ADAPTIVE LIGHT DIMMER

by

William Xiong, Jonathan Sato

Senior Project

ELECTRICAL ENGINEERING DEPARTMENT

San Luis Obispo

2017

TABLE OF CONTENTS

Figures *Page*

Abstract

This project, the Adaptive Light Dimmer for the DC House, senses a room's light content and adjusts a lamp's light intensity accordingly, thus regulating the room's light content. The device is designed to work with renewable energy sources such as wind and solar energy. This would be useful in less developed countries where AC electricity is not well spread and renewable DC, such as solar, can be better utilized. It functions by using the TSL2561 light sensor to sense light, ATTiny85 microcontroller to output PWM to the LED driver, LT3795 LED driver to output current to an LED and LT3014 low dropout regulator to lower the input voltage and power the microcontroller and sensor. The dimmer is designed to work with a 48V input voltage and an LED. It operates from an input light range of 20 to 100 Lux. Above 100 Lux the light is off and below 20 Lux the light is fully on. The efficiency of the device is 48.7%, which can be increased with improved design.

Chapter 1. Introduction

According to the International Energy Agency (IEA), it is estimated that 1.2 billion people around the world did not have access to electricity in 2013. This is about 17% of the total population of the world with most of these people being concentrated in developing Asian countries, the Middle East, and sub-Saharan Africa [1]. The trend was similar in 2012 as illustrated in Figure 1-1. On top of this, many who already have access to electricity, the quality of the electrical power is still considered poor and unreliable. While advancements are being made to give more people access to electricity, progress is still necessary to reach a point of worldwide electrification.

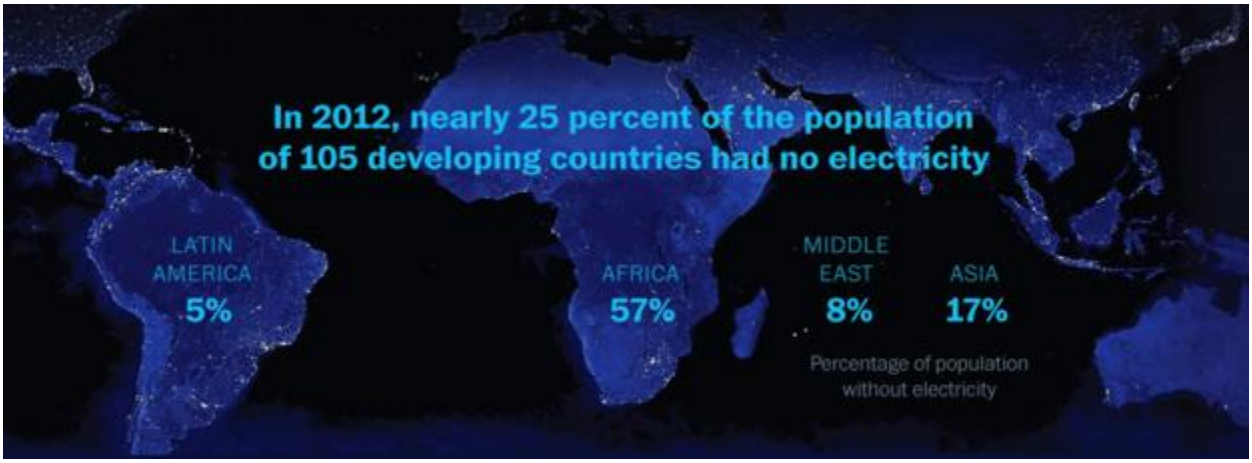


Figure 1-1. Percentage of Primary Countries Without Electricity [2]

One such effort to improve access to electricity especially in rural areas is the DC House Project. The project was initiated at Cal Poly State University in 2010. The DC House aims to establish electrical system in a house powered by direct current (DC) as opposed to the conventional alternating current (AC). It has two main purposes. The first is to act as a potential

model for providing electricity for people in rural areas where electrical infrastructure is lacking. The second is to promote the use of renewable energy sources and show that they can be practical. It runs solely on direct current generated from various sustainable sources such as solar, wind, and other renewables. For the DC House to be a successful model, it has to be able to provide enough electricity to power household essentials such as indoor and outdoor lighting, food storage, food preparation, and other necessities [3].

In regards to efficiently using sustainable energy, there is a significant amount of data supporting the energy saving capabilities of light dimming technology. Some state that smart lighting, using sensors and controllers to control lighting, saves between 50% and 70% of energy compared to an uncontrolled lighting system [4]. These energy savings have huge impacts, considering that lighting accounts for about 19% of the electrical energy generated worldwide. In commercial buildings, lighting accounts for even more, 30-40% [5]. This being said, smart lighting seems to be the next logical step in saving electricity and tackling the problems of world-wide electrification in today's world.

In order to improve smart lighting technology, it makes sense to utilize electrical devices, such as microcontrollers, to increase efficiency and usability of lighting systems. Microcontrollers would be a reasonable choice due to their decreasing costs, versatility, and ease of use. In addition to microcontrollers, DC-DC converters are also essential to improving smart lighting systems. These converters are the most efficient ways of converting DC power and consequently are used in many stages of power conversion. Without them, electronics, such as lighting systems, would be much more inefficient. Any improvements to these DC-DC converters directly improve the efficiency and performance of smart lighting systems.

Chapter 2. Background

Despite the growing number of people gaining access to electricity there are still those living without electricity or with limited access to electricity, particularly in the third world and developing countries. In these areas electricity is either unreliable or limited in its availability. Electrical infrastructure is less than adequate. An article in the Washington Post sums up the problem of people "living in the dark". According to this article, places such as India and Africa, where most of the people without electricity reside, have programs or intend on having programs that will utilize "mini-grids" using renewable energy sources. However, due to the large demand, they have been forced to increase their fossil fuel use [2]. While most of the urban areas have a higher percentage of people with power than in rural areas, there are still many people without access to power. In order to combat this, the DC House runs off of renewable energy independent from the grid and therefore needs to conserve the electricity it produces. To limit the use of electricity, a sort of light dimming device is a practical next step.

A DC House would be the perfect application for this project, although the project could also find use in other applications, particularly those that employ of renewable energy sources such as solar power. Unlike some other forms of energy generation, photovoltaics produce DC power. In most applications, this power must then be processed by converters to produce the desired AC power. Each of these various converters produce significant losses. However, for our project solar energy can more directly power the lighting system, decreasing the number of converters needed which in turn will decrease the amount of energy loss.

Because of the lack of energy available to many inhabitants of third world countries, many of those countries have begun to focus their efforts on renewable energy resources. For

example, according to Dinesh Kathaiyan, using photovoltaics to provide power to rural villages in India is a great way to increase accessibility to electricity for rural inhabitants [6]. Because of the lower population density of these rural areas, the energy demand is also lower. As a result, renewable energy sources, such as solar cells, are more capable of providing sufficient power to them without relying on the grid. Because a large portion of the energy will come from solar power, they will be DC in nature. Our project could then be a very important technology for these villages, as its abilities to directly use DC power and to autonomously dim lighting systems both reduce energy losses. Minimizing these losses are important both for costs and for the sustainability of these renewable energy resources.

There are existing solutions for AC lighting in the form of smart lighting and socket to bulb interfaces. Smart lighting uses “smart systems” to control lighting with applications in computers, tablet devices, or smart phones. On a larger scale, they can also be used to control the lighting system of a room, house, or building. They can wirelessly turn on and off lights, control their brightness, set the lights on a timer, and even integrate sensors. Existing socket to bulb interfaces use a sensor in an attempt to dim the bulb appropriately but they do not function very well. An example of an AC powered smart lighting device, called a home light control module (HLCM) and designed by Ying-Wen Bai and Yi-Te Ku, uses passive infrared (PIR) sensors, light sensors, a microprocessor, and an RF module to control light intensity in all the rooms of a house [7]. A single HLCM controls one set of luminaires. As a result, multiple microprocessors determine lighting levels, rather than a single central controller. This device uses the PIR sensors to determine the presence of any people in a room, turning off the lights if no one occupies the room. The light sensors determine the room's brightness levels. If outside sources, like daylight, provide enough light, then the luminaires are turned off. Otherwise, the system activates the

appropriate amount of luminaires to achieve the desired brightness levels. The device's RF module allows communication between different HLCMs. In the case that brightness levels are insufficient even when all luminaires are on, communication between HLCMs allows an adjacent HLCM to increase the number of lights to activate. This then affects the light intensity of the first device's room. While these components help this device operate with high efficiency, our project need not contain any unnecessary modules if their inclusion greatly increases purchase costs. For example, we would not need the RF module if we find its inclusion produces costs that outweigh its energy benefits. Additionally, the HLCM does not use light dimming technology while our project is based on light dimming.

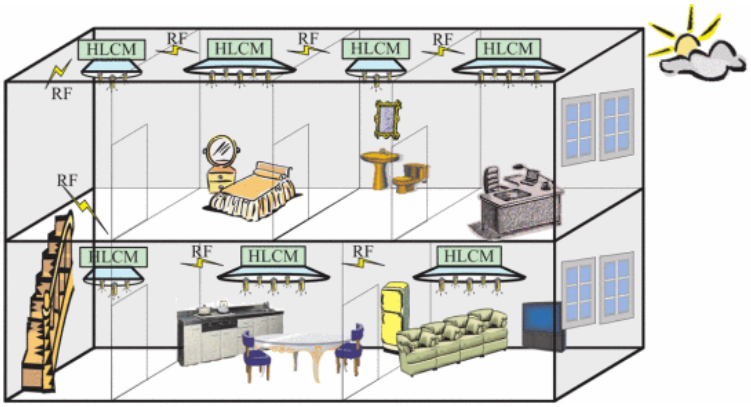


Figure 2-1. The configuration of an HLCM system for a 2 story house [7]

However, there are currently no existing DC dimmers that are inexpensive and entirely autonomous. The objective of this project is therefore to design and construct an adaptive DC light dimmer that autonomously dims a set of LEDs depending on the ambient light sensed in the room, providing the appropriate amount of light to the environment it is in. The end product of this project will save electricity usage; thus, reducing electrical costs and increasing the viability of renewable energy.

Chapter 3. Design Requirements

The adaptive light dimmer must be able to produce the same or more light than a normal 60W incandescent light bulb at maximum brightness. This was determined to be roughly 800 lumens. If the adaptive light dimmer is incapable of producing the appropriate amount of light, a house may not be bright enough during darker hours of the day.

Another important aspect is the efficiency of the device. Having better efficiency leads to less costs in terms of both energy and money. While both are definitely important, reduction in energy costs will likely have a greater impact on those rural areas depending on renewable energy sources, since these energy resources are much more limited. The adaptive light dimmer should have at least an efficiency of 70% at full load.

In addition to this, the device should consume a relatively low amount of power as well. Considering that the LED being used is a 5W rated LED, the maximum output power must be 5W. Derived from the efficiency requirement, the maximum input power must be 7.14W.

The device must be able to operate from a 48V input, as this is the voltage provided by the DC house. It should ideally be able to operate at other DC voltages as well for other applications.

The device must also be quick to respond without causing abrupt changes. So, the device should be able to adjust its output at least once every two seconds. Each adjustment may need to be gradual for any sudden changes in the surrounding brightness levels.

The sensor should be sensitive enough that it can differentiate changes in daylight throughout the day. The sensor measures light intensity in lux, so the placement of the sensor can have a significant effect on measurement values. The best place to put the sensor is near the

floor, as this minimizes the vertical movement of the sensor. Near the floor, the expected lux measurements range from 35 to 150 lux.

In terms of costs, the materials of the device should cost less than $100. The device should be as cheap as possible, to allow its use in rural areas that will benefit most from the technology.

The device will be connected to a lamp, so the entire system will be fairly large. Approximately, the height must be around 4 to 5 feet, and the base should be a foot or so in diameter. The height will also help to spread the light to the entire room.

The weight of the system will come mostly from the lamp, so it will weigh around 5 lbs. A lower weight improves the device's mobility, allowing users to relocate it to a different room when needed.

Figure 3-1. Level 0 Block Diagram

The level 0 block diagram as shown in Figure 3-1 describes the overall function of the system. At this level, the adaptive light dimmer acts a "black box" with its individual components and inner functionality hidden. It takes in two inputs, ambient light and power from the 48V DC bus. The output is the light from the led ranging from 0 to 800 lumens depending on the ambient light entering the adaptive light dimmer.

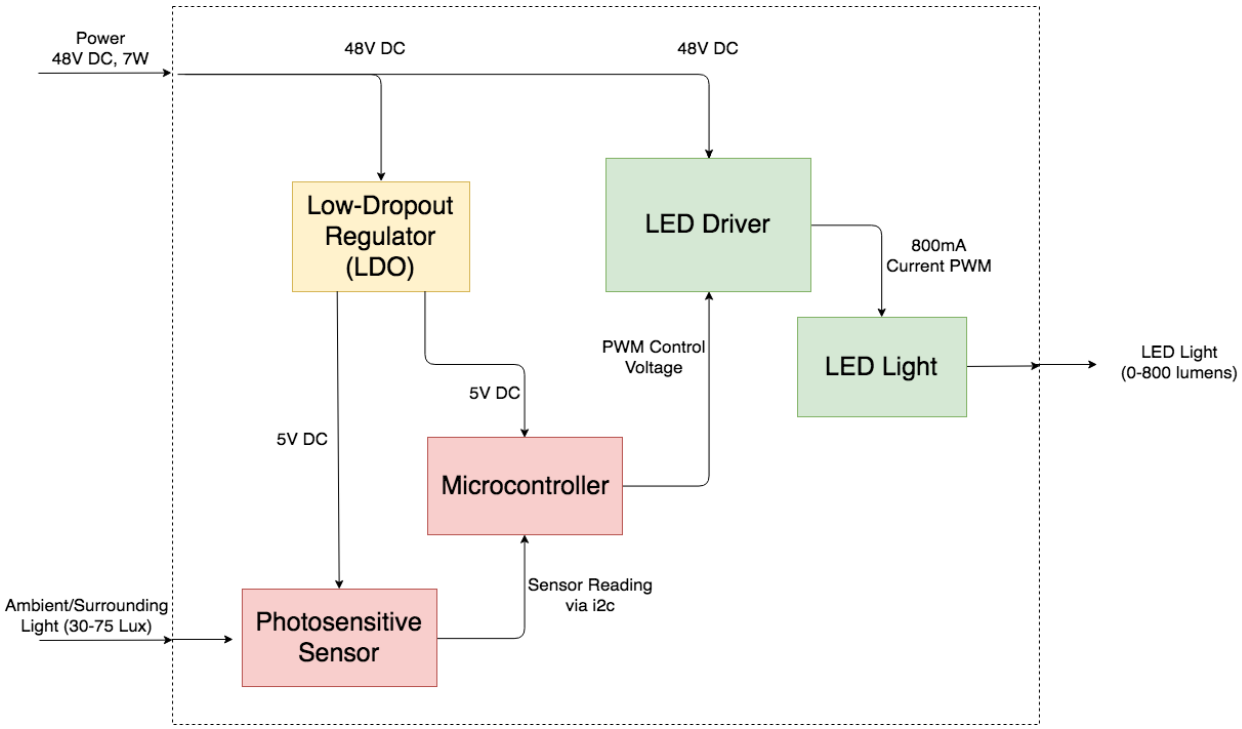


Figure 3-2. Level 1 Block Diagram

Figure 3-2 depicts the level 1 block diagram, which breaks the level 0 diagram into smaller more specialized blocks each with their individual inputs and outputs (I/O). It specifies the types and connections of these I/Os and how it achieves the desired output to the level 0 block diagram. The system first takes the 48V DC and sends it through a Low-Dropout Regulator (LDO) to lower the voltage to 5V so that it may be used to power the microcontroller and photosensitive sensor. The photosensitive sensor communicates via i2c to the microcontroller to give a light intensity reading. The microcontroller then maps this reading to a corresponding pulse width modulated (PWM) control voltage that it outputs to the LED driver. Based on the PWM voltage it received from the microcontroller, the LED driver outputs appropriate PWM current using the 48V DC from the bus. The LED is driven by the current output and produces the correct amount of light out.

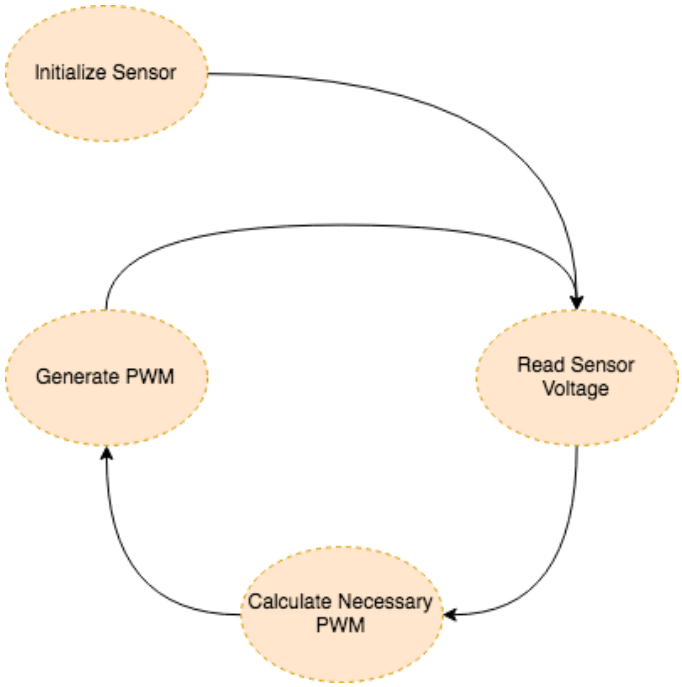


Figure 3-3. Software Flow Diagram

The software flow diagram as illustrated in Figure 3-3 describes the process performed by the microcontroller in software. The microcontroller initializes the photosensitive sensor and then goes into an endless loop consisting of three stages. First it requests a reading from the sensor. Once it has received and stored this reading, it maps the reading to an appropriate PWM output value through a lookup table. The PWM is then written out to the appropriate pin.

Table 3-1. Specifications of the Adaptive Light Dimmer

Maximum Output light brightness	800 lumens
Efficiency	70%
Output Power	5 W
Input Power	7.14 W
Input Voltage	48 V
Response Time	2 seconds
Sensor Sensitivity	35 to 150 lux
Cost	<$100
Dimensions	5’ tall x 1’ diameter
Weight	5 lbs

Chapter 4. Design and Simulation Results

The main purpose of the Adaptive Light Dimmer is to both minimize costs by using cheaper materials while maximizing efficiency and to operate from a DC voltage source, allowing renewables to be a direct source of power. Simplifying the design can also allow the device to operate without worry of installation. The Adaptive Light Dimmer has three main parts to design. First was the LED driver, which has to drive a 5V LED at a maximum current of 800mA. The second was the LDO linear regulator, which had to step down the 48V input to 5V in order to power the microcontroller and the light sensor. The last portion of the design was the combination of the microcontroller and the light sensor, both of which had to operate at 5V.

The LED driver used was Linear Technology's LT3795. This device was chosen because it can operate with an input voltage of 48V. In addition, because it is an LT device, it would be much easier to simulate and to test the design using LTSpice. To meet our specifications, the circuit we designed was a modification of an example circuit taken from the LT3795 datasheet, shown in Figure 4-1. The circuit is a buck mode LED driver meant to operate from 24V to 80V and output a controlled current to the LED. This current can then be controlled directly by changing the control voltage or by sending a PWM signal into the PWM pin of the device.

