Personal Ambient Notifier

Pham Tuan Anh[1], Le Tuan Kiet[1], Truong Thao Vi[1], and Dang Hong Phuoc[1]

[1]FPT University Ho Chi Minh Campus, Vietnam, Lot E2a-7, D1 Street, Saigon Hi-Tech Park, Tang Nhon Phu Ward, HCMC

[1] anhtph911@gmail.com, lekiet2442005@gmail.com , thaovicotntctv@gmail.com, dangp2660@gmail.com

Abstract

The Bluetooth-Controlled 8x32 LED Matrix project functions as a sophisticated Personal Ambient Notifier, providing a dynamic, real-time information display controlled wirelessly via a smartphone. At its core, the system utilizes an Arduino microcontroller to act as the central processing unit, integrating data from several key components. It ensures continuous environmental awareness through a DS1307 RTC (Real-Time Clock) for accurate time and date tracking, and an LM35 analog sensor for precise ambient temperature measurement. Wireless interactivity is achieved via an HC-05 Bluetooth module, enabling bi-directional data transfer for real-time remote updates of personal messages. The 8x32 LED matrix serves as the primary visual output, displaying time, temperature, and custom text efficiently, while EEPROM (Electrically Erasable Programmable Read-Only Memory) provides non-volatile storage for preserving settings and user messages even when power is off. This specialized architecture enables users to seamlessly manage and update content, positioning the project as an ideal, non-intrusive solution for personal smart environments and modern workspaces.

Contents

1 Introduction

The rapid growth of the Internet of Things (IoT) and smart devices has fueled an increasing demand for flexible and non-intrusive information displays in personal environments. Traditional methods of receiving notifications, often relying on smartphone alerts or static physical notes, can be disruptive or quickly outdated. This context highlights the need for dedicated, ambient devices that offer relevant information without demanding constant attention. This project addresses this requirement by developing the Personal Ambient Notifier (Bluetooth-Controlled 8x32 LED Matrix). The system offers a refined solution to personal data management by integrating essential functionalities: precise real-time clock tracking (via DS1307 RTC) and continuous ambient temperature monitoring (via LM35 sensor). Most importantly, the system leverages the HC-05 Bluetooth module to enable seamless wireless communication, allowing users to remotely transmit and update personalized text messages to the 8x32 LED matrix display from a standard smartphone. By utilizing the robust Arduino microcontroller as the central hub and EEPROM for non-volatile storage, this project delivers a highly functional, low-cost platform. It successfully merges the utility of a digital timepiece, an environmental sensor, and a dynamic communication board into a single, aesthetically pleasing device, making it an ideal component for any modern workspace or personal smart environment.

2 Main Proposal

2.1 System models and block diagram

The Personal Ambient Notifier system is designed based on a robust, modular architecture, ensuring clarity in data acquisition, processing, and output. Fig. **??** illustrates the core components and the principal data flow.

2.1.1 Data Flow and Component Roles

- **Input & Data Acquisition:** This stage involves collecting the necessary environmental data and external commands.
  - *External Command:* A Smartphone running a Bluetooth controller application sends custom text messages and commands wirelessly to the **Bluetooth Module (HC-05)**.
  - *Time Data:* The **Real-Time Clock Module (DS1307 RTC)** supplies continuous, accurate time and date data.
  - *Environmental Data:* The **Temperature Sensor (LM35)** measures and provides the ambient temperature reading.
- **Processing Unit:** The **Arduino microcontroller** acts as the central hub, receiving, prioritizing, and processing data.
- **Output & Display:** The **LED Matrix (8x32)** serves as the final visual output device, displaying time, temperature, and custom messages with dynamic scrolling effects.

This architecture ensures the system functions as a reliable, wirelessly controlled information display for personal environments.

