5-2018

Pyramid of Disco

Daniel Henkes
Trinity University, dhenkes@trinity.edu

Molly McCullough
Trinity University, mmccull2@trinity.edu

Pyramid of Disco

Daniel Henkes and Molly McCullough

ENGR 4367 - Spring 2018

Dr. Nickels

Pledged

TABLE OF CONTENTS

1. Design Summary

The purpose of the following project was to design, build, and test a device controlled by one or more microcontrollers. Three potential projects were first evaluated, shown by Figure A1 in Appendix A, and the Pyramid of Disco project was chosen. A high-level sketch of the project along with its functions is shown by Figure A2 in Appendix A, while the final product is shown by Figure A3 in Appendix A. The Pyramid of Disco is a 3D printed pyramid containing 221 light-emitting diodes that rotates above a fixed wooden base. The purpose of the Pyramid of Disco is to provide a light display that responds to music input by the user, lighting different LEDs in response to different frequencies in the music played through the system. The user can change the speed of rotation of the pyramid and the volume of the music manually, and the system automatically responds to changes in light level to adjust the LED brightness to best suit the environment.

2. System Details

a. Functional Diagram

An initial functional diagram for the Pyramid of Disco is shown by Figure B1 and the final functional diagram is shown by Figure B2 in Appendix B. The only difference between the initial and final functional diagram is the implementation of photoresistors in place of a rotation sensor. Once a slip ring was implemented into our design, there was no longer a need for the rotation sensor. The diagram highlights the inputs and outputs for the system, using a PIC16F88 as the microcontroller of the system. The inputs include 2 potentiometers for user control of speed and volume, two photoresistors to monitor the level of light in the environment, and 5 analog frequency filters passed through a 5-to-3 bit decoder. The outputs for the system are the LEDs that sit in the pyramid, a liquid crystal display to display the current speed and volume of the system, and a DC motor that rotates the pyramid.

b. Circuit Schematics

The interface circuits required to make the Pyramid of Disco are shown by Figures C1, C2, and C3 in Appendix C. The analog frequency filters are third order Butterworth filters and included one lowpass, one highpass, and 3 bandpass filters between 200 and 10,000 Hz. Third order filters were chosen in order to take advantage of their high roll-off rate to make the distinction between bandpasses clear. The filters are shown by Figure C1 in Appendix C. The design and calculations for the resistor and capacitor values were found using Equation 1. The lowpass filter was designed to capture the low frequencies, the high pass filter is used to capture frequencies above 10 kHz, and three bandpass filters were used to capture the frequencies in between this gap. The Circuitlab simulations for each filter are shown in Figures C5-C9 in Appendix C. The schematic was then copied into Eagle, a schematic capture and printed circuit board layout software in order to develop the printed circuit board. The final schematic is shown in Figure C10 in Appendix C, followed by the printed circuit board layout shown in Figure C11. The physical board in shown by Figure C12 along with many of the components of the first bandpass filter. Due to complications with the board, specifically detailed in the Lessons Learned section, it was not able to be used in the final product.

The 5 outputs from the figures, which were designed to be 5V when high and 0V when low were then passed through a 5-to-3 bit encoder before they were passed to the PIC. The purpose of the encoder was to minimize the inputs to the PIC. The 3 bits represented a binary number between 0 and 4, which was used to determine the row or section of LEDs to light. Due to the PIC microcontroller featuring TTL or CMOS inputs, each input requires a certain amount of voltage in order to register as a digital high. Due to the third order filters outputting less than 500 mV, a LM339 Quad Differential Comparator IC was used to

compare the input from the filter to 300mV and output a digital high or low that is unmistakable to the PIC. This connections for the comparator is shown in Figure C4.

The interface circuits to the potentiometers for speed and volume control were 50kΩ panel mount potentiometers connected between 0 and 5V. An additional circuit was needed to power the speakers, which is shown in Figure C2 in Appendix C. This circuit used the voltage from the volume control potentiometer to adjust the output of an amplifier that connected the two speakers in series. The speakers did not draw power directly from the PIC and the volume control potentiometer was read by the PIC in order to accurately update the LCD but was not read by the PIC to control the volume output of the system, making the volume control more reliable.

c. Software Flowcharts

Flowcharts for the PIC software are shown by Figures D1 and D2 in Appendix D, with the actual code in Appendix E. The final design used two PIC16F88 units for control, based on the number of pins needed versus the number of pins available for each part. One PIC was used for user interface as well as motor speed and volume control. The software flowchart for this PIC is shown in Figure D1 in Appendix D. The purpose of this code is to read the user input from the volume and speed control potentiometers, update the LCD to reflect state of both speed and volume, and to change the state of both the volume and the motor speed to reflect the new input. This program was made to continuously cycle so that any user change could be responded to instantly.

The purpose of the program on the second PIC was to read the output from the analog frequency filters that were connected to the audio input from the user. The output from the filters was passed through a 5-to-3 bit encoder before being passed to the PIC to minimize the pins needed. The program on the PIC associated the input code, a number between 0 and 7 and determined the LED row in the pyramid to be powered. The program also read and

averaged the input from the two photoresistors in order to determine the brightness of the LEDs, which was controlled through pulse width modulation. This PIC handled automatic inputs into the system, from the audio input and the photoresistors, and controlled the brightness of specific LEDs.

An advantage of separating the software between two PICs was that there was no delay caused by LED control to the process of reading user input and updating the liquid crystal display. The two systems could work simultaneously to make the overall user experience smoother.

3. Design Evaluation

a. Overall Design Performance

The overall design was successful in meeting the design goals overall. The proposed project was a rotating LED display that responded to music, which was accomplished. The project successfully incorporated both manual and automatic input and included sensors, an actuator, and user interface in the form of LEDs and a liquid crystal display.

b. Functional Elements

A requirement set by the project guidelines was that the device must contain functioning elements from six distinct categories including: output display, audio output device, manual user input, automatic sensor, actuators, mechanisms & hardware, and logic, processing, and control. In the sections below, our project will be evaluated in terms of each requirement.

i. Output Display

The output display of the Pyramid of Disco consisted of LEDs and a liquid crystal display. The LEDs were lit up according to the frequencies of the audio input, while the LCD was continually updated with the current level of speed and volume based on the analog input from the volume and speed control potentiometers.

ii. Audio Output Device

The audio output was a speaker amplified using an amplifier circuit, the gain of which was controlled with the volume control potentiometer. The speaker was not directly controlled by the PIC, but the volume level of the speaker was monitored by the PIC.

iii. Manual User Input

Manual user input for the Pyramid of Disco consisted of two potentiometers for motor speed and speaker volume control. Both potentiometers were interfaced with the PIC as analog inputs. The voltages at the PIC pins from the potentiometer were converted into digital values that were then relayed to the LCD and by extension, the user.

iv. Automatic Sensor

The automatic sensor used for the Pyramid of Disco were two photoresistors that monitored the level of light in the room. When the light was low, the photoresistors relayed that information to the PIC with an analog signal and the brightness of the output LEDs was decreased. When the light was high, the brightness was increased. This made the LED display automatically reactive to changes to the light level in the environment.

v. Actuators, Mechanisms, and Hardware

To turn the pyramid, a PWM-controlled DC motor was used. The duty cycle of the pulse width modulation was determined from the speed control potentiometer. The DC motor was connected to the same PIC as both potentiometers and the LCD. To translate the rotation of the DC motor to the pyramid, a series of 1:1 3D printed gears were used. The gears sat around a slip ring that allowed the connection of the LEDs in the pyramid to the circuit in the box without wires becoming twisted. The pyramid itself was also 3D printed with enough holes for 221 LEDs. Only 96 LEDs were used as a proof of concept for the prototype. The main circuit and user interface were contained in a wooden box.

vi. Logic, Processing, and Control

The program used in the Pyramid of Disco project contained open-loop control. The speed and volume control potentiometers that created a set point for the speed of the motor and the volume of the speakers to match. The program also used programmed logic to translate photoresistor levels from a scale between 51 and 255 to 10 to 255 to create a larger range of brightness levels from a smaller range of actual light levels in the environment. Logic was also used to determine which LED rows to light through a series of IF statements and WHILE loops. Finally, calculations were also used to translate 10 bit digitally converted analog inputs to 8 bit duty cycles in four cases and, for the photoresistors, to calculate the average analog input level between the two sensors.

c. Justification for Grading Adjustments

i. Construction quality, aesthetics, consumer appeal (+/- 10)

The quality of the Pyramid of Disco warrants a positive adjustment based on construction quality, aesthetics, and consumer appeal. The project was easy to use for the consumer with clear and simple manual input and interface. The use of the box to hold the main circuit and the use of 3D printed materials for the main pyramid and gears made the look of the project clean and professional.

ii. Level of effort (+/- 10)

The Pyramid of Disco represents a significant level of effort and therefore deserves a positive adjustment based on an apparent and meaningful level of effort. The design and printing of complex 3D parts, as well as the soldering of the LEDs inside the pyramid, the design and building of third order analog filters, the use of a 3 bit encoder, photoresistors, and several examples of pulse width modulation control represent considerable effort for this project. The successful implementation of most of the components into a functioning system is also the result of significant effort that should be rewarded with a positive adjustment.

