Dawson College

Electrical Engineering Technology Department

Introduction to Internet of Things

Project Name:

Smart Real-Time Alarm Clock

Team Members:

Kaiser Deen D. Tayone (ID: 2433966), Duke Wang (ID: 2433435)

May 13, 2025

2. Project Description

Many people tend to oversleep after a very busy day, especially students or workers with irregular schedules. Relying solely on a phone alarm can be risky since the phone could end up being on mute the whole time or that the device has run out of battery.

To solve this issue, we came up with the idea of creating a homemade Smart Real-Time Alarm Clock using an RTC module, LCD, keypad, and buzzer. This Real-Time Clock Alarm Interface allows the user to set an alarm time directly using the keypad, and when the time matches, a buzzer sounds, and a visual alert is shown on the screen. It's simple yet effective and can be useful for getting up on time and managing a daily routine.

How it works:

- The RTC (Real-Time Clock) module tracks time accurately.
- The LCD displays the current time and countdown to the alarm.
- Pressing * activates alarm setup mode via the keypad.
- When the alarm time is reached, the system activates a buzzer and flashing screen message.
- Pressing # stops the alarm.

Final assembly diagram:

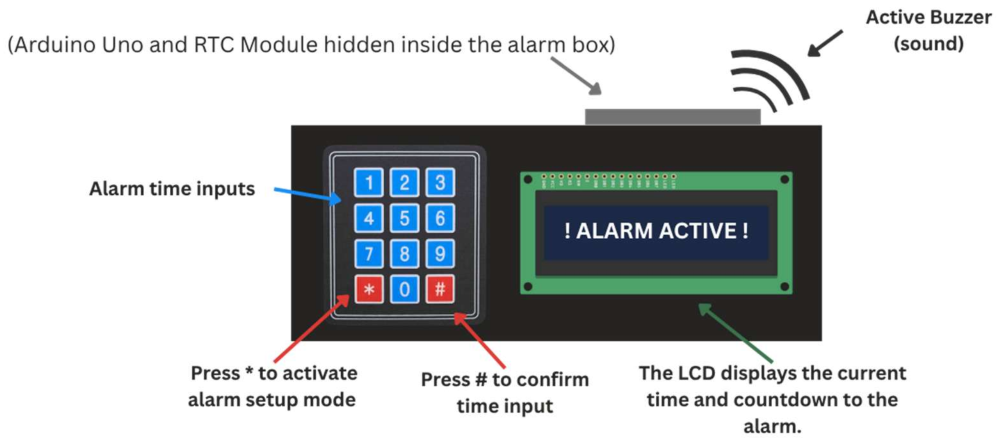
(Arduino Uno and RTC Module hidden inside the alarm box)
Active Buzzer (sound)
Alarm time inputs
1 2 3 4 5 6 7 8 9 * 0 #
! ALARM ACTIVE !
Press * to activate alarm setup mode
Press # to confirm time input
The LCD displays the current time and countdown to the alarm.

Figure 1: Smart Real-Time Alarm Clock Diagram

3. Circuit Diagram & Components

Inputs:

- **3x4 Keypad Module** (user sets alarm time)
- **RTC Module** (DS-1307) (real-time clock tracking)

Outputs:

- **LCD 16x2 Display** *(shows time, alarm, and messages)*
- **Active Buzzer** (A1 pin / auditory alarm)

Table 1: Circuit Connections/Wiring

Symbol	Part	Arduino Uno Pin	Notes
KPD	3x4 Keypad (input)	Rows → 13, 6, 5, 4 Columns → 3, 2, A0	Reads 4-digit input for the alarm (typed by the user). (Keypad Library)
RTC	RTC Module (DS-1307)	SDA → A4 (data), SCL → A5 (clock) VCC to 5V, GND to GND	No pinMode needed (uses I2C protocol) Powered by 5V and GND. (RTClib library)
BZ1	Active Buzzer	Positive → A1 Negative → GND	Digital output buzzer triggered by alarm. Use active buzzer.
LCD	16x2 LCD Display	RS→7, EN→8, D4–D7→9–12. VSS & RW to GND, VDD to 5V, V0 to GND (1kΩ). Backlight: A to 5V (330Ω), K to GND.	LCD uses LiquidCrystal library.

(Note: All GND connections are bused together and connected to Arduino GND. The 5V rail is also bused to power the LCD and RTC)

