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

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

• The majority of errors that TAs have observed are because people have not been reading the lab handouts.

• Please read them.
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
If we have the array declaration

```c
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!

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];`

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>myArr[3][0][0]</td>
<td>myArr[3][0][1]</td>
<td>myArr[3][0][2]</td>
<td>myArr[3][0][3]</td>
</tr>
<tr>
<td>myArr[2][0][0]</td>
<td>myArr[2][0][1]</td>
<td>myArr[2][0][2]</td>
<td>myArr[2][0][3]</td>
</tr>
<tr>
<td>myArr[1][0][0]</td>
<td>myArr[1][0][1]</td>
<td>myArr[1][0][2]</td>
<td>myArr[1][0][3]</td>
</tr>
<tr>
<td>myArr[0][0][0]</td>
<td>myArr[0][0][1]</td>
<td>myArr[0][0][2]</td>
<td>myArr[0][0][3]</td>
</tr>
<tr>
<td>myArr[0][1][0]</td>
<td>myArr[0][1][1]</td>
<td>myArr[0][1][2]</td>
<td>myArr[0][1][3]</td>
</tr>
<tr>
<td>myArr[0][2][0]</td>
<td>myArr[0][2][1]</td>
<td>myArr[0][2][2]</td>
<td>myArr[0][2][3]</td>
</tr>
<tr>
<td>myArr[0][3][0]</td>
<td>myArr[0][3][1]</td>
<td>myArr[0][3][2]</td>
<td>myArr[0][3][3]</td>
</tr>
</tbody>
</table>
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];
      }
    }
  }
  ```
The main Arduino loop

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

Looks up the LED state associated with an anode/cathode pair
loop() delegates to display()

```c
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()

```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];
}
```
Review: Pin multiplexing

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

```
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()

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

May 12, 2017

ENGR 40M Spring 2017 — C.Z. Lee, J. Plummer, R. Howe
You can reorder the anodes/cathodes however you like. Would a different ordering make the relationship simpler?
Mapping between 2-D and 3-D

A \times x \times 2 \times 3 \times z

B \times x \times 2

G \times x \times 2

y = 1 \Longleftrightarrow \text{assert } C \text{ or } F

2018

0 \lor 1

0 \lor 1

May 12, 2017

May 12, 2017
Mapping between 2-D and 3-D

\[ x = 0 \iff \text{ cascade 1 or 2} \]
What do bitwise operations do?

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

Examples (in binary):

```
00110101 | 01100011 == 01110111
00110101 & 01100011 == 00100001
```
Implementing `getLEDState()`

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.)
Decomposition

**loop()**

The main Arduino loop

calls

display()

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

which calls

getLEDState()

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:

```c
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");
}
```

May 12, 2017
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!");
  }
  print("winning!"); // always run.
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!
  
  *Leave one or two analog pins free*  
  *Don’t use pins 0 or T*