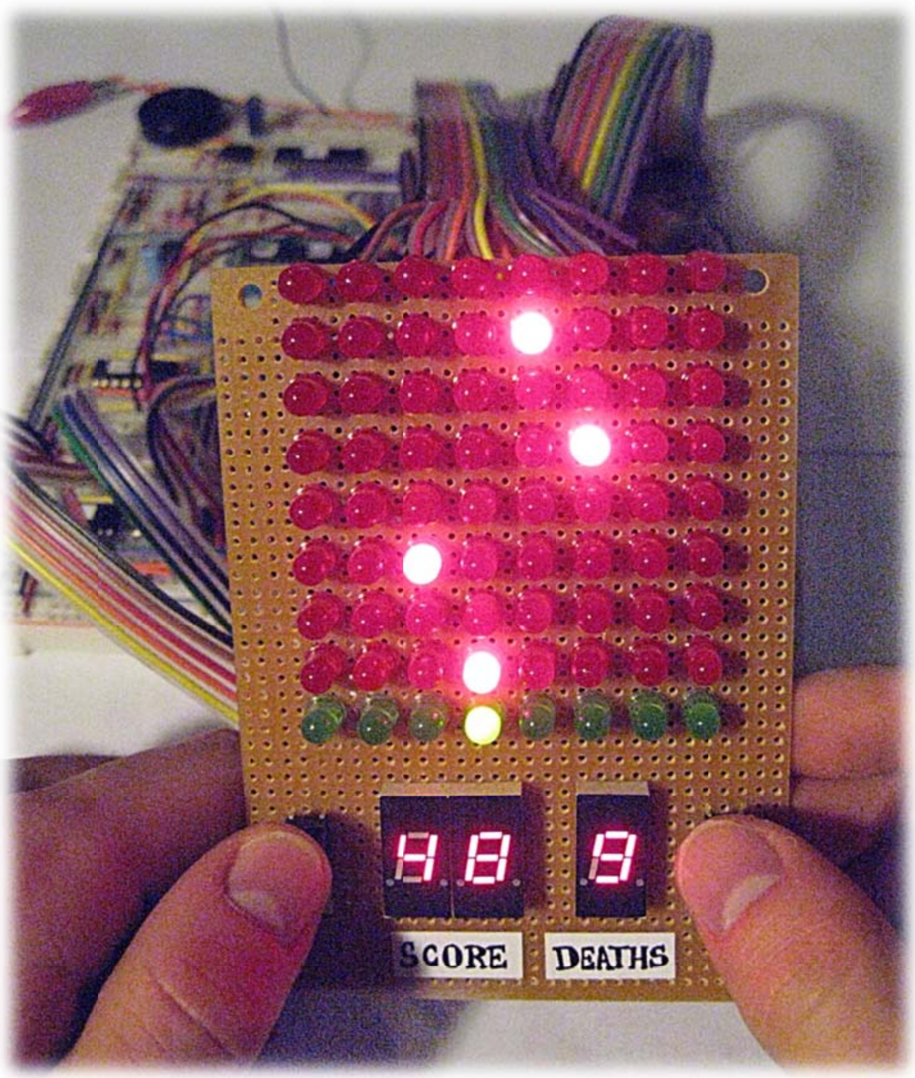
Catch or Die!



Julia A. and Andrew C.

ECE 150 – Cooper Union

Spring 2010

Andrew C. and Julia A.
DLD Final Project Spring 2010

Abstract

For our final project, we created a game on a grid of 72 LED's (9 rows by 8 columns) in which the user must move a light representing a platform from left to right to strategically to catch falling objects. The controls are two momentary pushbuttons. If the user successfully keeps the platform under a falling object while it is in the bottom row, the score will increment, as shown on the two-digit seven segment display. Otherwise, the death count will increment, as displayed on a third seven segment display. There are four different stages in the game, in which the objects fall at different rates, thus increasing the complexity and challenge of the game. The game ends when the user reaches 10 deaths. When this occurs, the game clock is disabled so the objects freeze in place on the screen to indicate that the game is over.

To implement the falling objects on the LED display, each row of LEDs was wired to a single 3:8 demux. We used an asynchronous counter connected to a fast clock to generate a pseudo-random 3 bit address for the top row, and then passed that address down one row at a time using 3 sets of flip flops. At the bottom, we used a comparator to compare the address of the falling object to the address of the platform light to determine whether the score or death count should be incremented. The platform row itself uses an up/down counter to increment or decrement the address selected depending on which button is pushed. We used a flip flop and fast clock to introduce just enough delay to allow the counter time to change counting modes before changing the count itself when a button is pressed.