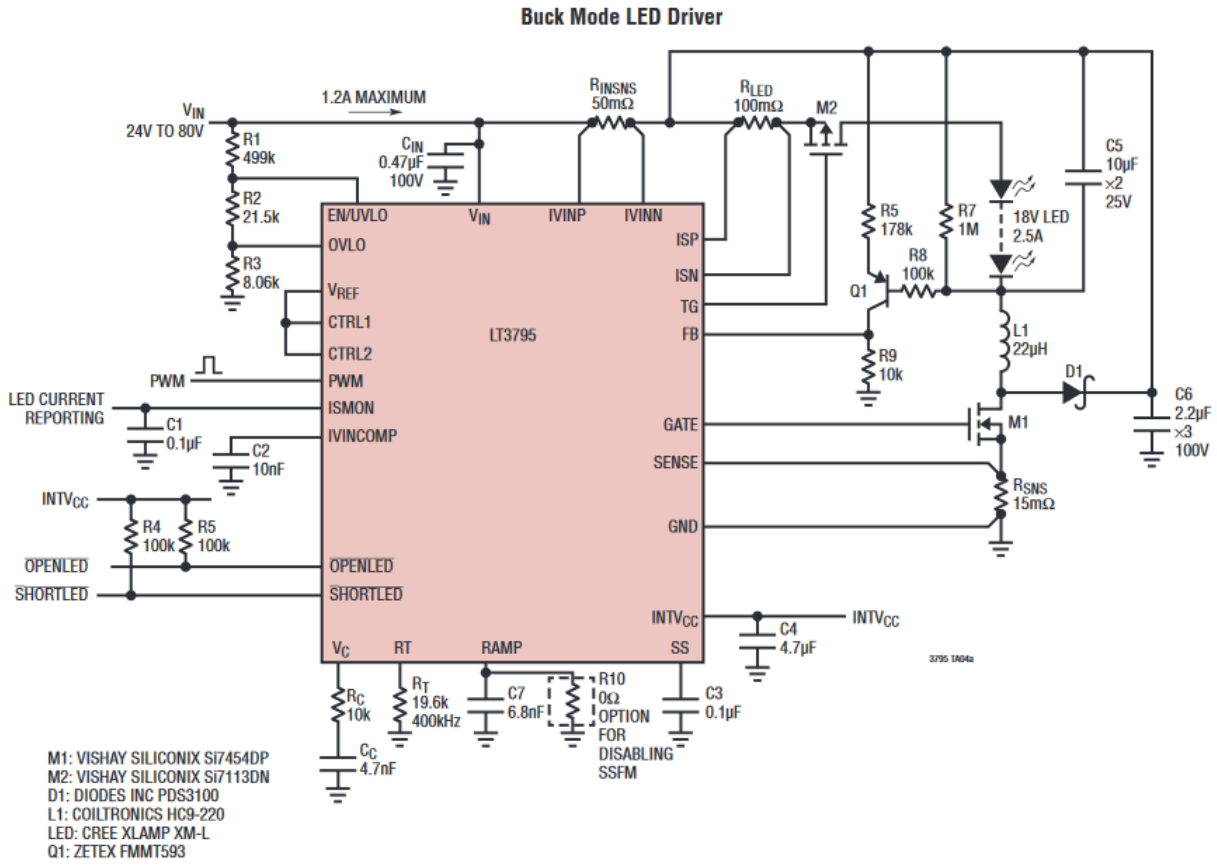


Figure 4-1. The Buck-Mode Example Circuit from the LT3795 Datasheet

The components that had to be changed were the input capacitor C_{IN}, the inductor L, the input current sense resistor $R_{INSENSE}$, the LED current resistor R_{LED}, the PMOS, the NMOS, and the PNP BJT. These components directly affect the operation of the circuit, so they were changed in order to produce the correct maximum output current and to keep the circuit from damaging any components in the design.

Because the input voltage was 48V, during transient operation, the voltage could reach upwards of 63V. So, voltage rating was an aspect that had to be taken into consideration when designing the circuit. The equation used to find the appropriate value of C_{IN} in microfarads was:

$C_{IN} = I_{LED} * V_{LED} * (V_{IN} - V_{LED})/V_{IN}^2 * T_{SW} * 10\mu F/(A * \mu s)$

$C_{IN} = 0.8\ A*5V * (48V - 5V)/(48V)^2 * 10^6/400000Hz * 10\mu F/(A*\mu s)$

$C_{IN} = 1.87\mu F$

The equation used to size the inductor was:

$L = (T_{SW} * R_{SENSE} * V_{LED} * (V_{IN} - V_{LED}))/(V_{IN} * 0.02V)$

$L = (1/400000Hz * 0.015\Omega * 5V * (48V - 5V))/(48V * 0.02V)$

$L = 8.4\mu H$

The input current sense resistor was determined more loosely. Because the configuration is a buck LED driver, the input current should not be very high, so R_{INSNS} was chosen as 100 mΩ. By the equation below, the maximum input current was determined.

$I_{IN(MAX)} = 0.06V/R_{INSNS}$

$I_{IN(MAX)} = 0.06V/0.1\Omega$

$I_{IN(MAX)} = 600mA$

The LED current resistor was chosen such that the maximum LED current was 800mA. Because the control pins were tied high, the equation used to determine the LED current resistor was simplified to:

$R_{LED} = .25V/I_{LED(MAX)}$

$R_{LED} = .25V/0.8A$

$R_{LED} = 312.5m\Omega$

The PMOS and PNP BJT transistors were chosen differently because the ones from the datasheet were not readily available for both simulation and in the coming hardware tests. The same was true for the NMOS transistor, but in addition, the NMOS transistor was chosen to

minimize the gate charge. Higher gate charge values increases the current in the INTVCC pin and could disrupt the operation of the circuit by dropping the INTVCC pin below the threshold.

$Q_{G(MAX)} = I_{INTVCC}/f_{OSC}$

$Q_{G(MAX)} = 0.02A/400000Hz$

$Q_{G(MAX)} = 50nC$

The resistor R_T, which determines the switching frequency of the LED driver, was determined to be adequate, as 400kHz is not too low, which would increase component sizing, or too high, which would increase losses. The feedback resistors, which set the maximum output voltage, were adequate because we are only using a 5V LED. The sense resistor R_{SNS} is used to measure the actual current through the LEDs. Its value must be less than $0.07V/I_{LED}$, which for 800mA was 87.5mΩ. All other components were not changed because they help the LT3795 function properly. Their change would not affect our output, and any changes could adversely affect the normal operation of the device.

The final design of the LED driver is shown in Figure 4-2.

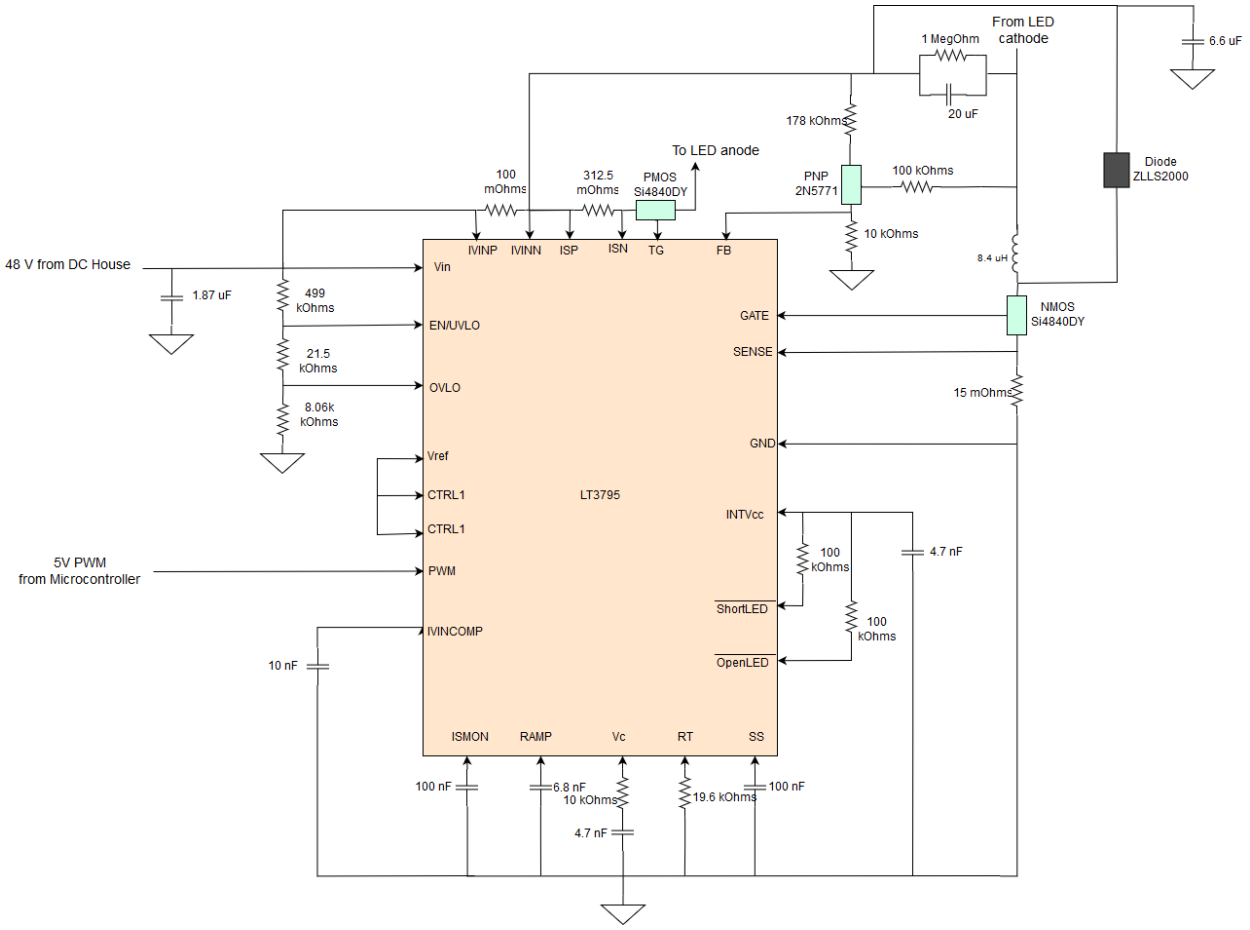


Figure 4-2. Final Design of the LED Driver

Simulations of the design were done using LTSpice since the LED driver is a Linear Technology device. The simulation circuit is shown in Figure 4-3. The simulation results for the output voltage is shown in Figure 4-4 and Figure 4-5 for 100% and 50% duty cycle, respectively. The output current at 100% duty cycle is shown in Figure 4-6 and the current at 50% duty cycle is shown in Figure 4-7. The input current at 100% duty cycle is shown in Figure 4-8 and at 50% duty cycle in Figure 4-9. The input power at 100% and 50% duty cycle is shown in Figure 4-10 and Figure 4-11, respectively.