iii. Construction cost and expected mass production cost (+/- 10)

The construction cost and expected mass production cost of the Pyramid of Disco is average to slightly frugal, possibly not warranting a positive adjustment, but not deserving of a deduction. The parts used in the project were inexpensive overall, especially if bought in bulk. The most expensive parts used were the slip ring, LCD, and possibly the DC motor, but none of these components were of significant cost. Additionally, the 3D printed materials were not expensive based on the low quality of the printing, and in a mass production scenario the price of the individual pieces would decrease significantly, especially with the CAD model already created.

iv. Integration of components and functionality (-10)

Most of the components used in the Pyramid of Disco were integrated into the system to work together to produce the desired output. The user interface LCD was updated immediately when the volume or motor speed level was changed by the user, and the LEDs responded to the audio input without interference of unnecessary pauses that stopped the user interface from being read. However, the volume control was did not adjust the actual volume of the music, despite being connected to the amplifier circuit. Each required category of component was present in the final product. Due to the level of integration of components and functionality, only small deductions would be warranted.

v. Performance during demonstration (-10)

The performance of the Pyramid of Disco during demonstration warrants only small deductions. The automatic sensor, the photoresistors, did not work during demonstration and the volume control did not affect the actual output through the speakers. However, the motor was controlled by the manual user input and the result was updated to the LCD automatically. The LEDs light up in response to musical input, and the speaker was able to play the audio

input. Almost all the components were functional and integrated with each other, but because it was not all the components in the project, small deductions would be warranted.

vi. Assembly or safety (-10)

The contained nature of the Pyramid of Disco prevents safety hazards for the user. All wiring and connections are covered and the limited speed of the motor prevents the pyramid from rotating at a dangerous speed. Additionally, the user interface is simple and has no potential hazards. Although the system does require an external power source, the power cord can simply be plugged into an outlet and does not rely on dangerous or complex wiring that is open and a hazard to the user. Because the Pyramid of Disco is well assembled so as to pose no hazard to the user, no deductions are warranted on the basis of poor assembly or safety.

vii. Multiple and/or expensive controllers (-10)

A deduction based on multiple or expensive controllers is not warranted. The controllers used in the Pyramid of Disco are PIC16F88 microcontrollers, which are inexpensive. The use of two microcontrollers is warranted based on the increased speed of each microcontroller with less programmed logic, which allows the system to function smoothly without odd delays or errors, which could occur in a long program run on one microcontroller. The appropriate use of inexpensive controllers for the Pyramid of Disco makes deductions on the basis of expensive controllers unreasonable.

viii. Inappropriate or inconvenient power source(s) (-10)

The power source used for the Pyramid of Disco used a power cord plugged into an outlet. While this makes the Pyramid of Disco less portable, it is not dangerous or necessarily inconvenient. The use of a protoboard within the box as the main power source also serves as a weight to prevent any movement of the whole system caused by the movement of the pyramid. While a protoboard is not the most inexpensive power source, the use of a single AC plug is convenient.

4. Partial Parts List

Pyramid of Disco Partial Parts List					
Category	**Part**	**Manufacturer Part Number**	**#**	**Description**	**Price for one**
Output Display	16x2 LCD Display	(provided by shop)	1	LCD display used to show the volume and speed	-
Audio Output Device	8-Ohm speaker	(provided by shop)	1	Amplifies music plugged into device	-
	Audio jack		1	Input music into circuit	$3.95
Manual User Input	50K Potentiometer	(provided by shop)	2	Control the volume and speed	-
	Potentiometer knobs	(provided by shop)	2	Covers hard to use potentiometer for a more user friendly interface	-
Automatic Sensor	Photoresistor	(provided by shop)	2	Adjust brightness of LEDS	-
Actuators, Mechanisms, and Hardware	Geared DC motor	(provided by shop)	1	Rotate pyramid	-
Logic, Processing and control.	PIC16F88	Microchip PIC16F88	2	PIC used for DC motor control, LCD display, and filter/LED processing	
Miscellaneous	Slip ring	Adafruit	1	Used to keep connection between LEDs in rotating pyramid and stationary PIC	$15.95
	3D printed pyramid	(printed in library)	1	Main LED display housing	-
	3D printed gears	(printed in library)	3	Move pyramid with DC motor	-

5. Lessons Learned

One issue faced occurred when interfacing the motor and the LCD with the same microcontroller. When the motor was connected to the microcontroller through a power mosfet, the LCD would clear and output special characters or solid blocks randomly. This problem was solved by connecting the power for both the motor and the LCD to ground through capacitors. This prevented the LCD from randomly changing when the motor was running at the same time.

Our group also faced many real-world challenges involved with converting an electrical schematic to a printed circuit board, or PCB. Coming into the project, our group had limited knowledge of printed circuit board design and printing. This project not only offered a great introduction to the topic, but a valuable lesson as well in the timeline associated with the task. Although first imagined as a method that would save time, designing the PCB required several iterations in order to get the correct specification and led to many problems associated with the connections with a DIP socket for a integrated circuit. Ultimately, we did not have the time to perform another iteration on the design and were forced to use the breadboard version. The board schematic our team designed is shown by Figures C10 – C12 in Appendix C. The design and implementation aspect can also be applied to the 3D printing that was used by our group for the LED pyramid, rotating gears, and other miscellaneous items. Although 3D printing offers many significant advantages for making unique parts, there are also disadvantages associated with printer and material quality and the time required to print. Our group was met with major setbacks while 3D printing the LED pyramid due to print material not sticking to the printing surface properly and ruining the part. However, once this was resolved, it was concluded that the 3D printed parts added tremendous quality to the project.

The last major lesson learned throughout this project involved the designing, building, and testing of the third order Butterworth filters. After first testing first-order filters, it was concluded that the roll-off after the cutoff frequency did not provide enough distinction between bandpass filters. Third order Butterworth filter was chosen, which would provide a significant distinction between filters, but caused problems due to the complexity of the design. The results from testing the filters came nowhere close to the results from CircuitLab simulation, causing multiple iterations of the design in order to come close. This lesson showed the importance of not relying on computer simulations for electronic circuits.

Appendix A

ENGR 4367 - Mechatronics
Design Project Deliverable #2 - Decision Matrix
Group G - Daniel Henkes, Molly McCullough

Does not fulfill
Does fulfill

Functional Element Categories	Design Consideration #1 Perfect-cup Coffee Brewer	Design Consideration #2 BACTLUTM (light display that responds to music)	Design Consideration #3 Robotic Hand
Output Display			
LED		Light based display	
7-segment digit display			
LCD			
display screen	Show selections/progress	Shows song slection and volume	Show position of hand
Audio Output Device			
buzzer	Notify when cup ready		Hand connected/disconnected
speaker with digitally pre-recorded music or voice		Play music	
speaker with software-generated sound effects	Notify when cup ready		
speaker with software-controlled synthesized music or voice			
any of the above with higher volume (e.g., through a transistor or amplifier circuit			
Manual User Input (interaction with user)			
switch or button	To turn on	Select song	Turn on/off
potentiometer		Control volume	
joystick	To adjust heat/caffine level		
keypad	Enter desired temp		Enter desired position
keyboard			
touch screen			
Automatic Sensor (response without user input)			
limit or proximity switch			
photo-optic pair			
potentiometer			Determine position from startup
photo cell		Measure light in room	
temperature sensor	Measure temp		
accelerometer			
encoder			
Actuators, Mechanisms & Hardware			
Actuators:			
RC servo motor		Rotate stand	
solenoid			
on-off DC motor	Mix		
reversible DC motor			
stepper motor			
PWM speed-controlled motor			
Mechanisms/Hardware:			
solid and reliable mechanical design, manufacturing, and assembly			Required for building
interesting and effective use of linkages, cams, screws, levers, gears			Required for building
appropiate and effective use of 3D-printed and machined parts			Required for building
Logic, Processing, and Control			
Open-loop control			
programmed logic		Light up LEDs based on freuqencies in music	
menu-driven software			
calculations and data storage/retrieval			
advanced and/or multiple interfaced microcontrollers			
closed-loop feedback control			Determine location from setpoint
components not included in other categories		Determine/sense frequencies in music played	
Design Evaluation			
Fulfills requirements A-F (at least one in each category)	Yes	Yes	Yes
Difficulty level (1-10)	8	8	9
Projected Cost (1-10)	7	5	10

Figure A1. Decision matrix for three design considerations with each either fulfilling or not fulfilling the requirements under each functional element category.

