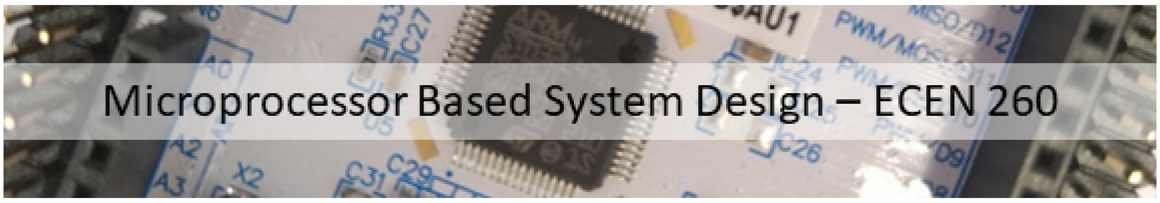

ECEN 260 - Final Project

Light Up Christmas Wreath

Brandon Ayton

Instructor: Brother Allred
December 16, 2024

Contents

List of Tables

List of Figures

1 Lab Overview

This project is a holiday-themed interactive system featuring a Christmas wreath equipped with 15 LEDs and a PCD8544 GLCD screen. The wreath's LEDs can display various lighting patterns, which are dynamically controlled based on user inputs sent from a laptop. The GLCD screen, interfaced through SPI communication, displays a festive message and the current LED mode, adding visual appeal. UART communication allows real-time control of the light modes, while interrupts are employed to promptly respond to user commands. This project effectively integrates SPI, UART, and interrupt-driven functionality to create a festive and engaging decoration that highlights technical design and interactivity.

2 Design

My Christmas Wreath project is designed to bring festive joy and interactiveness to holiday decorations. This system integrates three major engineering concepts: interrupts, UART communication, and SPI-based display. These elements combine to create a dynamic and engaging prototype that allows for customizable LED lighting patterns and user interaction through a laptop interface.

The primary need this project addresses is providing an interactive and customizable holiday decoration. The wreath features 15 LEDs controlled by the microcontroller, which can display various light modes, such as static lights and blinking patterns. An LCD screen connected via SPI displays a festive Christmas message alongside the current light mode, enhancing the visual appeal of the wreath. Additionally, the system includes a UART connection between the laptop and microcontroller, enabling the user to send commands to change the light modes. The microcontroller uses interrupts to detect incoming messages from the laptop, ensuring prompt response and efficient operation.

Public health, safety, and welfare were considered in the design to ensure the system operates reliably and without risk to users. For instance, the LEDs are low voltage and energy-efficient, minimizing fire hazards and promoting environmental sustainability. From a cultural and social perspective, the wreath aligns with global holiday traditions, making it an appealing decoration for diverse communities. The design's environmental impact was minimized by using energy-efficient components, while the economic factor was addressed by utilizing cost-effective materials, making the system affordable for potential users.

The integration of interrupts, UART, and SPI plays a critical role in the system. Interrupts allow the microcontroller to prioritize incoming commands, enabling real-time user interaction. UART communication facilitates the exchange of data between the laptop and the microcontroller, allowing users to customize the light modes effortlessly. The SPI-based LCD screen provides a clear and user-friendly display, enhancing the overall experience by communicating the current mode and festive message.

In summary, this project demonstrates how engineering principles can be applied to create a useful, enjoyable, and culturally relevant product. It showcases the potential for integrating technical concepts into a practical system while addressing considerations for public health, safety, and environmental sustainability.