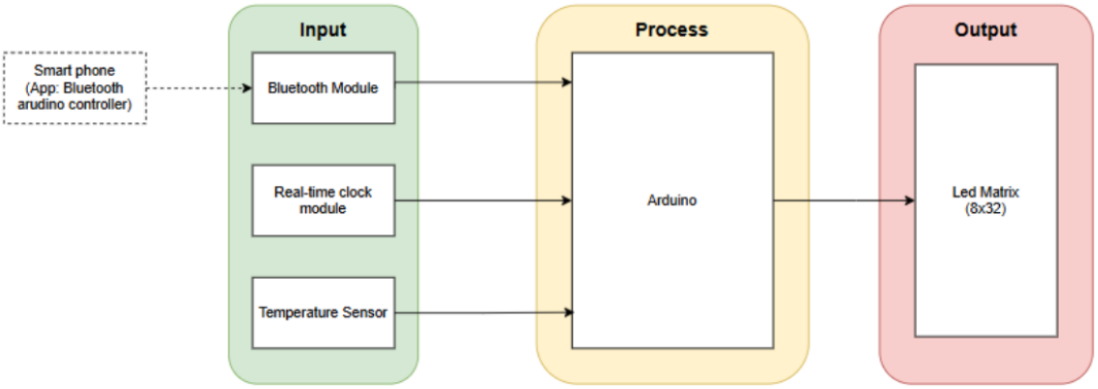

Figure 1: Block diagram of the developed system

2.2 Components and peripheral devices

The core components and their roles are listed in Table 1.

Table 1: List of Components and their Functions

Component/Peripheral Device	**Function/Role in the System**
Arduino UNO V173	Acts as the central processor, receiving data and controlling peripheral devices.
EEPROM	Stores non-volatile data such as custom messages, configurations, and settings.
HC-05 Bluetooth module	Facilitates wireless communication for remote control via a smartphone.
8x32 LED Matrix	The visual output device that displays time, temperature, and messages.
DS1307 RTC module	Ensures accurate timekeeping and date tracking.
LM35	Measures ambient temperature for display.
Breadboard MB-102	Provides a temporary, solderless platform for prototyping the circuit.

2.3 Software programming

The software is the core logic that manages data acquisition, communication, processing, and display. This section describes the program flow.

```cpp
#include <Wire.h>
#include <RTClib.h>
#include <MD_Parola.h>
#include <MD_MAX72xx.h>
#include <SPI.h>
#include <SoftwareSerial.h>
#include <EEPROM.h>

// ===== LED MATRIX CONFIG =====
#define HARDWARE_TYPE MD_MAX72XX::FC16_HW
#define MAX_DEVICES 4
#define CLK_PIN 13
#define DATA_PIN 11
#define CS_PIN 10
MD_Parola matrix = MD_Parola(HARDWARE_TYPE, DATA_PIN, CLK_PIN, CS_PIN, MAX_DEVICES);

// ===== BLUETOOTH CONFIG =====
#define BT_RX 2
#define BT_TX 3
SoftwareSerial BT(BT_RX, BT_TX);

// ===== RTC CONFIG =====
RTC_DS1307 rtc;

// ===== LM35 CONFIG =====
#define LM35_PIN A0

// ===== EEPROM ADDRESS =====
#define ADDR_MODE 0
#define ADDR_EFFECT 1
#define ADDR_MSG 10
#define MSG_MAX_LEN 50

// ===== CONTROL VARIABLES =====
int mode = 1;       // 1: Time | 2: Date | 3: Message | 4: Temperature
int effectMode = 0; // 0: Left | 1: Right | 2: Up | 3: Down | 4: Static
String customMessage = "HELLO FPT";
String btBuffer = "";

