Smart Clock on the AVR Platform

Student: Ilie Lucian-Alexandru

Grupa: 331CB

Introduction

The smart clock described in this project is a **stand-alone** device built around an AVR micro-controller (an Arduino-compatible UNO board). It continuously shows the time and date on a 0.96″ OLED display; it beeps briefly at every full hour; after 19:00 it gradually lights two LEDs acting as the hour- and minute-hands; and it adapts LED brightness to the room light level using an LDR sensor. The purpose of the project is to demonstrate features taken from **at least three PM laboratories**—Timers & ISR (Lab 2), PWM (Lab 3) and the I²C bus (Lab 6)—inside a useful, easy-to-demonstrate gadget. It also gives hands-on experience with an I²C OLED/LCD, a DS3231 RTC module and simple peripherals (LED, buzzer, LDR). The idea came from the fact that most low-cost electronic clocks drift by several minutes per month and have fixed brightness. By integrating a precision RTC and PWM/ADC control logic we eliminate these drawbacks. The clock is useful at home as a real gadget, can be easily extended (multiple alarms, temperature display, Bluetooth connection) and helps consolidate embedded-systems skills.

General Description

Smart-clock implementation on the AVR platform

The project is organized around an **Arduino UNO R3**, which acts as the brain of the system and links all modules. On the I²C bus two devices are connected: a **DS3231 real-time clock**, from which the micro-controller reads highly accurate time and date, and a **0.96″ OLED display** where this information is shown permanently. Two 5 mm LEDs, driven by PWM outputs, work as the clock “hands” and light up gradually after 19:00, indicating the passage of time without being intrusive during daytime. One more LED, a RGB one, is used with D9/D10/D11 pins with PWM for breathing effect and colour cycling. A passive buzzer, connected to a third PWM channel, emits a short beep at every full hour; the 1 kHz tone is generated by a hardware timer that starts and stops automatically. The ambient-light level is taken from a photo-resistor (LDR) on an analog pin; the ADC readings are used to dim the LEDs when the room is well lit.

Hardware Design

List of materials for the smart clock:

- **Arduino UNO R3** development board (CH340 compatible)
- **DS3231 RTC module** + **CR2032 battery**
- **0.96″ OLED display, I²C** (SSD1306)
- **MB102 breadboard**, 830 tie-points
- **Male-to-male jumper-wire set** (≥ 65 leads)
- **Passive buzzer, 5 V**
- **Red 5 mm LED** – hour indicator
- **Green 5 mm LED** – minute indicator
- **Common-cathode RGB LED, 5 mm** (+ 3 × 220 Ω resistors)
- **220 Ω resistors** × 2 (for the LEDs)
- **6 × 6 mm tactile button** (SET)
- **Photo-resistor (LDR) GL5528**
- **USB-A ↔ USB-B cable** (programming & power)
- **5 V / ≥ 1 A PSU** (phone charger)

Current hardware implementation status:

- All modules are wired on the breadboard and have been tested both **individually** and **together**
- **DS3231 RTC** keeps accurate time (drift \< ± 1 s / day)
- **SSD1306 OLED** shows the current time + ambient-light level (“Light level:”) once per second
- **LDR** reacts correctly (0 – 1023) and is used to drive a PWM brightness control
- **LEDs:**
  - **Red** – blinks at 1 Hz (handled by a Timer1 ISR)
  - **Green** – lights for 1 s whenever the minute changes
  - **RGB** – PWM “breathing” effect; switching colours on BTN press
- **Buzzer** plays short beeps and a longer “gong” on each full hour
- All three **push-buttons** operate with `INPUT_PULLUP` and falling-edge detection

Nr.	Component	Functional role in project	Pin / Library
1	**Arduino UNO R3**	MCU + Timers/ISR + PWM + I²C master	—
2	**RTC DS3231** (I²C)	Keeps real-time clock, internal temperature, alarm	**RTClib**
3	**0.96″ OLED SSD1306** (I²C)	Displays time, lux and LED mode	**Adafruit_SSD1306**
4	**Photo-resistor (LDR)** + 10 kΩ divider	Measures ambient light (auto-dimmer)	**A0**
5	**5 mm LED (red)**	Second indicator (Timer ISR)	**D5**
6	**5 mm LED (green)**	1 s flash whenever the minute changes (PWM-ready)	**D6**
7	**RGB LED**	PWM “breathing” effect; 7-colour cycling via PWM	**D9/D10/D11 (PWM)** + 3 × 220 Ω resistors
8	**Passive buzzer**	Beep feedback & hourly gong	**D3 (tone)**
9	**BTN HOUR**	Increment hour	**D2**