3 Specifications

3.1 Technical Details

- Microcontroller: Nucleo-L476RG board
- Display: PCD8544 GLCD screen
- Lighting: 15 LEDs
- Communication Protocols:
  - SPI for interfacing with the display
  - UART for communication between laptop and the microcontroller
- Interrupts: Used to handle incoming UART messages with priority

3.2 Use Cases

- Festive home decoration
- Customizable holiday lighting patterns
- Interactive control via laptop interface

3.3 Operating Instructions

- Create circuit according to schematic 4
- Power the system by connecting the project board to a laptop through USB connection
- Build and send the code to the microcontroller through the STM IDE
- Install and run PuTTY on your laptop and set the following peripherals:
  - Under Session, change connection type to Serial
  - Open device manager and locate under Ports which COM port the microcontroller is connected to
  - Enter the COM port into the serial line box in PuTTY
  - Confirm the Speed to be 9600 and head over to the Terminal tab
  - Under Local echo and Local line editing, set both to Force on then hit Open at the bottom
- A terminal will open, allowing you to communicate with the project board by typing in a phrase and pressing enter to send it

- Enter any of the phrases listed below to change the LCD screen and light configuration (each command will be shown on the LCD screen and the PuTTY terminal will also respond confirming your message was received)
  - "RED_ON" - Turns on red LEDS
  - "WHITE_ON" - Turns on white LEDS
  - "GREEN_ON" - Turns on green LEDS
  - "RED_OFF" - Turns off red LEDS
  - "WHITE_OFF" - Turns off white LEDS
  - "GREEN_OFF" - Turns off green LEDS
  - "ALL_ON" - Turns on all LEDS
  - "ALL_OFF" - Turns off all LEDS
  - "BLINK_LEDS" - Blinks all LEDS in a short light show

3.4 Operating Constraints

- Power Supply: Requires a stable 5V USB power source connected to a laptop for communication
- Wiring: Wiring takes up a decent amount of space so cable management is necessary

3.5 Design Limitations

- The project is designed only for indoor use.
- Limited to a predefined light mode unless further programming is performed.
- UART communication requires a laptop with a compatible terminal emulator.

3.6 Parts List

- Christmas Wreath (1ft diameter)
- 1 Nucleo-L476RG board and USB cable
- Laptop with PuTTY and STM IDE installed
- PCD8544 GLCD screen
- 5 of each LED color:
  - Red
  - Green
  - White

- 15 100 ohm resistors
- Jumper wires

Assembly of parts shown in Schematic 4."

4 Schematics

See Figure 1 for the wiring schematic for this lab.

