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of receiving notifications, often relying on smartphone alerts or static physical notes, can be disruptive or quickly outdated.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. Figure 1 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.

2.1.2 Block diagram

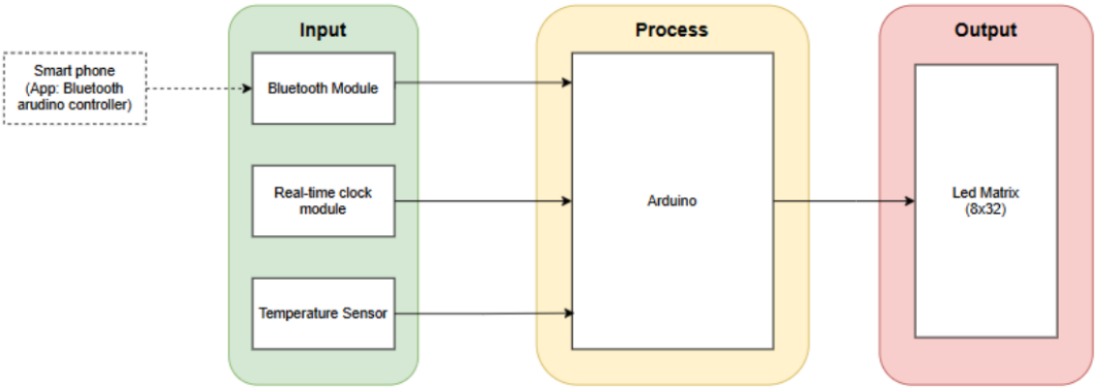

Figure 1: Block diagram of the developed system

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 Rev3	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.
LM35	Measures ambient temperature for display.
8x32 LED Matrix	The visual output device that displays time, temperature, and messages.
DS1307 RTC module	Ensures accurate timekeeping and date tracking.
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.

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

#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);

#define BT_RX 2
#define BT_TX 3
SoftwareSerial BT(BT_RX, BT_TX);

RTC_DS1307 rtc;
#define LM35_PIN A0

#define ADDR_MODE 0
#define ADDR_EFFECT 1
#define ADDR_MSG 10
#define MSG_MAX_LEN 50

int mode = 1;
int effectMode = 0;
String customMessage = "HELLO FPT";
String btBuffer = "";
unsigned long lastUpdate = 0;

int manualHour = 0;
int manualMinute = 0;

void setup() {
  Serial.begin(9600);
  BT.begin(9600);
  if (!rtc.begin()) while (1);
```

```
  if (!rtc.isrunning()) rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
  matrix.begin();
  matrix.setIntensity(5);
  matrix.displayClear();
  matrix.setFont(nullptr);
  loadEEPROM();
}

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

  while (BT.available()) {
    char c = BT.read();
    if (c != '\n' && c != '\r') btBuffer += c;
  }

  if (btBuffer.length() > 0) {
    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));
    } else {
      switch (cmd) {
        case '1': mode = 1; saveEEPROM(); break;
        case '2': mode = 2; saveEEPROM(); break;
        case '3': mode = 3; customMessage = btBuffer.substring(1); customMessage.trim(); customMessage.toUpperCase(); saveEEPROM(); break;
        case '4': mode = 4; saveEEPROM(); break;
        case '5': mode = 5; {
          DateTime n = rtc.now();
          manualHour = n.hour();
          manualMinute = n.minute();
          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 '+': if (mode == 5) manualHour = (manualHour + 1) % 24; break;
        case '-': if (mode == 5) manualHour = (manualHour + 23) % 24; break;
        case '>': if (mode == 5) manualMinute = (manualMinute + 1) % 60; break;
        case '<': if (mode == 5) manualMinute = (manualMinute + 59) % 60; break;
        case 'O': if (mode == 5) {
          DateTime n = rtc.now();
          rtc.adjust(DateTime(n.year(), n.month(), n.day(), manualHour, manualMinute, 0));
          mode = 1;
          saveEEPROM();
        } break;
      }
    }
    btBuffer = "";
  }

  unsigned long nowMs = millis();
  switch (mode) {
    case 1: if (nowMs - lastUpdate > 1000) { showTime(); lastUpdate = nowMs; } break;
    case 2: if (nowMs - lastUpdate > 2000) { showDate(); lastUpdate = nowMs; } break;
    case 3: showMessage(customMessage); break;
    case 4: if (nowMs - lastUpdate > 2000) { showTemperature(); lastUpdate = nowMs; } break;
    case 5: showManualAdjust(); delay(400); break;
  }
}