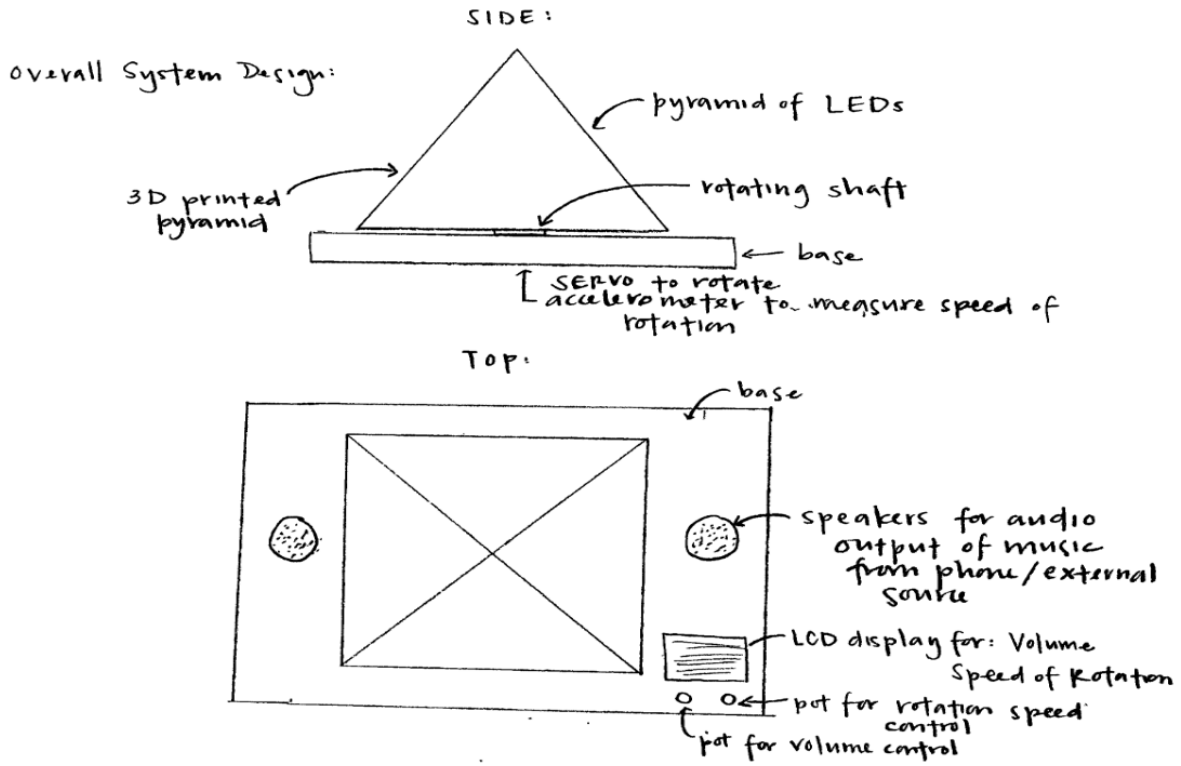

Figure A2. High-level overview of the project including all of the elements that fulfill the functional elements categories.

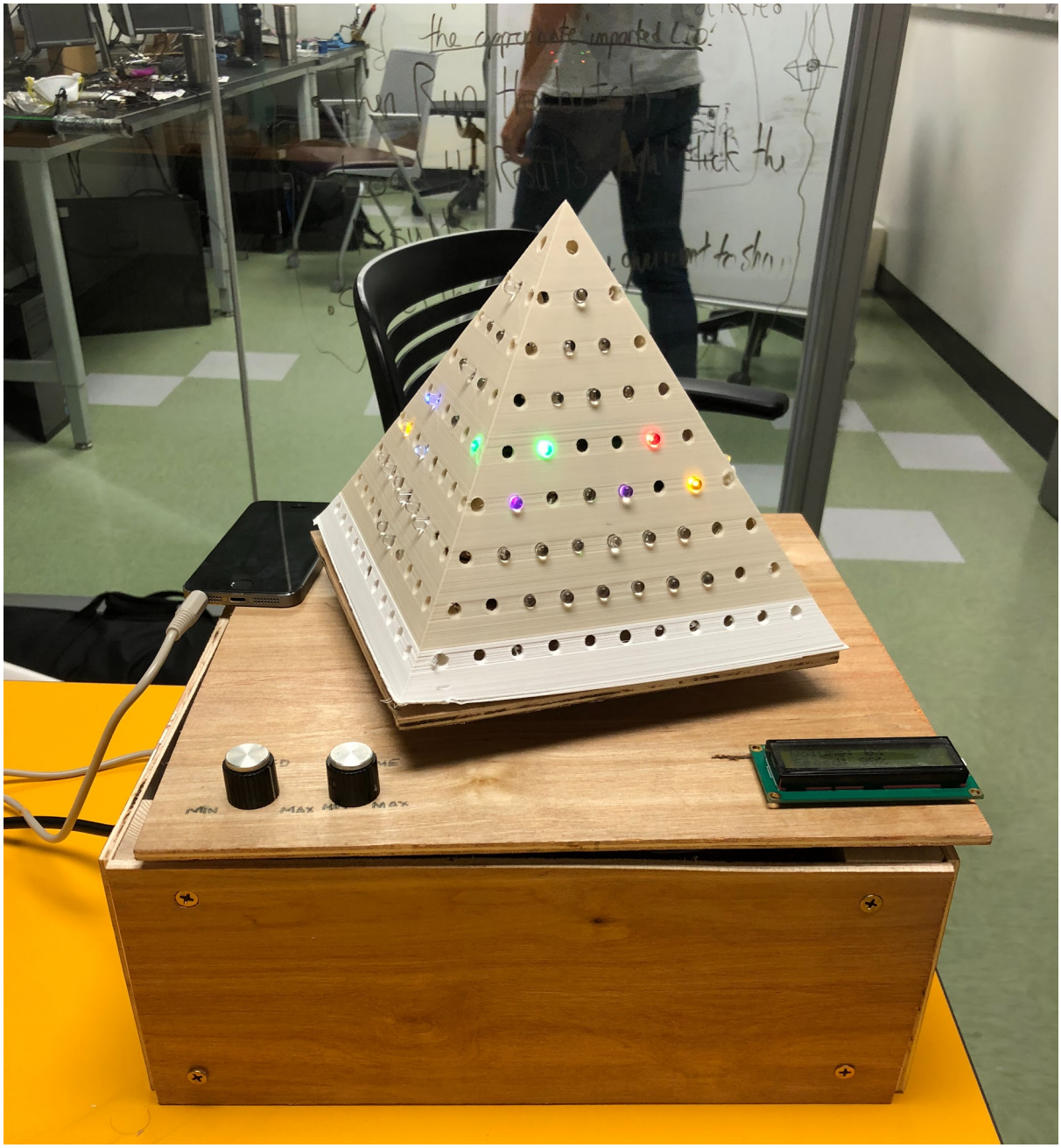

Figure A3. Final project with LCD, 2 potentiometers for manual input and audio jack for audio input. Photoresistors are located on the back of the base and are not visible in this image.

Appendix B

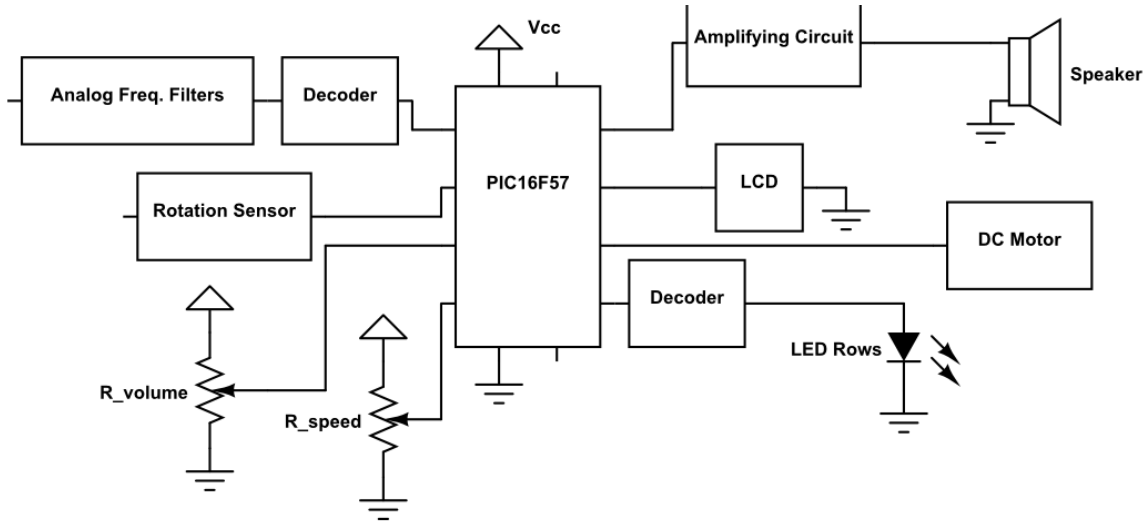

Figure B1. Initial functional diagram to PIC microcontroller for Pyramid of Disco develop using CircuitLab.

