ECEN 260 - Laboratory #11

Display Lab

Brodric Young

Instructor: Brother Allred
March 21, 2025

Contents

List of Tables

List of Figures

1 Lab Overview

In this lab, I learned how to use display peripherals, specifically focusing on displaying characters on both a graphics LCD (GLCD) and a character LCD. The primary objectives were to gain hands-on experience with the SPI and I2C communication protocols and to prepare for the final project by mastering these essential techniques. This lab provided a comprehensive introduction to interfacing with peripheral devices, setting a solid foundation for future projects and coursework.

1.1 Objectives

- Gain more insight and practice using peripherals
- Learn how to display characters on a graphics LCD (GLCD) and character LCD
- Learn how to use the SPI and I2C communication protocols.
- Prepare for the final project.

2 Specifications

The system is designed to display characters on two types of LCDs: the PCD8544 GLCD and the 1602 LCD with an I2C daughterboard. The specific requirements and functionalities of the system include initializing the PCD8544 GLCD using the SPI protocol to display characters and messages, supporting basic commands to set the cursor position, clear the screen, and adjust the contrast. Similarly, the system initializes the 1602 LCD using the I2C protocol to display characters and messages, with commands to set the cursor position, clear the screen, and control the backlight.

To operate the system, first set up the hardware by connecting the Nucleo-L476RG [1] board to the computer and connecting the PCD8544 GLCD and 1602 LCD to the Nucleo board for the respective SPI and I2C connections. For the PCD8544 GLCD, I configured the SPI pins (PA5 for SCK, PA7 for MOSI, PB6 for SS, PA0 for DC, and PA1 for RST) and for the 1602 LCD, I configured the I2C pins (PB8 for SCL and PB9 for SDA). I used the provided functions to send data and commands to both displays, writing characters to the screens by sending the appropriate data bytes. Troubleshooting involved adjusting the contrast settings for the PCD8544 GLCD if the characters were not clearly visible.

Known limitations of the system include potential variations in screen contrast settings for different PCD8544 GLCD screens, which may require adjusting the Vop and bias mode values to achieve optimal visibility. Additionally, some screens may require 5V instead of 3.3V for the backlight to function correctly, necessitating experimentation with voltage levels.

2.1 Parts List

- Nucleo-L476RG board and USB cable
- PCD8544 GLCD screen
- 1602 LCD with I2C daughterboard
- several jumper wires

For specifics on the wiring setup, see the schematics in Section 3.