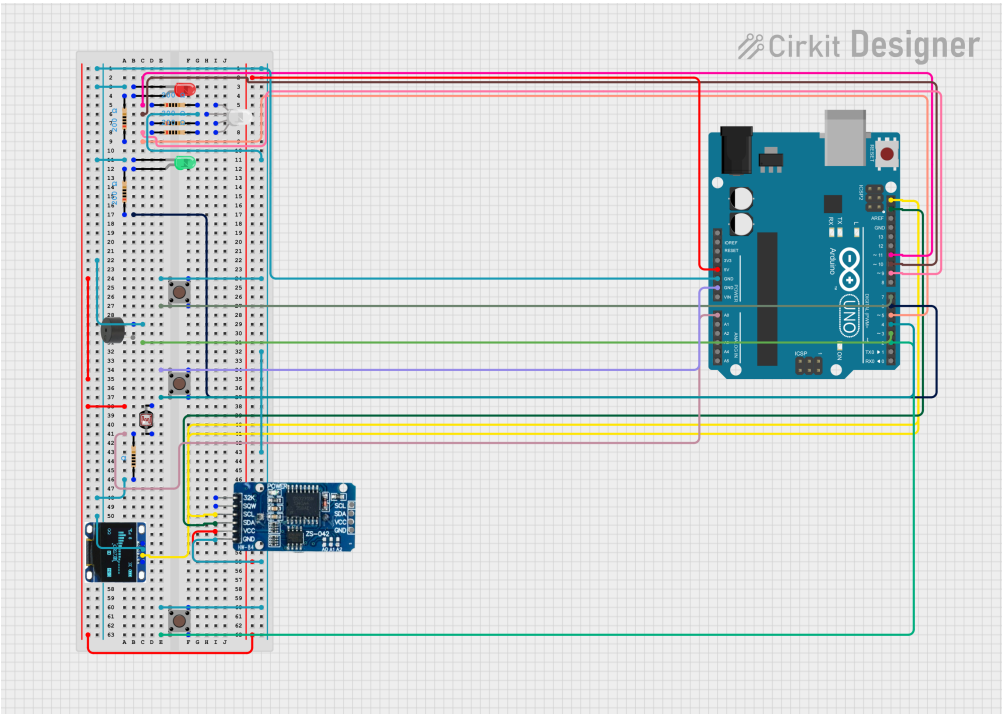
10	**BTN MIN**	Increment minute + beep	**D4**
11	**BTN MODE**	Toggle blue-LED mode	**D7**
12	**Breadboard + Dupont leads**	Rapid prototyping	—
13	**Resistors 220 Ω / 10 kΩ**	LED current limiters / LDR voltage divider	—

Pin mapping:

