Penn Engineering

LAB 4: PONG

ESE5190 Smart Devices

DUE: THURSDAY, NOVEMBER 16, 2023 23:59 EDT

Contents

1 Overview

We have finally arrived at the last lab and the final project is in view. The main objective of this lab is to write a graphics library for the LCD. The second objective is to write a pong game. The third objective is to gain some familiarity with the ESP32 Feather board which has Wi-Fi capabilities. Hopefully you have gained enough confidence in yourself and your Internet sleuthing skills that you can approach a new and unfamiliar microcontroller with no fear. :)

The lab consists of answering a series of questions and submitting a code repository. All the required questions are highlighted in green. Make a copy of the submission document and record your answers there. Since this lab requires a few video demonstrations, make sure that your links work .

Read through this entire document before beginning the lab.

2 Learning Objectives

After this lab, you should understand the following:

- LCD Library
- General use of the ESP32 Feather Wi-Fi board

3 Related Equipment

3.1 Hardware

- Elgoo Kit
- LCD Screen (Supplied by lab)
- ESP32 Feather (Supplied by lab)

3.2 Software

3.2.1 AVR Toolchain

It is highly recommended to install Atmel Studio. Refer to the Canvas (Windows) Canvas (OSX) page to set up your programming environment if you haven't already done so. There are guides to set up your environment to program your microcontroller on different platforms.

3.2.2 GitHub Repository

Click on this link to create your repository on Github Classroom. To facilitate easier grading, your repository name should include your pennkey. Please rename your repo to be labx-pennkey.

4 Part A: LCD Library

The LCD module we are interfacing with is a 1.8" 128x160 TFT LCD from Adafruit 358. This particular module utilizes the ST7735R controller (datasheet), which simplifies the communication protocol from hundreds of parallel lines to the nice and familiar 4-wire SPI. We will not be using the

SD card reader on the module for this lab (but it may be nice to look into for your final project since it is sharing the SPI lines).

You will be writing the high level graphics library for the LCD. Yes, there are many many tutorials and libraries available for these types of LCDs, but it is highly recommended that you do NOT seek those libraries for references since it is very difficult to "unsee" code and you do not want to risk violating the code of academic integrity.

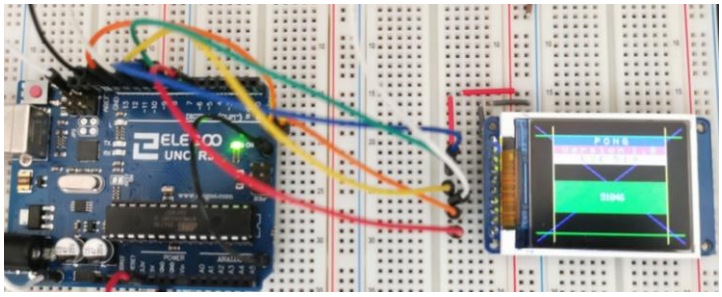
A starter project including a barebone driver for the ST7735R is provided in this lab to help get you started on initializing and communicating with the LCD module.

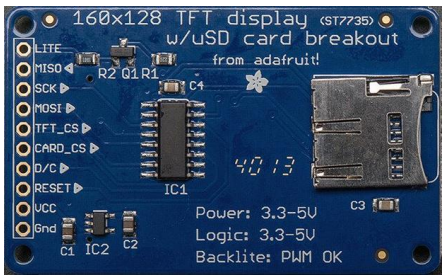
Before starting, it is strongly recommended that you familiarize yourself with the ST7735R datasheet. Understanding how intimidating the 167 page datasheet may seem, below are some important sections to refer to:

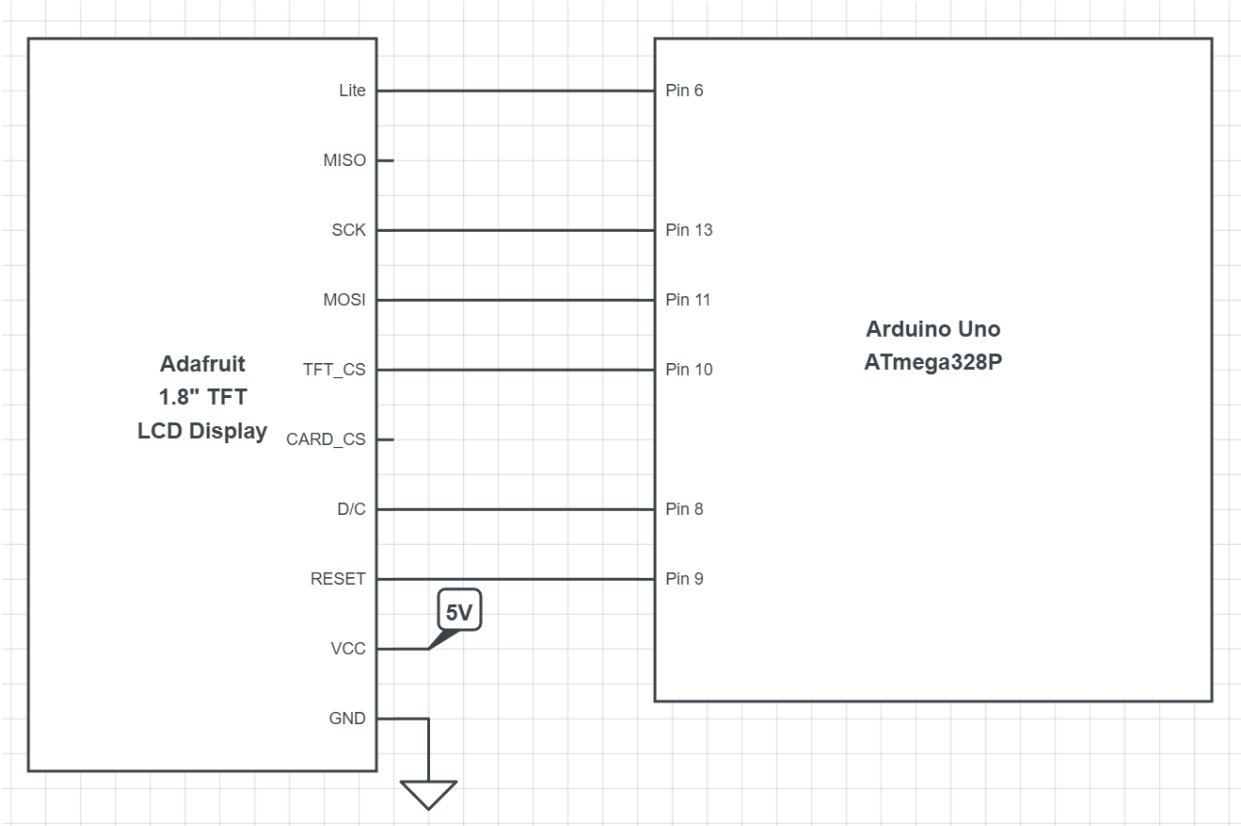
- 9.4. SPI protocol
- 9.8.3. 8-bit data bus for 16-bit/pixel (RGB 5-6-5-bit input)
- 9.10. Address Counter
- 9.11. Memory Data Write Direction
- 10. Commands

4.1 LCD Hardware Setup

Connect the LCD module to Arduino Uno using the following wiring diagram:







4.2 Project and Library Setup

When you click on the github classroom link, a template should already be created in your repository with the skeleton code.

- main.c: All it does at the moment is call the lcd_init() function to initialize the LCD module. Build your code from here.
- lib\ST7735.c: This library contains basic functions to interface with the ST7735R driver. The communication protocol is SPI, and all LCD pins are initialized here.

  Some questions to ask yourself:

  1. How does the controller differentiate between a command or a data packet?
  2. Colors are 16 bits but packet length is 8 bits, how are these sent?
  3. What is the purpose of the LCD_setAddr function? Read about the called command.