Some additional logic and several other chips were required to fully implement this design, and they are mentioned on the full inventory list included in the documentation, along with a description of purpose.

Design, Implementation and Testing

Display

To create a 72 LED display, we decided to use a strip board and solder the LEDs ourselves to avoid the confusion and difficulties of a premade display. We soldered ribbon cable to each row of LEDs for easy organization. Each row also shared a single resistor. We decided to use this simplification since only one light in given row should ever be lit. The entire board shares one positive rail connected to the 9 resistors.

We also soldered the score and death count seven-segment displays and the control buttons on the same strip board to create a familiar handheld game interface. We used a power supply and alligator clips to test each LED and its connection to the ribbon cable before actually connecting it to the breadboards so that we knew any problems we witnessed would be caused by the logic or breadboards and not a faulty display.

A buzzer was added to ensure that the user would notice when an object was missed (and the death count incremented).

The Falling Objects

To implement the falling objects on the LED display, each row of LEDs was connected to a single 3:8 demux. This design choice was made possible by the fact that only one light in any given row can be lit at a time. Otherwise, the platform light would have to be in two places at once to catch them both, which is physically impossible. Similarly, because the nature of our comparison logic requires the user to keep the platform in the right place for a whole clock cycle, it would be impossible to catch two objects falling in adjacent rows in different columns. Also, having a light turned on in every row would look busy and confusing. As a result, we decided to only light LED’s in every other row.

To solve this problem, we added a fourth set of flip flops to pass an alternating signal (a fixed sequence of 010101…) to the control pins of the demuxes so that only every other demux is enabled in the same clock cycle. We generated this alternating sequencing using the same flip flop that shifts the address down, but tied the Q output to all of the odd numbered demuxes and Q’ to all the even ones, ensuring that two adjacent rows can never have a lit LED. This design choice also allowed us to cut the number of flip flops needed to pass the address along in half, since there can only be four unique addresses at a time instead of eight. Each pair of demuxes receives the same address, but only one is enabled at a time.

To test the passing of the addresses, we added a series of LEDs to the breadboard connected to one address pin of each of the 8 demuxes. Using this method, we tested the C, B, and A address bits independently to make sure the address was being passed correctly.

To initialize the address for the top row, we used an asynchronous counter connected to a fast clock to generate a pseudo-random 3 bit number, and then passed that address on to subsequent rows. At this point, we tested our circuit by connecting the full display to observe how all of the demuxes were behaving simultaneously. We observed that certain frequencies of the fast clock generated numbers that were not at all random and actually a very predictable pattern. We added a potentiometer to fine-tune this frequency to achieve the appearance of randomness.

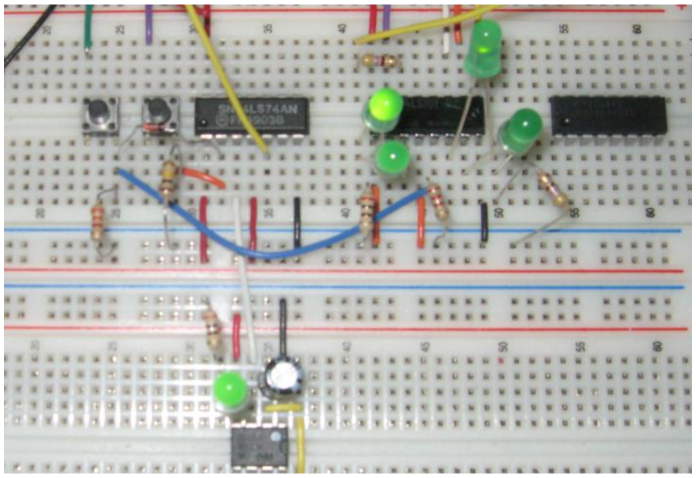


Figure 1. A demux being tested before the full circuit is constructed

The Speed Stages

To make the game more interesting and difficult, we decided to increase the frequency of the game clock as the game progresses to make the objects fall faster. To do this, we used a counter tied to the address of an analog mux/demux to select among different resistances for the RC circuit of the main 555 timer. We empirically chose 4 different resistor values for the different speeds. We tied the counter to the carry pin of the less significant score counter, so that address given to the analog mux would change every ten points. We had 8 pins to work with, and we doubled up resistance values so that each speed actually lasts for 20 points.