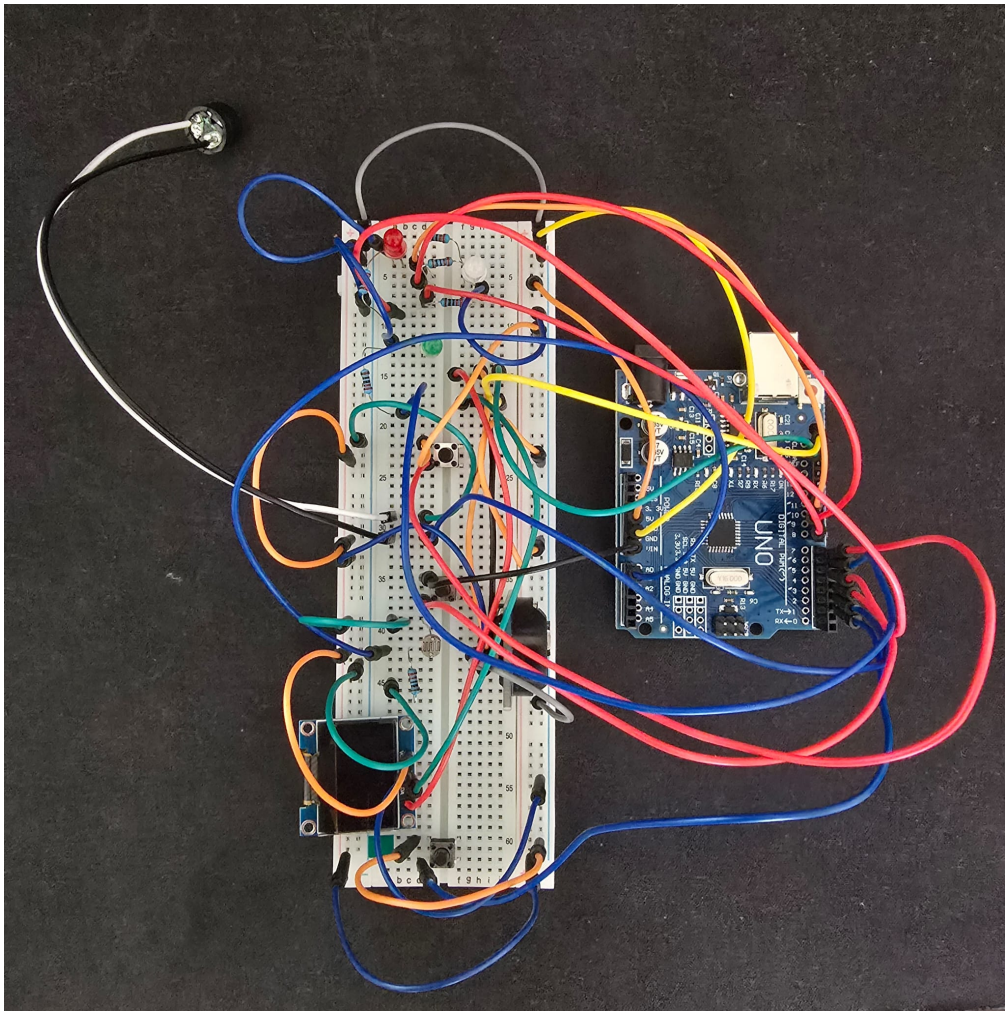
- **I²C bus (RTC + OLED)** — pins **SDA / SCL** (dedicated I²C lines, on-board pull-ups)
- **LDR (photo-resistor)** — **A0** · 10-bit ADC input
- **Red LED** — **D5** · toggled by Timer1 ISR at 1 Hz (no PWM needed)
- **Green LED** — **D6** · digital OUT / PWM-ready – flashes for one second each minute change
- **RGB LED** — **D9/D10/D11** · hardware PWM used for smooth *breathing* effect and colours cycling
- **Buzzer** — **D3** · `tone()` uses Timer2 for precise frequency generation
- **BTN HOUR** — **D2** · INT0-capable input (configured with `INPUT_PULLUP`)
- **BTN MIN** — **D4** · digital input (`INPUT_PULLUP`)
- **BTN MODE** — **D7** · digital input (`INPUT_PULLUP`)
- PWM-capable pins (**D3 / D5 / D6 / D9 / D10 / D11**) were selected so they don’t clash with the I²C bus (A4/A5) or the UART lines (D0/D1)