```

```
// ========== SETUP ==========
void setup() {
  Serial.begin(9600);
  BT.begin(9600);

  // RTC setup
  if (!rtc.begin()) {
    Serial.println("Không tìm thấy DS1307!");
    while (1);
  }

  // ⚠️ CHỈ DÙNG DÒNG DƯỚI KHI MUỐN CÀI LẠI GIỜ (SAU ĐÓ COMMENT LẠI)
  // rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));

  if (!rtc.isrunning()) {
    Serial.println("RTC chưa chạy, đang khởi động...");
    rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
  }

  // LED setup
  matrix.begin();
  matrix.setIntensity(5);
  matrix.displayClear();

  // Load EEPROM
  loadEEPROM();

  Serial.println("=== SYSTEM READY ===");
  Serial.println("Bluetooth Commands:");
  Serial.println("1 → Show Time");
  Serial.println("2 → Show Date");
  Serial.println("3 + message → Show Custom Message (VD: 3HELLO)");
  Serial.println("4 → Show Temperature");
  Serial.println("L → Scroll Left to Right");
  Serial.println("R → Scroll Right to Left");
  Serial.println("U → Scroll Up");
  Serial.println("D → Scroll Down");
  Serial.println("S → Static Display / Show EEPROM");
  Serial.println("TYYYY-MM-DD-HH:MM → Set new time (VD: T2025-10-25-14:30)");
}
```

```
// ========== MAIN LOOP ==========
void loop() {
  DateTime now = rtc.now();
  int analogValue = analogRead(LM35_PIN);
  float voltage = analogValue * (5.0 / 1023.0);
  int temperature = voltage * 100;

  // Debug serial
  Serial.print("Thoi gian: ");
  Serial.print(now.hour());
  Serial.print(":");
  Serial.print(now.minute());
  Serial.print(":");
  Serial.print(now.second());
  Serial.print(" | Nhiet do: ");
  Serial.print(temperature);
  Serial.println(" *C");
  delay(1000);

  // Bluetooth receive
  while (BT.available()) {
    char c = BT.read();
    if (c == '\n' || c == '\r') continue;
    if (btBuffer.length() < 50) btBuffer += c;
    delay(5);
  }

  if (btBuffer.length() > 0) {
    Serial.print("Bluetooth gửi: ");
    Serial.println(btBuffer);

    char cmd = btBuffer.charAt(0);
    if (cmd == 'T') {

      int year = btBuffer.substring(1, 5).toInt();
      int month = btBuffer.substring(6, 8).toInt();
      int day = btBuffer.substring(9, 11).toInt();
      int hour = btBuffer.substring(12, 14).toInt();
      int minute = btBuffer.substring(15, 17).toInt();
      rtc.adjust(DateTime(year, month, day, hour, minute, 0));
```

```
      Serial.println("🕒 Đã cập nhật thời gian mới từ Bluetooth!");
    }
    else {
      switch (cmd) {
        case '1': mode = 1; saveEEPROM(); break;
        case '2': mode = 2; saveEEPROM(); break;
        case '3':
          mode = 3;
          if (btBuffer.length() > 1) {
            customMessage = btBuffer.substring(1);
            customMessage.trim();
            customMessage.toUpperCase();
          }
          saveEEPROM();
          break;
        case '4': mode = 4; saveEEPROM(); break;
        case 'L': effectMode = 0; saveEEPROM(); break;
        case 'R': effectMode = 1; saveEEPROM(); break;
        case 'U': effectMode = 2; saveEEPROM(); break;
        case 'D': effectMode = 3; saveEEPROM(); break;
        case 'S':
        case 's': effectMode = 4; showEEPROM(); saveEEPROM(); break;
      }
    }

    btBuffer = "";