The Scoring Logic

We used a 4-bit comparator to compare the address of a falling object in the bottom row to the address of the platform light to determine whether the score or death count should be incremented. We had to introduce logic to convey information from the comparator to the counters for the score and deaths. One complication to be considered was that the death count should not be incremented if the bottom row demux is disabled, even though the address of that demux probably will not match the platform address. In other words, the death counter should only be incremented if neither the equal pin of the comparator NOR the inverse of the demux enable signal is high. We tied these two signals to the inputs of a NOR gate and connected the clock pin of the death counter to the output of this gate. Similarly, to increment the score, we used NOR logic to require that the bottom demux is enabled and its address matches the platform address.

The Game Controls

Two momentary push buttons allow the platform light to be moved left or right effectively as quickly as the user can press them. The platform row demux is tied to an up/down counter to increment or decrement the address selected depending on which button is pushed. We used a flip flop and fast clock to introduce just enough delay to allow the counter time to change counting modes before changing the count itself when a button is pressed. Wrapping is possible in both directions because otherwise it is very difficult to get from one side of the display to the other in the one clock cycle between falling objects. This design choice was easy to implement because the counters automatically start back at 000 when incremented at 111.

We tested the button circuits with LEDs tied to each bit of the counter output before we hooked up the full display.

Redundancy (and What Happens if Things Break)

The way our display is designed, if one light goes out the rest can still light. Similarly, if one demux chip should fail, the information will still be carried to the next row successfully, so neither individual LEDs nor broken demuxes could render the game completely dysfunctional.

Every wire that is not flat on the breadboard is unrelated to the actual logic of the game. So, if a wire should pop out, the logic will continue as it should and only the display or the controls would be temporarily altered.

We added the redundancy of the buzzer and the death counter to make it very clear when the user has failed to catch an object, and those devices are independent and would continue to function if the other failed.

Inventory

Part	**Quantity**	**Purpose**
3:8 Demux M74LS138N	9	1 tied to each of 8 rows of LEDs,
Quad-Dual NOR LS02	1	Inverter and scoring logic
Analog Demux 4051	1	Select a frequency for the game clock
555 Timer NE555N	4	Game clock, fast clock, one-shot, and buzzer
D Flip Flop SN74LS74AN	1	Delay for button circuit
Up/Down Counter 74LS191	1	Shifting the platform left and right (bottom row of LEDs)
Decade Counter 4026	3	1 for each digit of the score, 1 for lives (or deaths)
7-Segment display	3	Display the output from each decade counter
Octal Flip Flops SN74LS374N	2	Carry the 3 bit address to demuxes
Red LEDs	64	8x8 Display
Green LEDs	8	Platform row of display
Monetary push buttons	2	Left/Right controls
Counter LS93	2	Counter for speed selection and random counter
Comparator LS85	1	Check if the platform "catches" an object or misses it
Standard breadboards	4	General Circuit
Strip Board	1	Display the output from each decade counter
Resistors used: 47Ω, 200Ω, 470Ω, 2.2KΩ, 3.3KΩ , 4.7KΩ , 5.1KΩ , 8.3KΩ, 100KΩ		
Capacitors used: 100 µF, 1µF		