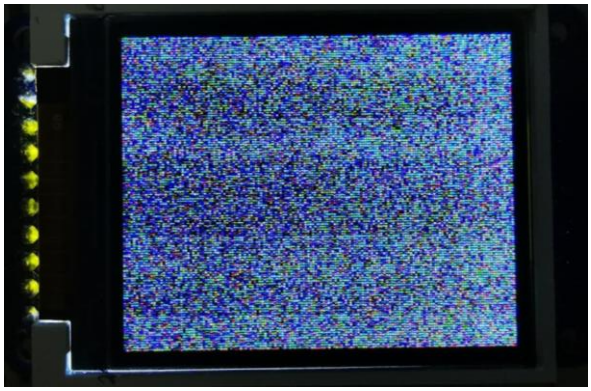
- lib\LCD_GFX.c: This is the graphics library to be completed by students.

  4. rgb565 is a useful function to convert a 24-bit (8-8-8) RGB color value to a 16-bit (5-6-5) value used by the controller. What is the difference between 24 bit RGB and 16 bit RGB? Is information lost during this conversion? If so, is the loss significant?

  - LCD_drawPixel is an example function to start drawing graphics

- LCD_drawChar uses a look-up table to draw individual characters pixel-by-pixel

Compile and flash this code. If the project is set up properly, the LCD should now turn on and show a static screen as shown (ha!).



4.3 Graphics Library

There are many graphics libraries floating around on the internet. It is strongly recommended that you DO NOT LOOK UP any sample graphics library code. It is very difficult to "unsee" code. All of your code must be original. If there are strong similarities between the code that you submit and existing graphics libraries, you will risk violating the code of academic integrity. Go through the sample code given and write your code and functions from scratch.

You can devise the simplest test to draw some pixels using loops to see if they show up on the screen.

In the LCD_GFX library (LCD_GFX.c and LCD_GFX.h), write the following functions:

1. LCD_drawCircle: This draws a filled circle.
2. LCD_drawLine: Bresenham's Line Algorithm may come in handy here.
3. LCD_drawBlock: This draws a filled rectangle
4. LCD_setScreen: Sets the entire screen to be one color
5. LCD_drawString: Draws a string starting at a point

Feel free to create any additional functions that may be useful in your library. You may also include any additional library files that you create.

Ensure drawing actions do not take unreasonably long to complete. If the drawing functions are too inefficient, they can severely impact the refresh rate of the game.

Hint: Refer to 9.10 and 9.11 of the ST7735R datasheet to see how to efficiently string together Tx packets when writing to the controller.

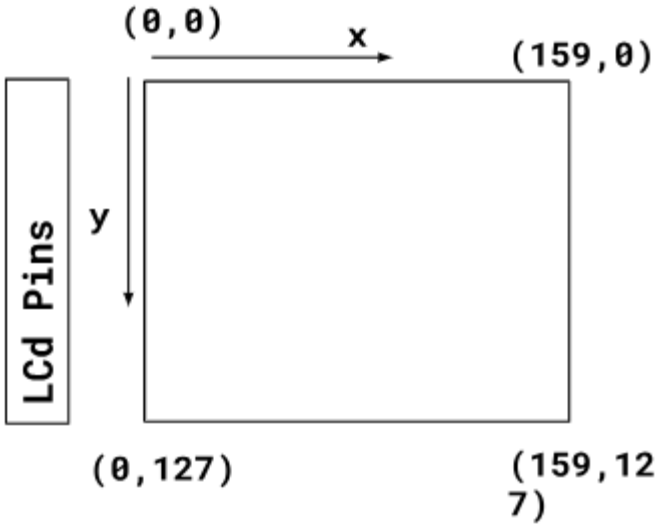


Figure 1 LCD Coordinates

5 Part B: Pong

You will need the following hardware:

- Buzzer - can be active or passive
- 1 joystick
- Two different color LEDs + their current limiting resistor

You have freedom in choosing how you want to connect the components above to the uno. You also have freedom in the color choice of the components.