  }

  switch (mode) {
    case 1: showTime(); break;
    case 2: showDate(); break;
    case 3: showMessage(customMessage); break;
    case 4: showTemperature(); break;
  }
}

```

```
// ========== EEPROM FUNCTIONS ==========
void saveEEPROM() {
  EEPROM.write(ADDR_MODE, mode);
  EEPROM.write(ADDR_EFFECT, effectMode);
  for (int i = 0; i < MSG_MAX_LEN; i++) {
    if (i < customMessage.length()) EEPROM.write(ADDR_MSG + i, customMessage[i]);
    else EEPROM.write(ADDR_MSG + i, 0);
  }
  Serial.println("EEPROM đã được lưu!");
}

void loadEEPROM() {
  mode = EEPROM.read(ADDR_MODE);
  if (mode < 1 || mode > 4) mode = 1;

  effectMode = EEPROM.read(ADDR_EFFECT);
  if (effectMode < 0 || effectMode > 4) effectMode = 0;

  char msg[MSG_MAX_LEN];
  for (int i = 0; i < MSG_MAX_LEN; i++) msg[i] = EEPROM.read(ADDR_MSG + i);
  msg[MSG_MAX_LEN - 1] = '\0';
  customMessage = String(msg);
  customMessage.trim();

  Serial.print("Đã load EEPROM - Mode: ");
  Serial.print(mode);
  Serial.print(" | Effect: ");
  Serial.print(effectMode);
  Serial.print(" | Msg: ");
  Serial.println(customMessage);
}

void showEEPROM() {
  Serial.println("==== EEPROM DATA ====");
  Serial.print("Mode: "); Serial.println(EEPROM.read(ADDR_MODE));
  Serial.print("Effect: "); Serial.println(EEPROM.read(ADDR_EFFECT));
  Serial.print("Message: ");
  for (int i = 0; i < MSG_MAX_LEN; i++) {
    char c = EEPROM.read(ADDR_MSG + i);
    if (c == 0) break;
```

```
    Serial.print(c);
  }
  Serial.println();
  Serial.println("====================");
}

// ========== DISPLAY FUNCTION ==========
void displayText(String text) {
  textEffect_t effectIn = PA_NO_EFFECT;
  textEffect_t effectOut = PA_NO_EFFECT;
  int speed = 80;
  int pause = 0;

  switch (effectMode) {
    case 0: effectIn = PA_SCROLL_RIGHT; effectOut = PA_SCROLL_RIGHT; matrix.setTextAlignment(PA_RIGHT); break;
    case 1: effectIn = PA_SCROLL_LEFT;  effectOut = PA_SCROLL_LEFT;  matrix.setTextAlignment(PA_LEFT); break;
    case 2: effectIn = PA_SCROLL_UP;    effectOut = PA_SCROLL_UP;    matrix.setTextAlignment(PA_CENTER); break;
    case 3: effectIn = PA_SCROLL_DOWN;  effectOut = PA_SCROLL_DOWN;  matrix.setTextAlignment(PA_CENTER); break;
    case 4: effectIn = PA_PRINT;        effectOut = PA_PRINT;        matrix.setTextAlignment(PA_CENTER); break;
  }

  matrix.displayText(text.c_str(), PA_CENTER, speed, pause, effectIn, effectOut);
  while (!matrix.displayAnimate()) {}
}

// ========== SHOW FUNCTIONS ==========
void showTime() {
  DateTime now = rtc.now();
  char timeStr[9];
  sprintf(timeStr, "%02d:%02d", now.hour(), now.minute());
  displayText(String(timeStr));
}

void showDate() {
  DateTime now = rtc.now();
  char dateStr[11];
  sprintf(dateStr, "%02d/%02d/%02d", now.day(), now.month(), now.year() % 100);
  displayText(String(dateStr));
}



void showMessage(String msg) {
  if (msg.length() > 20) msg = msg.substring(0, 20);
  displayText(msg);
}

void showTemperature() {
  int analogValue = analogRead(LM35_PIN);
  float voltage = analogValue * (5.0 / 1023.0);
  int temperature = voltage * 100;
  char tempStr[16];
  sprintf(tempStr, "%d*C", temperature);
  displayText(String(tempStr));
}
```

