LED Cube

Design Review
ECE 445: Senior Design
Team #4

Michael Lin, Raymond Yeh
TA: Joseph Shim

University of Illinois, Urbana-Champaign
October 2, 2013
Table of Contents

1. Introduction
   1.1. Statement of Purpose 3
   1.2. Objective 3

2. Design
   2.1. Block Diagram 4
   2.2. Block Description 5
      Power Supply 5
      LED Cube 5
      Controller 7
      Communication 9
      Sensor Array 10
      Android Device 10

3. Requirements and Verifications
   3.1. Requirements and Verifications 11
   3.2. Tolerance Analysis 13

4. Safety
   4.1. Building the LED Cube 14
   4.2. Operating the LED Cube 14

5. Cost and Schedule
   5.1. Cost 15
   5.2. Schedule 16

6. Ethical Issues 17

7. References 18

8. Appendix A
   8.1 Pseudo Code for LED Light Control 19

9. Appendix B
   9.1 Overall LED Cube Schematic 21
1. Introduction

**Title: LED Cube**

LED technology is more advanced and much more efficient than traditional incandescent light bulbs and as such our team decided we wanted to build a device related to LEDs. An LED cube is inherently aesthetically pleasing and ours will be capable of displaying 3D animations and lighting patterns with much increased complexity compared to any 2D display of comparable resolution. Environmental interaction and Android Bluetooth connection will also be able to control the various lighting effects on the cube. Although our plan is for a visually pleasing cube, our implementation can easily be adapted for more practical applications such as displaying 3D models.

**Objective**

*Goals and Function:*

Our project goal is to use 64 RGB LEDs to construct a 4x4x4 cube with a single serial connection. The cube will have three modes of operation-off, programmed animation, and environmental response. There will be several animations that can be cycled through during the operation of the cube. An IR rangefinder and ambient light sensor will be used so the LED cube will react to environmental changes by changing color and brightness of each LED in the cube. The mode of operation of the cube as well as cycling animations will be controlled wirelessly through Bluetooth using an Android application.

*Features:*

- Display programmed animations
- Capable of displaying any RGB specifiable color
- Bluetooth wireless control
- Environmental control of light patterns

*Benefits:*

- Aesthetically pleasing
- Easily controllable
- Inexpensive
2. Design

Block Diagram

Figure 1: Overall LED Cube Block Diagram
Block Description

**Power Supply**

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>• 120 VAC 60Hz from wall outlet</td>
<td>• 5 VDC to modules</td>
</tr>
</tbody>
</table>

The power module supplies power to the Sensor Array, Controller, LED Cube, and Communication module.

**LED Cube**

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>• 4 digital inputs from Controller (1 serial data input, 1 clock input, 1 latch input, 1 blank input)</td>
<td>• LED outputs</td>
</tr>
</tbody>
</table>

*Figure 2: Partial Circuit Schematic for LED Cube*
The LED cube takes input from the controller module. It consists of 64 RGB LEDs in a 4x4x4 cube and is controlled by the controller module. This is used to display the light patterns.
**Controller**

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
</table>
| • 5 VDC from Power Supply  
• 1 analog input from Sensor Array  
• 1 digital input from Sensor Array  
• 1 digital input from Communication Module | • 4 digital outputs to LED cube  
• 1 digital output to Communication Module |

This module will take input from the communication and sensor array module and then output controls to the LED Cube. The controller module will compute and output several different lighting patterns to the LED cube; this includes lighting patterns that depend on sensor module outputs from environmental inputs. The controller will also take input from the communication module to switch between modes of lighting patterns.

**LED Indexing**

The microcontroller program will output a serial 52 bit control command to the shift registers for the LED cube to display the different lighting patterns. The 52 bits control is indexed as follows with serial communication being MSB first.

