ECE 493
Smart Living Room LED
Final Report

December 3, 2012

Maryam Nasri
Marzia Shabbir
Mayank Mehta
Girmay Tewelde

Executive Summary:

The objective of this project was to design a Smart Living Room LED light. The LEDs and the circuitry for the device were nicely integrated into a tochiere lamp. The device was easily installed and worked well with a power supply. The device was designed in a manner such that it can operate in two different modes: automatic and manual. The automatic mode takes a time based on the user input and adjusts the light and imitates sunlight based on different times of the day. The manual mode would let the user to control the device through an infrared interface and set the different constant modes of light. The microcontroller was used to drive the functionality of the LEDs in conjunction with the h-bridges and interact with the user through an infrared detector. Furthermore, the power supply was used to step down the voltage as per as the required needs for all the different modules of the device. The device is made so that it is easily commutable from one place to another and be able to provide a natural and soothing light for the user.

Table of Contents

Approach

Need

Human brain shows different reactions to different colors of light. Some light colors are calming and relaxing while some are energetic. Cool colors reflect the fresh violets and blues of moonlight. Warm colors project the hot hues of sunlight and create a feeling of warmth in the room. In addition, light therapy can be used to treat illnesses such as depression. Some individuals become depressed during the long winter months when sunshine is limited. This problem is very common in countries located 60° latitude and above where the sun remains very low during the winter month. As a result, people are not able to get enough sunlight. These individuals can be exposed to a light specially designed to mimic that of the sun. In general, having the right kind of lightening in work and home environment can help people's mental and emotional health and improve their performance.

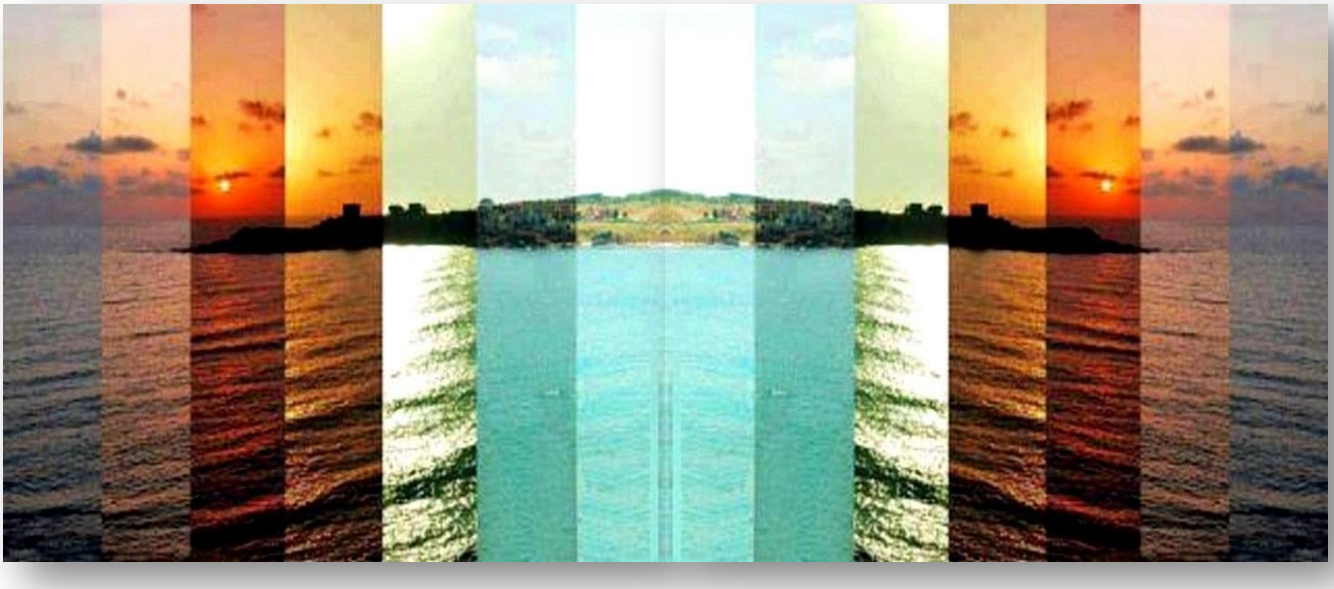


Figure 1. Sunlight Spectrum throughout the day

From the beginning of human life on this earth, people begin their work with the rise of the sun and sleep when the sun goes down as shown in the figure above. Their body clock is synchronized with the time zone they live in. However, there are some cases when this body clock is affected and a person develops certain sleeping disorders such as insomnia. Not only that but there could be several other cases in which a person finds it difficult to sleep at night and feel fresh and active in the day time. A person

experiences similar effects when going through a jet lag. A clinical research suggests that natural production of melatonin (hormone which makes one sleepy) could be reserved by exposing one to low levels of light in the blue part of the spectrum and blocking that part of spectrum could help melatonin to flow. LEDs (Light emitting diodes) are solid light bulbs, which are extremely energy efficient and durable. More over the brightness of LEDs could be controlled easily according to ones need. Thus, LEDs efficiency is not limited to its less energy consumption and durability but its precise control capabilities could be used to deal with issues of jet lag and sleep disorders such as insomnia. The Goal of this project was to design a 150W equivalent LED light source in intensity that imitates the sunlight in order to possibly deal with the conditions mentioned above.

Objective

The goal of this project was to design and implement a sunlight replicator via LEDs that can be purchased by a consumer and used at home or work. The lamp should be affordable and easy to use and maintain. Interacting with the device should be easy enough so it does not need any require additional training on how to use it.

Requirements Analysis Summary

- Possibly used as a sleep-aide for a user going through insomnia
- Acquire and compare power consumption with a regular light sources
- Opportunity to help passengers adjust to jet lag more easily
- Create a light source without being a by-product of mercury
- Develop a 150W light bulb equivalent LED based light source
- Designed focused for a torchiere floor lamp
- Imitate sunlight spectrum using LEDs
- The brightness and color output of the LEDs must be independent of each other
- Generate a long-lasting light source
- A user controlled wireless interface with the lamp through Infrared

- The integration of the design with a microcontroller
- The lamp output should have approximately 2600 lumens at 100 percent brightness

Contributions

The entire team worked on various aspects of the project in order to make this project workable. The major tasks for this project were the power supply, user interface, integration of LEDs with H-bridges and microcontroller, comparing color temperature of the LEDs with sunlight and intensity with an incandescent light bulb and CFLs. Some of the other tasks that were involved in this project were designing a PCB, assembling the lamp, designing the LED matrix, and putting everything together. The team was evenly split in doing all these tasks.

Maryam was the project manager. She worked on researching and buying the desired power supply for this project. She also made the LED matrix and contributed towards assembling the lamp together. Furthermore, she provided assistance in designing the schematic for this project and the PCB as well.

Mayank worked on the programming aspect of this project. He programmed the microcontroller in order to produce the entire color spectrum from the LEDs as well as producing the various colors in the manual mode. He also provided assistance in putting the hardware together, assembling the lamp as well designing the PCB.

Girmay also worked on the programming for this project. He programmed part of the User Interface which involves decoding the signal from IR detector and be able to carry the action based on that particular input. He also helped in programming the microcontroller to produce the color spectrum from the LEDs.

Marzia worked on different parts of this project. She also did programming for the User Interface which involves programming the LCD to show different options on the screen to help the user guide through choosing all the options. She did research on color temperature and measured the color temperature for sunlight, the lamp and compared both of them. She measured the light intensity of the lamp and bulbs and compared them as well. Furthermore, she provided assistance in designing the PCB as well.

Technical Section

Level Zero Design

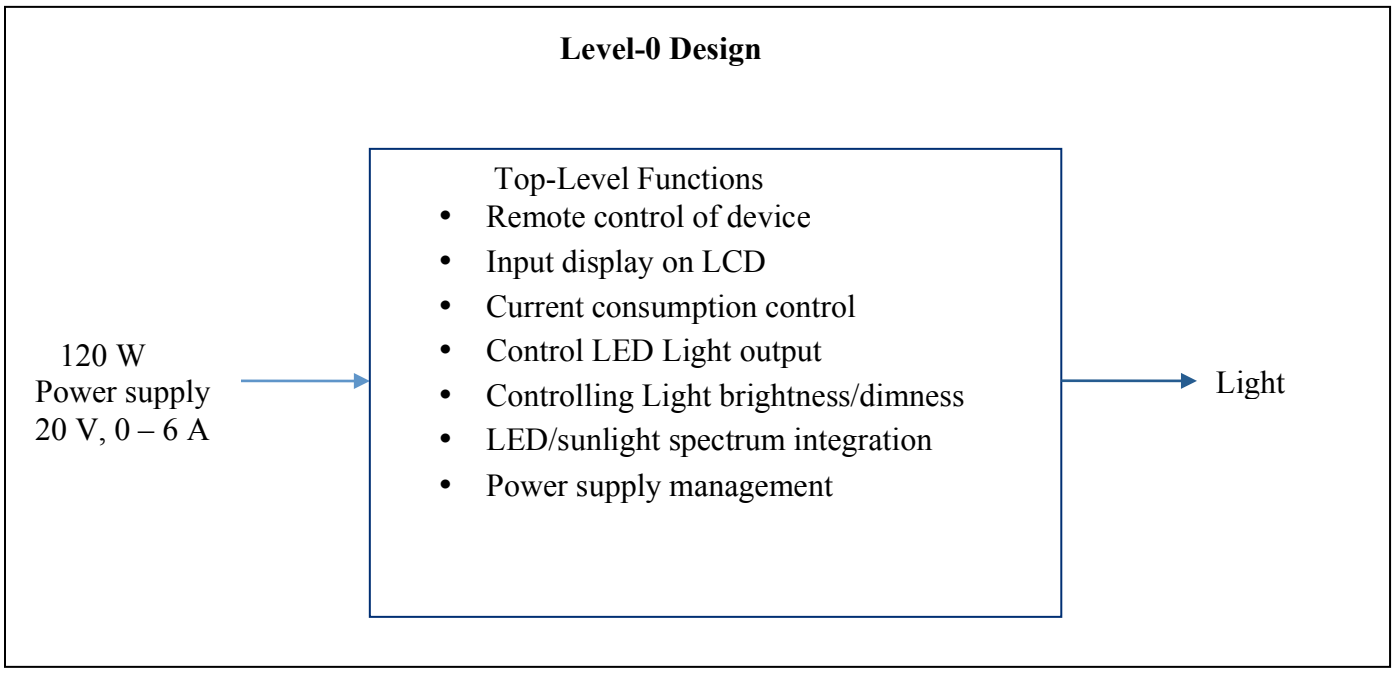


Figure 2: Level Zero

The above figure gives a top level idea of the device and its functions. The power to the system is provided by AC power outlet. The Power supply converts the AC voltage to DC voltage and acts as the input to the system. The Light being produced through LEDs acts as an output. The device should be able to perform necessary functions such as decoding signal being sent from the remote and act based on that signal. It will also control the LED light output and change the colors and the brightness independently of each other. It will also be able to drive an LCD as a guide for the user to interact with the device.

Level One Design

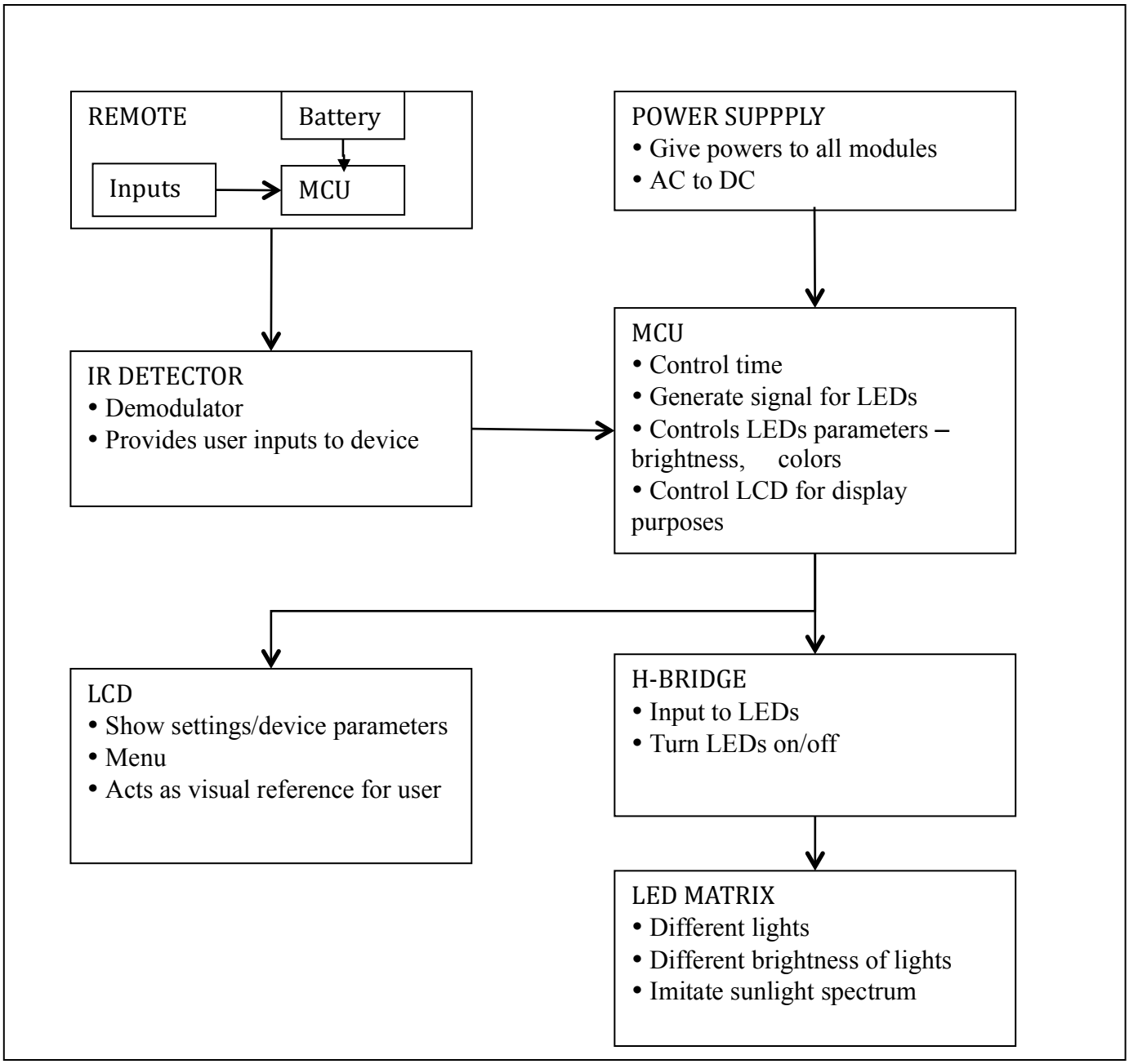


Figure 3: Level One

The above figure demonstrates the Level One design on our system. It goes into the details of each specific component that corresponds to the device. The device will be remotely controlled by the user and serve as a basis to control the device. All the other components are part of the device and would be mounted on the body of the device. The purpose of the IR detector is to decode the signal being sent by the remote control and provide this decoded signal to the microcontroller. The Microcontroller acts as the central part of the device. It controls various things such as the LCD and the H-bridge. It provides output to the LCD which

acts as a visual reference for the user and shows all parameters of the device and the time of day. The H-Bridge is used to provide higher current to the LEDs and hence, the microcontroller controls the H-Bridge through a PWM signal and is able to change the parameters such as color and brightness of the LEDs.

Level Two Design

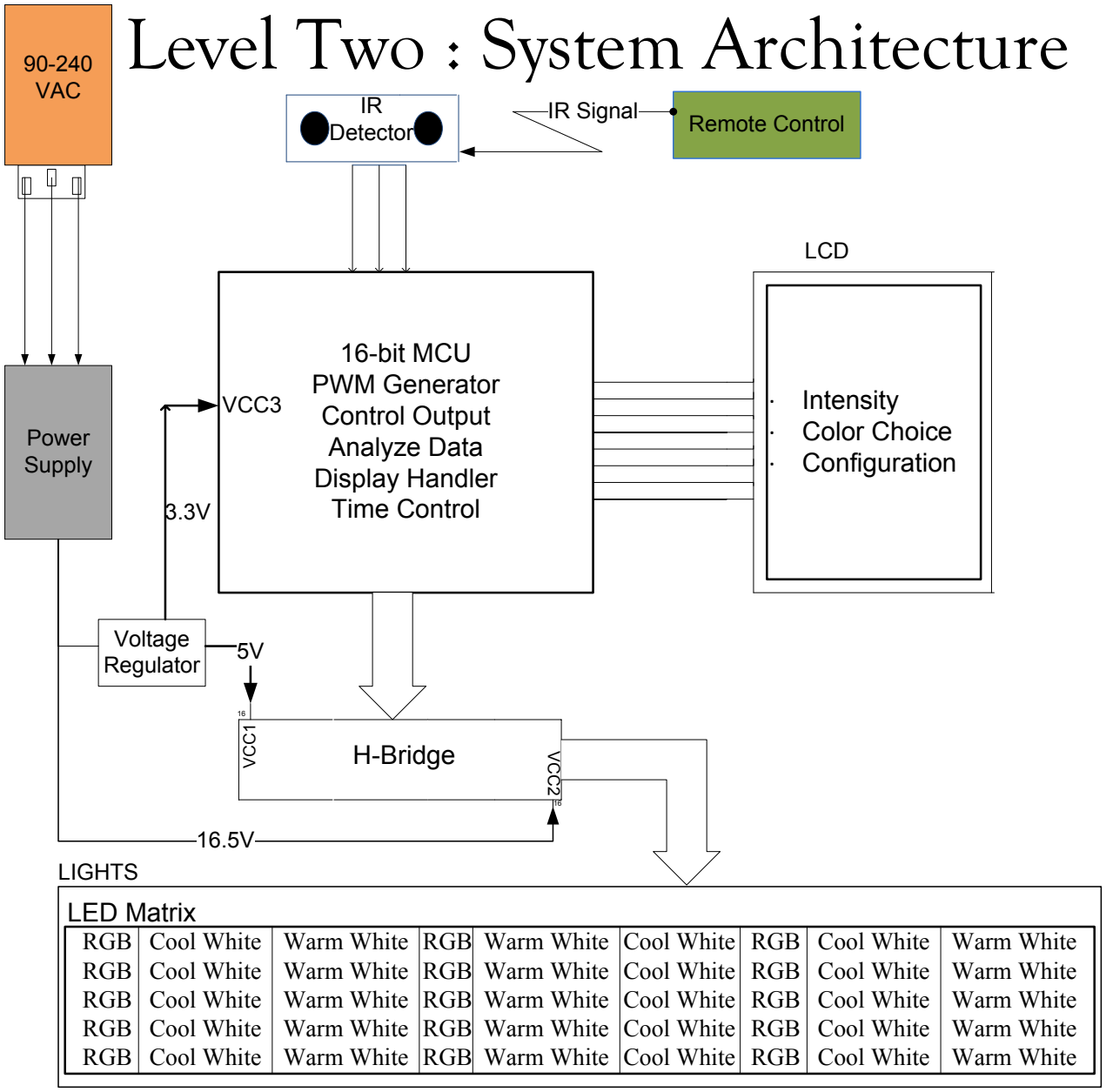


Figure 4. Level Two

This picture shows the level two design which goes into more detail towards describing each component. A total of 45 LEDs are used in this project which makes up the LED Matrix. The power supply can take input from 90-240 VACS and hence, can be used in different parts of the world. Furthermore, it outputs

16.5 Volts which are fed into the H-bridge. The H-bridge also requires a 5V logic supply which is given by taking the 16.5V and stepping into down by using a switch LM7805 voltage regulator. Moreover, this 5 V input is also fed to 3.3V voltage regulator to step it down more and provide as the voltage supply for the microcontroller. The microcontroller controls the various functions of the lamp. These functions include decoding signal from the IR detector in order to process the button input, and sending different PWM signal to the H-bridge to provide different colors and intensity for the light. It also controls the LCD and shows the output on there since the LCD acts as a visual reference to the user.

One Full Cycle of Light Transition

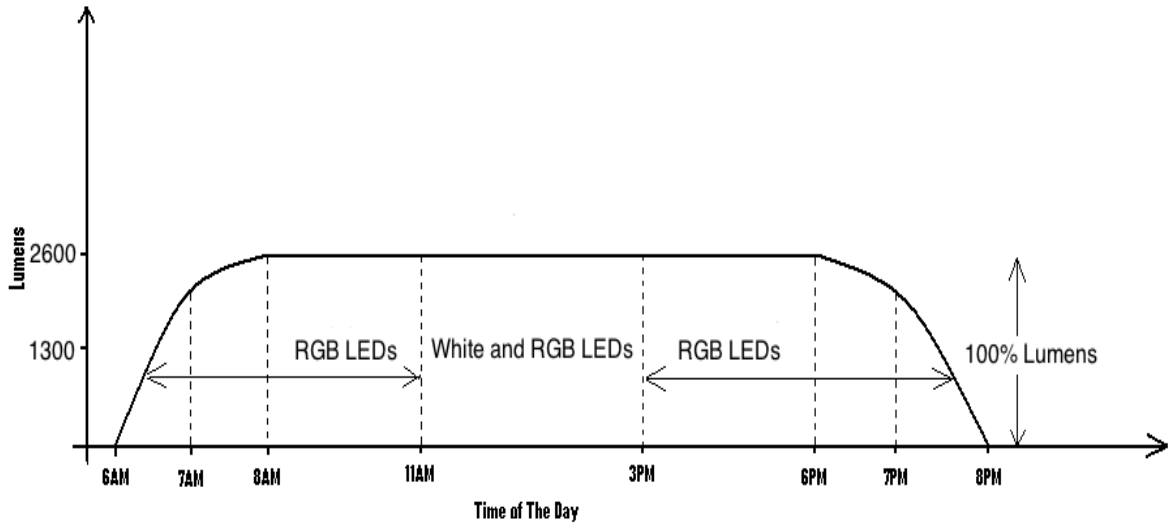


Figure 5. Lumens vs. Time of the Day

This figure above shows one full cycle of light transition of the lamp. The lamp runs from 6am to 8pm. As it turns on, the RGB LEDs are on and provide a light similar to sunrise. As time passes, the light output gets brighter and the color starts changing towards a white light. In mid-day RGB LEDs and White LEDs are both on to provide the necessary brightness and the light. The mid-day ranges from 11am to 3pm and then white LEDs turn off and RGB LEDs alone remain on. As it is close to 8pm the light intensity starts to decrease and color starts changing to provide similar to sunset. At 8pm the lamp turns off.

Components of the Device

Power Supply:

The power supply is one major component for this project. It is used to take the AC input and convert it to a DC voltage which can be used to power the various components of the lamp. The power supply being used for this system is Mean Well HLG-120H-20A. This power supply can take 90~305VAC as input with frequency 47~63Hz and generate output 20V DC with variable current from 0 to 6 Amps. This power supply should be sufficient to power our device properly.

Figure 6: Power Supply

Microcontroller

Figure 7: Msp430H5438 Header Board

A Microcontroller is nothing more than a small computer on a single integrated circuit with some processor core, memory and input/output peripherals. Most of the applications used for microcontrollers are embedded applications. They are designed to run one task and the program is stored in the ROM and

generally does not change unless it is being programmed again. Furthermore, they are low-power devices and consume less power.

There are several vendors for microcontrollers that are available in the market these days. Some of these vendors are Microchip providing microcontrollers from 8-bit to 32-bit, Infineon 8-bit to 32-bit, Texas Instruments (16-bit), Atmel AVR 8-bit and 32-bit. The microcontroller that is being used for this project is Texas Instruments TI MSP430F5438. This microcontroller is embedded into a header board provided by Olimex and the model number is MSP430H5438. This microcontroller provides a low supply voltage range 1.8V to 3.6V, 16 bit RISC architecture, three 16-bit timers, and real time clock. This microcontroller is able to provide enough PWM signals for all the H-Bridges and hence, control the LEDs. This microcontroller also has enough output pins to easily connect with the LCD and the IR detector. The real time clock will help to remember the time entered by the user and operate based on this time. Hence, this microcontroller is able to provide the necessary features for this project.

H-Bridge

An H-Bridge is a motor-driver often used to provide high current to DC motors than a microcontroller can provide. In our project, we are able to provide higher current to the LEDs using an H-Bridge. There are several different H-Bridges that are available in the market and the H-Bridge for our project is SN754410 Dual H-Bridge Motor Controller (see figure below). This is a dual H-Bridge so it can provide output to 2 columns of a LED matrix. The reason we are using this h-bridge is because we have prior experience of working with them to drive motors. It was efficient enough to supply 1 Ampere current to the motors and it could be re used to supply 350mA to the LEDs we are using.

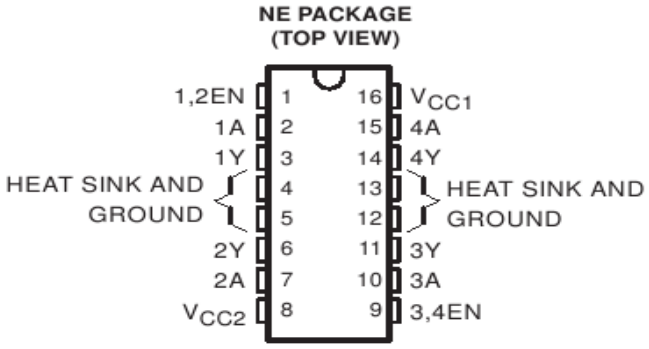


Figure 8: H-Bridge

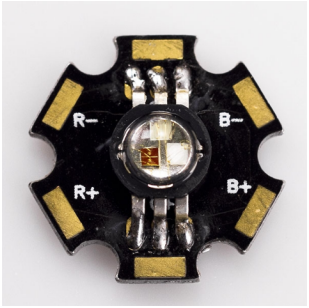
Liquid Crystal Display

An LCD is required in this project to serve as a visual reference to the user. The LCD will be mounted directly on the torchiere lamp and show different parameters such as brightness, color and time of the day to the user. It will also show a menu which will guide the user to change the settings of the device.

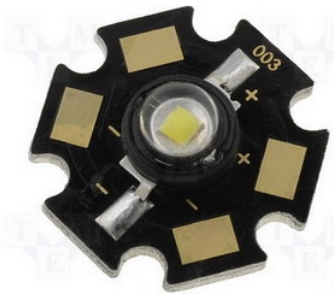
The LCD for this project is NEWHAVEN Display part number 0420AZ-FSW-GBW (see figure below). This LCD requires a +3.0 V supply in order to properly operate. It features a 4 lines * 20 characters and hence, giving enough space to draw the menu and show parameters. It also has a white LED backlight which can be adjusted to change the contrast of the LCD.

