2025 SPRING IOT102

Smart LED Matrix Display with Bluetooth Communication

1st Le Quoc Hoi, 2nd Le Chi Nhan, 3rd Nguyen Hung Thai, 4th Nguyen Thanh Dat, and Duc Ngoc Minh Dang
FPT University, Ho Chi Minh Campus, Vietnam
lequochoi7390@gmail.com, lechinhan18012005@gmail.com, hungthai1804@gmail.com, ndatyu11@gmail.com, ducdnm2@fe.edu.vn

Abstract

The Smart LED Matrix Display with Bluetooth Communication project is a real-time information display system that can be remotely controlled via smartphone using Bluetooth. The system uses the following main components:

- DS1307 RTC for accurate timekeeping.
- LM35 sensor to measure ambient temperature.
- HC-05 module for Bluetooth communication.
- Arduino as the central processing unit to control the entire system.
- EEPROM to store data without loss during power outages.
- LED matrix display to show time, temperature, and custom messages.

Users can easily update the displayed content remotely in real-time. The system is suitable for various applications such as personal notification boards, digital advertising, and smart home systems.

1 Introduction

In today's rapidly advancing technological era, the demand for real-time information display systems that are flexible, user-friendly, and easily accessible is growing significantly. However, traditional LED display systems still rely heavily on wired connections and manual control, which limit their ability to update content and personalize displays.

The development of wireless communication technologies, especially Bluetooth, has opened up new opportunities to improve the convenience and interactivity of display systems. Worldwide, various projects in countries such as the United States, Japan, and South Korea have demonstrated the effectiveness of integrating Bluetooth modules with LED displays in applications like smart homes, digital advertising, and public information

systems. These solutions often emphasize mobility, real-time content updating, and interaction through mobile devices, setting new standards for efficient information delivery.

In Vietnam, research and practical applications in wireless-controlled LED display systems remain relatively simple and have yet to meet the demand for remote control and personalized content management. This highlights the need for modern, accessible, and highly interactive solutions suitable for a variety of use cases, including educational institutions, public infrastructure, and personal environments.

The Smart LED Matrix Display with Bluetooth Communication project was developed to address these limitations by creating a real-time display system integrated with wireless control. By utilizing low-cost hardware components and a simple system architecture, the project offers a practical, scalable solution to meet the evolving demands of smart display technology both domestically and globally.

2 Main Proposal

2.1 System models and block diagram

The system is designed around a modular architecture. The flow of data is as follows:

- Smartphone/computer sends commands to Bluetooth module.
- Bluetooth module relays data to microcontroller.
- Microcontroller processes data from DS1307, LM - 35 and HC-05 sensor.
- Processed information is displayed on LED matrix.

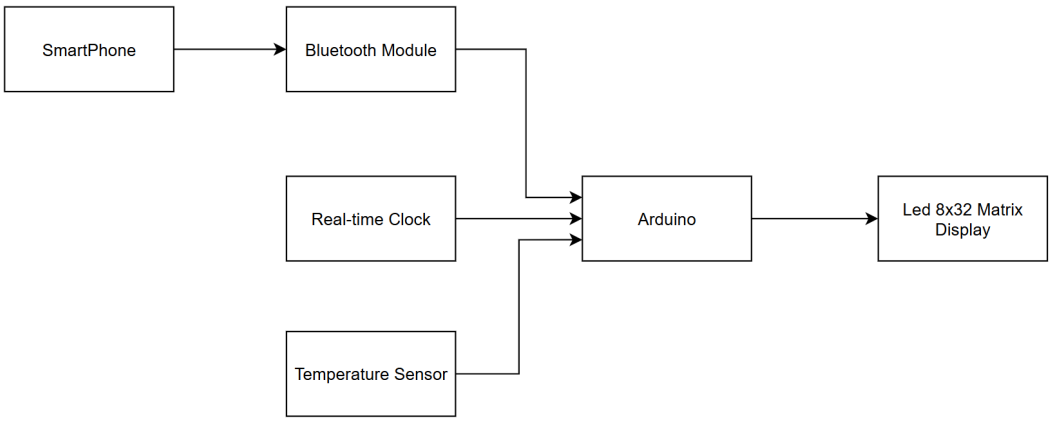

Fig. 1. Block diagram of the developed system

2.2 Components and peripheral devices

Component/Peripheral Device	**Function/Role in the System**
Arduino Uno R3	Acts as the central processor, receiving data and controlling peripherals.
LM35 Sensor	Measures ambient temperature for display.