void saveEEPROM() {
  EEPROM.write(ADDR_MODE, mode);
  EEPROM.write(ADDR_EFFECT, effectMode);
  for (int i = 0; i < MSG_MAX_LEN; i++)
    EEPROM.write(ADDR_MSG + i, (i < customMessage.length()) ? customMessage[i] : 0);
}

void loadEEPROM() {
  mode = EEPROM.read(ADDR_MODE);
  effectMode = EEPROM.read(ADDR_EFFECT);
  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();
}

void displayText(String text) {
  textEffect_t effect = PA_SCROLL_LEFT;
  int speed = 80, pause = 0;
  switch (effectMode) {
    case 0: effect = PA_SCROLL_RIGHT; break;
    case 1: effect = PA_SCROLL_LEFT; break;
    case 2: effect = PA_SCROLL_UP; break;
    case 3: effect = PA_SCROLL_DOWN; break;
  }
  matrix.displayText(text.c_str(), PA_CENTER, speed, pause, effect, effect);
  while (!matrix.displayAnimate());
}

void showTime() {
  DateTime now = rtc.now();
  char buffer[6];
  sprintf(buffer, "%02d:%02d", now.hour(), now.minute());
  displayText(buffer);
}

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

void showMessage(String msg) {
  msg.trim();
  msg.toUpperCase();
  if (effectMode == 2 || effectMode == 3) {
    int totalCols = MAX_DEVICES * 8;
    const int approxCharWidth = 6;
    int maxCharsPerLine = totalCols / approxCharWidth;
```

```
    if (maxCharsPerLine < 1) maxCharsPerLine = 1;
    const int padSpaces = 2;
    msg += " ";
    int n = msg.length();
    int pos = 0;
    while (pos < n) {
      while (pos < n && msg.charAt(pos) == ' ') pos++;
      if (pos >= n) break;
      int end = pos + maxCharsPerLine;
      if (end > n) end = n;
      int lastSpace = -1;
      for (int i = pos; i < end; i++) if (msg.charAt(i) == ' ') lastSpace = i;
      String part;
      if (end >= n) {
        part = msg.substring(pos, n);
        pos = n;
      } else if (msg.charAt(end) == ' ') {
        part = msg.substring(pos, end);
        pos = end + 1;
      } else if (lastSpace != -1) {
        part = msg.substring(pos, lastSpace);
        pos = lastSpace + 1;
      } else {
        part = msg.substring(pos, end);
        pos = end;
      }
      part.trim();
      if (pos < n) for (int s = 0; s < padSpaces; s++) part += " ";
      matrix.displayClear();
      matrix.displayReset();
      textEffect_t eff = (effectMode == 2) ? PA_SCROLL_UP : PA_SCROLL_DOWN;
      matrix.displayText(part.c_str(), PA_LEFT, 150, 0, eff, eff);
      while (!matrix.displayAnimate());
      delay(220);
    }
  } else {
    displayText(msg);
  }
}
```

```
void showTemperature() {
  int analogValue = analogRead(LM35_PIN);
  float voltage = analogValue * (5.0 / 1023.0);
  int temperature = voltage * 100;
  char buffer[10];
  sprintf(buffer, "%d*C", temperature);
  displayText(buffer);
}