Figure 9: LCD

Light Emitting Diodes



RGB Led



Cool/Warm White Led

Figure 10: LEDs

There are different types of LEDs that are being used for this project. These LEDs are RGB, cool white and warm white. The RGB LEDs has a power consumption of 3W as there are three LEDs (red, green, and blue) mounted together in one place. Each LED consumes about 1 W because with a forward voltage of 3.4 – 3.8 V for Blue/Green LEDs and forward current of 350 mA giving approximate power consumption of 1.19 W to 1.33 W for single LED. The power for Red LED is (2.5 – 2.8 V) * 400 mA being equal to 1 – 1.12 W. Hence, the total power consumption for a RGB LED goes from 3.38 to 3.66 W. The RGB LEDs when fed with the right PWM signals would be able to produce necessary colors required to imitate sunlight. The cool white and warm white LEDs have power dissipation of 1W each. These LEDs would mostly be on during mid-day in order to imitate a white-sunlight and parts of these LEDs would be on during the morning time and evening and act as a helper to the RGB LEDs towards producing the desired colors. The table below shows different parameters of the LEDs in great detail.

	Warm Whites (1W)	Cool Whites (1W)	RGB (3W)
Forward Voltage	3.0 – 3.8 V	3.0 – 3.8 V	Blue/Green 3.4 – 3.8 V Red 2.5 – 2.8 V
Forward Current	350 mA	350 mA	Blue/Green 350 mA Red 400 mA
Emitting Angle	120 – 140 °	120 – 140 °	120 °
Lumens	65 – 75 lumens	76.9 – 87.9 lumens	Red 60 lumens Green 55 lumens Blue 20 lumens

Table 1: LEDs parameters

IR Detector

For a user interface we implemented a remote control and an on-board control. We used an IR detector to remotely control our lamp. IR detectors are microchips tuned in to listen to infrared light. They are used for remote control

Figure 11. IR Detector

detection. The remote control emits IR pulses using matching IR LED tuned to a specific frequency. The IR detector receives and analyzes the pulse to run a particular command. IR LED inside the remote have to be pulse width modulating at a specific frequency in order for the IR detector to detect that frequency. So for our project we used an IR detector tuned to a frequency of 38 kHz. The remote control we used is a Sony Remote control that is specifically programmed for a radio. The specific model of the IR detector is GP1UX311QS, it can filter and demodulate incoming infrared signal. It also can receive signal from 13 feet away. It can be easily integrated to a microcontroller with a three-pin output. Once the user presses a button on the remote control the IR detector processes data and implements the command. We integrated the IR detector with the microcontroller so every time a specific button is pressed the lamp output is changed. So we used the button to control brightness and the color of the LED light source. At later point we can also use the remote to input the specific time that the lamp should be on for. The remote will be also used as a power on/off switch for the lamp. The remote we are using is a Sony remote control. Each remote control manufacturer uses different protocols to modulate infrared signal. Sony uses SIRC (Serial Infra-Red Control) protocol. This protocol uses pulse width modulation to encode the bits. The remote we are using is a 13-bit protocol. It has a 5-bit device code that identifies the specific device to be used on. It also has a 7-bit command code that represents the actual button pressed on the remote control.

The protocol starts off with a start bit that is 2.4ms long to signal the start of the SIRC message. A typical pulse modulation of an SIRC protocol is as shown below.

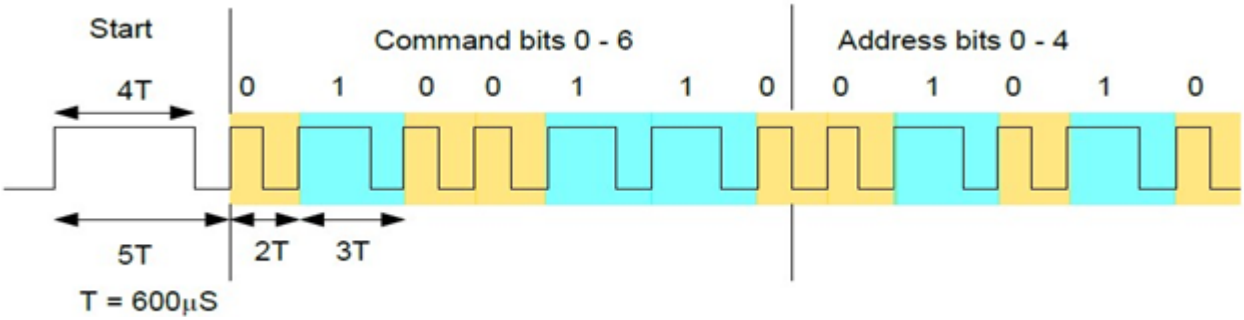
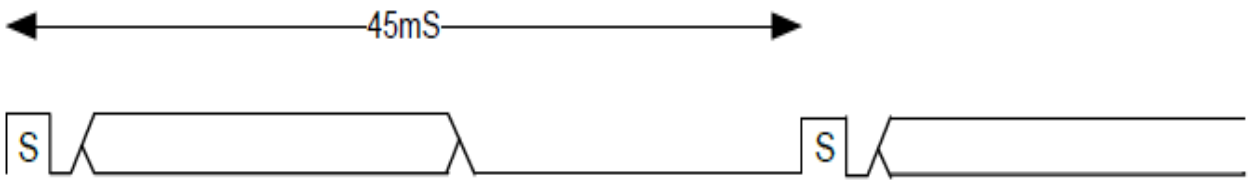


Figure 12: SIRC Message

The pulse modulation is based on multiples of .6ms. The data is modulated from signal with the least significant bit first as shown in figure 4. The pulse representing a bit '1' is a 1.2ms long burst of the 38 kHz carrier signal, while the burst width for bit '0' is .6ms long. All bursts are separated by .6ms. All SIRC messages are repeated every 45ms.



The possible use of the buttons on the remote is specified in the table below.

Button	Light Source Integration
Digit key 1	Time/ MENU select
Digit key 2	
Digit key 3	
Digit key 4	
Digit key 5	
Digit key 6	
Digit key 7	
Digit key 8	
Digit key 9	
Digit key 0	
Recall	Main Menu
Volume +	increase brightness
Volume -	decrease brightness
Power	Turn on/off lamp

Table 2: Command Table (Remote)

Simplified State Diagram

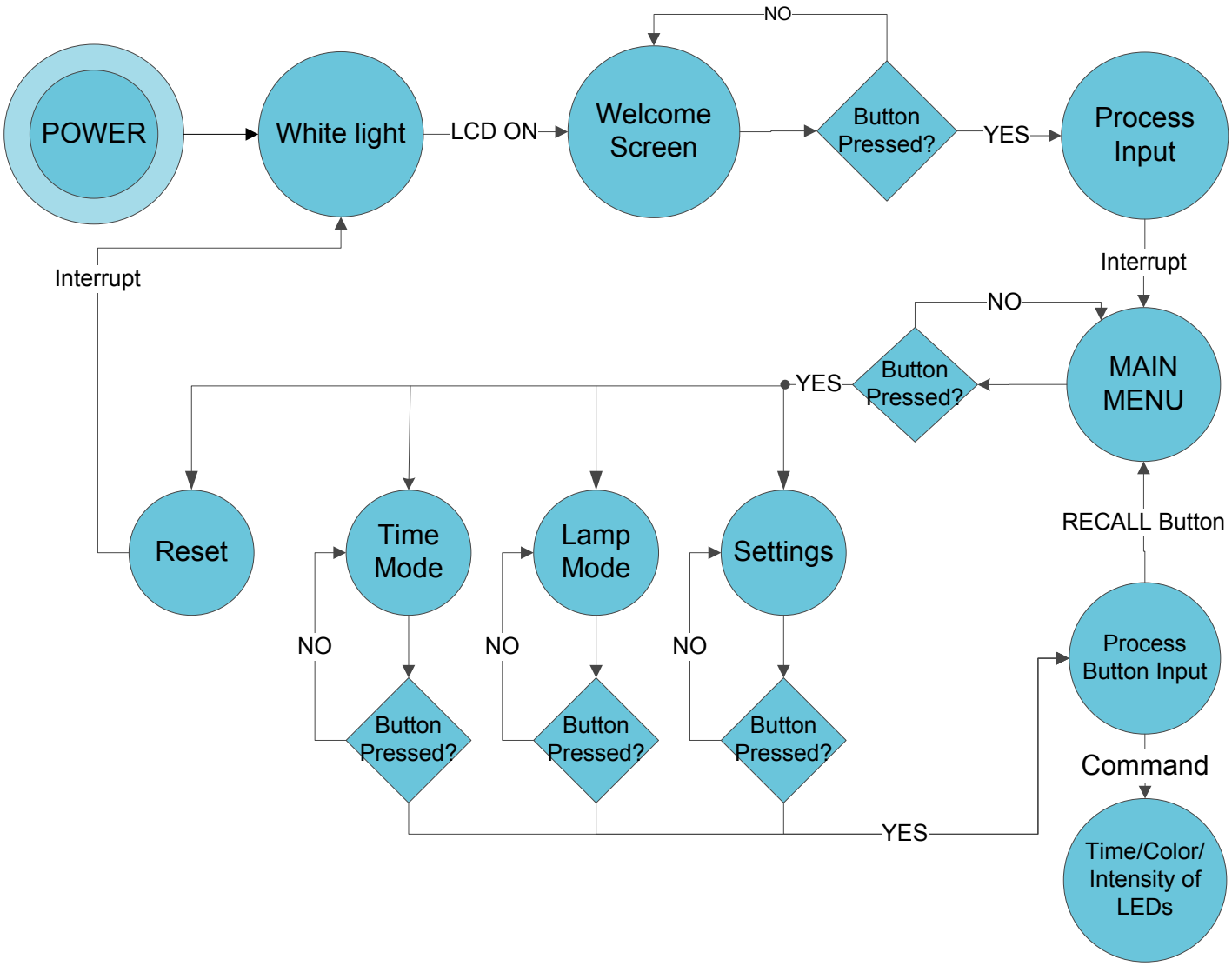


Figure 13. State Diagram

This figure shows the simplified state diagram for this lamp. As the lamp turns on, it produces a white light at 50 % brightness and the LCD turns on showing a welcome screen. The user is then welcome to press any input from the remote in order to control the different settings of the lamp. The button input is processed using an IR detector and microcontroller and different modes such as time mode, lamp mode, and settings can be selected. This can be used to control the time, color, and intensity of the LEDs.

The lamp mode is used set different settings for the lamp. The different options are Automatic Mode, Manual Mode, Normal Mode and Turn Off. The Automatic Mode is the main functionality of the lamp. It lets the lamp output a particular light based on the time showing on the LCD. The automatic mode works from 6 am to 8 pm. The Manual Mode lets the user choose different settings such as candle light,

incandescent, warm white, cool white, halogen, day light, direct sunlight, and blue sky. The third option is the normal light which means all LEDs are on and produce a normal light which is similar to white light output. The last option turn off lets the user turn off the lamp.

The Time Mode is another setting which can be used by the user to enter time for the lamp. It lets the user choose between AM/PM and Military time and enter the time. If Automatic Mode is set, the lamp light output would change corresponding to the new time being entered. This can also help the user if he/she wants to enter a fake time and produce light output similar to sun during the night.

The third mode is Settings. This mode lets the user choose from two different options. One is reset which causes software reset on the device and takes the light back to white light and time back to 6 am. The other is brightness which lets the user change the brightness of the current light being output. The Recall button is programmed in a manner that it takes the user back to the main menu anytime it is pressed.

Schematic

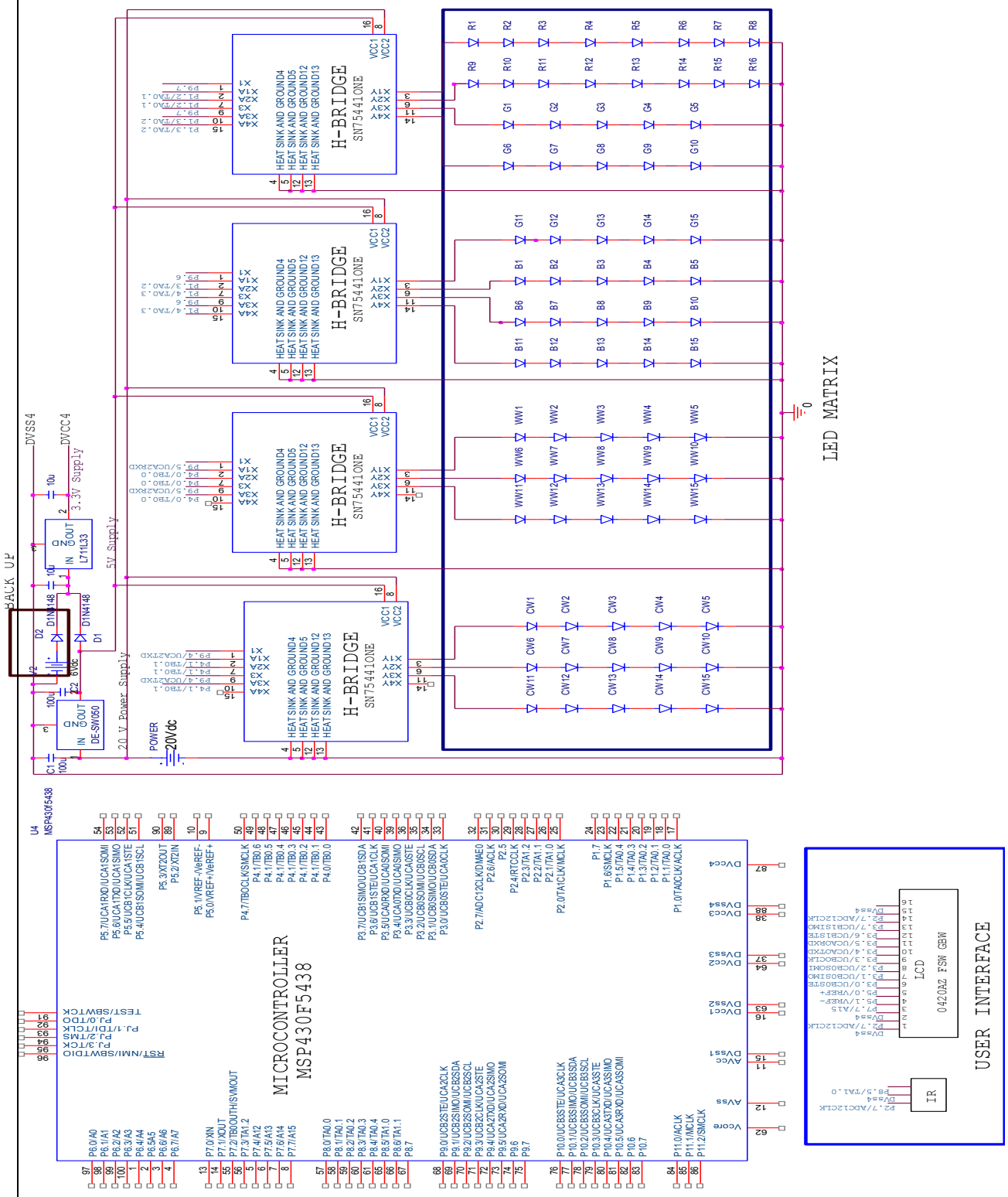
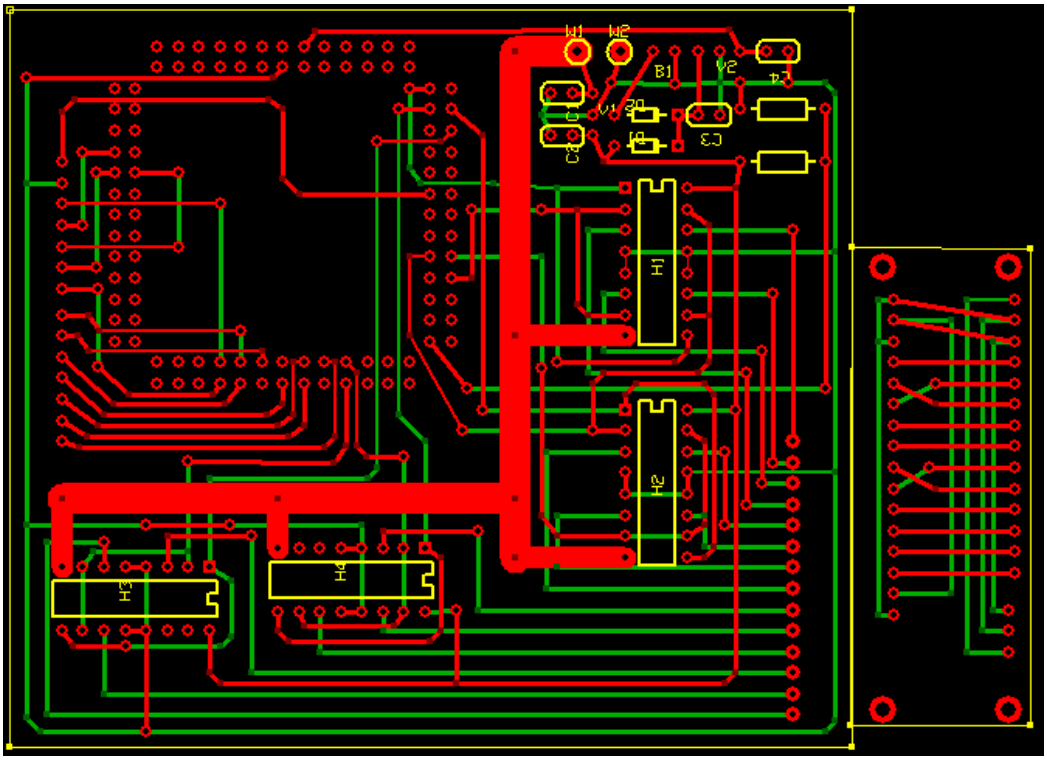


Figure 14. Schematic

PCB Layout



Red – Top Layer
Green – Bottom Layer
Yellow - Components

Figure 15. PCB Layout

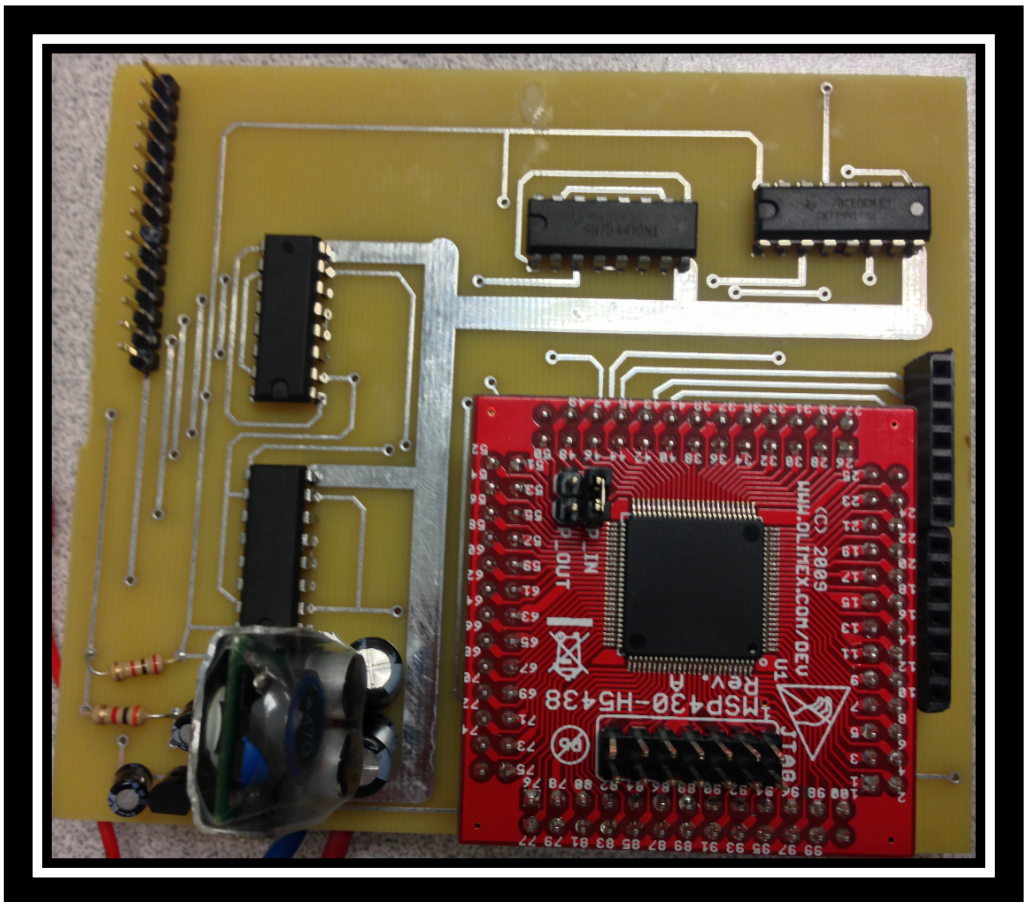


Figure 16. PCB

LED Matrix

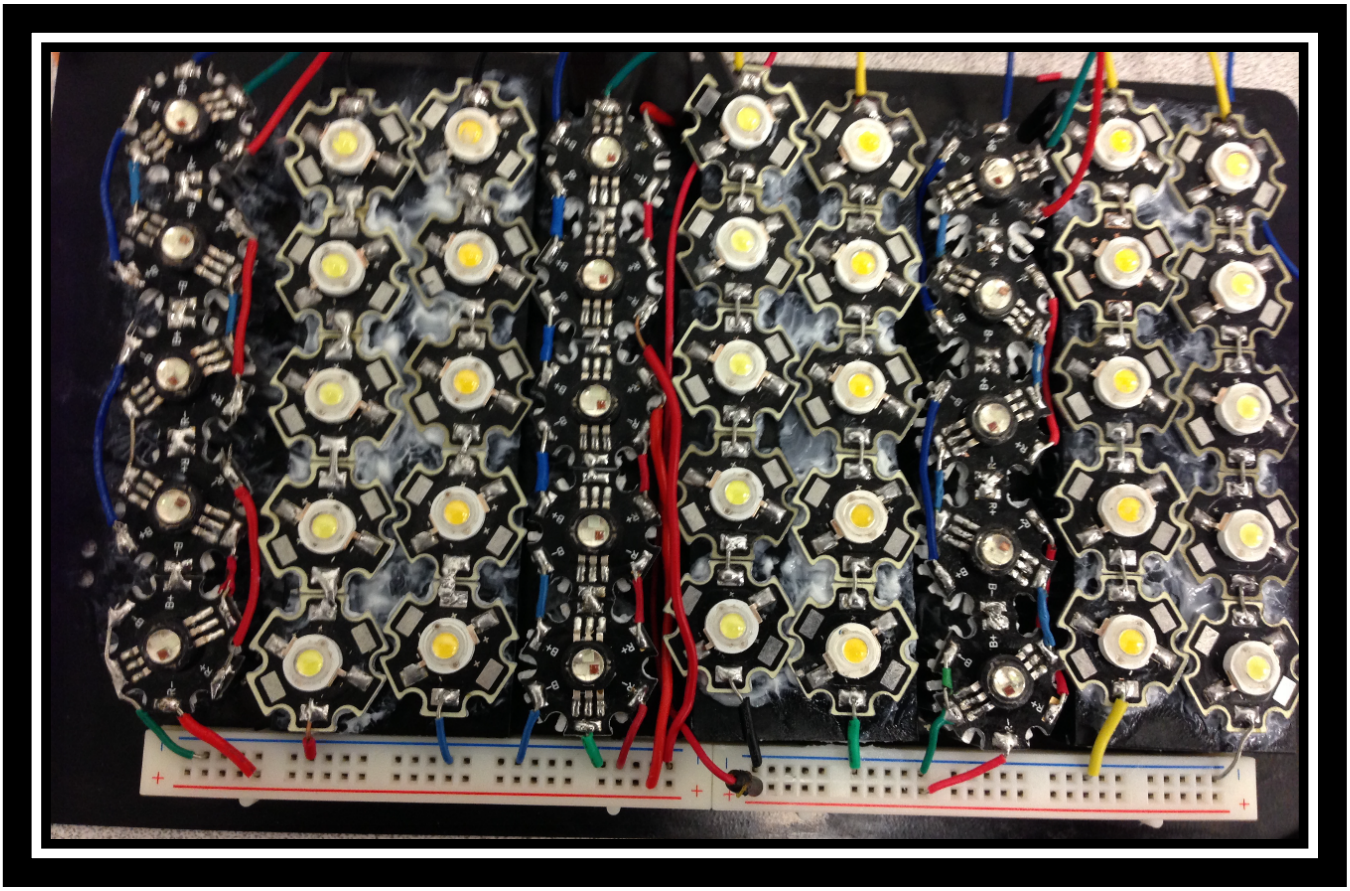


Figure 17. LED Matrix

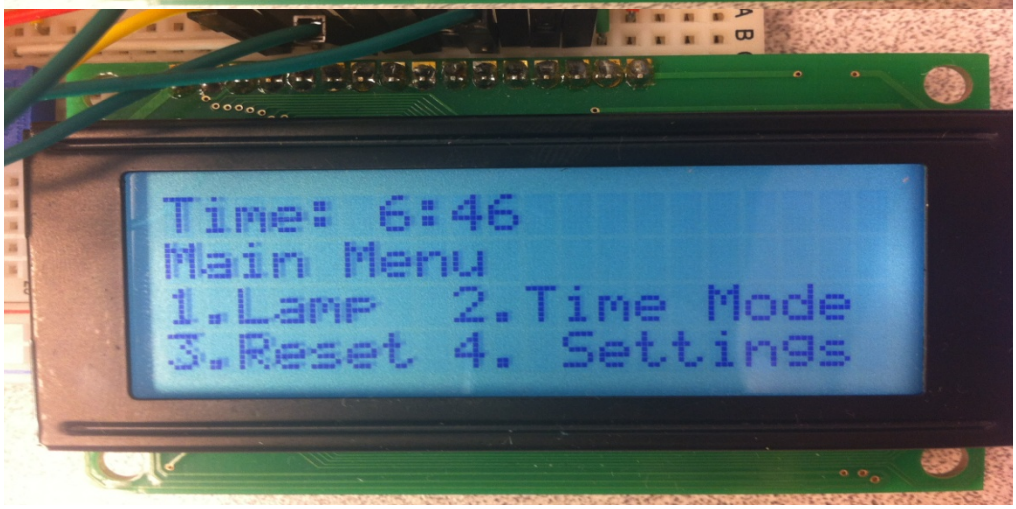
Experimentation

Test 1: Remote control access of the lamp

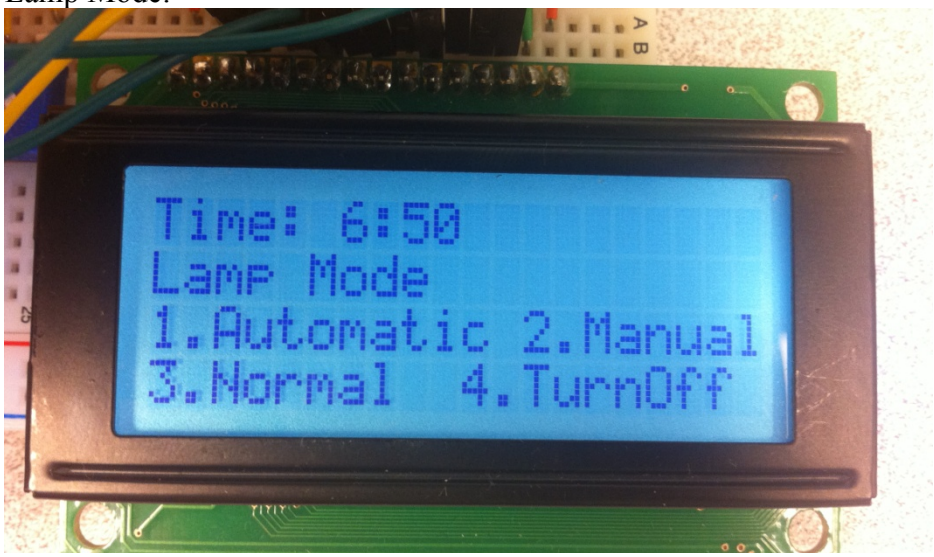
We captured the each button press and implemented a basic access of each mode of our project. Initially we tested that all button inputs were read accurately. Once that was done we moved to the next step, control of the lamp output. We handled that mode by first starting out by implementing the user time input portion of the project. The user can use the remote to input the curtain time of any specific day. Given the time the user can control when the lamp starts lighting up. The next step after this implementation is to handle which types of lights are outputted given the user input. The remote control sets the time when the full cycle of day can be started. This is implemented in automatic mode, where the sunlight replicator is started and ran for a specific time period. In the lamp mode, there is a section called manual mode that produces specific light output. Using the remote the user can select a desired light output. On top of that the user can select to output the lamp at full power. Brightness can be controlled by the user using the volume up/down button. Each button inputs in relation to each mode were tested for accuracy and handled according to each mode implementation.

