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.

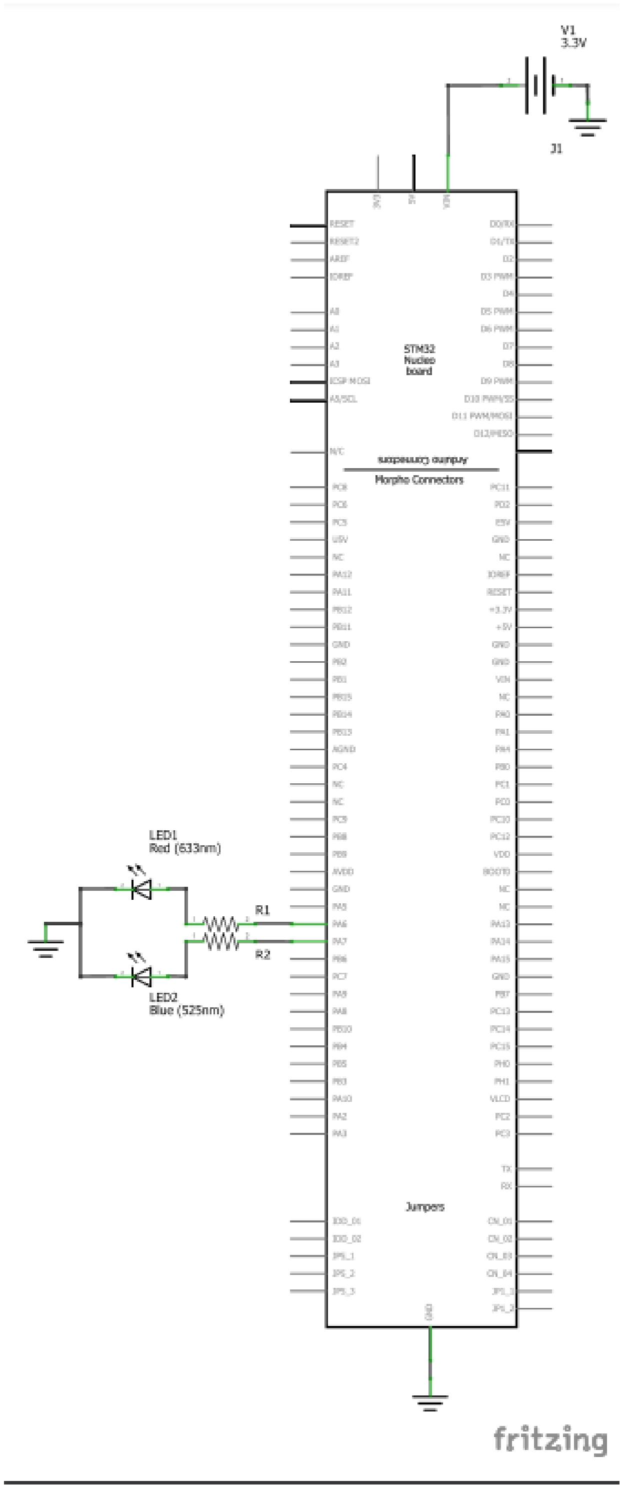
Activity:

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.

