Gesture-Controlled LED Coffee Table with BLOCK LETHAL OBLITERATOR XTREME

Design Review Presentation

Josh Pack | Esther Kim | Ryan Dwyer

Project #16

October 5, 2012

ECE 445 Fall 2012

TA: Lydia Majure

Contents

1. Introduction

Those classic, oversized arcade games like Pac-Man and Space Invaders just don't fit in with the atmosphere of a relaxing, stylish living room. We want to solve this problem by putting a modern twist on a retro video game. Our idea is to design and build a coffee table with an LED matrix display surface on which we can implement the game BLOCK LETHAL OBLITERATOR XTREME, or BLOX. The game is widely known as Tetris. We will use microcontrollers to control the LEDs. Some projects along these lines have already been completed, but we want to make the game much more interactive by adding motion sensors, which would allow control of the game through hand gestures. We also want to make the appearance of the table more refined, adding light-diffusing films above the LEDs to give a warm glow. The surface will be sturdy enough to be used as a common coffee table, as well.

Objectives

Goal: Our project goal is to have a fully functional coffee table with an LED matrix display surface that can play BLOX using gesture recognition technology.

Functions: The intended functions are to have a coffee table that users can play BLOX using hand gestures above the table to drop, move, and rotate the pieces. The BLOX game will have the basic functions of removing a row when it fills, randomizing the next piece, and ending the game when a piece has reached the top row.

Benefits: Benefits of the LED BLOX Coffee Table include:

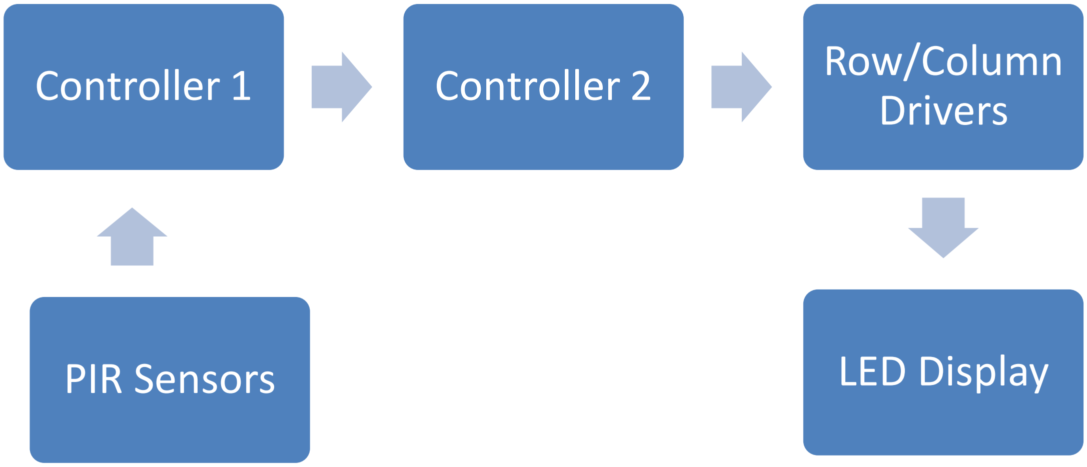
- Entertainment to users with the BLOX game
- Aesthetically pleasing in the living room
- Supportive coffee table

Features: Features of the LED BLOX Coffee Table include:

- Rows are removed when it fills
- The next piece is randomized
- Game ends when a piece reaches the top row
- Drop, move, and rotate are controlled with hand sensors

2. Design

Block Diagram



Block Descriptions

Controller 1:

(1) Arduino Uno board
Controller 1 will be an Arduino Uno used to run the main program, which will take input from the PIR sensors to determine the user input and send control signals to Controller 2. This controller will keep track of the locations of the game pieces and the progress of the player. Controller 1 will notify controller 2 when a row is cleared or when the player has lost.

The Arduino Uno board has 14 digital I/O pins, 6 analog inputs, a 16 MHz crystal oscillator, and USB connection. The board can be powered through USB, an AC-DC adapter, or a battery. Operating voltage is 5V; recommended input voltage is 7-12V.

Controller 2:

(1) Arduino Uno board
Controller 2 will be an Arduino Uno used solely to interface with the drivers to render the display based on the received control signals from Controller 1. Having a second controller dedicated to creating the display will allow us to ensure a smooth image with minimal flickering while the first controller deals with executing the BLOX program.

(See controller 1 description for specs)

Row/Column Drivers:

Row Drivers:

(3) 74HC595 Shift Registers
The anodes of each row of the matrix will be connected together to a pin of the shift registers. The shift registers will shift through a “walking bit” sequence (1000) for each set of 4 rows, which implements LED matrix multiplexing.

(3) MIC2981 High-Current Source Drivers
The anode rows of the matrix will draw more than the maximum current for each shift register output pin; therefore we will use these drivers as a switch. The shift register outputs will connect to the inputs of these drivers, which will source much more current (up to 500mA/pin) to the anode rows when the shift register output is high.

