John Danison

ECET 32900 – Lab 6

03/07/2025

**Goal**  
The goal of this lab is to work with the Nucleo board and get two separate projects working:

1. An alternating blinking effect with two different LEDs on two different GPIO pins.
2. Put a string of characters on a 16 x 2 LCD.

**Activity:**

The assigned activities for this lab were to develop handwritten flowcharts and wiring diagrams for each of the projects described in the goals section. Then we must demonstrate to the course instructor that each of the LEDs were flashing alternatively, and the text displayed on the LCD was working properly.

**Output:**

Professional Wiring Diagrams:

Alternative Flashing:

A diagram of a measuring device

AI-generated content may be incorrect.

LCD Screen:

A diagram of a measuring device

AI-generated content may be incorrect.

Professional Flowcharts:

Alternative Flashing:

A diagram of a flowchart

AI-generated content may be incorrect.

LCD Screen:

A diagram of a process

AI-generated content may be incorrect.

Source Code:

Alternative Flashing:

/\*\*

YOU NEED TO CONFIGURE PB\_6 and PB\_7 AS OUTPUTS BEFORE RUNNING THIS CODE!

\*\*/

/\* USER CODE BEGIN Header \*/

/\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @file : main.c

\* @brief : Main program body

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @attention

\*

\* Copyright (c) 2025 STMicroelectronics.

\* All rights reserved.

\*

\* This software is licensed under terms that can be found in the LICENSE file

\* in the root directory of this software component.

\* If no LICENSE file comes with this software, it is provided AS-IS.

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

/\* USER CODE END Header \*/

/\* Includes ------------------------------------------------------------------\*/

#include "main.h"

/\* Private includes ----------------------------------------------------------\*/

/\* USER CODE BEGIN Includes \*/

/\* USER CODE END Includes \*/

/\* Private typedef -----------------------------------------------------------\*/

/\* USER CODE BEGIN PTD \*/

/\* USER CODE END PTD \*/

/\* Private define ------------------------------------------------------------\*/

/\* USER CODE BEGIN PD \*/

/\* USER CODE END PD \*/

/\* Private macro -------------------------------------------------------------\*/

/\* USER CODE BEGIN PM \*/

/\* USER CODE END PM \*/

/\* Private variables ---------------------------------------------------------\*/

UART\_HandleTypeDef huart2;

/\* USER CODE BEGIN PV \*/

/\* USER CODE END PV \*/

/\* Private function prototypes -----------------------------------------------\*/

void SystemClock\_Config(void);

static void MX\_GPIO\_Init(void);

static void MX\_USART2\_UART\_Init(void);

/\* USER CODE BEGIN PFP \*/

/\* USER CODE END PFP \*/

/\* Private user code ---------------------------------------------------------\*/

/\* USER CODE BEGIN 0 \*/

/\* Define the GPIO Pins Used for LEDs \*/

#define myled1 GPIO\_PIN\_6

#define myled2 GPIO\_PIN\_7

/\* USER CODE END 0 \*/

/\*\*

\* @brief The application entry point.

\* @retval int

\*/

int main(void)

{

/\* USER CODE BEGIN 1 \*/

/\* USER CODE END 1 \*/

/\* MCU Configuration--------------------------------------------------------\*/

/\* Reset of all peripherals, Initializes the Flash interface and the Systick. \*/

HAL\_Init();

/\* USER CODE BEGIN Init \*/

/\* USER CODE END Init \*/

/\* Configure the system clock \*/

SystemClock\_Config();

/\* USER CODE BEGIN SysInit \*/

/\* USER CODE END SysInit \*/

/\* Initialize all configured peripherals \*/

MX\_GPIO\_Init();

MX\_USART2\_UART\_Init();

/\* USER CODE BEGIN 2 \*/

/\* USER CODE END 2 \*/

/\* Infinite loop \*/

/\* USER CODE BEGIN WHILE \*/

while (1)

{

/\* USER CODE END WHILE \*/

HAL\_GPIO\_WritePin(GPIOA,myled1,GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin(GPIOA,myled2,GPIO\_PIN\_RESET);

HAL\_Delay(1000);