HC-05 Bluetooth Module	Facilitates wireless communication for remote control via smartphone.
DS1307 RTC Module	Ensures accurate timekeeping and date tracking.
LED Matrix (MAX7219)	Visual output device displaying time, temperature, and messages with scrolling effects.
Smartphone	Remote control interface to send commands and update content.

2.3 Software programming

The software controls device initialization, mode switching, data display, and EEPROM management.

```
#include <Wire.h>
#include <RTClib.h>
#include <MD_Parola.h>
#include <MD_MAX72XX.h>
#include <SPI.h>
#include <SoftwareSerial.h>
#include <EEPROM.h>  // Include EEPROM library

#define HARDWARE_TYPE MD_MAX72XX::FC16_HW
#define MAX_DEVICES 4
#define CLK_PIN 13
#define DATA_PIN 11
#define CS_PIN 10
#define LM35_PIN A0

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

RTC_DS1307 rtc;
MD_Parola matrix = MD_Parola(HARDWARE_TYPE, DATA_PIN, CLK_PIN, CS_PIN,
    MAX_DEVICES);

String receivedData = "";
bool showTime = true, showTemperature = false, showDate = false;
String customMessage = "";
int directionRun = 1;

// EEPROM addresses
#define EEPROM_MODE_ADDR 0       // EEPROM Address to store mode
#define EEPROM_MSG_ADDR 1        // EEPROM Address for custom message
#define EEPROM_DIR_ADDR 2
#define MAX_MSG_LENGTH 50        // Max length of message

void setup() {
  Serial.begin(9600);
  bluetooth.begin(9600);

  if (!rtc.begin())
    if (!rtc.isrunning()) {
      rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
    }
```

```
42
43    matrix.begin();
44    matrix.setIntensity(5);
45    matrix.displayClear();
46
47    // Load mode and message from EEPROM
48    loadModeFromEEPROM();
49  }
50
51  void loop() {
52    while (bluetooth.available()) {
53      char c = bluetooth.read();
54      if (c == '\n') {
55        receivedData = "";
56      } else {
57        receivedData += c;
58        processCommand(receivedData);
59      }
60    }
61
62    receivedData = "";
63
64    if (showTime) displayTime();
65    if (showTemperature) displayTemperature();
66    if (showDate) displayDate();
67    if (customMessage != "") displayCustomMessage();
68  }
69
70  // ============================
71  // Function to process commands
72  // ============================
73  void processCommand(String command) {
74    if (command == "MODE1") {
75      showTime = true;
76      showTemperature = false;
77      showDate = false;
78      customMessage = "";
79      saveModeToEEPROM(1, customMessage);
80    } else if (command == "MODE2") {
81      showTime = false;
82      showTemperature = true;
83      showDate = false;
84      customMessage = "";
85      saveModeToEEPROM(2, customMessage);
86    } else if (command.indexOf("MODE3") != -1) {
87      showTime = false;
88      showTemperature = false;
89      showDate = false;
90      int index = command.indexOf("MODE3") + 5;
91      if (index < command.length()) {
92        customMessage = command.substring(index);
93        customMessage.trim();
94        saveModeToEEPROM(3, customMessage);
95      }
96    } else if (command.indexOf("MODE4") != -1) {
97      showTime = true;
98      showTemperature = false;
99      showDate = false;
```

```
100       int index = command.indexOf("MODE4") + 5;
101
102       String hour = command.substring(index, index + 2);
103       String min = command.substring(index + 3, index + 5);
104
105       DateTime now = rtc.now();
106       int receivedHour = hour.toInt();
107       int receivedMin = min.toInt();
108
109       // Check if the received time is different from the current RTC time
110       if (now.hour() != receivedHour || now.minute() != receivedMin) {
111         rtc.adjust(DateTime(now.year(), now.month(), now.day(),
              receivedHour, receivedMin, 0));
112         Serial.println("RTC Updated!");
113         bluetooth.println("RTC Updated!");
114       } else {
115         Serial.println("RTC Already Correct!");
116         bluetooth.println("RTC Already Correct!");
117       }
118
119       customMessage = "";
120       saveModeToEEPROM(4, customMessage);
121     } else if (command == "MODE5") {