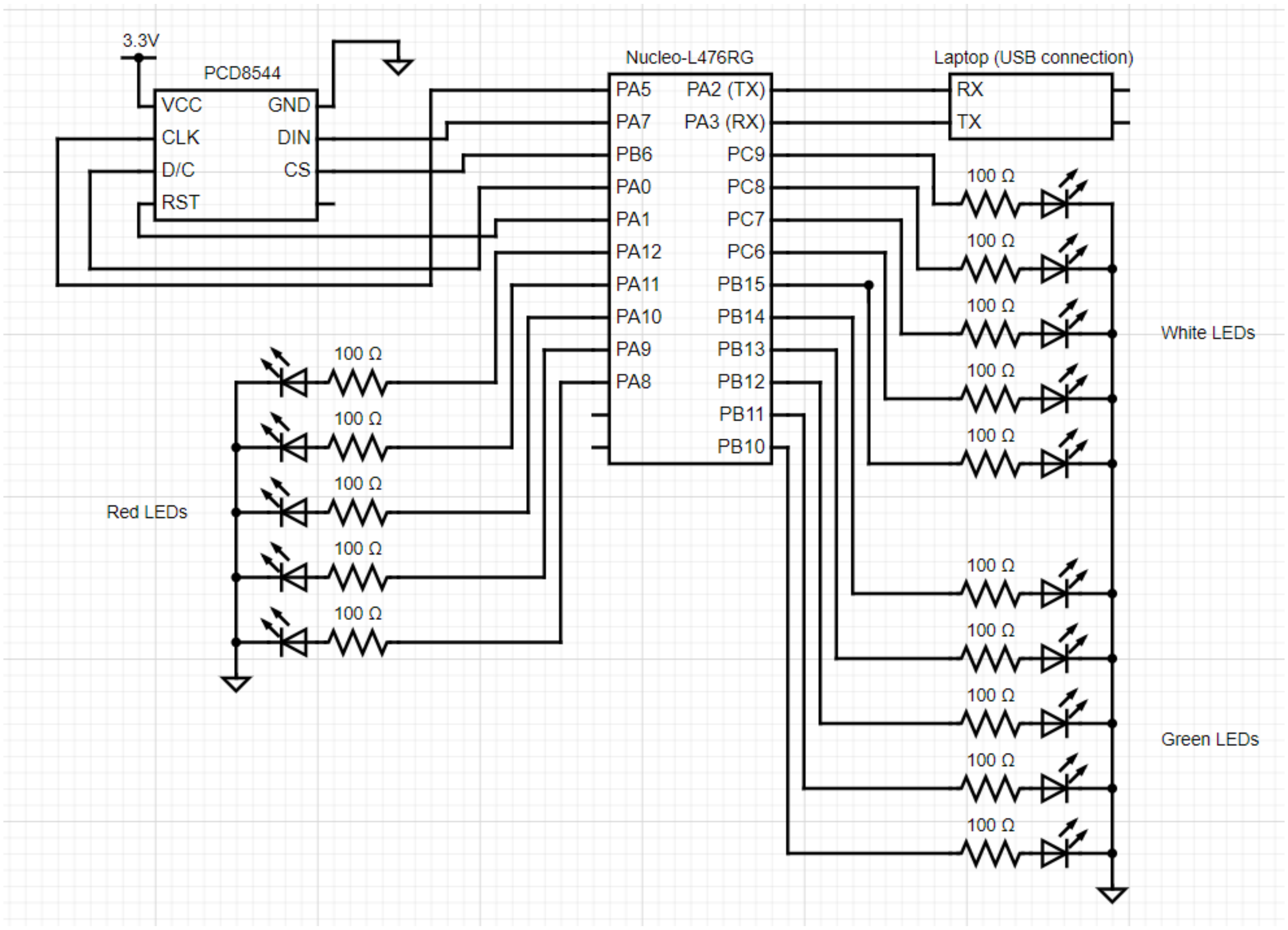

Figure 1: Schematic diagram for the Lab.

5 Test Plan and Test Results

The test plan covers the UART connection between the microcontroller and laptop to control the LEDs and the LCD screen. Video link to test cases and demo of project:
`https://youtu.be/SbXbb1d1-ug`

5.1 Test Plan Procedure

Test scenarios listed below.

- Test Scenario #1
  - In the PuTTY terminal:
    - Type in "RED_ON" and hit enter
    - Type in "WHITE_ON" and hit enter
    - Type in "GREEN_ON" and hit enter
- Test Scenario #2
  - In the PuTTY terminal:
    - Type in "RED_OFF" and hit enter
    - Type in "WHITE_OFF" and hit enter
    - Type in "GREEN_OFF" and hit enter
- Test Scenario #3
  - In the PuTTY terminal:
    - Type in "ALL_ON" and hit enter
    - Type in "ALL_OFF" and hit enter
- Test Scenario #4
  - In the PuTTY terminal:
    - Type in "BLINK_LEDS" and hit enter
    - Type in "ALL_OFF" and hit enter
- Test Scenario #5
  - In the PuTTY terminal:
    - Type in "RED_ON" and hit enter
    - Type in "BLINK_ON" and hit enter
    - Type in "GREEN_ON" and hit enter
- Test Scenario #6
  - In the PuTTY terminal:
    - Type in any command that isn't listed in the operating instructions 3.3 and hit enter

5.2 Expected and Observed Results

Expected and actual results from test scenarios below.