Lamp Mode	Time Control	Brightness
Automatic • **set time** • **let run**	AM/PM Time • 12 hour cycle using numbers	Intensity • increase/decrease brightness
Manual • **Choose lights**	24-hour time • Numbers used to input time	
Normal • **Full daylight**		

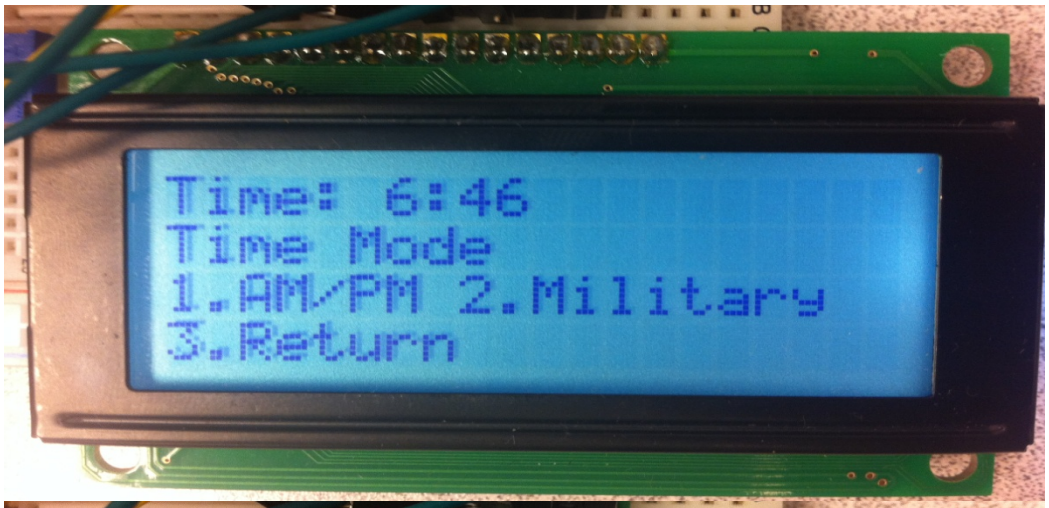
Main Menu:



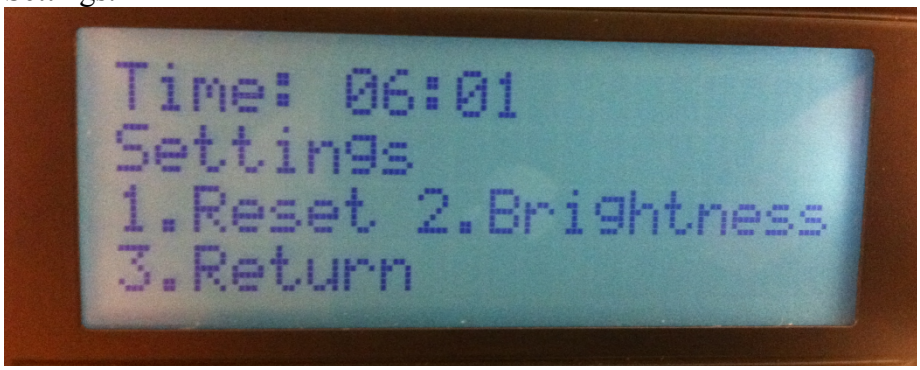
Lamp Mode:



Time Control:



Settings:

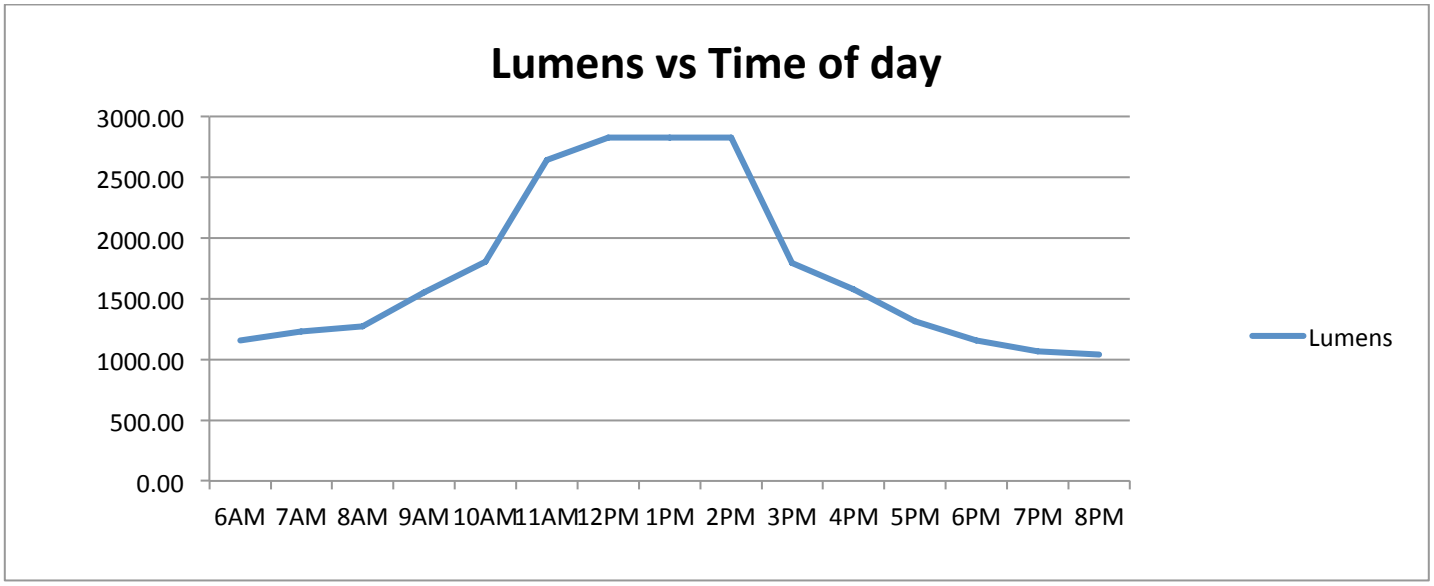


Test 2: Generate a spectrum of colors using LEDs

RGB's were tested to generate lights at different time of the day. Each light was outputted to replicate the skylight output. We produced a replica of skylight for duration of 14 hours. From 6am to 8pm to be exact. We got the lumens of each light source output at each time period.

Time	Lumens*
6AM	1156.84
7AM	1231.13
8AM	1273.58
9AM	1550.91
10AM	1805.05
11AM	2641.63
12PM	2828.42
1PM	2828.42
2PM	2828.42
3PM	1793.63
4PM	1576.06
5PM	1316.04
6PM	1156.84
7PM	1066.63
8PM	1040.09

*Lumens measured from a distance of .65cm



Also we tested individual light output using LED s and collected data as provided below.

Light Mode	**Lumens**
candle	918.04
warm white	987.03
incandescent	1193.99
halogen	1040.09
cool white	668.63
cloudy sky	1576.06
day white	2483.49
direct sunlight	1576.06
blue sky	1406.25
normal	2641.63

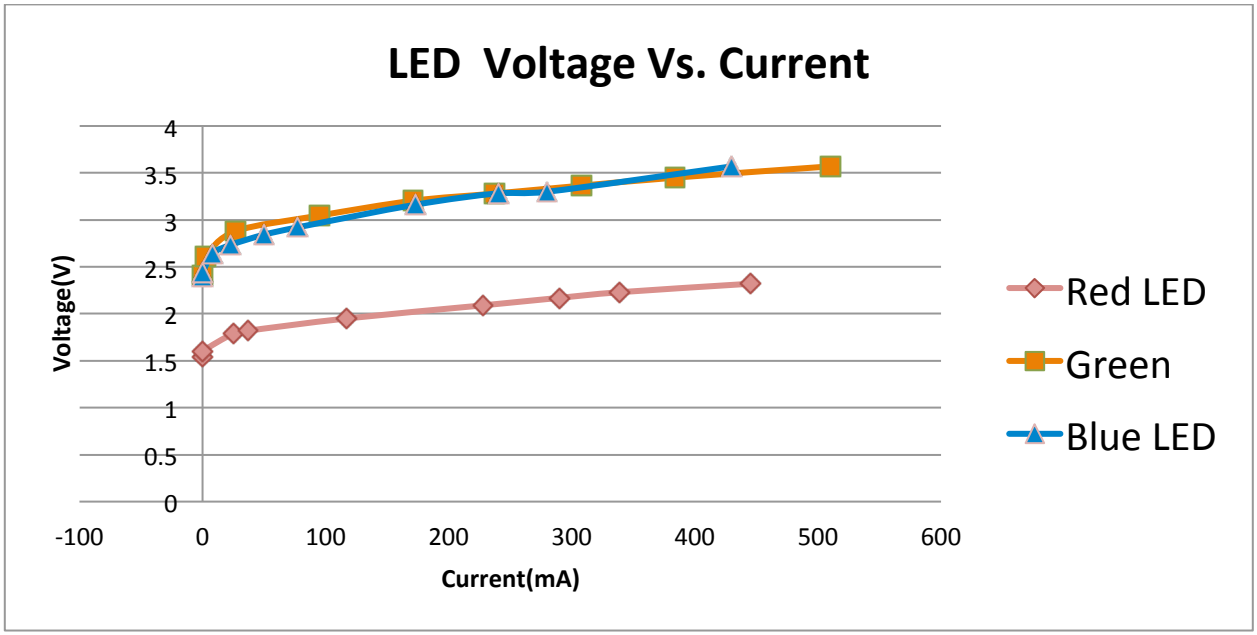
Picture of light output of each light produced above.

Test 3: Measure Power Consumption

We measured the power consummation of each LED at each specific time with each specific light output. We measured the voltage of each LEDs and their respective current consumption.

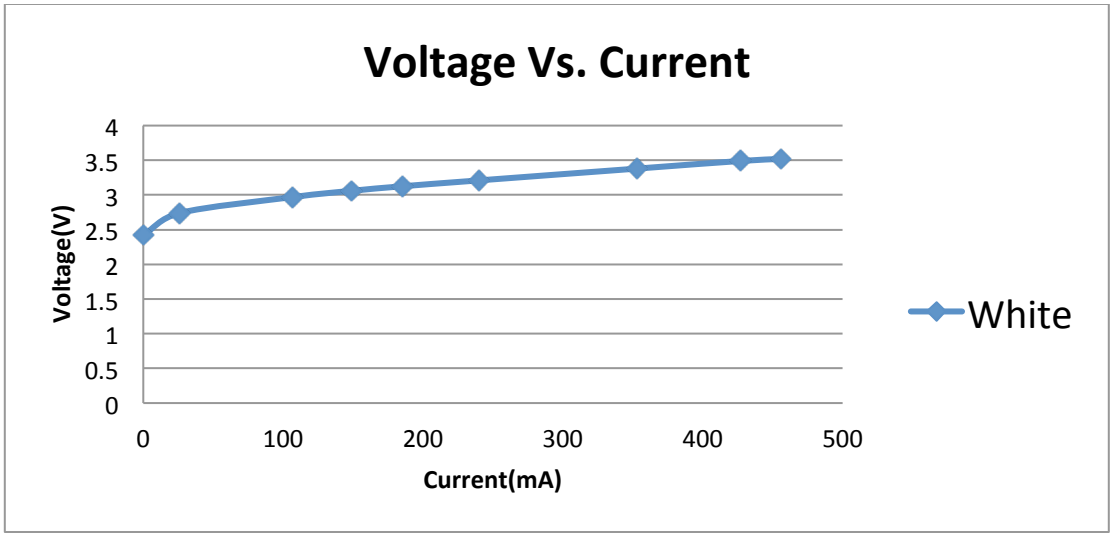
RGB:

Red LED		Green LED		Blue LED	
voltage (V)	current (mA)	voltage (V)	current (mA)	voltage (V)	Current (mA)
1.54	0.027	2.41	0.091	2.4	0.117
1.6	0.18	2.61	2.735	2.44	0.24
1.79	25	2.88	27	2.64	8
1.82	37	3.046	95	2.734	23
1.95	117	3.204	171	2.843	50
2.09	228	3.28	237	2.92	77
2.17	290	3.365	308	3.162	173
2.23	339	3.45	384	3.283	240
2.32	445	3.57	510	3.3	280



Warm/Cool White:

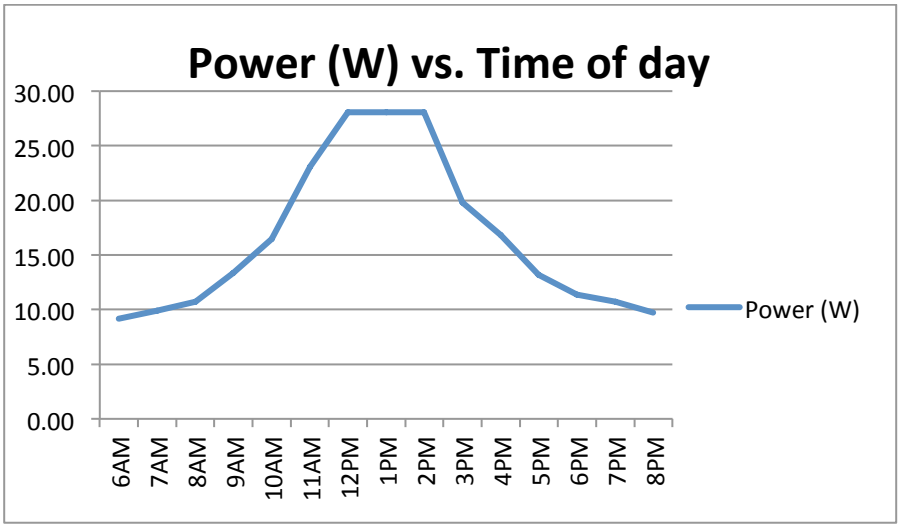
White LED voltage (V)	Current (mA)
2.42	0.07
2.74	26.1
2.97	107
3.06	149
3.125	185
3.21	240
3.38	353
3.49	427
3.52	456



Power consumption of Each LED at specific time period:

Time	Power(W)
6AM	9.17
7AM	9.90
8AM	10.70
9AM	13.37
10AM	16.50
11AM	23.10
12PM	28.05
1PM	28.05
2PM	28.05
3PM	19.80
4PM	16.83
5PM	13.20
6PM	11.39
7PM	10.73
8PM	9.74

Graph of above tables:



As can be seen as the day starts the power goes up and then decreases after reaching full cycle.
Power consumption of each individually produced light source.

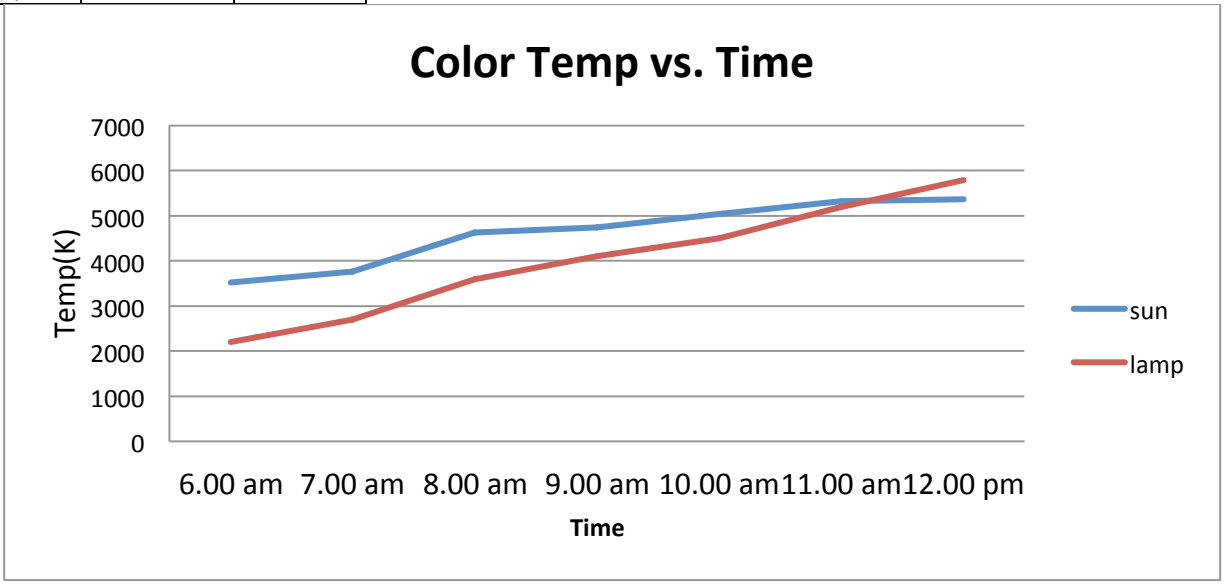
Light Mode	Power(W)
candle	7.75
warm white	4.95
incandescent	9.90
halogen	11.55
cool white	5.45
cloudy sky	17.16
day white	22.11
direct sunlight	17.66
blue sky	15.01
normal	39.60

Test 4: Verifying the functionality of the entire system:

Color temperature measured and compared with different times of the day. Lumens were measured and compared with standard 3-way CFL and incandescent light bulbs. We measured the color temperature of the lamp light output using an application on the IPhone.
We got these measurements for our design:
First we measured the color temperature of each of the light outputted by lamp from 6am to 8pm.

Time	Sunlight Color Temp(K)	Lamp Color Temp(K)
6.00 am	3523	2200
7.00 am	3764	2700
8.00 am	4629	3600
9.00 am	4750	4100
10.00 am	5037	4500
11.00 am	5324	5200
12.00 pm	5375	5800

Color Temperature of Certain Light produced	
Light Mode	Temp(K)
Candle light	2700
Cloudy Sky	4900
Warm White	3000
Cool White	4500
Incandescent	2700
Normal	5800
Blue Sky	7200



The sunlight and lamp light output have similar progression as can be shown by the graph. The comparison is that our lamp source is approximately equal to the output of the sunlight.

Second we compared the lumens output of Incandescent Light Bulbs and CFLs. We managed to get the lumens output of our lamp source to be equivalent to the once produced by a 150W Incandescent Light Bulb.

We compared our lamp source to other light sources.

Incandescent Light Bulbs	CFL	Lumens
40 W	9-13 W	450
60 W	13-15 W	800
75 W	18-25 W	1100
100 W	23-30 W	1600
150 W	30 - 55 W	2600

As can be seen by the below table we managed to get the same amount of intensity at a lower power output then incandescent light bulbs, and CFL.

Light Mode	Power(W)	Lumens
candle	7.75	918.04
warm white	4.95	987.03
incandescent	9.90	1193.99
halogen	11.55	1040.09
cool white	5.45	668.63
cloudy sky	17.16	1576.06
day white	22.11	2483.49
direct sunlight	17.66	1576.06
blue sky	15.01	1406.25
normal	39.60	2641.63

Experiment Validation:

Our evaluation criteria are defined by the design requirements specification which is listed below:

- The Nicely integrated into a torchiere lamp
- The user should be able to interact with the device and change the colors
- The brightness and color output of the LEDs must be independent of each other
- Generate a long-lasting light source that imitates sunlight using LEDs
- The system should be power efficient
- Design should be integrated with a microcontroller
- The lamp output should have approximately 2600 lumens at 100 percent brightness

The light source was nicely integrated into a torchiere lamp as seen below.

The user can interact with the device by a remote control and an LCD to display the menu. The colors can be changed by using the specific buttons on the remote.

We generated a long-lasting light source that imitates sunlight using LEDs and made a comparison to sunlight output, we have a backup power for the microcontroller to keep the time on continuously.
We measured the color temperature of our lamp from 6am to 12 pm and found that the color temperature is almost similar.

Time	Sunlight Color Temp(K)	Lamp Color Temp(K)
6.00 am	3523	2200
7.00 am	3764	2700
8.00 am	4629	3600
9.00 am	4750	4100
10.00 am	5037	4500
11.00 am	5324	5200
12.00 pm	5375	5800

The system is power efficient because the lamp can produce the same number of lumens at a lower power consumption as given by the below table.

Incandescent Light Bulbs	CFL	Lumens
40 W	9-13 W	450
60 W	13-15 W	800
75 W	18-25 W	1100
100 W	23-30 W	1600
150 W	30 - 55 W	2600

The lamp source can produce same lumens for lesser power consumption as shown in the below table.

Light Mode	Power(W)	Lumens
candle	7.75	918.04
warm white	4.95	987.03
incandescent	9.90	1193.99
halogen	11.55	1040.09
cool white	5.45	668.63
cloudy sky	17.16	1576.06
day white	22.11	2483.49
direct sunlight	17.66	1576.06
blue sky	15.01	1406.25
normal	39.60	2641.63

Every aspect of the design depends on a microcontroller. Each component of the project is integrated together by the microcontroller. The microcontroller handles user input and also controls the light output.

The lamp at full potential can produce at least 2600 lumens at lesser power consumption. As the time approaches noon the light source reaches its maximum light intensity at lesser power.

Time	Power(W)	Lumens
6AM	9.17	1156.84
7AM	9.90	1231.13
8AM	10.70	1273.58
9AM	13.37	1550.91
10AM	16.50	1805.05
11AM	23.10	2641.63
12PM	28.05	2828.42
1PM	28.05	2828.42
2PM	28.05	2828.42
3PM	19.80	1793.63
4PM	16.83	1576.06
5PM	13.20	1316.04
6PM	11.39	1156.84
7PM	10.73	1066.63
8PM	9.74	1040.09

In conclusion since our objective was to design a sunlight replicator using LEDs and we managed to do so, we can say that our project is somewhat successful. Our result is that we managed to meet all the requirements as specified by the customer. So we can say that based on the fact that our lamp is integrated into a torchiere and the user can interact with the device remotely. Also giving the data we have that proves that our system is power-efficient and approximately produces 2600 lumens at full potential. We can say that our project was successful but possible needs more improvement.

Project Issues

a. The reason for this project is to create a sunlight replicator light source. This will give the full effect of sunlight to people who may not spend enough time outside. People with sleep depression, learning issues due to lack of sleep, sleepiness during work hours and decrease productivity would benefit from this project. They would benefit by getting a simulated sunlight output that tells their body to sleep at a certain time. The benefit would be that the user can simulate sunlight to handle any symptoms caused by the lack of normal sunlight. The impact of success can be that this prototype can be eventually implemented as a working product in the future. This can help humanity by providing artificial light source that can help them be healthy.
b. This project can be used to reduce jet-lag on when flying on airplanes between different time zones. It can also be used for light therapy and sleep therapy. As well as in offices with no windows and underground transportation. The users can be anybody who works in offices with no window and companies that do business in areas with no direct sunlight provided. Anybody around the world can use this product to simulate sunlight because sunlight is same everywhere. It can be applied anywhere around the world where there is less exposure to direct sunlight. There are a lot of places around the world where there is less sunlight for a given amount of time.
c. The project in total cost about $636 dollars. The power supply used to provide the system power cost about $90 dollars. The two microcontrollers bought cost about $120 dollars together. The PCB design was about $100 dollars. We spent a significant amount of time coding each parts of our design and if each of us were to provide an hourly pay rate for the time spent coding it would be a significant amount of dollars. We spent a total of 659 hours of our time on designing and implementing the project, and based on the time we spent we estimate our cost of design and this first prototype to be about $1200 dollars roughly.
d. Since we had burned our first microcontroller our total cost would actually go down. An alternatives design would be to implement a design that is smaller and therefore costs less. Perhaps we can use a more power efficient power source. We can limit the power consumed by a community. So we can design a smaller light source use a cheaper microcontroller and design a cheaper power supply. This will decrease the power consumption and the cost of the finished product.
e. The final design can be maintained by keeping it clean and making sure that fragile components of the product are not damaged in anyway. It should be treated like a regular lamp dust it if it's dirty and make sure that you change the battery on the backup power for the microcontroller. The LEDs are guaranteed to last for more than eight years so the lamp can last for a long time as long it is kept usable and not physically damaged in anyways. Software has to be kept up to date to handle changes in programming languages and document an understanding of how it is programmed. If in the future the user requests a change in the capability of the design then knowledge of the software is required.
f. If when the project reaches the end of its "lifetime" it can be disposed according to the rules and regulation of each country/state. Each component can be taking apart and recycled to be used in other designs. The project can also be sold as artifact given the duration of time. Batteries need to be disposed according to the environmental means of disposal. Certain components can be used elsewhere depending on how it is required to be recycled. Also the product can be upgraded given that we may provide documentation that allows for replacement to the design.

Administrative Part

Project Progress

The project was broken down into seven major tasks and each task was assigned a time frame.