Column Drivers:

(10) TLC5940
The LED drivers we will use are the TLC5940, which use PWM to control the LEDs to produce the appropriate color from the 7 options in our BLOX design. They also supply the proper driving voltage for the LEDs without having to connect each LED to a power supply through a resistor. Controller 2 will send control signals to the appropriate LED drivers to create the desired display on the LEDs.

The TLC5940 features 16 channels with dot correction and grayscale PWM control. DC operating voltage is 3V-5.5V. Power dissipation rating is 2W-4W. We expect to use between 7 and 13 of these to be able to properly control all 200 RGB LEDs.

LED Display:

(200) RGB LEDs
These RGB LEDs will be arranged in a 10x20 matrix such that each RGB LED represents one pixel. These LEDs will have diffused lenses so that any colors which combine the red, green, or blue will look pure rather than looking like multiple, separate colors. They will be controlled by the TLC5940 LED drivers, which use PWM for each of the red, green, and blue components to create the desired color. Our BLOX implementation will consist of seven different colors for game pieces.

We are planning to use 5mm 4-pin RGB diffused, common anode LEDs. These have an operating voltage of approximately 2.1V-3.4V.

PIR Sensors:

(3) Passive-Infrared (PIR) sensors
These three sensors will be used to implement gesture control of our BLOX game.

The sensors have an operating voltage of 3.3V to 5V and a sensing range of up to 20 feet.

3. Block Level Requirements and Verification

PIR Sensors

<table>
<tr><th>Requirement</th><th>Verification</th></tr>
<tr><td>**Gesture Detection**
Three sensors positioned on the left, middle, and right of the table must be able to distinguish five control motions for the falling BLOX pieces. The truth table is as follows:
<table>
<tr><th>LEFT</th><th>MIDDLE</th><th>RIGHT</th><th>OPERATION</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>--</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>Shift Right</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>--</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>Rotate CW</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>Shift Left</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>Drop</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>Rotate CCW</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>--</td></tr>
</table>
</td><td>Connect each of the sensor outputs directly to an LED so that we can observe the proper LEDs lighting when each of the control motion combinations is performed above the sensors.</td></tr>
<tr><td>**Sensitivity and Positioning**
Sensors must detect hand gestures from up to, but no more than a 1 feet radius.
Sensors must be positioned 1.2 feet away from each other and not overlap radii, eg. the left sensor picking up signal from motions meant for the middle sensor.</td><td>Adjust the sensitivity and measure with a ruler the ranges for all sensors.
Test the middle sensor by waving hand 12 inches to the right and then 12 inches to the left to make sure that the left and right sensors do not go off. Do the same for the left and right sensors.</td></tr>
<tr><td>**Power**
The sensors must be supplied with a voltage between 4.5 and 6 volts.</td><td>Check voltages using a multimeter.</td></tr>
</table>

Controller 1

Requirement	Verification
Sensor Input Communication Controller 1 must be able to interpret each of the five input controls from the PIR sensors.	Write a test case in the Arduino Software to display the input from the sensors. Check for all five operations.
BLOX Game Logic Controller 1 must keep track of the position of the game pieces so that the pieces will stack properly.	Write a test case in the Arduino Software to stack random pieces. Output the pieces’ grid positions in the Arduino software.

Controller 1 must remove a completed row in one cycle of the gravity rate.	Write a test case in the Arduino Software to drop pieces in a complete row to make sure controller 1 detects a complete row and removes this row. Output detection flag and deletion flag. Check that pieces above shift down and pieces below remain.
Controller 2 Communication Controller 1 must send control signals to controller 2 with instructions for the next cycle of the display.	Write a test case in the Arduino Software to ensure that the display updates at a smooth rate and that the control motions are actualized on the display in a responsive manner with no more than 16ms delay between control motion and piece movement.

Controller 2

Requirement	Verification
Controller 1 Communication Controller 2 must receive updated input from Controller 1 at rate of at least 60Hz.	We will connect an oscilloscope to the output control signals from controller 1 to controller 2 to ensure that they refresh at approximately 60Hz.

LED Drivers

Requirement	Verification
LED Communication The LED drivers must provide each LED in the display with the proper operating voltage, which ranges from 2.1V to 3.4V, and sink ~20mA for each LED. The LED drivers must use proper PWM on the red, green, and blue portions of each RGB LED such that each LED can light with each of the seven necessary colors (cyan, blue, orange, yellow, green, purple, red in accordance with The BLOX Company’s color scheme).	Use a voltmeter on each pin to ensure that the output voltage is in the operating range of 2.1V to 3.4V Connect one RGB LED to the output of one of the LED drivers with the controllers sending the proper signals to the driver to create the desired colors. We will repeat for each of the seven colors to make sure our duty cycles are visibly correct for each color.