- Test Scenario #1
    - Expected Result: All the LEDs will turn on one color at a time with each command sent to the microcontroller, The LCD screen will display which command has been given to the microcontroller while displaying "MERRY XMAS! (smiley)" above it, and the PuTTY terminal will respond with a message confirming each command after each input.
    - Actual Result: Expected Result
- Test Scenario #2
    - Expected Result: All the LEDs will turn off one color at a time with each command sent to the microcontroller, The LCD screen will display which command has been given to the microcontroller while displaying "MERRY XMAS! (smiley)" above it, and the PuTTY terminal will respond with a message confirming each command after each input.
    - Actual Result: Expected Result
- Test Scenario #3
    - Expected Result: All the LEDs will turn on and off at the same time with each command sent to the microcontroller, The LCD screen will display which command has been given to the microcontroller while displaying "MERRY XMAS! (smiley)" above it, and the PuTTY terminal will respond with a message confirming each command after each input.
    - Actual Result: Expected Result
- Test Scenario #4
    - Expected Result: All the LEDs will blink in order, turn on, blink in order, turn off, then loop. After the ALL_OFF command is sent, all the LED's will turn off. The LCD screen will display which command has been given to the microcontroller while displaying "MERRY XMAS! (smiley)" above it, and the PuTTY terminal will respond with a message confirming each command after each input.
    - Actual Result: Expected Result
- Test Scenario #5
    - Expected Result: The red LEDS will all turn on, then all the LEDs will blink in order, turn on, blink in order, turn off, then loop. After the GREEN_ON command is sent, only the green and red LEDs will be turned on. The LCD screen will display which command has been given to the microcontroller while displaying "MERRY XMAS! (smiley)" above it, and the PuTTY terminal will respond with a message confirming each command after each input.

  - Actual Result: Expected Result

- Test Scenario #6
  - Expected Result: No change in the LEDs. The LCD screen and the putty terminal will display a message clarifying that the command was not recognized
  - Actual Result: Expected Result

6 Code

The code is separated into multiple parts:

- Setting up each character in an array. (Section6.1)
- While loop that loops the animation. (Section 6.2)
- Interrupt that is triggered from UART communication. (Section 6.3)

For the entire code for this project, it is on my GitHub with this link:
`https://github.com/brandonayton/Christmas-Wreath-C-Project`

6.1 Code for Character Array (LCD screen)

Array that holds the value of each character we want to display for the alphabet.

```c
const char font_table[][6] = {
 {0x00, 0x00, 0x00, 0x00, 0x00, 0x00}, // space
{0x7E, 0x11, 0x11, 0x11, 0x7E, 0x00}, // 'A'
{0x7F, 0x49, 0x49, 0x49, 0x36, 0x00}, // 'B'
{0x3E, 0x41, 0x41, 0x41, 0x22, 0x00}, // 'C'
{0x7F, 0x41, 0x41, 0x41, 0x3E, 0x00}, // 'D'
{0x7F, 0x49, 0x49, 0x41, 0x41, 0x00}, // 'E'
{0x7F, 0x09, 0x09, 0x09, 0x01, 0x00}, // 'F'
{0x3E, 0x41, 0x49, 0x49, 0x39, 0x00}, // 'G'
{0x7F, 0x08, 0x08, 0x08, 0x7F, 0x00}, // 'H'
{0x41, 0x41, 0x7F, 0x41, 0x41, 0x00}, // 'I'
{0x21, 0x41, 0x41, 0x3F, 0x01, 0x00}, // 'J'
{0x7F, 0x08, 0x14, 0x22, 0x41, 0x00}, // 'K'
{0x7F, 0x40, 0x40, 0x40, 0x40, 0x00}, // 'L'
{0x7F, 0x06, 0x78, 0x06, 0x7F, 0x00}, // 'M'
{0x7F, 0x06, 0x08, 0x30, 0x7F, 0x00}, // 'N'
{0x3E, 0x41, 0x41, 0x41, 0x3E, 0x00}, // 'O'
{0x7F, 0x11, 0x11, 0x11, 0x0E, 0x00}, // 'P'
{0x3E, 0x41, 0x51, 0x61, 0x3E, 0x00}, // 'Q'
{0x7F, 0x11, 0x11, 0x31, 0x4E, 0x00}, // 'R'
{0x46, 0x49, 0x49, 0x49, 0x31, 0x00}, // 'S'
{0x01, 0x01, 0x7F, 0x01, 0x01, 0x00}, // 'T'
{0x3F, 0x40, 0x40, 0x40, 0x3F, 0x00}, // 'U'
{0x0F, 0x30, 0x40, 0x30, 0x0F, 0x00}, // 'V'
{0x7F, 0x30, 0x08, 0x30, 0x7F, 0x00}, // 'W'
{0x63, 0x14, 0x08, 0x14, 0x63, 0x00}, // 'X'
{0x03, 0x04, 0x78, 0x04, 0x03, 0x00}, // 'Y'
{0x61, 0x51, 0x49, 0x45, 0x43, 0x00}, // 'Z'
{0x7E, 0x81, 0xB5, 0xA1, 0xA1, 0xB5}, // smile left
{0x81, 0x7E, 0x00, 0x00, 0x00, 0x00}, // smile right
{0x00, 0x00, 0x5F, 0x00, 0x00, 0x00}, // '!'
};
```