void showManualAdjust() {
  char buffer[10];
  sprintf(buffer, "%02d:%02d", manualHour, manualMinute);
  displayText(buffer);
}
```

2.4 Programming flowchart

The operational logic of the system is detailed in the flowchart in Figure 2. The logic incorporates initialization, continuous data reading (DS1307, LM35, HC-05), processing, and updating the display based on the current mode and persistent settings retrieved from EEPROM.

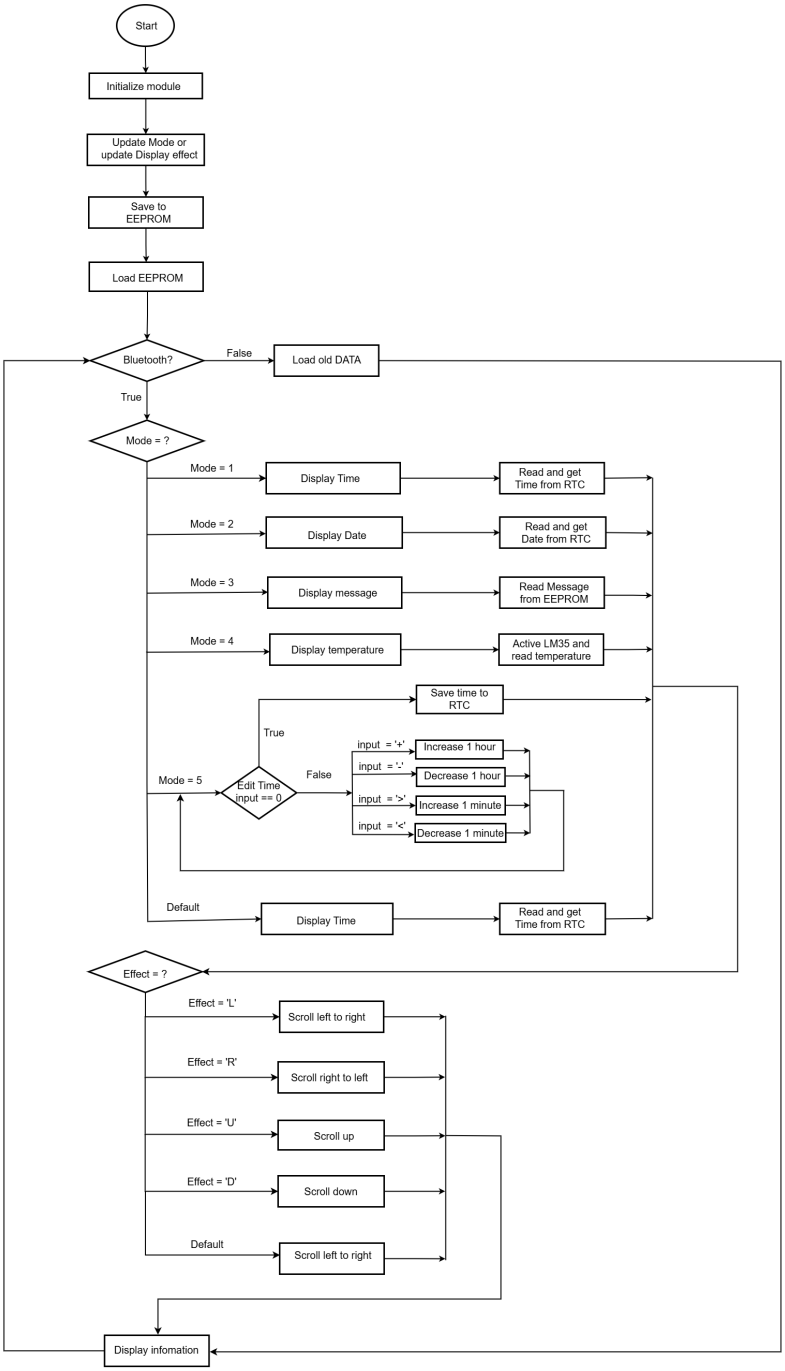

Figure 2: Programming flowchart of the developed system

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 4 Scroll Effects (Effects 1 to 4). 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 as shown in Table 2.

Table 2: Summary of Hardware and Software Testing Results

Feature Tested	**Component**	**Expected Result**	**Actual Result**	**Status**
RTC Accuracy	DS1307	Deviation $< 1\,\text{min/month}$	$\approx 1\,\text{min/month}$	Success
Temperature Reading	LM35	Reading must be within $\pm 0.5\,^\circ\text{C}$ of the reference thermometer.	Reading confirmed to be within $\pm 0.5\,^\circ\text{C}$ of the reference thermometer.	Success
Response Time	HC-05 & Arduino	Message update latency must be less than 1 s.	Message update latency observed to be less than 1 s.	Success
EEPROM Persistence	EEPROM	Configuration maintenance after 1 min power-off.	Configuration successfully retained after 1 min power-off.	Success
Mode & Effect Cycling	Arduino	System successfully executes all 5 defined Modes and 4 Scroll Effects.	All 5 Modes and 4 Scroll Effects executed successfully as defined.	Success

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 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 Rev3, HC-05, 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.

References

[1] *Arduino UNO Rev3 - Technical Specifications.* URL: `https://docs.arduino.cc/hardware/uno-rev3/`.

[2] *Component source code.* URL: `https://banlinhkien.com/`.

[3] *Refer to components.* URL: `https://hshop.vn/`.

Author's Contribution

Student ID	Student Name	Tasks Contribution
SE192861	Pham Tuan Anh	Write report, develop the Arduino code for system functionality (30%)
SE190124	Le Tuan Kiet	Develop the Arduino code for system functionality, assemble the circuit (30%)
SE190007	Truong Thao Vi	Assemble the circuit, draw the circuit schematic (25%)
SE190769	Dang Hong Phuoc	Draw block diagram and Flowchart, create presentation (20%)
	Total:	**100%**