HAL\_GPIO\_WritePin(GPIOA,myled1,GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOA,myled2,GPIO\_PIN\_SET);

HAL\_Delay(1000);

/\* USER CODE BEGIN 3 \*/

}

/\* USER CODE END 3 \*/

}

/\*\*

\* @brief System Clock Configuration

\* @retval None

\*/

void SystemClock\_Config(void)

{

RCC\_OscInitTypeDef RCC\_OscInitStruct = {0};

RCC\_ClkInitTypeDef RCC\_ClkInitStruct = {0};

/\*\* Configure the main internal regulator output voltage

\*/

if (HAL\_PWREx\_ControlVoltageScaling(PWR\_REGULATOR\_VOLTAGE\_SCALE1) != HAL\_OK)

{

Error\_Handler();

}

/\*\* Initializes the RCC Oscillators according to the specified parameters

\* in the RCC\_OscInitTypeDef structure.

\*/

RCC\_OscInitStruct.OscillatorType = RCC\_OSCILLATORTYPE\_HSI;

RCC\_OscInitStruct.HSIState = RCC\_HSI\_ON;

RCC\_OscInitStruct.HSICalibrationValue = RCC\_HSICALIBRATION\_DEFAULT;

RCC\_OscInitStruct.PLL.PLLState = RCC\_PLL\_ON;

RCC\_OscInitStruct.PLL.PLLSource = RCC\_PLLSOURCE\_HSI;

RCC\_OscInitStruct.PLL.PLLM = 1;

RCC\_OscInitStruct.PLL.PLLN = 10;

RCC\_OscInitStruct.PLL.PLLP = RCC\_PLLP\_DIV7;

RCC\_OscInitStruct.PLL.PLLQ = RCC\_PLLQ\_DIV2;

RCC\_OscInitStruct.PLL.PLLR = RCC\_PLLR\_DIV2;

if (HAL\_RCC\_OscConfig(&RCC\_OscInitStruct) != HAL\_OK)

{

Error\_Handler();

}

/\*\* Initializes the CPU, AHB and APB buses clocks

\*/

RCC\_ClkInitStruct.ClockType = RCC\_CLOCKTYPE\_HCLK|RCC\_CLOCKTYPE\_SYSCLK

|RCC\_CLOCKTYPE\_PCLK1|RCC\_CLOCKTYPE\_PCLK2;

RCC\_ClkInitStruct.SYSCLKSource = RCC\_SYSCLKSOURCE\_PLLCLK;

RCC\_ClkInitStruct.AHBCLKDivider = RCC\_SYSCLK\_DIV1;

RCC\_ClkInitStruct.APB1CLKDivider = RCC\_HCLK\_DIV1;

RCC\_ClkInitStruct.APB2CLKDivider = RCC\_HCLK\_DIV1;

if (HAL\_RCC\_ClockConfig(&RCC\_ClkInitStruct, FLASH\_LATENCY\_4) != HAL\_OK)

{

Error\_Handler();

}

}

/\*\*

\* @brief USART2 Initialization Function

\* @param None

\* @retval None

\*/

static void MX\_USART2\_UART\_Init(void)

{

/\* USER CODE BEGIN USART2\_Init 0 \*/

/\* USER CODE END USART2\_Init 0 \*/

/\* USER CODE BEGIN USART2\_Init 1 \*/

/\* USER CODE END USART2\_Init 1 \*/

huart2.Instance = USART2;

huart2.Init.BaudRate = 115200;

huart2.Init.WordLength = UART\_WORDLENGTH\_8B;

huart2.Init.StopBits = UART\_STOPBITS\_1;

huart2.Init.Parity = UART\_PARITY\_NONE;

huart2.Init.Mode = UART\_MODE\_TX\_RX;

huart2.Init.HwFlowCtl = UART\_HWCONTROL\_NONE;

huart2.Init.OverSampling = UART\_OVERSAMPLING\_16;

huart2.Init.OneBitSampling = UART\_ONE\_BIT\_SAMPLE\_DISABLE;

huart2.AdvancedInit.AdvFeatureInit = UART\_ADVFEATURE\_NO\_INIT;

if (HAL\_UART\_Init(&huart2) != HAL\_OK)