- [0 ... 15] for controlling the Red LEDs  
- [16 ... 31] for controlling the Green LEDs  
- [32 ... 47] for controlling the Blue LEDs  
- [48 ... 51] for controlling the level of LEDs

The underlying theory to this method of driving the cube is that at each point in time, there is only one “level” of the LEDs that is going to be on; meaning only 16 LEDs on at a time. The cube has been constructed to follow this demultiplexed method of control. Then each layer will have a certain time period to turn on or off. Because the entire cube will refresh with a rate of more than 100Hz, the average person won’t see the flickering of the LEDs. With this setup, we will be able to index to all of the LED diodes in the cube individually.

Since the above setup only allows each LED to be turned on and off we use the technique of bit angle modulation in order to display color. The idea is similar to PWM where the duty cycle of a square wave on separate diodes is used to vary the combined color. Bit Angle modulation (BAM) has multiple pulses in the equivalent of a single period of a PWM square wave. For the LED cube, a 4-bit brightness resolution will be used. Assuming that the refresh rate for the cube is set at 100 Hz (the best refresh rate will be experimentally determined) each cycle takes 0.01 sec.

Bit 3 will then pulse for half of 0.01 sec = 0.005 sec. Bit 2 will pulse for half of what bit 3 had--0.0025 sec. Continue to divide, then Bit 0 will have the bit taking 0.000625 second. By using these four bits and adding their pulse times together we achieve an equivalent of PWM.

Finally, to ensure that human eyes cannot perceive the flickering of the LEDs in this implementation the refresh rate have to be higher than 100 Hz. So by setting the interrupt to be 100Hz, each layer must be updating at 400Hz since there are 4 layers to update.
inside the 100Hz cube update. The Arduino UNO has a 16 MHz system clock. This means if the interrupt is at every 16 MHz / (400Hz) = 40,000 clock cycles. Then it takes 52 clock cycles at 8 MHz or 104 cycles at 16 MHz to update the shift registers. As can be seen, using a cube refresh rate of 100Hz, we find that the Arduino has 40,000 - 104 = 39896 clock cycles to complete other tasks.

**LED Lighting Patterns**

Once each of the LED can be indexed then the lighting pattern will be control by for loops that will iterate through all the LEDs and set the desired color to the LED during the process.

**Bluetooth Connection**

For the Bluetooth connection, the Serial library for the Arduino to received serial data from the Android device. Then serial library has two key functions, serial.available(), and serial.parseInt().

serial.available() will return the number of bytes available for reading from the serial port. serial.parseInt() will return the next integer value from the serial port.

Basically, the idea is periodically check the serial.available indicator so that when the number is greater than 0—i.e. there is data to be received—serial data is parsed. The integer value parse will correspond to which mode the LED cube should operate in. If necessary start and end characters will be used to avoid parsing at incorrect location. The necessity for this will have to be experimentally determined.

*Reminder: Pseudo code is attached in appendix A.*
Communication

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>• 5 VDC from Power Supply</td>
<td>• 1 digital output to Controller</td>
</tr>
<tr>
<td>• Wireless serial input from Android device</td>
<td></td>
</tr>
<tr>
<td>• 1 digital input from controller</td>
<td></td>
</tr>
</tbody>
</table>

The communication module takes serial data wirelessly from the paired Android device and outputs to the controller; the output to the controller is used to switch modes and lighting patterns.

Sensor Array

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>• 5 VDC from Power Supply</td>
<td>• 1 analog output to Controller</td>
</tr>
<tr>
<td>• Environment conditions</td>
<td>• 1 digital output to Controller</td>
</tr>
</tbody>
</table>
The sensor array module outputs to the controller module. This module consists of an IR rangefinder and an ambient light sensor which both collect data about the environment and will be used to determine lighting patterns.

**Android Device**

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>User input on screen (buttons)</td>
<td>Wireless serial output to Communication module</td>
</tr>
</tbody>
</table>

The sensor array module outputs to the controller module. This module consists of an IR rangefinder and an ambient light sensor which both collect data about the environment and will be used to determine lighting patterns.