6.2 Code for While Loop

While loop that will toggle the LEDs in order.

```
while (1)
 {
   if (blink){

   //blink all LEDs
   HAL_GPIO_TogglePin(RED1_GPIO_Port, RED1_Pin);
       HAL_GPIO_TogglePin(RED2_GPIO_Port, RED2_Pin);
       HAL_GPIO_TogglePin(RED3_GPIO_Port, RED3_Pin);
       HAL_GPIO_TogglePin(RED4_GPIO_Port, RED4_Pin);
       HAL_GPIO_TogglePin(RED5_GPIO_Port, RED5_Pin);
       HAL_GPIO_TogglePin(GREEN1_GPIO_Port, GREEN1_Pin);
       HAL_GPIO_TogglePin(GREEN2_GPIO_Port, GREEN2_Pin);
       HAL_GPIO_TogglePin(GREEN3_GPIO_Port, GREEN3_Pin);
       HAL_GPIO_TogglePin(GREEN4_GPIO_Port, GREEN4_Pin);
       HAL_GPIO_TogglePin(GREEN5_GPIO_Port, GREEN5_Pin);
       HAL_GPIO_TogglePin(WHITE1_GPIO_Port, WHITE1_Pin);
       HAL_GPIO_TogglePin(WHITE2_GPIO_Port, WHITE2_Pin);
       HAL_GPIO_TogglePin(WHITE3_GPIO_Port, WHITE3_Pin);
       HAL_GPIO_TogglePin(WHITE4_GPIO_Port, WHITE4_Pin);
       HAL_GPIO_TogglePin(WHITE5_GPIO_Port, WHITE5_Pin);
       HAL_Delay(200);

       //Blink red LEDs
       HAL_GPIO_TogglePin(RED1_GPIO_Port, RED1_Pin);
       HAL_GPIO_TogglePin(RED2_GPIO_Port, RED2_Pin);
       HAL_GPIO_TogglePin(RED3_GPIO_Port, RED3_Pin);
       HAL_GPIO_TogglePin(RED4_GPIO_Port, RED4_Pin);
       HAL_GPIO_TogglePin(RED5_GPIO_Port, RED5_Pin);
       HAL_Delay(200);
       HAL_GPIO_TogglePin(RED1_GPIO_Port, RED1_Pin);
       HAL_GPIO_TogglePin(RED2_GPIO_Port, RED2_Pin);
       HAL_GPIO_TogglePin(RED3_GPIO_Port, RED3_Pin);
       HAL_GPIO_TogglePin(RED4_GPIO_Port, RED4_Pin);
       HAL_GPIO_TogglePin(RED5_GPIO_Port, RED5_Pin);
       HAL_Delay(200);

       //blink white LEDs
       HAL_GPIO_TogglePin(WHITE1_GPIO_Port, WHITE1_Pin);
       HAL_GPIO_TogglePin(WHITE2_GPIO_Port, WHITE2_Pin);
       HAL_GPIO_TogglePin(WHITE3_GPIO_Port, WHITE3_Pin);
       HAL_GPIO_TogglePin(WHITE4_GPIO_Port, WHITE4_Pin);
       HAL_GPIO_TogglePin(WHITE5_GPIO_Port, WHITE5_Pin);
       HAL_Delay(200);
       HAL_GPIO_TogglePin(WHITE1_GPIO_Port, WHITE1_Pin);
       HAL_GPIO_TogglePin(WHITE2_GPIO_Port, WHITE2_Pin);
       HAL_GPIO_TogglePin(WHITE3_GPIO_Port, WHITE3_Pin);
       HAL_GPIO_TogglePin(WHITE4_GPIO_Port, WHITE4_Pin);
       HAL_GPIO_TogglePin(WHITE5_GPIO_Port, WHITE5_Pin);
       HAL_Delay(200);

       //blink green LEDs
```

```c
        HAL_GPIO_TogglePin(GREEN1_GPIO_Port, GREEN1_Pin);
        HAL_GPIO_TogglePin(GREEN2_GPIO_Port, GREEN2_Pin);
        HAL_GPIO_TogglePin(GREEN3_GPIO_Port, GREEN3_Pin);
        HAL_GPIO_TogglePin(GREEN4_GPIO_Port, GREEN4_Pin);
        HAL_GPIO_TogglePin(GREEN5_GPIO_Port, GREEN5_Pin);
        HAL_Delay(200);
        HAL_GPIO_TogglePin(GREEN1_GPIO_Port, GREEN1_Pin);
        HAL_GPIO_TogglePin(GREEN2_GPIO_Port, GREEN2_Pin);
        HAL_GPIO_TogglePin(GREEN3_GPIO_Port, GREEN3_Pin);
        HAL_GPIO_TogglePin(GREEN4_GPIO_Port, GREEN4_Pin);
        HAL_GPIO_TogglePin(GREEN5_GPIO_Port, GREEN5_Pin);
        HAL_Delay(200);

    }
}
```