Shift Registers

Requirement	Verification
LED Communication Each shift register must be able to serially shift walking bit sequence (10001000) 4 times per frame.	Place individual LEDs on each of the shift register output pins to make sure they display the desired walking bit sequence (10001000).

Power Shift registers can store and shift the sequence of 10001000 at 4 MHz. Should output the supply voltage with at most 35 mA when 2 output pins are on.	Run the shift register clock at 5V 4 MHz square wave. Power the chip at Vcc = 5V and load the 10001000 sequence. Use oscilloscope on each of the 8 output pins and check for the 10001000 sequence with the input of the next bit being the output of the last. This makes the sequence continuous. The output should be a Vcc=5V peak-to-peak square wave, on 1/4 of the time.

LED Matrix Display

Requirement	Verification
LED Brightness LED brightness reflects duty cycle of PWM on all RGB channels for accurate color.	Use function generator with a 1 kHz, 3V peak-to-peak square wave to output across the appropriate RGB pin and common anode. Not all 200 LEDs need be tested unless they stand out in the matrix.
For each red, green, and blue pin, test with duty factors 25%, 50%, 75%, and 100% to see that brightness increases respectively.	Use one LED to look for varying levels of brightness in a dark room.
Color Output LEDS must display the correct colors with the following duty factors (R, G, B) and subjectively determine that they match others in the matrix. (Red, green, and blue colors have already been tested at 100% duty factor) yellow: (100%, 100%, 0%) orange: (100%, 65%, 0%) cyan: (0%, 100%, 100%) purple: (67%, 0%, 100%)	Use one LED to look for all colors in a dark room. Repeat with one LED in each shift register and LED driver to ensure that the colors are the same across the matrix.
No Visible Flickering There must not be any noticeable flickering at any viewing angle to the LED.	At a duty factor of 65% for each RGB channel, look at the LED with it pointing directly at you and sweep across 90 degrees to be sure there is no noticeable flicker dependent on viewing angle at this 1 kHz. Also useful to try lowering the duty factor or frequency to find the minimum frequency/duty factor at which flickering becomes a problem.

Tolerance Analysis

The component that will most affect the performance of the project is the synchronization of the LEDs and its response time to gesture controls. This component matters the most because a responsive and synchronous display will influence the functionality of the BLOX game, as well as user interaction and overall enjoyment of the game.

To test the responsiveness of the system, we will start with a new BLOX game. We will set the middle PIR sensor signal to high for the BLOX pieces to drop continuously and as fast as possible. We will measure the time from the piece settling on the surface to a new piece appearing at the top of the display. We will also measure the gravity rate and the drop rate.

The expected time it takes to settle the old piece and have a new piece appear should be the same as the gravity rate. We will aim to have the gravity rate to be 500 ms per row. The tolerance for this time measurement is +/- 250 ms. The drop rate will be twice as fast, or 250 ms per row. When testing the drop rate, we will allow a tolerance of +/-125 ms.

Calculations

1. **LED Driver Current Sink**

   The constant current sink of each TLC LED driver is set by a reference current pin IREF on the driver. We need a 488ohm resistor to set the max current to ~80mA in accordance with the formula shown below from the TLC5940NT datasheet, so we will use 500ohm resistors to limit the current to 78mA.

$$I\_\mathrm{max} = 31.5 * \frac{V(IREF)}{R(IREF)}$$

$$where\ V(REF)\ \cong 1.24V\ typically$$

2. **Shift Register - High-Current Source Driver Interface**

   Each output pin of the shift register outputs 5V when high. Each input to the MIC2981 driver must be 2.4V at 140-200uA. Therefore, we must place a 13kOhm to 18.5kOhm resistor in the middle of each of these 20 connections to step down the voltage and limit the current. We plan to use 15k resistors.

3. **Power Supply**

   Our power supply must meet the following demands:

Component Name	Supply Voltage per Unit	Current per Unit	Power per Unit	Number of Units	Total Power
Arduino Uno	5V	200mA	1W	2	2W
LED Driver	5v	60mA	0.3W	10	3W
74HC595 Shift Register	5v	70mA	125mW	6	750mW
Resistor between shift registers and MIC drivers			0.375 mW	20	7.5 mW
MIC2981				3	
PIR Sensor	5V	60uA	0.3mW	3	<1mW
LED	3.4V	20mA	34mW	200	6.8W

Total Maximum Power = 12.8W

4. **PCB Size**

   PCB must be large enough for

   - 3 74HC595 shift registers
   - 3 MIC2981 high-current source drivers
   - 10 TLC5940NT LED drivers
   - 20 15k resistors

4. Application to IEEE Code of Ethics

The LED BLOX Coffee Table is a device meant for entertainment and should be used as such. Users should not tamper with the electronic parts of this device. In accordance with #9 in the IEEE code of ethics, we as engineers must "avoid injuring others, their property, reputation, or employment by false or malicious action." Safety is a concern when:

- Staring directly at powered LEDs without the diffusing film
- Using quick hand movements to activate the PIR sensors
- Moving the sturdy coffee table as it may be heavy

The diffusing film is meant to give the LED display a warm glow as well as provide protection to the user in the case of prolonged exposure to the light. Playing the BLOX game also requires the player's hands to move about, so the player should be aware of the surroundings. The LED BLOX Coffee Table is meant to serve as a living room table that can support common items placed on top of the display. This means that the table can be large and heavy. These safety concerns are few and may not affect most users who are careful with similar devices. We do not wish others to be harmed from the use of the LED BLOX coffee table.