3 Schematics

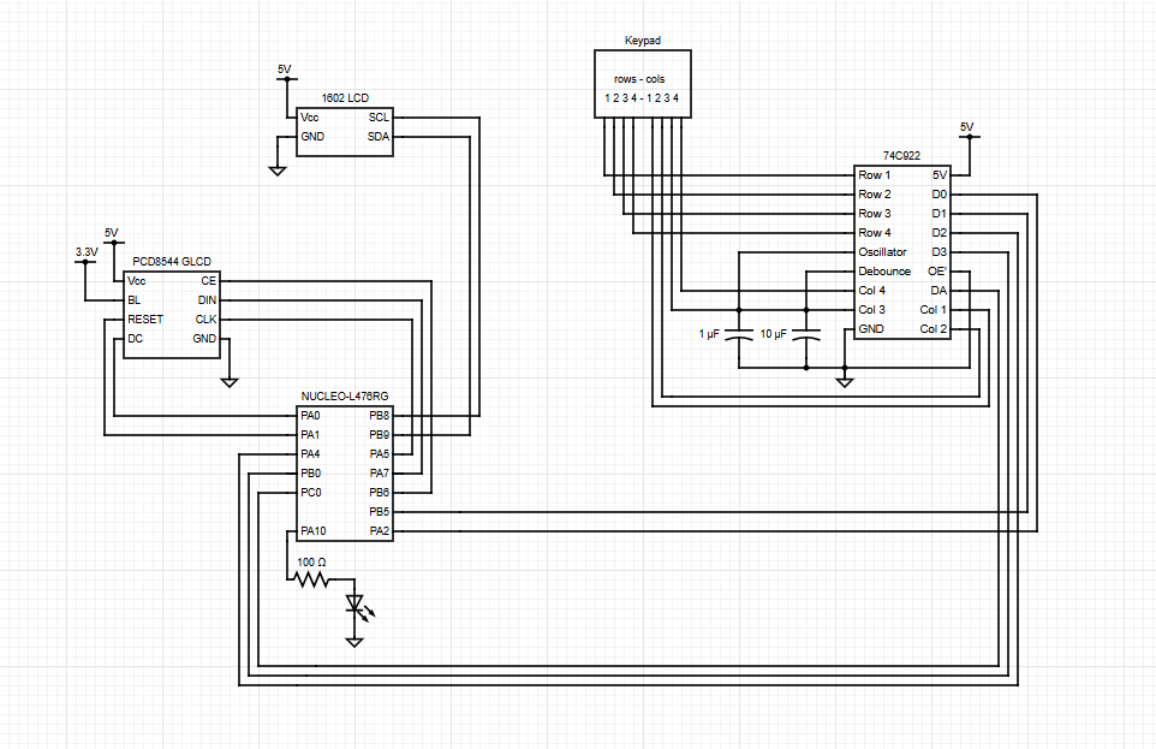

Figure 1: Complete schematic diagram for the lab.

4 Test Plan and Test Results

This test plan is short because there's not many ways to test the displays. Either they display what they should or the don't. This test plan will not be as detailed as to list off each pixel that should be on, but will verify the display generally looks as it should.

4.1 Test Plan Procedure

Table 1: Test plan with expected and observed results

Case	**Expected Results**	**Actual Results**
1	GLCD screen should display "ABC"	GLCD screen displays "ABC"
2	GLCD screen should display "HELLO WORLD" then a smiley face	GLCD screen displays "HELLO WORLD" then a smiley face
3	LCD screen should display "Hello, world!"	LCD screen displays "Hello, world!"
4	GLCD screen should show an animation of a stick figure and hoop with a basketball moving through the air and into the hoop.	GLCD screen displays the animation
5	Press each character on the keypad, the LCD screen should display each character as they're pressed one after the other	LCD screen displays each character as they're pressed

5 Code

This is where you write the text for this section. You could mention something about the code in Section 5.4 having the code for main.c.

5.1 Alphabet font table

```
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, 0x49, 0x41, 0x00}, // 'E'
  {0x7F, 0x09, 0x09, 0x09, 0x01, 0x00}, // 'F'
  {0x3E, 0x41, 0x49, 0x49, 0x7A, 0x00}, // 'G'
  {0x7F, 0x08, 0x08, 0x08, 0x7F, 0x00}, // 'H'
  {0x41, 0x41, 0x7F, 0x41, 0x41, 0x00}, // 'I'
  {0x20, 0x40, 0x40, 0x40, 0x3F, 0x00}, // 'J'
  {0x7F, 0x08, 0x14, 0x22, 0x41, 0x00}, // 'K'
  {0x7F, 0x40, 0x40, 0x40, 0x40, 0x00}, // 'L'
  {0x7F, 0x02, 0x0C, 0x02, 0x7F, 0x00}, // 'M'
  {0x7F, 0x04, 0x08, 0x10, 0x7F, 0x00}, // 'N'
  {0x3E, 0x41, 0x41, 0x41, 0x3E, 0x00}, // 'O'
  {0x7F, 0x09, 0x09, 0x09, 0x06, 0x00}, // 'P'
  {0x3E, 0x41, 0x51, 0x61, 0x7E, 0x00}, // 'Q'
  {0x7F, 0x09, 0x19, 0x29, 0x46, 0x00}, // 'R'
  {0x26, 0x49, 0x49, 0x49, 0x32, 0x00}, // 'S'
  {0x01, 0x01, 0x7F, 0x01, 0x01, 0x00}, // 'T'
  {0x3F, 0x40, 0x40, 0x40, 0x3F, 0x00}, // 'U'
  {0x1F, 0x20, 0x40, 0x20, 0x1F, 0x00}, // 'V'
  {0x3F, 0x40, 0x38, 0x40, 0x3F, 0x00}, // 'W'
  {0x63, 0x14, 0x08, 0x14, 0x63, 0x00}, // 'X'
  {0x03, 0x04, 0x78, 0x04, 0x03, 0x00}, // 'Y'
  {0x61, 0x51, 0x49, 0x45, 0x43, 0x00}, // 'Z'
  {0x00, 0x00, 0x5F, 0x00, 0x00, 0x00}, // '!'
  {0x00, 0x00, 0x7E, 0x81, 0xB5, 0xA1}, // left half of smiley face
  {0xA1, 0xB5, 0x81, 0x7E, 0x00, 0x00}, // right half of smiley face
};
```