Each major task involved minor tasks as it is shown in the table below:

Task Name	Completion Rate	Weeks												
		1	2	3	4	5	6	7	8	9	10	11	12	13
1.1.Remote control access (3 WEEKS)														
a. IR integration with microcontroller	100%													
b. Remote compatibility with IR	100%													
c. Pulse reading using MCU software	100%													
d. Handling user interrupt	100%													
1.2.Power circuit selection (3 WEEKS)														
a. Generate power for components	100%													
b. Integrate with microcontroller	100%													
c. Circuit design	100%													
1.3.Measurement (3 WEEKS)														
a. Color temperature	100%													
b. Light intensity	100%													
c. Power consumption	100%													
d. Light spectrum comparison	100%													
1.4.System Development (7 WEEKS)														
a. Build control panel	100%													
b. Program LCD and test	100%													
c. Microcontroller programming	100%													
d. Create Bread board	100%													
e. Final circuit implemented on PCB	100%													
f. Integrate into lamp	100%													
1.5.LED light source development (7 WEEKS)														
a. Light different LEDs	100%													
b. Use MCU to control color output	100%													
c. Brightness control using PWM	100%													
d. Test using Microcontroller	100%													
1.6.System Integration (7 WEEKS)														
a. Power supply w/ microcontroller	100%													
b. Remote control w/ MCU & LCD	100%													
c. Keypad w/ MCU and LCD	0%													
d. Remote control w/ system	100%													
e. LEDs w/ MCU	100%													
f. All components w/ power switch	0%													
1.7.Testing														
a. Experiment #1*	100%													
b. Experiment #2*	100%													
c. Experiment #3*	100%													
d. Experiment #4*	100%													

*The Experiments are shown in the previous section.

Table 3. Task Allocation

All the tasks of the project were completed within the assigned time frame. As the table shows, there were two small changes to the design. The initial plan was to have a keypad to use as a backup for the remote

control. This was not part of the requirement of the project, hence there was not a time frame assigned to it. The team decision was to include a keypad only if there were any difficulties using the remote control, but since the remote control was working perfectly there was not any need to include a keypad. The second change was that we eliminated the power switch and instead we included an on and off option in the user interface menu. This was making it more convenient for the user to switch the lamp on and off just by using the remote without having to come close to the lamp.

Unexpected Activity

The only unexpected activity that was faced during this project was that the micro-controller was burnt during integrating the system into the lamp due to the main power supply wire touching one of the pins of the micro-controller. Fortunately, there was enough time for ordering the part and it did not cause any delays in the progress of the project.

Funds Spent

The table below shows a breakdown of the funds that were spent on the project and how much was spent by each team member. Since Marzia was the financial manager of the team, she was mainly responsible for ordering parts. Other team members bought only the components that they needed to complete their tasks. The total amount that we spent on the project was slightly higher than the maximum we were allowed to spend, which was $600 dollars. We could easily stay below this amount is we did not have to order some parts as quickly as possible and pay for expedited delivery.

Description	Reciept Date	Vendor/Supplier	Mayank	Maryam	Marzia	Girmay
MSP430 Microcontroller * 2	2/12/12	Texas Instruments	$ 8.60			
IAR Detector	3/8/12	Digikey				$ 8.00
IAR detector	3/8/12	RadioShack				$4.19
LEDs x 6	2/25/12	HERO-LED			$ 15.33	
10 RGB LEDS	3/28/12	Super Bright Leds			$ 50.49	
1 x MSP430F5438 Header Board	9/9/12	microcontrollershop.com	$ 32.14			
10 RGB LEDS	9/9/12	Super Bright Leds	$ 50.49			
Power Supply	9/12/12	Powergate LLC		$ 80.87		
PWM	9/20/12	GMU		$ 6.40		
Remote Control	9/2/12	Amazon				$ 6.00
White LEDs	9/12/12	HERO-LED			$ 90.00	
Plug, tape, and connectors	9/28/12	Home Depot		$ 7.49		
Voltage Regulator & Heat Sink	10/15/12	eBay		$ 1.98		
Lamp	10/23/12	Target			$ 30.00	
Heatsink	11/1/12	ebay		$ 16.99		
PCB	11/13/12	ExpressPCB			$ 97.56	
IAR Detector	11/20/12	RadioShack				$ 5.25
Header Pins	11/15/12	Mason	$ 0.80			
Heatsink Com.	11/17/12	RadioShack	$ 16.85			
MSP430 Microcontroller	11/24/12	microcontrollershop.com	$ 71.26			
Header Pins	11/17/12	microcenter	$ 10.49			
Pizza	11/17/12	Domino		$ 18.22		
iPhone Apps	11/20/12	Apple Store			$ 5.98	
	Total Money Spent by each member:		$ 190.63	$ 131.95	$ 289.36	$ 23.44
Grand Total	**$**	**635.38**				
member owes:	**$**	**158.85**				
	Amount to/from each member		$ 31.79	$ (26.90)	$ 130.52	$ (135.41)

Table 4. Financial Sheet

Man Hours

All the team members spent almost equal amount of time on the project. All four team members were always present during the team meetings and lab hours. Most of the project work was done during weekends when all members where present and each member worked on their own task. We spent less time during the beginning of the semester since we were mainly ordering parts and waiting for them to arrive but as we got closer to the end of the semester we invested more time for the project since the final testing and assembly was very time consuming. Below is a table showing the amount of time each member spent on the project:

Week	Week Ending	Maryam		Mayank		Girmay		Marzia	
		Educational / Learning	Productive Effort	Educational / Learning	Productive Effort	Educational / Learning	Productive Effort	Educational / Learning	Productive Effort
1	31-Aug-12	3.0		2.0	2.0	2.0	1.0	2.0	5.0
2	7-Sep-12	1.0	2.5	1.0	3.0		2.0	1.0	4.0
3	14-Sep-12		2.0	0.5	4.0	1.0	3.0	1.0	2.0
4	21-Sep-12	2.0			4.0		3.0	1.5	
5	28-Sep-12	2.0	5.0	0.5	4.0	0.8	3.0		1.0
6	5-Oct-12	3.0	1.0		9.0		8.0	2.0	2.5
7	12-Oct-12	2.0	2.0		8.0		9.0	1.0	2.0
8	19-Oct-12	2.0	8.0		12.0		11.0	3.0	4.0
9	26-Oct-12	1.0	6.0		8.0	2.0	10.0	4.0	5.0
10	2-Nov-12		16.0		15.0		14.0	3.0	14.0
11	9-Nov-12		25.0		22.0		22.0		24.0
12	16-Nov-12		27.0		28.0		26.0		26.0
13	23-Nov-12		30.0		30.0	4.0	27.0	5.0	29.0
14	30-Nov-12		30.5		30.5		30.5		22.0
15	7-Dec-12								
	Time Spent:	16.0	155.0	4.0	179.5	9.8	169.5	23.5	140.5
	Total Time	171.0		183.5		179.3		164.0	

Table 5. Hours Spent

Lessons Learned

The benefit of being a student is that every day, there is something new to learn that will be useful somewhere down the road. However, one might never think why learning this information is important until the moment he or she has to use it. Thus, the senior design projects are the best way for the students to bring their theoretical knowledge into practice and see how this knowledge can be used in real world.

Our learning process started from the first day we picked up our project. The first step was to learn about different types of LEDs in order to fine the right match for our design. Then, we had to choose the right micro controller for our design. Next, we had to study about how to control the LEDs using a micro controller and learned about how to use an H-bridge. We had to do many different initial testing to see how different components of our project works and based on our findings we were able to design the entire system. One area that none of us had much experience with was reading the color temperature of different light spectrums and comparing them with each other. As a result we had to get out of the technical mode and think more like an artist. Last but not least, we became an expert in soldering and designing PCBs.

Beside the technical part of the project, we learned about teamwork and how to function as a part of a team. Each team member was responsible for a part of the project and all the tasks were dependent on each other. Therefore, in some cases we had to help each other out to finish the tasks so we can move to the next step.

Here are some of the highlights of what we experienced during our project:

1. Always have a backup of all components. It is a very good practice to buy two or more of each component used in the project. This way if one of the components breaks for any reason the project does not have to be stopped until the part arrives.
2. Stay focused at all time. There are always tasks that are easier or faster to finish. Therefore, it is the best to get these tasks out of the way so you have more time to work on the other more complex tasks. Never get too excited if a part of the project works since this might not be the case for the rest of the parts.
3. Always keep extra time for testing. Testing is one of the most important parts (if not the most important one) of the project. It is very common to come across issues during testing that were not addressed before and that is the whole purpose of testing. Therefore, try to at least leave two to three weeks' time for testing.
4. If the senior design project is taken in such a way that summer falls between the two semesters make sure you take advantage of your time during summer. Having a good action plan will defiantly work.

Source Code

main_function.c

```
#include "msp430f5438.h"
#include <stdio.h>
#include <stdlib.h>
#include <intrinsics.h>
#include "LED_code.h"
#include "LCD_code.h"

static void doDemoMode();
//*****************************************************************************
//Function: Main function
//Description: main loop
//*****************************************************************************
void main(void) {
	// Stop watchdog timer to prevent time out reset

	WDTCTL = WDTPW + WDTHOLD;
	P5DIR &= ~BIT4;

	__enable_interrupt(); //Enable global interrupt
	hbridgesPort_initRGB(1); //enabling all LEDs
	hbridgesPort_initWarmWhite(1); //initializing ports for warm white but disabling them
	hbridgesPort_initCoolWhite(1); //initializing ports for cool white but disabling them
	timerPorts_init();
	timer_init();
	irandLcd_init();
	customWhiteMaker(125, 125);
	while (1) {
	  if(ValueisSet /*&& ((P5IN & BIT4)==BIT4)*/) //if on and value set
	  {
		doUserInput();
	  }
	  if ((P5IN & BIT4)!=BIT4)//off turn lcd off
	  {
	P2OUT &= ~BIT7; //turn lcd off
	  }
	  else //on
	  {
	if ((P2OUT & BIT7)!= BIT7) //turn lcd on if off
	{
		P2OUT |= BIT7;
	}
	  }
	}
}

#pragma vector = TIMER1_A1_VECTOR		//Basic Timer interrupt to handle debouncing
__interrupt void timerVector() {
	TA1CTL &= ~TAIFG;
	holdDemoValue += 1;
	cursorPos(2,5);
	char temp[10];
	sprintf(temp, "t%d", holdDemoValue);
	writeString(temp);
	doDemoMode();
}
static void doDemoMode() {
	unsigned int g = 90;
	unsigned int b = 1;
	unsigned int v1 = 0;
```

```
unsigned int v2 = 0;
unsigned int v3 = 0;
unsigned int w = 4;

if (isDemoMode == 1)
{
	if (holdDemoValue > 0 && holdDemoValue <= 100)
	{
		g = 90 + (unsigned int)(((double)(holdDemoValue))* ((double)(0.59)));
		b = 1 + (unsigned int)(((double)(holdDemoValue))* ((double)(0.12)));;
	}
	else if (holdDemoValue > 100 && holdDemoValue <= 210) {
		g = 149 + (unsigned int)(((double)(holdDemoValue - 100))* ((double)(0.54)));
		b = 13 + (unsigned int)(((double)(holdDemoValue - 100)) * ((double)(1.63)));
	} else if (holdDemoValue > 210 && holdDemoValue <= 267) {
		g = 209;
		b = 193;
		w = 4 + (unsigned int)(((double)(holdDemoValue - 210)) * ((double)(4.40)));
		customWhiteMaker (w , w);
	} else if (holdDemoValue > 267 && holdDemoValue <= 323) {
		g = 209;
		b = 193;
		w = 255;
		customWhiteMaker(w, w);
	} else if (holdDemoValue > 323 && holdDemoValue <= 378) {
		g = 209;
		b = 193;
		w = 255 - (unsigned int)(((double)(holdDemoValue - 323)) * ((double)(4.40)));
		customWhiteMaker(w, w);
	}
	else if (holdDemoValue > 378 && holdDemoValue <= 489)
	{
		customWhiteMaker (0 , 0);
		g = 209 - (unsigned int)(((double)(holdDemoValue - 378)) * ((double)(0.54)));
		b = 193 - (unsigned int)(((double)(holdDemoValue - 378)) * ((double)(1.63)));
	}
	else if (holdDemoValue > 489 && holdDemoValue <= 590) {
		g = 149 - (unsigned int)(((double)(holdDemoValue - 489)) * ((double)(0.58)));
		b = 13 - (unsigned int)(((double)(holdDemoValue - 489)) * ((double)(0.12)));
	} else {
		isDemoMode = 0;
		turnOff();
		holdDemoValue = 0;
		TA1CTL = TASSEL_2 | MC_2 | TACLR; // SMCLK, continuous mode
		TA1CCTL0 = CM_2 | CCIS_1 | CAP | CCIE; // falling edge capture mode, CCI0A, enable ISR
		return;
	}
	v1 = (unsigned int) (((double) 225 / (double) 255) * pulseRate);
	v2 = (unsigned int) (((double) g / (double) 255) * pulseRate);
	v3 = (unsigned int) (((double) b / (double) 255) * pulseRate);
	customColorMaker(v1, v2, v3);
}
else if (isDemoMode == 2)
{
	if (holdDemoValue > 0 && holdDemoValue <= 3)
	{
		v1 = (unsigned int) (((double) 210 / (double) 255) * pulseRate);
		v2 = (unsigned int) (((double) 100 / (double) 255) * pulseRate);
		v3 = (unsigned int) (((double) 5 / (double) 255) * pulseRate);
		customWhiteMaker(0 , 0);
		customColorMaker(v1, v2, v3);
	}
```

```
else if (holdDemoValue > 3 && holdDemoValue <= 6)
{
        customColorMaker(0, 0, 0);
        customWhiteMaker(255, 0);
}
else if (holdDemoValue > 6 && holdDemoValue <= 9)
{
        v1 = (unsigned int) (((double) 235 / (double) 255) * pulseRate);
        v2 = (unsigned int) (((double) 140 / (double) 255) * pulseRate);
        v3 = (unsigned int) (((double) 10 / (double) 255) * pulseRate);
        customColorMaker(v1, v2, v3);
        customWhiteMaker(0, 0);
}
else if (holdDemoValue > 9 && holdDemoValue <= 12)
{
        v1 = (unsigned int) (((double) 235 / (double) 255) * pulseRate);
        v2 = (unsigned int) (((double) 200 / (double) 255) * pulseRate);
        v3 = (unsigned int) (((double) 30/ (double) 255) * pulseRate);
        customColorMaker(v1, v2, v3);
        customWhiteMaker(0, 0);
}
else if (holdDemoValue > 12 && holdDemoValue <= 15)
{
        customColorMaker(0, 0, 0);
        customWhiteMaker(0, 255);
}
else if (holdDemoValue > 15 && holdDemoValue <= 18)
{
        v1 = (unsigned int) (((double) 210 / (double) 255) * pulseRate);
        v2 = (unsigned int) (((double) 215 / (double) 255) * pulseRate);
        v3 = (unsigned int) (((double) 150 / (double) 255) * pulseRate);
        customColorMaker(v1, v2, v3);
        customWhiteMaker(0, 0);

}
else if (holdDemoValue > 18 && holdDemoValue <= 21)
{
        v1 = (unsigned int) (((double) 225/ (double) 255) * pulseRate);
        v2 = (unsigned int) (((double)209 / (double) 255) * pulseRate);
        v3 = (unsigned int) (((double) 193/ (double) 255) * pulseRate);
        customColorMaker(v1, v2, v3);
        customWhiteMaker(100, 100);
}
else if (holdDemoValue > 21 && holdDemoValue <= 24)
{
        v1 = (unsigned int) (((double) 200 / (double) 255) * pulseRate);
        v2 = (unsigned int) (((double) 215 / (double) 255) * pulseRate);
        v3 = (unsigned int) (((double) 150 / (double) 255) * pulseRate);
        customColorMaker(v1, v2, v3);
        customWhiteMaker(0, 0);
}
else if (holdDemoValue > 24 && holdDemoValue <= 27)
{
        v1 = (unsigned int) (((double) 64 / (double) 255) * pulseRate);
        v2 = (unsigned int) (((double) 145 / (double) 255) * pulseRate);
        v3 = (unsigned int) (((double) 235 / (double) 255) * pulseRate);
        customColorMaker(v1, v2, v3);
        customWhiteMaker(0, 0);
}
else
{
        isDemoMode = 0;
```

```
                                turnOff();
                                holdDemoValue = 0;
                                TA1CTL = TASSEL_2 | MC_2 | TACLR; // SMCLK, continuous mode
                                TA1CCTL0 = CM_2 | CCIS_1 | CAP | CCIE; // falling edge capture mode, CCI0A, enable ISR
                        }
                }
        }
```

LCD_code.h

```
#include <stdio.h>
#include "msp430f5438.h"

#define duration   600
#define start_ms   duration*4    //start of 2.4ms
#define bit1_ms   duration*3  //bit 1 dur. 1.8ms
#define bit0_ms   duration*2  //bit 0 dur. 1.2ms

//LCD Defenations
#define EN BIT0
#define R_S BIT7
#define R_W BIT1
#define L2_ADDR 0x40
#define L3_ADDR 0x14
#define L4_ADDR 0x54
#define D7MASK 0x80

#define INVALID          0x00
#define WELCOMESCREEN     0x01
#define MAINMENU                  0x02
#define LAMPMODESELECTION 0x03
#define TIMEMODESELECTION 0x04
#define SETTINGS                0x05
#define AUTOMATICMODE     0x06
#define MANUALMODE        0x07
#define NORMALMODE        0x08
#define AMPMSCREEN        0x09
#define MILITARYSCREEN    0x0A
#define RESETSCREEN       0x0B
#define BRIGHTNESSSCREEN  0x0C
#define CONFIRMSCREEN     0x0D
#define MANUALMODEPAGE1   0x0E
#define MANUALMODEPAGE2   0x0F
#define MILITARYTIMEENTER 0x10;
#define DEMOMODE                  0x11;

extern int ValueisSet;
extern int valueRead;
//LCD Initializations
void irandLcd_init();
static void init();
static void MenuDisplay(char num1);
static void command( char i );
static void write (char i);
void writeString (char *s);
static void LcdReset();
static char DDRAMAddr(char row, char col);
void cursorPos(char row, char col);
static void realTimeClock_init();
static void main_menu();
static void Bright(char vol);
static void special_char(char c);
```

```
static void Lamp_Mode();
static void Time_mode();
static void settings();
static void manualMode();
static void manualModePage1();
static void manualModePage2();
static void militaryTimeEnter(char num);
static void AM_PM(char number);
static void goBack(char * temp);
static void welcome_screen();
void doUserInput();
static void displayTime();
```