The rights to Tetris belong to The Tetris Company, which is why we renamed our game BLOCK LETHAL OBLITERATOR XTREME, or BLOX. In doing so, we avoid copyright infringement.

5. Cost and Schedule

Cost Analysis

Parts

Part Name	Amount	Cost per Unit	Price (including shipping est.)
Arduino Uno Microcontroller	2	$23	$65
RGB LED	300	$12.34/100	$45
TLL5940 LED Driver	10	$3.50	$40 (free samples available)
74HC595 Shift Register	3	$0.35	$2
PIR Sensor	3	$10	$35
High Current Source Driver	3	$2	$10
PCB	1	$0	$0
Coffee Table	1	$20	$20
Light Diffusion Grid	1	$10	$15

Labor

Group Member	Rate	Total Number of Hours	2.5 x Total
Josh Pack	$40/hr	150	$15,000
Esther Kim	$40/hr	150	$15,000
Ryan Dwyer	$40/hr	150	$15,000

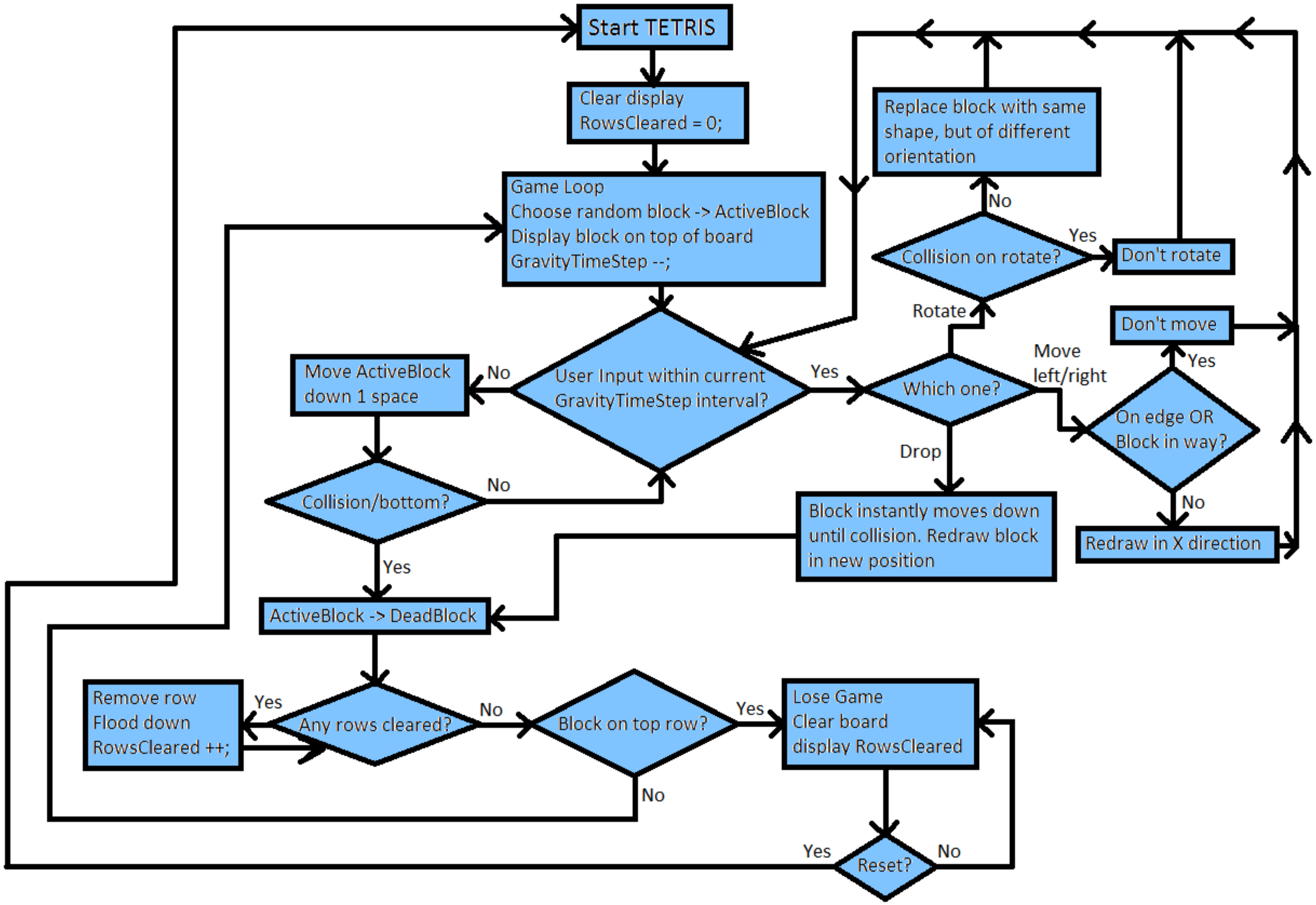
Parts Total Cost:	**$ 232**
Labor Total Cost:	**$ 45,000**
Grand Total Cost:	**$ 45,232**