6.3 Code for Interrupt

Interrupt function code that will receive bytes through a UART connection and send them directly to the second STM32 micro controller connected, the bytes will be stored in an array, converted to a string when the code detects a new line was sent, then compare the string in the ensuing if statements to change the status of the output pins connected to leds, as well as change the LCD screen to display the current light mode being shown.

The code below shows the first instance of LED pins being altered, as well as the LCD screen being updated. After, each case is simplified.

```c
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart){

    // Check if byte received was on UART2 (from laptop)
    if (huart == &huart2){

        // If we get here, it means we received a byte from UART2 and it was placed in "uart2_byte"

        // If uart2_byte isn't \r, \n, or \0, it means the message isn't over yet
        if ((uart2_byte != '\r') && (uart2_byte != '\n') && (uart2_byte != '\0')){

            // Add uart2_byte to the message
            message[buffer_position] = uart2_byte;
            buffer_position++;

        } else {

            // If we get here, it means we received \r, \n, or \0 and the message is over

            // Check message
            if (strcmp((char*)message, "RED_ON") == 0) {

                // If "RED_ON\n" was the message received -> Turn on Red LEDs
                HAL_GPIO_WritePin(RED1_GPIO_Port, RED1_Pin, GPIO_PIN_SET);
                HAL_GPIO_WritePin(RED2_GPIO_Port, RED2_Pin, GPIO_PIN_SET);
                HAL_GPIO_WritePin(RED3_GPIO_Port, RED3_Pin, GPIO_PIN_SET);
                HAL_GPIO_WritePin(RED4_GPIO_Port, RED4_Pin, GPIO_PIN_SET);
                HAL_GPIO_WritePin(RED5_GPIO_Port, RED5_Pin, GPIO_PIN_SET);
                blink = false;

                //Change the LCD display
                GLCD_clear(); // clear the screen
                //row 1
                GLCD_putchar(0); // space
                GLCD_putchar(0); // space
                GLCD_putchar(0); // space
                GLCD_putchar(0); // space
                GLCD_putchar(0); // space
                GLCD_putchar(0); // space
                GLCD_putchar(0); // space
                GLCD_putchar(0); // space
                GLCD_putchar(0); // space
                GLCD_putchar(0); // space
                GLCD_putchar(0); // space
                GLCD_putchar(0); // space
                GLCD_putchar(0); // space
                GLCD_putchar(0); // space
                //row 2
                GLCD_putchar(13); // 'M'
                GLCD_putchar(5); // 'E'
                GLCD_putchar(18); // 'R'
                GLCD_putchar(18); // 'R'
                GLCD_putchar(25); // 'Y'
                GLCD_putchar(0); // space
                GLCD_putchar(24); // 'X'
                GLCD_putchar(13); // 'M'
                GLCD_putchar(1); // 'A'
                GLCD_putchar(19); // 'S'
                GLCD_putchar(29); // '!'
                GLCD_putchar(27); // left smiley
                GLCD_putchar(28); // right smiley
                GLCD_putchar(0); // space
                //row 3
                GLCD_putchar(0); // space
                GLCD_putchar(0); // space
                GLCD_putchar(0); // space
                GLCD_putchar(0); // space
                GLCD_putchar(0); // space
                GLCD_putchar(0); // space
                GLCD_putchar(0); // space
                GLCD_putchar(0); // space
                GLCD_putchar(0); // space
                GLCD_putchar(0); // space
                GLCD_putchar(0); // space
                GLCD_putchar(0); // space
                GLCD_putchar(0); // space
                GLCD_putchar(0); // space
                //row 4
                GLCD_putchar(0); // space
                GLCD_putchar(0); // space
                GLCD_putchar(0); // space
                GLCD_putchar(0); // space
                GLCD_putchar(18); // 'R'
                GLCD_putchar(5); // 'E'
                GLCD_putchar(4); // 'D'
