MUSIC RESPONSIVE LIGHT SYSTEM

By

Andrew John Groesch

Final Report for ECE 445, Senior Design, Spring 2013

TA: Lydia Majure

1 May 2013

Project 49

Abstract

The system takes in a musical signal as an acoustic pressure from the environment and performs analyses to dictate behavior such that LEDs respond in concert with the perceived tempo of the music. The input is comprised of an electret microphone that is amplified such that full range of the microcontroller's analog to digital converter input capabilities are utilized. The microcontroller then performs a beat detection algorithm on the data to determine relative energy peaks in the frequency bands. This information is then passed through the microcontroller's digital output pins to an LED driver shield that controls the behavior of three separate LED strands.

Contents

1. Introduction

Music production is a passion of mine that was first engendered from the teachings of Electronic Music Synthesis (ECE 402). I've found that when I've played some personal productions of mine, however, it's been difficult to differentiate my performance from others. Furthermore, I've found that I inject a great deal of personal touch and emotion into each track, and want each viewer to feel a connection on a subliminal level between the music that they hear and the emotions that naturally arise within them. The aim of this project is to provide a visual component to perceived musical beats.

1.1 Functionality

The main purpose of the project is to provide a reactive visual component to perceived musical beats. The main functions of the project are as follows: microphone input unit receives input signal, amplifies it, adds a DC offset, and separates the signal into frequency bands; the microcontroller processes the data based on the beat detection algorithm; the LED driver provides the correct current gain for the LEDs and passes the digital PWM output pin data to the LEDs; the LEDs pulsate between low and high logic depending on this data stream.

1.2 Top Level Block Diagram

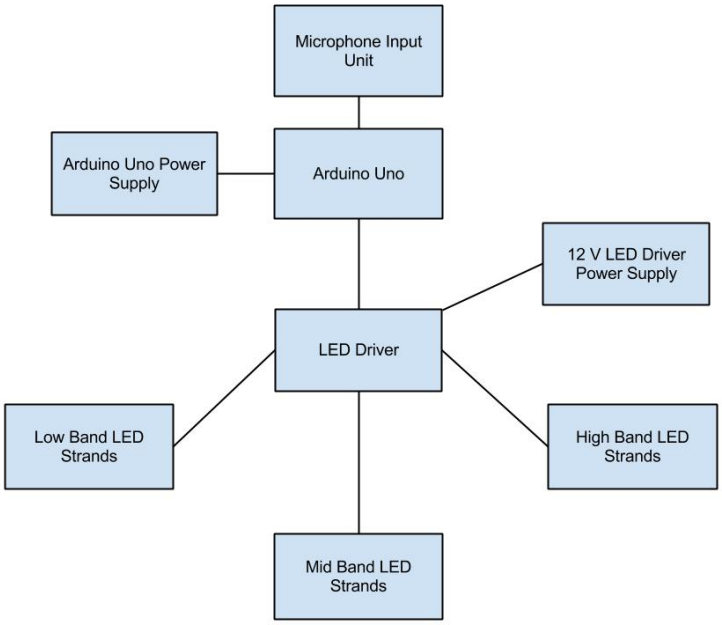

Figure 1 Top Level Block Diagram

2 Design

This is implemented by first detecting the signal with an electret microphone. This signal is then amplified with an operational amplifier gain circuit. After this, a DC offset is added such that the output signal is within the proper range for the microcontroller's analog input pins to register the signal. The signal is then split into frequency bands using low pass, band pass, and high pass filters. The outputs of these three filters are then sent into the microcontroller's analog input pins. The microcontroller then implements the beat detection algorithm using the data. The beat detection algorithm compares the relative instantaneous energy of the signal with the relative average energy. The variance between these two values is then calculated, from which a comparison constant is found. A beat is detected if the instantaneous energy is greater than the average energy times the comparison constant. This data is then passed to the LED Driver, which controls when the behavior of the LEDs.