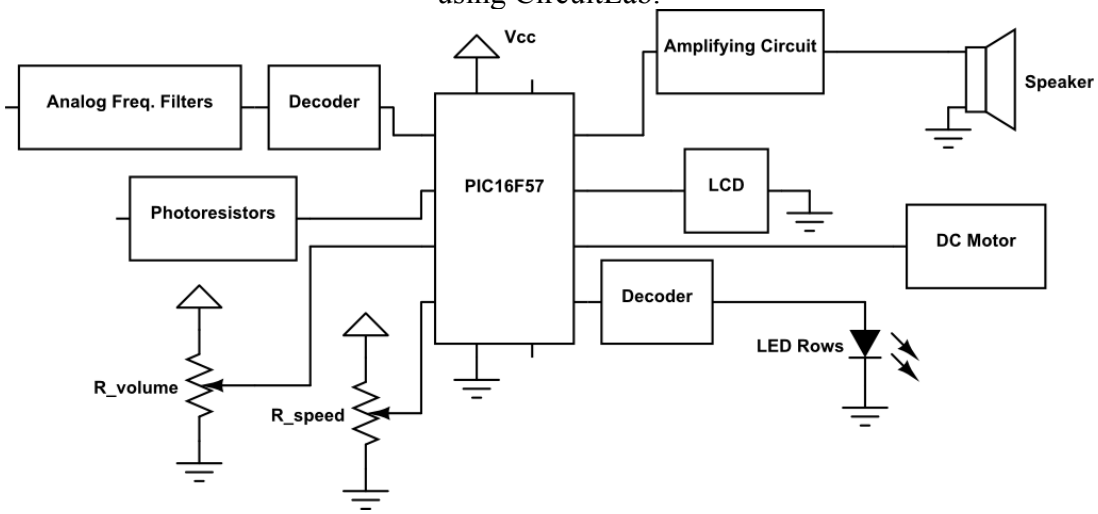

Figure B2. Final functional diagram to PIC microcontroller for Pyramid of Disco, developed using CircuitLab.

Appendix C

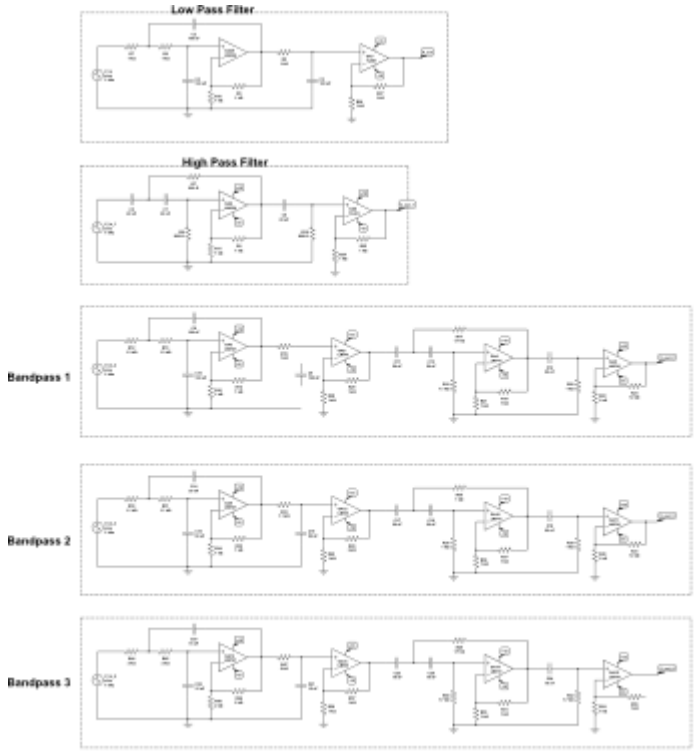

Figure C1. CircuitLab schematic of the Butterworth filters used to filter out select frequencies.

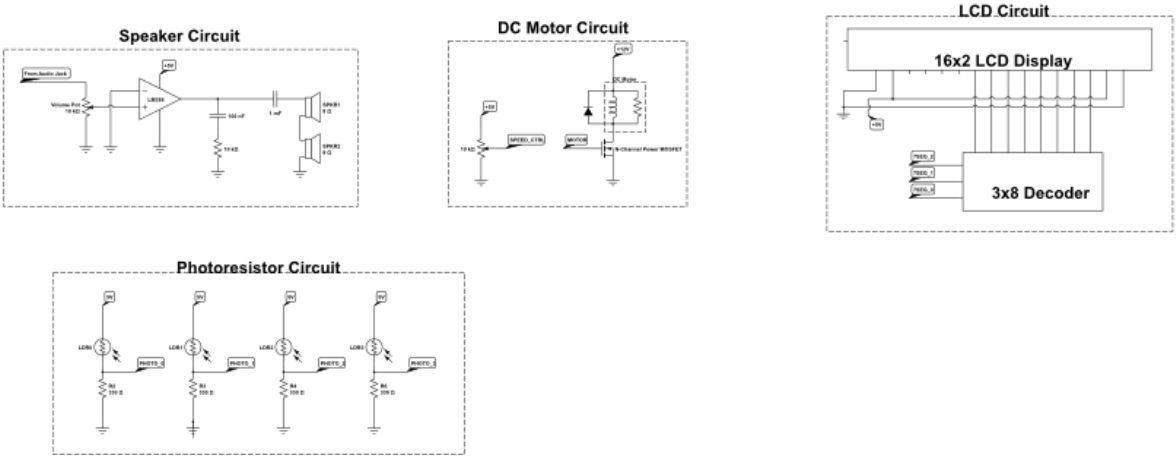

Figure C2. CircuitLab schematic of the speaker, DC motor, LCD display, and photoresistor driving circuits.

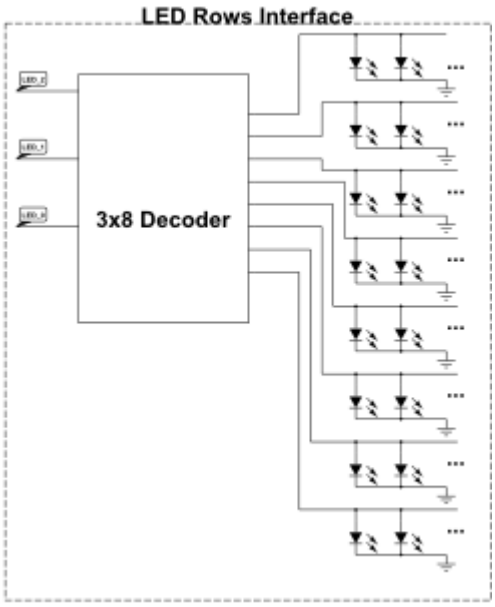

Figure C3. CircuitLab schematic of the LED row interface, located in the rotating pyramid.

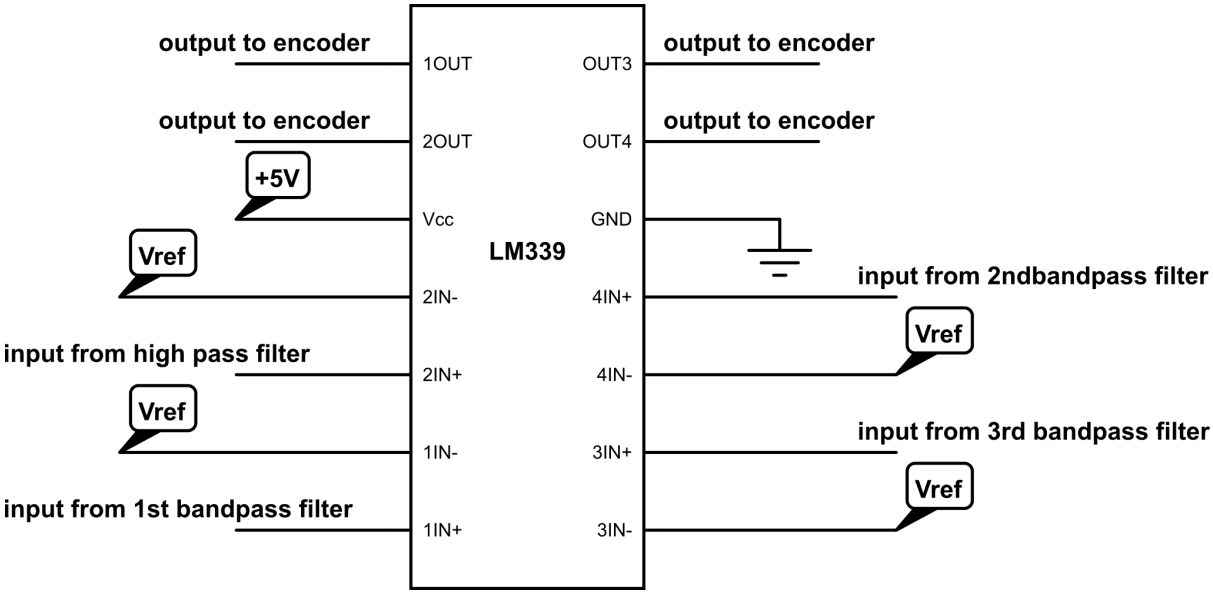

Figure C4. Pin connections for comparator circuit.

