← Week 6 Overview

Assignment 6: Pixel Display

Logistics

Assigned: Wed, Oct 12

Due: Wed, Oct 26

1 The idea

In this assignment, we will build a method of displaying characters and images on the 5 by 7 array of LEDs. This extends the investigation we began in studio by using the entire display at any given time via **multiplexing**.

2 The background

Hardware setup

You should be familiar with how to use the LED display from studio. If not, **Studio 6: LED-based 5x7 pixel display** should have you covered.

Remember that the pins on the display correspond to either a row or column on the display, and that LEDs will only light if the *column* is at a lower voltage than the *row*. When trying to turn on LEDs, do it one column at a time, turning the column `LOW` and all the rows you want *on* to `HIGH`. Keep all the rows you want *off* `LOW`. This way there is a path from high to low for your *on*

> LEDs, and not your *off* ones.
>
> You can find the pinout of the display on the first page of the manufacturer's data sheet.

2.1 Output multiplexing

It is not physically possible to fully control more than one row of this LED display at once, due to the way it's wired (what if I wanted to make a checkerboard pattern?).

In order to control the *entire* board at any given time, we must **multiplex** our connection. Multiplexing is communication with mutiple outputs (the columns, in this case) using shared wires (the rows in this case). There are many different types of multiplexing, but today we will be using **time-division multiplexing**.

Taking advantage of the very slow speed of the human eye (it thinks that static images played back at 30fps or so look like motion), we can *cycle through* the columns, one by one, very quickly, so that the change is invisible to our perception.

For example, if I wanted to draw two columns "at once," I could do this:

- Set column 1's output `LOW` and all the other column outputs `HIGH`.
- Output `HIGH` on the rows that are to be lit, and `LOW` on the rows that are to stay dark.
- Wait for $\frac{1}{2}$ of the length of a single **frame**, or the time period that this whole process takes. You will choose this in a bit.
- Set column 2's output `LOW` and all the other column outputs `HIGH`.
- Output `HIGH` on the rows that are to be lit, and `LOW` on the rows that are to stay dark.
- Wait for $\frac{1}{2}$ of the length of a single frame.
- Repeat.

An overall **frame length** of 5 seconds or so is good for debugging purposes, but you will want the length to be less than about 30ms for normal operation (33fps).

3 The assignment

3.1 Fonts?

We've provided a file, `font.h` (in `assignments/assign6/display/font.h`), that defines an array called `font_5x7` that defines which LEDs should be on or off to display any of the 96 printable ASCII characters on a 5 by 7 pixel display.

It's a two-dimensional array: the first dimension corresponds to the character code (using its ASCII code minus `0x20`, so `A` is index 33, not 65), while the second dimension corresponds to the column (of the 5 by 7 display). The individual bits of each index of the array represent the state of the LEDs in each row, with an extra useless bit tacked on at the end (bytes are 8 bits, but our LED only has 7 rows).

1. Make sure you understand how the font is represented by printing some different characters in binary form:

```
c = 0x03;  // '#'
Serial.println(font_5x7[c][0],BIN);
Serial.println(font_5x7[c][1],BIN);
Serial.println(font_5x7[c][2],BIN);
Serial.println(font_5x7[c][3],BIN);
Serial.println(font_5x7[c][4],BIN);
```

Column 0 to 4:
0101000 Row 0 to 6 + one useless bit
11111110
0101000
11111110
0101000

Hint: look at it sideways.

For this piece of the assignment, do not set the `pinMode()` of the display pins to `OUTPUT`, as that will interfere with serial communications. The point of this part is just to help you understand how to interpret the contents of `font_5x7[][]`.

2. Try the above with different values of `c`, especially values for which the character is not symmetric left and right.

3.2 Author the Arduino sketch

1. Open the `display` Arduino sketch.

2. **Difficulties**

If you have difficulty uploading new software with the 5 by 7 display attached, add a delay of about 5 seconds or so (using `delay()`) prior to setting the pins to `OUTPUT` with `pinMode()`. When you wish to upload new software to the Arduino, press and release the reset button (`RESET`) and then start your upload. This should ensure that your upload starts prior to the pin modes being set to `OUTPUT`, so pins shared with the serial port can be effective in uploading the software.

Author an Arduino sketch that displays individual characters on the 5 by 7 pixel display, using the provided font file `font.h`.

Initially, the display should be blank (the space character). When the user presses buttons, the displayed character should change. One button should increment the character (increment the current ASCII character displayed by 1) and another button should decrement the character (decrement it by 1). Wrap around to the beginning (or end) as appropriate. You may need to use the analog pins to read the buttons, since most of the digital pins are used for the 5x7 display. Test out a button by itself before hooking it up to the rest of your circuit, so that you know what values to expect from analogRead().

3. As you will notice as soon as you look for it, the actual font information for the ASCII characters `'0'` through `'9'` is not present in the file that we provided (the entries are all there, but they are all blank).

   Design an appropriate image for each of these characters and include it in the font file.

3.2.1 Guidelines

A few things to watch out for and/or consider:

- It is possible to extract the individual bit information from the font file in a single line of code, `if` statements are not needed. Think back to your bitwise arithmetic.

  (You will not be penalized if you revert to a solution that does use `if` statements. I'm just nudging you to try it without to begin with.)
- Make sure you update your font file to draw numbers.
- You might find it useful to use the LCD for debugging, so setting it up first (*before* you use the LED) might be very helpful.

4 The check-in

1. Commit your code and verify in your web browser that it is all there.

2. Follow the checklist below to see if you have everything done before demo your assignment to a TA.
   - `font.h` is included
   - `Serial.begin()` is removed
   - LED displays character correctly and clearly
   - All buttons are defined correctly and functioned properly
     - - increment the current character by 1
     - decrement the current character by 1
   - Your sketch is able to go back to the beginning when it reaches the last character
   - Number 0 through 9 are disigned appropriately
   - All of your files are committed
3. Check out with a TA. Make sure to show off the characters that you designed.

Things that should be present in your repo structure:

```
└── assignments
    └── assign6/
        └── display/
            ├── display.ino
            └── font_5x7.h
```

4.1 The rubric

- 100pts: Did the demoed assignment work?
  - Displays characters correctly (40 pts)
  - Moves between characters correctly (25 pts)
  - Reasonable character design for 0 to 9 (30 pts)
  - Easy to grade? (5 pts)

Generated at 2016-10-23 21:09:04 -0500.