Schedule

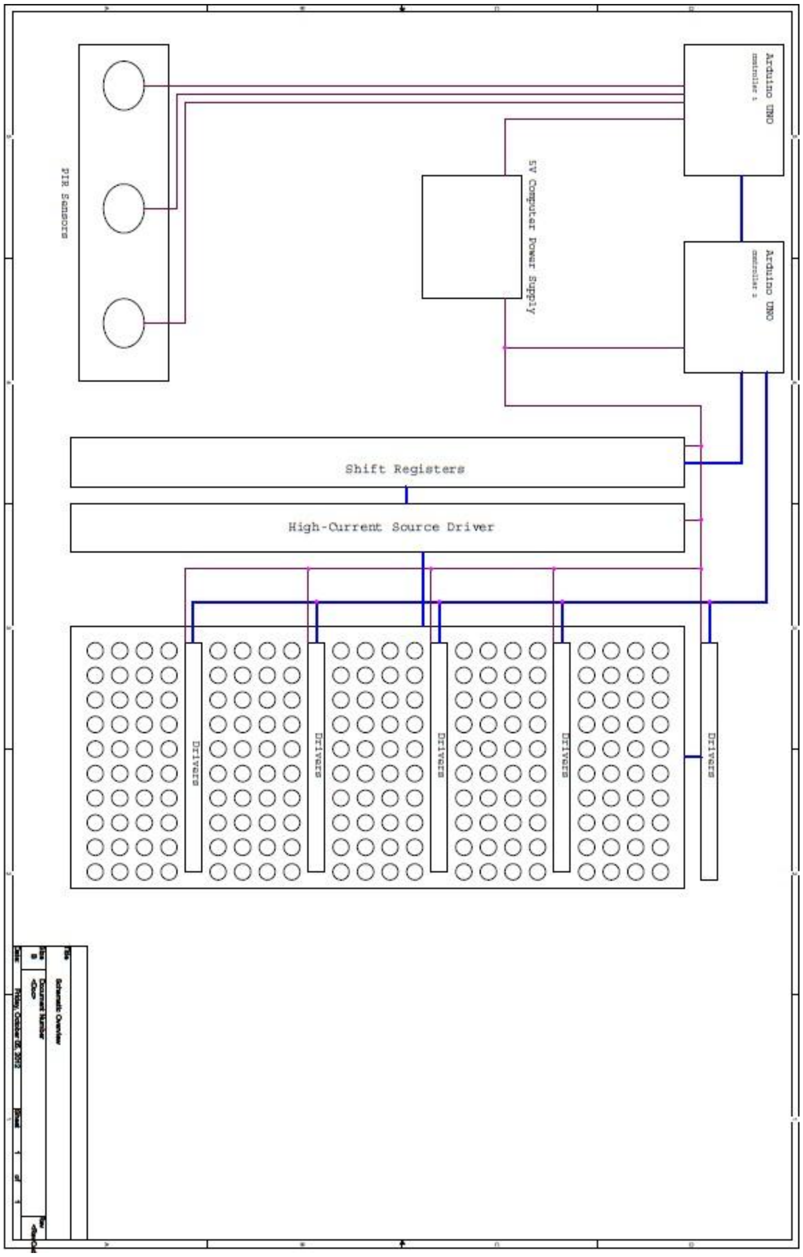
Week of	Milestone	Overall Task	Josh	Esther	Ryan
09/16	Proposals Due	Brainstorm design Modulate requirements	Get feedback from Lydia on gesture control and initial design Order Bundle 1* of parts		
09/23		Finalize design Gather parts	Order Bundle 2** of parts Write pseudo code for BLOX game		
09/30	Design Review	Feedback on design Tweak design Construct	Design LED matrix on Eagle	Program syncing and rendering on 2nd microcontroller	Program BLOX on 1st microcontroller
10/07		Construct	Submit PCB request Build LED Matrix display	Program syncing and rendering on 2nd microcontroller	Program BLOX on 1st microcontroller
10/14		Construct Test	Test the LED matrix display	Test LED matrix for synchronization and rendering	Write test cases to test the 1st microcontroller
10/21		Integrate Test	Write test cases for integration with LED matrix	Test the PIR sensors	Test the 1st microcontroller
10/28		Integrate Test	Integrate 1st microcontroller with the 2nd microcontroller and LED matrix	Integrate 1st microcontroller with PIR sensors	Test the LED drivers and the power supply
11/04	Mock-up Demo	Feedback on performance Debug	Integrate 1st microcontroller with the 2nd microcontroller and LED matrix	In charge of presentation PowerPoint	Test the shift registers
11/11	Mock-up Presentation	Debug	Build table with space for the LED matrix display, sensors, and power supply	Build the diffusion grid and surface top	Test the BLOX game on the LED matrix display
11/18	Thanksgiving Break				
11/25		Debug Finalize	Test the PIR sensors on the table	In charge of documenting test results	Test the LED matrix display on the table
12/02	Final Demo	Finalize	Test the BLOX game on the LED matrix display	In charge of presentation PowerPoint and paper	Test the BLOX game on the LED matrix display
12/09	Final Presentation Final Paper Due	Finalize	Test the BLOX game on the LED matrix display	In charge of presentation PowerPoint and paper	Test the BLOX game on the LED matrix display