2.1 Design Alternatives

I've considered using digital filters and a direct stereo line input to change how the project receives its input. These will be included in future iterations to compare which work best. Furthermore, I developed an LED driver circuit to compare with the LED Driver shield. The microcontroller and LED strand choices are both relatively interchangeable with other pieces of hardware; that being said, these components received favorable reviews regarding their performance before they were ordered.

2.2 Microphone Input Unit Design

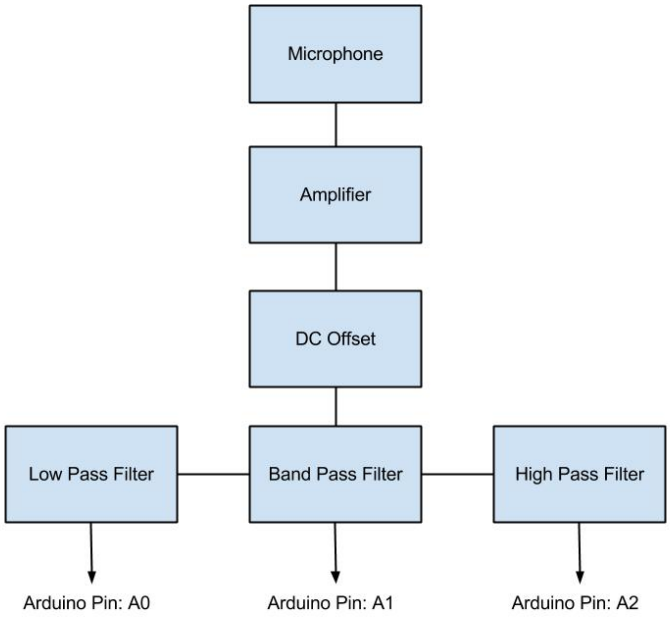

Figure 2 Microphone Input Unit to Microcontroller Interface

2.2.1 Microphone

This is used to detect an acoustic pressure and convert it into an analog signal. This is done by taking the air pressure fluctuations which affect the motion of the conductor inside the microphone. This motion creates an electric field which can be sent out as an electrical signal. In this particular case, the behavior of the electric signal is evident: because the music is a summation of many different oscillating frequency components, which summate to a sinusoid.

2.2.2 Amplifier

This module was built using operational amplifiers and resistors to build a circuit that provided a theoretical voltage gain of one hundred. This, in concert with the DC offset, was built such that the data was in the zero to five volt range; this would enable the proper functionality of the microcontroller's analog input pin.

2.2.2 DC Offset

This entity adds a DC voltage offset to the signal such that the bottom bound of the signal is above zero volts. This was intended to utilize the functionality of the microcontroller's analog input pin.

2.2.3 Low Pass Filter

This circuit permits frequencies below 160 Hz to be passed. Frequencies above this threshold are attenuated. This particular bound was chosen to include the sounds in the bass band; this subjectively is associated with instruments such as the kick drum, toms, bass, and tuba. This circuit was accomplished using an operational amplifier, a resistor, and a capacitor.

2.2.4 Band Pass Filter

This circuit permits frequencies that are both above 160 Hz and below 1000 Hz to be passed. Frequencies outside this range are attenuated. These bounds were chosen to include the sounds in the mid band. This circuit was built using consecutive filters: a low pass filter and a high pass filter.

2.2.5 High Pass Filter

This circuit permits frequencies that are above 1000 Hz to be passed; all frequencies below are attenuated. This allows an LED strand to follow the behavior of the frequencies in the high band of the frequency spectrum. This circuit was constructed using an operational amplifier, a resistor, and a capacitor.

2.3 Arduino Uno Microcontroller

The microcontroller is powered by the Arduino Uno Power Supply. It accepts an analog audio input from the microphone circuit, and detects voltage input ranges from 0 to 5 Volts. The microcontroller first separates the signal into three different frequency ranges by the way of a low pass, band pass, and high pass filter. These three signals are then analyzed by the beat detection algorithm on the data. The output data is then dispatched to the 4 Channel High Power LED Driver.