**Android Device**

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>User input on screen (buttons)</td>
<td>Wireless serial output to Communication module</td>
</tr>
</tbody>
</table>

The Android device module is either an Android tablet or an Android phone. This module will output to the communication module via Bluetooth, in order for the controller to know when switch modes of operations.
## 3. Requirements and Verification

### Requirements and Verifications

<table>
<thead>
<tr>
<th>Requirements</th>
<th>Verification</th>
</tr>
</thead>
</table>
| **1. Power** | a) Test each module to make sure that the maximum power consumed by each part does not exceed: Microcontroller-0.5W, TLC5940-0.13W, LED cube-2.88W, 74HC595-0.35W, MIC2981-0.03W, IR rangefinder-0.2W, light sensor-0.02W, Serial Bluetooth-0.2W.  
   b) Using an oscilloscope to confirm the output voltage of the power supply stays within the 5±1V and ripple of 1V bounds. |
| a) Supply at least 5W for the system including, LED Cube, controller, sensor array, and communication.  
   b) Supply 5±0.5V output voltage and an output voltage ripple not exceeding 0.5V. | |
| **2. Controller** | a) A test circuit with the shift-registers will be built and using LED or multi-meter to confirm that the shift registers output the data inputted by the user.  
   b) A test circuit with the IR range sensor and light frequency sensor will be built. And observe if the control signal changes according to our specifications. The ambient light and range sensors will be tested separately to make sure that both can be used to control the brightness of a single LED.  
   c) Connect the range sensor and light frequency sensor to the Arduino and use this sensor data to drive LEDs on the Arduino to confirm that the Arduino is correctly receiving sensor data and outputting the control signals to drive the cube in the expected manner (color and motion).  
   d) Confirm Bluetooth connection by LED and check that serial data is received on the Arduino by running simple code to change LED status. Observe that cube changes lighting animations and modes when the change animation signal is received. |
| a) Compute and shift out data serially to the shift registers with a clock speed of at least 8MHz.  
   b) Compute and output control signal to for displaying environmental sensing lighting patterns.  
   c) Receive input from the sensor array, both ambient light and range sensor.  
   d) Switches between lighting animations and modes when signal received from the communication block.  
   e) The minimum base clock of the microcontroller must be greater than 8MHz. | |
3. LED Cube

- **a)** No shorts or non-conducting connections in the cube. The diodes in the RGB LEDs used should emit at least 1000 mcd of brightness each.
- **b)** When control signal is sent from the controller module the LEDs light up with the correct color and brightness.
- **c)** Shift registers and LED drivers shift in data at a clock speed of at least 8MHz.

4. Communication

- **a)** Bluetooth serial communication is received on the Arduino at the baud rate of 9600.
- **b)** No errors occur in the received data when the Bluetooth module loses connection or power.

5. Sensor Array

- **a)** The IR rangefinder outputs distinct voltages for distances between 10cm and 50cm in increments of 2.5cm.
- **b)** Light frequency sensor able to output sensor data that reflects the light intensity.

---

e) Use an oscilloscope to test if the frequency of the clock on the microcontroller exceeds 8MHz

a) Test each LED individually before soldering into a cube, test columns before put into cube, test planes before put into cube, then use a test input of the controller and observe if the entire cube lights up.

b) Use test inputs from the controller to observe if the LED cube demonstrates the correct color and brightness output according to the user input.

c) Test different clock speeds for the serial data line to shift into the cube to ensure that it continues to correctly shift in data with at least an 8MHz clock.

a) Check if the input is received by sending out test inputs from the Android Device, by using the app “Bluetooth SPP”.

b) Confirm the operation of the microcontroller continues without errors—i.e. unexpected microcontroller behavior—when the Bluetooth module loses connection.

a) With an input voltage of 5±1V to the sensor, move a white box or sheet of paper along the distance measurement axis and at points 2.5cm apart from between 10cm and 50cm check that the voltages are distinct by using a multi-meter on the output. Also measure the output voltage with the multi-meter and record these values along with their corresponding distance to determine an equation that models the distance-output voltage relationship of the sensor. This equation can later be used by the controller to determine distances of
b) Use a ceiling light or lamp above the light sensor and measure the output signal frequency on an oscilloscope to confirm that the frequency of the output signal is linearly proportional to the amount of light between 0.001 and 1000 μW/cm² with at least 85% confidence interval.