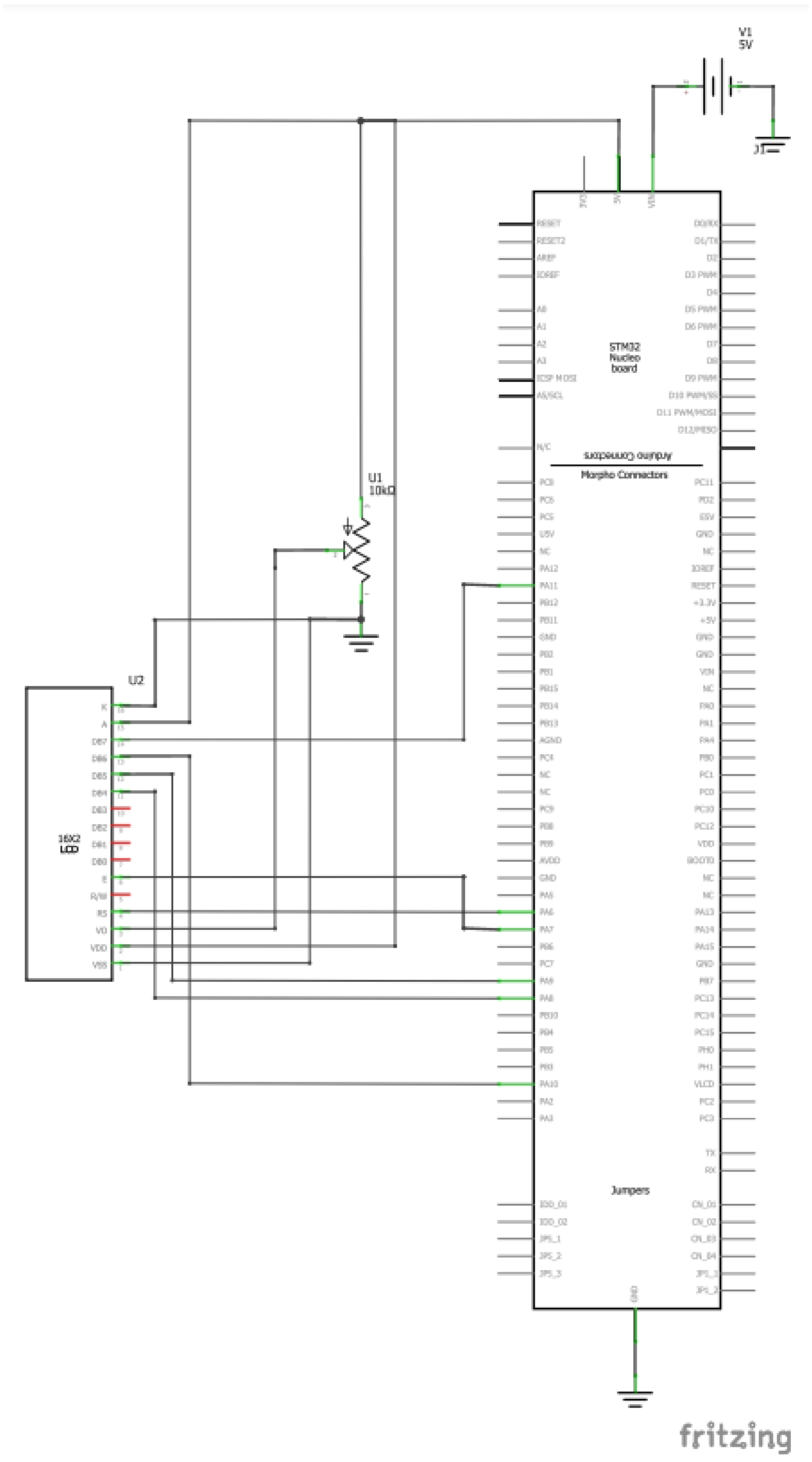
Output:

Professional Wiring Diagrams:

Alternative Flashing:



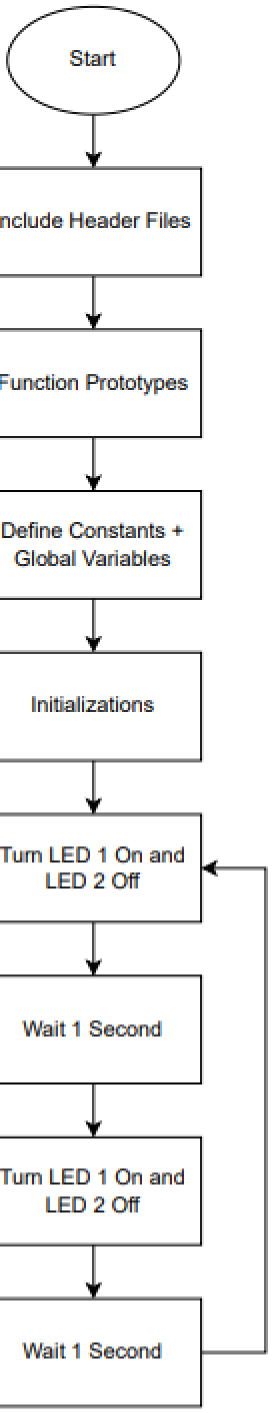
LCD Screen:



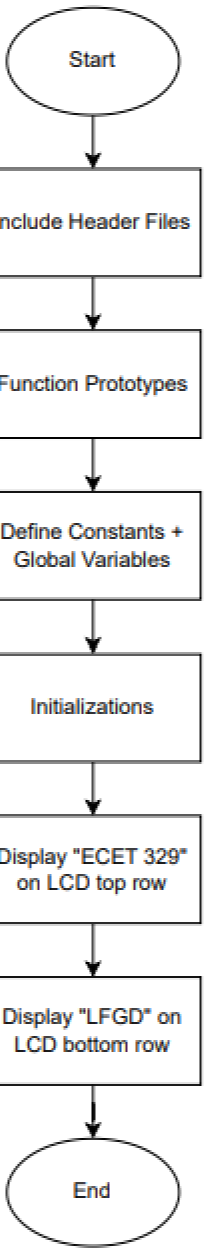
V1
5V
J1
3V3
5V
VIN
STM32
Nucleo
board
Arduino Connectors
Morpho Connectors
U1
10kΩ
U2
16X2
LCD
K
A
DB7
DB6
DB5
DB4
DB3
DB2
DB1
DB0
E
R/W
RS
VO
VDD
VSS
PA11
PA6
PA7
PA9
PA8
PA10
Jumpers
GND
fritzing

