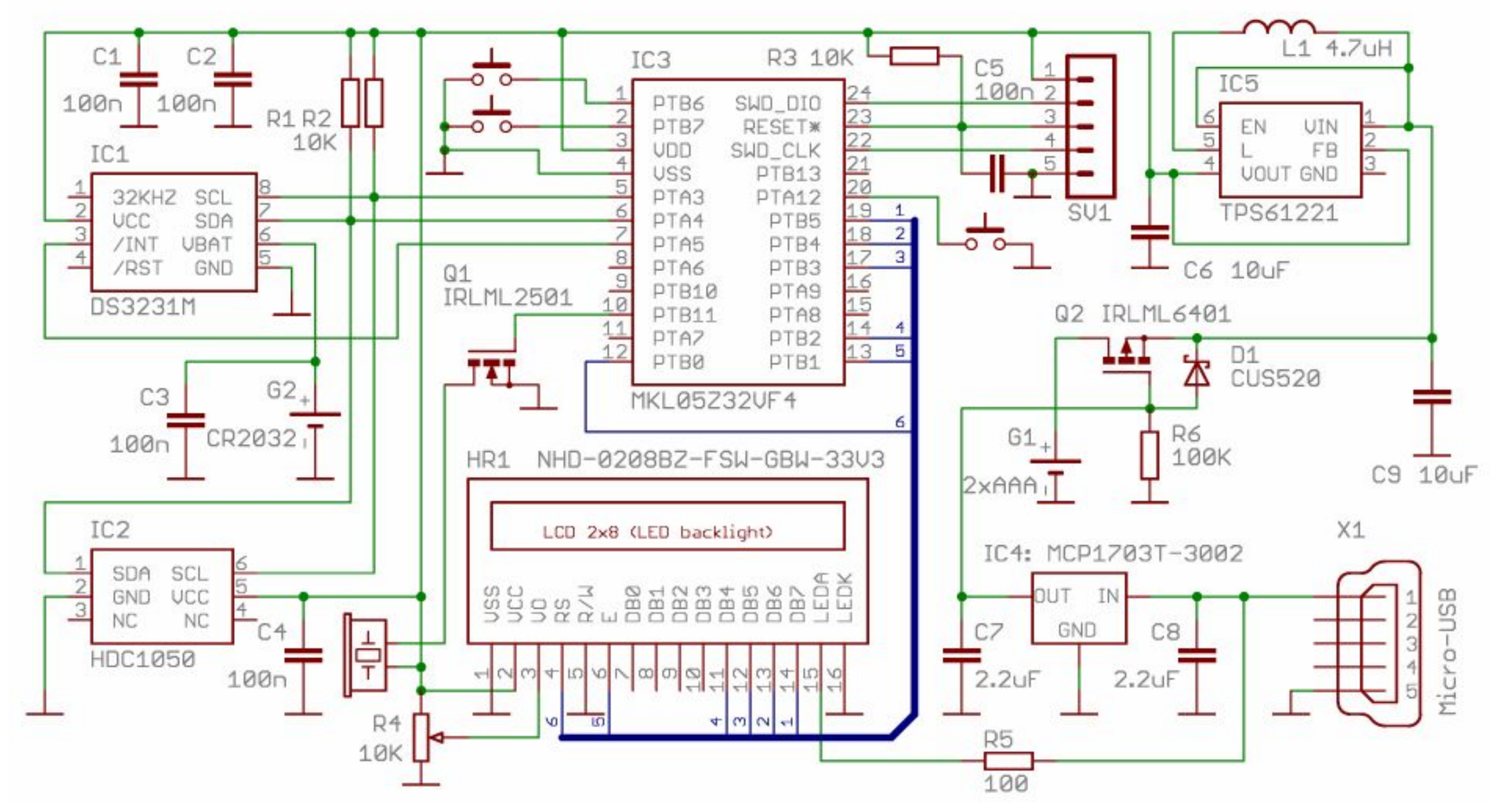
Digital Alarm Clock

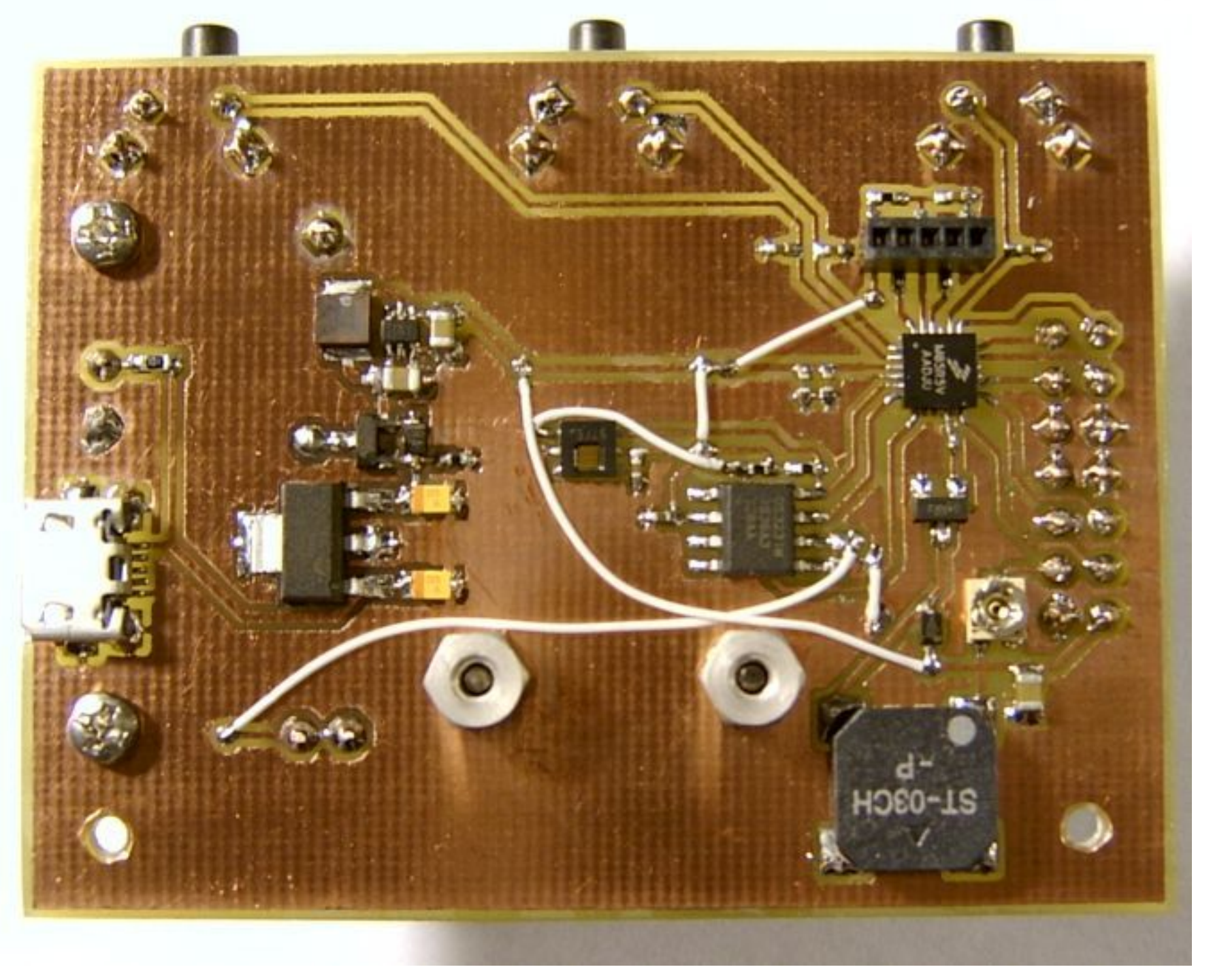
Regina Rex , Dept. of Math and Computer Science
Mentor: Sergei Bezroukov, Dept. of Math and Computer Science
University of Wisconsin – Superior

Hardware

The device is based on Kinetis ARM 48MHz Cortex-M0+ microcontroller from NXP because it has good low power mode capability and DS3231MZ - a real-time clock with the Inter-Integrated Circuit Interface (I2C). The RTC chip is powered by a separate CR2032 battery to keep the time updated when the alarm clock is turned off. For the display, I used a 2x8 character Newhaven LCD with backlight. The Texas Instrument HDC1050 sensor, which is highly accurate and also has low power capability, is used to measure temperature and humidity. A micro-USB connector or a block of two AAA batteries can be used to power the alarm clock.