LCD_code.c

```
#include "LCD_code.h"
#include <intrinsics.h>
#include "LED_Code.h"
#include <string.h>

static unsigned int IRData = 0; // received signal
static unsigned int pwmCounter = 1; //count pulse input fromremote
static unsigned int start = 0; //control button use var.
static int func = 0;
static char count_out = 10;
int valueRead = 0;
int ValueisSet = 0;
static unsigned int count = 0;
static unsigned int enter_time = 1;
static unsigned int hourInput = 0x06;
static unsigned int minInput = 0x45;
static unsigned int hourIn = 0x06;
static unsigned int minIn = 0x45;
static unsigned int isMilitary = 1;
static unsigned int isAMPM = 0;
static unsigned int adjust=0;
static unsigned int how_brig=0;
static char lights[6];
static char time[5];

void irandLcd_init() {
        P8DIR &= ~BIT5; //Timer Input
        P8SEL |= BIT5;

        P5OUT &= ~(BIT7); //FOR LEDs IR
        P7OUT &= ~(BIT2);
        P5DIR |= (BIT7);
        P7DIR |= (BIT2);

        P3DIR = 0xFF; //LCD
        P5DIR |= (R_W | EN); //LCD controller

        P7DIR |= (R_S);

        P2DIR |= BIT7; //for vcc for LCD
        P2OUT |= BIT7;

        strcpy(time, "AM");

        init(); // LCD Initialization
        realTimeClock_init();
        welcome_screen();
```

```
	func = WELCOMESCREEN;

	TA1CTL = TASSEL_2 | MC_2 | TACLR; // SMCLK, continuous mode
	TA1CCTL0 = CM_2 | CCIS_1 | CAP | CCIE; // falling edge capture mode, CCI0A, enable ISR
}

void doUserInput() {
	switch (IRData) {
	case 512:
		count_out = '1';
		if (start == 1) {
			P5OUT ^= BIT7;
		} //turn led on
		break;
	case 516: // num 2
		count_out = '2';
		if (start == 1) {
			P5OUT ^= BIT7;
		}
		break;
	case 520: // num 3
		count_out = '3';
		if (start == 1) {
			P5OUT ^= BIT7;
		}
		break;
	case 524: // NUM 4
		count_out = '4';
		if (start == 1) {
			P5OUT ^= BIT7;
		}
		break;
	case 528: // NUM 5
		count_out = '5';
		if (start == 1) {
			P5OUT ^= BIT7;
		}
		break;
	case 532: // num 6
		count_out = '6';
		if (start == 1) {
			P5OUT ^= BIT7;
		}
		break;
	case 536: // num 7
		count_out = '7';
		if (start == 1) {
			P5OUT ^= BIT7;
		}
		break;
	case 540: // num 8
		count_out = '8';
		if (start == 1) {
			P5OUT ^= BIT7;
		}
		break;
	case 544: // num 9
		count_out = '9';
		if (start == 1) {
			P5OUT ^= BIT7;
		}
		break;
```

```
        case 548: // num 0
                count_out = '0';
                if (start == 1) {
                        P5OUT ^= BIT7;
                }
                break;
        case 584: // vol +
                count_out = '#';
                if (start == 1) {
                        P5OUT ^= BIT7;
                }
                break;
        case 588: // vol -
                count_out = '*';
                if (start == 1) {
                        P5OUT ^= BIT7;
                }
                break;
        case 576: // CH +
                count_out = '+';
                if (start == 1) {
                        P5OUT ^= BIT7;
                }
                break;
        case 580: // CH -
                count_out = '-';
                if (start == 1) {
                        P5OUT ^= BIT7;
                }
                break;
        case 596: //   Power
                P7OUT ^= BIT2;
                start ^= 1;
                if (start == 0)
                        P5OUT &= ~BIT7;
                break;
        case 748:
                count_out = 'R'; //Recall
                break;
        case 556:
           count_out = 'E'; //Enter
           break;
        }
        MenuDisplay(count_out);
        IRData = 0;
        int k = 0;
        for (k = 0; k < 20000; k++)
                ;
        __delay_cycles(100000);
        ValueisSet = 0;
}

#pragma vector = TIMER1_A0_VECTOR
__interrupt void Timer_A(void) {
        if (ValueisSet == 0) {
                if ((TA1R > 1500 && TA1R < 2000) && pwmCounter > 2)
                        IRData |= pwmCounter;
                pwmCounter <<= 1;
                if (pwmCounter == 0x2000) {
                        ValueisSet = 1;
                        pwmCounter = 1;
                }
```

```
                TA1CTL |= TACLR;
        }
}
//new rtccode
static void realTimeClock_init() {
        RTCCTL01 = RTCBCD | RTCMODE | RTCTEVIE;

        //All these values define seconds,minutes, hours, day of week, day, month and year so you can use any values for now

        RTCMIN = 0x00;
        //RTCSEC = 0x01;
        RTCHOUR = 0x06&0x3F;
        RTCDOW = 0x00;
}
static void LcdReset() {
        command(0x01); //clear display
        command(0x02); //return to home
        command(0x38); //function set as 8-bit length|two-line display|5x8 character font
}
static void init() {
        P5OUT &= ~(R_W | EN);
        P7OUT &= ~(R_S);
        // start LCD
        unsigned int x = 0;
        for (x = 0; x < 9000; x++)
                ;
        command(0x30);
        for (x = 0; x < 800; x++)
                ;
        command(0x30);
        for (x = 0; x < 400; x++)
                ;
        command(0x30);
        for (x = 0; x < 60600; x++)
                ;

        LcdReset();
        command(0x0F); // Display on | cursor on | Cursor Blinks ON
        command(0x06); // increment cursor after write
        command(0x38); // function set: 8-bit length|two-line display|5x8 character font
}
static void MenuDisplay(char num1) {
        if (num1 == 'R') {
                func = MAINMENU;
                main_menu();
                num1 = 10;
        } else if (func == WELCOMESCREEN) {
                if (num1 != 10) { //check if any input pressed
                        func = MAINMENU;
                        main_menu();
                        num1 = 10;
                }
        } else if (func == MAINMENU) { //main menu display
                                                                        //main_menu();
                if (num1 == '1') { //lamp_mode
                        func = LAMPMODESELECTION;
                        Lamp_Mode();
                        num1 = 10;
                } else if (num1 == '2') { //time_mode
                        func = TIMEMODESELECTION;
                        Time_mode();
                        num1 = 10;
```

```
        } else if (num1 == '3') { //reset
                //func=0;
                customColorMaker(0,0,0);
                customWhiteMaker(0,0);
                PMMCTL0 |= PMMSWPOR;
                num1 = 10;
        } else if (num1 == '4') { //settings menu
                func = SETTINGS;
                settings();
                num1 = 10;
        }
} else if (func == LAMPMODESELECTION) { //lamp_mode display
        if (num1 == '1') {
                func = AUTOMATICMODE;
                if (!isAutomaticMode)
                {
                        isAutomaticMode = 1;
                        doAutomaticMode();
                }
                LcdReset();
                displayTime();
                cursorPos(2, 0);
                writeString("Automatic mode");
                cursorPos(3, 0);
                writeString("Set");
                num1 = 10;
        } else if (num1 == '2') {
                func = MANUALMODE;
                isAutomaticMode = 0;
                manualMode();
                num1 = 10;
        } else if (num1 == '3') {
                func = NORMALMODE;
                customColorMaker(235, 200, 30);
                customWhiteMaker(125, 125);
                isAutomaticMode = 0;
                LcdReset();
                displayTime();
                cursorPos(2, 0);
                writeString("Normal Mode");
                cursorPos(3, 0);
                writeString("Coming Soon");
                num1 = 10;
        } else if (num1 == '4') {
                func = MAINMENU;
                turnOff();
                main_menu();
                num1 = 10;
        }
} else if (func == TIMEMODESELECTION) { //time_mode
        if (num1 == '1') {
                AM_PM(num1);
                func = AMPMSCREEN;
                num1 = 10;
        } else if (num1 == '2') {
                militaryTimeEnter(num1);
                func = MILITARYSCREEN;
                num1 = 10;
        } else if (num1 == '3') {
                func = MAINMENU;
                num1 = 10;
                main_menu();
```

```
		}
	} else if (func == SETTINGS) { //settings
		if (num1 == '1') {
			func = RESETSCREEN;
			LcdReset();
			cursorPos(2, 0);
			writeString("Reset");
			cursorPos(3, 0);
			writeString("Coming Soon");
			num1 = 10;
		} else if (num1 == '2') {
			Bright(num1);
			func = BRIGHTNESSSCREEN;
			num1 = 10;
		} else if (num1 == '3') {
			func = MAINMENU;
			main_menu();
			num1 = 10;
		}
	}
	// automatic mode
	else if (func == AUTOMATICMODE) {
	}
	//manual mode
	else if (func == MANUALMODE) {
		if (num1 == '1') {
			func = MANUALMODEPAGE1;
			manualModePage1();
		} else if (num1 == '2') {
			func = MANUALMODEPAGE2;
			manualModePage2();
		}
	}
	//normal mode
	else if (func == NORMALMODE) {
	}
	//military time
	else if (func == MILITARYSCREEN) {
		militaryTimeEnter(num1);
	}
	//AM PM Screen
	else if (func == AMPMSCREEN) {
		AM_PM(num1);
	}
	//reset
	else if (func == RESETSCREEN) {
	}
	//brightness adjust
	else if (func == BRIGHTNESSSCREEN) {
		Bright(num1);
	} else if (func == MANUALMODEPAGE1) {
		unsigned int val1 = 0;
		unsigned int val2 = 0;
		unsigned int val3 = 0;
		char temp[20];
		strcpy(temp, "INVALID");
		if (num1 == '1') {
			strcpy(temp, "Candle");
			val1 = (unsigned int) (((double) 210 / (double) 255) * pulseRate);
			val2 = (unsigned int) (((double) 100 / (double) 255) * pulseRate);
			val3 = (unsigned int) (((double) 5 / (double) 255) * pulseRate);
			customWhiteMaker(0 , 0);
```

```
                customColorMaker(val1, val2, val3);
            } else if (num1 == '2') {
                strcpy(temp, "WarmWhite");
//                val1 = (unsigned int) (((double) 255 / (double) 255) * pulseRate);
//                val2 = (unsigned int) (((double) 197 / (double) 255) * pulseRate);
//                val3 = (unsigned int) (((double) 143 / (double) 255) * pulseRate);
                customColorMaker(0, 0, 0);
                customWhiteMaker(255, 0);
            } else if (num1 == '3') {
                strcpy(temp, "Incandescent");
                val1 = (unsigned int) (((double) 235 / (double) 255) * pulseRate);
                val2 = (unsigned int) (((double) 140 / (double) 255) * pulseRate);
                val3 = (unsigned int) (((double) 10 / (double) 255) * pulseRate);
                customColorMaker(val1, val2, val3);
                customWhiteMaker(0, 0);
            } else if (num1 == '4') {
                strcpy(temp, "Halogen");
                val1 = (unsigned int) (((double) 235 / (double) 255) * pulseRate);
                val2 = (unsigned int) (((double) 200 / (double) 255) * pulseRate);
                val3 = (unsigned int) (((double) 30/ (double) 255) * pulseRate);
                customColorMaker(val1, val2, val3);
                customWhiteMaker(0, 0);
            } else if (num1 == '5') {
                strcpy(temp, "CoolWhite");
//                val1 = (unsigned int) (((double) 201 / (double) 255) * pulseRate);
//                val2 = (unsigned int) (((double) 226 / (double) 255) * pulseRate);
//                val3 = (unsigned int) (((double) 255 / (double) 255) * pulseRate);
                customColorMaker(0, 0, 0);
                customWhiteMaker(0, 255);
            }
            else if (num1 == '0'){
                strcpy(temp, "Demo");
                TA1CCR0 = 32000;
                TA1CCTL0 = CCIE;
                TA1CTL = TASSEL_1 + MC_1 + TAIE + TACLR;
                isDemoMode = 2;
                holdDemoValue = 0;
            }
            goBack(temp);
        } else if (func == MANUALMODEPAGE2) {
            unsigned int val1 = 0;
            unsigned int val2 = 0;
            unsigned int val3 = 0;
            char temp[20];
            strcpy(temp, "INVALID");
            if (num1 == '6') {
                strcpy(temp, "Cloudy Sky");
                val1 = (unsigned int) (((double) 210 / (double) 255) * pulseRate);
                val2 = (unsigned int) (((double) 215 / (double) 255) * pulseRate);
                val3 = (unsigned int) (((double) 150 / (double) 255) * pulseRate);
                customColorMaker(val1, val2, val3);
                customWhiteMaker(0, 0);
            } else if (num1 == '7') {
                strcpy(temp, "Day White");
                val1 = (unsigned int) (((double) 225/ (double) 255) * pulseRate);
                val2 = (unsigned int) (((double)209 / (double) 255) * pulseRate);
                val3 = (unsigned int) (((double) 193/ (double) 255) * pulseRate);
                customColorMaker(val1, val2, val3);
                customWhiteMaker(100, 100);
            } else if (num1 == '8') {
                strcpy(temp, "Direct Sunlight");
                val1 = (unsigned int) (((double) 200 / (double) 255) * pulseRate);
```

```
                        val2 = (unsigned int) (((double) 215 / (double) 255) * pulseRate);
                        val3 = (unsigned int) (((double) 150 / (double) 255) * pulseRate);
                        customColorMaker(val1, val2, val3);
                        customWhiteMaker(0, 0);
                } else if (num1 == '9') {
                        strcpy(temp, "Blue Sky");
                        val1 = (unsigned int) (((double) 64 / (double) 255) * pulseRate);
                        val2 = (unsigned int) (((double) 145 / (double) 255) * pulseRate);
                        val3 = (unsigned int) (((double) 235 / (double) 255) * pulseRate);
                        customColorMaker(val1, val2, val3);
                        customWhiteMaker(0, 0);
                }else if (num1 == '0') {
                   strcpy(temp, "Demo Mode");
                        if (!isDemoMode)
                        {
                         isDemoMode = 1;
                         RTCSEC = 0x01;
                         holdDemoValue = 1;
                         val1 = (unsigned int) (((double) 225 / (double) 255) * pulseRate);
                         val2 = (unsigned int) (((double) 90 / (double) 255) * pulseRate);
                         val3 = (unsigned int) (((double) 1 / (double) 255) * pulseRate);
                         customColorMaker(val1, val2, val3);
                         customWhiteMaker(0, 0);
                         TA1CCR0 = 32000;
                         TA1CCTL0 = CCIE;
                         TA1CTL = TASSEL_1 + MC_1 + TAIE + TACLR;
                         addGreen = 0;
                         addBlue = 0;
                        }
                }
                goBack(temp);
        }
}
//output welcome screen
static void welcome_screen() {
        LcdReset();
        displayTime();
        cursorPos(2, 0); //row 2
        writeString("Welcome!"); //second line
        cursorPos(3, 0); //row 3
        writeString("Press any key to"); // third line
        cursorPos(4, 0); // row 4
        writeString("continue.."); // fourth line
        command(0x0C);

}
//lamp mode
static void Lamp_Mode() {
        LcdReset();
        // 3. Mode
        displayTime();
        cursorPos(2, 0);
        writeString("Lamp Mode");
        cursorPos(3, 0); //row 2
        writeString("1.Automatic 2.Manual"); //second line
        cursorPos(4, 0); //row 3
        writeString("3.Normal  4.TurnOff"); // third line
        command(0x0C);
}
//time mode
static void Time_mode() {
        LcdReset();
```

```
		//  // 4. Time Mode
		displayTime();
		cursorPos(2, 0);
		writeString("Time Mode");
		cursorPos(3, 0); //row 3
		writeString("1.AM/PM 2.Military"); // third line
		cursorPos(4, 0); // row 4
		writeString("3.Return");
		command(0x0C);
	}
	//settings control
	static void settings() {
		LcdReset();
		//5.Settings
		displayTime();
		cursorPos(2, 0);
		writeString("Settings ");
		cursorPos(3, 0); //row 2
		writeString("1.Reset 2.Brightness"); //second line
		cursorPos(4, 0); //row 3
		writeString("3.Return            "); // third line
		command(0x0C);
	}
	//handle main menu
	static void main_menu() {
		LcdReset();
		// 2. Main Menu
		displayTime();
		cursorPos(2, 0); //row 2
		writeString("Main Menu"); //second line
		cursorPos(3, 0); //row 3
		writeString("1.Lamp  2.Time Mode"); // third line
		cursorPos(4, 0); // row 4
		writeString("3.Reset 4. Settings"); // fourth line
		command(0x0C);

	}
	static void manualMode() {
		LcdReset();
		displayTime();
		cursorPos(2, 0);
		writeString("Manual Mode");
		cursorPos(3, 0);
		writeString("1. Page 1");
		cursorPos(4, 0);
		writeString("2. Page 2");
		command(0x0C);
	}
	static void manualModePage1() {
		LcdReset();
		displayTime();
		cursorPos(2, 0);
		writeString("1.Candle  4.Halogen");
		cursorPos(3, 0);
		writeString("2.W White 5.C White");
		cursorPos(4, 0);
		writeString("3.Incandescent 0.Dem");
	}
	static void manualModePage2() {
		LcdReset();
		displayTime();
		cursorPos(2, 0);
```

```
            writeString("6.Cloudy  9.BlueSky");
            cursorPos(3, 0);
            writeString("7.DayWhite 0.Demo");
            cursorPos(4, 0);
            writeString("8.DirectSunlight");
}
static void Bright(char vol){
   int change=5;
   if (func != BRIGHTNESSSCREEN)
            {
        LcdReset();
        displayTime();
        cursorPos(2, 0);
        writeString("Brightness Control");
        unsigned int i;
        //hold previous values of brightness
        for(i=0;i<adjust;i++){
          cursorPos(3, i);
          special_char(0xFF);
        }
        //reprint how bright it is
        how_brig=adjust*change;
        cursorPos(4,0);
        sprintf(lights,"%d%% Bright ",how_brig);
        writeString(lights);
           }
      else
      {
        if(vol== '#'&& adjust <= 19){
          cursorPos(3,adjust);
          special_char(0xFF);
          adjust++;
          how_brig=adjust*change;
          cursorPos(4,0);
          sprintf(lights,"%d%% Bright ",how_brig);
          writeString(lights);
          if(adjust == 20){
            adjust=19;
          }
        }
        else if(vol=='*' && adjust <=19){
          cursorPos(3,adjust);
          special_char(0x20);
          adjust--;
          how_brig=(adjust+1)*change;
          cursorPos(4,0);
          sprintf(lights,"%d%% Bright ",how_brig);
          writeString(lights);
        }
        if(adjust== 65535)
          adjust=0;
      }
}
//special char dispaly on LCD
static void special_char(char c){
  P3OUT = c;                //0xE8 outputted
  P7OUT |= R_S;       //set RS high data being passed
  P5OUT |= EN;       //pass data
  unsigned int x;
  for (x = 0; x < 100; x++);
  P5OUT &=~ EN;   //disabe data/command passage
  P7OUT &= ~R_S;    //cleared for command
```

```
}
static void AM_PM(char num){

        int number = num-'0';

        if (func != AMPMSCREEN)
        {
                LcdReset();
                displayTime();
                cursorPos(2, 0);
                writeString("AM PM TIME ENTER");
    cursorPos(3,7);
    writeString(time);
                cursorPos(3, 0);
                writeString(" __:__ ");
                cursorPos(3, 1);
                command(0x0F);
        }
        else
        {
                if(enter_time == 1){
                        if(number >=0 && number <= 1)
                        {
        write(num);
        hourIn = ((num - '0') << 4);
        enter_time++;
        cursorPos(3, enter_time);
        num=10;
      }
     }
      else if (enter_time == 2)
      {
       if(hourIn == 0x10)
       {
        if(number >=0 && number <= 2){
          write(num);
          hourIn |= (num - '0');
          enter_time +=2;
          cursorPos(3, enter_time);
          num=10;
        }
       }
       else if(hourIn == 0)
       {
        if(number >=0 && number <= 9){
          write(num);
          hourIn |= (num - '0');
          enter_time +=2;
          cursorPos(3, enter_time);
          num=10;
        }
       }
      }
      else if(enter_time == 4)
      {
       if(number >=0 && number <= 5){
        write(num);
        minIn = ((num - '0') << 4);
        enter_time++;
        cursorPos(3, enter_time);
        num=10;
       }
```

```
        }
        else if (enter_time == 5)
        {
         if(number >=0 && number <= 9){
           write(num);
           minIn |= (num - '0');
           command(0x0C);
           num=10;
           enter_time++;
         }
        }
        if (enter_time > 5)
        {
         if (hourIn <= 0x12 && minIn <= 0x59)
         {
           RTCCTL01 |= RTCHOLD;
           RTCMIN  = minIn;
           RTCSEC = 0x01;
           RTCHOUR = hourIn&0x3F;
           RTCCTL01 &= ~RTCHOLD;
         }
         if(num=='+'){
           strcpy(time, "AM");
           cursorPos(3,7);
           writeString(time);
         }
         else if(num=='-'){
           strcpy(time, "PM");
           cursorPos(3,7);
           writeString(time);
         }
         if(num=='E'){
           isMilitary = 0;
           isAMPM=1;
           enter_time = 1;
           hourIn = minIn = 0x00;
           computeNewRates();
           func = MAINMENU;
           main_menu();
         }
         if(time == "PM"){
           RTCHOUR = (hourIn+0x12)&0x3F;
           RTCMIN = minIn;
         }
        }
           }
}
static void militaryTimeEnter(char num) {

           if (func != MILITARYSCREEN)
           {
                   LcdReset();
                   displayTime();
                   cursorPos(2, 0);
                   writeString("Military Time Enter");
                   cursorPos(3, 0);
                   writeString(" __:__ ");
                   cursorPos(3, 1);
                   command(0x0F);
           }
           else
           {
```

```
                count++;
                if (count == 1)
                {
                        write(num);
                        hourInput = ((num - '0') << 4);
                        cursorPos(3, count+1);
                        num=10;
                }
                else if (count == 2)
                {
                        write(num);
                        hourInput |= (num - '0');
                        count++;
                        cursorPos(3, count+1);
                        num=10;
                }
                else if (count == 4)
                {
                        write(num);
                        minInput = ((num - '0') << 4);
                        cursorPos(3, count+1);
                        num=10;
                }
                else if (count == 5)
                {
                        write(num);
                        minInput |= (num - '0');
                        command(0x0C);
                        num=10;
                }
                if (count >= 5)
                {
                        if (hourInput <= 0x23 && minInput <= 0x59)
                        {
        isMilitary = 1;
        isAMPM=0;
        RTCCTL01 |= RTCHOLD;
        RTCHOUR = hourInput&0x3F;
                                RTCMIN = minInput;
                                //RTCSEC = 0x01;
                                RTCCTL01 &=~ RTCHOLD;
                                hourInput = minInput = 0x00;
        count = 0;
        computeNewRates();
                                func = MAINMENU;
                                main_menu();
                        }
                        else
                        {
                                cursorPos(4, 0);
                                writeString("INVALID TIME");
                                cursorPos(3, 0);
                                writeString(" __:__ ");
                                cursorPos(3, 1);
                                command(0x0F);
                                hourInput = minInput = count = 0;
                        }
                }
        }
}
static void goBack(char * temp) {
        LcdReset();
```

```
        displayTime();
        cursorPos(2, 0);
        writeString(temp);
        cursorPos(3, 0);
        writeString("Press Recall");
        cursorPos(4, 0);
        writeString("to Go Back");
        func = INVALID;
}
static void command(char i) {
        P3OUT = i;
        P7OUT &= ~(R_S);
        P5OUT &= ~(R_W);
        P5OUT |= EN;
        unsigned int x;
        for (x = 0; x < 100; x++)
                ;
        P5OUT &= ~EN;
        for (x = 0; x < 100; x++)
                ;
}
static void write(char i) {
        P3OUT = i;
        P5OUT &= ~(R_W);
        P7OUT |= (R_S);
        P5OUT |= EN;
        unsigned int x = 0;
        for (x = 0; x < 100; x++)
                ;
        P5OUT &= ~EN;
}
void writeString(char *s) {
        while (*s != '\0') {
                write(*s);
                s++;
        }
}
static char DDRAMAddr(char row, char col) {
        switch (row) { //from data sheet
        case 1:
                return col;
        case 2:
                return 0x40 + col;
        case 3:
                return 0x14 + col;
        default:
                return 0x54 + col;
        }
}
void cursorPos(char row, char col) {
        //  address or with D7MASK done to know the command
        char in = DDRAMAddr(row, col);
        in |= D7MASK;
        command(in);
}
static void displayTime() {
                cursorPos(1, 0);
                char str[20];
           char ampm[20];
           RTCCTL01 |= RTCHOLD;
                unsigned int hour1 =((RTCHOUR)&0x0F) ;
           unsigned int hour2 =(RTCHOUR >> 4);
```

```
                unsigned int min1 = RTCMIN >> 4;
            unsigned int min2 = RTCMIN & 0x0F;
            RTCCTL01 &= ~RTCHOLD;

            if (isMilitary)
            {
                sprintf(str, "Time: %x%x:%x%x", hour2,hour1,min1,min2);
               writeString(str);
            }
            else if(isAMPM){
                //change pm to am
                if((RTCHOUR) >= 0x13){
                        RTCCTL01 |= RTCHOLD;
                    hour2=(RTCHOUR-0x12)>>4;
                    hour1=(RTCHOUR-0x12)&0x0F;
                    RTCCTL01 &= ~RTCHOLD;
                    strcpy(time, "PM");
                }
                else
                        strcpy(time, "AM");
            sprintf(ampm, "Time: %x%x:%x%x %s", hour2,hour1,min1,min2,time);
            writeString(ampm);
        }
}
#pragma vector = RTC_VECTOR             //Basic Timer interrupt to handle debouncing
__interrupt void rtc_isr() {
        displayTime();
        doAutomaticMode();
        RTCCTL01 &= ~RTCTEVIFG;
}
LED_code.h
#ifndef LED_CODE_H
#define LED_CODE_H

extern unsigned int pulseRate;
extern unsigned int redRate;
extern unsigned int greenRate;
extern unsigned int blueRate;
extern unsigned int warmWhiteRate;
extern unsigned int coolWhiteRate;
extern unsigned int isAutomaticMode;
extern unsigned int isDemoMode;
extern unsigned int holdDemoValue;
extern unsigned int addGreen;
extern unsigned int addBlue;

void hbridgesPort_initRGB(int enable);
void hbridgesPort_initCoolWhite(int enable);
void hbridgesPort_initWarmWhite(int enable);
void timerPorts_init();
void timer_init();

void customColorMaker(unsigned int red, unsigned int green, unsigned int blue);
void customWhiteMaker(unsigned int wWhite, unsigned int cWhite);

void turnOff();
void sampleFunction();
void doAutomaticMode();
void computeNewRates();
```

```
#endif
```

LED_code.c

```
#include "msp430f5438.h"
#include "LED_code.h"

//***************************************************************************
//Defines
//***************************************************************************

//***************************************************************************
//Globals
//***************************************************************************

unsigned int pulseRate = 64;
unsigned int redRate = 225;
unsigned int greenRate = 125;
unsigned int blueRate = 1;
unsigned int warmWhiteRate = 125;
unsigned int coolWhiteRate = 125;
unsigned int isAutomaticMode = 0;
unsigned int isDemoMode = 0;
unsigned int addGreen = 0;
unsigned int addBlue = 0;
unsigned int holdDemoValue = 0;

//***************************************************************************
//Function: hbridgesPort_init()
//Description: initializes all ports being used for high and low signals for h-bridge
//***************************************************************************
void hbridgesPort_initRGB(int enable) {
	P9DIR |= (BIT6 | BIT7);
	if (enable)
		P9OUT |= (BIT6 | BIT7); //Enable lines for h-bridges
	else
		P9OUT &= ~(BIT6 | BIT7);
}
//***************************************************************************
//Function: hbridgesPort_initCoolWhite()
//Description: initializes all ports being used for high and low signals for h-bridge
//***************************************************************************
void hbridgesPort_initCoolWhite(int enable) {
	P9DIR |= BIT5;

	if (enable)
		P9OUT |= BIT5;
	else
		P9OUT &= ~BIT5;
}
//***************************************************************************
//Function: hbridgesPort_initWarnWhite()
//Description: initializes all ports being used for high and low signals for h-bridge
//***************************************************************************
void hbridgesPort_initWarmWhite(int enable) {
	P9DIR |= BIT4;

	if (enable)
		P9OUT |= BIT4;
	else
		P9OUT &= ~BIT4;
}
```

```
//*******************************************************************************
//Function: timerPorts_init
//Description:initializes all the ports that are being used for timers
//*******************************************************************************
void timerPorts_init() {
	P8DIR |= (BIT1 | BIT2 | BIT3);

	P8SEL |= (BIT1 | BIT2 | BIT3);

	P4DIR |= (BIT1 | BIT2); //FOR COOL AND WARM WHITES

	P4SEL |= (BIT1 | BIT2);
}
//*******************************************************************************
//Function: timer_init()
//Description:timer are haulted by default
//*******************************************************************************
void timer_init() {
	TA0CCR0 = pulseRate;

	TA0CCTL1 = OUTMOD_7;
	TA0CCR1 = 0;

	TA0CCTL2 = OUTMOD_7;
	TA0CCR2 = 0;

	TA0CCTL3 = OUTMOD_7;
	TA0CCR3 = 0;

	TA0CTL = TASSEL_1 + MC_0; // ACLK, stop mode

	TBCCR0 = pulseRate;

	TBCCTL1 = OUTMOD_7;
	TBCCR1 = (unsigned int) (((double) warmWhiteRate / (double) 255) * pulseRate);

	TBCCTL2 = OUTMOD_7;
	TBCCR2 = (unsigned int) (((double) coolWhiteRate / (double) 255) * pulseRate);

	TBCTL = TASSEL_1 + MC_1; //stop mode
}
void customColorMaker(unsigned int red, unsigned int green, unsigned int blue) {
	TA0CTL |= MC_0;
	TA0CCR1 = red;
	TA0CCR2 = green;
	TA0CCR3 = blue;
	TA0CTL |= TACLR | MC_1;
}
void customWhiteMaker(unsigned int wWhite, unsigned int cWhite) {
	TBCTL |= MC_0;
	TBCCR1 = (unsigned int) (((double) wWhite / (double) 255) * pulseRate);
	TBCCR2 = (unsigned int) (((double) cWhite / (double) 255) * pulseRate);
	TBCTL |= MC_0;
}
//*******************************************************************************
//Function: turnOff()
//Description: turnOff RGBs
//*******************************************************************************
void turnOff() {

	TA0CTL |= MC_0;
```

```
        TA0CCR1 = (unsigned int) (((double) 0 / (double) 255) * pulseRate);
        TA0CCR2 = (unsigned int) (((double) 0 / (double) 255) * pulseRate);
        TA0CCR3 = (unsigned int) (((double) 0 / (double) 255) * pulseRate);
        TBCCR1 =  (unsigned int) (((double) 0 / (double) 255) * pulseRate);
        TBCCR2 =  (unsigned int) (((double) 0 / (double) 255) * pulseRate);

        TA0CTL |= TACLR | MC_0;
}
void sampleFunction() {

        TA0CTL |= MC_0;

        TA0CCR1 = (unsigned int) (((double) 0 / (double) 255) * pulseRate);
        TA0CCR2 = (unsigned int) (((double) 255 / (double) 255) * pulseRate);
        TA0CCR3 = (unsigned int) (((double) 0 / (double) 255) * pulseRate);

        TA0CTL |= TACLR | MC_1;
}
//*****************************************************************************
//Function: void doAutomaticMode()
//Description: starts automatic mode
//*****************************************************************************
void doAutomaticMode() {
        unsigned int val1 = 0;
        unsigned int val2 = 0;
        unsigned int val3 = 0;
        if (RTCHOUR < 0x06 || RTCHOUR >= 0x20) {
                if (isAutomaticMode)
                        turnOff();
                redRate = 225;
                greenRate = 125;
                blueRate = 1;
                return;
        }
        if (RTCHOUR >= 0x06 && RTCHOUR < 0x08) //6 and 7
                {
                addBlue++;
                addGreen++;
                if (addGreen >= 5) {
                        greenRate += 1;
                        addGreen = 0;
                }
                if (addBlue >= 10) {
                        blueRate += 1;
                        addBlue = 0;
                }
                if (isAutomaticMode) {
                        customWhiteMaker(0, 0);
                        val1 = (unsigned int) (((double) redRate / (double) 255)* pulseRate);
                        val2 = (unsigned int) (((double) greenRate / (double) 255)* pulseRate);
                        val3 = (unsigned int) (((double) blueRate / (double) 255)* pulseRate);
                        customColorMaker(val1, val2, val3);
                }
        } else if (RTCHOUR >= 0x08 && RTCHOUR < 0x11) {
                blueRate += 1;
                addGreen++;
                if (addGreen >= 3) {
                        greenRate += 1;
                        addGreen = 0;
                }
                if (isAutomaticMode) {
                        customWhiteMaker(0, 0);
```

```
                        val1 =(unsigned int) (((double) redRate / (double) 255)* pulseRate);
                        val2 = (unsigned int) (((double) greenRate / (double) 255)* pulseRate);
                        val3 = (unsigned int) (((double) blueRate / (double) 255)* pulseRate);
                        customColorMaker(val1, val2, val3);
                }
        } else if (RTCHOUR >= 0x11 && RTCHOUR < 0x15) {
                //Turn off RGB, turn on whites
                redRate = 225;
                greenRate = 209;
                blueRate = 193;
                warmWhiteRate = 150;
                coolWhiteRate = 150;
                if (isAutomaticMode) {
                        val1 =(unsigned int) (((double) redRate / (double) 255)* pulseRate);
                        val2 = (unsigned int) (((double) greenRate / (double) 255)* pulseRate);
                        val3 = (unsigned int) (((double) blueRate / (double) 255)* pulseRate);
                        customColorMaker(val1, val2, val3);
                        customWhiteMaker(warmWhiteRate, coolWhiteRate);
                }
        } else if (RTCHOUR >= 0x15 && RTCHOUR < 0x18) {
                customWhiteMaker(0, 0);
                blueRate -= 1;
                addGreen++;
                if (addGreen >= 3) {
                        greenRate -= 1;
                        addGreen = 0;
                }
                if (isAutomaticMode) {
                        customWhiteMaker(0, 0);
                        val1 =(unsigned int) (((double) redRate / (double) 255)* pulseRate);
                        val2 = (unsigned int) (((double) greenRate / (double) 255)* pulseRate);
                        val3 = (unsigned int) (((double) blueRate / (double) 255)* pulseRate);
                        customColorMaker(val1, val2, val3);
                }
        } else if (RTCHOUR >= 0x18 && RTCHOUR < 0x20) {
                addBlue++;
                addGreen++;
                if (addGreen >= 5) {
                        greenRate -= 1;
                        addGreen = 0;
                }
                if (addBlue >= 10) {
                        blueRate -= 1;
                        addBlue = 0;
                }
                if (isAutomaticMode) {
                        customWhiteMaker(0, 0);
                        val1 =(unsigned int) (((double) redRate / (double) 255)* pulseRate);
                        val2 = (unsigned int) (((double) greenRate / (double) 255)* pulseRate);
                        val3 = (unsigned int) (((double) blueRate / (double) 255)* pulseRate);
                        customColorMaker(val1, val2, val3);
                }
        }
}
void computeNewRates() {
        unsigned int val = (RTCMIN >> 4) * 10;
        val += (RTCMIN & 0x0F);

        unsigned int doCustom = 1;
        unsigned int doWhite = 0;
        unsigned int val1 = 0;
        unsigned int val2 = 0;
```

```
	unsigned int val3 = 0;
	addBlue = 0;
	addGreen = 0;
	redRate = 225;

	if (RTCHOUR == 0x06) {
		greenRate = 125 + (val / 5);
		blueRate = 1 + (val / 10);
	} else if (RTCHOUR == 0x07) {
		greenRate = 125 + (60 / 5) + (val / 5);
		blueRate = 1 + (60 / 10) + (val / 10);
	} else if (RTCHOUR == 0x08) {
		greenRate = 149 + (val / 3);
		blueRate = 13 + val;
	} else if (RTCHOUR == 0x09) {
		greenRate = 149 + (60 / 3) + (val / 3);
		blueRate = 13 + 60 + val;
	} else if (RTCHOUR == 0x10) {
		greenRate = 149 + (120 / 3) + (val / 3);
		blueRate = 13 + 120 + val;
	} else if (RTCHOUR >= 0x11 && RTCHOUR < 0x15) {
		greenRate = 209;
		blueRate = 193;
		doWhite = 1;
		doCustom = 0;
	} else if (RTCHOUR == 0x15) {
		greenRate = 209 - (val / 3);
		blueRate = 193 - val;
	} else if (RTCHOUR == 0x16) {
		greenRate = 209 - (60 / 3) - (val / 3);
		blueRate = 193 - 60 - val;
	} else if (RTCHOUR == 0x17) {
		greenRate = 209 - (120 / 3) - (val / 3);
		blueRate = 193 - 120 - val;
	} else if (RTCHOUR == 0x18) {
		greenRate = 149 - (val / 5);
		blueRate = 13 - (val / 10);
	} else if (RTCHOUR == 0x19) {
		greenRate = 149 - (60 / 5) - (val / 5);
		blueRate = 13 - (60 / 10) - (val / 10);
	} else {
		greenRate = 125;
		blueRate = 1;
		doCustom = 0;
	}
	if (doCustom && isAutomaticMode) {
		val1 =(unsigned int) (((double) redRate / (double) 255)* pulseRate);
		val2 = (unsigned int) (((double) greenRate / (double) 255)* pulseRate);
		val3 = (unsigned int) (((double) blueRate / (double) 255)* pulseRate);
		customColorMaker(val1, val2, val3);
		customWhiteMaker(0, 0);
	} else if (doWhite && isAutomaticMode) {
		val1 =(unsigned int) (((double) redRate / (double) 255)* pulseRate);
		val2 = (unsigned int) (((double) greenRate / (double) 255)* pulseRate);
		val3 = (unsigned int) (((double) blueRate / (double) 255)* pulseRate);
		customColorMaker(val1, val2, val3);
		customWhiteMaker(255, 255);
	} else if (isAutomaticMode){
		customColorMaker(0, 0, 0);
		customWhiteMaker(0, 0);
	}
}
```

Appendix A

Smart Living Room LED Proposal

3/2/2012
Maryam Nasri
Marzia Shabbir
Mayank Mehta
Girmay Tewelde

Executive Summary:

The objective of this project is to design a Smart Living Room LED light. The LEDs and the circuitry for the device shall be nicely integrated into a tochiere lamp. The device shall be easy to install and work well with a power supply. The device will be designed such that it can operate in two different modes: automatic and manual. The automatic mode will get a time based on user input or wirelessly through a local time zone and be able to fully adjust the light and imitate sunlight based on different times of the day. The manual mode would let the user to control the device through an IR interface and sets its own parameters such as brightness, and colors. There will be a microcontroller that will drive the functionality of the LEDs and interact with the user through an infrared detector. Furthermore, there will be a power supply that can step down the voltage as per as the required needs for all the different modules of the device. The device should be easily commutable from one place to another and be able to provide a natural and soothing light for the user.