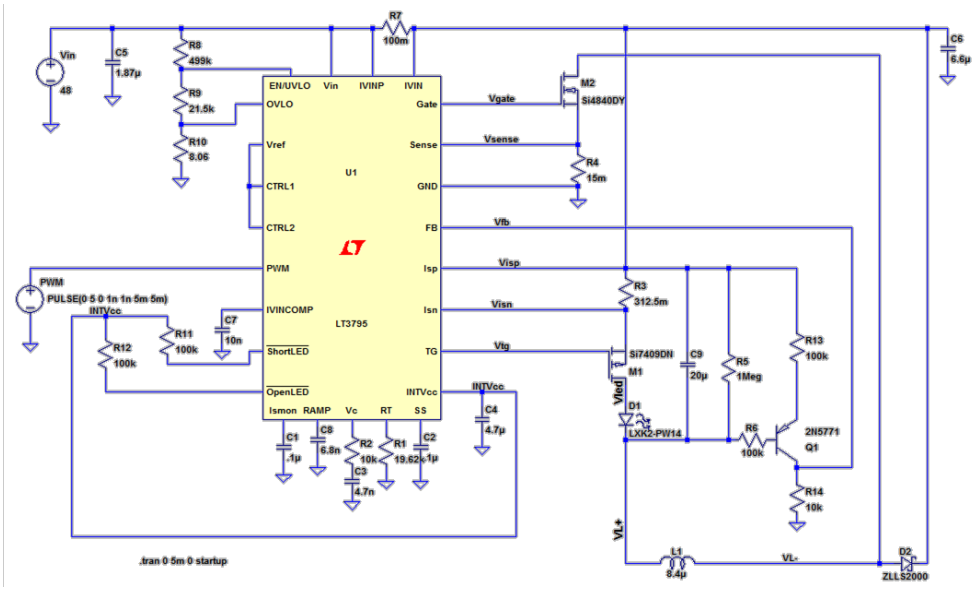


Figure 4-3. LED Driver Circuit used for LTSpice Simulation

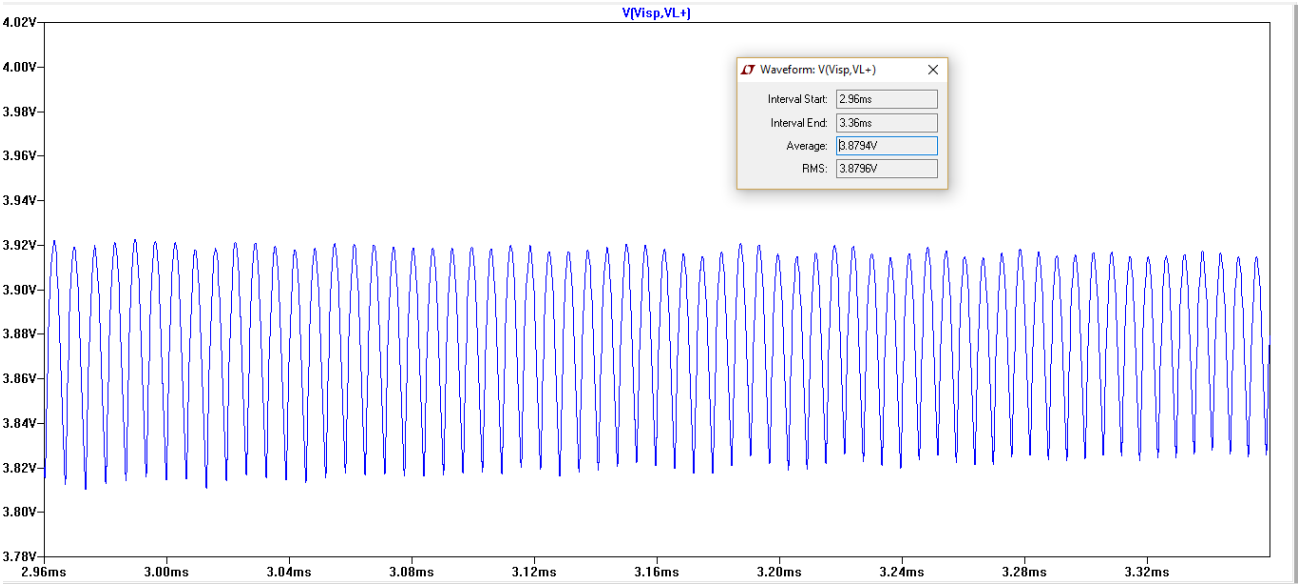


Figure 4-4. Simulation Results for Output Voltage at 100% Duty Cycle

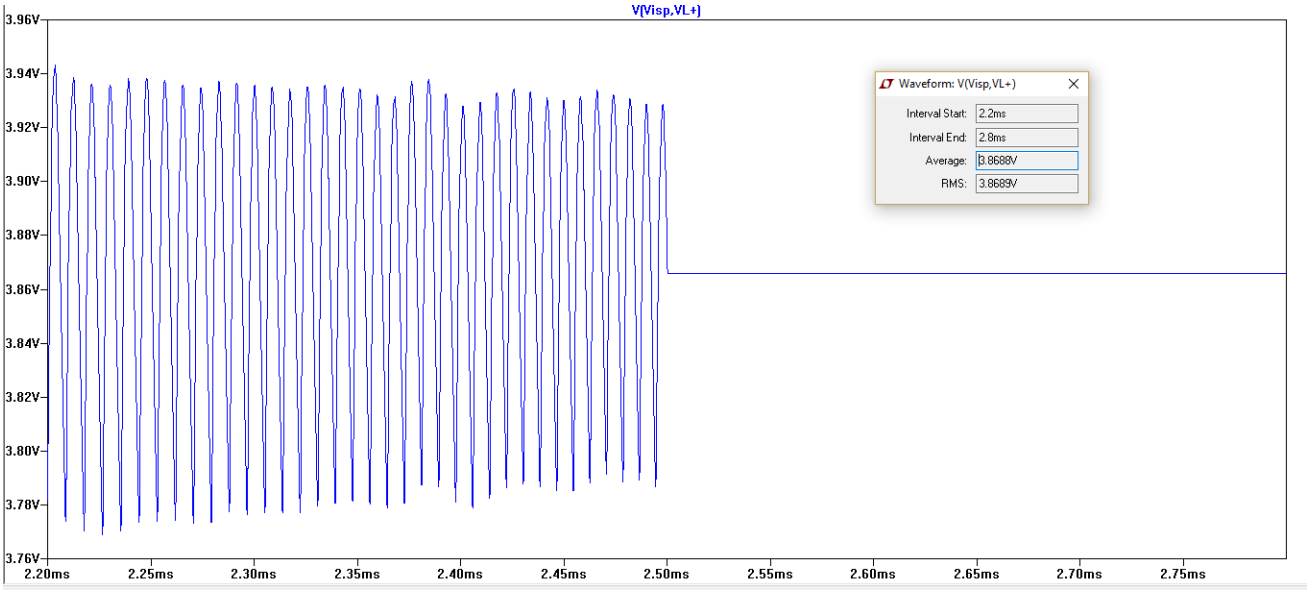


Figure 4-5. Simulation Results for Output Voltage at 50% Duty Cycle

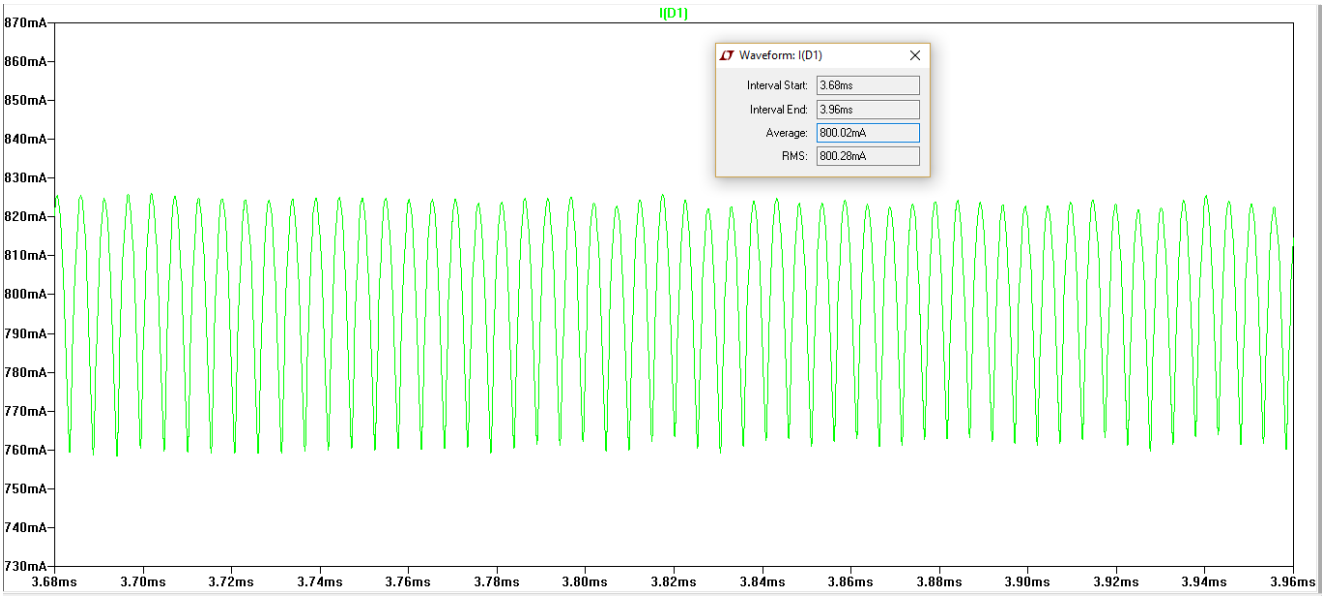


Figure 4-6. Simulation Results for Output Current at 100% Duty Cycle

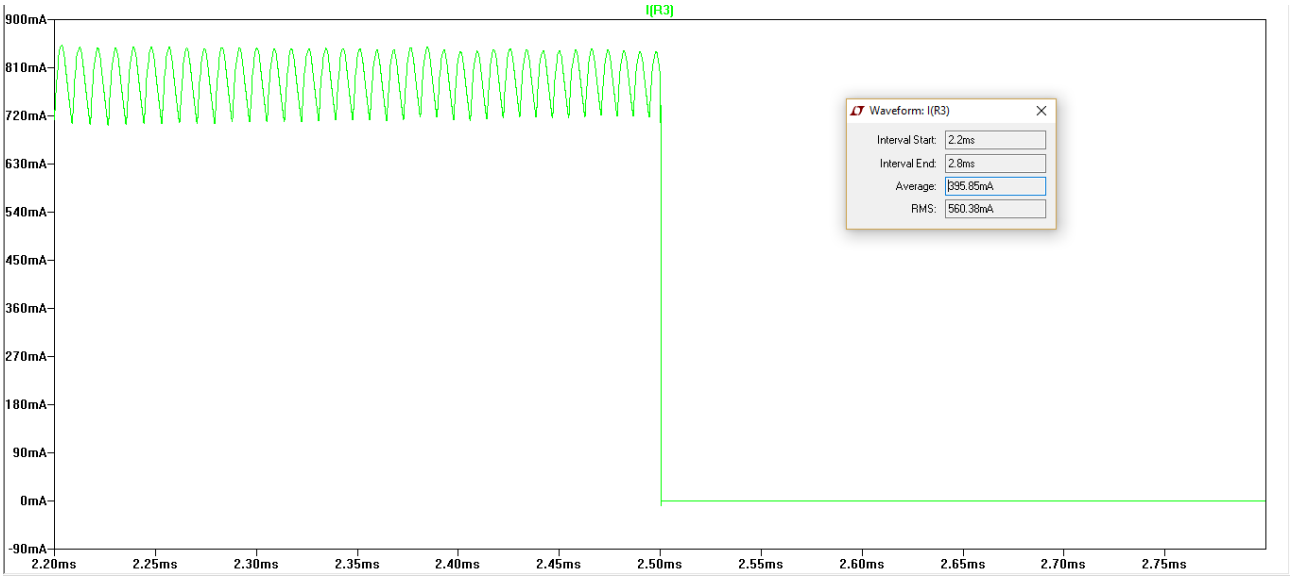


Figure 4-7. Simulation Results for Output Current at 50% Duty Cycle

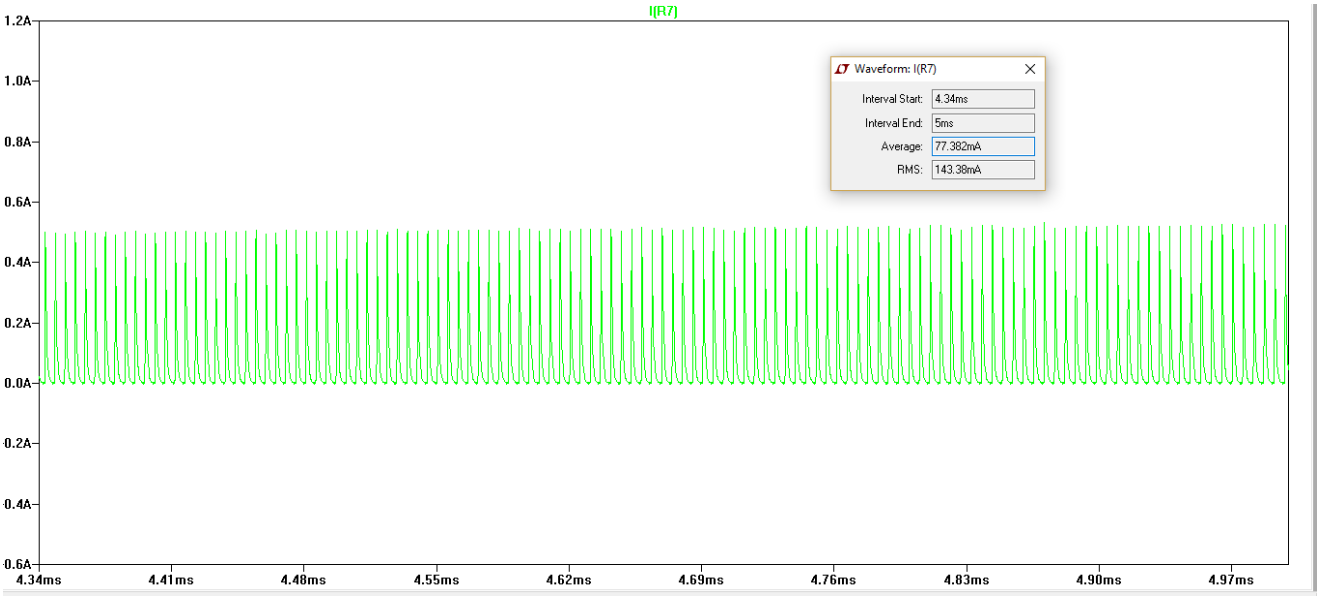


Figure 4-8. Simulation Results for Input Current at 100% Duty Cycle

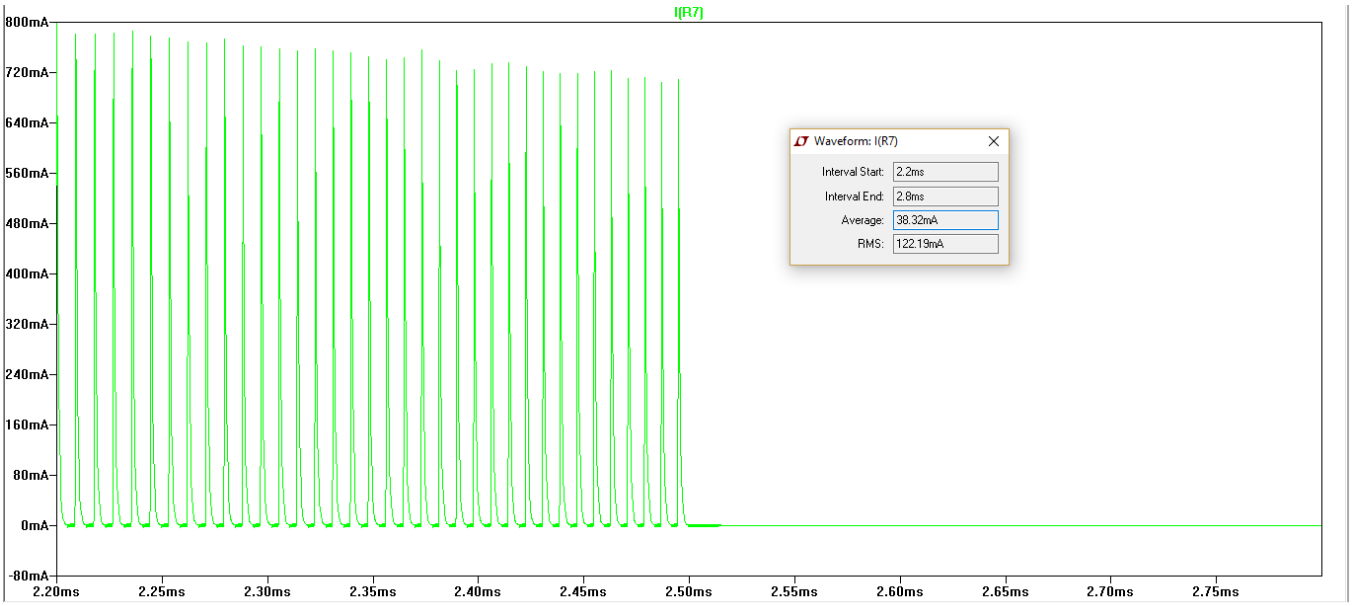


Figure 4-9. Simulation Results for Input Current at 50% Duty Cycle

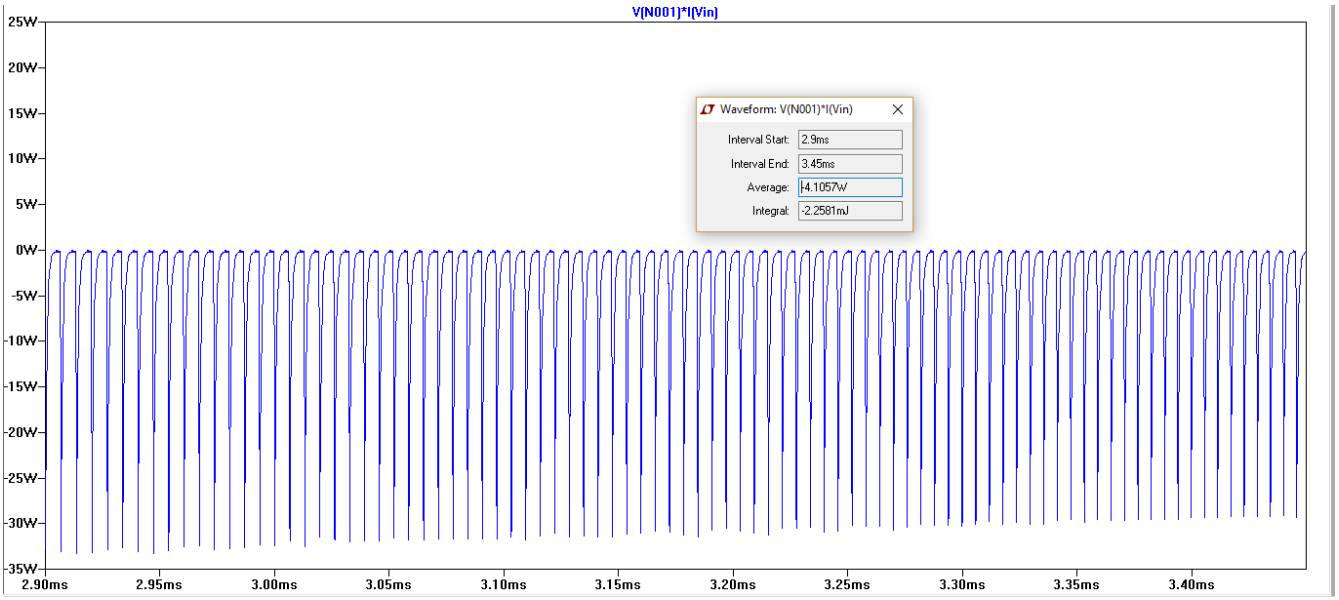


Figure 4-10. Simulation Results for Input Power at 100% Duty Cycle

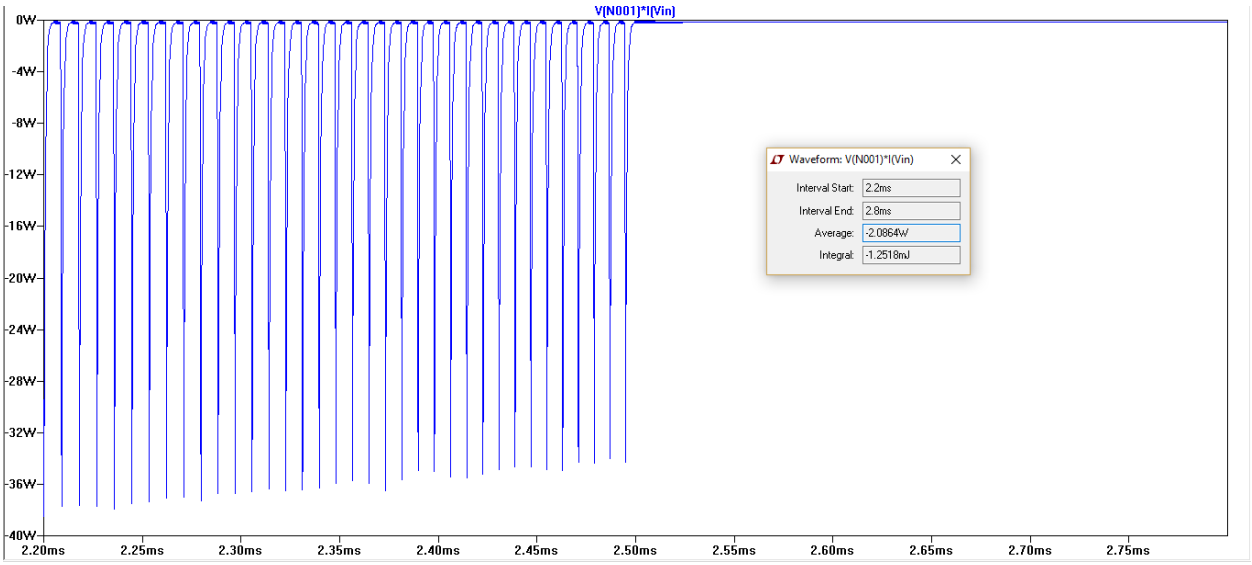


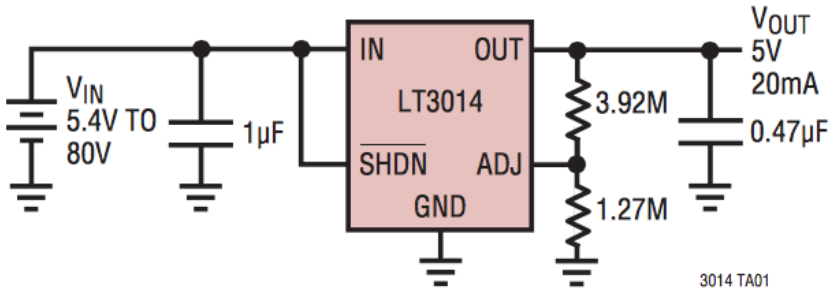
Figure 4-11. Simulation Results for Input Power at 50% Duty Cycle

Because the LED used in simulation is a 4V LED, we see that the output voltage is 4V at 100% duty cycle. At 50% duty cycle, the output voltage stayed relatively the same. The output current, however, was cut in half from 800mA to 400mA at 50% duty cycle. This is what we expected, as halving the duty cycle should halve the power drawn by the LED. Rather than a change in the output voltage, the output current is halved for this to occur. The input current at 100% duty cycle was about 77mA, which is far below the limit that was set in the design. At 50% duty cycle, the input current was 38mA. Decreasing the duty cycle caused a proportional decrease in the input current, which fits expectations. The input power dropped from 4.1W to 2.1W at 50% duty cycle. Since input power is proportional to input current, this was expected.

The efficiency of the system between different duty cycles was essentially unchanged, around 76%, since the change in input current was proportional to the change in output current while the input of output voltages stayed the same. When built, we expect the lower duty cycle operation to be less efficient due to additional switching from the PWM signal.

The second portion of the dimmer is the low drop-out regulator (LDO) circuit which lowers the 48V input to a usable 5V for the microcontroller and sensor. The LT3014 was chosen because it fits well into the project; has the capability of dropping 48V to 5V, and would incur little power loss since the microcontroller/sensor circuit would not pull a significant amount of current.

Figure 4-12 shows the configuration of the LDO to be implemented which is taken from the LT3014 datasheet. This configuration is setup for "5V Supply with Shutdown" according to the datasheet and fits the needs of the light dimmer.



$V_{\overline{SHDN}}$	OUTPUT
<0.3V	OFF
>2.0V	ON

Figure 4-12. Linear Technology's "5V Supply with Shutdown" Circuit

The calculation of the resistors needed to obtain the desired V_{out} and the equation used was found in the LT3014 datasheet and is as follows:

$$V_{out} = 1.22V \times (1 + \frac{R2}{R1}) + (I_{ADJ})(R2)$$

$$V_{out} = 1.22V \times (1 + \frac{3.92M\Omega}{1.27M\Omega}) + (4nA)(3.92M\Omega)$$

$$V_{out} = 5.0055V$$

The value R2 corresponds to the resistor in between the OUT and ADJ pins and the value R1 corresponds to the resistor connected to ADJ and ground.

The value of the capacitor on the output was specified to be at least 0.47μF to prevent oscillations; hence, it is sufficient for the device.

The shutdown pin was set to high since the pin is active low. By doing this, the LDO is set to be on whenever there is power supplied to it.

With the above considerations in mind, it was determined that the "5V Supply with Shutdown" circuit would fulfill the needs of the dimmer. The circuit, including the 48V supply and output to the microcontroller/sensor circuit, is shown in Figure 4-13.

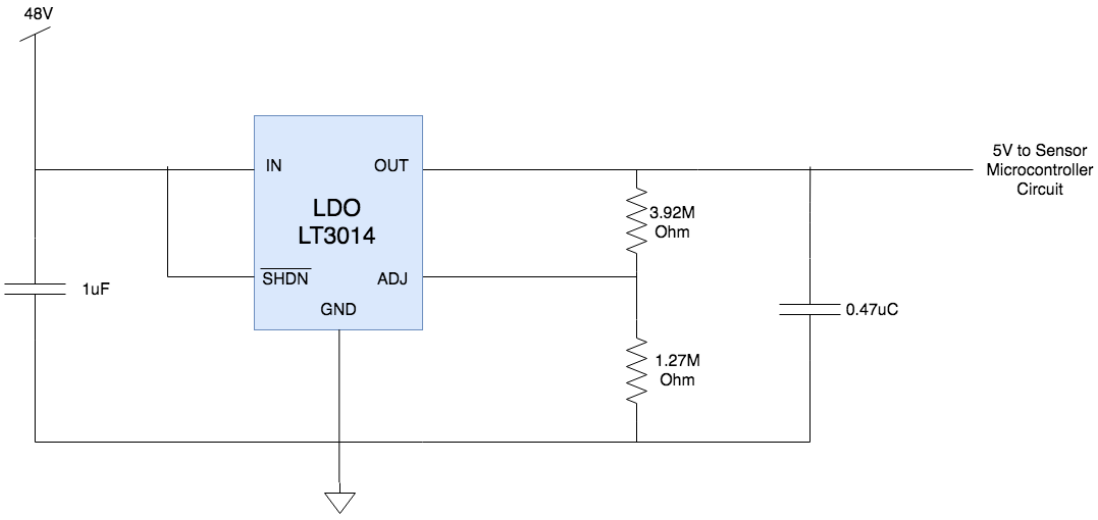


Figure 4-13. LDO Circuit Design Diagram

The circuit was simulated using LTSpice and their model of the LT3014, shown in Figure 4-14. The input voltage was set to 48V and the R3 resistor was setup to act as the load of the microcontroller/sensor circuit. According to the ATTiny85 datasheet, at 1MHz the microcontroller draws 1mA so R3 has a value of 5V/0.001A = 5kΩ.

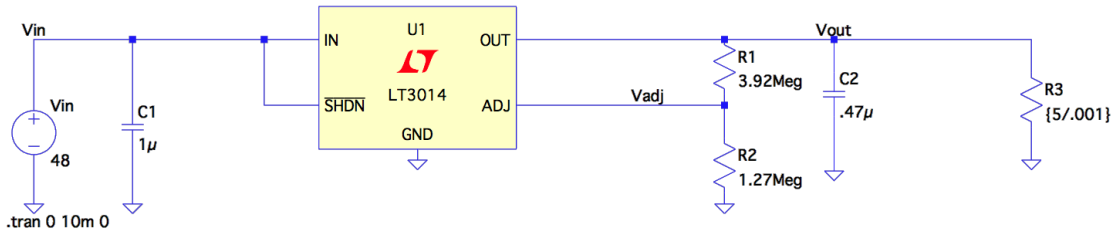


Figure 4-14. LDO Circuit Used for LTSpice Simulation

The simulation results of the output voltage and current are shown in Figure 4-15. The results of the input voltage and current are shown in Figure 4-16. The output voltage stays at 5V as expected and the current drawn is 1mA. This results in 5mW of power drawn by the microcontroller and sensor. The input voltage is 48V and the input current is 1.05mA. The input power is 50.4mW. This results in an efficiency of 9.9%, but this does not pose a problem because the power usage is much smaller than the power consumed by the LED driver. It does not significantly impact the system's efficiency as a whole.

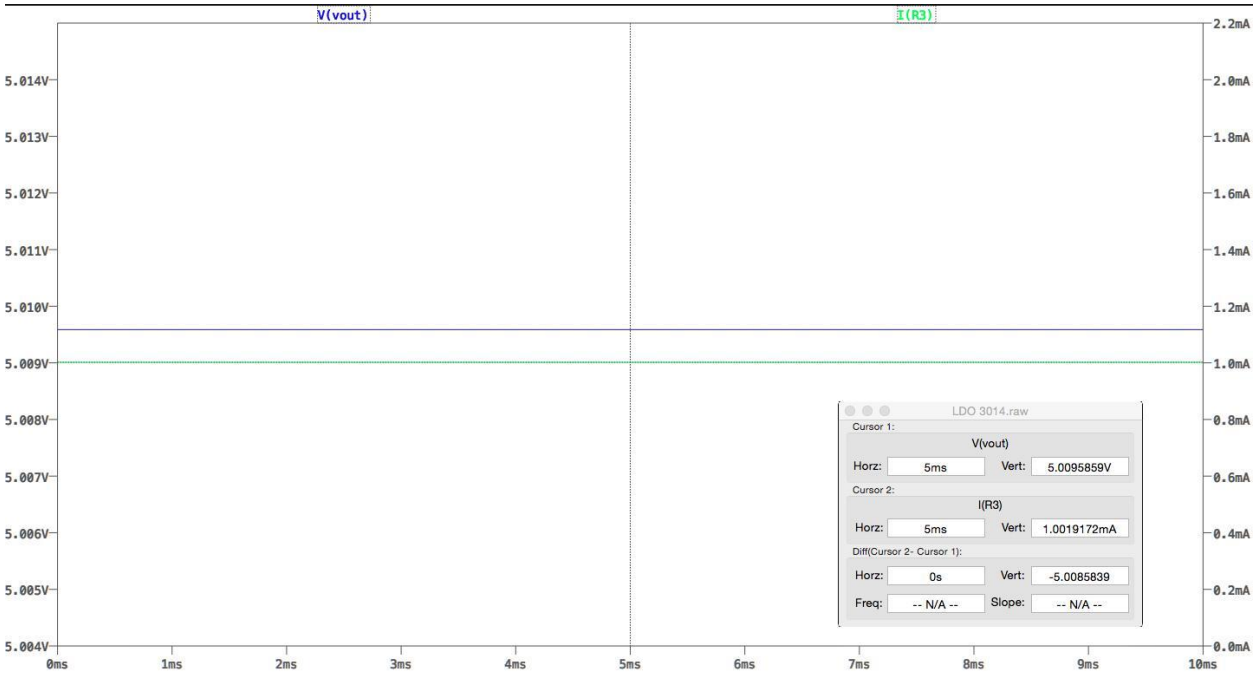


Figure 4-15. Simulation Results of the Output Voltage and Output Current

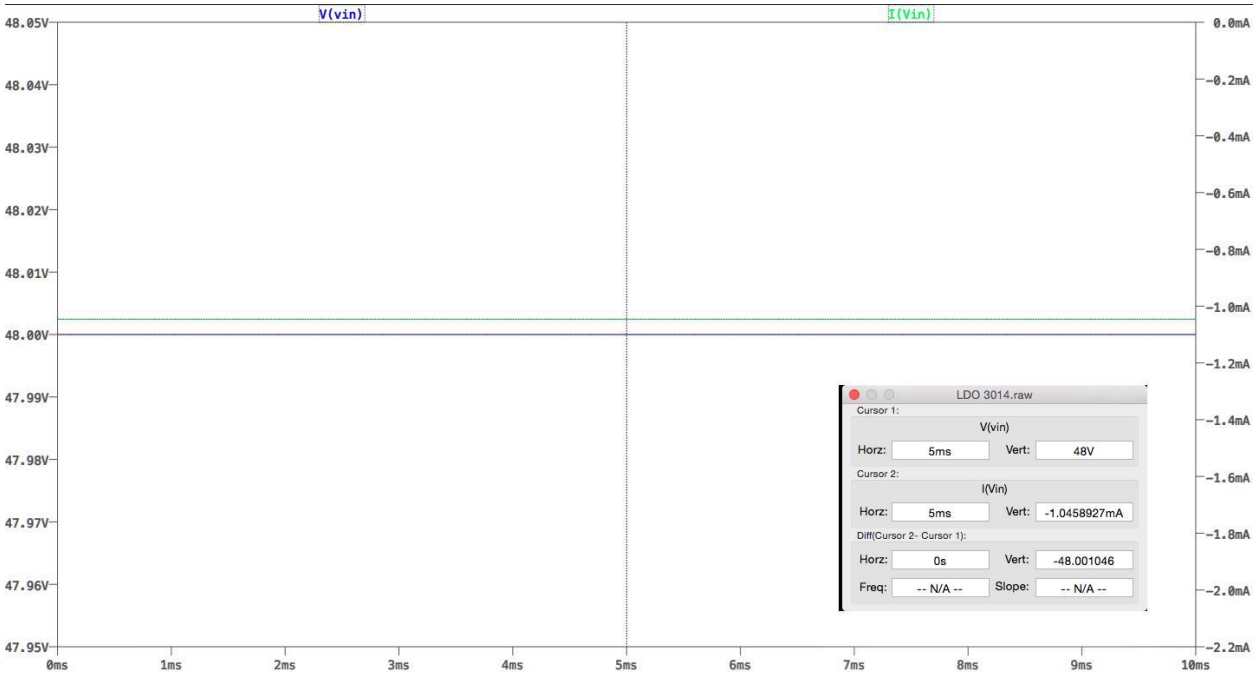


Figure 4-16. Simulation Results of the Input Voltage and Input Current

The final portion of the dimmer consists of the sensor and microcontroller circuit. The VEML6070 was chosen because of its ability to communicate via I^2C, to detect UV light, and because its operating voltage range allows it to use the same voltage source as the ATTiny85. The ATTiny85 was chosen for its low cost, I^2C capability and ability to produce a PWM output. In addition, it also can be programmed with the Arduino UNO using Sketch's built in "ArduinoISP" library.