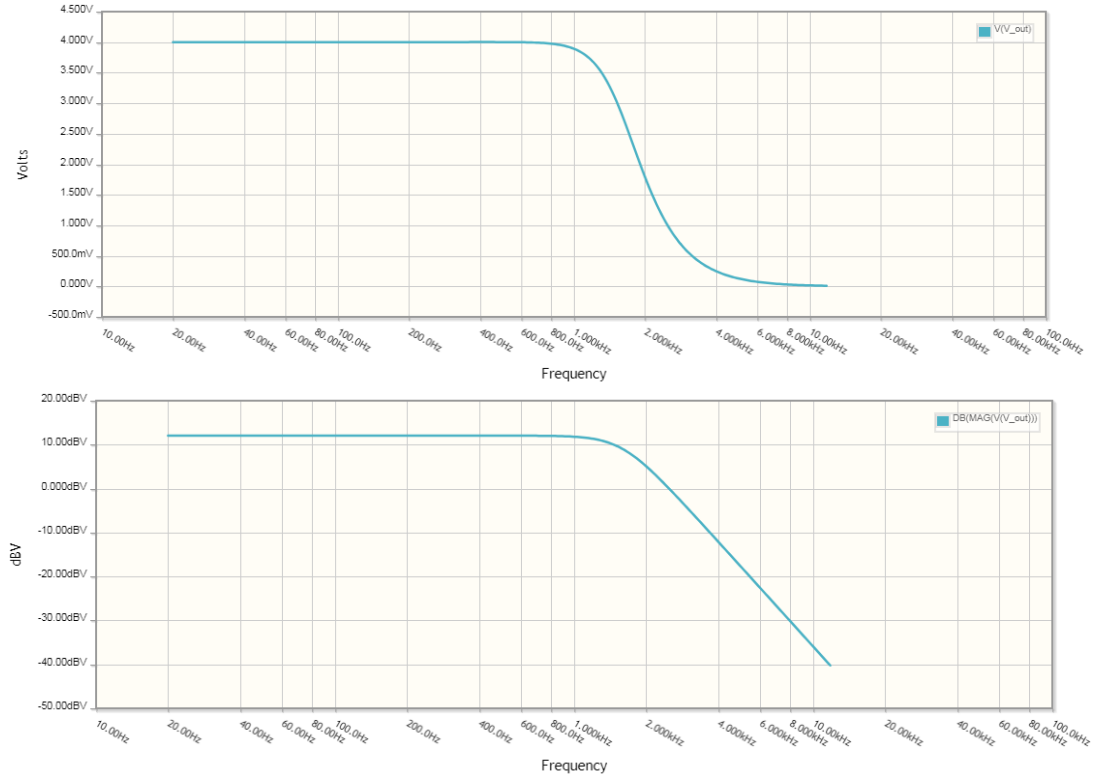

Figure C5. CircuitLab frequency-domain simulation of the third order lowpass filter.

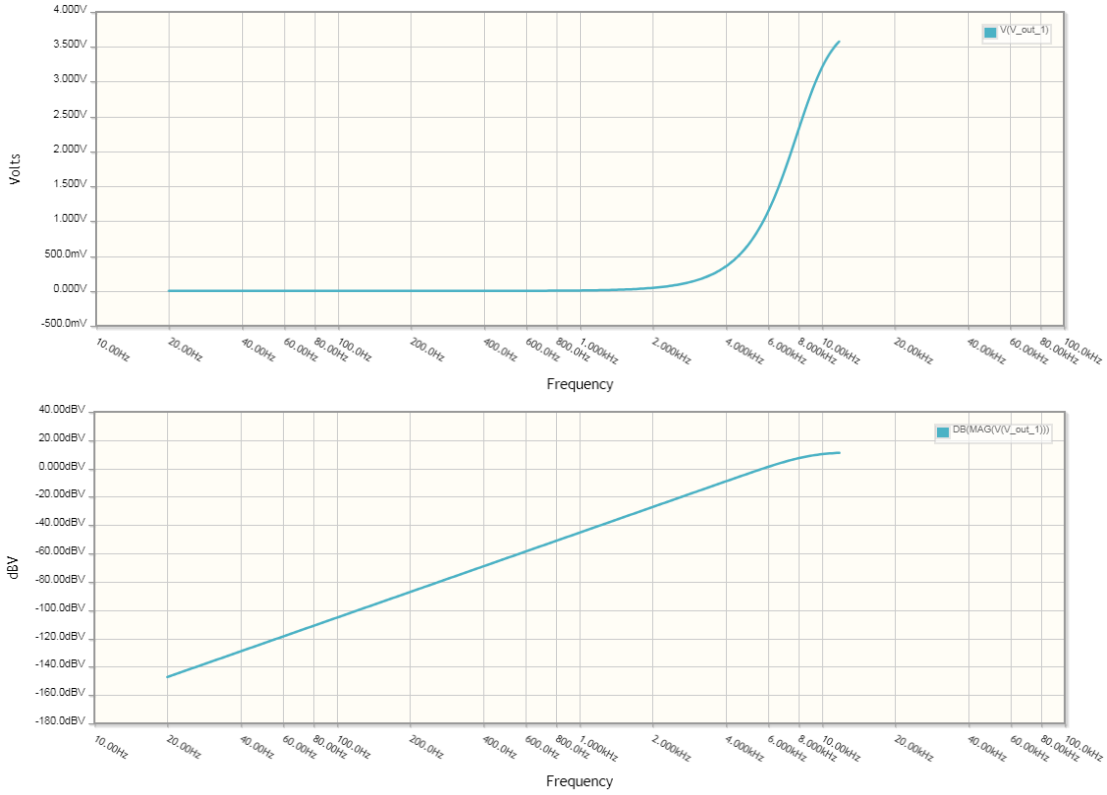

Figure C6. CircuitLab frequency-domain simulation of the third order highpass filter.

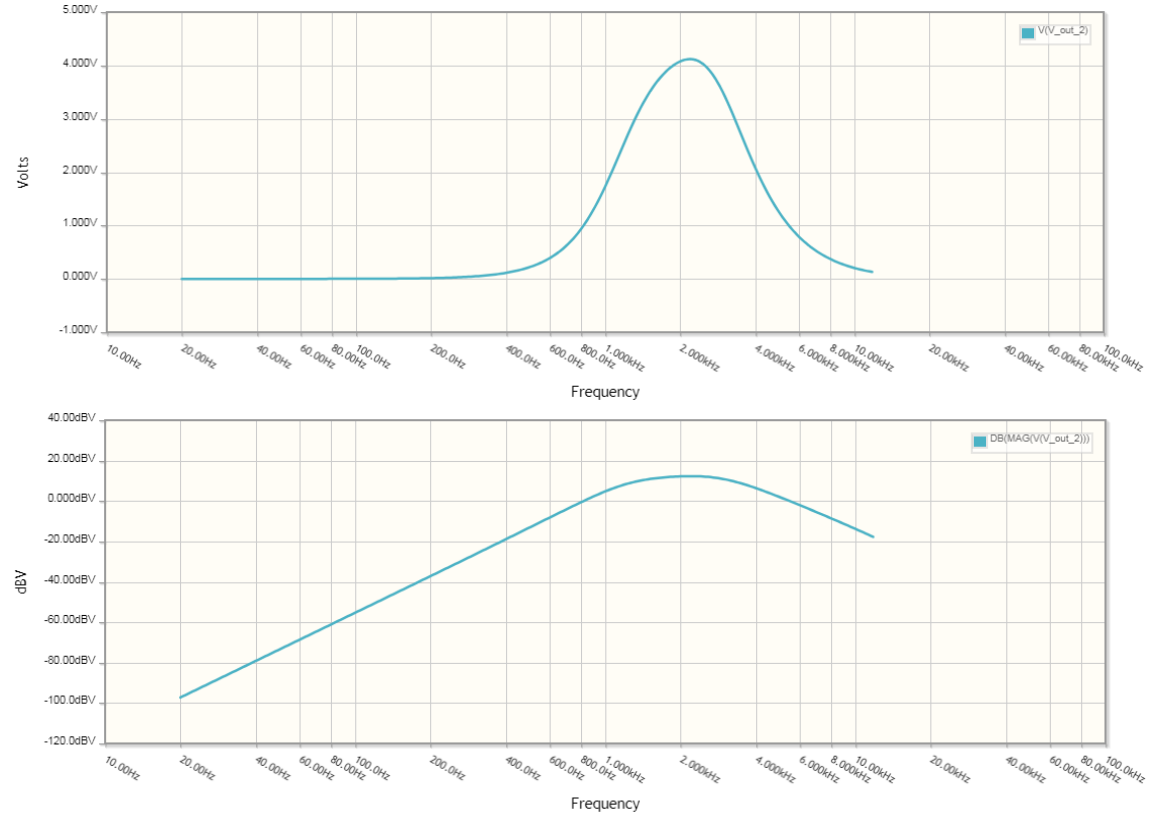

Figure C7. CircuitLab frequency-domain simulation of the first third order bandpass filter.