2.4 LED Driver Shield

This shield accepts any voltage range from 8 to 27 Volts. A 12 Volt power supply was chosen to power the Driver Shield. This value was chosen because the amount of LEDs is related to the applied voltage;

thus, as the applied voltage increases, more LEDs can be powered by the Driver Shield. Each shield will have four LED strands connected to its output terminals.

2.5 LED Strands

2.5.1 Low Band LED Strand

This LED strand receives its input from the LED Driver and strobes the LEDs based on the logic sent from the LED Driver. This LED strand is intended to follow the behavior of the low pass filter.

2.5.2 Mid Band LED Strand

This LED strand receives its input from the LED Driver and strobes the LEDs based on the logic sent from the LED Driver. This LED strand is intended to follow the behavior of the band pass filter (the mid band frequencies).

2.5.3 High Band LED Strand

This LED strand receives its input from the LED Driver and strobes the LEDs based on the logic sent from the LED Driver. This LED strand is intended to follow the behavior of the high pass filter.

2.6 Power Supplies

2.6.1 Arduino Uno Power Supply

The power supply delivers 12 volts with 2.0 amps of current to the Arduino Microcontroller.

2.6.2 12V LED Driver Power Supply

This device was chosen to power the LED Driver Shield. The power supply delivers 12 volts with 2.0 amps of current.

2.7 Beat Detection Algorithm

From Frederic Patin, "Beat Detection Algorithms"

Description	Input(s)	Equation	Output
Fast Fourier Transform	a(t)	$A(f) \Leftrightarrow a(t)$	A(f)
Relative Energy	A(f)	$B(f) = A(f)^2$	B(f)
Instantaneous Energy	B(f)	$E[i] = \sum(B(f))$	E[i]
Average Energy	E[i]	$\langle E \rangle = \sum(E[i])$	<E>
Variance	E[i], <E>	$V(E) = \sum[(E[i] - \langle E \rangle)^2]$	V(E)
Comparison Value	V(E)	C = (-.00257* V(E)+1.51	C
Beat Detected	E[i], <E>, C	E[i] > C*<E>	B = 1
Beat Not Detected	E[i], <E>, C	E[i] < C*<E>	B = 0

Figure 3 Beat Detection Algorithm

3. Design Verification

The main functional testing involved ensuring that each block functioned individually to assist in top level debugging. Once a given module is guaranteed to function properly, general continuity between blocks was established to ensure that the data was transferred properly and was unaltered. When these basic checks did not pass, debugging ensued.

3.1 Testing Procedures

3.1.1 Microphone Input Unit

For all analog circuitry, a function generator was used to provide a test signal. The characteristics of the test signal were used to mimic what each input should be from the previous module. Generally speaking, this included displaying a sinusoid with varying amplitudes and frequencies depending on the circuit. The output was then measured with a volt-meter to determine what action was taken in each module.

3.1.2 Arduino Uno Microcontroller

To ensure that the microcontroller functioned properly, the basic program “Blink” that’s provided in the Arduino library was used to prove the microcontroller’s functionality. The microcontroller was then loaded with the compiled DMX512 code representing the beat detection algorithm. The beat detection algorithm was tested separate from the microcontroller functionality in order to separate hardware and software components for debugging purposes.

3.1.3 Power Supplies

The power supplies were plugged into a breadboard in series with a resistor that was then passed to the ground terminal. When powered, a multi-meter was used to ensure that the correct voltage drop was observed.

3.1.4 LED Driver Shield

Correct orientation on the microcontroller was necessary for the driver shield to function properly. Once the shield was oriented and power properly, the blink function was adapted to use the digital output pins that the driver accepts data from. This signal was then passed to the functional LED strands. The blink function was then initiated to prove functionality of the driver shield.

3.1.5 LED Strand

The LED strands were tested similar to the power supplies in that they were attached to a breadboard, with the proper voltage and current being passed to them. This should produce a visible luminesce given the input.