{

Error\_Handler();

}

/\* USER CODE BEGIN USART2\_Init 2 \*/

/\* USER CODE END USART2\_Init 2 \*/

}

/\*\*

\* @brief GPIO Initialization Function

\* @param None

\* @retval None

\*/

static void MX\_GPIO\_Init(void)

{

GPIO\_InitTypeDef GPIO\_InitStruct = {0};

/\* USER CODE BEGIN MX\_GPIO\_Init\_1 \*/

/\* USER CODE END MX\_GPIO\_Init\_1 \*/

/\* GPIO Ports Clock Enable \*/

\_\_HAL\_RCC\_GPIOC\_CLK\_ENABLE();

\_\_HAL\_RCC\_GPIOH\_CLK\_ENABLE();

\_\_HAL\_RCC\_GPIOA\_CLK\_ENABLE();

\_\_HAL\_RCC\_GPIOB\_CLK\_ENABLE();

/\*Configure GPIO pin Output Level \*/

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_6|GPIO\_PIN\_7, GPIO\_PIN\_RESET);

/\*Configure GPIO pin : B1\_Pin \*/

GPIO\_InitStruct.Pin = B1\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_IT\_FALLING;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

HAL\_GPIO\_Init(B1\_GPIO\_Port, &GPIO\_InitStruct);

/\*Configure GPIO pins : PA6 PA7 \*/

GPIO\_InitStruct.Pin = GPIO\_PIN\_6|GPIO\_PIN\_7;

GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;

HAL\_GPIO\_Init(GPIOA, &GPIO\_InitStruct);

/\* USER CODE BEGIN MX\_GPIO\_Init\_2 \*/

/\* USER CODE END MX\_GPIO\_Init\_2 \*/

}

/\* USER CODE BEGIN 4 \*/

/\* USER CODE END 4 \*/

/\*\*

\* @brief This function is executed in case of error occurrence.

\* @retval None

\*/

void Error\_Handler(void)

{

/\* USER CODE BEGIN Error\_Handler\_Debug \*/

/\* User can add his own implementation to report the HAL error return state \*/

\_\_disable\_irq();

while (1)

{

}

/\* USER CODE END Error\_Handler\_Debug \*/

}

#ifdef USE\_FULL\_ASSERT

/\*\*

\* @brief Reports the name of the source file and the source line number

\* where the assert\_param error has occurred.

\* @param file: pointer to the source file name

\* @param line: assert\_param error line source number

\* @retval None

\*/

void assert\_failed(uint8\_t \*file, uint32\_t line)

{

/\* USER CODE BEGIN 6 \*/

/\* User can add his own implementation to report the file name and line number,

ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) \*/

/\* USER CODE END 6 \*/

}

#endif /\* USE\_FULL\_ASSERT \*/

LCD Screen:

Main.c:

/\* USER CODE BEGIN Header \*/

/\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @file : main.c

\* @brief : Main program body

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @attention

\*

\* Copyright (c) 2025 STMicroelectronics.

\* All rights reserved.

\*

\* This software is licensed under terms that can be found in the LICENSE file

\* in the root directory of this software component.

\* If no LICENSE file comes with this software, it is provided AS-IS.

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

/\* USER CODE END Header \*/

/\* Includes ------------------------------------------------------------------\*/

#include "main.h"

#include "LCD.h"

#include "stdlib.h"

/\* Private includes ----------------------------------------------------------\*/

/\* USER CODE BEGIN Includes \*/

/\* USER CODE END Includes \*/

/\* Private typedef -----------------------------------------------------------\*/

/\* USER CODE BEGIN PTD \*/

/\* USER CODE END PTD \*/

/\* Private define ------------------------------------------------------------\*/

/\* USER CODE BEGIN PD \*/

/\* USER CODE END PD \*/

/\* Private macro -------------------------------------------------------------\*/

/\* USER CODE BEGIN PM \*/

/\* USER CODE END PM \*/

/\* Private variables ---------------------------------------------------------\*/

/\* USER CODE BEGIN PV \*/

/\* USER CODE END PV \*/

/\* Private function prototypes -----------------------------------------------\*/

void SystemClock\_Config(void);

static void MX\_GPIO\_Init(void);

/\* USER CODE BEGIN PFP \*/

/\* USER CODE END PFP \*/

/\* Private user code ---------------------------------------------------------\*/

/\* USER CODE BEGIN 0 \*/

/\* USER CODE END 0 \*/

/\*\*

\* @brief The application entry point.