122       showTime = false;
123       showTemperature = false;
124       showDate = true;
125       customMessage = "";
126       saveModeToEEPROM(5, customMessage);
127     }
128     else if (command.indexOf("MODES") != -1) {
129       int index = command.indexOf("MODES") + 5;
130       int temp = command.substring(index, index + 1).toInt();
131
132       if (temp >= 1 && temp <= 5) {
133         directionRun = temp;
134       } else {
135         directionRun = 1;
136       }
137     }
138   }
139
140   // ============================
141   // Function to display time
142   // ============================
143   void displayTime() {
144     static unsigned long lastSent = 0;
145     unsigned long nowMillis = millis();
146
147     if (nowMillis - lastSent >= 1000) {
148       lastSent = nowMillis;
149
150       DateTime now = rtc.now();
151       char timeStr[9];
152       sprintf(timeStr, "%02d:%02d", now.hour(), now.minute());
153
154       displayText(timeStr);
155       while (!matrix.displayAnimate()) {}
156     }
```

```
157 }
158
159 // ============================
160 // Function to display temperature
161 // ============================
162 void displayTemperature() {
163   int analogValue = analogRead(LM35_PIN);
164   float voltage = analogValue * (5.0 / 1023.0);
165   int temp = voltage * 100;
166
167   char temperatureStr[16];
168   snprintf(temperatureStr, sizeof(temperatureStr), "T:%dC", temp);
169
170   displayText(temperatureStr);
171   while (!matrix.displayAnimate()) {}
172   delay(3000);
173 }
174
175 // ============================
176 // Function to display message
177 // ============================
178 void displayCustomMessage() {
179   matrix.displayClear();
180
181   if (directionRun == 5) {
182     matrix.setTextAlignment(PA_CENTER);
183     matrix.print(customMessage.c_str());
184   } else {
185     matrix.displayText(customMessage.c_str(), PA_CENTER, 80, 0,
          getScrollEffect(), getScrollEffect());
186     while (!matrix.displayAnimate()) {}
187   }
188   delay(3000);
189 }
190
191 // ============================
192 // Function to display date
193 // ============================
194 void displayDate() {
195   static unsigned long lastSent = 0;
196   unsigned long nowMillis = millis();
197
198   if (nowMillis - lastSent >= 1000) {
199     lastSent = nowMillis;
200
201     DateTime now = rtc.now();
202     char dateStr[16];
203     sprintf(dateStr, "%02d/%02d", now.day(), now.month());
204
205     displayText(dateStr);
206     while (!matrix.displayAnimate()) {}
207   }
208   delay(3000);
209 }
210
211 textEffect_t getScrollEffect() {
212   switch (directionRun) {
213     case 1: return PA_SCROLL_LEFT;
```

```
    case 2: return PA_SCROLL_RIGHT;
    case 3: return PA_SCROLL_UP;
    case 4: return PA_SCROLL_DOWN;
    case 5: return PA_PRINT;
    default: return PA_SCROLL_LEFT;
  }
}

void displayText(String text) {
  matrix.displayClear();

  if (directionRun == 5) {
    matrix.setTextAlignment(PA_CENTER);
    matrix.print(text.c_str());
  } else {
    matrix.displayText(text.c_str(), PA_CENTER, 80, 0, getScrollEffect()
        , getScrollEffect());
    while (!matrix.displayAnimate()) {}
  }
}

// ============================
// Function to save mode & message to EEPROM
// ============================
void saveModeToEEPROM(int mode, String message) {
  EEPROM.write(EEPROM_MODE_ADDR, mode);
  for (int i = 0; i < MAX_MSG_LENGTH; i++) {
    EEPROM.write(EEPROM_MSG_ADDR + i, i < message.length() ? message[i]
        : '\0');
  }
  EEPROM.write(EEPROM_DIR_ADDR, directionRun);
}

// ============================
// Function to load mode & message from EEPROM
// ============================
void loadModeFromEEPROM() {
  int mode = EEPROM.read(EEPROM_MODE_ADDR);
  char message[MAX_MSG_LENGTH];
  for (int i = 0; i < MAX_MSG_LENGTH; i++) {
    message[i] = EEPROM.read(EEPROM_MSG_ADDR + i);
  }
  message[MAX_MSG_LENGTH - 1] = '\0';
  customMessage = String(message);

  directionRun = EEPROM.read(EEPROM_DIR_ADDR);

  showTime = (mode == 1 || mode == 4);
  showTemperature = (mode == 2);
  showDate = (mode == 5);
}
```