6. Android Device
   a) Able to transmit Bluetooth signals to the communication block.
   a) Check if the communication block receives the signal by using an LED on the Arduino. If the signal is received the LED goes on.

**Tolerance Analysis**

The component that most affects the projects performance is the power component. This is because the power supply has to function properly in order for all the other blocks to work.

For correct operation of the shift registers, the supply voltage cannot go below 3VDC or above 6VDC otherwise the chips will either not output or they will go above their rated heat dissipation limits.

The serial Bluetooth module must receive between 3.6 and 6VDC to ensure operation. If this is not met the Bluetooth module may frequently disconnect or not output serial data.

For operation of the project, the power supply has to provide more than 5W. This can be verified by using an ammeter and/or voltmeter on the terminals from the power supply to the LED cube circuit. If the 5W requirement is not fulfilled then the LEDs may not light up or they may be dim and less aesthetically pleasing because of this.
4. Safety

**Building LED Cube**

The main safety concern when building the LED cube is soldering the LEDs to the LED cube. This is because soldering is a potentially very hazardous task if not done correctly. The following few safety rules will be follow during soldering.

1. Never touch the tip of the soldering iron.
2. Wear eye protection when soldering.
3. Use lead free solder when possible.
4. Always wash hands after soldering.
5. Always solder in a well-ventilated area so toxic fumes are not inhaled.

**Operating the LED Cube:**

When operating the LED cube, be aware that the flashing lights and patterns that are displayed on the LED cube may cause a very small percentage of people to experience a seizure. Please turn off the LED cube immediately if strange or unusual body movement developed.
5. Cost and Schedule

Cost Analysis

**Parts**

<table>
<thead>
<tr>
<th>Item</th>
<th>Part Number</th>
<th>Qty</th>
<th>Unit Cost</th>
<th>Total Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>microcontroller board - Arduino UNO</td>
<td>-</td>
<td>1</td>
<td>$27.83</td>
<td>$27.83</td>
</tr>
<tr>
<td>16 channel constant current sink SIPO LED driver</td>
<td>TLC5940</td>
<td>3</td>
<td>$5.52</td>
<td>$16.56</td>
</tr>
<tr>
<td>8 bit SIPO shift register</td>
<td>74HC595</td>
<td>1</td>
<td>$0.63</td>
<td>$0.63</td>
</tr>
<tr>
<td>8 channel High-current source driver array</td>
<td>MIC2981</td>
<td>1</td>
<td>$3.03</td>
<td>$3.03</td>
</tr>
<tr>
<td>Common Anode 5mm RGB LED</td>
<td>MA475</td>
<td>100</td>
<td>$0.0999</td>
<td>$9.99</td>
</tr>
<tr>
<td>Serial Bluetooth Transceiver</td>
<td>RS232</td>
<td>1</td>
<td>$8.86</td>
<td>$8.86</td>
</tr>
<tr>
<td>IR Rangefinder</td>
<td>GP2Y0A21YK</td>
<td>1</td>
<td>$13.95</td>
<td>$13.95</td>
</tr>
<tr>
<td>JST jumper wire</td>
<td>SEN-08733</td>
<td>1</td>
<td>$1.50</td>
<td>$1.50</td>
</tr>
<tr>
<td>Light-to-frequency sensor</td>
<td>TSL235R</td>
<td>1</td>
<td>$2.95</td>
<td>$2.95</td>
</tr>
<tr>
<td>0.1 uF capacitor for light sensor</td>
<td>-</td>
<td>1</td>
<td>$0.20</td>
<td>$0.20</td>
</tr>
<tr>
<td>Resistor for Red diode current control</td>
<td>-</td>
<td>1</td>
<td>$0.10</td>
<td>$0.10</td>
</tr>
<tr>
<td>Resistor for Green diode current control</td>
<td>-</td>
<td>1</td>
<td>$0.10</td>
<td>$0.10</td>
</tr>
<tr>
<td>Resistor for Blue diode current control</td>
<td>-</td>
<td>1</td>
<td>$0.10</td>
<td>$0.10</td>
</tr>
<tr>
<td>Total:</td>
<td></td>
<td></td>
<td></td>
<td>$85.80</td>
</tr>
</tbody>
</table>