Since the VEML6070 communicates via I^2C, the SDA (data line) and SCL (clock line) have 4.7kΩ pull up resistors to ensure proper logic levels. These are connected to the five volts from the LDO, which supplies voltage to the microcontroller and UV light sensor. The output PWM on pin six on the microcontroller connects to the PWM input of the LED Driver. The ACK pin of the VEML6070 sends a signal if the UV light reading drops below a built in threshold and was not connected since the feature is not necessary for the function of the device. The ground is common to the LED driver and LDO circuits as well.

The design of the circuit is shown in Figure 4-17.

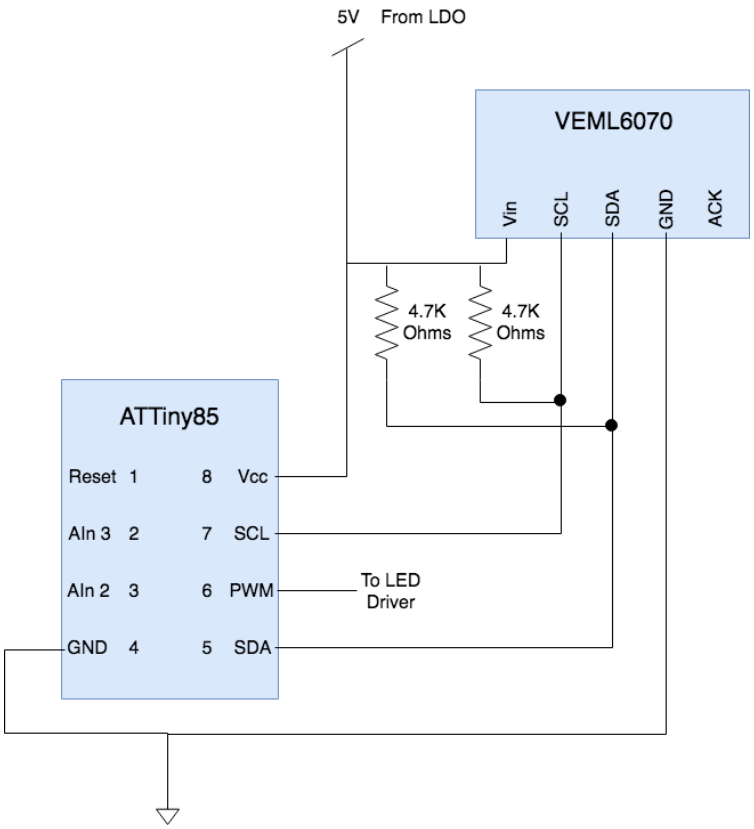


Figure 4-17. Sensor and Microcontroller Circuit Diagram

Chapter 5. Hardware Test and Results

The hardware of the light dimmer was built and tested in separate stages. The LED Driver, LDO, Microcontroller, and Sensor were tested individually and then combined to construct the entire system. First the LED Driver was assembled on a protoboard with the appropriate components. The LDO was initially built on a breadboard with its accompanying components and then moved to the protoboard. The microcontroller and ambient light sensor were first tested on a breadboard using jumper wires and then moved to the protoboard after confirming that they both communicated with each other and produced the correct outputs.

Linear Technology's LT3795 LED driver comes in a 28 pin TSSOP package. In order to test it on the protoboard, the LT3795 was soldered onto a TSSOP to DIP adapter board and headers were soldered to the board. The chip was then soldered to the protoboard along with the appropriate components for the Buck Mode configuration. The schematic for the LED driver is shown in Figure 5-1. A protoboard was used for testing rather than a breadboard because breadboards are generally too noisy for a switching device like the LT3795 to work properly. Many of the connections were under the protoboard to help with noise and to minimize confusion on the top of the board. Functionality of the circuit was tested using a power supply for the 48V DC input voltage and measuring the output current with a digital multimeter while driving the PWM pin with PWM signals of varying duty cycles from a function generator. Testing involved connecting the LED to the output of the driver and varying the duty cycle of the function generator to produce different levels of light. The output currents of the LED driver were measured for 0% to 100% duty cycle, with 5% steps. At the same duty cycles, a luxmeter, placed about 4 feet from the bulb, was used to measure the amount of light produced by the

LED. When testing, the LED and LED driver were connected through an IKEA lamp bought from Amazon. By using the lamp, we could use the lamp's sconce to diffuse the light and have better lux measurements. Due to diffusion, only the distance from the LED is important, so we could place the lux meter below the lamp. It was determined that it would be simpler to use a 12V LED with a socket adapter to connect to the lamp. The lux measurements were made 1 inch from the bulb as well to measure the light output more closely. The block diagram of the test setup is shown in Figure 5-2. A picture of the protoboard is shown in Figure 5-3 and a picture of the setup is shown in Figure 5-4. The currents and lux measured are shown in Table 5-1. Notably, the protoboard did not operate properly, as the maximum output current at 100% duty cycle was only 101.8 mA compared to the LED's maximum output of 321 mA.

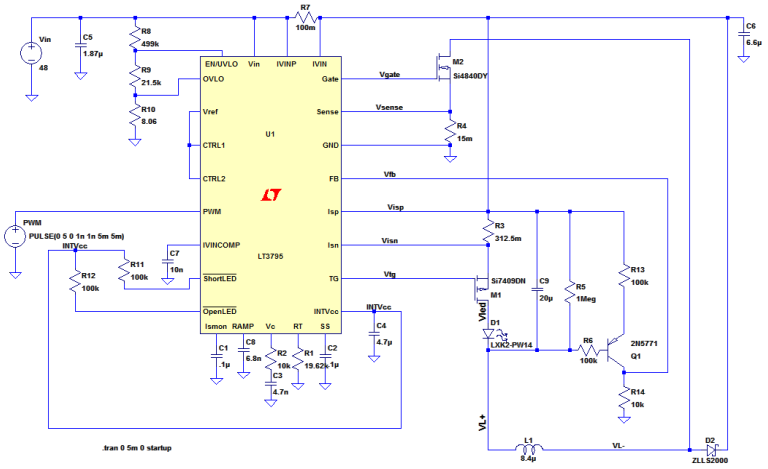


Figure 5-1. Schematic used to construct the Protoboard Circuit

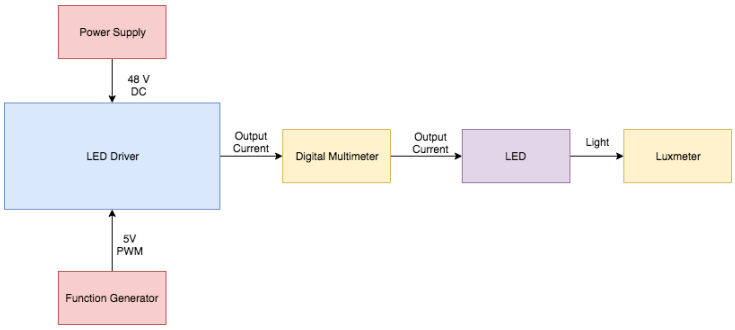


Figure 5-2. Block Diagram of the LED Driver Test Setup

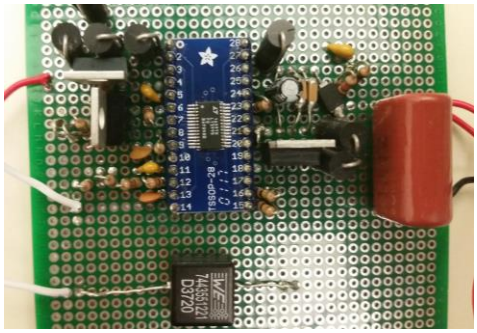


Figure 5-3. Picture of the LED Driver on the Protoboard

Figure 5-4. Picture of the LED Driver Test Setup

Table 5-1. Protoboard LED Lux and Current Measurements

Duty Cycle (%)	Output Current (mA)	Lux (4 Foot Radius)	Lux (1 Inch From Bulb)
0	10.63	0	0
5	896	0	312
10	1.56	0	629
15	2.315	0	991
20	3.05	0	1050
25	3.84	0	1340
30	4.06	3.3	1750
35	33.2	3.8	2060
40	37.3	4.2	2360
45	43.5	4.5	2680
50	50.1	4.8	3140
55	56.6	5.1	3480
60	61.5	5.4	3780
65	65.9	5.5	4030
70	70.86	5.8	4320
75	76.57	6.1	4650
80	82.3	6.4	5170
85	87.1	6.6	5460
90	91.55	6.8	5730
95	96.7	7.1	6010
100	101.8	7.3	6310
max voltage 12V	321	16	17880

Similarly, Linear Technology's LT3014 Linear Dropout Regulator (LDO) has a very small package and some modifications had to be made in order to test it on the protoboard. The five pin LDO was connected to a 28 pin SOIC to DIP adapter, using only pins 1, 2, 3, 26, and 28. While the adapter board is not made for the LDO's package, it was good enough for connecting and testing of the LDO. The LDO was connected to a breadboard and the necessary resistors and capacitors were connected to complete the circuit. The schematic for the LDO is shown in Figure 5-5. A power supply was used for the 48V input voltage for the LDO and the output voltage was measured with a digital multimeter to ensure that it was at 5V. The block diagram of the test

setup is shown in Figure 5-6. A picture of the LDO is shown in Figure 5-7. A picture of the test setup is shown in Figure 5-8.

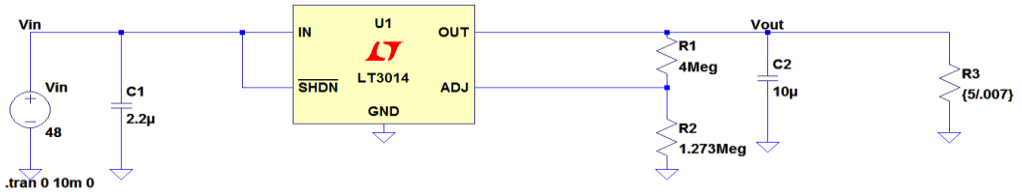


Figure 5-5. Schematic Used for the LDO

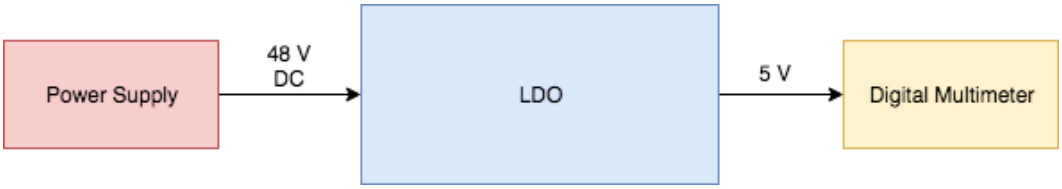


Figure 5-6. Block Diagram of the LDO Test Setup

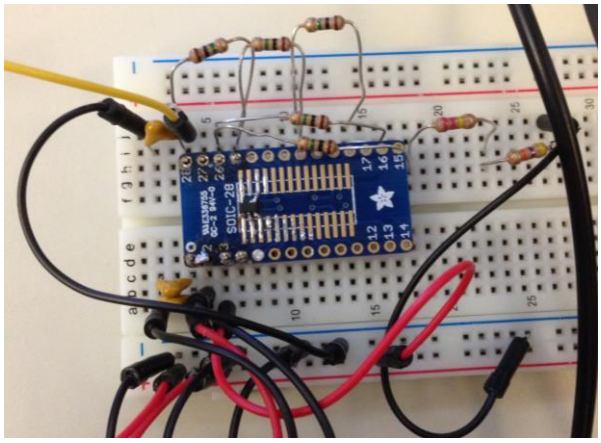


Figure 5-7. Picture of the LDO on Breadboard

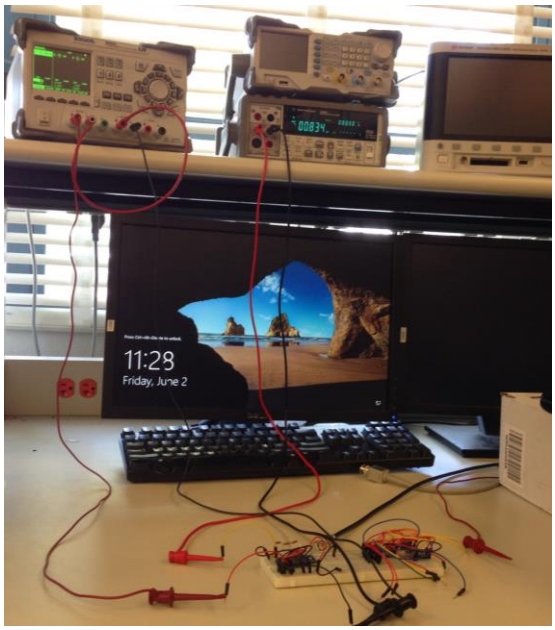


Figure 5-8. Picture of the LDO Test Setup

The ATTiny85 microcontroller comes in a DIP package, making prototyping and testing fairly easy using a breadboard. Arduino's built in "ArduinoISP" in system programmer program combined with the Tiny AVR Programmer from High-Low Tech was used to program the ATTiny85 through Arduino's Sketch environment. To program the ATTiny85, the Arduino Uno's Dig 10 through 13 pins were connected to the ATTiny85's 5 through 8 pins, as seen in Figure 5-9, using Serial Peripheral Interface (SPI) to upload the program. Additionally, a 10µF electrolytic capacitor was connected between Reset and Ground pins to prevent accidental reset of the Arduino. To first test that uploading was successful, the blink example was used and an LED was connected between pins 5 and ground.

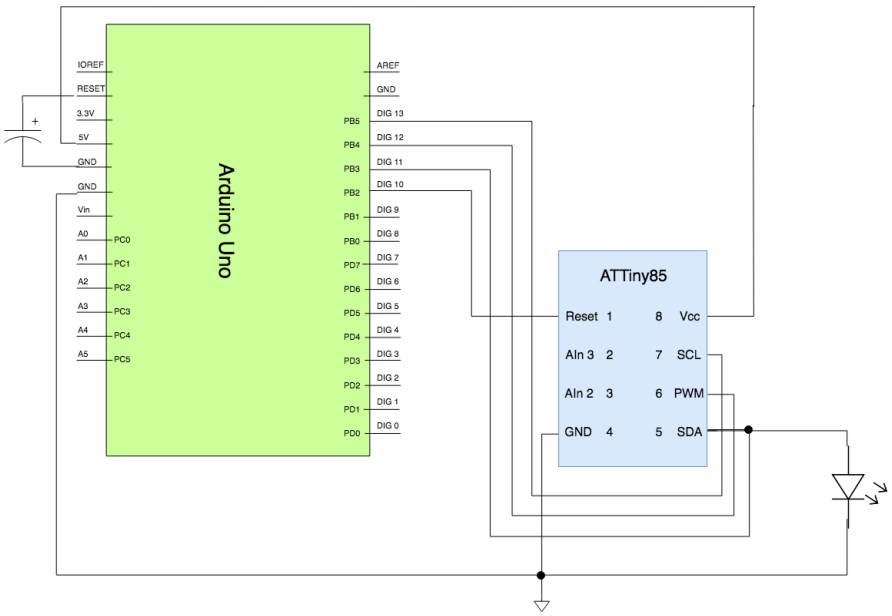


Figure 5-9. ATTiny85 Blink Test Circuit Setup

Next the AnalogWrite() function of the ATTiny85 was tested to ensure that the microcontroller was outputting the correct PWM waveform. The circuit setup of this test is shown in Figure 5-10.

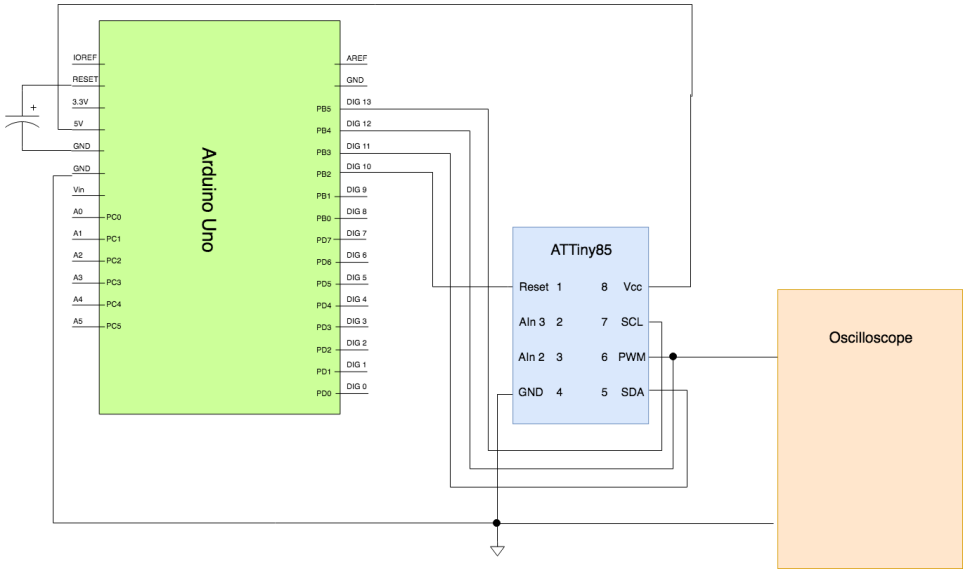


Figure 5-10. ATTiny85 PWM Test Circuit Setup

To get data from the sensor which communicates via I^2C, a library called TinyWire was used which implements I^2C on the ATTiny85. The code for this library can be found in Appendix A.

This code was tested in conjunction with the sensor to read its value and follows the form of the block diagram shown in Figure 5-11.