5.2 Animation font table

```
const char font_table[][8] = {
  {0x00, 0x80, 0x40, 0x20, 0x20, 0x20, 0x40, 0x80}, // person top
  {0x00, 0x0F, 0x10, 0x2D, 0xE8, 0x2D, 0x90, 0x4F}, // person middle
  {0x00, 0x04, 0x82, 0x41, 0x3F, 0x41, 0x80, 0x00}, // person bottom
  {0x20, 0x10, 0x08, 0x06, 0x09, 0x09, 0x06, 0x00}, // person arm with ball
  {0x20, 0x10, 0x08, 0x00, 0x00, 0x00, 0x00, 0x00}, // person arm no ball
  {0x00, 0x18, 0x24, 0x24, 0x18, 0x00, 0x00, 0x00}, // ball phase 1
  {0x30, 0x48, 0x48, 0x30, 0x00, 0x00, 0x00, 0x00}, // ball phase 2
  {0x00, 0x00, 0x00, 0x00, 0x60, 0x90, 0x90, 0x60}, // ball phase 3,4,7,10-12
```

```
  {0x00, 0x00, 0x00, 0x00, 0x06, 0x09, 0x09, 0x06}, // ball phase 5
  {0x00, 0x00, 0x00, 0x00, 0x0C, 0x12, 0x12, 0x0C}, // ball phase 6
  {0x00, 0x00, 0x00, 0xE6, 0x19, 0x19, 0x16, 0x10}, // ball top hoop
  {0x00, 0x00, 0x00, 0x01, 0x32, 0x4A, 0x4A, 0x32}, // ball bottom hoop
  {0x00, 0xE0, 0xE0, 0x00, 0x00, 0xFF, 0x00, 0x00}, // hoop top
  {0x00, 0x00, 0x00, 0xE0, 0x10, 0x10, 0x10, 0x10}, // hoop mid left
  {0xE0, 0xFF, 0xFF, 0x10, 0x10, 0x1F, 0x00, 0x00}, // hoop mid right
  {0x00, 0x00, 0x00, 0x01, 0x02, 0x02, 0x02, 0x02}, // hoop bottom left
  {0x01, 0x1F, 0x1F, 0x00, 0x00, 0x00, 0x00, 0x00}, // hoop bottom right
  {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}, // clear bank
};
```

