Jimmy Fung
May 2, 2018
Real Time Embedded System
Final Project Write Up

Customizable Scrolling Text On Matrix LED

Introduction

Scrolling message on a LED display is a low cost and effective way to communicate information. Practical use cases include showing the next arrival time of train or bus at the wait stop, or used for advertising by businesses in public spaces like street food cart. In contrast to conventional printed poster, the scroll display allows almost no overhead cost to change the message (though at the expensive of electricity cost).

General Description

At the high level, the product of this project is an LED display and a joystick that allows a user to sketch custom images or texts, and then play it in a scrolling fashion. The system has a “sketch mode” to let the user sketch up to 10 images that will be stored in a memory buffer. When the system is in “scrolling mode”, the LED plays the sketches stored in the buffer.

In terms of components, the project consists of

- An Arduino UNO board
- A 8x8 Red LED matrix display
- A driver chipset called MAX7219 to drive the 8x8 LED matrix display.
- A joystick with clickable button
    - the joysticks move the cursor along x and y-axis.
    - the click button saves the selected pixel for the current sketch
- A push button to save a user’s sketch
- A push button to toggle the system between “sketch mode” and “scrolling mode”
- A potentiometer to adjust the scroll speed during “scrolling mode”

Learning

The learning comes in two categories

1. Hardware
2. Programming

In terms of hardware, I had near zero experience with how most of these breadboard-electronic components work. Soon a pattern emerged in that they generally require a power pin and a ground pin, with output pins that are either digital or analog. This is the case for the joystick, the push buttons, and the potentiometer.

Then comes the first challenge of working with 8x8 matrix display, which upon inspection has an intimidating number of connection of 16 pins. It seems the Arduino board couldn’t handle. After studying the matrix display hardware I learned that, in order to light up 1 LED, one pin is used to

light up a row of LEDs, and seven pins are used to turn off 7 LEDs, so that the an LED is off because both the anode and the cathode of the LED transistor are held at 5v.

Therefore controlling which LED to light up seems very complex. Some kinds of abstraction would be needed to simplify the usage. This helps me understand the purpose of the MAX7219 chipset that was used to drive the LED. Although the MAX7219 only takes 5 connections from the Arduino board, there are 16 connections to the matrix display. The amount of wiring is very prone to human error. Eventually I bought the integrated board containing the 8x8 matrix LED and the MAX7219. So another lesson is that commonly used hardware combination are likely made by some vendor as a single integrated hardware that you can purchase.

At first I wanted to use the 7-segment display (7seg) to show the number of user sketches saved in the memory buffer. The difficulty was it did not come with data sheet, so I attempted to figure out its behavior by applying 5v and ground on random pair of pins on the 7seg. (I did the same for the matrix LED). This was disastrous because it burned out some of the LED segments. I eventually learn from similar models of 7seg that they generally take 3.3V. I end up abandoning the 7seg. I bought a replacement 7seg and got it to work. However, in the end I decided to replace it with a potentiometer that the user can use to change the speed of the sliding sketch, which I thought is a more useful feature than the 7seg, and the Arduino board has insufficient pins to support the 7seg.

The overall lesson is that I should not haphazardly test components without knowing their voltage limit, that I should use a resistor in series with whatever I want to test (especially LEDs), and that I should never buy from a vendor without datasheet. A good number of my purchase came from eBay and Amazon where the sellers have little to no knowledge/supporting materials for the product.

There are programming learning as well. The first one is the joystick. The joystick is intended to let the user move the cursor on the sketchboard in discrete steps, i.e. one push up on the joystick would move the cursor up by one row. However, the joystick outputs the x and y positions values in analog signal that ranges from 0 to 1023. For example, when the joystick is at its natural, center position, x=512. When it's at the left end, x=0. When it's at right end, x=1023. So I had to come up with a programming logic that converts the analog signals to a binary signal that indicates whether the joystick has been pushed in a direction. The idea is to watch for when the joystick moved away from its center position, and crosses a predetermined deviation threshold, say, 30%. Then the joystick is considered "pushed" when it crosses the 30% the second time (as it makes its way back to the natural position). This method of registering push signals worked flawlessly as far as user experience goes.

The most difficult challenge is converting the user sketch into a format that the LED matrix library accepts. A user sketch is represented by a two-dimensional, 8 by 8, array of chars (e.g. ***char sketch[8][8]***). The rows are the y values (vertical movement on the display), and the

columns are the x values (horizontal movement), respectively. This is the natural and intuitive way to represent the sketchboard. However, the matrix LED driver library accepts:

1. A one-dimensional array of chars, where each char element is a “bit vector” (of length 8bit), and
2. Each char element represent a column of LEDs (as opposed to a row - which is what one would expect). This means the sketch is rotated by 90 degrees clockwise.

The reason for either seems sensible after some thought. For 1, it is because an array of “bit vector” encodes the maximum amount of information with least amount of space possible. E.g. a char bit-vector represents one whole row (as opposed to using an array of 8 chars). This is important because data bandwidth is limited - there is only one data line transmitting the matrix representation of the sketch going from Arduino to the matrix driver (MAX7219). Also the LEDs are turned on/off one at a time, so in order to create the illusion of showing the full sketch in an instance, they have to be drawn really fast.

For 2, it is because of memory access efficiency during “play mode”. Scrolling a message is about left-shifting the columns of a sketch. So in the naive “char sketch[8][8]” representation, not only do you have to move every element to the left for each row (which O(n^2) operations), it is also inefficient to access memory across rows (in order to traverse along the column). So by representing a sketch in a 90 degree clockwise rotation, the columns becomes rows. And shifting a column simply becomes reading the next row in the rotated data structure. Very efficient, no copying and moving elements.

Therefore I had to write a conversion function that not only rotate the user sketch by 90 degrees, but also turn each row (an array of 8 chars) into a bit vector (a single char with 8 binary bits). The latter can be accomplished by a series of logical OR operations.

RTES and Future Work

In terms of RTES aspects, I ran into an issue where toggling between sketch mode and play mode was not immediate. For example, while a potentially lengthy sketch is being scrolled on the display, toggling to sketch mode would require waiting the scroll to complete. There is no workaround this in a basic for-loop construct (inside main()). But using interrupt or a scheduler, this issue can be resolved. I went employed the interrupt solution since I didn’t use a scheduler. But for improvement, it seems using a scheduler is better because it gives you more flexibility, especially when the number of things to do increase. However, this comes with the trade-off of increasing code complexity and careful reasoning of meeting deadline. For this project, using a for-loop to visit each tasks is more than sufficient.

For future work, we can improve the usability of system by getting rid of the joystick as well as push button - and replace them with a mobile app to let user to sketch, adjust speed, and toggle mode. This will require a wireless communication addon, perhaps via bluetooth. The matrix display can be replaced with 3-color LEDs instead of just red, and they can chained to create a larger display. A third playing mode can be implemented to allow sketches to be played by

frames rather than by scrolling. This will be a very practical product for advertising and/or public announcements.