Which LED(s) turn on?
Lab 3b
Programming the LED cube

ENGR 40M
Chuan-Zheng Lee
Stanford University
12 May 2017
Overview

• **Goal:** To write `display()`, which takes a 4×4×4 pattern `[][][]` array and displays it on the LED cube.

• **Challenges:**
  • Electrically, the LED “cube” actually an 8×8 grid.
  • We’d rather not think about time-division multiplexing.

• How can we write a function that allows to *abstract* these *hardware* details away from our “main” software?
Arrays in C

An array is a sequential collection of elements of the same data type

e.g. int myArr[7];

declares space for seven ints:


These are stored next to each other in memory.
Arrays in C

• You can **access** array elements with []:
  ```c
  int fourthElement = myArray[3];
  myArray[4] = -29;
  ```

• You can **iterate** over array elements:
  ```c
  int sum = 0;
  for (int i = 0; i < 7; i++) {
      sum = sum + myArray[i];
  }
  ```

• Remember, indices **start at zero**
Quiz

If we have the array declaration

```
unsigned byte x[10];
```

what are valid ways to iterate over the array?

a)  `for (int i=0; i < 10; i++)
    Serial.println(x[i]);`

b)  `for (int i=0; i <= 10; i++)
    Serial.println(x[i]);`

c)  `for (int i=10; i > 10; i--)
    Serial.println(x[i]);`

d)  `for (int i=9; i >= 0; i--)
    Serial.println(x[i]);`
Warning

The compiler will **not** tell you what went wrong!

```cpp
unsigned byte x[10];
for (int i = 1; i <= 10; i++)
  Serial.println(x[i]);
```

It will just happily read whatever is in memory *after* the array, whether it makes sense to or not. If it doesn’t make sense, your program may crash **without warning or error message**.
Multidimensional arrays in C

Arrays can be multidimensional

e.g. `int cube[4][4][4];`
Multidimensional arrays in C

• You **access** elements with multiple []:
  
cube[2][0][1] = 1;

• You **iterate** using nested loops:

  ```c
  int sum = 0;
  for (int i = 0; i < 4; i++) {
    for (int j = 0; j < 4; j++) {
      for (int k = 0; k < 4; k++) {
        sum = sum + cube[i][j][k];
      }
    }
  }
  ```
Decomposition

The main Arduino loop

Does one pass through the LEDs (time-division multiplexing)

Looks up the LED state associated with an anode/cathode pair
void loop() {
    byte pattern[4][4][4];
    // (assemble the LEDs pattern)
    display(pattern);
}

void display(byte pattern[4][4][4]) {
    for (byte aNum = 0; aNum < 8; aNum++) {
        for (byte cNum = 0; cNum < 8; cNum++) {
            // LED multiplexing code
        }
    }
}
display() delegates to getLEDState()

```java
void display(byte pattern[4][4][4]) {
    for (byte aNum = 0; aNum < 8; aNum++) {
        for (byte cNum = 0; cNum < 8; cNum++) {
            byte value = getLEDState(pattern, aNum, cNum);
            // LED multiplexing code
        }
    }
}

byte getLEDState(byte pattern[4][4][4], byte aNum, byte cNum) {
    // map from (a,c) to (x,y,z)
    return pattern[z][y][x];
}
```
Review: Pin multiplexing

- anode (+) wires
- cathode (−) wires

Diagram showing the multiplexing of anode and cathode wires.
Review: Providing enough current

Use a PMOS to provide enough power to the LEDs
Implementing `display()`

```c
void display(byte pattern[4][4][4]) {
    for (byte aNum = 0; aNum < 8; aNum++) {
        for (byte cNum = 0; cNum < 8; cNum++) {
            byte value = getLEDState(pattern, aNum, cNum);
            // Activate the cathode (-) wire (column) if value > 0, 
            // deactivate it otherwise
        }
        // Activate the anode (+) wire (row)
        // Wait a bit
        // Deactivate the anode (+) wire (row)
    }
}
```
display() delegates to getLEDState()

```c
void display(byte pattern[4][4][4]) {
    for (byte aNum = 0; aNum < 8; aNum++) {
        for (byte cNum = 0; cNum < 8; cNum++) {
            byte value = getLEDState(pattern, aNum, cNum);
            // LED multiplexing code
        }
    }
}

byte getLEDState(byte pattern[4][4][4], byte aNum, byte cNum) {
    // map from (a,c) to (x,y,z)
    return pattern[z][y][x];
}
```
Mapping between 2-D and 3-D
Mapping between 2-D and 3-D

We want a function that maps

*from* anode/cathode pairs $(a, c)$

(e.g. “D6”)

to 3D coordinates $(x, y, z)$
Mapping between 2-D and 3-D

Anodes

Cathodes
Making the mapping function easier

You can reorder the anodes/cathodes however you like. Would a different ordering make the relationship simpler?
Mapping between 2-D and 3-D
Mapping between 2-D and 3-D
What do bitwise operations do?

• Bitwise operations apply to each bit in the binary representation of a number individually

Examples (in binary):

\[
\begin{align*}
00110101 & \quad | \quad 01100011 \quad == \quad 01110111 \\
00110101 & \quad \& \quad 01100011 \quad == \quad 00100001 \\
00110101 & \quad | \quad 01100011 \\
& \quad \& \quad 01100011
\end{align*}
\]
Implementing `getLEDState()`

```c
void getLEDState(byte pattern[4][4][4], byte aNum, byte cNum) {
    x = ???;
    y = ???;
    z = ???;
    return pattern[z][y][x];
}
```

(Your function can be more than four lines, depending on how you implement it.)
The main Arduino loop

Does one pass through the LEDs (time-division multiplexing)

Looks up the LED state associated with an anode/cathode pair
A note on coding style

• Indent your code correctly  
  (Seriously, pay attention to this!)

• Don’t just keep adding braces until your code compiles
A note on coding style

Why indentation matters:

```python
for(int i = 0; i < 5; i++){
    for(int j = 0; j < 2; j++){
        if(i == 0 && j == 0)
            print("Hello!\n");
    }
    print("Hello!\n");
}
```
A few pointers

• Give variables sensible names
  ```
  byte whichOne;
  byte currentPatternIndex;
  ```

• Use constants for things that shouldn’t change
  ```
  const byte TEST_PIN = 3;
  ```

• Watch out for = and ==
  ```
  if (x = 3) {
    // always true
    print("winning!");
  }
  ```
A note on coding style

KEEP IN MIND THAT I'M SELF-TAUH, SO MY CODE MAY BE A LITTLE MESSY.

...WOW.
THIS IS LIKE BEING IN A HOUSE BUIIl BY A CHILD USING NOTHING BUT A HATCHET AND A PICTURE OF A HOUSE.

IT'S LIKE A SALAD RECIPE WRITTEN BY A CORPORATE LAWYER USING A PHONE AUTO CORRECT THAT ONLY KNEW EXCEL FORMULAS.

IT'S LIKE SOMEONE TOOK A TRANSCRIPT OF A COUPLE ARGUING AT IKEA AND MADE RANDOM EDITS UNTIL IT COMPILED WITHOUT ERRORS.

OKAY, I'LL READ A STYLE GUIDE.

https://xkcd.com/1513/
Planning your breadboard
What makes a good breadboard layout?

- Easy to follow
- Wires are relatively direct (and color-coded)
  
  *No spaghetti-like crossovers*

- Resistors don’t touch each other
- Your cube wiring should also be neat
- Try to keep fairly compact, to allow for things to add next week!