Board Images

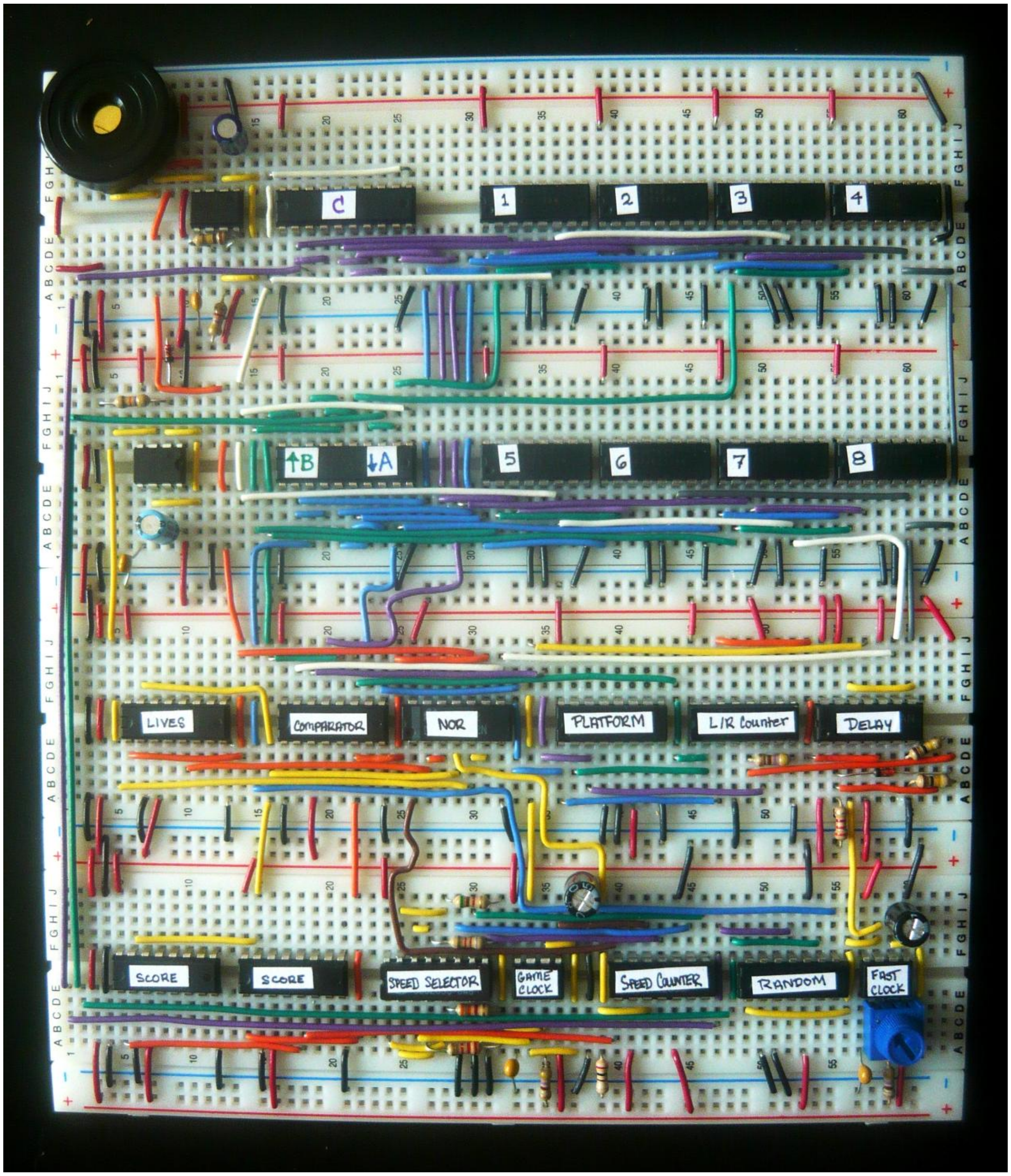


Figure 1. The full board with all chips and logic without display

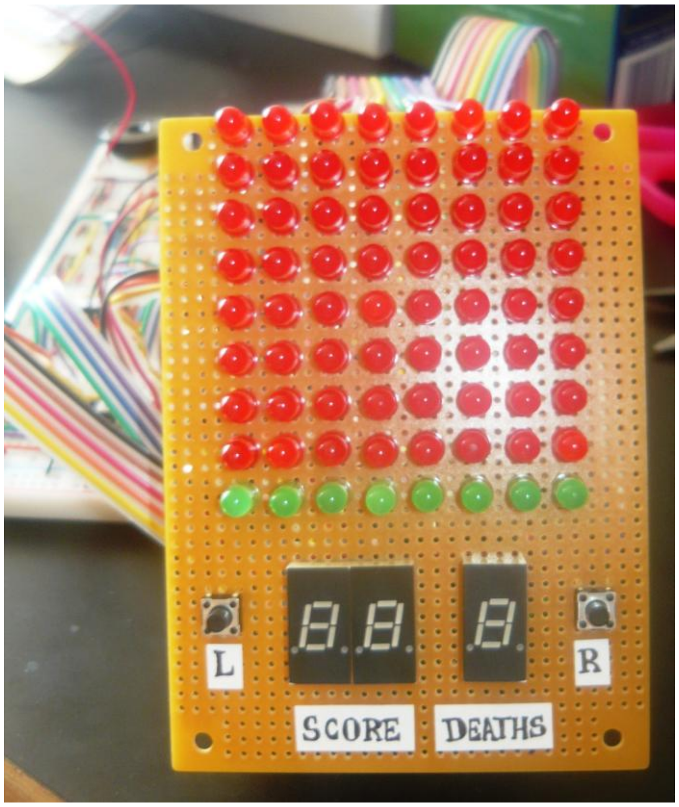


Figure 2. The display connected to the board

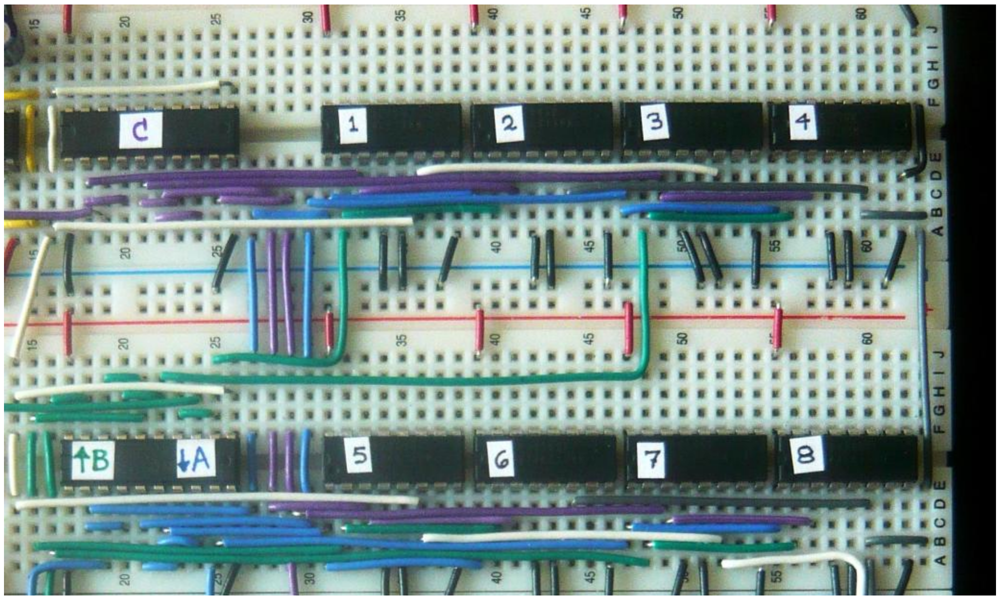