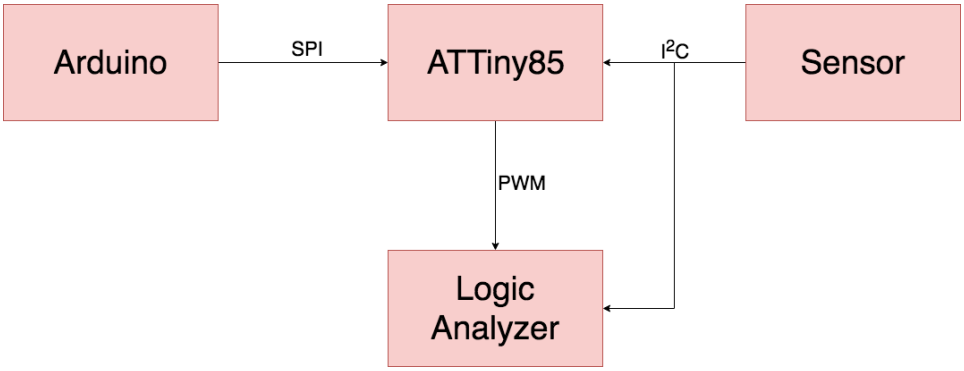


Figure 5-11. Microcontroller and Sensor Test Block Diagram

The VEML6070 breakout board has holes where headers can be attached making testing easy using a breadboard. To initialize the sensor, the datasheet was read to determine the address of the command register and what needed to be written to it. The Arduino was used to write to this register. To make sure the sensor was initialized a logic analyzer was used to check that the SCL pin of the sensor was providing an expected serial clock waveform. The Arduino was then used to poll the two data registers and the results were output to the serial monitor to see what values were being returned from the sensor.

After ensuring that the sensor communicated successfully with the Arduino, the ATTiny85 was connected and a similar set of commands were sent using using functions from the TinyWire library. To ensure proper logic levels, two 4.7 kΩ pull up resistors were added to the SDA and SCL lines. Code was then uploaded to the ATTiny85 to read from the sensor and output a PWM signal from the ATTiny85 so it could work independently from the Arduino. The code for this can be found in Appendix A. A logic analyzer was used to check the SDA, SCL, and output PWM pins to ensure the correct outputs. A circuit setup for this test can be seen in

Figure 5-12. A picture of the physical setup on the breadboard to test the sensor can be seen in Figure 5-13 and Figure 5-14.

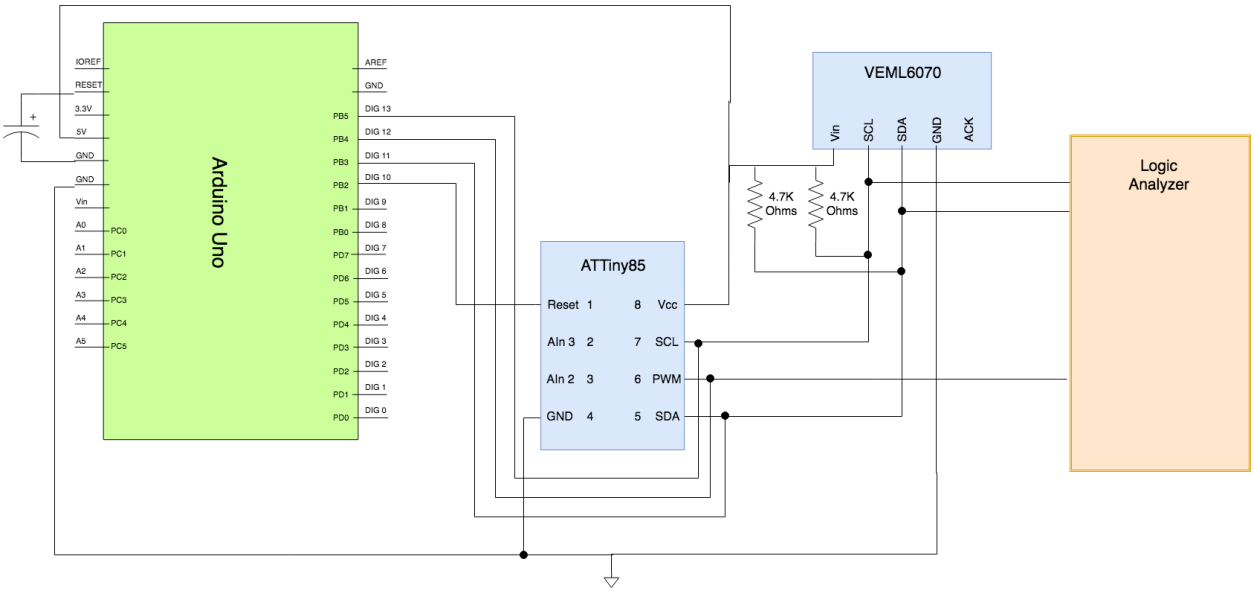


Figure 5-12. VEML6070 and ATTiny85 Test Circuit Setup with Logic Analyzer

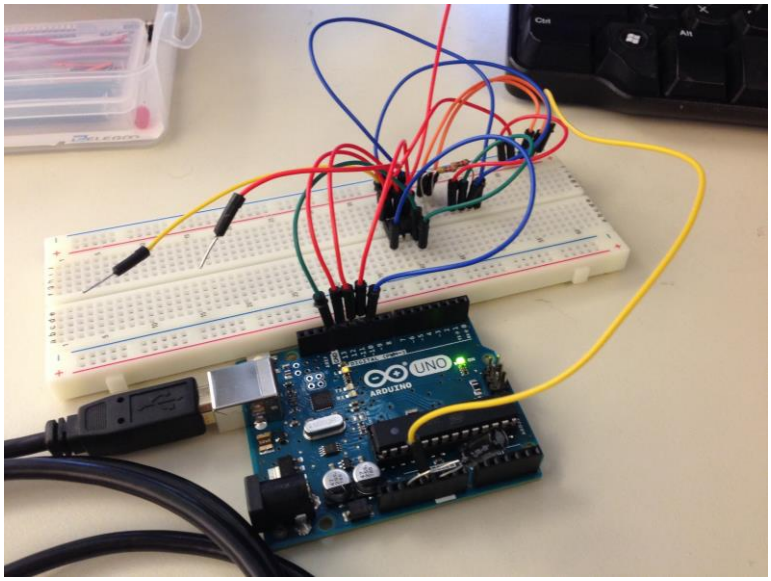


Figure 5-13. Physical Test Setup for Microcontroller and Sensor with Arduino

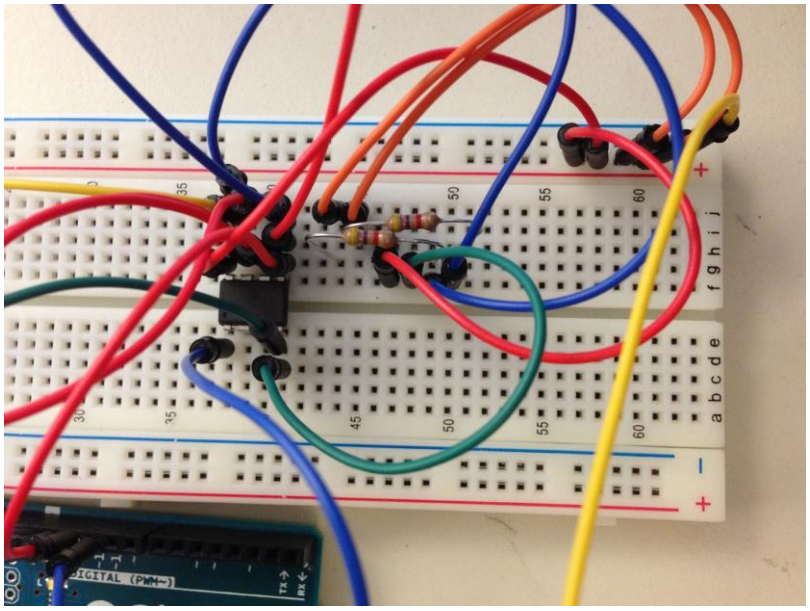


Figure 5-14. Breadboard Setup for Microcontroller and Sensor

After these initial tests, it was discovered that the lack of sensitivity to UV light and the sensor grouping different light levels led to a small variance of output values. This meant limited number of brightness levels for the LED. Because of this, a different sensor was chosen and tested. The TSL2561 takes measurements which can be converted into Lux which offer significantly more light levels. The connections are very similar to the VEML6070 as it also uses I^2C. The same approach to setup and testing used for the VEML6070 was used with the TSL2561. The circuit setup for final stages of testing this sensor can be seen in Figure 5-15. Additionally the code necessary to run the sensor and control brightness can be found in Appendix A.

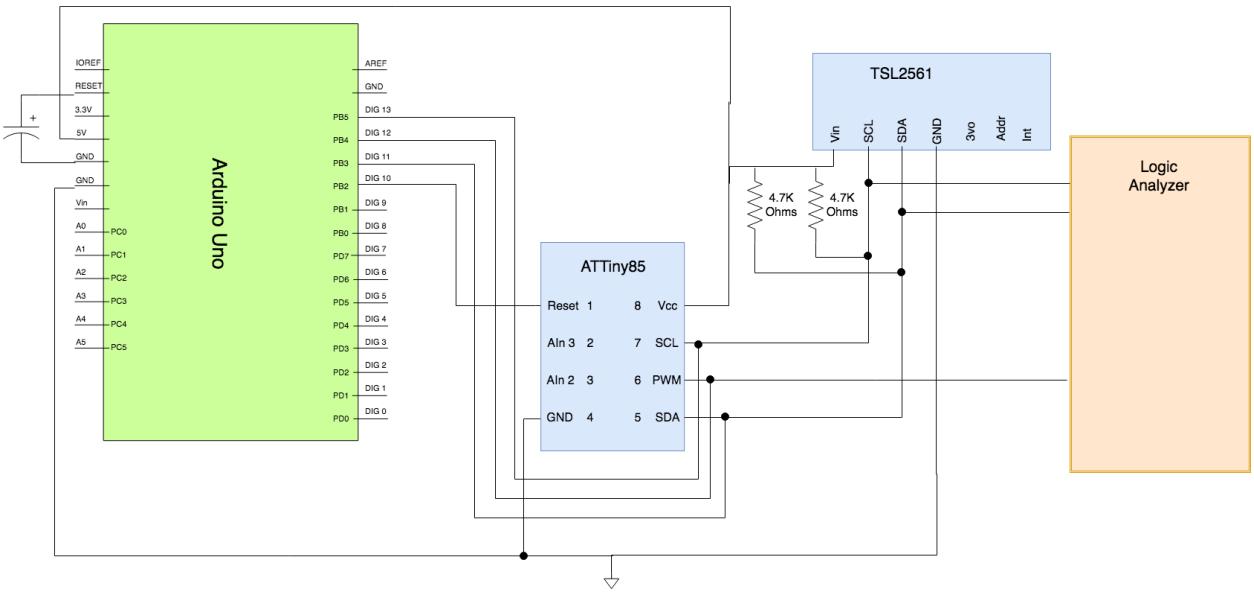


Figure 5-15. TSL2561 and ATTiny85 Test Circuit Setup with Logic Analyzer

The Eagle software was used to design a PCB using schematics similar to the ones used for the protoboard. However, some components and component values had to be changed slightly, as the appropriate surface mount components were not exactly the same as the through hole components used for the protoboard. Various resistor values had to be slightly changed. Among the most significant changes is the LED resistor, from 312.5 mΩ to 714 mΩ. The through hole NMOS, PMOS, and PNP BJT were changed to surface mount chips. The input capacitor was changed to three 100 V rating 2.2 µF ceramic capacitors in parallel. The ISP to GND capacitor was changed to four 100 V rating 2.2 µF ceramic capacitors in parallel. The inductor was changed from 8.4 µH to 22 µH to reduce current ripple. The new LED driver schematic is shown in Figure 5-16. The Eagle board layout is shown in Figure 5-17. The PCB is shown in Figure 5-18.

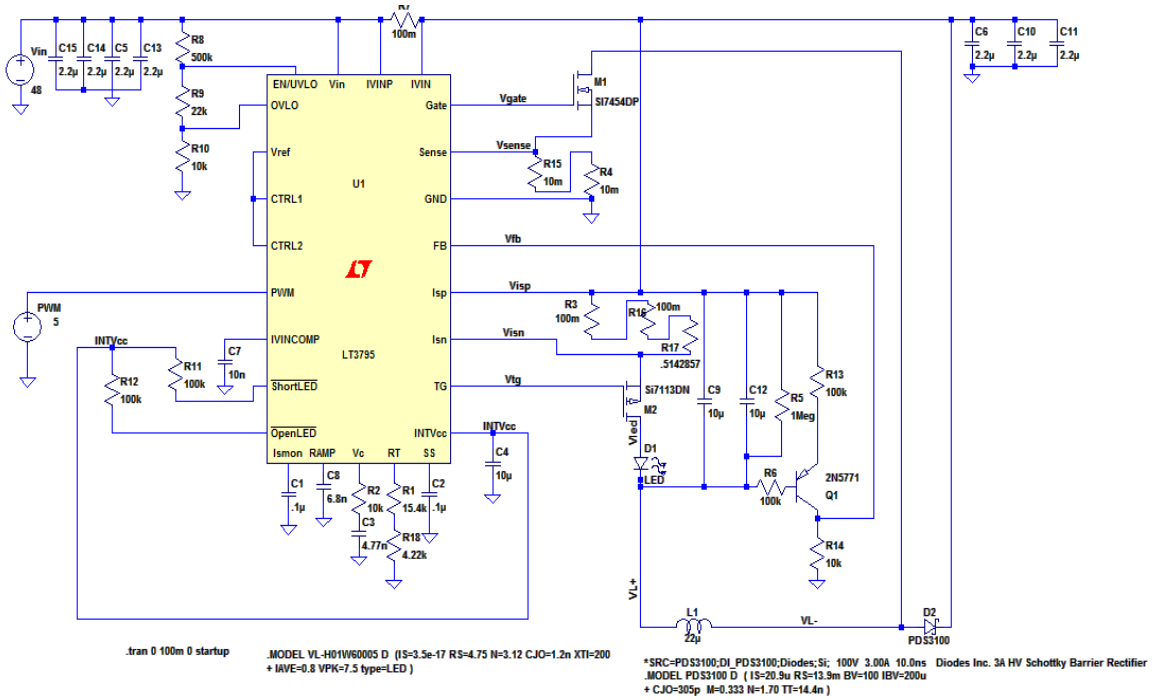


Figure 5-16. Schematic for the LED Driver on PCB

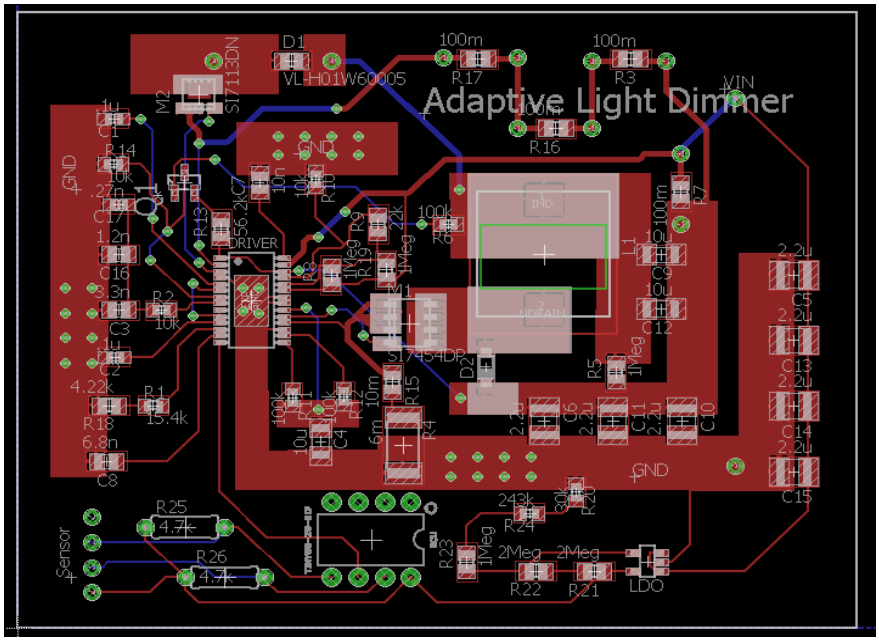


Figure 5-17. PCB Board Design

Figure 5-18. PCB with components soldered

The LED driver and its components were first soldered onto the board. It was tested in a similar manner to the protoboard. The LED driver was connected to a 48 V power supply and an LED, and the input and output currents and the output voltage were measured using digital multimeters. The lux was measured using a lux meter 3 feet below the LED. The test setup is shown in Figure 5-19. Unlike the protoboard, the PCB was able to output significantly higher current, likely due to reductions in noise. Using the voltages and currents, we could determine the input and output power, and thus efficiency. The table of measurements is shown in Table 5-2 and the table of calculated power and efficiency is shown in Table 5-3. Of note is that the output current and voltage are not at the LED's maximum ratings of 12 V and 321 mA. This is likely due to trace and jumper resistance increasing the value of the LED resistors, decreasing the output current. Unfortunately, the maximum efficiency at 48 V was only 61.35%. According to the LT3795 datasheet, the LED driver should ideally be able to reach about 90% efficiency. The lower efficiency could be due to many different factors. A factor is that the output voltage and current are not at maximum, so the output power is not maximized, limiting efficiency. Additionally, there are likely significant switching losses, both in the NMOS, which is switching at 400 kHz, and the PMOS, which is switching at the PWM frequency of 2 kHz. An additional note is that decreasing the input voltage to 22 V increases efficiency to 74.6%, which is a

significant difference from 48 V. An increase in efficiency is expected, but such a large difference can help determine where power is lost.

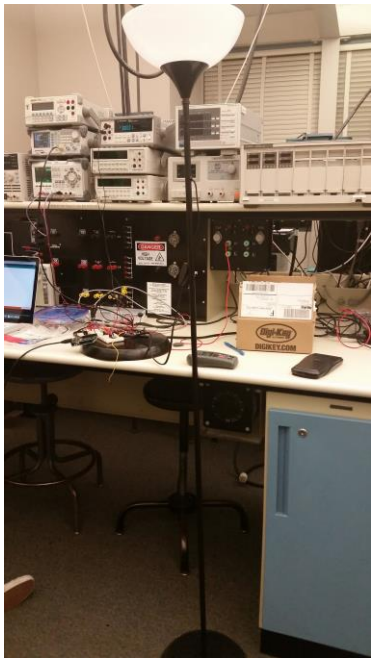


Figure 5-19. The Test Setup for the LED Driver on the PCB

Table 5-2. PCB LED Lux, Current, and Voltage Measurements

Duty Cycle (%)	**Input Current (mA)**	**Output Current (mA)**	**Output Voltage (V)**	**Lux (4 Foot Radius)**
0	2.6152	0	0	0
5	8.433	10.76	7.98	312
10	14.203	18.12	8.18	629
15	17.969	46.63	8.35	991
20	23.231	62.28	8.49	1050
25	28.599	78.05	8.64	1340
30	33.894	93.57	8.77	1750
35	39.382	109.22	8.9	2060
40	44.779	124.73	9.05	2360
45	50.3	140.27	9.18	2680
50	55.93	155.83	9.31	3140
55	61.59	171.49	9.44	3480
60	67.27	187.03	9.58	3780
65	72.97	202.45	9.71	4030

70	78.75	217.98	9.86	4320
75	84.65	233.49	9.99	4650
80	90.65	249.07	10.12	5170
85	96.61	264.44	10.25	5460
90	102.67	279.95	10.38	5730
95	108.74	295.53	10.5	6010
100	114.4	312.2	10.79	6310

Table 5-3. PCB LED Power and Efficiency Calculations

Duty Cycle (%)	Input Power (W)	Output Power (W)	Efficiency
0	0.125	0	0
5	0.404	0.085	0.212
10	0.681	0.148	0.217
15	0.862	0.389	0.451
20	1.115	0.528	0.474
25	1.372	0.674	0.491
30	1.626	0.820	0.504
35	1.890	0.972	0.514
40	2.149	1.128	0.525
45	2.414	1.287	0.533
50	2.684	1.450	0.540
55	2.956	1.618	0.547
60	3.228	1.791	0.554
65	3.502	1.965	0.561
70	3.780	2.149	0.568
75	4.063	2.332	0.574
80	4.351	2.520	0.579
85	4.637	2.710	0.584
90	4.928	2.905	0.589
95	5.219	3.103	0.594
100	5.491	3.368	0.613

The LDO was soldered onto the PCB along with its components. When we were soldering the LDO, we found that we had forgotten to add the output capacitor to the PCB.

However, the ambient light sensor breakout board had an input capacitor, which could be used as the LDO's output capacitor as well. Otherwise, the schematic did not change much since the LDO was previously tested. The output voltage was tested for 5 V using a digital multimeter. Using open circuit voltage measurements, the LDO had an output voltage close to 48 V. Adding a load resistor of 100 kΩ did not change much, but adding the output capacitor of 0.47 μF regulated the output voltage to 5 V. We cannot explain this phenomenon confidently, as the LDO works properly without the capacitor in simulation. However, it is likely that the capacitor is important in regulating the pole of the system, and thus the stability of the output.