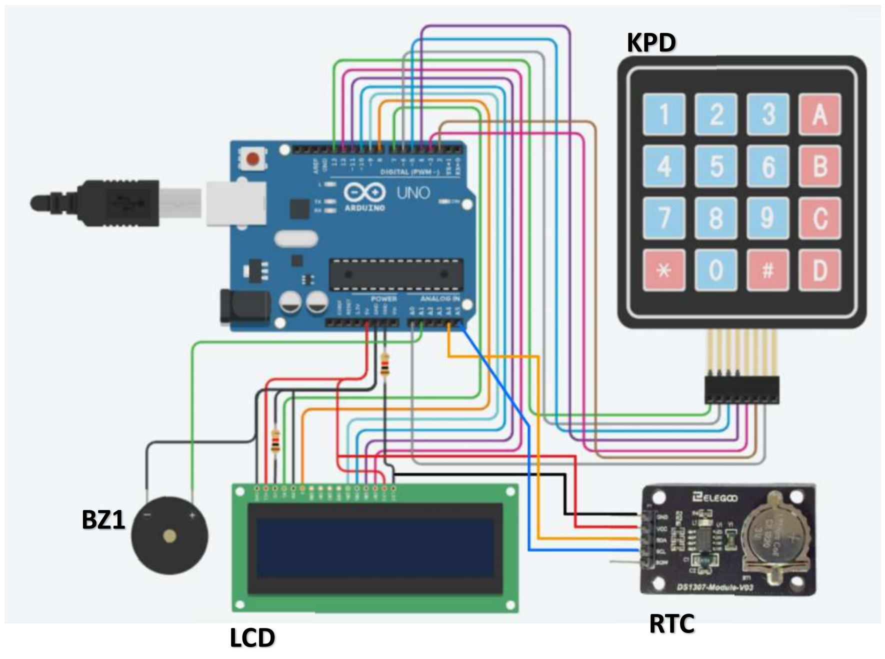

Figure 2: Full Wiring Diagram of the Smart Alarm Clock System

3.1 Alarm Workflow and Display Logic

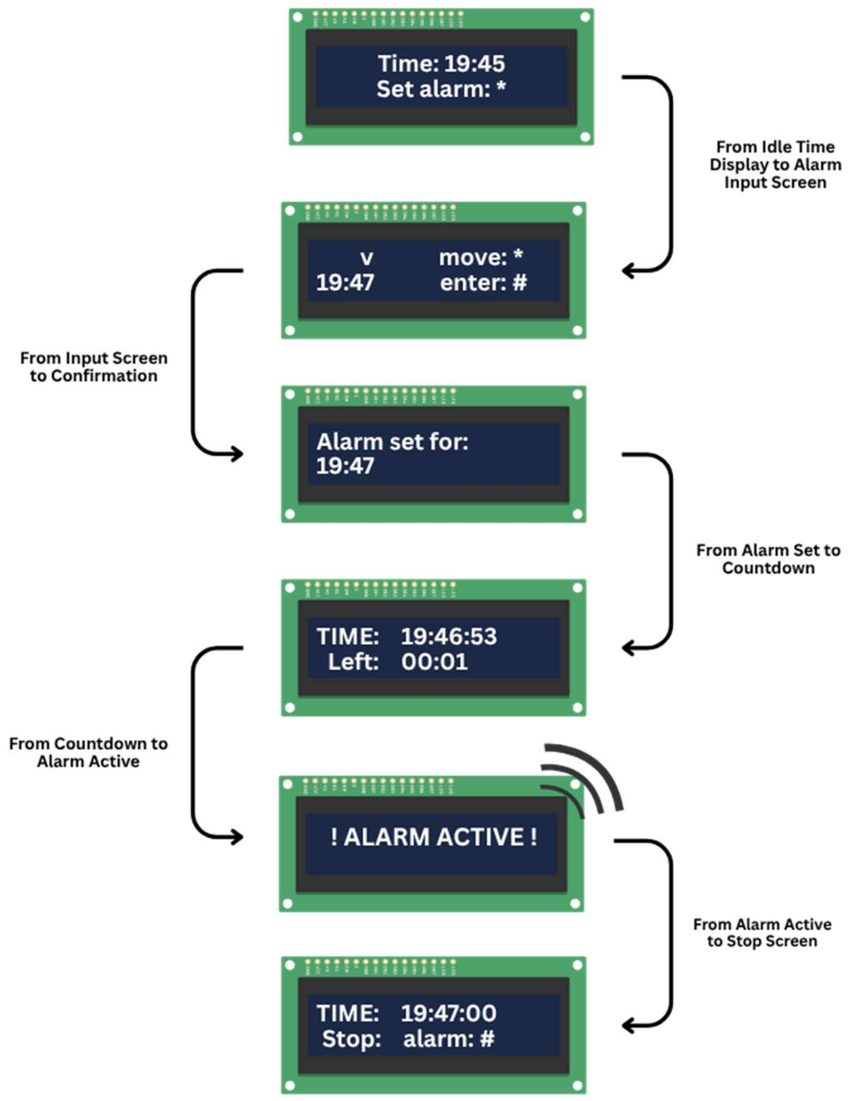
Time: 19:45
Set alarm: *
From Idle Time Display to Alarm Input Screen
v move: *
19:47 enter: #
From Input Screen to Confirmation
Alarm set for:
19:47
From Alarm Set to Countdown
TIME: 19:46:53
Left: 00:01
From Countdown to Alarm Active
! ALARM ACTIVE !
From Alarm Active to Stop Screen
TIME: 19:47:00
Stop: alarm: #

Figure 3: Alarm System Display Phases

This diagram shows the main LCD display phases of the alarm clock system: starting from idle mode, through alarm input and countdown, to activation and manual stop. (Used Canva to construct this diagram)

4. Code documentation

Table 2: GLOBAL CONSTANTS AND PINS.