**Labor**

<table>
<thead>
<tr>
<th>Name</th>
<th>$/Hour</th>
<th>Hours per week</th>
<th>Number of Weeks</th>
<th>(Total/Person)*2.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raymond Yeh</td>
<td>$35.00</td>
<td>15</td>
<td>12</td>
<td>$15,750</td>
</tr>
<tr>
<td>Michael Lin</td>
<td>$35.00</td>
<td>15</td>
<td>12</td>
<td>$15,750</td>
</tr>
<tr>
<td>Total:</td>
<td></td>
<td></td>
<td></td>
<td>$31,500</td>
</tr>
</tbody>
</table>

**Grand Total**

<table>
<thead>
<tr>
<th>Total Labor</th>
<th>Total Parts</th>
<th>Grad Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>$31,500</td>
<td>$85.80</td>
<td>$31,585.80</td>
</tr>
<tr>
<td>Week</td>
<td>Raymond Yeh</td>
<td>Michael Lin</td>
</tr>
<tr>
<td>-------</td>
<td>-------------------------------------------------</td>
<td>-------------------------------------------------</td>
</tr>
<tr>
<td>9/16</td>
<td>• Finalize proposal &amp; order parts</td>
<td>• Finalize proposal &amp; order parts</td>
</tr>
</tbody>
</table>
| 9/23  | • Test the Bluetooth communication between Arduino and Android  
• PCB Tutorial | • Design and order template from machine shop  
• Complete circuit design for project  
• PCB Tutorial |
| 9/30  | • Design Review  
  - Microcontroller & Bluetooth  
  - Safety & Ethics  
  - Requirements  
  - References | • Design Review  
  - Introduction  
  - LED Cube Circuit  
  - Parts & Labor  
  - Verification |
| 10/7  | • Perform tests on small scale prototype of cube. Be able to control simple light patterns, such as all on, or a few LEDs on. | • Start building LED cube  
• Prototype sensor module. Test and record typical voltages and frequencies from sensors. |
| 10/14 | • Test sensor module output data on the Arduino; make sure that sensor data is received. | • Design and print PCBs for separate modules |
| 10/21 | • Implement lighting patterns on the Arduino microcontroller, including taking inputs from the sensor data. | • Implement code to output to cube  
• Continue work on LED cube |
| 10/28 | • Implement an Android application for control. | • Finish building LED cube |
| 11/4  | • Revise the Arduino microcontroller programming  
• Test functionality including Bluetooth | • Test LED cube and make any fixes as necessary  
• Integrate LED cube onto PCB for cube |
| 11/11 | • Test lighting patterns and functionality of switching between modes. | • Test LED cube functionality on PCB |
| 11/18 | • Fix any remaining bugs in the programming. | • Fix any remaining bugs in the hardware, circuit |
| 11/25 | • Thanksgiving Break | • Thanksgiving Break |
| 12/2  | • Demo & Presentation | • Demo & Presentation |
| 12/9  | • Final Paper & Check Out | • Final Paper & Check Out |
6. Ethical Issues

For this project, we will follow the IEEE code of Ethics along with our own moral standards. Our project process and our final product will adhere to the following relevant IEEE Code of Ethics.

a) Our project, the LED cube, is potentially enjoyed by people of all ages and many different locations. Therefore, our product must consider the safety, health and welfare of the public; in order to protect the users of our product.

