Digital Clock & Alarm System on Wio Terminal

Tanner Roberson

Requirements (Part 1)

- LCD displays running digital clock (HH:MM:SS, 24-hour format)
- Display centered on screen, text size = 6
- LCD initialized with: Background = green, Text color = black
- Blue LED blinks every second change

Requirements (Part 2)

- Push Buttons in Clock Mode:
  - WIO_KEY_B → Increase hour (00 after 23)
  - WIO_KEY_A → Increase minute (00 after 59, no hour rollover)
- Alarm Mode:
  - WIO_KEY_C → Switch between clock mode and alarm mode
  - Alarm time displayed as HH:MM
  - WIO_KEY_B → Increase alarm hour
  - WIO_KEY_A → Increase alarm minute
- Continuous beep when clock = alarm time (stops when C pressed)

Requirements (Part 3)

- Beep generated when hour/minute is changed
- LCD backlight OFF when device still for 5s
- LCD backlight ON when tilted (~50° rotation)
- No delay loops allowed
- All push button inputs must be interrupt-based

Learning Objectives

- Combine concepts from past labs into one project
- Implement multi-function button inputs depending on state
- Synchronize code functions (e.g., LED blink with seconds)

General Steps

- Display clock in HH:MM:SS format
- Allow alarm creation (HH:MM)
- Enable clock time adjustment via push buttons
- Use accelerometer to detect inactivity and manage backlight
- Maintain correct time and LED blinking even in alarm mode or backlight OFF

Detailed Steps

- Format hours/minutes/seconds with leading zeros
- Implement increment/rollover for hours (23 → 00) and minutes (59 → 00)
- Switch to alarm mode and adjust alarm time with buttons
- Ensure clock continues running & LED blinking in all states
- Configure accelerometer for:
- - 5s inactivity → backlight OFF
- - Tilt (~50°) → backlight ON
- Generate short buzz on A/B key press

Observations

- Correct state management of the clock/alarm is crucial
- Refreshing LCD too often causes unreadable display
- Alarm & clock must stay synchronized in all states

Summary

- Configured Wio Terminal as a digital clock + alarm system
- Used interrupt-based inputs for buttons
- Avoided delay loops for accurate timing
- Practiced managing device states effectively