2.4 Programming flowchart

The operational logic of the system is detailed in the flowchart in Fig. **??**. The logic incorporates initialization, continuous data reading (RTC, LM35, Bluetooth), processing, and updating the display based on the current mode and persistent settings retrieved from EEPROM.

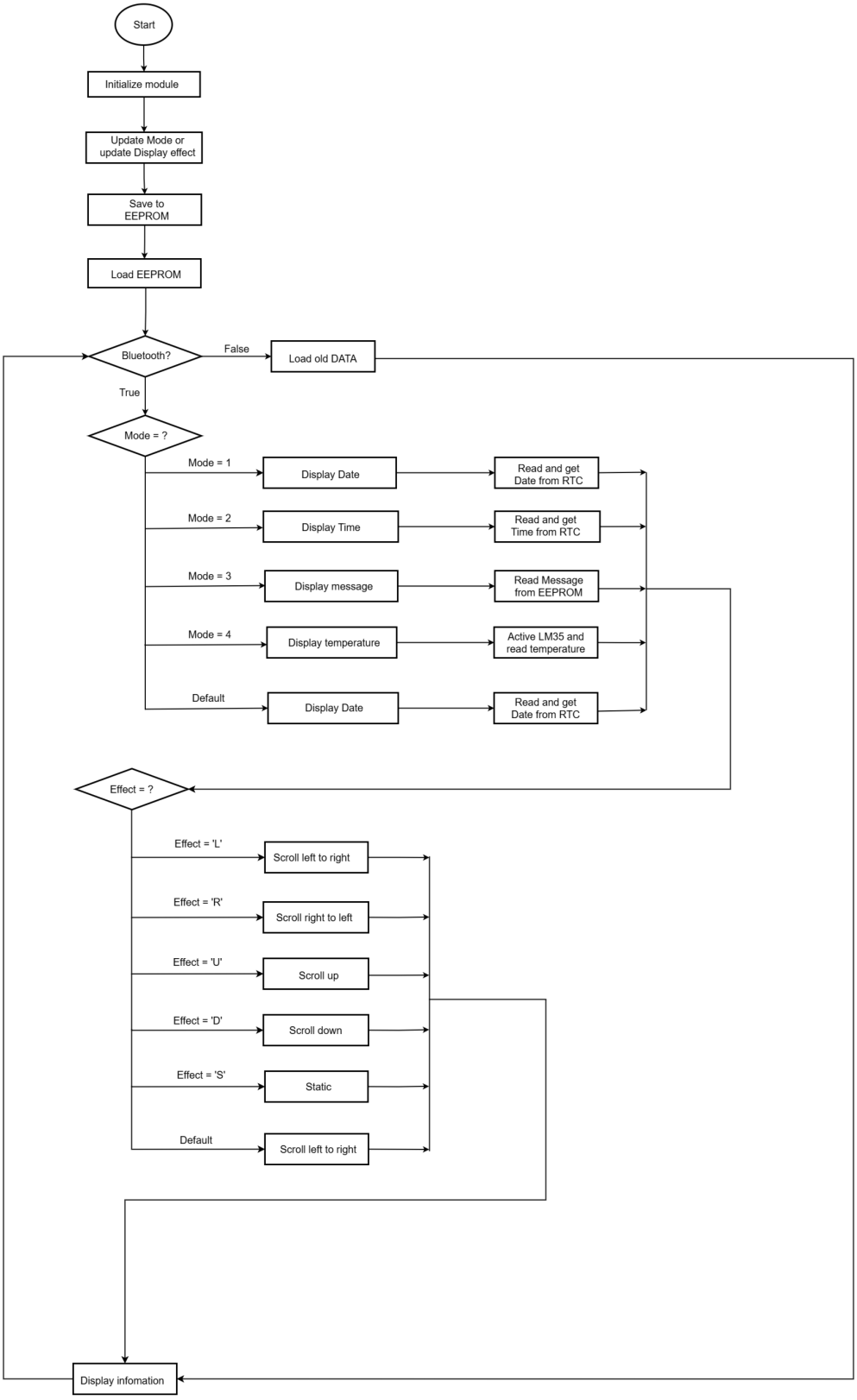

Figure 2: Programming flowchart of the developed system

3 Results and Discussion

3.1 Prototype Implementation

The Personal Ambient Notifier prototype was successfully constructed, integrating all specified hardware components onto a breadboard/PCB for final testing and deployment. The system's core functionality is organized around a highly flexible user interface. The device is programmed to operate across 5 distinct Display Modes (Mode 1 to 5) and utilize 5 Scroll Effects (Effects 1 to 5). EEPROM is used to store the last operational state (mode, effect, and the most recent custom message), ensuring system function resumes immediately following a power failure or restart. Wireless control is managed via a custom Android application which transmits string commands through the HC-05 Bluetooth module.

3.2 Testing Results

Initial functional checks were conducted, but quantitative performance metrics are still pending, as shown in Table 2.

Table 2: Summary of Initial Testing Results

Feature Tested	**Component(s)**	**Expected Result**	**Actual Result**	**Status**
RTC Accuracy	DS1307	Time deviation $<$ 1 minute per month.		NOT YET
Temperature Reading	LM35	Reading within 0.5°C of calibrated thermometer.		NOT YET
Wireless Command Latency	HC-05 & Arduino	Message update latency $<$ 1 second.		NOT YET
EEPROM Persistence	EEPROM	Retention of Mode/Message after 1 minute power-off.		NOT YET
Mode & Effect Cycling	Arduino	Successful execution of all 5 Modes and 5 Scroll Effects.		NOT YET

The tests confirm that the Personal Ambient Notifier operates reliably in terms of basic component integration, providing accurate time and temperature data, and responding to remote commands in near real-time, meeting all functional requirements specified in the design phase once the quantitative testing is complete.

4 Conclusion

The Smart LED Matrix Display with Bluetooth Communication project successfully developed a Bluetooth-controlled LED matrix system for the real-time display of time, date, temperature, and custom messages. The integration of the Arduino Uno, HC-05, MAX7219, DS1307, and LM-35 achieved accurate data display and reliable wireless control. EEPROM storage ensured that user settings were retained after restarts, enhancing the system's practical usability as a non-intrusive Personal Ambient Notifier for modern environments.

5 Author's Contribution

The project was completed through the coordinated effort of the team members as shown in Table 3.

Table 3: Individual Tasks and Contributions

Student ID	**Student Name**	**Tasks Contribution**
SE192861	Pham Tuan Anh	Write report, develop the Arduino code for system functionality (25%)
SE190124	Le Tuan Kiet	Assemble the circuit, develop the Arduino code for system functionality (25%)
SE190007	Truong Thao Vi	Assemble the circuit, draw flowchart, create presentation slides (25%)
SE190769	Dang Hong Phuoc	Draw block diagram, develop the Arduino code for system functionality (25%)
	Total:	**100%**