\* @retval int

\*/

int main(void)

{

/\* Reset of all peripherals, Initializes the Flash interface and the Systick. \*/

HAL\_Init();

/\* Configure the system clock \*/

SystemClock\_Config();

/\* Initialize all configured peripherals \*/

MX\_GPIO\_Init();

/\* Initialize LCD \*/

lcd\_init();

/\* Put ECET 329 on Top Row \*/

lcd\_goto(0,0);

lcd\_puts("ECET 329");

/\* Put LFGD on Bottom Row \*/

lcd\_goto(0, 1); // Column 0, row 1

lcd\_puts("LFGD"); // Display Count

/\* Infinite loop \*/

/\* USER CODE BEGIN WHILE \*/

while (1)

{

/\* USER CODE END WHILE \*/

//HAL\_Delay(1000); // Wait 1 second

/\* USER CODE BEGIN 3 \*/

}

/\* USER CODE END 3 \*/

}

/\*\*

\* @brief System Clock Configuration

\* @retval None

\*/

void SystemClock\_Config(void)

{

RCC\_OscInitTypeDef RCC\_OscInitStruct = {0};

RCC\_ClkInitTypeDef RCC\_ClkInitStruct = {0};

/\*\* Configure the main internal regulator output voltage

\*/

if (HAL\_PWREx\_ControlVoltageScaling(PWR\_REGULATOR\_VOLTAGE\_SCALE1) != HAL\_OK)

{

Error\_Handler();

}

/\*\* Initializes the RCC Oscillators according to the specified parameters

\* in the RCC\_OscInitTypeDef structure.

\*/

RCC\_OscInitStruct.OscillatorType = RCC\_OSCILLATORTYPE\_HSI;

RCC\_OscInitStruct.HSIState = RCC\_HSI\_ON;

RCC\_OscInitStruct.HSICalibrationValue = RCC\_HSICALIBRATION\_DEFAULT;

RCC\_OscInitStruct.PLL.PLLState = RCC\_PLL\_ON;

RCC\_OscInitStruct.PLL.PLLSource = RCC\_PLLSOURCE\_HSI;

RCC\_OscInitStruct.PLL.PLLM = 1;

RCC\_OscInitStruct.PLL.PLLN = 10;

RCC\_OscInitStruct.PLL.PLLP = RCC\_PLLP\_DIV7;

RCC\_OscInitStruct.PLL.PLLQ = RCC\_PLLQ\_DIV2;

RCC\_OscInitStruct.PLL.PLLR = RCC\_PLLR\_DIV2;

if (HAL\_RCC\_OscConfig(&RCC\_OscInitStruct) != HAL\_OK)

{

Error\_Handler();

}

/\*\* Initializes the CPU, AHB and APB buses clocks

\*/

RCC\_ClkInitStruct.ClockType = RCC\_CLOCKTYPE\_HCLK|RCC\_CLOCKTYPE\_SYSCLK

|RCC\_CLOCKTYPE\_PCLK1|RCC\_CLOCKTYPE\_PCLK2;

RCC\_ClkInitStruct.SYSCLKSource = RCC\_SYSCLKSOURCE\_PLLCLK;

RCC\_ClkInitStruct.AHBCLKDivider = RCC\_SYSCLK\_DIV1;

RCC\_ClkInitStruct.APB1CLKDivider = RCC\_HCLK\_DIV1;

RCC\_ClkInitStruct.APB2CLKDivider = RCC\_HCLK\_DIV1;

if (HAL\_RCC\_ClockConfig(&RCC\_ClkInitStruct, FLASH\_LATENCY\_4) != HAL\_OK)

{

Error\_Handler();