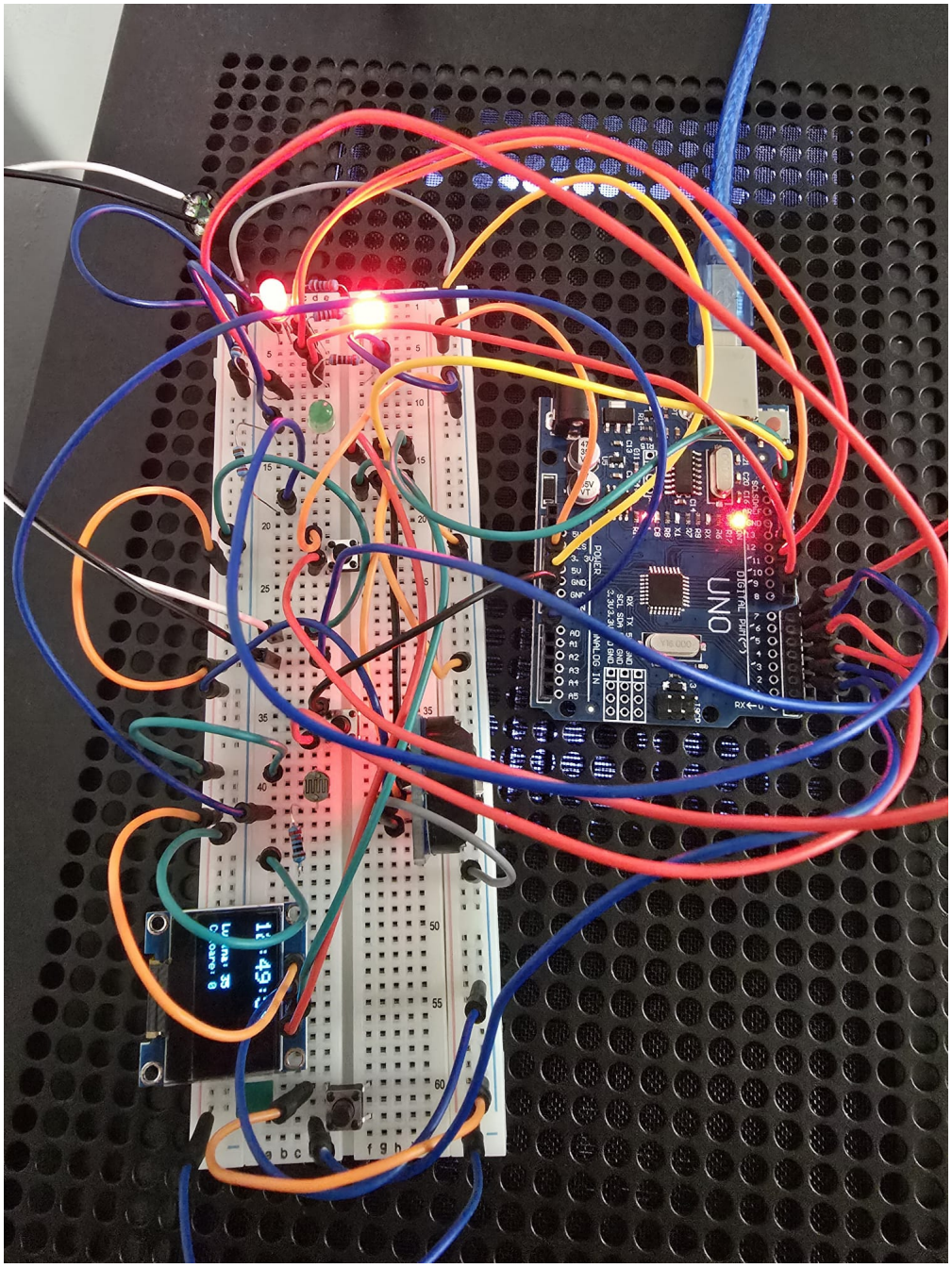
Electric scheme

The following diagram shows all components connected on the breadboard, including the RTC, OLED, LDR, 3 LEDs (Red, Green, RGB), passive buzzer, and 3 buttons.



Connected components & functionalities


UNO



The following pictures show the physical assembly of all modules connected on a breadboard:

- **OLED SSD1306** displays the following information, updated every second:
  - Current time in format `HH:MM:SS`
  - Ambient light level (lux value from LDR sensor)
  - Current RGB LED mode (`colorIdx`) + OFF status if index is 0
  - Dim mode status: “Dim mode: ON” or “Dim mode: OFF”
- **RTC DS3231** is used to keep accurate time with ±1s/day drift
- **LDR sensor** is placed on the breadboard and provides real-time ambient light readings (0–1023)
  - Used to activate automatic dimming if value is below 400
- **Push-buttons** (HOUR, MINUTE, MODE) are connected to D2, D4, D7 respectively
  - Used for adjusting the hour/minute and switching LED modes
- **Red and Green LEDs** are individually wired and behave as follows:

    - Red LED:
        - Blinks at 1 Hz before 19:00 using `millis()` logic
        - After 19:00, represents hour value via PWM intensity
    - Green LED:
        - Briefly flashes when the minute changes before 19:00
        - After 19:00, represents minute value via PWM intensity

- **Additional functionality (RGB LED + BTN MODE):**
    - **RGB LED (common cathode) connected to pins D9 / D10 / D11**
        - Displays one of **7 fixed colours** based on `colorIdx` value
        - Colour order: **Red → Green → Blue → Cyan → Magenta → Yellow → White**
        - Automatically dims in dark environments (detected via LDR)
        - While holding **BTN MODE**, RGB blinks white at 4 Hz as visual feedback

- **BTN MODE (D7, INPUT_PULLUP)**
    - **Short press**: advances to the next colour (circular logic using `colorIdx`)
    - **Long press**: enables fast white blinking for visual feedback
    - Every action also triggers a short beep (3 kHz, 50 ms) from the buzzer

- **Passive Buzzer (D3)**
    - Emits short beeps on button presses
    - Plays a long “gong” at every full hour
    - 1 Hz ticking sound every second (disabled between 19:00–23:59)

Software Design

Development environment: Arduino IDE 2.3.7

Libraries:

- *Wire.h* – establishes I²C communication with the RTC and OLED display
- *RTClib.h* – used to read and write time from the DS3231 real-time clock module
- *Adafruit_GFX.h* – provides core graphics functions (fonts, shapes, text positioning) for the OLED
- *Adafruit_SSD1306.h* – handles display control over I²C for the 128×64 OLED screen

Core functionalities implemented and tested:

- Time tracking with **RTC DS3231**, updated every second
- OLED display shows time, ambient light (LDR), RGB LED mode and dimMode status
- **Dim mode** activates automatically when LDR reads below threshold (value < 400), reducing RGB LED brightness
- **LED behavior is time-aware**:
    - Before 19:00: Red LED blinks at 1 Hz; Green LED flashes on minute change
    - After 19:00: Red/Green LEDs act as clock hands (PWM based on hour/minute)
- RGB LED cycles through 7 preset colours using PWM; its brightness adjusts dynamically based on ambient light level measured by the LDR (auto-dim at night)

- Buzzer generates:
  - Short beep on button press (except 19–23 when silent)
  - Longer “gong” at the top of each hour
  - 1 Hz ticking sound (disabled 19–23)
- All buttons use **INPUT_PULLUP** and falling-edge detection to avoid bounce

Functions implemented

- **void setup()** – initializes I/O pins, OLED display, RTC module, and serial communication
- **void loop()** – non-blocking main loop, checks buttons, updates LEDs and OLED, processes tick events
- **void setRGB(uint8_t r, uint8_t g, uint8_t b)** – sets the RGB LED to a specific color using PWM
- **void beep(uint16_t freq, uint16_t dur)** – plays a tone on the buzzer unless time is between 19:00 and 23:59
- **uint8_t hourToPWM(uint8_t h)** – maps hour (0–23) to a PWM value (0–255) for clock-hand LED behavior
- **uint8_t minuteToPWM(uint8_t m)** – maps minute (0–59) to PWM value for clock-hand LED behavior

Program logic

- The program uses `millis()` to generate a 1-second tick without blocking the main loop
- **At each tick:**
  1. Reads current time from RTC
  2. Updates system LEDs (red/green) depending on time of day
  3. Triggers buzzer if needed (heartbeat, hourly gong, button press)
  4. Refreshes the OLED with current time, light level, color index, and dim mode status

- **RGB LED:**
  1. Changes color with each MODE button press
  2. Blinks white rapidly while MODE is held down (hold-feedback)
  3. Dims automatically in low light (based on LDR reading)

- **LDR:**
  1. Continuously read via `analogRead()`
  2. Triggers dim mode below threshold (`ldrThreshold = 400`)

- **Buttons:**
  1. BTN_H: increments hour on press
  2. BTN_M: increments minute + beeps
  3. BTN_MODE: cycles through color presets