*Bundle 1 includes Arduino microcontrollers, LEDs, LED drivers, PIR sensors

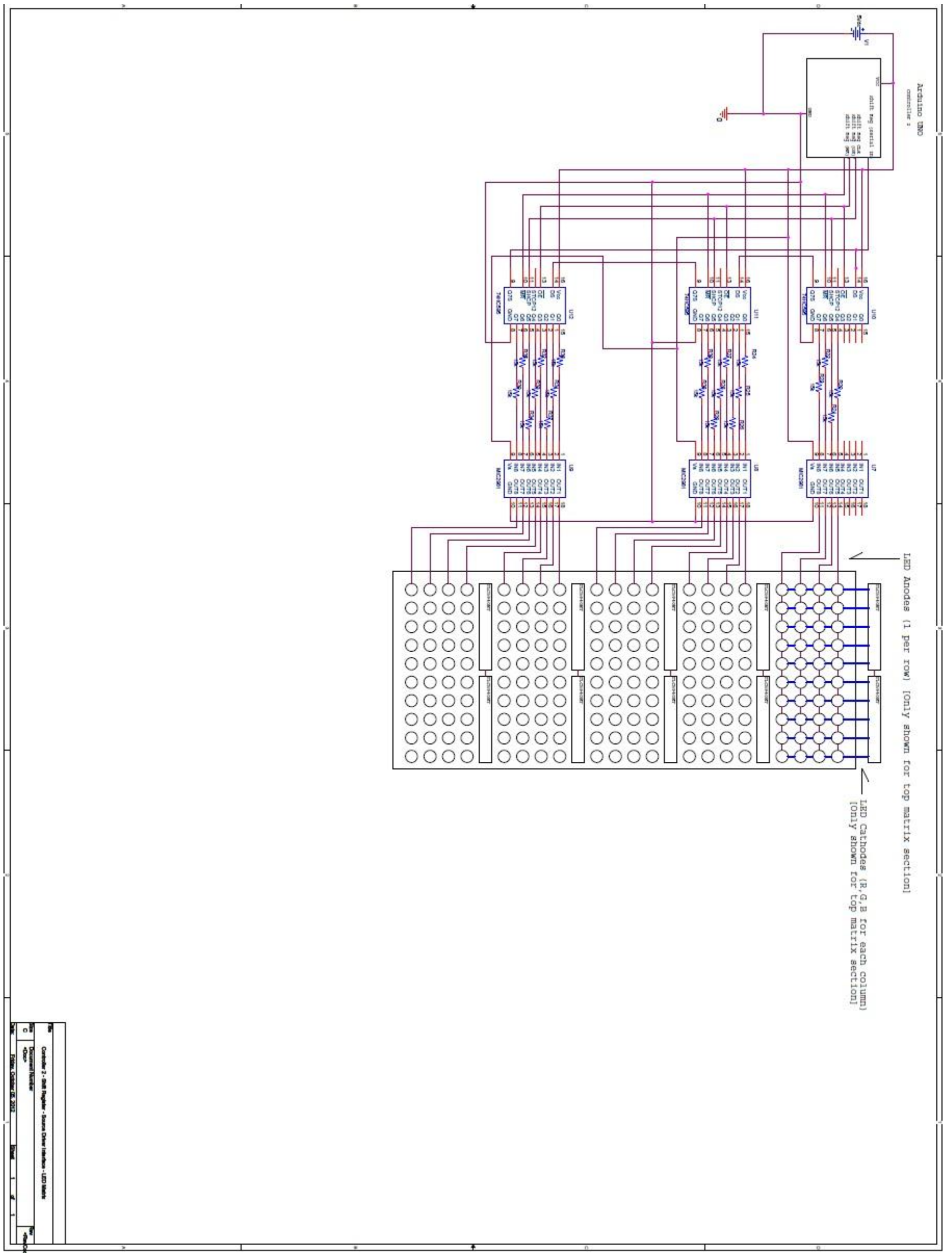
**Bundle 2 includes coffee table, light diffusion grid, additional hardware as needed