}

}

/\*\*

\* @brief GPIO Initialization Function

\* @param None

\* @retval None

\*/

static void MX\_GPIO\_Init(void)

{

GPIO\_InitTypeDef GPIO\_InitStruct = {0};

/\* USER CODE BEGIN MX\_GPIO\_Init\_1 \*/

/\* USER CODE END MX\_GPIO\_Init\_1 \*/

/\* GPIO Ports Clock Enable \*/

\_\_HAL\_RCC\_GPIOA\_CLK\_ENABLE();

/\*Configure GPIO pin Output Level \*/

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_6|GPIO\_PIN\_7|GPIO\_PIN\_8|GPIO\_PIN\_9

|GPIO\_PIN\_10|GPIO\_PIN\_11, GPIO\_PIN\_RESET);

/\*Configure GPIO pins : PA6 PA7 PA8 PA9

PA10 PA11 \*/

GPIO\_InitStruct.Pin = GPIO\_PIN\_6|GPIO\_PIN\_7|GPIO\_PIN\_8|GPIO\_PIN\_9

|GPIO\_PIN\_10|GPIO\_PIN\_11;

GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;

HAL\_GPIO\_Init(GPIOA, &GPIO\_InitStruct);

/\* USER CODE BEGIN MX\_GPIO\_Init\_2 \*/

/\* USER CODE END MX\_GPIO\_Init\_2 \*/

}

/\* USER CODE BEGIN 4 \*/

/\* USER CODE END 4 \*/

/\*\*

\* @brief This function is executed in case of error occurrence.

\* @retval None

\*/

void Error\_Handler(void)

{

/\* USER CODE BEGIN Error\_Handler\_Debug \*/

/\* User can add his own implementation to report the HAL error return state \*/

\_\_disable\_irq();

while (1)

{

}

/\* USER CODE END Error\_Handler\_Debug \*/

}

#ifdef USE\_FULL\_ASSERT

/\*\*

\* @brief Reports the name of the source file and the source line number

\* where the assert\_param error has occurred.

\* @param file: pointer to the source file name

\* @param line: assert\_param error line source number

\* @retval None

\*/

void assert\_failed(uint8\_t \*file, uint32\_t line)

{

/\* USER CODE BEGIN 6 \*/

/\* User can add his own implementation to report the file name and line number,

ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) \*/

/\* USER CODE END 6 \*/

}

#endif /\* USE\_FULL\_ASSERT \*/

LCD.c:

#include <main.h>

#include <LCD.h>

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* START OF LCD CODE \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//

// In the following LCD code, it is assumed that the LCD is connected to the

// Nucleo-L476RG board as follows in 4-bit mode:

//

// RS to PA6

// E to PA7

// D4 to PA8

// D5 to PA9

// D6 to PA10

// D7 to PA11

//

// LCD C Code:

void LCD\_STROBE() {

HAL\_GPIO\_WritePin(GPIOA, LCD\_EN, GPIO\_PIN\_SET);

HAL\_Delay(0.1);

HAL\_GPIO\_WritePin(GPIOA, LCD\_EN, GPIO\_PIN\_RESET);

HAL\_Delay(0.1);

}

// Send a command to the LCD

void lcd\_write\_cmd(unsigned char c) {

unsigned int d = c;

d = (d << 4) & 0x0F00; // Extract upper nibble

GPIOA->ODR = d; // Output to GPIOA

HAL\_GPIO\_WritePin(GPIOA, LCD\_RS, GPIO\_PIN\_RESET); // Clear RS

LCD\_STROBE(); // Clock enable bit

d = c;

d = (d << 8) & 0x0F00; // Extract lower nibble

GPIOA->ODR = d; // Output to GPIOA

HAL\_GPIO\_WritePin(GPIOA, LCD\_RS, GPIO\_PIN\_RESET); // Clear RS

LCD\_STROBE(); // Clock enable bit

HAL\_Delay(0.1);