1. To accept responsibility in making decisions consistent with the safety, health, and welfare of the public, and to disclose promptly factors that might endanger the public or the environment;

b) The description, data, and design of our project will be honest and realistic. We ensure that all claims about our project are real and truly reflect our project.

3. To be honest and realistic in stating claims or estimates based on available data;

c) We will accept and seek honest criticism of our technical work in order to improve our project, and giving credit to whoever contributed to our project.

7. To seek, accept, and offer honest criticism of technical work, to acknowledge and correct errors, and to credit properly the contributions of others;

d) We will adhere to the code of ethics; additionally make sure that all group members do follow the code of ethics throughout this project, as well as help each other for professional development.

10. To assist colleagues and co-workers in their professional development and to support them in following this code of ethics;
7. References


Appendix A

Pseudo Code for LED Light Control

#include <SPI.h> //Use the Serial Peripheral Interface (SPI) library to shift data into
    //registers for control

//Pin Section
//Define the pin number used by the shift registers
#define enable_pin 2 //Pin 2 to control the enable of the shift register
#define output_pin 11 //The output pin number to connect to shift register
#define output_clk_pin 13 //The clock pin used by shift register

//Variable Section
//Define the following variable to keep track of the LEDs
//Using Bit angle modulation, with resolution of 4, need 4 array for each color.
byte red3[8], red2[8], red1[8], red0[8];
byte green3[8], green2[8], green1[8], green0[8];
byte blue3[8], blue2[8], blue1[8], blue0[8];

int z_index = 0; //This index which layer currently the shift registers is shifting into
int mod_count = 0; //This will count through the BAM
int light_mode = 0; //This keeps track of which lighting mode, the cube is in

//Setup Section
void setup(){
    Setup clock, mode, and refreshing rate variable for the library
    Setup the pins defined above to outputs
    Setup Bluetooth baud rate
}

//Loop Section
void loop(){
    switch(light_mode){
    case 0:
        //LED Cube is off
    case 1:
        //Display Light Pattern 1
    case 2:
        //Display Light Pattern 2
    case 3:
        //Display Environmental lighting patterns.
        … //more cases if more light patterns
    char data_recieved = ' '; //Initialize a variable to Bluetooth received
    if(Serial.available()<1) return //Check if Bluetooth available, if not keep looping

    set data_recieved to the data received from Bluetooth
    set light_mode to the integer parsed the serial data from Bluetooth
    }
}
```c
//Interrupt Section
ISR(interrupt_rate){
    //This routine is the interrupt section that outputs the values to the shift registers.
    if(mod_count == 32)
    {
        shift out the data stored in red3,green3,blue3 to the output;
        hold the data till the next output time.
    }
    if(mod_count ==16)
    {
        shift out the data stored in red2,blue2,green2, to the output;
        hold the data till the next output time.
    }
    if(mod_count == 8)
    {
        shift out the data stored in red1 blue1 green1, to the output;
        hold the data till the next output time.
    }
    if(mod_count ==0)
    {
        z_index ++;
        shift out the data stored in red1 blue1 green1, to the output;
        hold the data till the next output time.
    }
    Shift out which level we are outputting.
    if(z_index == 4){
        z_index =0; // reached the top level, go back to layer 0;
    }
    mod_count++; 
    if(mod_count == 64){
        mod_count =0; //finished the modulation go back to 0;
    }
}
void write_to_RGB(int x,int y, int z, byte red, byte green, byte blue){
    //First in the RGB arrays, the size is 64, bits, where each LED can be index by the
    //following equation index = x+4*y+16*z.
    //Then write the corresponding color and bit number to the location.
    set red3[index] to 3 bit of red.
    set red2[index] to 2 bit of red.
    set red1[index] to 1 bit of red.
    set red0[index] to 0 bit of red.
    ...//similarly for blue and green.
}
```
Appendix B

Overall LED Cube Schematic