Problem Statement:

Need:

Human brain shows different reactions to different colors of light. Some light colors are calming and relaxing and some are energetic. Cool colors reflect the fresh violets and blues of moonlight. Warm colors project the hot hues of sunlight and create a feeling of warmth in a room. In addition, light therapy can be used to treat illnesses such as depression. Some individuals become depressed during the long winter months when sunshine is limited. This problem is very common in countries located above 60° latitude, where the sun remains very low during the winter month and people do not see much of the sunlight. These individuals can be exposed to a light specially designed to mimic that of the sun. In general, having the right kind of lightening in work and home environment can help people's mental and emotional health and improve their performance.

From the beginning of human life on this earth, people begin their work with the rise of the sun and sleep when the sun goes down. Their body clock is synchronized with the time zone they live in. However, there are some cases when this body clock is affected and a person develops certain sleeping disorders such as insomnia. Not only that but there could be several other cases in which a person finds it difficult to sleep at night and feel fresh and active in the day time. A person experiences similar effects when going through a jet lag. A clinical research suggests that natural production of melatonin (hormone which makes one sleepy) could be reserved by exposing one to low levels of light in the blue part of the spectrum and blocking that part of spectrum could help melatonin to flow. LEDs (Light emitting diodes) are solid light bulbs, which are extremely energy efficient and durable. More over the brightness of LEDs could be controlled easily according to ones need. Thus, LEDs efficiency is not limited to its less energy consumption and durability but its precise control capabilities could be used to deal with issues of jet lag and sleep disorders such as insomnia. The Goal of this project is to design a 150W equivalent LED light source that imitates the sunlight in order to possibly deal with the conditions mentioned above.

Objective:

The goal of this project is to design and implement a prototype system that can be purchased by a consumer and used at home or work. It should be affordable and easy to use and maintain. Interacting with the device should be easy enough to not need any training on how to use it.

Requirements Analysis Summary:

- Possibly used as a sleep-aide for a user going through insomnia
- Acquire and compare power consumption with a regular light sources
- Opportunity to help passengers adjust to jet lag more easily
- Create a light source without being a by-product of mercury
- Develop a 150W equivalent LED based light source
- Designed focused for a torchiere floor lamp
- Imitate sunlight spectrum using LEDs
- Generate a long-lasting light source
- A user controlled wireless interface with the lamp
- The integration of the design with a microcontroller
- The LEDs should only use 13 W maximum power.
- The LED should have a minimum of 800 lumens

Approach

LED Lights

The main idea of this project is to use a combination of high power LEDs approximately around 1 Watt each. The color spectrum needed requires a combination of bright white LEDs along with red, blue and green. The goal is to build a light source equivalent to 150 W, which is approximately 2600 lumens. With such high power LEDs an estimated number of 20 to 30 LEDS will be enough to produce a light source equivalent to the desired efficiency of light.

Microcontroller

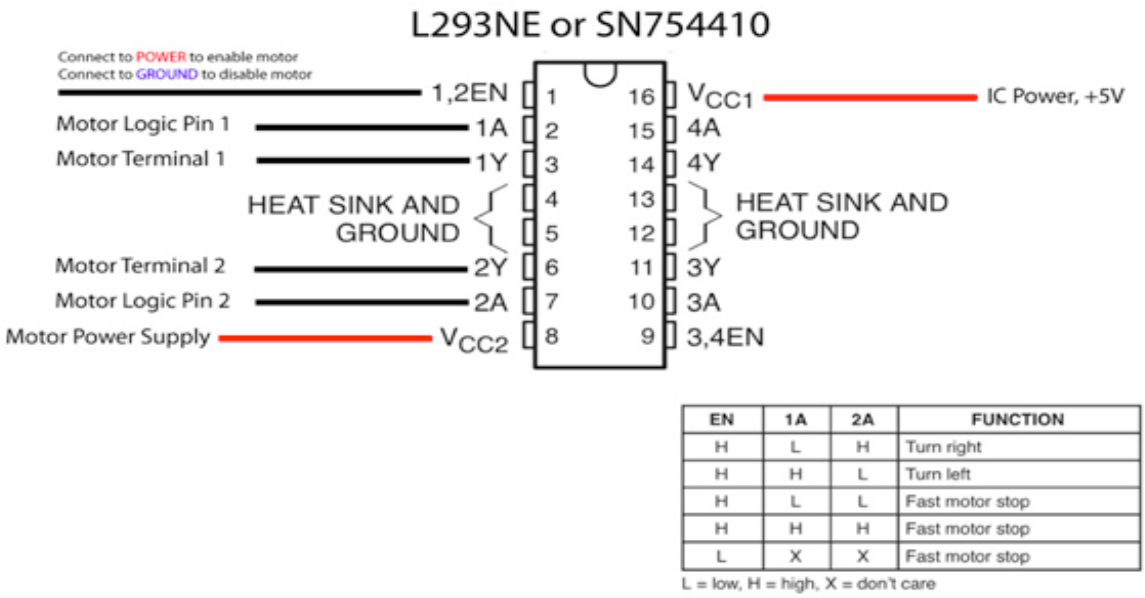
A Microcontroller is nothing more than a small computer on a single integrated circuit with some processor core, memory and input/output peripherals. Most of the applications used for microcontrollers are embedded applications. They are designed to run one task and the program is stored in the ROM and generally does not change unless it is being programmed again. Furthermore, they are low-power devices and consume less power.

There are several vendors for microcontrollers that are available in the market these days. Some of these vendors are Microchip providing microcontrollers from 8-bit to 32-bit, Infineon 8-bit to 32-bit, Texas Instruments (16-bit), Atmel AVR 8-bit and 32-bit. The microcontroller that is being used for this project is Texas Instruments TI MSP430FG4618. This microcontroller provides a low supply voltage range 1.8V to 3.6V, 16 bit RISC architecture, 12-bit A/D converter, two 16-bit timers, real time clock, Universal Serial Communication Interface, 116KB Flash and 8KB RAM. For the scope of this project, this microcontroller should be able to provide the desired actions necessary.

H-Bridge

An H-Bridge is a motor-driver often used to provide high current to DC motors than a microcontroller can provide. In our project, we will be able to provide higher current to the LEDs using an H-Bridge since the microcontroller can only provide about 20mA current through its output pin. There are several different H-Bridges that are available in the market and the H-Bridge for our project is SN754410

Dual H-Bridge Motor Controller (see figure 1). This is a dual H-Bridge so it can provide output to 2 columns of a LED matrix.



EN	1A	2A	FUNCTION
H	L	H	Turn right
H	H	L	Turn left
H	L	L	Fast motor stop
H	H	H	Fast motor stop
L	X	X	Fast motor stop

L = low, H = high, X = don't care

Figure 1. Dual H-Bridge Motor Controller

Connecting LEDs

There are different ways to connect the High Power LEDs in order to achieve the desired behavior for this project. One of this ways is to connect the LEDs in a daisy chain manner, in which all the LEDs are connected in series with each other and the same current flows through all of them. In order to accommodate this connection, a high voltage needs to be providing to this chain so that all the LEDs can function properly. This voltage can be provided through an H-bridge however, the input voltage has to be correspondingly high as well. For example, if the desired voltage across each LED is about 3.5 V and ten LEDs are connected in series a voltage of 35 V would be required for each of the LED to function properly. In order to provide this voltage, an input voltage of at least 37 V is required for the H-Bridge.

The second way of connecting the LEDs is parallel with the anode of all LEDs tied together to a certain voltage and cathode being connected to ground. This would not require a high voltage but require a higher amount of current for the LEDs since the current would split between each LED. Furthermore, this current can be provided by the H-Bridge as well providing a sufficient amount of voltage as input. For this

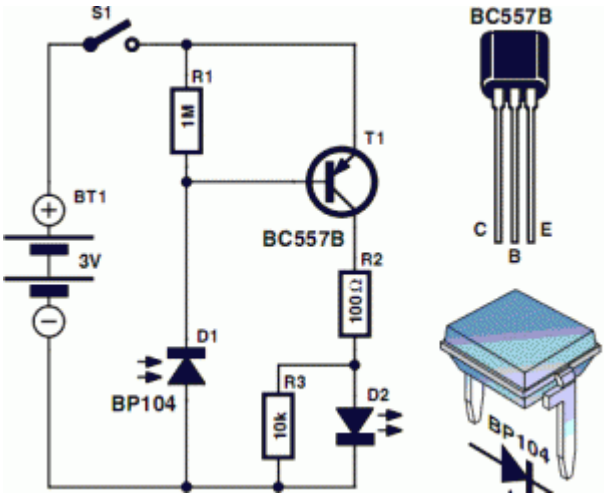
project, the LEDs would be connected in parallel so that the voltage can be easily controlled through a PWM signal and different behaviors can be achieved via the LEDs.

User Interface

There are three possible approaches to implement user interface for our project. Given that there is a possibility that the Microcontroller we choice for our project supports an MSP430 Application called UART, which allows serial communication to PC. We can use a HyperTerminal window to interact with the microcontroller. Thus the user can send a start signal from the PC to the microcontroller at any specific time. Using given input the system can respond appropriately. For example the user can type a “start” command on the PC, and then the microcontroller reads the command and runs or stops the system. It’s also possible to combine the UART with the real-time clock, in order to generate an alarm clock to synchronize with the outside light source.

Our second approach is to use a simple Plug-in Dimmer/Relay wireless Light switch. The user plugs the lamp into the Dimmer/Relay then plugs the Dimmer/Relay into an outlet. The user controls the lamp using a wireless light switch. The switch can be used as an off/on switch to start or turn-off the lamp.

The third approach which is what we probability aiming for at this point is using an IR detector. IR detectors are microchips tuned in to listen to infrared light. They are used for remote control detection. The remote control emits IR pulses using matching IR LED tuned to a specific frequency. The IR detector receives and analyzes the pulse to run a particular command. IR LED inside the remote have to PWM blinking at a specific frequency in order for the IR detector to detect that frequency. So for our project we can use an IR detector tuned to a specific frequency of a PWM frequency from the remote control. Once the user presses a button on the remote control the IR detector processes data and implements the command. We integrate the IR detector with the microcontroller so every time a specific button is pressed something happens. So we can use the button to control brightness or color. We can also use the remote control to set the time. It would take a lot of pulse measurement to implement a remote control but it is doable.



infrared light emitted by the remote control unit. Once the IR receives the signal it can do the specific output. So the user can use a remote control to turn on or off the lamp or set the specific time to turn the lamp on. Since most IR detectors are set to receive a particular frequency

The picture shows a circuit diagram for the Infrared Detector of a Remote Control. The circuit uses a photo-diode (D1) to sense the the remote control should be compatible to the IR used. Initially an IR detector should be tested to make sure that it senses remote button presses. Then using a remote control and a microcontroller raw IR codes should be read from the remote control into the microcontroller analyzed and separated based on button presses. Using the data the microcontroller can differentiate between button presses and run the specific remote command. An IR detector is compatible with most microcontrollers it runs on 5v power supply. Connecting the IR detector to the microcontroller is pretty simple, the IR has three pins, one to power, the other to ground and the third to a port on the microcontroller. We can possible use the same IR detector as on the diagram above or we can choice another.

Preliminary Design

Driving LEDs through Microcontroller via H-Bridge

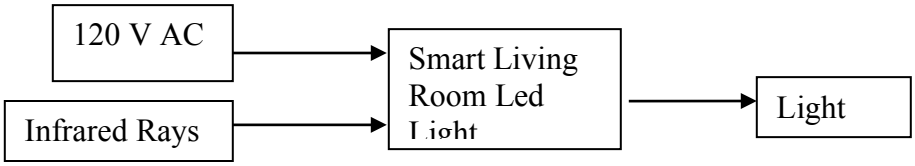
The first stage in this project is to drive the power for this device. This will be done by design a circuit that will take the 120 VAC and convert it into 12-15 VDC. The circuit shown in Figure 2 is what will be used to drive the power in to the microcontroller and the H-Bridges.

One of the major tasks of this project is to drive the High Power LEDs with the required current needed for the LEDs and keeping the power dissipation low. The LEDs would be driven through an H-Bridge since the microcontroller only provides about 20mA and the LEDs require a much higher current to

operate properly. The H-Bridge chosen for this project can provide about 2A current to the LEDs. Assuming each LED consumes about 500mA of current, this H-Bridge should be able to light up four LEDs properly when the LEDs are connected in parallel.

Furthermore, the LEDs will also be controlled through a PWM signal that comes from the microcontroller and acts as the enable for the H-Bridge. This PWM signal would be varied for different LEDs such as Red, Green, Blue and White based on the behavior desired and hence, would also control the brightness of the LEDs. This would also help to incorporate the desired amount of color needed for each LED to accommodate different colors other than just red, green, blue and white. Hence, this would help to imitate sunlight as well.

Top Level Design



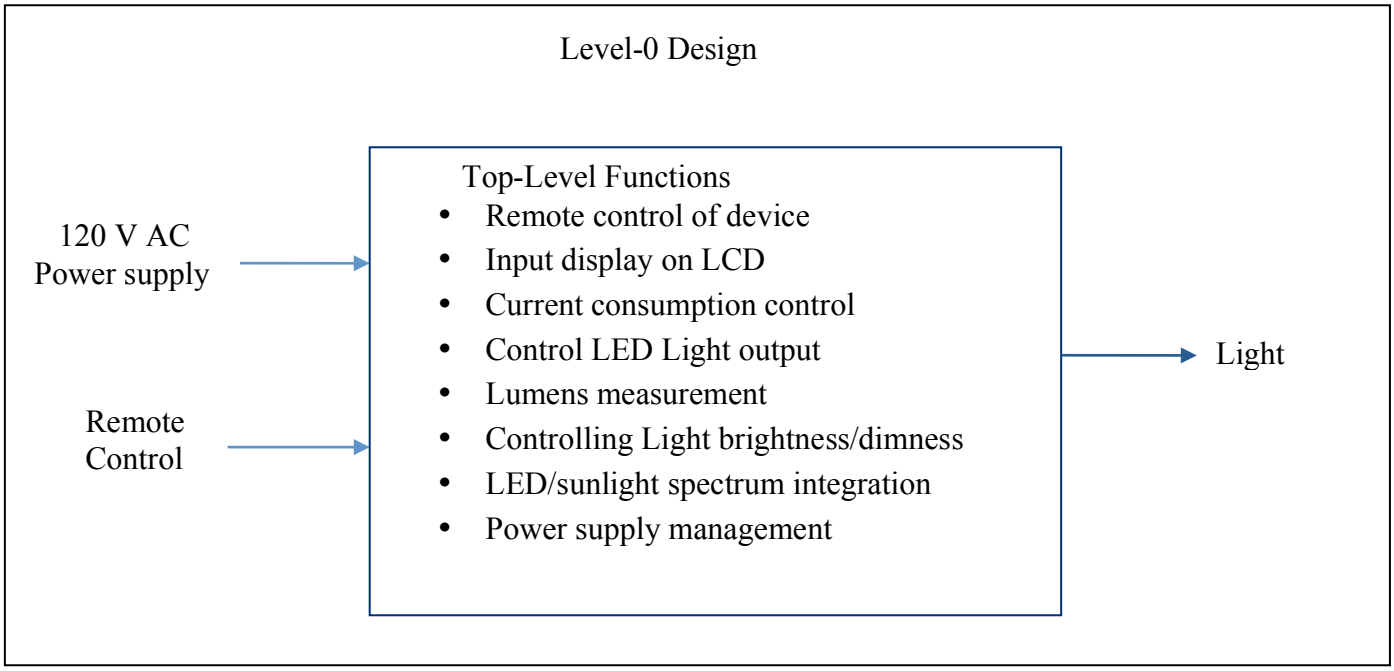


Figure 1: Level zero

Level 1

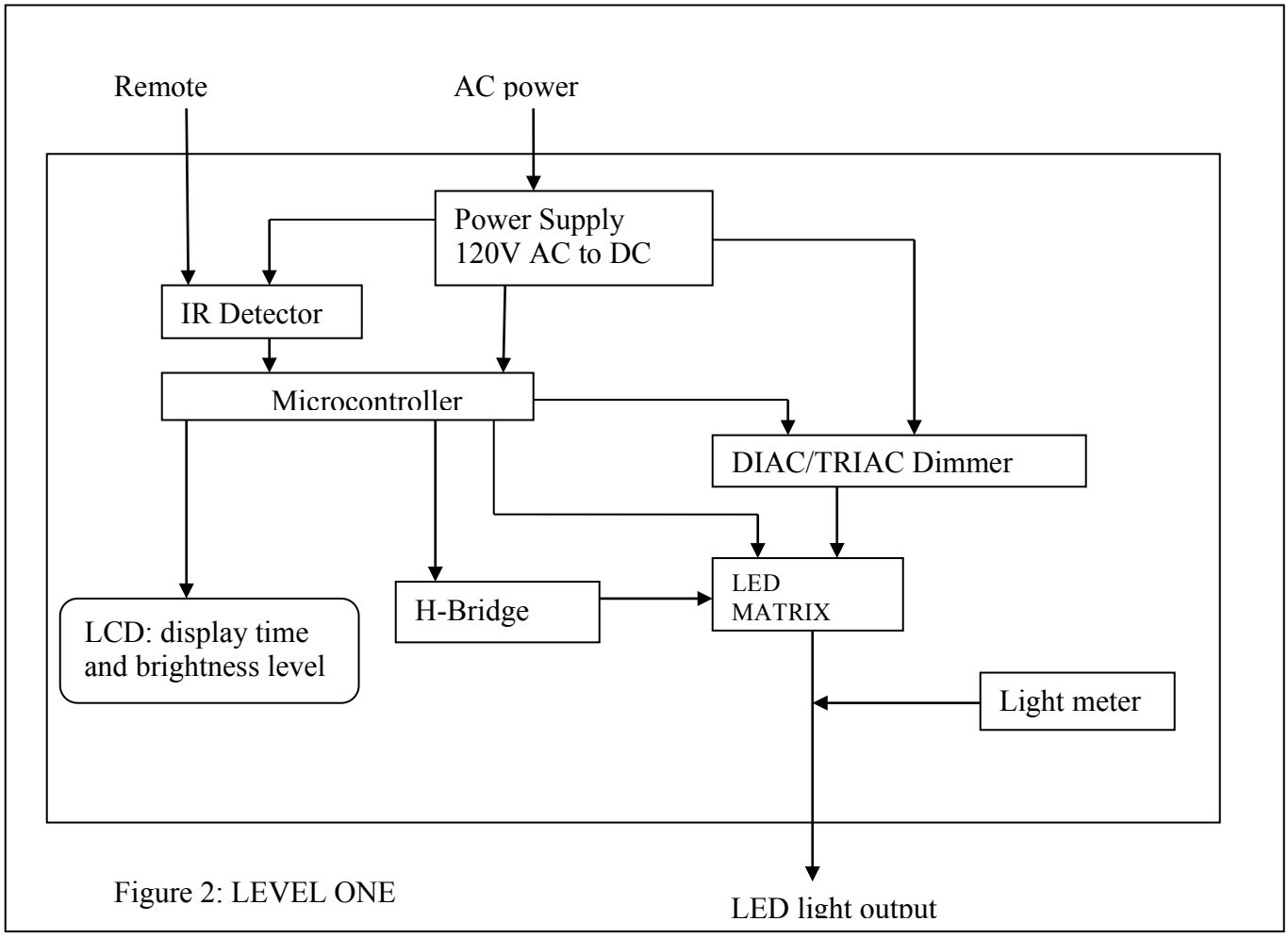


Figure 2: LEVEL ONE

Preliminary Experimentation Plan:

- LED Arrangement Testing:

  The LEDs will be arranged in an array of rows and columns of specific colors. The idea is to control these LEDs via PWM signals such that a row or column of a specific color is dimmed /turned off and the others are provided with high voltages to brighten them or low voltages for dimming the specific LEDs. This will require testing different patterns for LED arrangement.

- LED Power Testing:

  This testing will be performed basically to check if the H-bridge we are using to provide high power to the LEDs is providing the required amount of current to drive such LEDs.

We will also make sure that no part of our circuit is over heating or if there are significant voltage or current drops by varying the load.

- Color testing:

  This testing would be performed to test if the light color is changing appropriately as expected. For example if it is night time outside and the light source is producing bright lights imitating the day light then there could be some problems with the voltage levels and PWM supplied to the LEDs or the signal to the RTC.

- IR Detector:

  For the user interface we will be using an IR detector. There is a lot of testing to be done to select the appropriate frequencies which the IR detector can receive to analyze the specific task assigned according to the PWM frequencies tuned from a remote control, for example setting the time by the user.

Requirements specification:

- Be nicely integrated into the torchiere lamps
- Be equipped with a power switch
- Be equipped with a control panel through which the user can program the unit or manually control it
- Feature four color LED lights, which are bright white, red, green, and blue
- The user should have the capability to control the device automatic or manual
- The range of the colors of LEDs should imitate sunlight during different times of the day
- The user should have the capability to control the device automatic or manual
- The user should be able to interact with the device and increase/decrease the brightness or change the colors
- The user should be able to select different modes of operation for the device
- The brightness of the lights should be controlled by a dimmer
- The brightness and color of the LEDs should be independent of each other

Preliminary Schedule:

Week 5: Begin reverse engineering similar products.

Week 6: Design DIAC/TRIAC circuit. Create Parts List.
Show parts list to FS for approval.

Week 7: Begin building DIAC/TRIAC circuit.
Begin working on IR detector – finalize design and ensure it syncs with microcontroller

Week 9: Oral Presentation. Proposal Due. Team Evaluations.
Show built circuit to FS for approval and testing.
Finalize microcontroller selection

Week 10: Identify all parts needed to build the unit. Begin building DIAC/TRIAC circuit.
Begin preliminary design of the lamp in which components will reside. Select LEDs.

Week 11: Order Parts needed to complete the design. Begin working on the control panel.

Week 12: Draft Design Document Due. Prototyping Progress Report.
Begin coding microcontroller. Test single channel brightness control using the micro.

Week 13: Finalize software design. Finalize lamp Design. Continue coding microcontroller.

Week 14: Design Document Delivery. Document Tracking Form. Team Evaluations.

Task List:

1- Research the circuit operation

2- Research best zero detecting circuit for the microcontroller

3- Research best microcontroller for the project

4- Research power controlling for the microcontroller

5- Build a zero detector circuit to test the microcontroller (vector board)

6- Research and test the circuit independently

7- Built and test the control panel

8- Program the microcontroller

9- Build the IR detector and sync it with the microcontroller

10- Test the microcontroller with LEDs

11- Build the light dimmer circuit, and test it without the microcontroller (vector board)

12- Connect the microcontroller to the light dimmer circuit

13- Test the final circuit, and design the PCB

14- Build the PCB

15- Build the containing lamp

16- Buy the connectors, outlets, cables, and build the unit

17- Test the final product

Appendix B

Design Document
Smart Living Room LED Light

Faculty Advisor: Dr. Kaps

Maryam Nasri
Marzia Shabbir
Mayank Mehta
Girmay Tewelde
Due Date: April 27, 2012

Table of Contents

Executive Summary:

The objective of this project is to design and implement a sunlight simulator using an LED light source. The LED light source will be nicely integrated into a torchiere lamp. The device will be an easily controlled lamp that is portable and energy efficient. The completed product will be affordable and easily accessible. The device will be designed such that it can operate in two different modes: automatic and manual. The automatic mode will get a time from the user to fully adjust the light and imitate sunlight based on different times of the day. The manual mode would let the user to control the device through a remote control and sets its own parameters such as brightness, colors, and time to turn off/on the lamp. The applications for this device are countless. The devices color and brightness output can be used to create a pleasant atmosphere and reduce jet-lag. It can be applied in sleep therapy to generate an atmosphere for sleeping. It can also be used in underground office spaces and underground transportation as a sunlight simulator. Since the device will be easily commutable from one place to another, it will provide a natural and soothing light for the user anywhere.