}

// Send data to the LCD

void lcd\_write\_data(unsigned char c) {

unsigned int d = c;

d = (d << 4) & 0x0F00; // Extract upper nibble

GPIOA->ODR = d; // Output to GPIOA

HAL\_GPIO\_WritePin(GPIOA, LCD\_RS, GPIO\_PIN\_SET); // Set RS HIGH

LCD\_STROBE(); // Clock enable bit

d = c;

d = (d << 8) & 0x0F00; // Extract lower nibble

GPIOA->ODR = d; // Output to GPIOA

HAL\_GPIO\_WritePin(GPIOA, LCD\_RS, GPIO\_PIN\_SET); // Set RS HIGH

LCD\_STROBE(); // Clock enable bit

}

// Clear LCD

void lcd\_clear(void) {

lcd\_write\_cmd(0x1);

HAL\_Delay(5);

}

// Display text message on LCD

void lcd\_puts(const char \*s) {

while (\*s) {

lcd\_write\_data(\*s++);

}

}

// Display single character on LCD

void lcd\_putch(char c) {

unsigned int d = c;

d = (d << 4) & 0x0F00;

GPIOA->ODR = d;

HAL\_GPIO\_WritePin(GPIOA, LCD\_RS, GPIO\_PIN\_SET);

LCD\_STROBE();

d = c;

d = (d << 8) & 0x0F00;

GPIOA->ODR = d;

HAL\_GPIO\_WritePin(GPIOA, LCD\_RS, GPIO\_PIN\_SET);

LCD\_STROBE();

}

// Position the cursor at column, row

void lcd\_goto(int col, int row) {

char address;

if (row == 0) address = 0;

if (row == 1) address = 0x40;

address += col - 1;

lcd\_write\_cmd(0x80 | address);

}

// Initialize the LCD

void lcd\_init(void) {

GPIOA->ODR = 0;

HAL\_Delay(50);

GPIOA->ODR = 0x0300;

LCD\_STROBE();

HAL\_Delay(30);

LCD\_STROBE();

HAL\_Delay(20);

LCD\_STROBE();

HAL\_Delay(20);

GPIOA->ODR = 0x0200;

LCD\_STROBE();

HAL\_Delay(5);

lcd\_write\_cmd(0x28);

HAL\_Delay(5);

lcd\_write\_cmd(0x0F);

HAL\_Delay(5);

lcd\_write\_cmd(0x01);

HAL\_Delay(5);

lcd\_write\_cmd(0x06);

HAL\_Delay(5);

}

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* END OF LCD CODE \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

LCD.h:

/\*

\* LCD.h

\*

\* Created on: Mar 5, 2025

\* Author: John

\*/

#ifndef INC\_LCD\_H\_

#define INC\_LCD\_H\_

/\* Define Pins \*/

#define LCD\_EN GPIO\_PIN\_7

#define LCD\_RS GPIO\_PIN\_6

#define LCD\_D4 GPIO\_PIN\_8

#define LCD\_D5 GPIO\_PIN\_9

#define LCD\_D6 GPIO\_PIN\_10

#define LCD\_D7 GPIO\_PIN\_11

/\* Functions in LCD.c \*/

void LCD\_STROBE(void);

void lcd\_write\_cmd(unsigned char);

void lcd\_write\_data(unsigned char);

void lcd\_clear(void);

void lcd\_puts(const char \* s);

void lcd\_putch(char c);

void lcd\_goto(int,int);

void lcd\_init(void);

#endif /\* INC\_LCD\_H\_ \*/

Picture of LCD Output:



**Conclusion:**

Through this lab I learned how to control the LCD screen using the Nucleo board rather than using an Arduino board. I struggled getting the physical connection of the LCD display working properly due to a couple of reasons. The first being I had wired the LCD Data 4 pin to the wrong GPIO pin and the second being that the diagram given in the textbook wasn’t entirely accurate the LCD that I had. After figuring out these issues, I learned a great deal of the troubleshooting process.

Lab Signoff:

A diagram of a flowchart

AI-generated content may be incorrect.

A circuit board with wires and text

AI-generated content may be incorrect.

**References:**

Ibrahim, D. (2020). *Nucleo boards: Programming with the STM32CubeIDE—Hands-on in more*

*than 50 projects*. Elektor International Media B.V.

Purdue University. (n.d.). *ECET 329 lab instructions*. Purdue University.