Constant / Pin	Purpose	Value / Pin(s)
lcd (RS, E, D4–D7)	LCD screen connection pins	7, 8, 9, 10, 11, 12
ROWS (Keypad)	Number of rows in the keypad matrix	4
COLS (Keypad)	Number of columns in the keypad matrix	3
rowPins (Keypad)	Arduino pins connected to keypad rows	{13, 6, 5, 4}
colPins (Keypad)	Arduino pins connected to keypad columns	{3, 2, A0}
buzzerPin	Arduino pin for the buzzer alert	A1
interval (Blinking)	Millisecond interval for LCD blinking effect	1000

4.1 Libraries Used

- **RTClib.h:** Handles communication with the DS1307 RTC module to get and keep accurate time.
- **LiquidCrystal.h:** Controls the 16x2 LCD display for showing the current time, countdown, and alarm messages.
- **Keypad.h:** Enables input from the 4x3 keypad, allowing the user to enter and confirm the alarm time.

4.2 Functions & Code Structure

- **void setup():** Initializes the LCD, serial communication, buzzer pin, and the RTC. Sets RTC time to compile time on first upload.
- **void loop():** Continuously gets current time from RTC and calls ShowTime(). Manages alarm logic:
  - If alarm is set and matches current time, alarmTriggered becomes true.
  - While alarmTriggered: displays blinking alert, calls ringBuzzer(), and checks for '#' to stop alarm.
  - If alarm is set but not triggered, displays countdown.
  - If no alarm set, prompts user to set one ('*'). Listens for '*' key press to call setAlarm(). Ensures buzzer is off if alarm not active.
- **void ShowTime(int h, int m, int s):** Displays the provided hour, minute, and second on the LCD's first line
- **void setAlarm():** Provides an interface on the LCD for the user to input alarm hour and minutes using the keypad. Uses '*' to move cursor, '#' to confirm. Validates input and stores it, then confirms on LCD.
- **void ringBuzzer():** Activates the buzzer in a distinct pattern (on-off-on).
- **void movecursor():** Helper function for setAlarm() to position the input cursor ('v') on the LCD, skipping the colon.

4.3 Alarm Clock Code Explanation

Here is the LINK to the GitHub Repo: DrDuky/clock-alarm-pbl

(The following section explains how the code works.)

1. Continuous Time Display – In each loop() cycle, the current time is read from the RTC_DS1307 module. This time is then immediately formatted and displayed on the first line of the LCD using the ShowTime() function, ensuring an always-up-to-date time anzeige.

2. Alarm Setting Interface – When the user presses the '*' key, the setAlarm() function is called. This function guides the user through an on-screen LCD menu to input the desired alarm hour and minute using the keypad. The input is validated (e.g., for correct hour/minute ranges), and upon confirmation with the '#' key

3. Alarm Monitoring – The main loop() continuously compares the current time (hour and minute) with the stored alarmHour and alarmMinute, but only if an alarm has been alarmSet and has not yet been alarmTriggered.

4. Alarm Activation & Alert – If the current time matches the set alarm time, the alarmTriggered boolean value is set to true. This initiates the alert sequence.

- A blinking warning message (e.g., "! ALARM ACTIVE !") appears on the LCD.
- The ringBuzzer() function is called repeatedly to generate an audible alarm.
- A prompt to press '#' to stop the alarm is also shown.

5. Silencing the Alarm – When the alarm is active (alarmTriggered is true), pressing the '#' key on the keypad will set alarmTriggered back to false. This stops the ringBuzzer() calls and clears the alert message from the LCD, returning the clock to its normal time display mode.

6. Time-to-Alarm Countdown – If an alarm is currently alarmSet but not yet alarmTriggered, the loop() calculates the remaining time until the alarm. This countdown (in hours and minutes) is displayed on the second line of the LCD, providing the user with an indication of when the alarm will sound.

5. Ethics, Privacy, or Security Disclaimer

Thinking about our alarm clock in the right way, its main purpose is to make a sound at the set time. It's important to understand that this clock is a helpful tool, not a perfect solution if you absolutely can't afford to miss something. Always double-check that it is set up correctly.
Disclaimer: This reminder encourages the fair and responsible use of a basic alarm clock.

References:

https://www.canva.com/ [Used to create the project diagram -figure 1]
https://www.tinkercad.com/ [Used to create the circuit diagram -figure 2]
Elegoo Ds1307 Module V03 – Digilog.pk [For the RTC png image in the circuit diagram -figure 2]
https://docs.arduino.cc/libraries/liquidcrystal/
https://docs.arduino.cc/libraries/rtclib/#Compatibility [repository]
https://docs.arduino.cc/libraries/keypad/ [repository]
https://docs.arduino.cc/built-in-examples/digital/BlinkWithoutDelay/
How to interface I2C LCD display with Arduino ? | GeeksforGeeks

Our Project GitHub:

GitHub - DrDuky/clock-alarm-pbl