2.4 Programming Flowchart

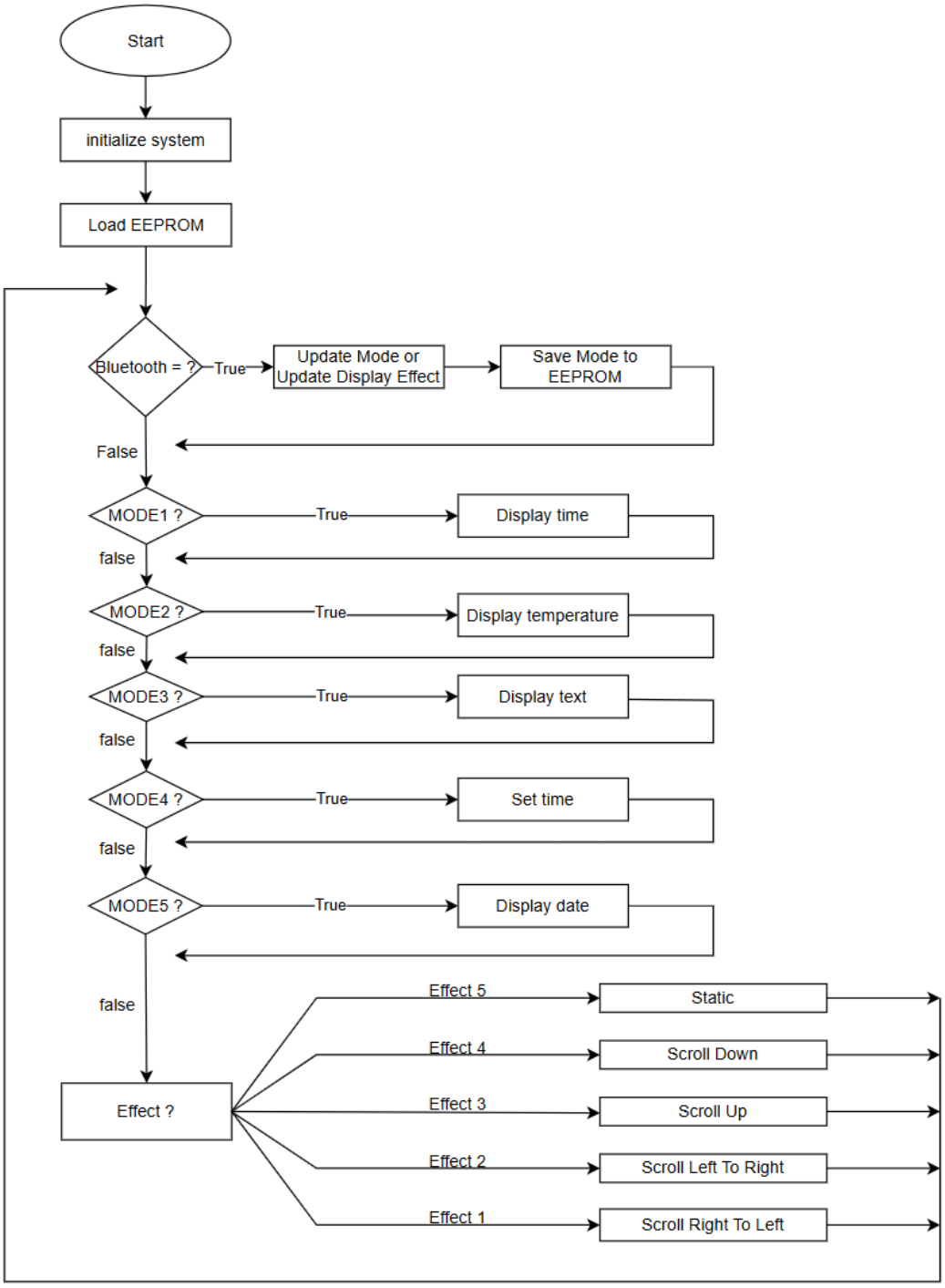

Fig. 2. Programming flowchart of the developed system

3 Results and Discussion

3.1 Prototype Implementation

The prototype of the Bluetooth-Controlled LED Matrix Display System was built by integrating both hardware and software components to achieve the desired functionality. The central processing unit of the system is the Arduino microcontroller, which interfaces

with all other components and manages the flow of data.