Requirements Analysis Summary:

- Possibly used as a sleep-aide for a user going through insomnia
- Acquire and compare power consumption with a regular light sources
- Opportunity to help passengers adjust to jet lag more easily
- Develop a 150W equivalent LED based light source
- Designed focused for a torchiere floor lamp
- Imitate sunlight spectrum using LEDs
- Generate a long-lasting light source that imitates sunlight through LEDs
- A user controlled wireless interface with the lamp
- The integration of the design with a microcontroller
- The LED should have an approximation of 2600 lumens at 100 percent brightness

Requirements specification:

- Be nicely integrated into a torchiere lamp
- Be equipped with a power switch
- Feature four color LED lights, which are cool/warm white, red, green, and blue
- The user should have the capability to control the device automatically or manually
- The range of the colors of LEDs should imitate sunlight during different times of the day
- The user should be able to interact with the device and increase/decrease the brightness or change the colors
- The user should be able to select different modes of operation for the device
- The brightness and color of the LEDs should be independent of each other

Design Architecture

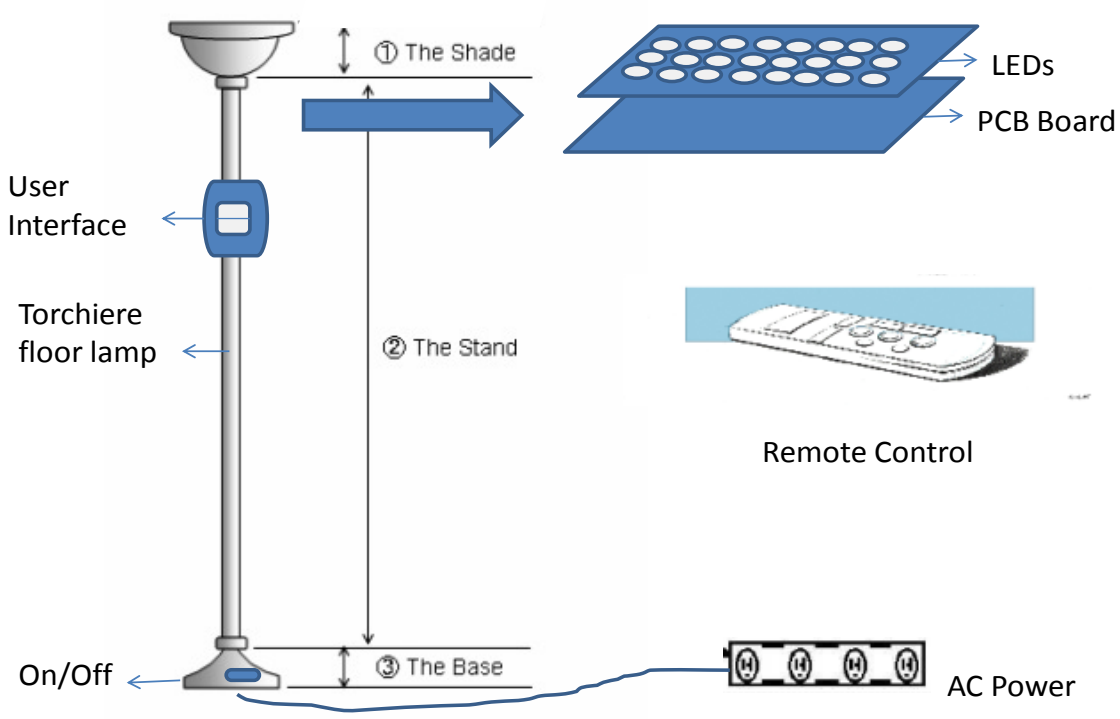


Figure 1: Design Architecture

Conceptual Design:

The LED light that is being designed will have a power cord connected to the AC power outlet. The main PCB circuit and the LED matrix will be placed inside the shade of the lamp. The user could control the device by a user interface attached to the stand of the torchiere lamp or wirelessly using a remote control. An LCD would be mounted on the body of the lamp to show the parameters of the device. There will be a simple power button or knob to turn the LED light On/Off. The height of the lamp will be the average size of a torchiere lamp.

Top Level Design

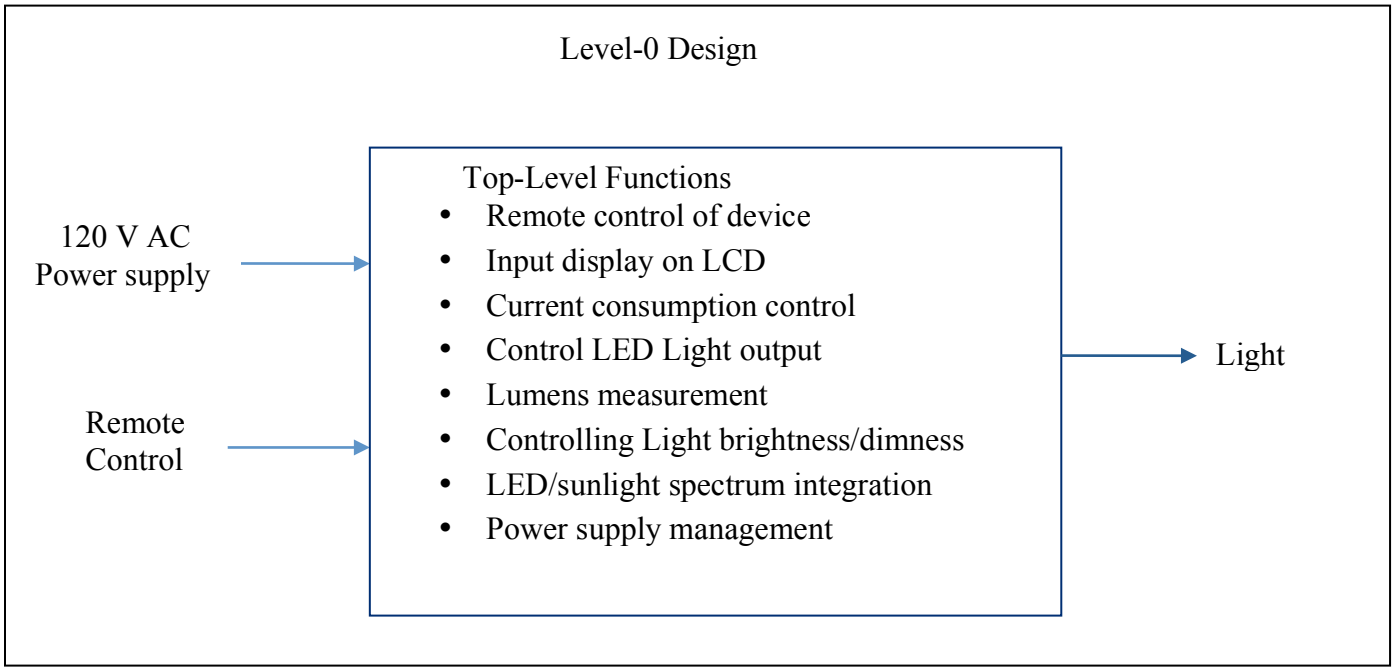


Figure 2: Level Zero

The above figure gives a top level idea of the device and its functions. The power to the system is provided by AC power outlet. The Remote control acts as the user interface input to the system whereas the Light being produced through LEDs acts as an output. The device should be able to perform necessary functions such as converting power from AC to DC, decode signal being sent from the remote and act based on that signal. It will also control the LED light output and change the colors and the brightness independently of each other. It will also be able to drive an LCD as a guide for the user to interact with the device.

Level 1 Design:

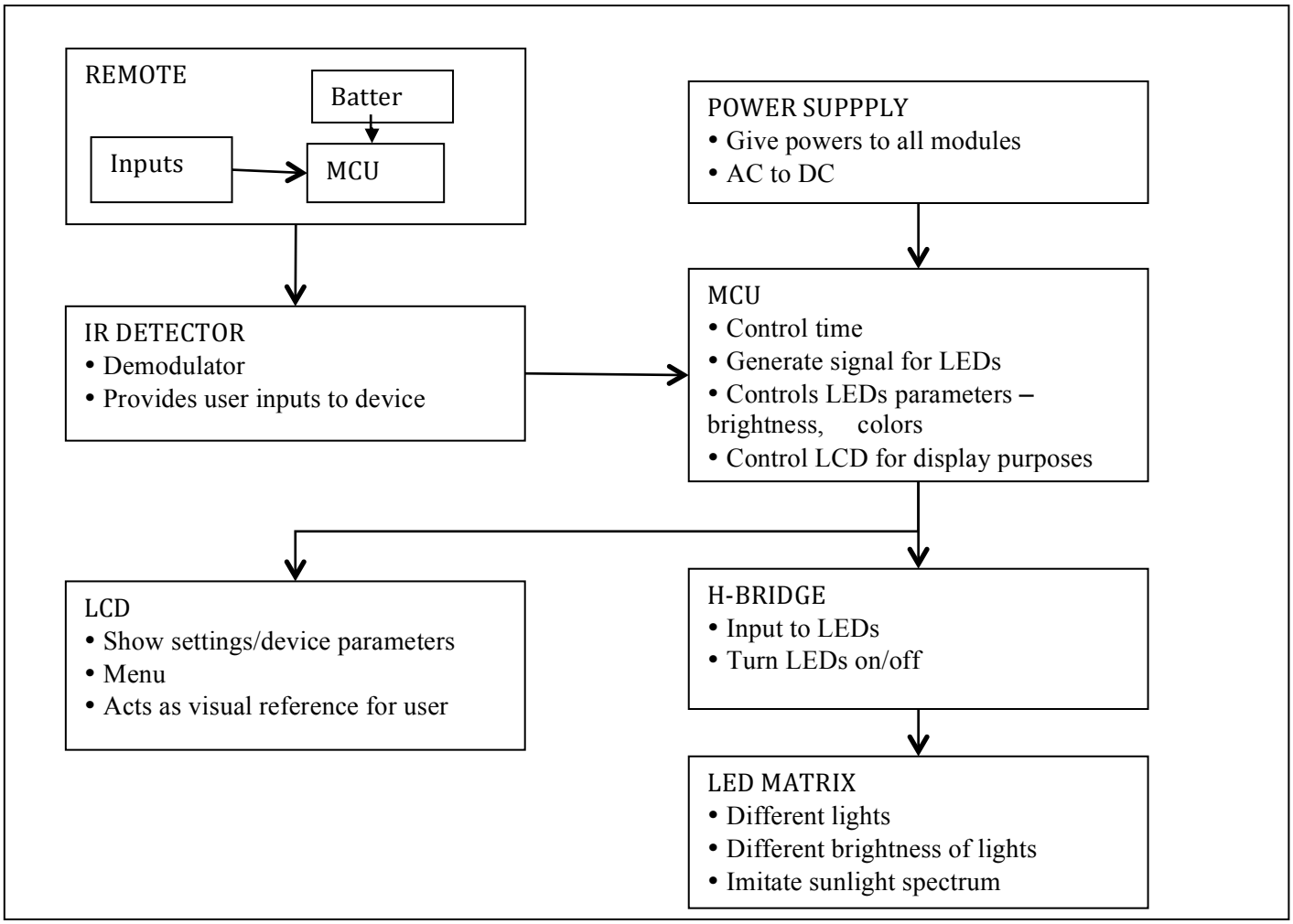


Figure 3: Level One

The above figure demonstrates the Level One design on our system. It goes into the details of each specific component that corresponds to the device. The device will be remotely controlled by the user and serve as a basis to control the device. All the other components are part of the device and would be mounted on the body of the device. The purpose of the IR detector is to decode the signal being sent by the remote control and provide this decoded signal to the microcontroller. The Microcontroller acts as the central part of the device. It controls various things such as the LCD and the H-bridge. It provides output to the LCD which acts as a visual reference for the user and shows all parameters of the device and the time of day. The H-Bridge is used to provide higher current to the LEDs and hence, the microcontroller

controls the H-Bridge through a PWM signal and is able to change the parameters such as color and brightness of the LEDs.

Level 2 Design:

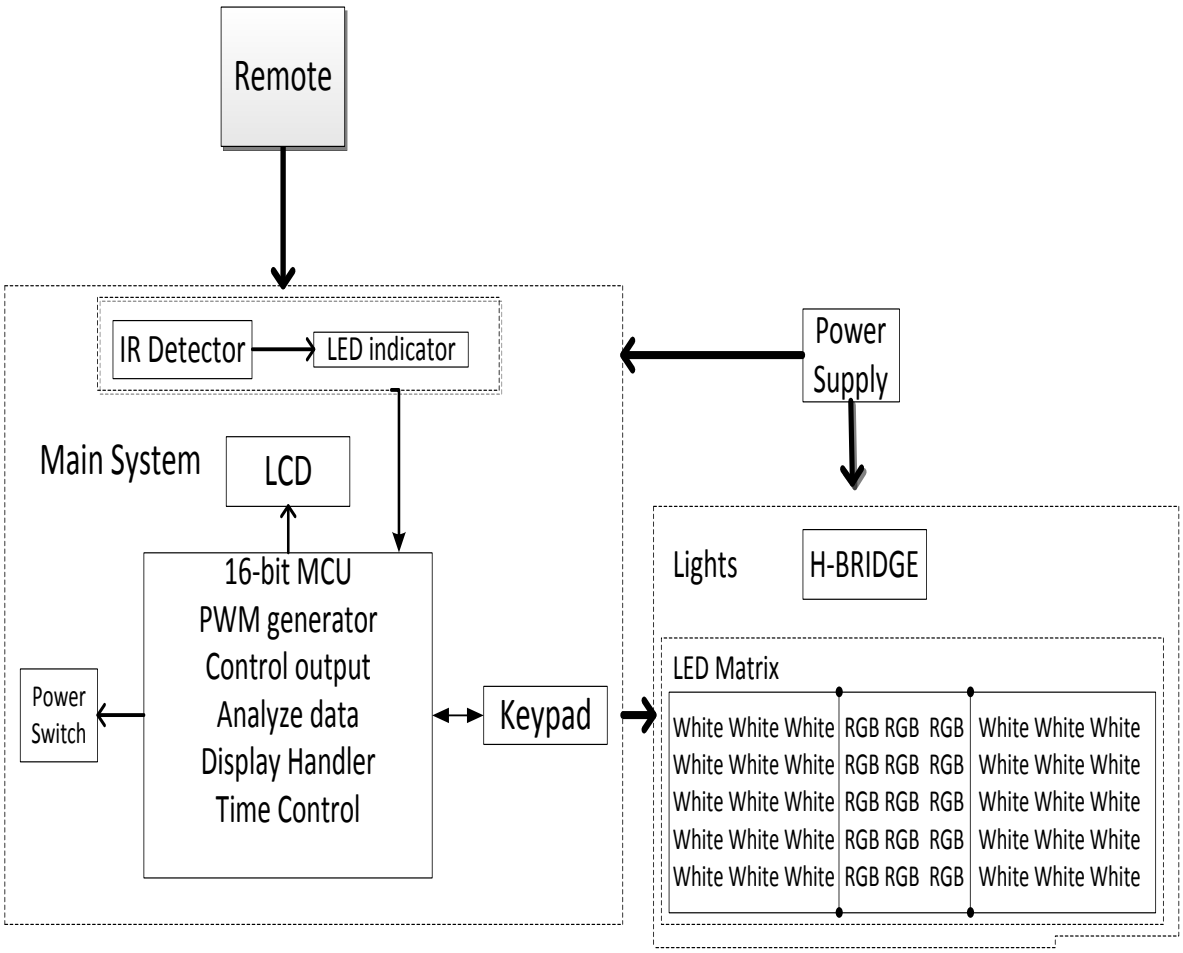


Figure 4: Level Two

This above figure demonstrates the Level Two design and provides a greater insight into the design of the device. It shows a power switch which will be used to cut the power supply going to the LEDs. This would able the user to turn off the light and have the microcontroller on and show the current time on the LCD. A user can unplug the power cord to fully turn off the device. It also shows the LED matrix in more detail and gives an overview of design of LEDs as well. There are three rows of cool white LEDs, three rows of warm white LEDs, and three rows of RGB LEDs. The combination of these LEDs would be used to produce the desired color of sunlight and the amount of brightness. Since, every single LED will not be on at every instance in time it will also help to control the power consumption of the LEDs as well. The

white LEDs produce the highest amount of lumens with the same power consumption of 1W compared to RGB LEDs. The graph below gives an overview of the different LEDs being on during different times of the day.

One Full Cycle of Light Transition

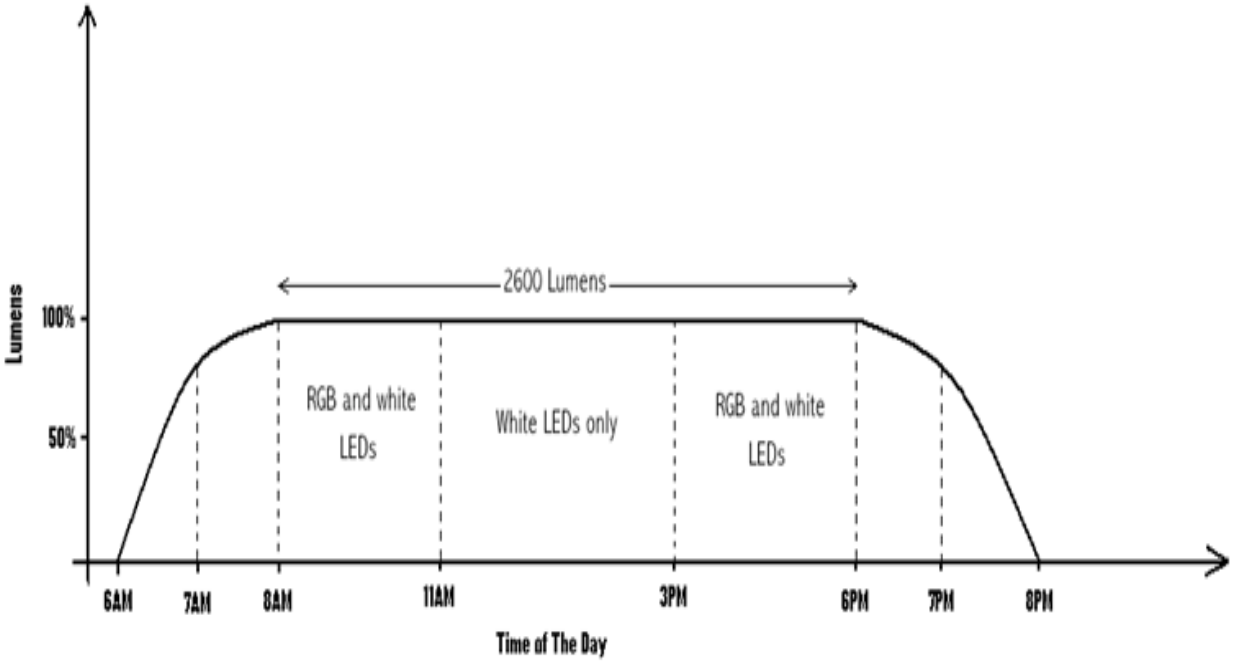


Figure 5: One Full Cycle of Light Transition

Figure 5 above illustrates the implementation in natural light from dawn through dusk using an LED light source. At dawn only RGB LEDs will be used to simulate that time of day. As the day approaches afternoon cool and warm white LEDs will be used to imitate afternoon sunlight. Figure 5 is a general outline of how this system will work if started in the morning and allowed to continue until the evening. But the user can set the system at any time of day. For example the user can set the time to 6am, yet the user's current time is 8pm. The user can also specify the duration of the simulation. So he can set the lamp time to 6pm and run it for 2 hours at the user's current time of 10pm. The graph in figure 5 can be changed based on the time set.

Components of the Device

Power Supply:

The circuit shown below will be used to supply power to the device. This circuit will take 120V AC at the input side and it will output 30V DC with a 1.5A current at the output side. A voltage regulator will be used in this design to convert the rectified signal into a pure DC voltage at the output. An adjustable voltage regulator will be used to be able to adjust the output voltage to the desired value. This power supply should be sufficient to power our device properly.

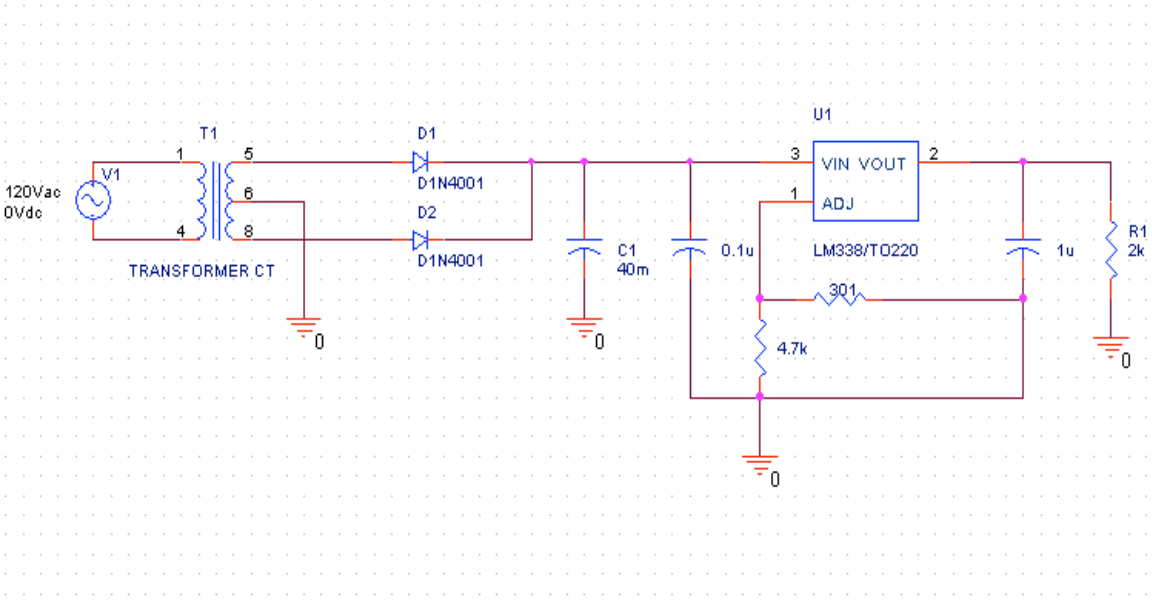


Figure 6: Power Supply Circuit

Microcontroller

Figure 7: Msp430FG4618

A Microcontroller is nothing more than a small computer on a single integrated circuit with some processor core, memory and input/output peripherals. Most of the applications used for microcontrollers are embedded applications. They are designed to run one task and the program is stored in the ROM and generally does not change unless it is being programmed again. Furthermore, they are low-power devices and consume less power.

There are several vendors for microcontrollers that are available in the market these days. Some of these vendors are Microchip providing microcontrollers from 8-bit to 32-bit, Infineon 8-bit to 32-bit, Texas Instruments (16-bit), Atmel AVR 8-bit and 32-bit. The microcontroller that is being used for this project is Texas Instruments TI MSP430FG4618. This microcontroller provides a low supply voltage range 1.8V to 3.6V, 16 bit RISC architecture, two 16-bit timers, real time clock, Universal Serial Communication Interface, 116KB Flash and 8KB RAM. This microcontroller should be able to provide enough PWM signals for all the H-Bridges and hence, should be able to control the LEDs. This microcontroller also has enough output pins to easily connect with the LCD and the IR detector. The real time clock will help to remember the time entered by the user and operate based on this time. Hence, this microcontroller should be able to provide the necessary features for this project.

H-Bridge

An H-Bridge is a motor-driver often used to provide high current to DC motors than a microcontroller can provide. In our project, we will be able to provide higher current to the LEDs using an H-Bridge. There are several different H-Bridges that are available in the market and the H-Bridge for our project is SN754410 Dual H-Bridge Motor Controller (see figure below). This is a dual H-Bridge so it can provide output to 2 columns of a LED matrix. The reason we are using this h-bridge is because we have prior experience of working with them to drive motors. It was efficient enough to supply 1 Ampere current to the motors and it could be re used to supply 350mA to the LEDs we are using.

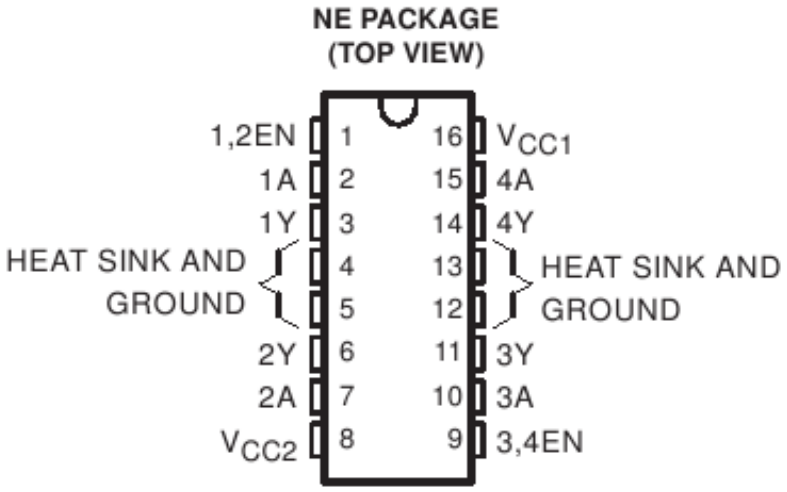


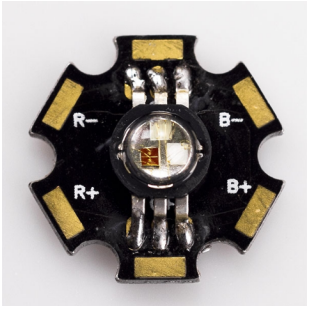
Figure 8: H-Bridge

Liquid Crystal Display

An LCD is required in this project to serve as a visual reference to the user. The LCD will be mounted directly on the torchiere lamp and show different parameters such as brightness, color and time of the day to the user. It will also show a menu which will guide the user to change the settings of the device. The LCD for this project is NEWHAVEN Display part number 0420AZ-FSW-GBW (see figure below). This LCD requires a +3.0 V supply in order to properly operate. It features a 4 lines * 20 characters and hence, giving enough space to draw the menu and show parameters. It also has a white LED backlight which can be adjusted to change the contrast of the LCD.