Professional Flowcharts:

Alternative Flashing:


Start
Include Header Files
Function Prototypes
Define Constants + Global Variables
Initializations
Turn LED 1 On and LED 2 Off
Wait 1 Second
Turn LED 1 On and LED 2 Off
Wait 1 Second

LCD Screen:



Start
Include Header Files
Function Prototypes
Define Constants + Global Variables
Initializations
Display "ECET 329" on LCD top row
Display "LFGD" on LCD bottom row
End

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
```

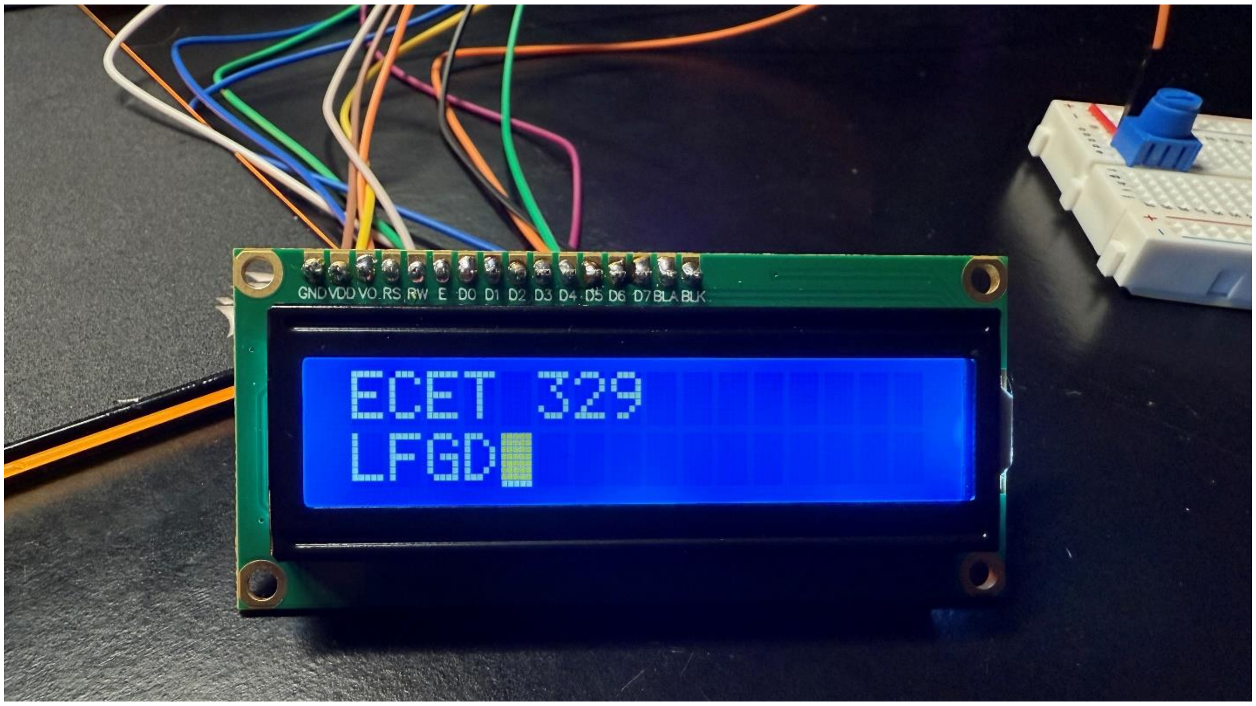
```
void lcd_init(void) {