Start TETRIS
Clear display
RowsCleared = 0;
Game Loop
Choose random block -> ActiveBlock
Display block on top of board
GravityTimeStep --;
User Input within current
GravityTimeStep interval?
No
Move ActiveBlock
down 1 space
Collision/bottom?
No
Yes
ActiveBlock -> DeadBlock
Any rows cleared?
Yes
Remove row
Flood down
RowsCleared ++;
No
Block on top row?
No
Yes
Lose Game
Clear board
display RowsCleared
Reset?
Yes
No
Yes
Which one?
Rotate
Collision on rotate?
No
Replace block with same
shape, but of different
orientation
Yes
Don't rotate
Move
left/right
On edge OR
Block in way?
Yes
Don't move
No
Redraw in X direction
Drop
Block instantly moves down
until collision. Redraw block
in new position

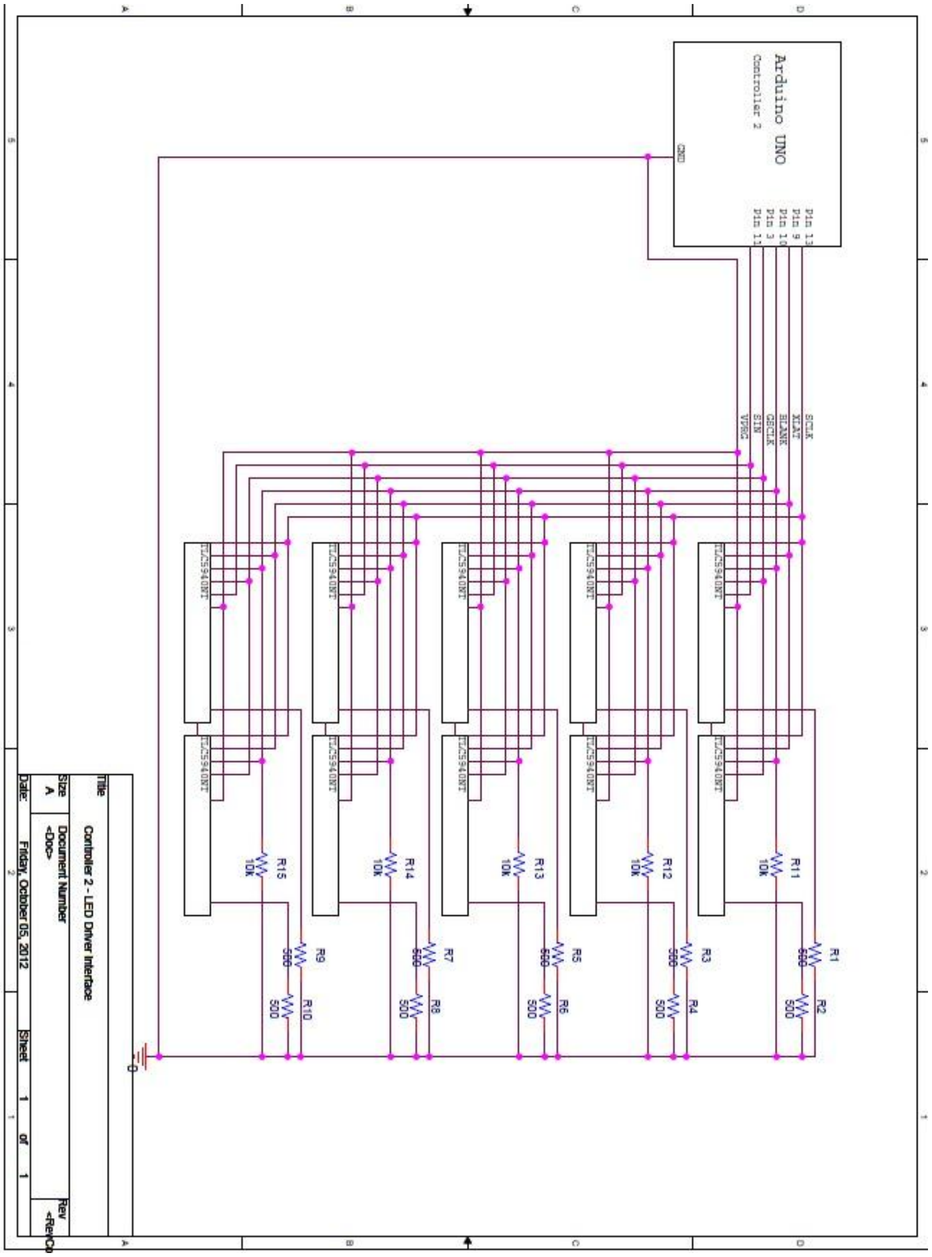
Schematic 1: High-Level Overview



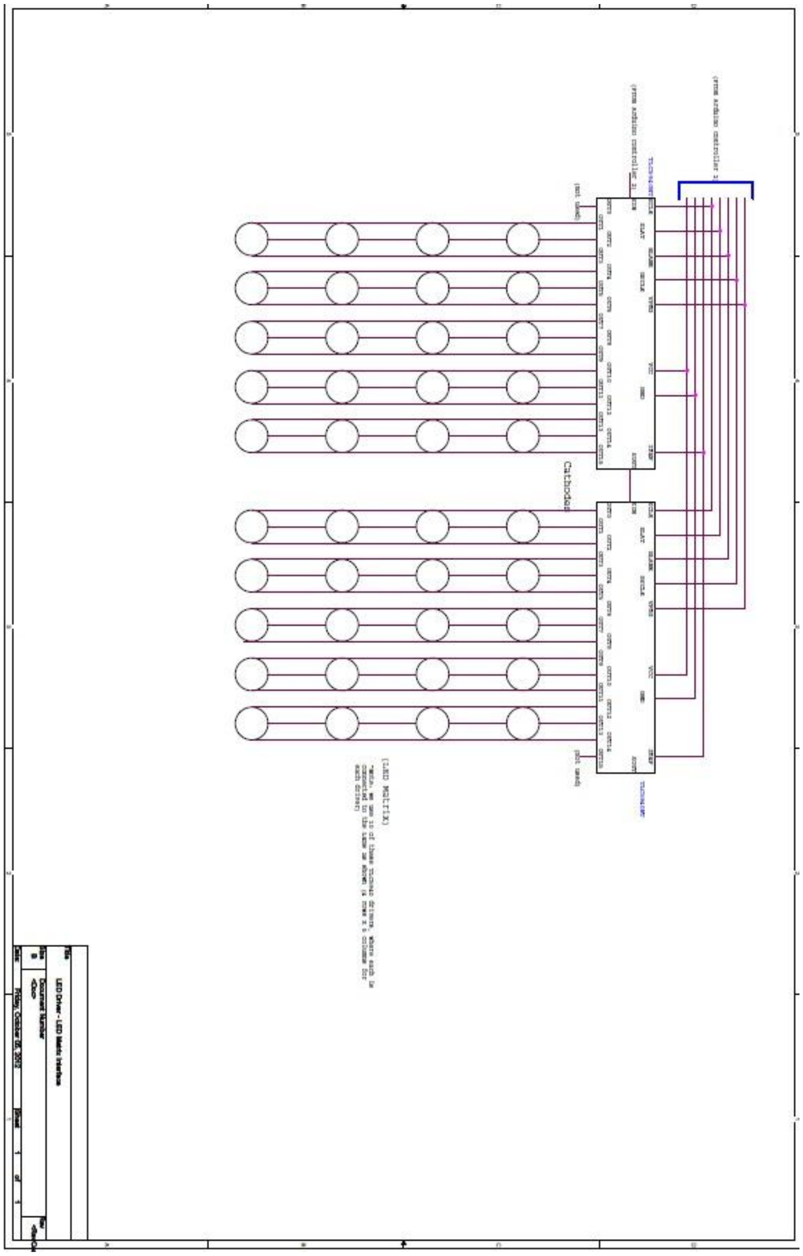
Schematic 2: Controller 2 – Anode Row Driver – LED Matrix Interface


Arduino UNO
controller 2
VCC