After initial tests, it was found that the AnalogOut() function outputs a PWM that was not at a high enough frequency for the LED driver, leading to flickering. To try to remedy this, the prescaler on Timer0 of the microcontroller was changed to increase the frequency. This however interfered with the microcontroller's built in millis() function which is necessary for timing the reading of the sensor. Using a different approach, a new PWM source was created using Timer1. This timer creates a PWM on the same pin as the built in AnalogOut() function without interfering with the reading of the sensor. After this, 10 input light levels were chosen to dim the light to 10 different brightnesses. Once this was functioning properly, it was found that the transitions between output levels was obvious and visually distracting. A function called walk_level was created to smoothly change the brightness of the LED from one level to another by incrementing or decrementing the duty cycle quickly to the next level giving the appearance of a smooth transition. On initial power up of the project, the LED would output full brightness instead of going to an appropriate brightness for the environment. After checking the setup function responsible for initialization, it was found that the OCR1A register that set the PWM duty cycle was initialized to full output after powering on instead of no output. Once this was

changed, the initial light output was correct. Lastly, the sensor would not respond well to drastic changes in surrounding light such as from 300 Lux to 0 Lux. This was found to be because the method of reading the sensor would not go into the if statement that changes the Lux global variable. This meant that the LED output would not change. To fix this, the Lux variable is constantly changed instead of conditionally changed.

After constructing the circuit as a whole on the PCB, testing the individual components, and checking all possible connections, power was applied and the entire system functioned as desired. When dark, below 10 Lux, the LED driver received 100% duty cycle and the lamp went to full brightness and when bright, above 100 Lux, the LED driver receiver 0% duty cycle and the lamp outputs no light. Different duty cycles were assigned to different ambient light levels in between 0 and 100 Lux. The LED dimmed and grew brighter according to how much light reached the sensor, and the brightness changed smoothly without visible leaps in light output.

The issue came when trying to reposition the sensor from the breadboard to the stem of the lamp using relatively long wires of 2 to 3 feet. Because the input capacitor on the sensor breakout board was used as the output capacitor for the LDO, it was very important that it functioned properly. However, it seems the long wires connected to the sensor caused the input capacitor of the sensor to stop functioning as the LDO's output capacitor, causing the LDO's output to swing to 48 V. This destroyed the entire microcontroller/sensor circuit, including the Arduino used as the in system programmer to send code to the ATTiny85. Because the ATTiny85 was destroyed, it could no longer communicate with the sensor. While some microcontrollers were in reserve, they could not be programmed because the Arduino was destroyed. While this was extremely unfortunate, we had an extra Arduino board and an extra ATTiny85 in reserve just for these scenarios. We used these to test the functionality of the

ambient light sensor and found that the light sensor was still operational. Knowing that the output capacitor of the LDO was the problem, we connected a through hole 0.47 µF capacitor from the LDO's output to ground in order to ensure that a capacitor is seen by the LDO regardless of what is done to the microcontroller or sensor. Testing the LDO again, we found that its output was not very consistent, likely due to burning off one of the pads of the resistors during soldering of the PCB. We replaced two of the LDO's resistors with a through hole 1 MΩ resistor and a through hole 220 kΩ resistor. Replacing the resistors caused the LDO to consistently output 5 V. After these changes, the project worked well. It dimmed as expected as ambient light levels increased and increased in brightness during low light. Additionally, by placing the sensor about 2.5 feet above the base, it stabilized the LED's output, as it did not sense enough light from the LED to cause a change in brightness. However, the efficiency further dropped because the microcontroller and sensor together draw about 30 mA, increasing the input power needed, while the output power of the LED stayed constant. This current is significant in our case. Since the input current needed for the LED driver is only about 115 mA, the microcontroller and sensor increased the input current, and therefore input power, by 26%. The high input voltage of 48 V meant that input power increased by about 1.44 W, 1.29 W of which is dropped through the LDO. The efficiency after connecting the microcontroller and LDO was 48.7%. Regardless, the project met its main function of sensing ambient light levels and dimming appropriately. The finished project is shown in Figure 5-20.

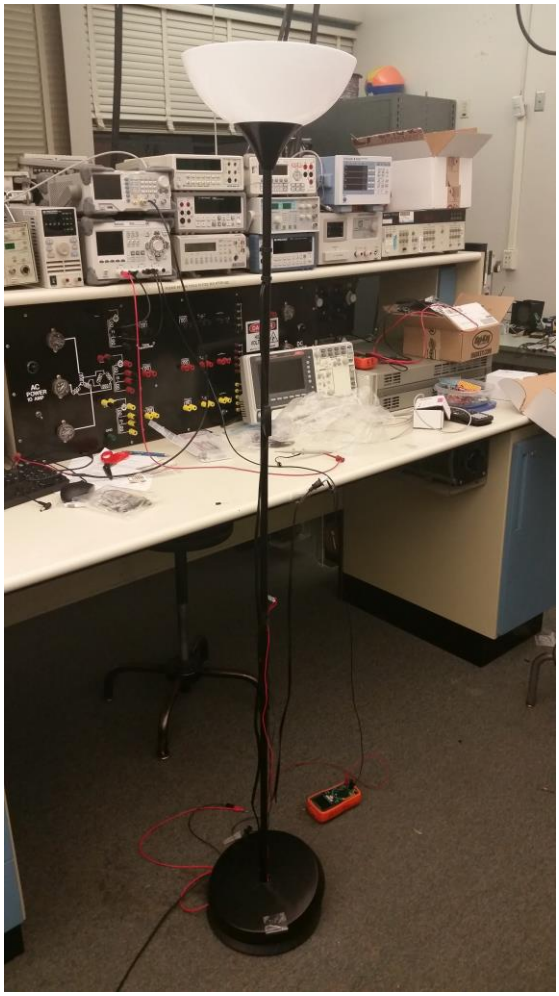


Figure 5-20. The Ambient Light Sensor

Chapter 6. Conclusion

The adaptive light dimmer unfortunately did not meet the efficiency design requirement as it was only 61.35% efficient, 8.65% less than our target of 70% efficiency. The input power at maximum duty cycle was 5.49 W, well below the maximum limit of 7.14 W, but because a different LED was used, the output power was only 3.37 W rather than 5 W. It met the requirement that it must be able to operate off of 48V. The project was also tested at different voltages. At higher than 48 V, the device's output power drops and is unable to regulate properly, but at lower voltages down to 22 V, it operated well, maintaining the same output power and even improving efficiency. The sensor portion of the circuit functioned in accordance with the design requirements and was able to sense light well beyond the ranges originally described. Although the sensor was able to pick up light from 0 to upwards of thousands of Lux, the range used for the project was finally determined to be from 20 to 100 Lux since after testing it seemed to be qualitatively better than 35 to 135 Lux. Because the LED was changed, the maximum output of the LED was only 250 lumens instead of 800 lumens, so it was not as bright as it should have been. The project was quick to respond to changes in the environment, as the LED was able to change more quickly than once every two seconds and each change was gradual. This is important to prevent the LED's dimming from bothering users. The cost of designing the project was about $315. This is much more than the our expected cost of $100, but some of the cost was for buying extra components in the event that any broke during testing.

If we were to do this project over and improve on the procedure of what could have been done better, we would make sure that the 0.47 μF LDO output capacitor is included on the PCB design to ensure that the microcontroller and sensor circuit were receiving 5V instead of 48V. If

someone were to continue this project, they would need to either include this capacitor on a new PCB or connect one close to the supply of the microcontroller and sensor. The easiest way to do this would likely be to connect a 0.47 μF through hole ceramic capacitor between the V_{in} hole to the sensor on the PCB and the ground hole to the sensor on the PCB. Additionally, the LDO would be better tested on the protoboard to the same extent as the LED driver to ensure its consistent operation. This would have avoided destroying our Arduino and ATTiny85.

Another aspect that would have been done differently would be to choose a different microcontroller with I^2C capabilities built in. Choosing a microcontroller with I^2C would allow us to avoid using the TinyWire library which inefficiently bit bangs to get an I^2C reading. This would allow for clearer logic levels when using longer wires, avoiding concerns about the microcontroller reading incorrectly.

Because the PCB did not allow external wires to be connected to the microcontroller, calibrating the light output by reprogramming the microcontroller was done on a breadboard and required additional connections and hardware, making testing more difficult and opening more chances for error. Knowing this, if we were to design the PCB again, we would make extra holes on the PCB to allow for reprogramming of the microcontroller on the PCB, making calibration easier. It would also allow us, or any other person improving the project, to more easily reprogram the microcontroller without needing to desolder and remove it every time a change in the code is needed.

There are various ways that efficiency could be improved. Because there is such a large difference between the LED driver's efficiencies at 48 V and 22 V while the output voltage and current stay constant, it seems that a significant amount of power is lost on the input side of the circuit. This is likely to due to the MOSFET switches. Adding a snubber to the NMOS in the

circuit could significantly reduce switching losses, since the NMOS switches at 400 kHz. The PMOS should have significantly lower switching losses, but it may still be enough to consider a snubber as well. The PCB could be designed more efficiently, as many of the components were farther apart than they needed to be in order to allow us to make any necessary changes during testing. The longer traces may add too much resistance to the smaller resistors, such as the sense and LED resistors, and the inductance of the traces can cause coupling and noise. These resistances were significant enough that they limited the output current to be lower than the rated max current of the LED. If we reached the LED's maximum power output, it likely would have improved efficiency. Additionally, the LED itself was not bright enough for our expectations, so a brighter LED should be used. As the brightness increases, output power would also increase, which should further improve efficiency. Ideally, a high voltage, high current LED should be used. By having high output voltage, on the order of 20 V, the voltage drops and power lost from the diodes and switches should be lower. By increasing both the current and voltage, the LED's output power is increased. This increases the input power, which should cause the input current to increase significantly, as input voltage is constant. By doing this, the extra current drawn by the microcontroller and sensor should be less significant, so they should have a smaller effect on efficiency. The ideal output current should be considered carefully. Since an increase in the output current, and therefore input current as well, can increase losses, the output current should be chosen such that efficiency is maximized. Because the LDO is so inefficient and dissipates much more power than was initially anticipated, it may be beneficial to replace it with a buck mode DC-DC converter. Assuming at least 70% efficiency, the input power would decrease from 1.44 W to 214 mW, with only a 64 mW drop through the converter, as compared to 1.29 W lost

through the LDO. For our project, this would decrease input power from 6.912 W to 5.67 W and increase efficiency from 48.7% to 59.2%.

While the project worked, the major problem is that the efficiency did not meet our requirements. Because efficiency is such an important part of our project, it must be improved before the Adaptive Light Dimmer could be implemented commercially. Additionally, we spent more money on the project than the device could likely be sold at, but much of the price was due to buying components for testing and replacements. The PCB was also an expensive part of the project. The cost of the PCB was even more expensive than what it otherwise would be, as we had it made as a priority and we expedited the shipping, which more than tripled the cost. If the project were to be sold commercially, these can be reduced by buying the components and PCBs in bulk and by designing the PCB more tightly to reduce board area, and consequently costs. If the efficiency were improved and costs reduced, the Adaptive Light Dimmer would be a great way to reduce energy costs without compromising lighting capabilities for any homes using DC power.

References

[1]. "IEA - Energy access database", *Worldenergyoutlook.org*, 2016. [Online]. Available: http://www.worldenergyoutlook.org/resources/energydevelopment/energyaccessdatabase/. [Accessed: 25- Oct- 2016].

[2]. "Without electricity, 1.3 billion are living in the dark", *Washington Post*, 2016. [Online]. Available: https://www.washingtonpost.com/graphics/world/world-without-power/. [Accessed: 25- Oct- 2016].

[3]. "The DC House Project", *Calpoly.edu*, 2016. [Online]. Available: http://www.calpoly.edu/~taufik/dchouse/. [Accessed: 25- Oct- 2016].

[4]. C. Basu *et al*., "Sensor-Based Predictive Modeling for Smart Lighting in Grid-Integrated Buildings," in *IEEE Sensors Journal*, vol. 14, no. 12, pp. 4216-4229, Dec. 2014.

[5]. I. Galkin, L. Bisenieks and A. Suzdalenko, "Impact of pulse modulation method of LED dimmer for street lighting on its efficiency," *Education and Research Conference (EDERC), 2010 4th European*, Nice, 2010, pp. 160-164.

[6]. D. Kathaiyan, "Bringing off-grid electricity to rural villages in India: Opportunities and pitfalls for solar photovoltaics," *2015 IEEE 15th International Conference on Environment and Electrical Engineering (EEEIC)*, Rome, 2015, pp. 1795-1799.

[7]. Y. W. Bai and Y. T. Ku, "Automatic room light intensity detection and control using a microprocessor and light sensors," in *IEEE Transactions on Consumer Electronics*, vol. 54, no. 3, pp. 1173-1176, August 2008.

Appendix A - Microcontroller Code

The following contains the code necessary for the operation of the microcontroller including the code to read the sensor and output PWM, the Tinywire library, and the TSL2561 breakout board library.

Figure A-1. Sensor Read to PWM code

```
/*
   This code is a modified version of the code off the Adafruit website
   It takes in the reading from the light sensor, converts it into Lux,
   and outputs the appropriate PWM.
*/

#include <Adafruit_Sensor.h>
#include <Adafruit_TSL2561_U.h>
#include<TinyWireM.h>

#define PWM_Pin 1   //PWM output to Pin 1

//Threshold value for light levels
#define Level_0 20
#define Level_1 25
#define Level_2 30
#define Level_3 40
#define Level_4 50
#define Level_5 60
#define Level_6 70
#define Level_7 80
#define Level_8 90
#define Level_9 100

//Output values for LED
#define Max 199
#define Out_1 140
#define Out_2 120
#define Out_3 90
#define Out_4 80
#define Out_5 70
#define Out_6 60
#define Out_7 50
#define Out_8 40
#define Out_9 30
#define Min 0

//SDA PB0 Pin 5
//PWM PB1 Pin 6
//SCK PB2 Pin 7


// Set I2C address of TSL2561 light sensor
```

```
Adafruit_TSL2561_Unified tsl = Adafruit_TSL2561_Unified(TSL2561_ADDR_FLOAT, 0x39);

// Global variables for output PWM and Lux value
uint16_t Lux = 0;
int old_lev=0;



// Configures the gain and integration time for the TSL2561
void configureSensor(void)
{
  /* You can also manually set the gain or enable auto-gain support */
  // tsl.setGain(TSL2561_GAIN_1X);      /* No gain ... use in bright light to avoid sensor
saturation */
  // tsl.setGain(TSL2561_GAIN_16X);     /* 16x gain ... use in low light to boost
sensitivity */
  tsl.enableAutoRange(true);            /* Auto-gain ... switches automatically between 1x
and 16x */

  /* Changing the integration time gives you better sensor resolution (402ms = 16-bit data)
*/
  tsl.setIntegrationTime(TSL2561_INTEGRATIONTIME_13MS);      /* fast but low resolution */
  // tsl.setIntegrationTime(TSL2561_INTEGRATIONTIME_101MS);  /* medium resolution and speed
*/
  // tsl.setIntegrationTime(TSL2561_INTEGRATIONTIME_402MS);  /* 16-bit data but slowest
conversions */

}

// walk_level function which moves to the next light level in gradual steps
void walk_level(int new_lev){
  int dir = 0;
  // find direction to go to, up or down
  if((new_lev-old_lev) >= 0){
    dir = 1;
  }
  else{
    dir = -1;
  }
  // loop through and increment or decrement by 1 until at new light level
  while(old_lev != new_lev){
    old_lev+=dir;
    OCR1A = 199 - old_lev;
    delay(10);
  }
  // set new global old light level variable
  old_lev = new_lev;
}

// Setup ATTiny85 and initialize sensor
void setup(void)
{
  //Initialize the sensor
  if(!tsl.begin())
  {
    /* There was a problem detecting the TSL2561 ... check your connections */
    //Serial.print("Ooops, no TSL2561 detected ... Check your wiring or I2C ADDR!");
    while(1);
  }
  // Setup the sensor gain and integration time
```

```
  configureSensor();

  // Set prescaler of 1 on TCCR0B Timer to get PWM freq of 2.5kHz
  pinMode(PB1,OUTPUT);
  OCR1A = 199;  // compare register to be compared with to determine duty cycle
  OCR1C = 199; // compare register that is continuously compared to OCR1A
  TCCR1 = 0b11110101; // initialize timer1 PWM for PB1 aka PWM pin
}

// Loop through reading lux measurements and changing through appropriate output
void loop(void)
{
  // Get a new sensor event
  sensors_event_t event;
  tsl.getEvent(&event);
  // Assign Lux value to variable "Lux"(light is measured in lux)
  if (event.light+1)
  {
    Lux = event.light;
  }
  // Do nothing if no light event
  else
  {
    ;
  }

  // If statements to determine output based on Lux
  if (Lux >= Level_9){
    walk_level(Min);
  }
  if (Lux < Level_9 && Lux >= Level_8){
    walk_level(Out_9);
  }
  if (Lux < Level_8 && Lux >= Level_7){
    walk_level(Out_8);
  }
  if (Lux < Level_7 && Lux >= Level_6){
    walk_level(Out_7);
  }
  if (Lux < Level_6 && Lux >= Level_5){
    walk_level(Out_6);
  }
  if (Lux < Level_5 && Lux >= Level_4){
    walk_level(Out_5);
  }
  if (Lux < Level_4 && Lux >= Level_3){
    walk_level(Out_4);
  }
  if (Lux < Level_3 && Lux >= Level_2){
    walk_level(Out_3);
  }
  if (Lux < Level_2 && Lux >= Level_1){
    walk_level(Out_2);
  }
  if (Lux < Level_1 && Lux >= Level_0){
    walk_level(Out_1);
  }
  if (Lux < Level_0){
    walk_level(Max);
  }
```

```
delay(500); // delay so there are not any sporadic changes in light output
}
```