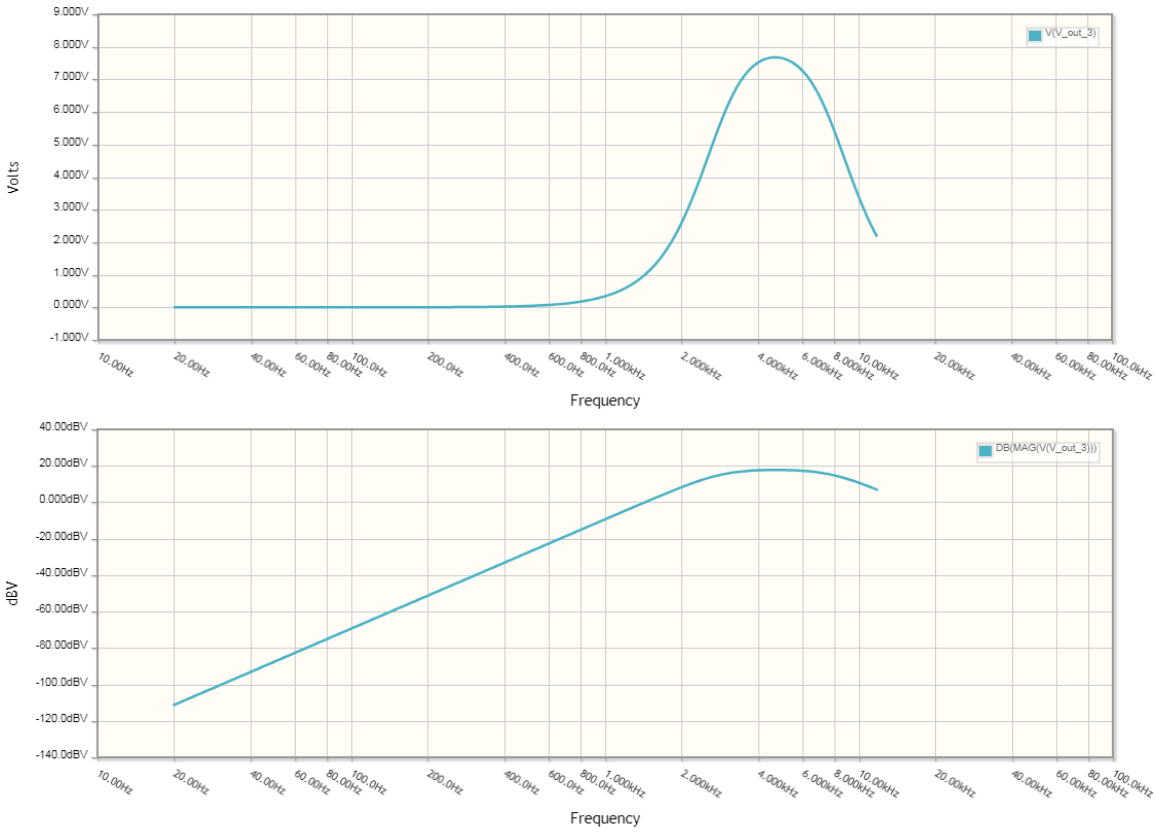

Figure C8. CircuitLab frequency-domain simulation of the second third order bandpass filter.

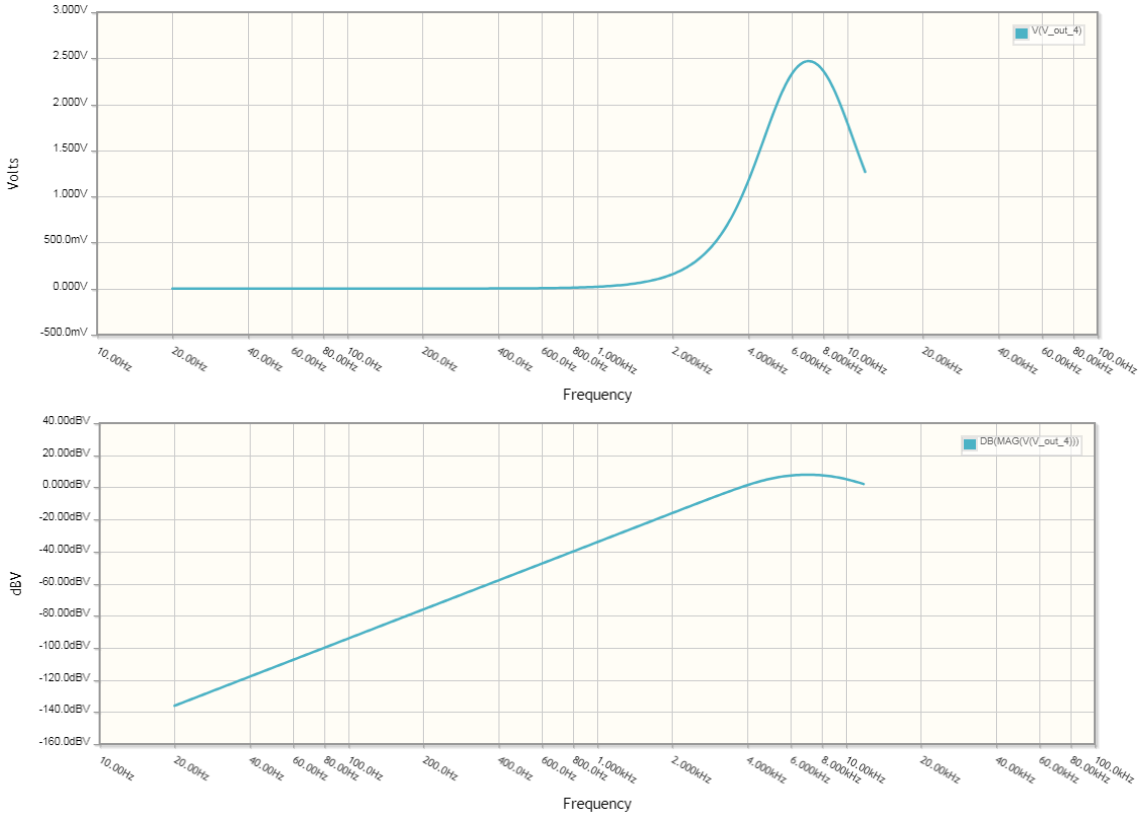

Figure C9. CircuitLab frequency-domain simulation of the third and last third order bandpass filter.

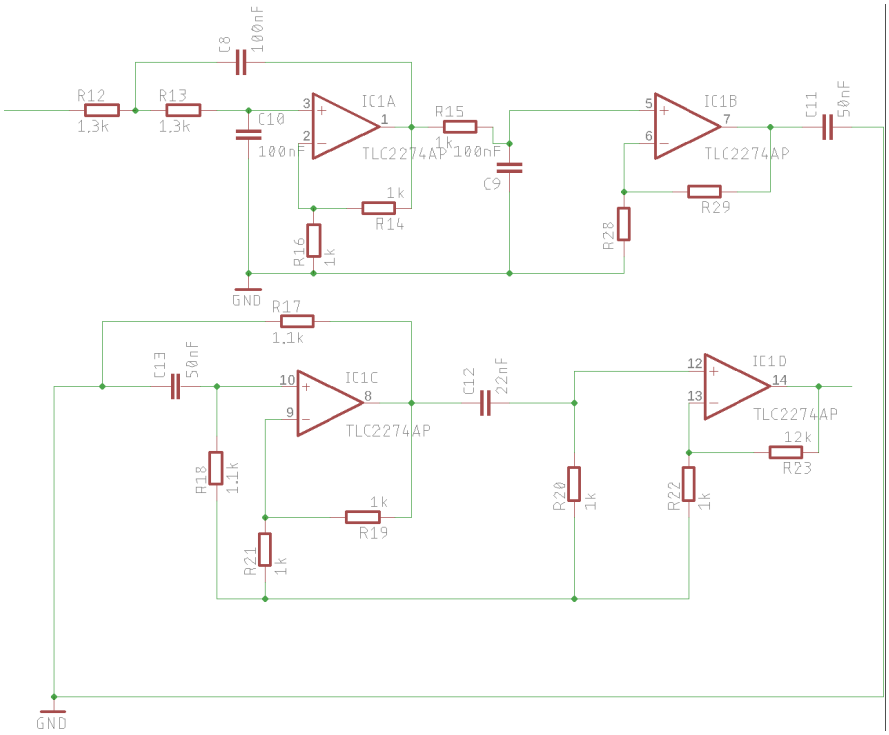

Figure C10. Final schematic generated in Eagle for general filter layout that can be used for all bandpass filters.

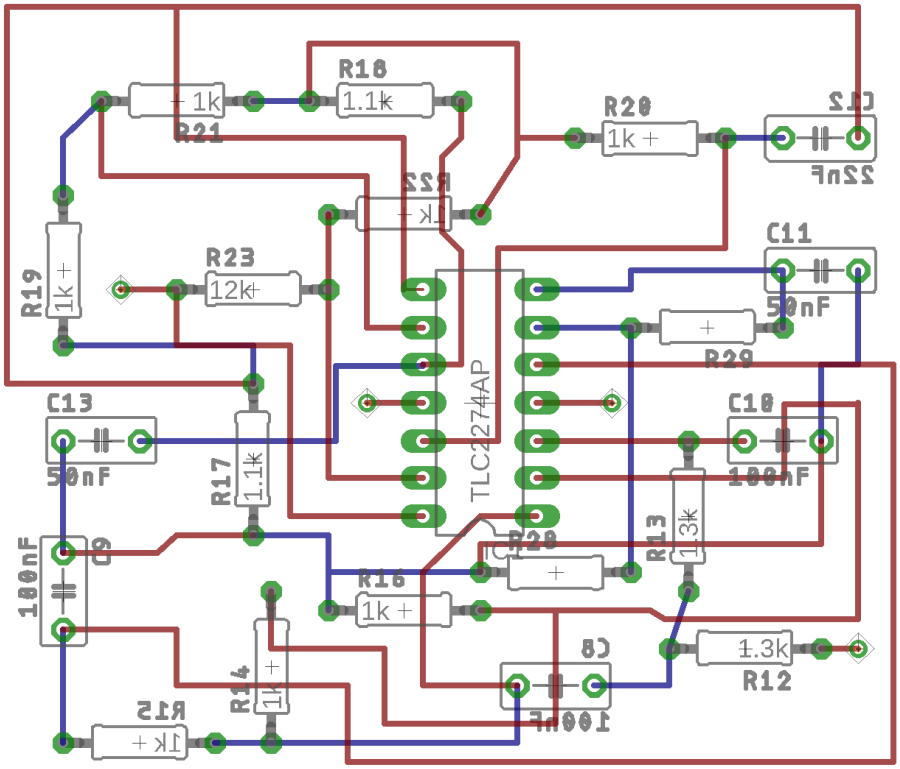

Figure C11. Eagle layout for the printed circuit board design. The red wire represents the top layer of the PCB and the blue layer represents the bottom layer. Eagle develops a footprint for each electrical components automatically through its library.