GND
shift Reg serial in
shift Reg clk
shift Reg latch
shift Reg OE
5Vdc
V1
0
74HC595
MIC2981
U7
LED Anodes (1 per row) [Only shown for top matrix section]
LED Cathodes (R,G,B for each column) [Only shown for top matrix section]
Title
Controller 2 - Shift Register - Source Driver Interface - LED Matrix
Size
C
Document Number
<Doc>
Rev
<RevCode>
Date: Friday, October 05, 2012
Sheet 1 of 1

Schematic 3: Controller 2 – Column Driver Interface (LED Matrix hidden)



Schematic 4: Column Driver – LED Matrix Interface (1 of 5 sections shown; row drivers hidden)


TLC5940NT
(from Arduino controller 1)
(from Arduino controller 2)
(not used)
SCLK
XLAT
BLANK
GSCLK
VPRG
VCC
GND
IREF
SIN
SOUT
OUT0
OUT1
OUT2
OUT3
OUT4
OUT5
OUT6
OUT7
OUT8
OUT9
OUT10
OUT11
OUT12
OUT13
OUT14
OUT15
Cathodes
(LED Matrix)
File
LED Driver - LED Matrix Interface
Size
B
Document Number
<Doc>
Date:
Friday, October 05, 2012
Sheet
1
of
1
Rev
<RevCo