3.1.6 Beat Detection Algorithm

The beat detection algorithm was tested using the “printf” function to output the current state of the variable that is being operated on. This was used after each equation that transformed the data in order to ensure that the code was correctly implemented.

4. Costs

4.1 Parts

Table 1 Parts Costs

Part	Manufacturer	Retail Cost ($)	Number (#)	Actual Cost ($)
Arduino Uno	Arduino	21.95	1	21.95
LM358 Op Amp	Texas Instruments	.95	3	2.85
Capacitor – 265 nF	Panasonic	.10	2	.20
Capacitor – 16 nF	Panasonic	.10	2	.20
Vector board	RadioShack	3.35	2	7.70
TIP122 MOSFET	Darlington	1.65	3	4.95
10 kOhm Resistor	Ohmite	.10	11	1.10
1 kOhm Resistor	Ohmite	.10	2	.20
PDMIC58 Microphone	Pyle-Pro	9.87	1	9.87
5 M LED Lighting Strip	Hitlights	13.88	2	27.76
LED Strip Connector (4)	Hitlights	7.98	1	7.98
12 V Power Supply	Triad	8.67	1	8.67
LED Driver Shield	Ethermania	68.45	1	68.45
Total				**161.88**

4.2 Labor

200 Hours * 2.5 * 20 Dollars/hour = 10,000

4.3 Total Cost

Total Cost = 10,161.88

5. Conclusion

Unfortunately, my project did not fully function such that I could complete the demo to the fullest extent. This was due to not following my schedule well enough and not having enough manpower to fully test the components all together. For the accomplishments, the system had continuity for both power and data transfer. The LEDs and their driver functioned such that the microcontroller could control their actions (strobe on/off). The main uncertainties were related to how the input circuit gave passed data to the microcontroller and if the beat detection algorithm was properly implemented.

5.1 Ethics

The IEEE Code of Ethics details guidelines upon which all members are supposed to act on. From the listed guidelines, this project makes pertinent the need to accept full responsibility for the claims that the project states it will perform. Furthermore, the data must support the design reasoning. Furthermore, circuit designs should be shared such that they can be tested, verified, and improved upon by the scientific community in the name of progress and the betterment of mankind. Any errors found must be corrected such that the intended outcome is realized. I fully understand my limitations and the extent of my knowledge when it comes to constructing circuits.

5.2 Future Work

I plan on implementing digital filters inside the Arduino to receive better input data. I've considered using the spectrum shield to combine a stereo line in with the microphone to have two similar data sets that output different data streams to provide a unique experience. Also, the Arduino Uno is limited in the amount of digital output pins it has, so I've considered using more microcontrollers to increase the number of usable LED strands.

References

[1] Patin, Frederic: *Beat Detection Algorithms*, February 2003

[2] *Low Power Dual Operational Amplifiers* (LM358 datasheet), Texas Instruments, January 2000.

[3] *8-bit AVR Microcontroller* (Arduino Uno datasheet), Atmel, October 2009.

[4] *High Power 4 Channel LED Driver Shield* (LED Driver Shield datasheet), Ethermania, 2013

Appendix A Requirement and Verification Table

Microphone, Gain, and DC Offset Table

Requirement	Verification
Microphone detects audio signal 1) Output voltage from microphone has range of -.25 V to .25V +/- 20%	Play audio signal possessing frequency of 1000 Hz. 1) Using a volt-meter, find the voltage difference at the microphone output and display on the oscilloscope.
Maximum output voltage amplitude level maintained by Gain factor of 100+/- 20% 1) Measure input voltage range [-25m V, 25 mV] +/-20% 2) Measure output voltage range [-2.5 V, 2.5 V] +/-20% 3) Gain = 100 +/- 20%	Apply 1000 Hz signal of amplitude = .25 V with function generator to Vin terminal. 1) Using a volt-meter, measure input voltage from filter. Display on oscilloscope. 2) Using a volt-meter, measure output voltage from filter. Display on oscilloscope. 3) Divide the maximum output voltage by the maximum input voltage.
DC Offset shifts signal 1) Output voltage from Offset Circuit between [0, .5] V +/-.2V	Input 1000 Hz signal with voltage amplitude = 2.5 V. 1) Measure the output with a voltmeter and display output on oscilloscope to ensure signal range is from [0, .5] V +/- .20 V