How it works

- When powered on, the system initializes all components and retrieves the current time from the DS3231 RTC
- The red LED blinks every second as a heartbeat until 19:00
- After 19:00, the red LED brightness reflects the current hour (PWM), while the green LED reflects the current minute
- The green LED flashes once every new minute before 19:00
- The RGB LED can be set to one of 7 preset colors; it dims automatically at night based on ambient light
- Holding the MODE button causes the RGB LED to blink white rapidly as visual feedback

- The buzzer emits a short beep on button press and a longer “gong” at each full hour, except during the silent hours (19:00–23:59)
- The OLED displays the current time (HH:MM:SS), ambient light level, RGB mode, and whether dim mode is ON/OFF

New features introduced (vs. template)

- Time-dependent LED behavior:
  - Classic blinking/flashing before 19:00
  - LED “hands” with proportional PWM after 19:00
- Auto-dimming feature using LDR sensor and PWM reduction
- Dynamic buzzer behavior (silent hours implemented in software)
- Multi-mode RGB LED, including a white flashing effect on long-press
- OLED updates every second: Time, light level, current RGB mode, dim status

Features implemented from labs

Lab 3 – PWM, LED breathing & analogRead

- PWM control for RGB LEDs and system LEDs (red/green after 19:00)
- Smooth “breathing” animation effect using PWM on RGB LED
- Light detection using `analogRead()` from an LDR sensor
- Threshold-based logic (`ldrThreshold = 400`) for enabling automatic “dim mode” at night

Lab 4 – millis(), timer-based updates & OLED/RTC integration

- Use of `millis()` for implementing non-blocking 1-second ticks
- Red LED blinking at 1 Hz without using `delay()`
- Real-time clock support via DS3231 and `RTClib`
- Time display updated every second on OLED using `Adafruit_SSD1306`

Lab 5 – tone() and auditory feedback

- Use of `tone()` to play short sounds on button interaction
- Hourly chime (“gong”) at full hour events
- Conditional muting logic: buzzer is disabled between 19:00–23:59 to avoid disturbance

Lab 6 – I²C bus, debouncing, edge detection, and button logic

- Communication with both **RTC DS3231** and **OLED SSD1306** via the **I²C bus** (`Wire.h`)
- All buttons are configured with `INPUT_PULLUP`
- Falling-edge detection using previous state comparison
- Three buttons implemented with distinct actions (hour, minute, mode switching)
- Every interaction triggers a visual and/or auditory feedback

Testing & calibration

- LDR threshold value (`ldrThreshold = 400`) chosen after testing under normal room lighting

- RGB LED tested with dimming and full brightness in varying light conditions
- Buttons debounced and tested for both short press and hold
- Verified correct PWM values for hour (0–23) and minute (0–59) mapping
- Buzzer output silenced reliably in “quiet hours” 19:00–23:59

Software optimizations

- No `delay()` usage – entire loop is non-blocking using `millis()`
- Only updates OLED once per second to avoid flicker
- PWM outputs adjusted only when needed
- All state variables and timing use unsigned types (overflow-safe)
- `tick` flag separates logic from timing checks

Video explanation

- Demo includes:
  - Full operation cycle (LED behavior before and after 19:00)
  - RGB LED mode change & white blink effect
  - LDR dimming effect with covered/uncovered sensor
  - Time update via buttons (HOUR, MINUTE)
  - Buzzer feedback with conditional silence
  - OLED display showing all real-time parameters

Rezultate Obţinute

The results after completing all the project:

https://streamable.com/nzzcwh

This video presents the base functionalities of the smart clock: * The **OLED screen** displays real-time data: current time, ambient light level (lux), and the active RGB color index. * The **RGB LED** changes color with each short press of **BTN MODE**, cycling through 7 predefined colors. * Holding **BTN MODE** triggers a visual feedback mode where the RGB LED blinks white at 4 Hz. * The **Dim mode** (based on LDR readings) reduces LED brightness in low light conditions. * The **hour** and **minute buttons** allow manual time adjustments.

https://streamable.com/ryv6rw

This clip shows the additional functionality activated after 19:00: * The **red LED** (D5) no longer blinks but gradually brightens depending on the current **hour**. * The **green LED** (D6) lights up with intensity proportional to the current **minute**. This simulates a simple hour-and-minute hand system using LED brightness. After 00:00, both LEDs return to their standard behavior (blinking and minute-flash).