Figure 3. The “Falling Objects” sub circuit. The wires from the LEDs on the display connect to the output pins along the tops of the 8 numbered demuxes.

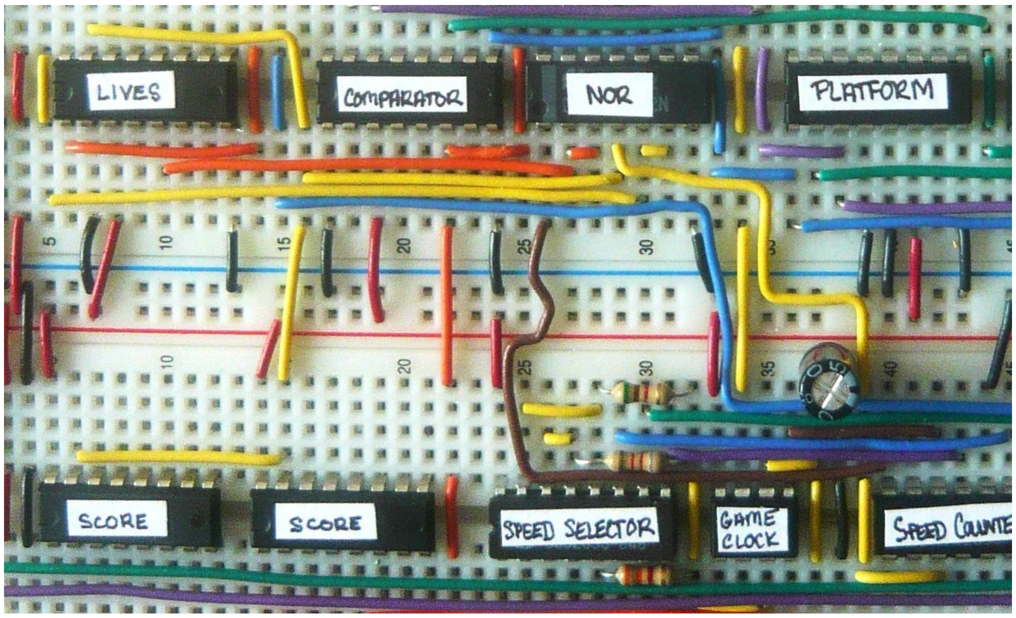


Figure 4. The two score counters, the death counter, comparator, NOR chip and platform demux form the scoring sub circuit.