Low Pass Filter Table

Requirement	Verification
Frequencies below 160 Hz passed and frequencies above 160 attenuated	Apply a sinusoid with the function generator possessing amplitude of 2.5 V to the input. Connect the output to an oscilloscope. Sweep the filter with a frequencies from 20 Hz to 1500 Hz

High Pass Filter Table

Requirement	Verification
Frequencies above 1000 Hz passed and frequencies below 1000 attenuated	Apply a sinusoid with the function generator possessing amplitude of 2.5 V to the input. Connect the output to an oscilloscope. Sweep the filter with a frequencies from 20 Hz to 1500 Hz

Band Pass Filter Table

Requirement	Verification
Frequencies above 160 Hz and below 1000 Hz are passed, with frequencies below 160 Hz and above 1000 Hz attenuated	Apply a sinusoid with the function generator possessing amplitude of 2.5 V to the input. Connect the output to an oscilloscope. Sweep the filter with a frequencies from 20 Hz to 1500 Hz

Arduino Microcontroller Table

Requirement	Verification
Device utilizes input and output pins 1) Analog input detects range of [0, 5] V 2) Digital output pins display output from compiled code	1) Apply 500 Hz signal of amplitude = 2.5 V and DC offset 2.5 V with function generator to analog input 1. Graph input data in Arduino programming module 2) Enable “blink” test code to output to digital PWM pins D3, D9, & D10. Display output on oscilloscope

Beat Detection Algorithm

Calculates Fast Fourier Transform	Input data series of 16 data points into Fast Fourier Transform component of Algorithm. Use “printf” function to display data output
Calculates the sum of a series	Apply data series of 16 data points into Sum component of Algorithm. Calculate expected value for the sum. Use “printf” function to display data output. Find ratio of ‘Algorithm Output’ to ‘Calculated Sum’. Ratio = 1+/- 20%
Variance between Instantaneous Energy and Average Energy Calculated	Enter values for Instantaneous Energy and Average Energy. Compare output value with calculated value. Use “printf” function to display data output. Find ratio of ‘Algorithm Output’ to ‘Calculated Variance’. Ratio = 1 +/- 20%
Comparison Value Calculated	Enter value for Variance. Compare output value with calculated value. Use “printf” function to display data output. Find ratio of ‘Algorithm Output’ to ‘Calculated C”. Ratio = 1 +/- 20%
Comparison between Instantaneous Energy and Average Energy Calculated 1) Beat Detected 2) No Beat Detected	1) Enter values for Instantaneous Energy, Average Energy, and ‘C’ that indicate a beat is detected. Ensure that algorithm indicates that a beat has been detected with ‘1’ output. 2) Enter values for Instantaneous Energy, Average Energy, and ‘C’ that indicate a beat is not detected. Ensure that algorithm indicates that a beat has been detected with ‘0’ output.