Additionally, the **RGB LED** adapts to ambient light using the **LDR sensor**. When you shine light onto the photoresistor, the RGB LED's brightness increases accordingly. In darker environments, the RGB LED automatically dims (enters **dim mode**) for smoother visual integration during nighttime. </note>

Conclusions

The development of this project provided valuable experience across both hardware and software domains, offering a practical understanding of embedded systems and real-time interaction.

Skills and competencies acquired:

- **Embedded programming and software logic:**
  - Implemented real-time clock tracking, dynamic LED behavior, PWM dimming, and non-blocking execution using `millis()`
  - Improved understanding of writing efficient, responsive code for microcontrollers
- **Sensor integration and environmental feedback:**
  - Used an LDR sensor to adjust LED brightness based on ambient light
  - Gained insight into environmental sensing and intelligent system response
- **User interaction and input handling:**
  - Handled multiple button inputs (hour, minute, mode)
  - Applied debounce and edge-detection techniques for stable input detection
- **Visual output and real-time status display:**
  - Utilized the OLED to show real-time time, light level, RGB mode, and dim status
  - Strengthened knowledge of I²C communication and display control
- **Audio feedback and conditional logic:**
  - Implemented buzzer feedback that adapts to time of day (silent hours)
  - Applied `tone()` and timers to manage auditory signals
- **Problem-solving and debugging:**
  - Overcame challenges in timer synchronization, display refresh, and RGB transitions
  - Improved analytical thinking and troubleshooting in embedded contexts

Practical benefits and applications:

- **Time-aware visual indicators:**
  - Offers LED behavior that changes based on time, useful for ambient clocks or alerts
- **Custom interaction patterns:**
  - Button logic and color modes are adaptable to other user-controlled systems
- **Expandability and real-world relevance:**
  - Modular design supports easy addition of sensors and extensions
  - Strong foundation for IoT, automation, or smart home projects

In conclusion, this project served as a comprehensive introduction to embedded system design—combining real-time programming, sensor integration, display systems, and human interaction into a cohesive and practical implementation.

Download

Link for the code: https://github.com/Ilie28/Proiect-PM

Journal

06.05.2025 – decided on the project, wrote the description and the hardware materials
11.05.2025 – materials bought, starting on the hardware design
13.05.2025 – tested RTC, OLED, LDR, buzzer and buttons independently
15.05.2025 – implemented LED functionality (blinking, PWM, mode switching)
16.05.2025 – finished integration: all modules tested together
17.05.2025 – added RGB LED support and 7-color switching logic
18.05.2025 – finalized code, added edge detection, PWM breathing, and buzzer feedback
19.05.2025 – created and uploaded schematic + wiring diagram
20.05.2025 – documented project structure and updated project page
22.05.2025 – finalized software logic and display integration; completed all wiki documentation

Bibliography/Resources

Hardware Resources

- **Components:**
  - https://shorturl.at/wJVZz
  - https://shorturl.at/tPXaz
  - https://shorturl.at/vd9zp
  - https://shorturl.at/7pFHC
  - https://tinyurl.com/bdhm9uck
  - https://tinyurl.com/2b9mh7cf
  - https://tinyurl.com/j2tesvn9
  - https://tinyurl.com/47mrsvj7
- **Datasheets:**
  - https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf
  - https://datasheets.maximintegrated.com/en/ds/DS3231.pdf
  - https://cdn-shop.adafruit.com/datasheets/SSD1306.pdf
  - https://docs.arduino.cc/hardware/uno-rev3

Software Resources:

- https://learn.adafruit.com/adafruit-oled-display-library
- https://learn.adafruit.com/adafruit-gfx-graphics-library
- https://github.com/adafruit/RTClib

- https://www.arduino.cc/reference/en/
- https://platformio.org/lib/show/80/Adafruit%20GFX%20Library
- https://platformio.org/lib/show/562/Adafruit%20SSD1306

Export to PDF