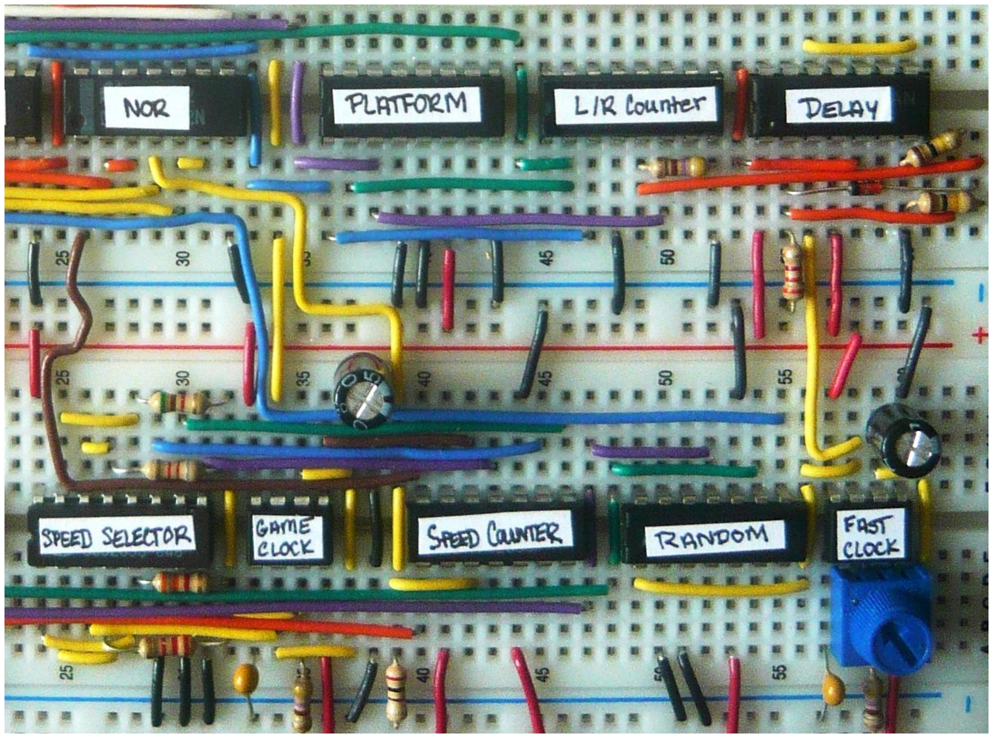


Figure 5. The chips labeled delay, L/R counter, and fast clock form the button sub circuit. The buttons connect to the resistors in the top right corner. The game clock, speed counter and speed selector are the counter and demux that form the sub circuit that controls the rate of the game.

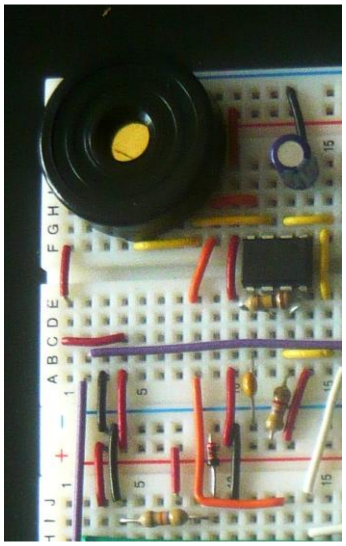


Figure 6. The buzzer sub circuit includes the buzzer and the 555 to determine its pitch.