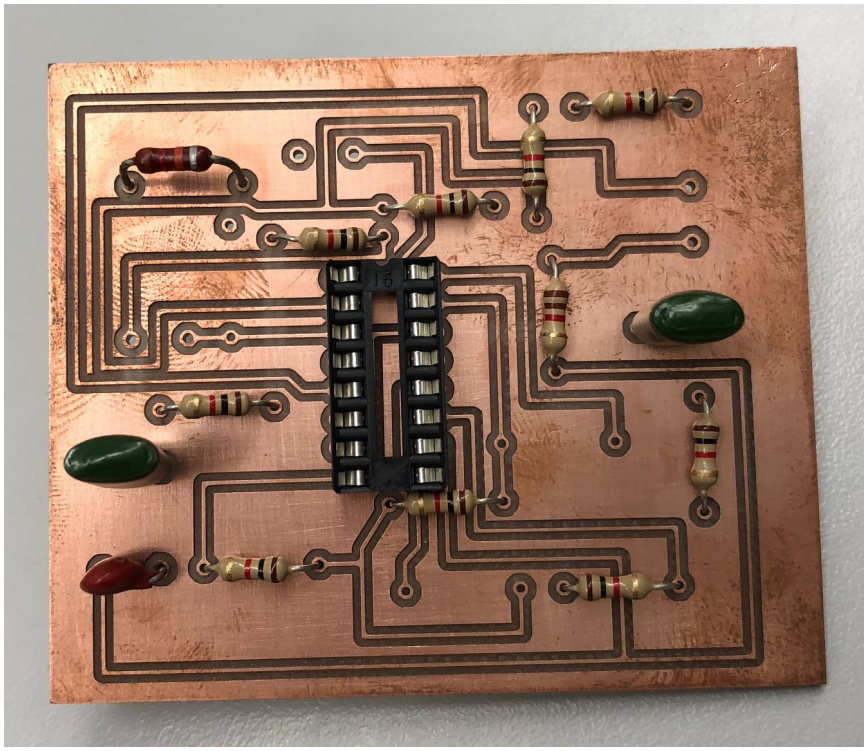

Figure C12. Physical PCB with components. The quad op-amp integrated circuit would sit in the dual inline package socket, but is not shown in the image.

Appendix D

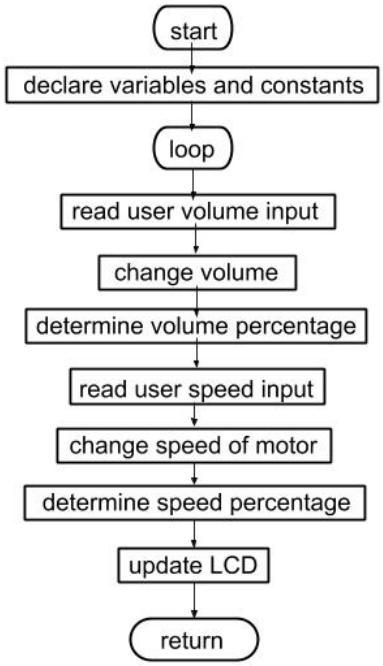

Figure D1. Program flowchart for microcontroller reading user speed and volume input, powering motor, and updating LCD display.

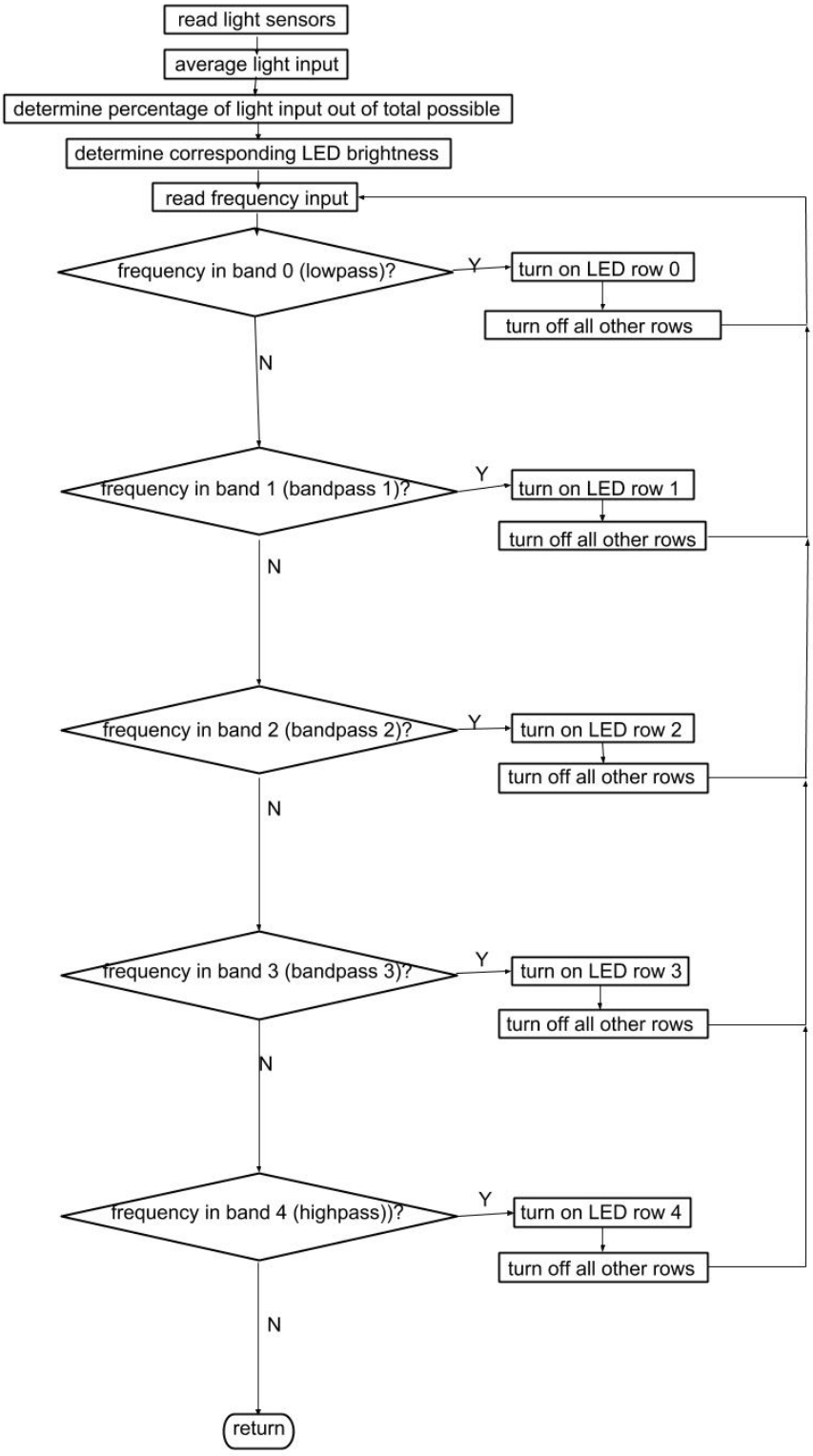

Figure D2. Program flowchart for microcontroller reading analog frequency and light sensor input and determining the which LEDs are powered at what brightness level.