                GLCD_putchar(0); // space
                GLCD_putchar(15); // 'O'
                GLCD_putchar(14); // 'N'
                GLCD_putchar(0); // space
                GLCD_putchar(0); // space
                GLCD_putchar(0); // space
                GLCD_putchar(0); // space

                // Prepare response message
                strncpy((char*)response, "Red LEDs turned on.\n\r", MAX_MESSAGE_SIZE);

            } else if (strcmp((char*)message, "RED_OFF") == 0) {

                // If "RED_OFF\n" was the message received -> Turn off Red LEDs

                //Code to change LED pins

                //Code to update LCD screen

                // Prepare response message
                strncpy((char*)response, "Red LEDs turned off.\n\r", MAX_MESSAGE_SIZE);

            } else if (strcmp((char*)message, "GREEN_ON") == 0) {

                // If "GREEN_ON\n" was the message received -> Turn on GREEN LEDs

                //Code to change LED pins

                //Code to update LCD screen

                // Prepare response message
                strncpy((char*)response, "Green LEDs turned on.\n\r", MAX_MESSAGE_SIZE);

            } else if (strcmp((char*)message, "GREEN_OFF") == 0) {

                // If "GREEN_OFF\n" was the message received -> Turn off GREEN LEDs

                //Code to change LED pins

                //Code to update LCD screen
```

```c

            // Prepare response message
            strncpy((char*)response, "Green LEDs turned off.\n\r", MAX_MESSAGE_SIZE);

        } else if (strcmp((char*)message, "WHITE_ON") == 0) {

            // If "WHITE_ON\n" was the message received -> Turn on WHITE LEDs

            //Code to change LED pins

            //Code to update LCD screen

            // Prepare response message
            strncpy((char*)response, "White LEDs turned on.\n\r", MAX_MESSAGE_SIZE);

        } else if (strcmp((char*)message, "WHITE_OFF") == 0) {

            // If "WHITE_OFF\n" was the message received -> Turn off WHITE LEDs

            //Code to change LED pins

            //Code to update LCD screen

            // Prepare response message
            strncpy((char*)response, "White LEDs turned off.\n\r", MAX_MESSAGE_SIZE);

        } else if (strcmp((char*)message, "ALL_ON") == 0) {

            // If "ALL_ON\n" was the message received -> Turn on all LEDs

            //Code to change LED pins

            //Code to update LCD screen

            // Prepare response message
            strncpy((char*)response, "All LEDs turned on.\n\r", MAX_MESSAGE_SIZE);

        } else if (strcmp((char*)message, "ALL_OFF") == 0) {

            // If "ALL_OFF\n" was the message received -> Turn off all LEDs

            //Code to change LED pins

            //Code to update LCD screen

            // Prepare response message
            strncpy((char*)response, "All LEDs turned off.\n\r",
MAX_MESSAGE_SIZE);

            } else if (strcmp((char*)message, "BLINK_LEDS") == 0) {

                // If "BLINK_LEDS" was the message received -> tell main() to blink all LEDs
                blink = true;

                //Code to update LCD screen

                // Prepare response message
                strncpy((char*)response, "LEDs are now blinking.\n\r", MAX_MESSAGE_SIZE);

            } else {

                //Code to update LCD screen

                // Else message was not recognized
                strncpy((char*)response, "Task not recognized.\n\r", MAX_MESSAGE_SIZE);
            }

            // Send the response message to laptop
            HAL_UART_Transmit(&huart2, response, strlen(response), UART_DELAY);

            // Zero out message array and response array to get ready for the next message
            memset(message,  0, sizeof(message));
            memset(response, 0, sizeof(response));
            buffer_position = 0;
        }

        // Restart UART2's receive interrupt to wait for next byte
        HAL_UART_Receive_IT(&huart2, &uart2_byte, 1); //start next byte receive interrupt
    }
}
```

