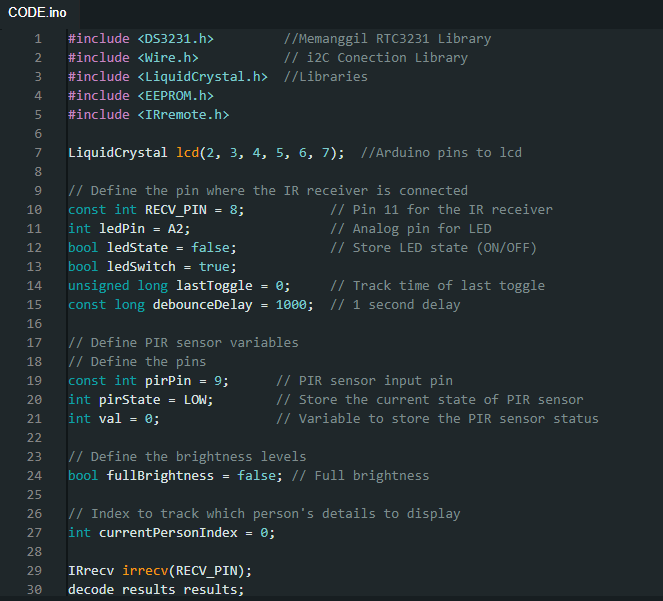
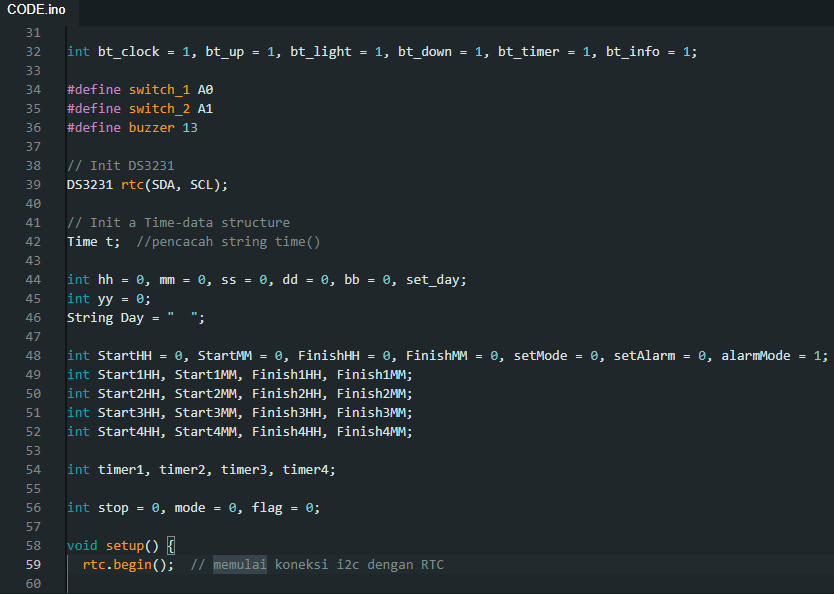
**CODE**

The code provided is a comprehensive program written for an Arduino microcontroller to manage a clock, an alarm system, a PIR (Passive Infrared) motion sensor, and LED control using an IR remote. It uses various libraries and hardware components such as:



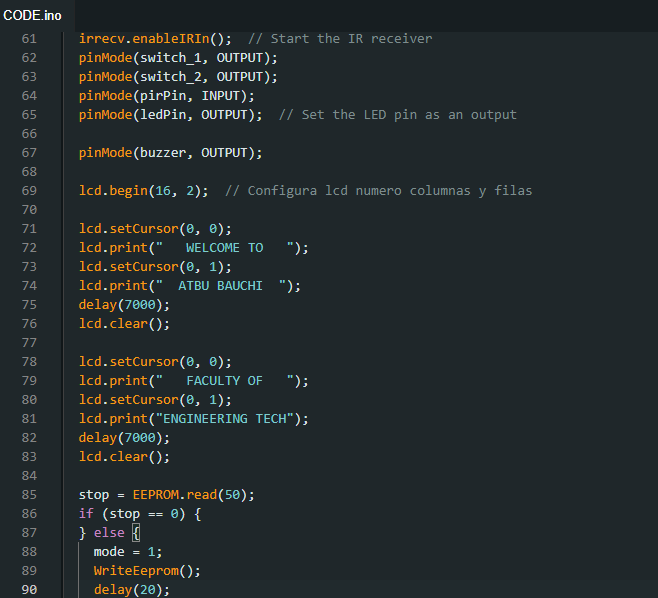
**1. Libraries Used:**

* **DS3231.h**: To interface with the DS3231 Real-Time Clock (RTC) module, which provides accurate timekeeping.
* **Wire.h**: For I2C communication, which is used by the RTC module.
* **LiquidCrystal.h**: To control an LCD display connected to the Arduino.
* **EEPROM.h**: To store and retrieve data in the Arduino’s EEPROM memory.
* **IRremote.h**: To decode signals from an infrared (IR) remote control.



**2. Hardware Components:**

* **LCD Display**: Connected to the Arduino to show the time, date, and other information.
* **DS3231 RTC Module**: For real-time clockkeeping (hours, minutes, seconds, day, month, and year).
* **PIR Sensor**: Detects motion and triggers actions like dimming or controlling the LEDs.
* **IR Receiver**: Captures signals from an IR remote control to adjust settings like time, alarm, and LEDs.
* **LED**: Connected to an analog pin for brightness control, triggered by the timer or the PIR sensor.
* **Buttons and IR Remote**: Used to set the time, control alarms, and manage other configurations.
* **Buzzer**: Provides sound feedback when buttons or IR remote commands are received



### 3. ****Core Features:****

### ****Time and Date Display:****

### The program uses the DS3231 RTC module to get the current time and date.

* The time is displayed in the format "HH:MM" on the first line of the LCD.
* The date is displayed in the format "DD/MM/YYYY" on the second line of the LCD.
* The day of the week is also displayed (e.g., Monday).

#### ****Alarm and Timer System:****

* The system can set up to four timers, each with start and end times (Start1HH, Start1MM, Finish1HH, Finish1MM, etc.).
* When the current time matches a timer’s start time, an LED is turned ON.
* When the current time matches a timer’s end time, the LED is turned OFF.
* The timers are configurable through the IR remote or buttons connected to the Arduino.

#### ****PIR Motion Sensor:****

* A PIR motion sensor is connected to the system to detect movement.
* When motion is detected, the LED brightness is adjusted, or specific actions are taken depending on the logic defined in the code.

#### ****EEPROM for Persistent Data Storage:****

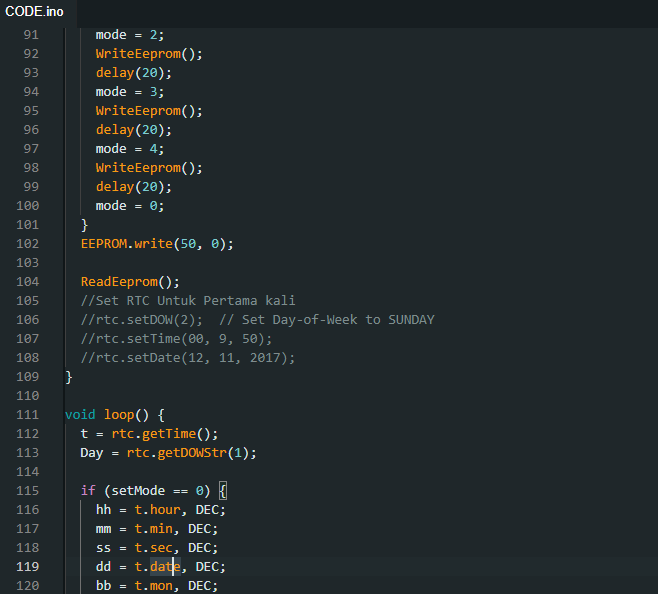
* The system stores certain settings, such as alarm configurations, in the Arduino’s EEPROM, ensuring that the settings are not lost when the device is powered off.
* The EEPROM read and write functions are called to store or retrieve the timer settings for the alarms.