Appendix E

```
'****************************************************************
'*  Name    : Speed and Volume Control                          *
'*  Author  : Molly and Daniel                                  *
'*  Notice  : Copyright (c) 2018 [select VIEW...EDITOR OPTIONS] *
'*          : All Rights Reserved                               *
'*  Date    : 4/22/2018                                         *
'*  Version : 1.0                                               *
'****************************************************************
'  The following code is taken from Dr. Nickels set up of internal oscillator
' PIC16F88 code template for MECH307 Labs

' The following configuration bits and register settings
' enable the internal oscillator, set it to 8MHz,
' disables master clear, and turn off A/D conversion

' Configuration Bit Settings:
' Oscillator                          INTRC (INT102) (RA6 for I/O)
' Watchdog Timer                 Enabled
' Power-up Timer                 Enabled
' MCLR Pin Function                   Input Pin (RA5 for I/O)
' Brown-out Reset                Enabled
' Low Voltage Programming        Disabled
' Flash Program Memory Write          Enabled
' CCP Multiplexed With                RB0
' Code                                Not Protected
' Data EEPROM                         Not Protected
' Fail-safe Clock Monitor        Enabled
' Internal External Switch Over       Enabled

' Define configuration settings (different from defaults)
#CONFIG
    __CONFIG _CONFIG1, _INTRC_IO & _PWRTE_ON & _MCLR_OFF & _LVP_OFF
#ENDCONFIG

' Set the internal oscillator frequency to 8 MHz
DEFINE OSC 8
OSCCON.4 = 1
OSCCON.5 = 1
OSCCON.6 = 1

 ' Define ADCIN parameters
DEFINE  ADC_BITS        10      ' Set number of bits in result
DEFINE  ADC_CLOCK       3       ' Set clock source (3=rc)
DEFINE  ADC_SAMPLEUS    15      ' Set sampling time in uS

' Set up ADCON1
ADCON1 = %10000000 ' Right-justify results (lowest 10 bits)

' borrowed code ends here

' Set up A/D converters and inputs and outputs
ansel = %01100000
TRISB.6 = 1
TRISB.7 = 1
TRISB.0 = 0

'  Define variables
adcVar_v  VAR WORD    ' ADC  result for volume input
dutyCycle_v var byte       ' Duty cycle for PWM for volume control
level_v var byte              ' Variable to hold percentage of volume level

adcVar_s var word          ' ADC result for speed input
dutyCycle_s var byte       ' Duty cycle for PWM for speed control
level_s var byte             '  Variable to hold percentage of speed level
```

```
main:
   pause 1000                ' Set up time for LCD

   while(1)                  ' loop

      adcin 5, adcVar_v      ' get ADC input from volume pot
      dutyCycle_v = adcVar_v/4          ' convert to 0 to 255 duty cycle
      level_v = dutycycle_v*100/255     ' determine percentage

      adcin 6, adcVar_s          ' get ADC input from speed pot
      dutyCycle_s = adcVar_s/4          ' convert to 0 to 255 duty cycle
      level_s = dutyCycle_s*100/255     ' determine percentage
      dutyCycle_s = dutyCycle_s/128     ' reduce speed of motor

      lcdout $FE, 1, "Volume:", $14, #level_v, "%"          ' update LCD with volume level
      lcdout $FE, $C0, "Speed:", $14, #level_s, "%"         ' update LCD with speed level
      pwm portB.0, dutyCycle_s, 10                          ' output motor speed

   wend

end
```

```
'****************************************************************
'*  Name    : LED control                                       *
'*  Author  : Molly and Daniel                                  *
'*  Notice  : Copyright (c) 2016                                *
'*          : All Rights Reserved                               *
'*  Date    : 4/26/2018                                         *
'*  Version : 1.0                                               *
'****************************************************************
' Internal oscillator set up taken from Dr. Nickels, provided to ENGR 4367 class
' PIC16F88 code template for MECH307 Labs

' The following configuration bits and register settings
' enable the internal oscillator, set it to 8MHz,
' disables master clear, and turn off A/D conversion

' Configuration Bit Settings:
' Oscillator                                    INTRC (INT102) (RA6 for I/O)
' Watchdog Timer                        Enabled
' Power-up Timer                        Enabled
' MCLR Pin Function                             Input Pin (RA5 for I/O)
' Brown-out Reset                       Enabled
' Low Voltage Programming               Disabled
' Flash Program Memory Write                    Enabled
' CCP Multiplexed With                          RB0
' Code                                          Not Protected
' Data EEPROM                                   Not Protected
' Fail-safe Clock Monitor               Enabled
' Internal External Switch Over                 Enabled

' Define configuration settings (different from defaults)
#CONFIG
    __CONFIG _CONFIG1, _INTRC_IO & _PWRTE_ON & _MCLR_OFF & _LVP_OFF
#ENDCONFIG

' Set the internal oscillator frequency to 8 MHz
DEFINE OSC 8
OSCCON.4 = 1
OSCCON.5 = 1
OSCCON.6 = 1

' Borrowed code ends here

' Set input/output pints
TRISA.0 = 1
TRISA.1 = 1
TRISA.2 = 1
TRISB.0 = 0
TRISB.1 = 0
TRISB.2 = 0
TRISB.3 = 0
TRISB.4 = 0

 ' Define ADCIN parameters
DEFINE  ADC_BITS        10      ' Set number of bits in result
DEFINE  ADC_CLOCK       3       ' Set clock source (3=rc)
DEFINE  ADC_SAMPLEUS    15      ' Set sampling time in uS

' Set up ADCON1
ADCON1 = %10000000 ' Right-justify results (lowest 10 bits)

' Set up A/D converters and input/output pins
ansel = %00100100  'Turn on A/D converters 2 and 5
TRISB.6 = 1
TRISA.2 = 1


‘Define photoresistor variables
```

```
photo1_adc  VAR WORD    ' ADC  result for first photoresistor
photo1_duty var byte    ' Duty cycle calculated from first photoresistor
photo2_adc var word     ' ADC result for second photoresistor
photo2_duty var byte    ' Duty cycle calculated from second photoresistor
photo_add var word      ' Total binary value from both photoresistors
photo_avg var byte      ' Average photoresistor level

'Define input/output variables
inputA var portA.0      ' Bit 2 of 3-bit input code
inputB var portA.1      ' Bit 1 of 3-bit input code
inputC var portA.3      ' Bit 0 of 3-bit input code
output0 var portB.0     ' Output LED row 0
output1 var portB.1     ' Output LED row 1
output2 var portB.2     ' Output LED row 2
output3 var portB.3     ' Output LED row 3
output4 var portB.4     ' Output LED row 4


 main:

   while(1)                          ' start while loop
   adcin 2, photo1_adc           ' convert first photoresistor level to digital value
   adcin 5, photo2_adc           ' convert second photoresistor level to digital value
      photo1_duty = photo1_adc/4          ' determine duty cycle of first photoresistor
      photo2_duty = photo2_adc/4          ' determine duty cycle of second photoresistor
      photo_add = photo1_duty + photo2_duty       ' determine total photoresistor level
      photo_avg = photo_add/2              ' find average photoresistor level (LED brightness)
      if(photo_avg < 51) then              ' set minimum brightness of LEDs
         photo_avg = 10
      endif
      if((photo_avg >= 51) and (photo_avg < 69)) then
         photo_avg = 20                    ' set brightness of LEDs
      endif
      if((photo_avg >= 69) and (photo_avg < 81)) then
         photo_avg = 40                    ' set brightness of LEDs
      endif
      if((photo_avg >= 81) and (photo_avg < 93)) then
         photo_avg = 60                    ' set brightness of LEDs
      endif
      if((photo_avg >= 93) and (photo_avg < 105)) then
         photo_avg = 80                    ' set brightness of LEDs
      endif
      if((photo_avg >= 105) and (photo_avg < 114)) then
         photo_avg = 100                   ' set brightness of LEDs
      endif
      if((photo_avg >= 114) and (photo_avg < 129)) then
         photo_avg = 120                   ' set brightness of LEDs
      endif
      if((photo_avg >= 129) and (photo_avg < 141)) then
         photo_avg = 140                    ' set brightness of LEDs
      endif
      if((photo_avg >= 141) and (photo_avg < 153)) then
         photo_avg = 160                   ' set brightness of LEDs
      endif
      if((photo_avg >= 153) and (photo_avg < 165)) then
         photo_avg = 180                   ' set brightness of LEDs
      endif
      if((photo_avg >= 165) and (photo_avg < 177)) then
         photo_avg = 200                    ' set brightness of LEDs
      endif
      if((photo_avg >= 177) and (photo_avg < 189)) then
         photo_avg = 220                    ' set brightness of LEDs
      endif
      if((photo_avg >= 189) and (photo_avg < 201)) then
         photo_avg = 240                   ' set brightness of LEDs
      endif
      if((photo_avg >= 201)) then
```

```
        photo_avg = 255                 ' set maximum brightness of LEDs
      endif

   if(inputa != 0) then                 ' read Bit 2 of input code
      if(inputB != 0) then              ' read Bit 1 of input code
         pwm output0, photo_avg, 10     ' turn on corresponding rows w/ duty cycle determined by photoresistors
         low output1
         pwm output2, photo_avg, 10
         low output3
         low output4
      else
         if(inputc != 0) then
            low output0                  ' turn on corresponding rows
            pwm output1, photo_avg, 10
            low output2
            pwm output3, photo_avg, 10
            low output4
         else
            low output0       ' turn on row 4 of LEDs if code is 100
            low output1
            low output2
            low output3
            pwm output4, photo_avg, 10
         endif
      endif
   else
      if(inputb != 0) then
         if(inputc != 0) then
            low output0       ' turn on row 3 of LEDs if code is 011
            low output1
            low output2
            pwm output3, photo_avg, 10
            low output4
         else
            low output0       ' turn on row 2 of LEDs if code is 010
            low output1
            pwm output2, photo_avg, 10
            low output3
            low output4
         endif
      else
         if(inputc != 0) then
            low output0       ' turn on row 1 of LEDs if code is 001
            pwm output1, photo_avg, 10
            low output2
            low output3
            low output4
         else
            pwm output0, photo_avg, 10  ' turn on row 0 of LEDs if code is 000
            low output1
            low output2
            low output3
            low output4
         endif
      endif
   endif

   wend

end
```