7 Conclusion

The system used the Nucleo microcontroller (STM32L4 series) to manage both the SPI and UART communication. Fifteen LEDs were connected to the microcontroller through GPIO pins, and their behavior was modified based on the commands received from the laptop. Commands were transmitted over UART, and the microcontroller used interrupts to immediately respond to incoming messages. The PCD8544 display received graphical updates through SPI, showing real-time information about the LED modes and festive messages.

This project provided a hands-on opportunity to integrate key concepts learned in class, such as synchronous communication using SPI, asynchronous communication with UART, and the use of interrupts to handle real-time events. Previous labs laid the foundation for understanding these concepts individually, but this project combined them into a single functional system. For example, prior experience with UART communication helped streamline the design of the command system, while earlier exposure to SPI ensured efficient data transfer to the display. Implementing these concepts together reinforced the importance of combining different communication protocols and techniques for real-world applications.

There were a few challenges encountered during the project. Mapping characters onto the PCD8544 display required precise placement of bytes in the array. To simplify this process, I used a visual map to identify the correct array indices for each character. Debugging the display was straightforward, as running the code immediately revealed whether the output was rendered correctly. Another challenge was ensuring that UART interrupts triggered reliably and efficiently. By carefully managing the UART interrupt routine, I was able to process commands in real time without affecting other system functions.

Overall, this project was a significant learning experience. I gained a deeper understanding of SPI and UART communication and learned how to use interrupts to enhance system responsiveness. These skills will be valuable for future projects, particularly where real-time data handling and display updates are required. Moving forward, I hope to expand on this project by exploring additional display types, incorporating PWM for LED brightness control, and integrating more advanced user interfaces. This project has not only strengthened my technical abilities but also demonstrated how multiple course concepts can be applied to create an engaging and useful system.