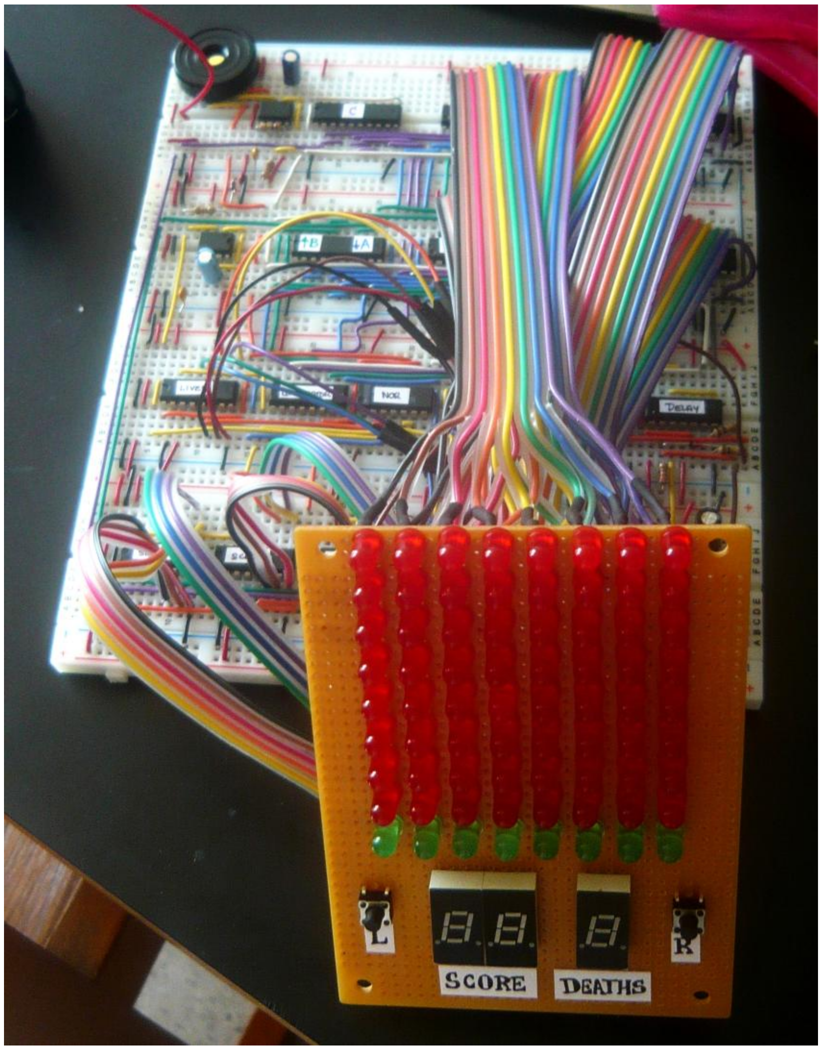
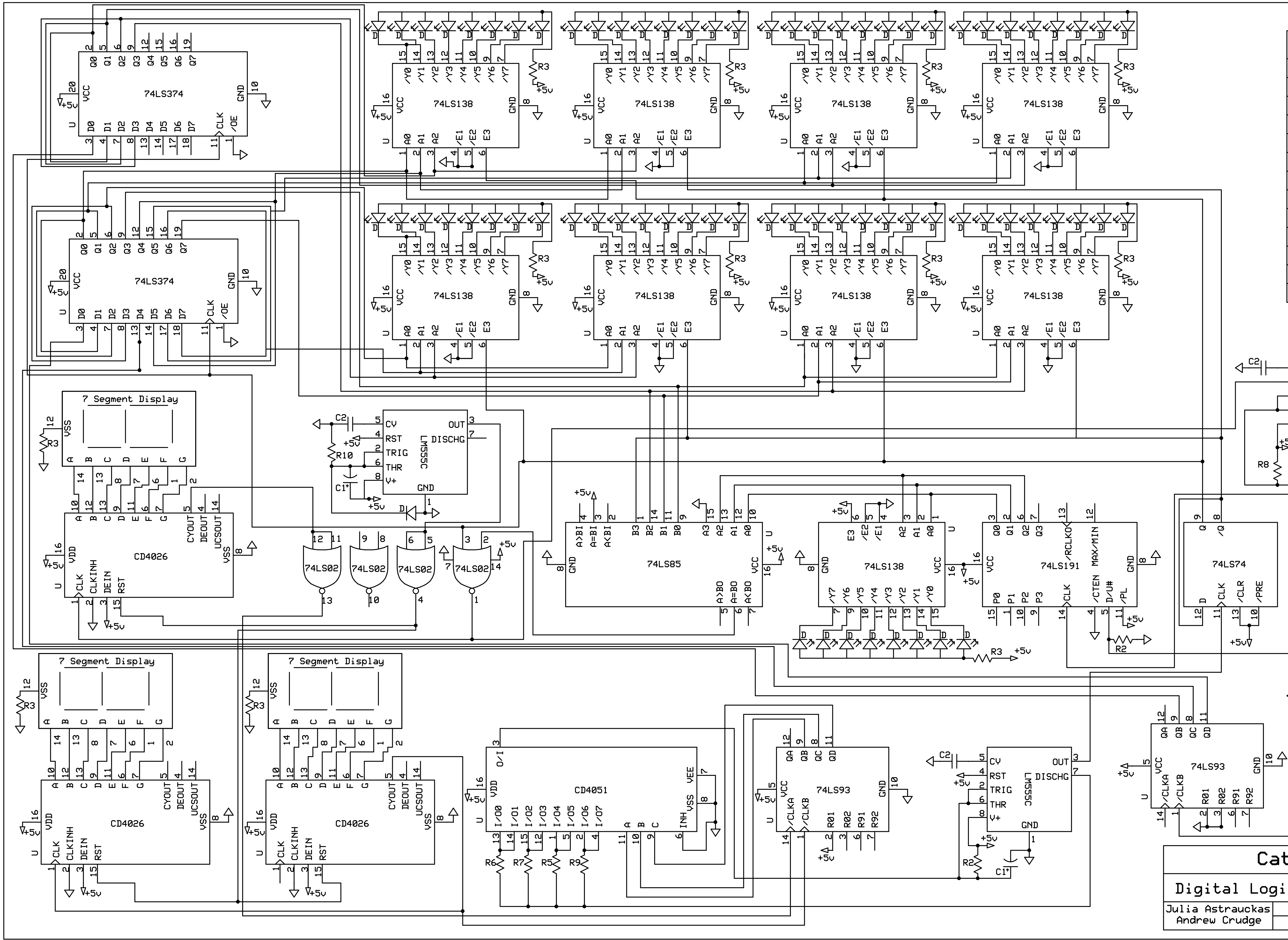