Appendix A Component Schematics

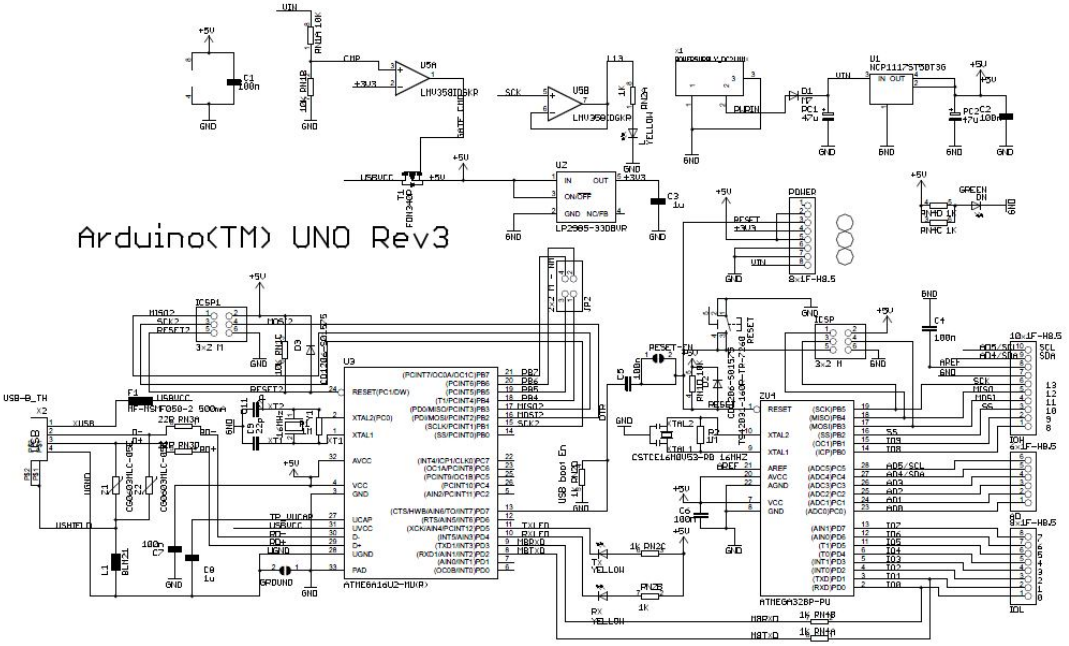

Figure 4 Arduino Uno Schematic

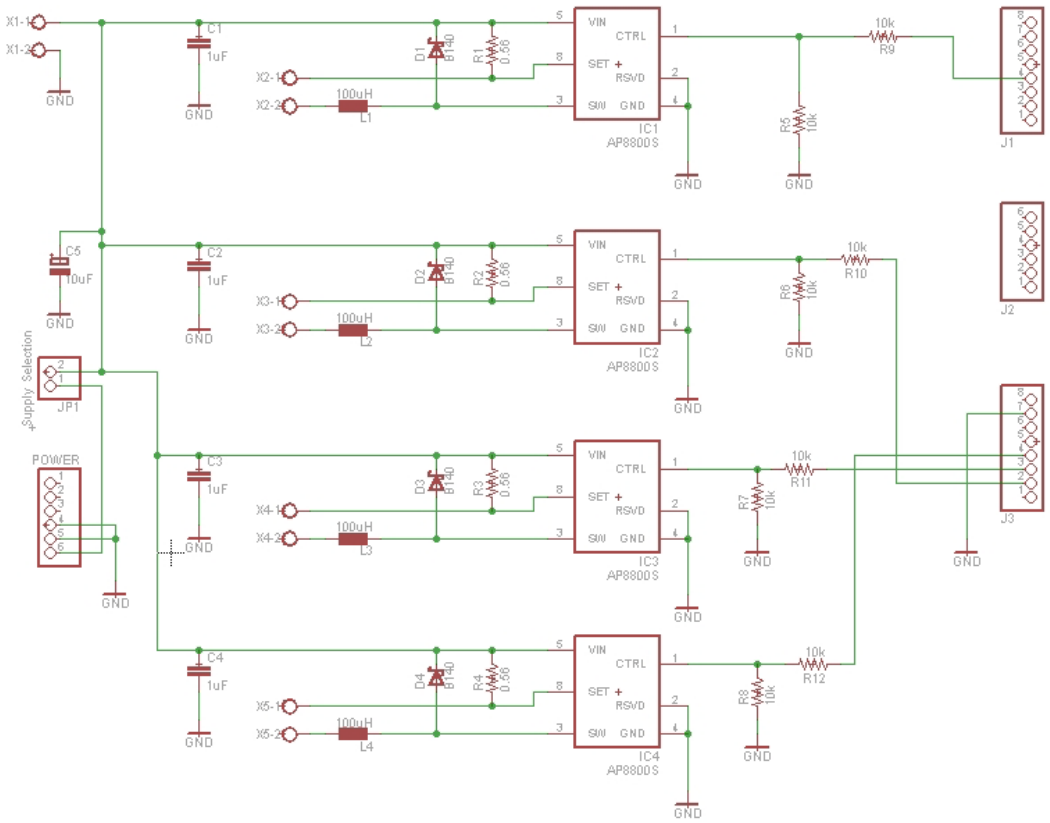