Figure A-2. Adafruit TSL2561 Code for Reading Sensor and Lux Conversion

```
/**************************************************************************/
/*!
    @file     Adafruit_TSL2561.cpp
    @author   K.Townsend (Adafruit Industries)
    @license  BSD (see license.txt)

    Driver for the TSL2561 digital luminosity (light) sensors.

    Pick one up at http://www.adafruit.com/products/439

    Adafruit invests time and resources providing this open source code,
    please support Adafruit and open-source hardware by purchasing
    products from Adafruit!

    @section  HISTORY

    v2.0 - Rewrote driver for Adafruit_Sensor and Auto-Gain support, and
           added lux clipping check (returns 0 lux on sensor saturation)
    v1.0 - First release (previously TSL2561)
*/
/**************************************************************************/
#if defined(__AVR__)
#include <avr/pgmspace.h>
#include <util/delay.h>
#else
#include "pgmspace.h"
#endif
#include <stdlib.h>

#include "Adafruit_TSL2561_U.h"

#define TSL2561_DELAY_INTTIME_13MS    (15)
#define TSL2561_DELAY_INTTIME_101MS   (120)
#define TSL2561_DELAY_INTTIME_402MS   (450)

/*========================================================================*/
/*                            PRIVATE FUNCTIONS                           */
/*========================================================================*/

/**************************************************************************/
/*!
    @brief  Writes a register and an 8 bit value over I2C
*/
/**************************************************************************/
void Adafruit_TSL2561_Unified::write8 (uint8_t reg, uint32_t value)
{
  Wire.beginTransmission(_addr);
  Wire.send(reg);
  Wire.send(value & 0xFF);
  Wire.endTransmission();
}
```

```
/**************************************************************************/
/*!
    @brief  Reads an 8 bit value over I2C
*/
/**************************************************************************/
uint8_t Adafruit_TSL2561_Unified::read8(uint8_t reg)
{
  Wire.beginTransmission(_addr);
  Wire.send(reg);
  Wire.endTransmission();

  Wire.requestFrom(_addr, 1);
  return Wire.receive();
}

/**************************************************************************/
/*!
    @brief  Reads a 16 bit values over I2C
*/
/**************************************************************************/
uint16_t Adafruit_TSL2561_Unified::read16(uint8_t reg)
{
  uint16_t x; uint16_t t;

  Wire.beginTransmission(_addr);
  Wire.send(reg);
  Wire.endTransmission();

  Wire.requestFrom(_addr, 2);
  t = Wire.receive();
  x = Wire.receive();

  x <<= 8;
  x |= t;
  return x;
}

/**************************************************************************/
/*!
    Enables the device
*/
/**************************************************************************/
void Adafruit_TSL2561_Unified::enable(void)
{
  /* Enable the device by setting the control bit to 0x03 */
  write8(TSL2561_COMMAND_BIT | TSL2561_REGISTER_CONTROL, TSL2561_CONTROL_POWERON);
}

/**************************************************************************/
/*!
    Disables the device (putting it in lower power sleep mode)
*/
/**************************************************************************/
void Adafruit_TSL2561_Unified::disable(void)
{
  /* Turn the device off to save power */
  write8(TSL2561_COMMAND_BIT | TSL2561_REGISTER_CONTROL, TSL2561_CONTROL_POWEROFF);
}
```

```
/**************************************************************************/
/*!
    Private function to read luminosity on both channels
*/
/**************************************************************************/
void Adafruit_TSL2561_Unified::getData (uint16_t *broadband, uint16_t *ir)
{
  /* Enable the device by setting the control bit to 0x03 */
  enable();

  /* Wait x ms for ADC to complete */
  switch (_tsl2561IntegrationTime)
  {
    case TSL2561_INTEGRATIONTIME_13MS:
      delay(TSL2561_DELAY_INTTIME_13MS);  // KTOWN: Was 14ms
      break;
    case TSL2561_INTEGRATIONTIME_101MS:
      delay(TSL2561_DELAY_INTTIME_101MS); // KTOWN: Was 102ms
      break;
    default:
      delay(TSL2561_DELAY_INTTIME_402MS); // KTOWN: Was 403ms
      break;
  }

  /* Reads a two byte value from channel 0 (visible + infrared) */
  *broadband = read16(TSL2561_COMMAND_BIT | TSL2561_WORD_BIT | TSL2561_REGISTER_CHAN0_LOW);

  /* Reads a two byte value from channel 1 (infrared) */
  *ir = read16(TSL2561_COMMAND_BIT | TSL2561_WORD_BIT | TSL2561_REGISTER_CHAN1_LOW);

  /* Turn the device off to save power */
  disable();
}

/*========================================================================*/
/*                            CONSTRUCTORS                                */
/*========================================================================*/

/**************************************************************************/
/*!
    Constructor
*/
/**************************************************************************/
Adafruit_TSL2561_Unified::Adafruit_TSL2561_Unified(uint8_t addr, int32_t sensorID)
{
  _addr = addr;
  _tsl2561Initialised = false;
  _tsl2561AutoGain = false;
  _tsl2561IntegrationTime = TSL2561_INTEGRATIONTIME_13MS;
  _tsl2561Gain = TSL2561_GAIN_1X;
  _tsl2561SensorID = sensorID;
}

/*========================================================================*/
/*                           PUBLIC FUNCTIONS                             */
/*========================================================================*/

/**************************************************************************/
/*!
    Initializes I2C and configures the sensor (call this function before
```

```
    doing anything else)
*/
/**************************************************************************/
boolean Adafruit_TSL2561_Unified::begin(void)
{
  Wire.begin();

  /* Make sure we're actually connected */
  uint8_t x = read8(TSL2561_REGISTER_ID);
  if (!(x & 0x0A))
  {
    return false;
  }
  _tsl2561Initialised = true;

  /* Set default integration time and gain */
  setIntegrationTime(_tsl2561IntegrationTime);
  setGain(_tsl2561Gain);

  /* Note: by default, the device is in power down mode on bootup */
  disable();

  return true;
}

/**************************************************************************/
/*!
    @brief  Enables or disables the auto-gain settings when reading
            data from the sensor
*/
/**************************************************************************/
void Adafruit_TSL2561_Unified::enableAutoRange(bool enable)
{
   _tsl2561AutoGain = enable ? true : false;
}

/**************************************************************************/
/*!
    Sets the integration time for the TSL2561
*/
/**************************************************************************/
void Adafruit_TSL2561_Unified::setIntegrationTime(tsl2561IntegrationTime_t time)
{
  if (!_tsl2561Initialised) begin();

  /* Enable the device by setting the control bit to 0x03 */
  enable();

  /* Update the timing register */
  write8(TSL2561_COMMAND_BIT | TSL2561_REGISTER_TIMING, time | _tsl2561Gain);

  /* Update value placeholders */
  _tsl2561IntegrationTime = time;

  /* Turn the device off to save power */
  disable();
}

/**************************************************************************/
/*!
```

```
    Adjusts the gain on the TSL2561 (adjusts the sensitivity to light)
*/
/**************************************************************************/
void Adafruit_TSL2561_Unified::setGain(tsl2561Gain_t gain)
{
  if (!_tsl2561Initialised) begin();

  /* Enable the device by setting the control bit to 0x03 */
  enable();

  /* Update the timing register */
  write8(TSL2561_COMMAND_BIT | TSL2561_REGISTER_TIMING, _tsl2561IntegrationTime | gain);

  /* Update value placeholders */
  _tsl2561Gain = gain;

  /* Turn the device off to save power */
  disable();
}

/**************************************************************************/
/*!
    @brief  Gets the broadband (mixed lighting) and IR only values from
            the TSL2561, adjusting gain if auto-gain is enabled
*/
/**************************************************************************/
void Adafruit_TSL2561_Unified::getLuminosity (uint16_t *broadband, uint16_t *ir)
{
  bool valid = false;

  if (!_tsl2561Initialised) begin();

  /* If Auto gain disabled get a single reading and continue */
  if(!_tsl2561AutoGain)
  {
    getData (broadband, ir);
    return;
  }

  /* Read data until we find a valid range */
  bool _agcCheck = false;
  do
  {
    uint16_t _b, _ir;
    uint16_t _hi, _lo;
    tsl2561IntegrationTime_t _it = _tsl2561IntegrationTime;

    /* Get the hi/low threshold for the current integration time */
    switch(_it)
    {
      case TSL2561_INTEGRATIONTIME_13MS:
        _hi = TSL2561_AGC_THI_13MS;
        _lo = TSL2561_AGC_TLO_13MS;
        break;
      case TSL2561_INTEGRATIONTIME_101MS:
        _hi = TSL2561_AGC_THI_101MS;
        _lo = TSL2561_AGC_TLO_101MS;
        break;
      default:
        _hi = TSL2561_AGC_THI_402MS;
```

```
        _lo = TSL2561_AGC_TLO_402MS;
        break;
    }

    getData(&_b, &_ir);

    /* Run an auto-gain check if we haven't already done so ... */
    if (!_agcCheck)
    {
      if ((_b < _lo) && (_tsl2561Gain == TSL2561_GAIN_1X))
      {
        /* Increase the gain and try again */
        setGain(TSL2561_GAIN_16X);
        /* Drop the previous conversion results */
        getData(&_b, &_ir);
        /* Set a flag to indicate we've adjusted the gain */
        _agcCheck = true;
      }
      else if ((_b > _hi) && (_tsl2561Gain == TSL2561_GAIN_16X))
      {
        /* Drop gain to 1x and try again */
        setGain(TSL2561_GAIN_1X);
        /* Drop the previous conversion results */
        getData(&_b, &_ir);
        /* Set a flag to indicate we've adjusted the gain */
        _agcCheck = true;
      }
      else
      {
        /* Nothing to look at here, keep moving ....
           Reading is either valid, or we're already at the chips limits */
        *broadband = _b;
        *ir = _ir;
        valid = true;
      }
    }
    else
    {
      /* If we've already adjusted the gain once, just return the new results.
         This avoids endless loops where a value is at one extreme pre-gain,
         and the the other extreme post-gain */
      *broadband = _b;
      *ir = _ir;
      valid = true;
    }
  } while (!valid);
}

/*********************************************************************/
/*!
    Converts the raw sensor values to the standard SI lux equivalent.
    Returns 0 if the sensor is saturated and the values are unreliable.
*/
/*********************************************************************/
uint32_t Adafruit_TSL2561_Unified::calculateLux(uint16_t broadband, uint16_t ir)
{
  unsigned long chScale;
  unsigned long channel1;
  unsigned long channel0;
```

```
  /* Make sure the sensor isn't saturated! */
  uint16_t clipThreshold;
  switch (_tsl2561IntegrationTime)
  {
    case TSL2561_INTEGRATIONTIME_13MS:
      clipThreshold = TSL2561_CLIPPING_13MS;
      break;
    case TSL2561_INTEGRATIONTIME_101MS:
      clipThreshold = TSL2561_CLIPPING_101MS;
      break;
    default:
      clipThreshold = TSL2561_CLIPPING_402MS;
      break;
  }

  /* Return 65536 lux if the sensor is saturated */
  if ((broadband > clipThreshold) || (ir > clipThreshold))
  {
    return 65536;
  }

  /* Get the correct scale depending on the intergration time */
  switch (_tsl2561IntegrationTime)
  {
    case TSL2561_INTEGRATIONTIME_13MS:
      chScale = TSL2561_LUX_CHSCALE_TINT0;
      break;
    case TSL2561_INTEGRATIONTIME_101MS:
      chScale = TSL2561_LUX_CHSCALE_TINT1;
      break;
    default: /* No scaling ... integration time = 402ms */
      chScale = (1 << TSL2561_LUX_CHSCALE);
      break;
  }

  /* Scale for gain (1x or 16x) */
  if (!_tsl2561Gain) chScale = chScale << 4;

  /* Scale the channel values */
  channel0 = (broadband * chScale) >> TSL2561_LUX_CHSCALE;
  channel1 = (ir * chScale) >> TSL2561_LUX_CHSCALE;

  /* Find the ratio of the channel values (Channel1/Channel0) */
  unsigned long ratio1 = 0;
  if (channel0 != 0) ratio1 = (channel1 << (TSL2561_LUX_RATIOSCALE+1)) / channel0;

  /* round the ratio value */
  unsigned long ratio = (ratio1 + 1) >> 1;

  unsigned int b, m;

#ifdef TSL2561_PACKAGE_CS
  if ((ratio >= 0) && (ratio <= TSL2561_LUX_K1C))
    {b=TSL2561_LUX_B1C; m=TSL2561_LUX_M1C;}
  else if (ratio <= TSL2561_LUX_K2C)
    {b=TSL2561_LUX_B2C; m=TSL2561_LUX_M2C;}
  else if (ratio <= TSL2561_LUX_K3C)
    {b=TSL2561_LUX_B3C; m=TSL2561_LUX_M3C;}
  else if (ratio <= TSL2561_LUX_K4C)
    {b=TSL2561_LUX_B4C; m=TSL2561_LUX_M4C;}
```

```
  else if (ratio <= TSL2561_LUX_K5C)
    {b=TSL2561_LUX_B5C; m=TSL2561_LUX_M5C;}
  else if (ratio <= TSL2561_LUX_K6C)
    {b=TSL2561_LUX_B6C; m=TSL2561_LUX_M6C;}
  else if (ratio <= TSL2561_LUX_K7C)
    {b=TSL2561_LUX_B7C; m=TSL2561_LUX_M7C;}
  else if (ratio > TSL2561_LUX_K8C)
    {b=TSL2561_LUX_B8C; m=TSL2561_LUX_M8C;}
#else
  if ((ratio >= 0) && (ratio <= TSL2561_LUX_K1T))
    {b=TSL2561_LUX_B1T; m=TSL2561_LUX_M1T;}
  else if (ratio <= TSL2561_LUX_K2T)
    {b=TSL2561_LUX_B2T; m=TSL2561_LUX_M2T;}
  else if (ratio <= TSL2561_LUX_K3T)
    {b=TSL2561_LUX_B3T; m=TSL2561_LUX_M3T;}
  else if (ratio <= TSL2561_LUX_K4T)
    {b=TSL2561_LUX_B4T; m=TSL2561_LUX_M4T;}
  else if (ratio <= TSL2561_LUX_K5T)
    {b=TSL2561_LUX_B5T; m=TSL2561_LUX_M5T;}
  else if (ratio <= TSL2561_LUX_K6T)
    {b=TSL2561_LUX_B6T; m=TSL2561_LUX_M6T;}
  else if (ratio <= TSL2561_LUX_K7T)
    {b=TSL2561_LUX_B7T; m=TSL2561_LUX_M7T;}
  else if (ratio > TSL2561_LUX_K8T)
    {b=TSL2561_LUX_B8T; m=TSL2561_LUX_M8T;}
#endif

  unsigned long temp;
  temp = ((channel0 * b) - (channel1 * m));

  /* Do not allow negative lux value */
  if (temp < 0) temp = 0;

  /* Round lsb (2^(LUX_SCALE-1)) */
  temp += (1 << (TSL2561_LUX_LUXSCALE-1));

  /* Strip off fractional portion */
  uint32_t lux = temp >> TSL2561_LUX_LUXSCALE;

  /* Signal I2C had no errors */
  return lux;
}

/**************************************************************************/
/*!
    @brief  Gets the most recent sensor event
    returns true if sensor reading is between 0 and 65535 lux
    returns false if sensor is saturated
*/
/**************************************************************************/
bool Adafruit_TSL2561_Unified::getEvent(sensors_event_t *event)
{
  uint16_t broadband, ir;

  /* Clear the event */
  memset(event, 0, sizeof(sensors_event_t));

  event->version   = sizeof(sensors_event_t);
  event->sensor_id = _tsl2561SensorID;
  event->type      = SENSOR_TYPE_LIGHT;
```

```
  event->timestamp = millis();

  /* Calculate the actual lux value */
  getLuminosity(&broadband, &ir);
  event->light = calculateLux(broadband, ir);

  if (event->light == 65536) {
    return false;
  }
  return true;
}

/**************************************************************************/
/*!
    @brief  Gets the sensor_t data
*/
/**************************************************************************/
void Adafruit_TSL2561_Unified::getSensor(sensor_t *sensor)
{
  /* Clear the sensor_t object */
  memset(sensor, 0, sizeof(sensor_t));

  /* Insert the sensor name in the fixed length char array */
  strncpy (sensor->name, "TSL2561", sizeof(sensor->name) - 1);
  sensor->name[sizeof(sensor->name)- 1] = 0;
  sensor->version     = 1;
  sensor->sensor_id   = _tsl2561SensorID;
  sensor->type        = SENSOR_TYPE_LIGHT;
  sensor->min_delay   = 0;
  sensor->max_value   = 17000.0;  /* Based on trial and error ... confirm! */
  sensor->min_value   = 0.0;
  sensor->resolution  = 1.0;
}
```

Figure A-3. TinyWire Code for Implementing I^2C (Wrapper Class)

```
/*
   TinyWireM.cpp - a wrapper class for TWI/I2C Master library for the ATtiny on Arduino
  1/21/2011 BroHogan -  brohoganx10 at gmail dot com

  **** See TinyWireM.h for Credits and Usage information ****

  This library is free software; you can redistribute it and/or modify it under the
  terms of the GNU General Public License as published by the Free Software
  Foundation; either version 2.1 of the License, or any later version.

  This program is distributed in the hope that it will be useful, but WITHOUT ANY
  WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A
  PARTICULAR PURPOSE.  See the GNU General Public License for more details.
*/

extern "C" {
  //#include "USI_TWI_Master.h"
  //#include <USI_TWI_Master.h>
  //#include <USI_TWI_Master\USI_TWI_Master.h>
  //#include <USI_TWI_Master/USI_TWI_Master.h>
}

#include "USI_TWI_Master.h"
#include "TinyWireM.h"


// Initialize Class Variables //////////////////////////////////////////////////
       uint8_t USI_TWI::USI_Buf[USI_BUF_SIZE];             // holds I2C send and receive
data
       uint8_t USI_TWI::USI_BufIdx = 0;                    // current number of bytes in the
send buff
       uint8_t USI_TWI::USI_LastRead = 0;                  // number of bytes read so far
       uint8_t USI_TWI::USI_BytesAvail = 0;                // number of bytes requested but
not read

// Constructors ////////////////////////////////////////////////////////////////

USI_TWI::USI_TWI(){
}

// Public Methods //////////////////////////////////////////////////////////////

void USI_TWI::begin(){ // initialize I2C lib
  USI_TWI_Master_Initialise();
}

void USI_TWI::beginTransmission(uint8_t slaveAddr){ // setup address & write bit
  USI_BufIdx = 0;
  USI_Buf[USI_BufIdx] = (slaveAddr<<TWI_ADR_BITS) | USI_SEND;
}

void USI_TWI::send(uint8_t data){ // buffers up data to send
  if (USI_BufIdx >= USI_BUF_SIZE) return;         // dont blow out the buffer
  USI_BufIdx++;                                   // inc for next byte in buffer
  USI_Buf[USI_BufIdx] = data;
}
```

```
uint8_t USI_TWI::endTransmission(){ // actually sends the buffer
  bool xferOK = false;
  uint8_t errorCode = 0;
  xferOK = USI_TWI_Start_Read_Write(USI_Buf,USI_BufIdx+1); // core func that does the work
  USI_BufIdx = 0;
  if (xferOK) return 0;
  else {                                  // there was an error
    errorCode = USI_TWI_Get_State_Info(); // this function returns the error number
    return errorCode;
  }
}

uint8_t USI_TWI::requestFrom(uint8_t slaveAddr, uint8_t numBytes){ // setup for receiving
from slave
  bool xferOK = false;
  uint8_t errorCode = 0;
  USI_LastRead = 0;
  USI_BytesAvail = numBytes; // save this off in a global
  numBytes++;                // add extra byte to transmit header
  USI_Buf[0] = (slaveAddr<<TWI_ADR_BITS) | USI_RCVE;   // setup address & Rcve bit
  xferOK = USI_TWI_Start_Read_Write(USI_Buf,numBytes); // core func that does the work
  // USI_Buf now holds the data read
  if (xferOK) return 0;
  else {                                  // there was an error
    errorCode = USI_TWI_Get_State_Info(); // this function returns the error number
    return errorCode;
  }
}

uint8_t USI_TWI::receive(){ // returns the bytes received one at a time
  USI_LastRead++;     // inc first since first uint8_t read is in USI_Buf[1]
  return USI_Buf[USI_LastRead];
}

uint8_t USI_TWI::available(){ // the bytes available that haven't been read yet
  return USI_BytesAvail - (USI_LastRead);
}


// Preinstantiate Objects //////////////////////////////////////////////////

USI_TWI TinyWireM = USI_TWI();
```