At the core of the system is the DS1307 Real-Time Clock (RTC) module, which ensures accurate timekeeping, enabling the system to display the current time. The system also includes the LM35 temperature sensor, which measures the environmental temperature and displays it on the LED matrix in real time.

The HC-05 Bluetooth module facilitates wireless communication, allowing the system to receive commands from a smartphone or other Bluetooth-enabled device. This allows users to interact with the system by sending commands to switch between modes, set the time, and display custom messages. The 8x32 LED matrix display serves as the output device for the system, showing the time, temperature, custom messages, date and supporting various scrolling effects (left-to-right, right-to-left, up, down, and static).

The system operates in several Modes, as follows:

- **Mode 1:** Displays the current time based on the RTC module.
- **Mode 2:** Displays the current temperature readings from the LM35 sensor.
- **Mode 3:** Displays custom messages sent from the Android app via Bluetooth.
- **Mode 4:** Allows the user to update the time of the DS1307 RTC module via Bluetooth commands.
- **Mode 5:** Displays the current date based on the RTC module.
- **Effect 1-5:** Allows for customization of scrolling effects (right-to-left, left-to-right, up, down or static).

To ensure that the system retains its last state after power is lost, the EEPROM on the Arduino stores the last selected mode and custom messages. When the system is powered back on, it restores the previous settings.

The system is controlled through a custom Android application developed using Kotlin and Android Studio. This application provides an easy-to-use interface, where users can interact with the system to:

- Switch between different display modes (time, temperature, custom messages, date).
- Send custom messages to the system via Bluetooth.
- Synchronize the displayed time with the smartphone's current time and set the RTC module accordingly.
- Adjust scrolling effects, including direction (right-to-left, left-to-right, up, down or static).

Bluetooth communication is managed using Android's built-in Bluetooth API, which sends simple string commands to the Arduino. The Arduino parses these commands and executes the corresponding actions, such as changing the display mode, setting the time, or updating the message on the LED matrix.

3.2 Testing Results

- **Functionality Testing**: All modes operated correctly:
  - **MODE 1**: Displayed the current time accurately.
  - **MODE 2**: Measured and displayed temperature from the LM35 sensor reliably.
  - **MODE 3**: Displayed custom messages entered via Bluetooth without errors.
  - **MODE 4**: Successfully updated the DS1307 RTC time via Bluetooth commands.
  - **MODE 5**: Displayed the date correctly.
  - **EFFECT 1-5**: Allowed smooth scrolling in multiple directions (right-to-left, left-to-right, up, down or static).
- **Bluetooth Communication Range**: Stable up to 10 meters in open space, 5–7 meters in obstructed environments.
- **Power Consumption**: Averaged 150 mA, peak 200 mA.
- **Data Reliability**: EEPROM successfully retained mode, messages and scroll effect after power loss.
- **User Experience**: The Android app was user-friendly, and users appreciated scrolling effect customization.

4 Conclusion

The Smart LED Matrix Display with Bluetooth Communication project successfully developed a Bluetooth-controlled LED matrix system for real-time display of time, date, temperature, and custom messages. By integrating Arduino Uno, HC-05 Bluetooth module, MAX7219 LED matrix, DS1307 RTC module, and LM-35 sensor, the system achieved accurate data display and reliable wireless control. EEPROM storage ensured that user settings and messages were retained after restarts. Despite minor challenges with Bluetooth signal interference and memory limitations, the prototype performed well. This project demonstrates the potential of combining IoT technologies with dynamic display systems for interactive and user-friendly applications.

Author's Contribution

Student ID	**Student Name**	**Tasks Contribution**
SE190044	Nguyen Hung Thai	Draw block diagram, draw flowchart, write report (25%)
SE190104	Le Chi Nhan	Assemble the circuit and develop the Arduino code for system functionality (25%)

SE190070	Le Quoc Hoi	Create presentation slides and develop the Arduino code for system functionality (25%)
SE190283	Nguyen Thanh Dat	Assemble the circuit, draw flowchart and write report (25%)
Total: 100%		