    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_ */
```

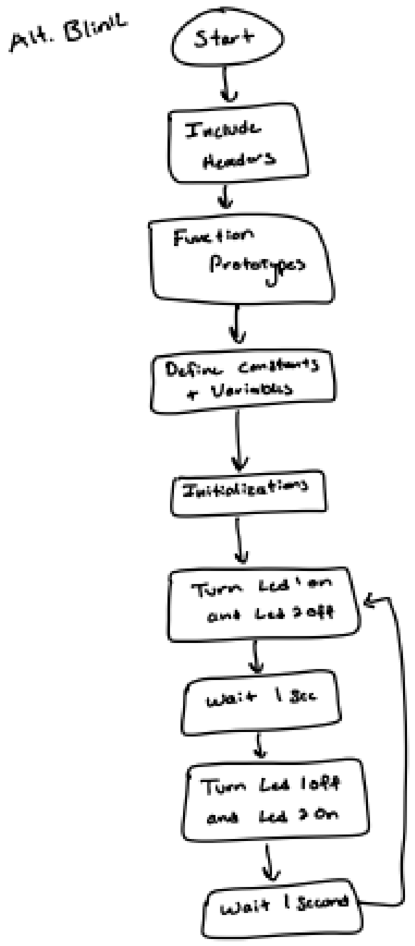
Picture of LCD Output:

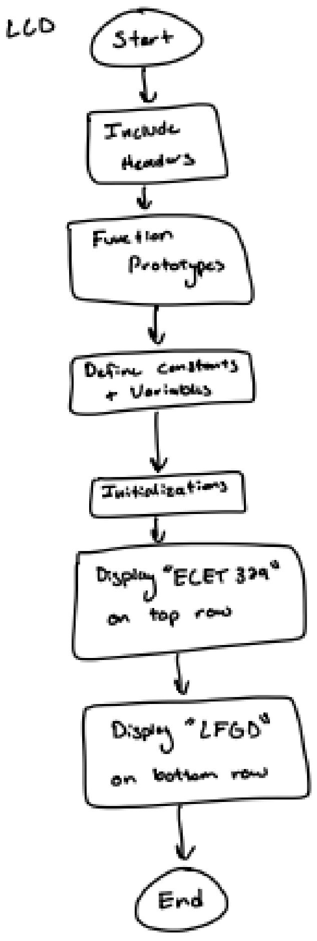


Conclusion:

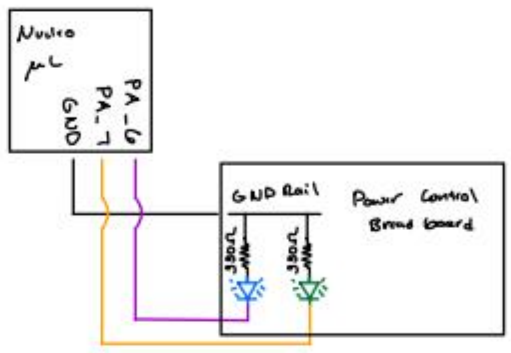
Through this lab I learned how to control the LCD screen using the Nucleo board rather than using an Arduino board. I struggled getting the physical connection of the LCD display working properly due to a couple of reasons. The first being I had wired the LCD Data 4 pin to the wrong GPIO pin and the 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:

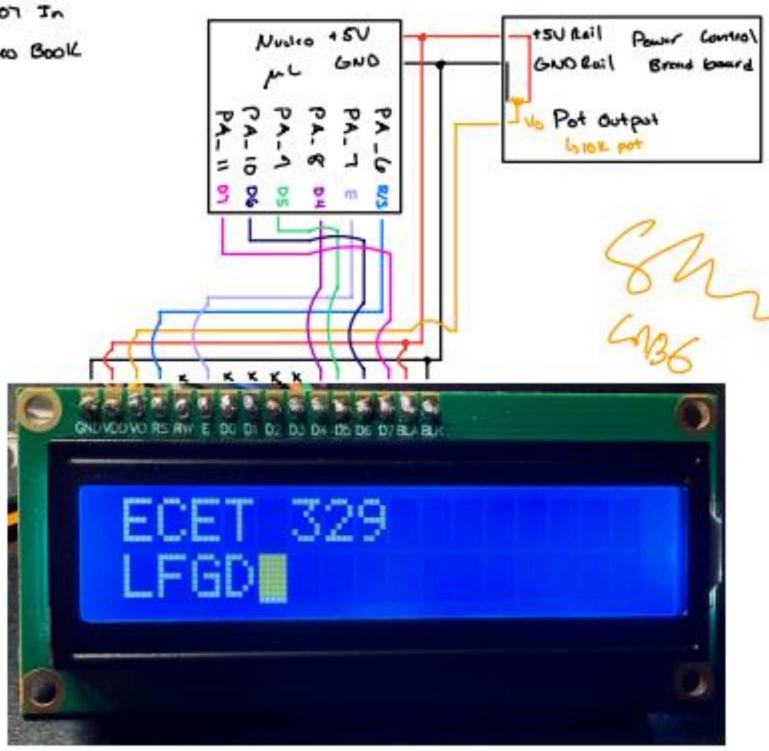



LCD
Start
Include Headers
Function Prototypes
Define Constants + Variables
Initializations
Display "ECET 329" on top row
Display "LFGD" on bottom row
End

Alt. Blink



LCD   Pg 207 In
Nucleo Book



[illegible]

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.