Schematic



Assembly

Loading Code

The system was programmed in C-language and developed in Keil µVision integrated development environment (IDE). The compiled code was loaded into the microcontroller using Segger J-Link ARM Programmer.

Overview

The purpose of this project is to design and program a system using a microcontroller. For this project, I chose to design an alarm clock that displays time along with air temperature and humidity. It uses low power mode to save power and maximize battery life. The clock can also be used to set an alarm and the time can be adjusted. It can be powered using a battery or a micro-USB cable. The clock comes with a snooze function, and the alarm can be turned on/off whenever the user pleases. The top line of the LCD displays the time in hh:mm format followed by A or P with indicates AM or PM and the symbol (ö) which means the alarm is on. The bottom line of the LCD shows the temperature (°C) and humidity (%), or the clock state.

Digital Alarm Clock (Back)

Digital Alarm Clock (Front)

Firmware

The program uses a Finite State Machine to implement all clock functions, such as setting current time and alarm time and activating and deactivating the alarm. First, the program configures the hardware and runs through the setup routine. When the main loop runs, it checks the buttons to see if they are pressed and performs the function based on the current clock state. It also puts the microcontroller to sleep and wakes it up every 20 milliseconds to save power and efficiently use the battery life. The program also includes code for accurate the button debouncing. In state 0, at the end of each loop, if the *alarmOn* variable is equal to 1 (meaning the alarm is on), the current time is compared with the alarm time, and if the alarm time is equal to the current time, the buzzer starts to beep. The buzzer beeps and stops every 20 milliseconds to produce a repetitive sound. The snooze button (Button 3) is used o turn stop the buzzer from making noise when the Alarm goes on.

Finite State Machine Design

There are five possible states in the finite state machine design of the Digital Alarm Clock. The buttons on the alarm clock are used to change the clock states, set the clock and alarm time, and to switch the alarm on/off. These are the alarm clock buttons and their functionalities:

- Button 1 (B1 - The leftmost button): This button is used to change the clock's state
- Button 2 (B2 - The middle button): This button is used to increase hours and minutes
- Button 3 (B3 - The rightmost button): This button changes the clock mode to state 0. It also turns the alarm on if it is off and vice versa

Below is a table showing the Finite State Machine Design.

State 0 (Display Time)	**State 1 (Set Hours)**	**State 2 (Set Minutes)**	**State 3 (Set Alarm Hours)**	**State 4 (Set Alarm Minutes)**
B1 Pressed: - To state 1 - Blink hours	B1 Pressed: - To state 2 - Blink minutes	B1 Pressed: - To state 3 - Blink alarm hours	B1 Pressed: - To state 4 - Blink minutes	B1 Pressed: - To state 3 - Blink hours
B1 and B3 Pressed: - To State 3 - Blink hours	B2 Pressed: - Increase hours - Blink hours	B2 Pressed: - Increase minutes - Blink minutes	B2 Pressed: - Increase alarm hours - Blink alarm hours	B2 Pressed: - Increase alarm minutes - Blink alarm minutes
B3 Pressed: - Turns alarm on / off	B3 Pressed: - To state 0 - No blink	B3 Pressed: - To state 0 - No blink	B3 Pressed: - To state 0 - No blink	B3 Pressed: - To state 0 - No blink

Table of States

Learning Outcome

The Digital Alarm Clock was my first microcontroller project. I learned how to design and program a system using a microcontroller. And how to optimize the battery life of a microcontroller based project by putting the microcontroller in a state of low power. I still use the alarm clock to ensure I wake up on time for my classes. I also gained hardware skills that would be useful for a career path as an Embedded Software Engineer.