Figure 9: LCD

Light Emitting Diodes



RGB Led Cool/Warm White Led

Figure 10: LEDs

There are different types of LEDs that are being used for this project. These LEDs are RGB, cool white and warm white. The RGB LEDs has a power consumption of 3W as there are three LEDs (red, green, and blue) mounted together in one place. Each LED consumes about 1 W because with a forward voltage of 3.4 – 3.8 V for Blue/Green LEDs and forward current of 350 mA giving approximate power consumption of 1.19 W to 1.33 W for single LED. The power for Red LED is (2.5 – 2.8 V) * 400 mA being equal to 1 – 1.12 W. Hence, the total power consumption for a RGB LED goes from 3.38 to 3.66 W. The RGB LEDs when fed with the right PWM signals would be able to produce necessary colors required to imitate sunlight. The cool white and warm white LEDs have power dissipation of 1W each. These LEDs would mostly be on during mid-day in order to imitate a white-sunlight and parts of these LEDs would be on during the morning time and evening and act as a helper to the RGB LEDs towards producing the desired colors. The table below shows different parameters of the LEDs in great detail.

	Warm Whites (1W)	Cool Whites (1W)	RGB (3W)
Forward Voltage	3.0 – 3.8 V	3.0 – 3.8 V	Blue/Green 3.4 – 3.8 V Red 2.5 – 2.8 V
Forward Current	350 mA	350 mA	Blue/Green 350 mA Red 400 mA
Emitting Angle	120 – 140 °	120 – 140 °	120 °
Lumens	65 – 75 lumens	76.9 – 87.9 lumens	Red 60 lumens Green 55 lumens Blue 20 lumens

Table 1: LEDs parameters

User interface

For a user interface we implemented a remote control and an on-board control. We used an IR detector to remotely control our lamp. IR detectors are microchips tuned in to listen to infrared light. They are used for remote control detection. The remote control emits IR pulses using matching IR LED tuned to a specific frequency. The IR detector receives and analyzes the pulse to run a particular command. IR LED inside the remote have to be pulse width modulating at a specific frequency in order for the IR detector to detect that frequency. So for our project we used an IR detector tuned to a frequency of 38 kHz. The remote control we used is a Sony Remote control that is specifically programmed for a radio. The specific model of the IR detector is GP1UX311QS, it can filter and demodulate incoming infrared signal. It also can receive signal from 13 feet away. It can be easily integrated to a microcontroller with a three-pin output. Once the user presses a button on the remote control the IR detector processes data and implements the command. We integrated the IR detector with the microcontroller so every time a specific button is pressed the lamp output is changed. So we used the button to control brightness and the color of the LED light source. At later point we can also use the remote to input the specific time that the lamp should be on for. The remote will be also used as a power on/off switch for the lamp. The remote we are using is a Sony remote control. Each remote control manufacturer uses different protocols to modulate infrared signal. Sony uses SIRC (Serial Infra-Red Control) protocol. This protocol uses pulse width modulation to encode the bits. The remote we are using is a 13-bit protocol. It has a 5-bit device code that identifies the specific device to be used on. It also has a 7-bit command code that represents the actual button pressed on the remote control.

Figure 11. IR Detector

The protocol starts off with a start bit that is 2.4ms long to signal the start of the SIRC message. A typical pulse modulation of an SIRC protocol is as shown below.

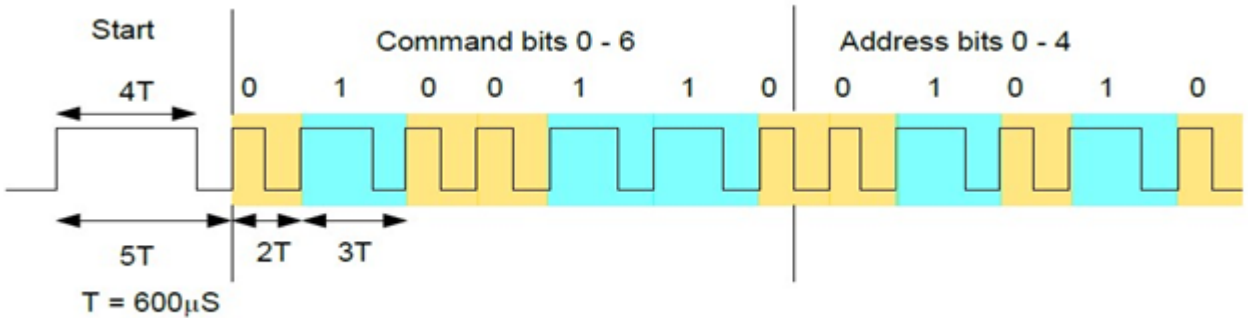
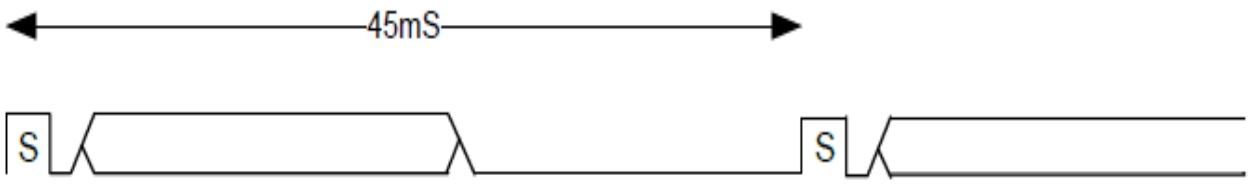


Figure 12: SIRC Message

The pulse modulation is based on multiples of .6ms. The data is modulated from signal with the least significant bit first as shown in figure 4. The pulse representing a bit '1' is a 1.2ms long burst of the 38 kHz carrier signal, while the burst width for bit '0' is .6ms long. All bursts are separated by .6ms. All SIRC messages are repeated every 45ms.



The possible use of the buttons on the remote is specified in the table below.

Command	Button	Light Source Integration
0	Digit key 1	Time/ MENU select
1	Digit key 2	
2	Digit key 3	
3	Digit key 4	
4	Digit key 5	
5	Digit key 6	
6	Digit key 7	
7	Digit key 8	
8	Digit key 9	
9	Digit key 0	
16	Channel +	(increase color output)
17	Channel -	(decrease color output)

18	Volume +	(increase brightness)
19	Volume -	(decrease brightness)
21	Power	(Turn on/off lamp)

Table 2: Command Table (Remote)

We will connect a keypad on the light source; just in case the remote control breaks the user can use the on-board control. The keypad will be integrated directly on the Torchiere lamp and used in the same way the remote control is used.

State Diagram

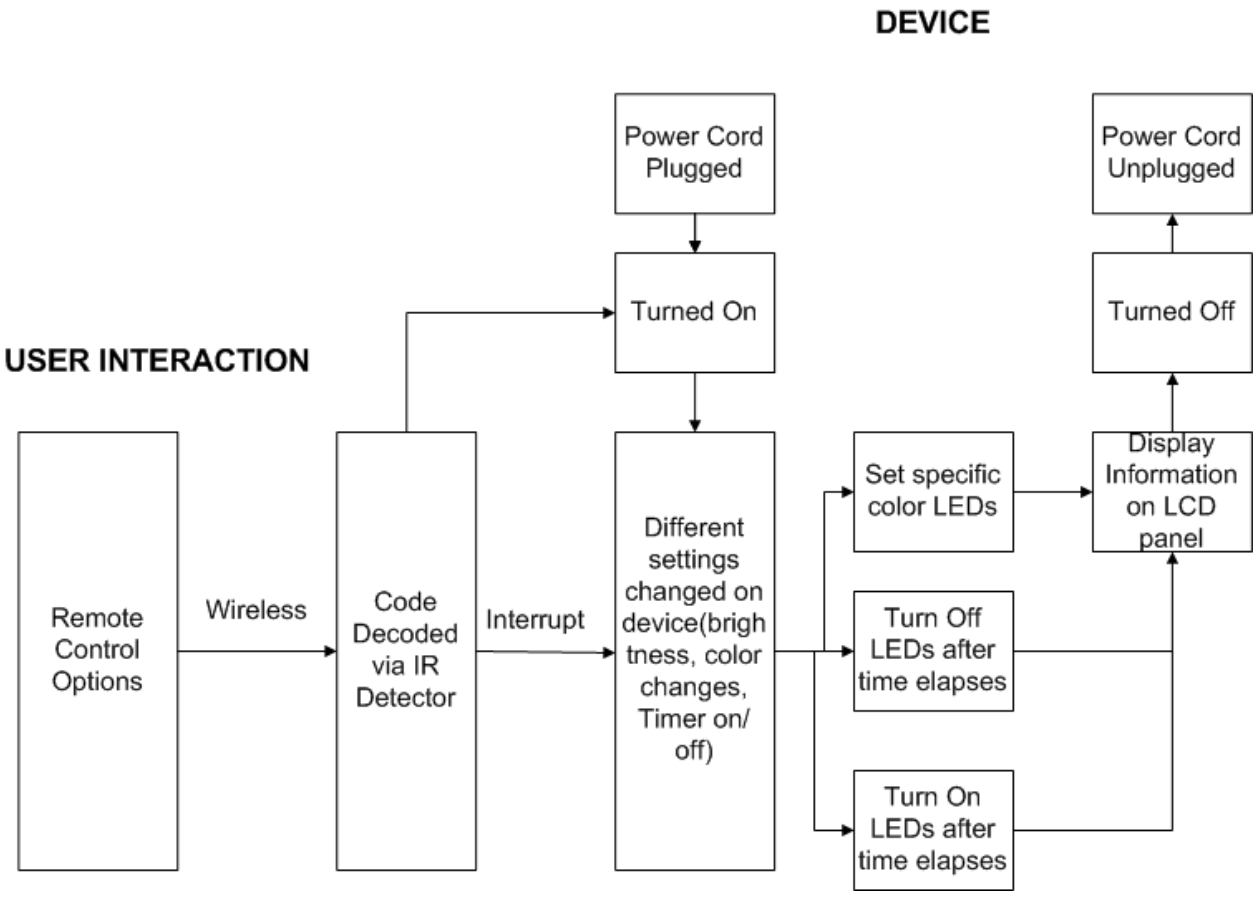


Figure 13: State Diagram

The figure above shows a brief overview of the operation of the device and different parameters associated with it. The device would start up in automatic mode and follow the figure shown above (figure

5: One Full Cycle). It would imitate the behavior of sunrise as start-up. Since, the device is user-controllable, the user has the full ability to change different parameters such as the time of the day, brightness level, color level, and time to set it to sleep and turn on. The user would select these options through the remote and this signal would go to the IR detector through infrared and then wired to microcontroller as shown in the figure above. The microcontroller would understand this signal and behave accordingly. It would turn on/off LEDs; change their brightness level and/or their colors. As the figure above shows, the different parameters would also get shown on the LCD as they are being changed. This will give the user a visual reference and will make it easier to control the device.

Color Spectrum:

From dawn to dusk there is a smooth transition of natural light color from warm(yellow) to cool (blue). To achieve this transition of color spectrum the color temperatures and their respective RGB values will be considered as follows**.**

Color	Color Temperature	RGB Values	Duty cycle(LEDs)
Candle	**1900**	**255, 147, 41**	**100, 58, 16**
Warm White	**2600**	**255, 197, 143**	**100, 77, 56**
Incandescent	**2850**	**255, 214, 170**	**100, 84, 67**
Halogen	**3200**	**255, 241, 224**	**100, 95, 88**
Direct Sunlight	**5200**	**255, 250, 244**	**100, 98, 96**
Day White	**5400**	**255, 255, 251**	**100, 100, 98**
Cloudy Sky	**6000**	**255, 255, 255**	**100, 100, 100**
Cool White	**7000**	**201, 226, 255**	**79, 87, 100**
Blue Sky	**20000**	**64, 156, 255**	**25, 62, 100**

Table 3: Color Spectrum Values

Light Meter:

Mastech Digital Light Meter LX13300 with a range from 0 - 200,000 LUX will be used to measure the amount of light produced. The sensor of this meter would be placed right above the LEDs which would display the number of lux it is producing. The amount of lux will be converted to Lumens by using formula as follows.

Figure 14: Light Meter Early

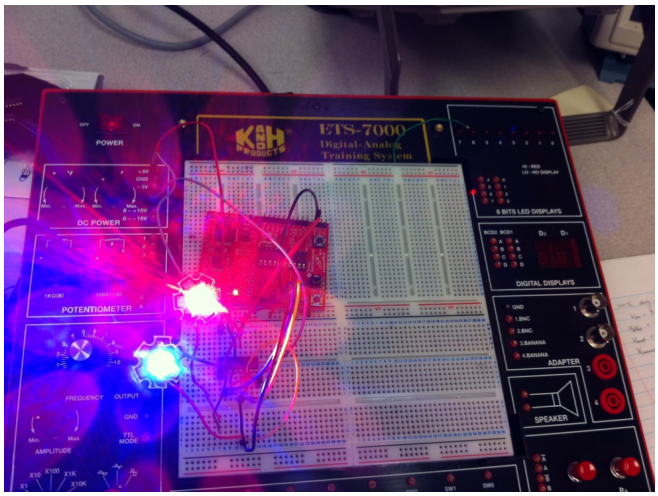
1 lux = 1 lumen/square meter

Prototyping:

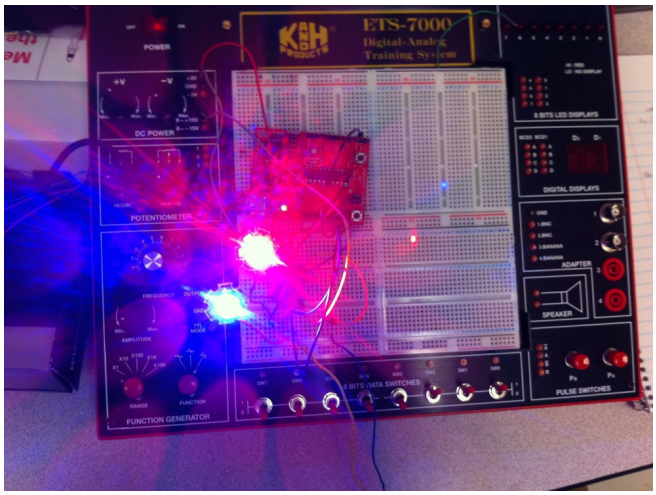
LED Circuit

This circuit was built to check if the high intensity LEDs is working in parallel with the H-bridge and microcontroller.

Next they were controlled via a PWM signal with 100%-60% duty cycle and the brightness, current and voltages were observed across each duty cycle.



100 % duty cycle



90 % duty cycle

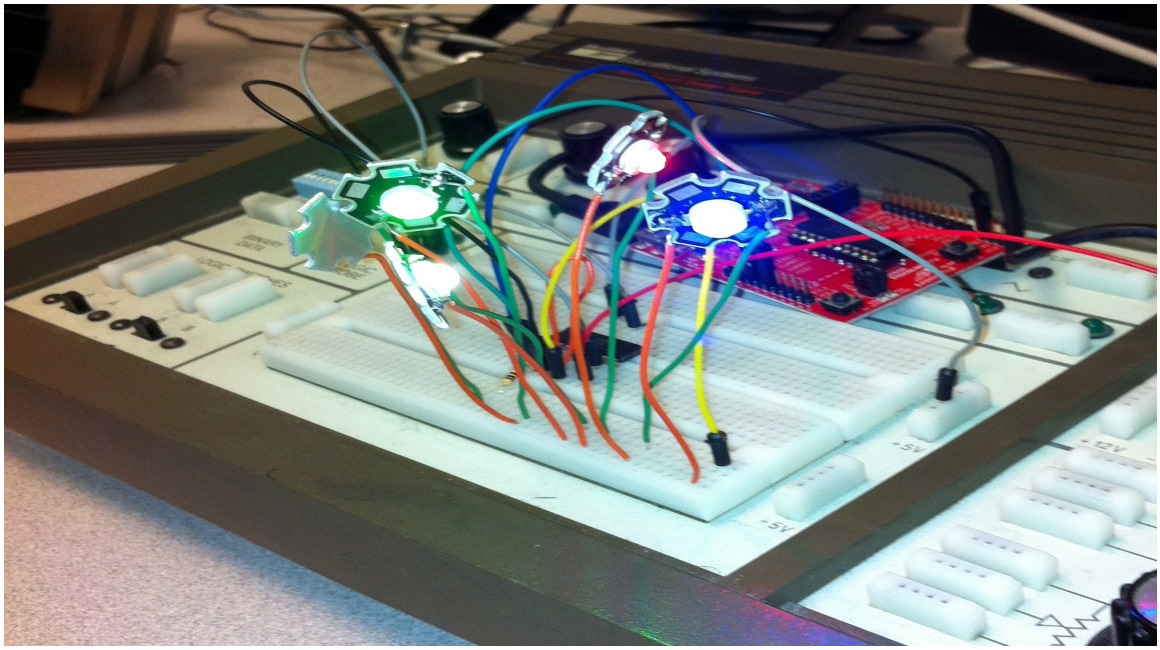


Figure 15 LED Prototyping

We learned from this experiment that the LEDs get hot and providing accurate voltage and current to the LEDs needs more research. We also need an accurate means of measuring light intensity.

LED Code

```
#include "io430.h"

int main( void )
{
  // Stop watchdog timer to prevent time out reset
  WDTCTL = WDTPW + WDTHOLD;

  P1DIR |= (BIT2|BIT4);     // P1.2 to output
  P1DIR |= (BIT5);          // P1.5 to ouptut
  P1SEL |= BIT2;            // P1.2 to TA0.1
  P1OUT |= BIT4;            // set bit 3 high for H-bridge
  P1OUT &= ~(BIT5);
  CCR0 = 1000-1;            // PWM Period
  CCTL1 = OUTMOD_7;         // CCR1 reset/set
  CCR1 = 600;               // CCR1 PWM duty cycle 25 % duty cycle
  TACTL = TASSEL_2 + MC_1;  // SMCLK, up mode

  while(1);
  return 0;
}
```

IR prototype:

The IR detector was tested using TI Launchpad MSP430G2553. A Sony remote was used. A circuit was implemented using two led indicators and two resistors. The IR detector has three pins, one for ground, power and input to microcontroller. The input was connected to one of the port on the Launchpad, and signal received by the IR Detector was analyzed using a timer on board the Launchpad. Based on how long the signal remained on the rising edge; the specific bit representation was generated. Then the bits were combined and based on the commands (or button they represented) an action took place. Every time a button was pressed the LED indicators lit up to signal that the IR detector had received an infrared signal from the remote control. The code used for the prototype is shown, it basically measures the pulse and assigns bit '1' or '0' based on the remote control encoding. It outputs the specific LED indicator for a button press. From this experiment we learned that repeating pulses from the remote control every 45ms is going to be a problem to research.

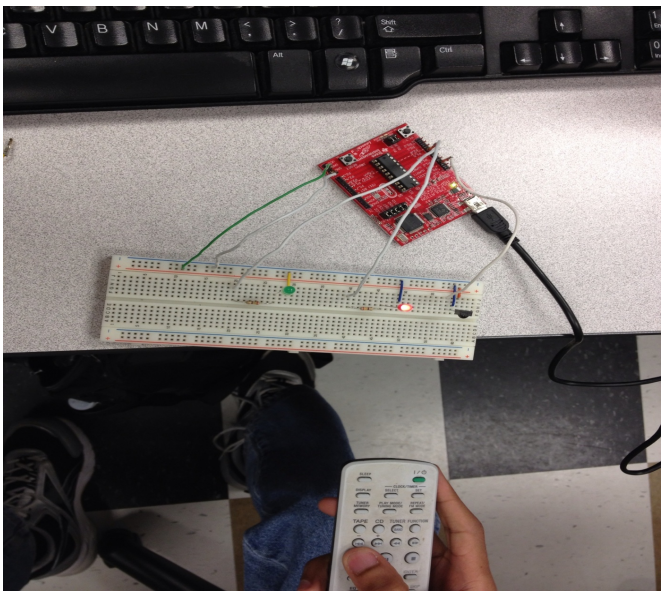


Figure 16: IR Prototype

IR Detector Code

```
#include "msp430g2553.h"

#define duration   300
#define T65   duration*3
#define T2   duration*4
#define T3   duration*6

unsigned int IRData = 0;            // received signal
unsigned int pwmCounter = 0;
unsigned int start=0;
void main(void)
{
  WDTCTL = WDTPW + WDTHOLD;          // stop WDT
  BCSCTL1 = CALBC1_1MHZ;             // set range
  DCOCTL = CALDCO_1MHZ;              //set DCO setup and modulation

  P1DIR &= ~BIT1;                  // P1.1 input
  P1SEL = BIT1;                  // P1.1 Timer_A CCI0A

  P1OUT &= ~(BIT4 + BIT5);        // P1.4-1.5 out
  P1DIR |= (BIT4 + BIT5);

   TACTL = TASSEL_2 | MC_2;             // SMCLK, continuous mode
   CCTL0 = CM_2 | CCIS_0 | CAP | CCIE;     // falling edge capture mode, CCI0A, enable IE

  __bis_SR_register(LPM0_bits + GIE);     // switch to LPM0 with interrupts
}

#pragma vector=TIMER0_A0_VECTOR
__interrupt void Timer_A (void)
{
  if(CCTL0 & CAP) {               // start bit
    pwmCounter++;                 // start counting bits
    CCR0 += T65;                // add 6.5 bits to counter
    CCTL0 &= ~ CAP;                 // compare mode
  } else {
    switch (pwmCounter) {
      case 0x1000:              // received all bits
        pwmCounter = 0;           // reset counter
          // process received data, toggle LEDs, turn on power
          switch (IRData & 0x001F) {   // mask device number
            case 17:        //
              if((start)==1){
                P1OUT ^= BIT4;}
              break;
            case 16:        //
              if(start==1){
                P1OUT ^= BIT4;}
              break;
            case 18:        //
              if(start==1){
                P1OUT ^= BIT4;}
              break;
            case 19:        //
              if(start==1){
                P1OUT ^= BIT4;}
                    break;
            case 15:        //   Volume -
              if(start==1){
                P1OUT ^= BIT4;}
              break;
            case 14:        //   Volume +
              if(start==1){
                P1OUT ^= BIT4;}
              break;

            case 21:        //   Power
              P1OUT ^= BIT5;
              start^=1;
              if(start==0)
                P1OUT &=~BIT4;
              break;
          }
          IRData = 0;
          // end process received data
        CCTL0 |= CAP;           // capture mode
        break;
      default:                // data bit
        if (CCTL0 & SCCI) {        // bit = 1
          CCR0 += T2;           // add 2 bits to counter
        } else {              // bit = 0
          IRData |= pwmCounter;   // set bit of IRData
          CCR0 += T3;           // add 3 bits to counter
        }
        pwmCounter <<= 1;         // increase (shift) bit counter
        break;
    }
  }
}
```

Experimentation

- Experiment #1
  - Goal: To evaluate remote access of LED light source
  - System components: Remote, IR ,LCD and microcontroller
  - Testing process: System configured via microcontroller. IR connected to microcontroller. User uses remote to input choices to system. Process repeated until an accurate user input is registered.
  - Testing completed to get an accurate reading from the user. User inputs are verified by the output on the LCD. Modes of operation of light source tested using remote.
  - Data evaluated for erroneous readings, skipping inputs and malfunction in remotely controlling the remote.
- Experiment #2
  - Goal: Generate a spectrum of colors using the varies LEDs
  - System components: All components
  - Testing process: System configured via microcontroller. Specific PWM are generated to output a range of colors. Light the different LED and verify color output. Use user input to generate different color outputs. Lumens meter to test lumens output.
  - Testing verified to represent an accurate color spectrum. User inputs are verified by the output on the LCD. Make sure that microcontroller output is fully represented by light source output. Maximum of 2600 lumens output from LED matrix.
  - Check color spectrum output. Output should represent different colors. Brightness and color output should be independent verified by spectrum output.
- Experiment #3
  - Goal: Verify power supply compatibility with system and power consumption
  - System components: All components
  - Testing process: Measure power consumption using multimeters and do some circuit analysis. Run the system completely and measure voltage output from each component.
  - Testing verified if Light source is 150W or less. Comparison with a regular light source to check for power consumption. Generation of a safe power output.
  - Output evaluation focused on comparison with regular light source power output. Verification of power supply voltage output to run the system.

Major Task List:

1.1.Remote control access (4 WEEKS)

a. IR integration with microcontroller
b. Remote compatibility with IR
c. Pulse reading using MCU software
d. Handling user interrupt

1.2.Power circuit design (3 WEEKS)

a. Generate power for components
b. Integrate with microcontroller
c. Circuit design

1.3.Measurement (3 WEEKS)

a. Color temperature
b. Light intensity
c. Power consumption
d. Light spectrum comparison

1.4.System Development (7 WEEKS)

a. Build control panel
b. Program LCD and test
c. Microcontroller programming
d. Create Vector board
e. Final circuit implemented on PCB
f. Integrate into lamp

1.5.LED light source development (7 WEEKS)

a. Light different LEDs
b. Use MCU to control color output
c. Brightness control using PWM

d. Test using Microcontroller

1.6.System Integration (7 WEEKS)

a. Power supply w/ microcontroller

b. Remote control w/ microcontroller & LCD

c. Keypad w/ MCU and LCD

d. Remote control w/ system

e. LEDs w/ MCU

f. All components w/ power switch

1.7.Testing

a. Experiment #1

b. Experiment #2

c. Experiment #3

1.8.Reporting

a. Progress Report #1

b. In-progress report

c. Final report

1.9.Milestones/Demos

a. In-progress presentation demo

b. Demo to FS

c. Final presentation

Allocation of Tasks:

18- Research how to measure color temperature and light intensity (Marzia)

19- Research and test the power supply circuit independently (Maryam)

20- Built and test the control panel (Girmay)

21- Programing and test the LCD screen. (Marzia)

22- Build the User Interface and sync it with the microcontroller/LCD (Girmay and Marzia)

23- Integrate the microcontroller with LEDs/program LEDs with different behaviors (Mayank)

24- Build the LED circuit on breadboard/vector board (Maryam)

25- Test the final circuit, and design the PCB (Maryam, Mayank, Girmay, and Marzia)

26- Build the PCB (Marzia will order)

27- Build the containing lamp (Maryam and Mayank)

28- Buy the connectors, outlets, cables, and build the unit (Marzia)

29- Test the final product (Maryam, Mayank, Girmay, and Marzia)

Fall Semester Schedule:

Week 1: Project Title Form delivery. Project Title Form delivery. Test plan (cases) and WBS delivery.

Week 2: Start building power supply.

Week 3: Start programming and testing the LCD screen.

Week 4: Create Parts List. Show parts list to FS for approval. Finalize LED matrix design and start building the circuit on vector board.

Week 5: Progress Report #1 delivery. Finalize design of the IR detector circuit and begin building it. Start program microcontroller and ensure it syncs with microcontroller

Week 6: Testing the circuit individually and making sure all are working correctly. Show built circuits to FS for approval and testing.

Week 7: Start putting the individual components together and test them.

Week 8: Finalizing the circuit design and start designing the PCB.

Week 9: Finalize software design. Finalize lamp Design. Order the PCB.

Week 10: Start installing the PCB and LCD matrix into the torchiere lamp.

Week 11: Testing the final design.

Week 12: Draft Final Report delivery.

Week 13: Finishing the project and start working on Final Report presentation and report.

Week 14: Oral Presentation. Final Report delivery. Document Tracking Form delivery.

Week 15: Project Poster delivery.