#### ****IR Remote Control:****

* The IR remote is used to control different functionalities like setting the clock, adjusting timers, and controlling the LED.
* The code listens for IR commands and processes them based on specific IR button codes.
* Each button on the IR remote is mapped to a different function (e.g., adjusting the clock, toggling the alarm, etc.).

#### ****Buzzer Feedback:****

* A buzzer is used to provide audible feedback when certain actions are performed, such as pressing a button or triggering an alarm.



### 4. ****Functions and Features Breakdown:****

### ****Setup Function:****

* Initializes the LCD, IR receiver, PIR sensor, LED, and RTC module.
* Displays welcome messages on the LCD screen.
* Reads data from EEPROM to load saved settings (such as alarms).

#### ****Loop Function:****

* Continuously updates the current time from the RTC and displays it on the LCD.
* Manages alarm logic, checking if the current time matches any timer start or end time.
* Listens for input from the IR remote to adjust settings like the time, alarms, and LED control.
* Handles the PIR sensor’s input to adjust LED brightness when motion is detected.
* Updates the LCD to display the time, date, and any other relevant information.

#### ****Blinking Function:****

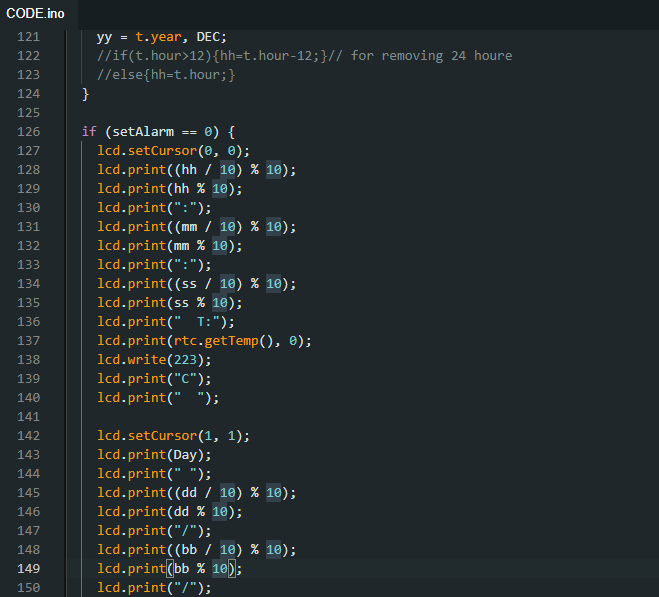
* This function creates a blinking effect on the LCD when setting the clock or alarms.
* The blinking effect helps the user visually track which part of the time (hour, minute, second) or alarm is being set.

#### ****setupClock() Function:****

* Handles the setup and adjustment of the clock and timers.
* Processes IR remote input to adjust the time and alarm settings.
* Updates the RTC module with the new time or alarm settings when changes are made.

#### ****EEPROM Functions (ReadEeprom and WriteEeprom):****

* These functions manage the reading and writing of data to the EEPROM, ensuring that alarm settings persist after the Arduino is powered off.
* The system writes the current alarm settings into specific memory addresses and reads from them when needed.



The complete code is in the github repository below:  
  
<https://github.com/hancho6319/ATBUproject4solarfloodlamp.git>

### Summary:

This Arduino project manages a real-time clock, timers, and an LED control system using multiple input methods like buttons, an IR remote, and a PIR sensor. The system stores alarm configurations in EEPROM and can control an LED based on timer settings and motion detection. It provides feedback through both an LCD and a buzzer, making it an interactive and user-friendly clock and alarm system.