Write a pong game that satisfies the following requirements:

5.1 Game Modes

1. Player v. Computer

a. Your computer can be as smart as you like or as simple as the paddle steadily going up and down the screen or at random speeds.

5.2 Paddles

1. Two paddles along the Y axis on each edge
2. Choose a paddle width and height that is visible.
3. Paddles move up and down when the joystick is used. You will only need to use one axis of the joystick.
4. Joystick is pushed up to move the paddle up. Joystick is pushed down to move the paddle down. You can decide whether or not the paddle will continue to move up if the joystick is held in the up position or if the user has to pulse in order to move the paddle upwards.

5. Keep in mind the bounded region of the screen. The paddles should not be able to move beyond the boundaries of the screen even if the user tries to continue to press the button.

5.3 Ball

6. Choose a radius that yields a visible ball.
7. The ball should start at the center of the screen at the start of the game and then start to move in a random direction and at random speed. This can change with each round if you'd like or change during the round.
8. The ball should follow proper game dynamics when it interacts with other components in the game such as the paddles and the boundaries of the game. In other words, the ball should bounce off the top and bottom of the walls and the paddles.

5.4 Scoring

1. Include a scoreboard on the screen
2. Player 1 wins a point if Player 2 fails to return the ball.
3. Game resets when a player reaches a certain number of points (you decide this) and the score resets after a certain number of rounds (you decide this).
4. For example, best of 3 rounds, first player to reach 10 points wins the round
5. Flash one LED every time a player wins a point.

5.5 Buzzer

Generate a sound on the buzzer when:

1. The ball hits the horizontal boundary surfaces
2. The ball hits the paddle
3. he ball misses the paddle

Note on the buzzer - You may choose any frequency that you'd like. Frequency of 2.5kHz and a duty cycle of 0.5% - 5% may be relatively bearable.

These are not exhaustive limits. Feel free to add more features of the game. Additional features will be added as extra credit.

Important Things to Note:

- Timer0 is used to PWM the LCD brightness so be careful if you choose to use this peripheral for an additional feature
- If you are using UART and printing to the serial monitor, Pins 0 and 1 on the Uno will not be accessible.
- Redrawing and refreshing the entire screen may take quite a bit of time so you may want to think of other ways to update the visual position of the moving components.

Submit a link to the video demonstrating this section of the assignment.

6 Part C: ESP32 Feather USB Driver Installation

Follow the ESP32 Setup instructions on Canvas.

7 Part D: Blynk

Blynk is a cool platform that has an easy-to-use graphical interface with simple drag and drop widgets. Maybe you want to turn on the light in another room. We are going to control player 1's paddle wirelessly.

Follow the **Blynk Installation with Feather** instructions on Canvas.

Note the correspondence between **Pin Name on the Board** and **Pin Number in Arduino IDE**. The **Pin Number in Arduino IDE** is the pin number that you will use in the Arduino IDE.