Figure 7. The full project with the display and board connected



Components Used	
R1	100K Ohms
R2	470 Ohms
R3	200 Ohms
R4	10K Potentiometer
R5	3.3K Ohms
R6	5.1K Ohms
R7	8.3K Ohms
R8	4.7K Ohms
R9	2.2K Ohms
R10	47 Ohms
C1	100 uFarad
C2	.1 uFarad
C3	1 uFarad

Catch or Die

Digital Logic Design - Spring '10

Julia Astrauckas
Andrew Crudge

5/2/2010

Problems Encountered

Some of the chips we used have multiple control pins or other input pins we did not want to use. We ignored them at first and didn't tie them to anything. When we tried to test the circuit, some signals that should have been constant or synchronized with a clock were just fluctuating or oscillating in ways we couldn't explain. We corrected this problem by tying every input pin in question to power or ground appropriately. It seemed that without doing so, the chips we used did not assume 0 or 1 as an input, but rather oscillated between assuming the two, and so they behaved inconsistently.

We wanted the counter which was supposed to keep track of the speed stage in the game to initialize to zero, but it usually started out in the middle. After looking more closely at the datasheet, we noticed that we could use a one-shot clock to clear the counter to 0 on startup.

The first octal flip flops we were planning to use were 74 series LS373. We could not get any kind of predictable sequence out of them. All of the datasheets we found for the LS373 were shared with the LS374. It turned out that we missed the line in the datasheet where it mentioned that the 373 is positive-triggered rather than edge-triggered like the 374. We switched to the 374, which behaved the way we had been expecting.

During testing, the test LEDs plugged into the breadboard diverted current from the chips and sometimes changed the behavior of the circuit we were trying to test. This problem was ultimately avoided by connecting the whole display for testing purposes.

At some points, we had a signal going from low to high when we wanted it to trigger a falling-edge J/K flip flop. To correct this problem, we used the remaining gates on the NOR chip to invert these signals (by NORing them with themselves) before sending them to the clock pin of the flip flops.

In an attempt to minimize the number of chips on our boards, we used the same fast clock for the random number generating counter and the button circuit buffer. When calibrating this clock to achieve more randomness, we slowed it down too much so that the button only incremented the counter once if pressed several times in a row. To fix this, we found a faster frequency that still provided satisfactorily arbitrary numbers.

Both buttons had to be connected at the same node as the trigger for the counting pin of the up/down counter, but only one of the buttons was intended to be connected to the up/down direction pin. Since both buttons were connected at the same node, pressing either button triggered the direction pin from low to high, so we could still only count in one direction. We added a diode to the circuit to prevent current from flowing from that common node back into the direction pin on the counter. We could have fixed this problem using an OR gate but preferred this solution because it did not require an extra chip.

Conclusion

For the most part, the construction of this project went as expected with some added complications from the problems mentioned as well as normal wiring mistakes and misreading datasheets. We sketched out a very rough schematic before we started building the circuits, and we did not have to make any drastic changes from our initial plan. This project allowed us to apply the knowledge we obtained in DLD this semester to create a physical, working game.