Figure 5 LED Driver Shield Diagram

Appendix C Circuit Pictures

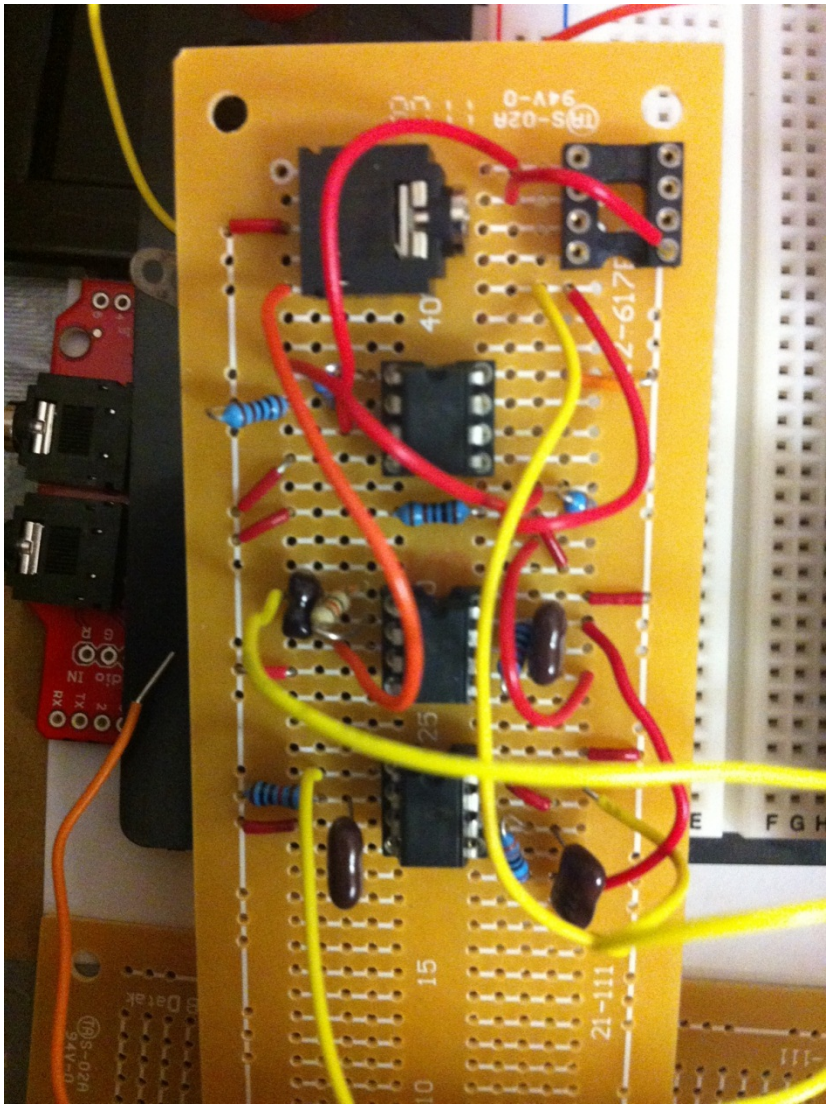

Figure 6 Data Input Circuit

Figure 7 LED Configuration