8 Part E: Wireless Control of Pong

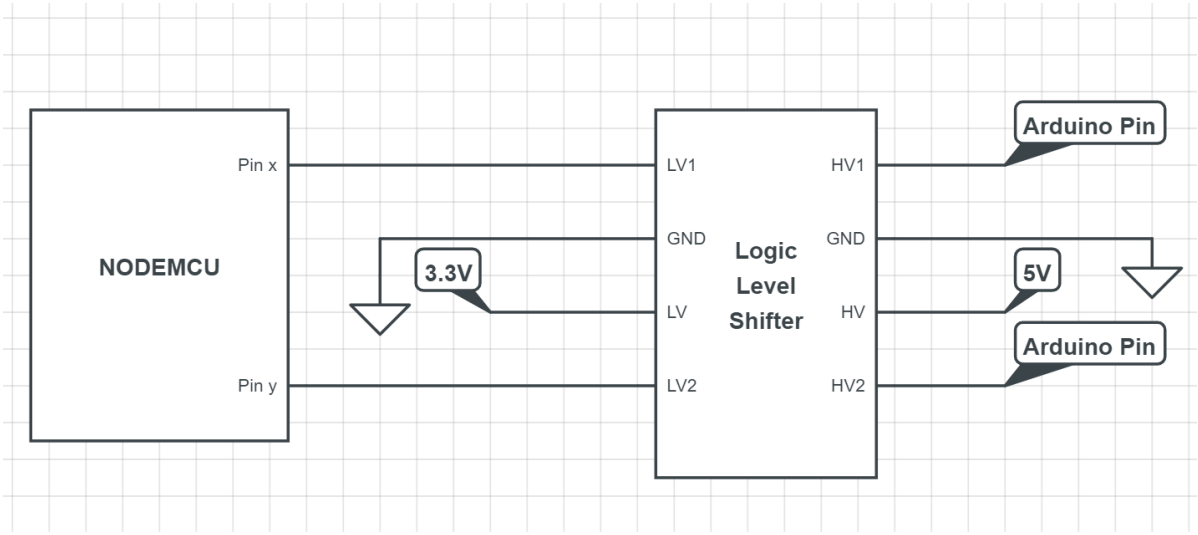
Incorporate wireless control into your game so that the paddles of player 1 can be controlled wirelessly via the Blynk app. Because the ESP32 Feather uses 3.3V logic while the Uno uses 5V logic, a logic level shifter will have to be used in between. Connect the pins that you are controlling from the ESP32 Feather to the low voltage side of the level shifter and connect the Arduino pins to the high voltage side. You can choose to keep the joystick on the breadboard and use other pins to read in the ESP32 Feather output or remove the joystick.

The implementation for this section is also up to you. You can choose a button on the app or a slider or some other combination. As long as you are able to control the paddle, then it is acceptable.

Note 1: If the ESP32 Feather sometimes has trouble connecting when it has worked well before, try powering the ESP32 Feather first (i.e. plugging in the ESP32 Feather) and then power the Uno.

Note 2: If you're having trouble flashing code onto the ESP32 Feather, disconnect the Uno. If you're having trouble flashing code on the Uno, disconnect the ESP32 Feather. In other words, only have one powered at a time when flashing code so that the lines don't interfere with each other.

5. Is debouncing of the "buttons" used to control the paddle needed here?



Submit a link to the video demonstrating this section of the assignment.

9 Part F: Extra Credit

Any additional features that you'd like to add to your game. Give a brief explanation of your addition. Some examples are provided below but the amount of extra credit granted will depend on the complexity of the feature implemented. Having an LED indicator to show the mode will not count for extra credit in this assignment.

Some examples are:

- Game is controllable by two players (2 pts)
- Switch to pause and resume the game (2 pts)
- Interesting ways of controlling or playing the game via Blynk
- Switch to select between wireless or wired control
- Home screen with game selection mode (1 pt)
- Different levels of difficulty (2 pts)
- Accelerometer integration (1 pt)
- Display bitmap image files from a microSD card (Intro screen, perhaps? Winner announcement screen?) (10 pts)
- Snake game (and maybe user can choose between either game?) (10 pts)
- Some other game of your choice

Note that there is a 20pt extra credit maximum.

Submit a link to the video demonstrating this section of the assignment

10 Submission Requirements

1. Click on this link to create your repository on Github Classroom, if you haven't already. Name your repo to be labx-pennkey. (e.g. if my pennkey is 'luke', then my repo name will be lab4-luke.)

2. The following files are required to be in your repository:

a. Skeleton code files

b. Any additional libraries that you created

c. README including any additional information that would help us grade your lab

11 Grading Rubric

Q1 - Q5	10 pts (2 pts x 5 questions)
Part B Demo + GFX Library	200
Part E Demo	50
Extra Credit	20 points maximum
	260 pts