5.3 Animation display

```
while (1)
{
  GLCD_putchar(0, 9, 12);  // hoop top
  GLCD_putchar(1, 8, 13);  // hoop mid left
  GLCD_putchar(1, 9, 14);  // hoop mid right
  GLCD_putchar(2, 8, 15);  // hoop bottom left
  GLCD_putchar(2, 9, 16);  // hoop bottom right
  GLCD_putchar(3, 0, 0);  // person top
  GLCD_putchar(4, 0, 1);  // person middle
  GLCD_putchar(5, 0, 2);  // person bottom
  GLCD_putchar(4, 1, 3);  // person arm with ball
  HAL_Delay(500);
  GLCD_putchar(4, 1, 4);  // person arm no ball
  GLCD_putchar(3, 2, 5);  // ball phase 1
  HAL_Delay(500);
  GLCD_putchar(3, 2, 17);  // clear ball phase 1
  GLCD_putchar(2, 3, 5);  // ball phase 2
  HAL_Delay(500);
  GLCD_putchar(2, 3, 17);  // clear ball phase 2
  GLCD_putchar(1, 3, 7);  // ball phase 3
  HAL_Delay(500);
  GLCD_putchar(1, 3, 17);  // clear ball phase 3
  GLCD_putchar(0, 4, 7);  // ball phase 4
  HAL_Delay(500);
  GLCD_putchar(0, 4, 17);  // clear ball phase 4
  GLCD_putchar(0, 5, 8);  // ball phase 5
  HAL_Delay(500);
  GLCD_putchar(0, 5, 17);  // clear ball phase 5
  GLCD_putchar(0, 6, 9);  // ball phase 6
  HAL_Delay(500);
  GLCD_putchar(0, 6, 17);  // clear ball phase 6
  GLCD_putchar(0, 7, 7);  // ball phase 7
  HAL_Delay(500);
  GLCD_putchar(0, 7, 17);  // clear ball phase 7
  GLCD_putchar(1, 8, 10);  // ball phase 8 top hoop
  HAL_Delay(500);
  GLCD_putchar(1, 8, 13);  // clear ball phase 8 top hoop
  GLCD_putchar(2, 8, 11);  // ball phase 9 bottom hoop
```

```
  HAL_Delay(500);
  GLCD_putchar(2, 8, 15);  // clear ball phase 9 bottom hoop
  GLCD_putchar(3, 8, 7);  // ball phase 10
  HAL_Delay(500);
  GLCD_putchar(3, 8, 17);  // clear ball phase 10
  GLCD_putchar(4, 8, 7);  // ball phase 11
  HAL_Delay(500);
  GLCD_putchar(4, 8, 17);  // clear ball phase 11
  GLCD_putchar(5, 8, 7);  // ball phase 12
  HAL_Delay(500);
  GLCD_putchar(5, 8, 17);  // clear ball phase 12
}
```

5.4 Key press interrupt LCD display

```
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
    if (HAL_GPIO_ReadPin(DATA_AVAILABLE_GPIO_Port, DATA_AVAILABLE_Pin)) {
      // Turn on LED2 to show key press detected
      HAL_GPIO_WritePin(KEY_LED_GPIO_Port, KEY_LED_Pin, 1); // turn on the
    KEY_LED (LED2)

      uint8_t key = keypad_decode();  // determine which key was pressed (key
    is in ascii)
      lcd_send_data(key); // Send ascii key data to the LCD

      // wait for user to release key, then turn off LED2
     while (HAL_GPIO_ReadPin(DATA_AVAILABLE_GPIO_Port, DATA_AVAILABLE_Pin) ==
    GPIO_PIN_SET);
     HAL_GPIO_WritePin(KEY_LED_GPIO_Port, KEY_LED_Pin, 0); // turn off the
    KEY_LED (LED2)
    }
}
```

6 Conclusion

Completing this lab was a rewarding experience. I successfully interfaced with both the PCD8544 GLCD and the 1602 LCD using the SPI and I2C protocols. I displayed characters, messages, and animations on these screens, which enhanced my understanding of synchronous serial communication. This lab also provided valuable insights into configuring and troubleshooting peripheral devices, skills that are crucial for my final project. Through this process, I gained confidence in my ability to work with different communication protocols and peripheral devices, which will be beneficial for future projects and coursework.

This lab built on the knowledge I gained from previous labs, particularly the use of communication. By introducing SPI and I2C protocols, I expanded my understanding of synchronous serial communication, which is vital for interfacing with various peripheral devices. The skills I acquired in this lab are directly applicable to my final project which will use both these displays and future coursework. I had some issues getting the correct characters to be displayed on the LCD from the corresponding key presses on the keypad, but in the end after getting some help I got the decoder functioning properly to display the proper characters. There was also occasionally some issues with connections in the breadboard where I had to adjust the wires going in to make a good connection.

References

[1] "Stm32 nucleo-64 development board with stm32l476rg mcu." https://www.st.com/en/evaluation-tools/nucleo-l476rg.html. Accessed: 2023-06-21.