Figure A-4. USI_TWI Code Needed for Implementing I^2C

```
/****************************************************************************
*
*
* File              USI_TWI_Master.c compiled with gcc
* Date              Friday, 10/31/08        Boo!
* Updated by        jkl
*

* AppNote           : AVR310 - Using the USI module as a TWI Master
*
*             Extensively modified to provide complete I2C driver.
*
*Notes:
*             - T4_TWI and T2_TWI delays are modified to work with 1MHz default clock
*                   and now use hard code values. They would need to change
*                   for other clock rates. Refer to the Apps Note.
*
*       12/17/08       Added USI_TWI_Start_Memory_Read Routine            -jkl
*             Note msg buffer will have slave adrs ( with write bit set) and memory adrs;
*                   length should be these two bytes plus the number of bytes to read.
****************************************************************************/
#include <avr/interrupt.h>
#define F_CPU 8000000UL            // Sets up the default speed for delay.h
#include <util/delay.h>
#include <avr/io.h>
#include "USI_TWI_Master.h"

unsigned char USI_TWI_Start_Transceiver_With_Data( unsigned char * , unsigned char );
unsigned char USI_TWI_Master_Transfer( unsigned char );
unsigned char USI_TWI_Master_Stop( void );
unsigned char USI_TWI_Master_Start( void );

union  USI_TWI_state
{
  unsigned char errorState;         // Can reuse the TWI_state for error states since it
will not be needed if there is an error.
  struct
  {
    unsigned char addressMode         : 1;
    unsigned char masterWriteDataMode : 1;
       unsigned char memReadMode              : 1;
    unsigned char unused              : 5;
  };
}   USI_TWI_state;

/*---------------------------------------------------------------
 USI TWI single master initialization function
---------------------------------------------------------------*/
void USI_TWI_Master_Initialise( void )
{
  PORT_USI |= (1<<PIN_USI_SDA);           // Enable pullup on SDA, to set high as released
state.
  PORT_USI |= (1<<PIN_USI_SCL);           // Enable pullup on SCL, to set high as released
state.

  DDR_USI  |= (1<<PIN_USI_SCL);           // Enable SCL as output.
  DDR_USI  |= (1<<PIN_USI_SDA);           // Enable SDA as output.
```

```
  USIDR    =  0xFF;                      // Preload dataregister with "released level"
data.
  USICR    =  (0<<USISIE)|(0<<USIOIE)|                          // Disable Interrupts.
              (1<<USIWM1)|(0<<USIWM0)|                          // Set USI in Two-wire
mode.
              (1<<USICS1)|(0<<USICS0)|(1<<USICLK)|              // Software stobe as
counter clock source
              (0<<USITC);
  USISR    =  (1<<USISIF)|(1<<USIOIF)|(1<<USIPF)|(1<<USIDC)|    // Clear flags,
              (0x0<<USICNT0);                                   // and reset counter.
}

/*---------------------------------------------------------------
Use this function to get hold of the error message from the last transmission
---------------------------------------------------------------*/
unsigned char USI_TWI_Get_State_Info( void )
{
  return ( USI_TWI_state.errorState );                          // Return error state.
}
/*---------------------------------------------------------------
 USI Random (memory) Read function. This function sets up for call
 to USI_TWI_Start_Transceiver_With_Data which does the work.
 Doesn't matter if read/write bit is set or cleared, it'll be set
 correctly in this function.

 The msgSize is passed to USI_TWI_Start_Transceiver_With_Data.

 Success or error code is returned. Error codes are defined in
 USI_TWI_Master.h
---------------------------------------------------------------*/
unsigned char USI_TWI_Start_Random_Read( unsigned char *msg, unsigned char msgSize)
{
  *(msg) &= ~(TRUE<<TWI_READ_BIT);          // clear the read bit if it's set
  USI_TWI_state.errorState = 0;
  USI_TWI_state.memReadMode = TRUE;

  return (USI_TWI_Start_Transceiver_With_Data( msg, msgSize));
}
/*---------------------------------------------------------------
 USI Normal Read / Write Function
 Transmit and receive function. LSB of first byte in buffer
 indicates if a read or write cycles is performed. If set a read
 operation is performed.

 Function generates (Repeated) Start Condition, sends address and
 R/W, Reads/Writes Data, and verifies/sends ACK.

 Success or error code is returned. Error codes are defined in
 USI_TWI_Master.h
---------------------------------------------------------------*/
unsigned char USI_TWI_Start_Read_Write( unsigned char *msg, unsigned char msgSize)
{

       USI_TWI_state.errorState = 0;                     // Clears all mode bits also

       return (USI_TWI_Start_Transceiver_With_Data( msg, msgSize));

}
/*---------------------------------------------------------------
```

```
 USI Transmit and receive function. LSB of first byte in buffer
 indicates if a read or write cycles is performed. If set a read
 operation is performed.

 Function generates (Repeated) Start Condition, sends address and
 R/W, Reads/Writes Data, and verifies/sends ACK.

 This function also handles Random Read function if the memReadMode
 bit is set. In that case, the function will:
 The address in memory will be the second
 byte and is written *without* sending a STOP.
 Then the Read bit is set (lsb of first byte), the byte count is
 adjusted (if needed), and the function function starts over by sending
 the slave address again and reading the data.

 Success or error code is returned. Error codes are defined in
 USI_TWI_Master.h
---------------------------------------------------------------*/
unsigned char USI_TWI_Start_Transceiver_With_Data( unsigned char *msg, unsigned char
msgSize)
{
  unsigned char const tempUSISR_8bit = (1<<USISIF)|(1<<USIOIF)|(1<<USIPF)|(1<<USIDC)|
// Prepare register value to: Clear flags, and
                                (0x0<<USICNT0);                                     // set
USI to shift 8 bits i.e. count 16 clock edges.
  unsigned char const tempUSISR_1bit = (1<<USISIF)|(1<<USIOIF)|(1<<USIPF)|(1<<USIDC)|
// Prepare register value to: Clear flags, and
                                (0xE<<USICNT0);
                      // set USI to shift 1 bit i.e. count 2 clock edges.
       unsigned char *savedMsg;
       unsigned char savedMsgSize;

//This clear must be done before calling this function so that memReadMode can be specified.
//  USI_TWI_state.errorState = 0;                         // Clears all mode bits also

  USI_TWI_state.addressMode = TRUE;               // Always true for first byte

#ifdef PARAM_VERIFICATION
  if(msg > (unsigned char*)RAMEND)                // Test if address is outside SRAM space
  {
    USI_TWI_state.errorState = USI_TWI_DATA_OUT_OF_BOUND;
    return (FALSE);
  }
  if(msgSize <= 1)                                // Test if the transmission buffer is
empty
  {
    USI_TWI_state.errorState = USI_TWI_NO_DATA;
    return (FALSE);
  }
#endif

#ifdef NOISE_TESTING                               // Test if any unexpected conditions
have arrived prior to this execution.
  if( USISR & (1<<USISIF) )
  {
    USI_TWI_state.errorState = USI_TWI_UE_START_CON;
    return (FALSE);
  }
  if( USISR & (1<<USIPF) )
  {
```

```
    USI_TWI_state.errorState = USI_TWI_UE_STOP_CON;
    return (FALSE);
  }
  if( USISR & (1<<USIDC) )
  {
    USI_TWI_state.errorState = USI_TWI_UE_DATA_COL;
    return (FALSE);
  }
#endif

  if ( !(*msg & (1<<TWI_READ_BIT)) )                // The LSB in the address byte
determines if is a masterRead or masterWrite operation.
  {
    USI_TWI_state.masterWriteDataMode = TRUE;
  }

//      if (USI_TWI_state.memReadMode)
//      {
              savedMsg = msg;
              savedMsgSize = msgSize;
//      }

        if ( !USI_TWI_Master_Start( ))
  {
        return (FALSE);                             // Send a START condition on the TWI bus.
  }

/*Write address and Read/Write data */
  do
  {
    /* If masterWrite cycle (or inital address tranmission)*/
    if (USI_TWI_state.addressMode || USI_TWI_state.masterWriteDataMode)
    {
      /* Write a byte */
      PORT_USI &= ~(1<<PIN_USI_SCL);                // Pull SCL LOW.
      USIDR     = *(msg++);                        // Setup data.
      USI_TWI_Master_Transfer( tempUSISR_8bit );    // Send 8 bits on bus.

      /* Clock and verify (N)ACK from slave */
      DDR_USI  &= ~(1<<PIN_USI_SDA);                // Enable SDA as input.
      if( USI_TWI_Master_Transfer( tempUSISR_1bit ) & (1<<TWI_NACK_BIT) )
      {
        if ( USI_TWI_state.addressMode )
          USI_TWI_state.errorState = USI_TWI_NO_ACK_ON_ADDRESS;
        else
          USI_TWI_state.errorState = USI_TWI_NO_ACK_ON_DATA;
        return (FALSE);
      }

       if ((!USI_TWI_state.addressMode) && USI_TWI_state.memReadMode)// means memory start
address has been written
        {
              msg = savedMsg;                                   // start at slave address
again
              *(msg) |= (TRUE<<TWI_READ_BIT);  // set the Read Bit on Slave address
              USI_TWI_state.errorState = 0;
              USI_TWI_state.addressMode = TRUE;    // Now set up for the Read cycle
              msgSize = savedMsgSize;                           // Set byte count correctly
              // NOte that the length should be Slave adrs byte + # bytes to read + 1 (gets
decremented below)
```

```
              if ( !USI_TWI_Master_Start( ))
              {
                     USI_TWI_state.errorState = USI_TWI_BAD_MEM_READ;
                     return (FALSE);                          // Send a START condition on
the TWI bus.
              }
         }
         else
         {
              USI_TWI_state.addressMode = FALSE;            // Only perform address
transmission once.
         }
    }
    /* Else masterRead cycle*/
    else
    {
      /* Read a data byte */
      DDR_USI   &= ~(1<<PIN_USI_SDA);               // Enable SDA as input.
      *(msg++)  = USI_TWI_Master_Transfer( tempUSISR_8bit );

      /* Prepare to generate ACK (or NACK in case of End Of Transmission) */
      if( msgSize == 1)                             // If transmission of last byte was
performed.
      {
        USIDR = 0xFF;                               // Load NACK to confirm End Of
Transmission.
      }
      else
      {
        USIDR = 0x00;                               // Load ACK. Set data register bit 7
(output for SDA) low.
      }
      USI_TWI_Master_Transfer( tempUSISR_1bit );    // Generate ACK/NACK.
    }
  }while( --msgSize) ;                              // Until all data sent/received.

  if (!USI_TWI_Master_Stop())
  {
       return (FALSE);                             // Send a STOP condition on the TWI bus.
       }

/* Transmission successfully completed*/
  return (TRUE);
}

/*---------------------------------------------------------------
 Core function for shifting data in and out from the USI.
 Data to be sent has to be placed into the USIDR prior to calling
 this function. Data read, will be return'ed from the function.
----------------------------------------------------------------*/
unsigned char USI_TWI_Master_Transfer( unsigned char temp )
{
  USISR = temp;                                      // Set USISR according to temp.
                                                     // Prepare clocking.
  temp  =  (0<<USISIE)|(0<<USIOIE)|                  // Interrupts disabled
           (1<<USIWM1)|(0<<USIWM0)|                  // Set USI in Two-wire mode.
           (1<<USICS1)|(0<<USICS0)|(1<<USICLK)|      // Software clock strobe as source.
           (1<<USITC);                               // Toggle Clock Port.
  do
  {
```

```
        _delay_us(T2_TWI);
    USICR = temp;                           // Generate positve SCL edge.
    while( !(PIN_USI & (1<<PIN_USI_SCL)) );// Wait for SCL to go high.
        _delay_us(T4_TWI);
    USICR = temp;                           // Generate negative SCL edge.
  }while( !(USISR & (1<<USIOIF)) );         // Check for transfer complete.

        _delay_us(T2_TWI);
  temp  = USIDR;                            // Read out data.
  USIDR = 0xFF;                             // Release SDA.
  DDR_USI |= (1<<PIN_USI_SDA);              // Enable SDA as output.

  return temp;                              // Return the data from the USIDR
}
/*---------------------------------------------------------------
 Function for generating a TWI Start Condition.
---------------------------------------------------------------*/
unsigned char USI_TWI_Master_Start( void )
{
/* Release SCL to ensure that (repeated) Start can be performed */
  PORT_USI |= (1<<PIN_USI_SCL);                     // Release SCL.
  while( !(PORT_USI & (1<<PIN_USI_SCL)) );          // Verify that SCL becomes high.
  _delay_us(T2_TWI);

/* Generate Start Condition */
  PORT_USI &= ~(1<<PIN_USI_SDA);                    // Force SDA LOW.
        _delay_us(T4_TWI);
  PORT_USI &= ~(1<<PIN_USI_SCL);                    // Pull SCL LOW.
  PORT_USI |= (1<<PIN_USI_SDA);                     // Release SDA.

#ifdef SIGNAL_VERIFY
  if( !(USISR & (1<<USISIF)) )
  {
    USI_TWI_state.errorState = USI_TWI_MISSING_START_CON;
    return (FALSE);
  }
#endif
  return (TRUE);
}
/*---------------------------------------------------------------
 Function for generating a TWI Stop Condition. Used to release
 the TWI bus.
---------------------------------------------------------------*/
unsigned char USI_TWI_Master_Stop( void )
{
  PORT_USI &= ~(1<<PIN_USI_SDA);            // Pull SDA low.
  PORT_USI |= (1<<PIN_USI_SCL);             // Release SCL.
  while( !(PIN_USI & (1<<PIN_USI_SCL)) );  // Wait for SCL to go high.
        _delay_us(T4_TWI);
  PORT_USI |= (1<<PIN_USI_SDA);             // Release SDA.
        _delay_us(T2_TWI);

#ifdef SIGNAL_VERIFY
  if( !(USISR & (1<<USIPF)) )
  {
    USI_TWI_state.errorState = USI_TWI_MISSING_STOP_CON;
    return (FALSE);
  }
#endif
```

```
  return (TRUE);
}
```

Appendix B - Timeline and Milestones

At the beginning of each quarter, a timeline was established with milestones necessary to complete the project and the report chapters on time. Figures B-1 and B-2 show the weeks of each quarter and what milestones are to be accomplished in those weeks.

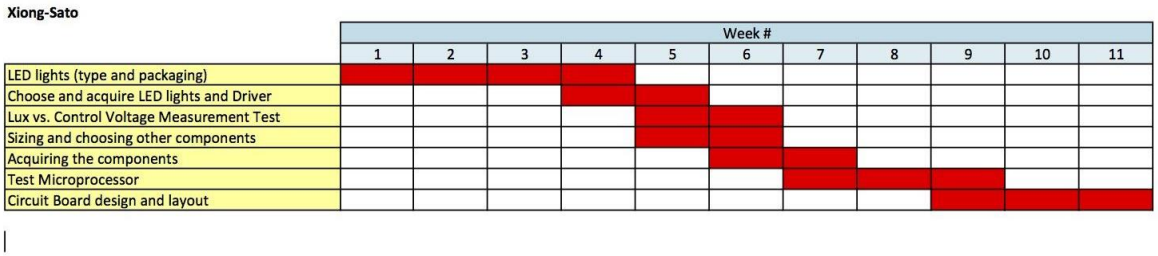


Figure B-1. Winter Quarter Senior Project Timeline

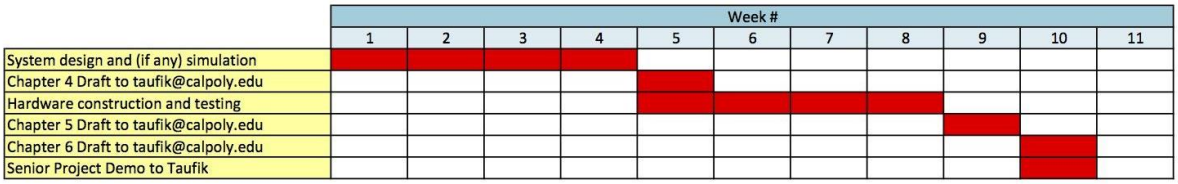


Figure B-2. Spring Quarter Senior Project Timeline

Appendix C - Bill of Materials

Table C-1. Table of Materials used Including Description, Price, and Quantity

Description	**Price (USD)**	**Quantity**
16 Value Resistor Kit	10.79	1
645 Piece Capacitor Kit	19.99	1
Double Sided Protoboard Kit	12.99	1
Ikea NOT Upright Lamp 69-Inch Black/White	18.21	1
SMT Breakout PCB for SOIC 28 or TSSOP-28 to DIP 3 Pack	4.95	1
Solid Core Wire 25ft	2.95	4
Breakaway 0.1” 36-pin strip male header 10 pieces	4.95	1
Adafruit TSL2651 Digital/Luminosity/Lux/Light Sensor Breakout Board	11.90	2
5 Watt Warm White LED	6.95	1
A-15 Natural White 12V LED	2.95	1
ATTiny85 Microcontroller	1.24	3
TSL25711 Light Sensor	1.99	4
SI7454DP Mosfet N-channel	1.63	2
SI7113DN Mosfet P-channel	1.63	2
PDS3100Q Schottky 100V 3A Diode	1.09	2
FMMT593CT PNP 100V 1A BJT	0.45	2
VEML6070 UV Index Breakout Board	5.95	1

TIP32CGOS 100V 3A PNP BJT	0.50	2
Custom PCB Fabrication 3 pieces including priority manufacturing and one-day shipping	102.40	1
Lux Meter	15.99	1
Aggregate Shipping	51	N/A
Total	**315**	

Appendix D - Analysis of Senior Project Design

Project Title: Adaptive Light Dimmer

Student's Name: William Xiong, Jonathan Sato

Advisor's Name: Taufik

1. Summary of Functional Requirements

The Adaptive Light Dimmer regulates the dimming levels of a lighting system depending on the brightness present. The luminaires dim more as brightness levels increase and dim less as brightness levels decrease.

2. Primary Constraints

Implementing this project while keeping costs low presents a significant problem we must consider seriously. Energy efficiency also plays a large role in determining the constraints of the project. It means that we cannot use a linear regulator to step down the high DC input voltage; instead we must implement a buck mode LED driver. This should take care of dropping down the voltage and driving the LED with varying current.

Additionally, testing for the ambient light sensor increases the time needed to implement the project. We had to record the ambient light sensor's outputs at different angles and distances from a light source so that we could program the microcontroller accordingly.

3. Economic

Since manufacturing the device uses machinery, financial capital would increase. Manufacturing equipment prevents human capital due to workers from increasing. However, since the device would need engineers to manage and upgrade the device, human capital would increase. Manufactured capital would increase because the project requires components from other industries, specifically resistors, capacitors, and integrated circuits (ICs). The materials needed for the manufacturing machinery and to manufacture the components would cause a decrease in natural capital. The energy needed for manufacturing would also decrease natural capital.

Costs occur during the design and manufacturing part of the project. Much of the cost of the design comes from labor costs because of the large time investment. The cost of manufacturing

consists of monetary costs, such as for components, energy costs, equipment costs, and environmental costs, due to needing energy resources and materials. Benefits would occur after production, when the device decreases energy usage, thereby decreasing the burden on the environment while also decreasing the monetary cost of the customer.

The project requires 48 V DC to power it. The total project costs roughly $11000, mostly coming from the time and labor costs needed to design and build the project. The designers pay these costs. Besides the labor costs, we initially predicted the components of the project to cost only about $30-$40. We could also need some extra equipment for testing the device, specifically testing the light content at particular distances and angles from the ambient light sensor. This could increase costs to about $75.

The project earns very little when compared to the initial design and manufacturing costs. The project would mainly benefit the customer through saving energy. However, energy savings would also decrease environmental burdens, which implies benefits for all.

The product emerges after building the device and after enough tests show that it meets operational goals. The device should last between 6 months to a year. Because of the autonomous nature of the device, it would not require any maintenance or operation costs; instead, if the product ceased operating, it would need replacement. We estimated development time to amount to the entirety of a school year or 3 quarters. This translates to about 34 weeks of development time. After the project ends, we can further improve it or begin trying to market it. Conversely, we could keep it as reference as we focus our attention on other endeavors instead.

4. If manufactured on a commercial basis

Since the device uses DC power, it's fairly specialized. It's not realistic to have a large amount of sold units, probably in the range of 1000 units sold per year.

Because of the low amount of units sold and the specialization needed for the manufacturing equipment, each device would have fairly expensive costs. However, they cannot cost too much, since that would defeat the project's purpose. Mass manufacturing would also decrease the price. Consequently, each device would cost about $17-$20.

We would need a profit, so each device must cost more than the manufacturing cost. The probable cost would range between $22-$25.

Profit per year would come from the purchase price minus the manufacturing price, then multiplied by the amount of units sold per year, resulting in a yearly profit of $5000. Due to the initial cost of about $11000, it would take about 2 years and 3 months for us to break even.

Because the device operates automatically, users spend essentially no time to operate it except in the initial installation. This would last until the product breaks, which realistically would occur between 6 months to 1 year. As a result, no cost comes from operating the device.

5. Environmental

The manufacturing process of the device makes use of energy resources, specifically fossil fuels, and materials, such as plastics, silicon, and metals. The energy usage occurs throughout the manufacturing process, starting from the creation of the components to putting the components together to form the final product. Similarly, material usage occurs during the creation of the components, silicon for the ICs and plastics for the packaging, and during assembly, soldering and the manufacturing equipment themselves.

The project uses energy resources directly during the manufacturing process and during operation. It uses material resources during manufacturing as well, through the manufacture of the components.

Since any energy use could come from fossil fuels, the project's manufacturing process could indirectly create greenhouse gases, which would affect the entire biosphere. In the long run, however, the project would help save energy, decreasing greenhouse gases. Because of the usage of plastics, which do not decompose, if the user does not take the proper steps, the project could cause harm for ecosystems after the product breaks down. Similarly, the electronic components would also cause similar harm if not taken care of properly.

Because the project would save energy in the long run, it would help decrease greenhouse gas emissions, helping all species. However, if users do not take the proper steps, the usage of plastics and silicon could cause potential harm to ecosystems, such as fish, plants, and animals displaced by landfills. The landfills themselves could also produce dangerous gases. Most of the gas consists of methane and carbon dioxide, greenhouse gases that could hurt all life on Earth. Though much of the gas consists of methane and carbon dioxide, trace amounts of more dangerous gases exist that could cause more direct harm to local wildlife.

6. Manufacturability

Possible challenges with manufacturing consist of finding the machinery needed to perform delicate procedures on the small area of our project. It would likely cost a large amount of

money to obtain such machinery, which would eventually pass on to the customer. This would go against our goal of decreasing the customer costs.

7. Sustainability

Since the device would essentially operate at all times, this could lead to damage of the device. Though very small, the power usage from constant operation could cause damage to components in the device. However, since the device does not turn "on" and "off, high current surges would not exist in the device. This means high power spikes, the source for many device failures, would not cause problems for the device, increasing its lifespan. In the case that the device breaks, it's unlikely that repairing it would cost less than replacing it, since only an expert could possibly find and fix the problems.

Manufacturing the project would increase usage of material resources, such as silicon, plastics, etc., but the savings in energy usage would help to counteract this by decreasing the impact on energy resources like natural gas and fossil fuels.

Improvements of the project's operational capabilities would likely come at an increased cost to the project as a whole. Depending on applications, we may or may not desire this. For our purposes, because of the importance of cost, any ways to decrease the project's costs while maintaining acceptable operation would constitute an improvement.

Because of the compact nature of the project, when upgrading the project's design, we must have significant changes in the manufacturing process. Ensuring that no extra problems arise from changing the design requires more thorough testing afterwards than a more spacious design.

8. Ethical

IEEE Code of Ethics

i. Because the project uses high voltages, it may cause some dangers if handled improperly. As a result, customers would obtain written information about how to handle the product, and the potential dangers if mishandled.

ii. Potential conflicts could occur throughout the project, such as conflicts in design principles, manufacturing issues, and any potential problems that customers could find with the product after its use. We must keep these potential conflicts in mind when designing and building the project, and disclose them to customers and other involved parties.

iii. Documents pertaining to the project would have realistic claims and estimates with proper sources and citations for those claims.

iv. It's not likely for bribery to occur for this project as it doesn't have much tactical or political advantages.

v. This project could definitely improve technology because it uses DC power, as opposed to the AC power most used in the developed world. As a result, it explores an area of electrical engineering that could be significantly improved.

vi. Only professional engineers and technicians would maintain and upgrade the project.

vii. Many possible criticisms for the project could occur since we can only run through relatively few iterations. These criticisms could help the project improve.

viii. Because the device uses DC power, it's unlikely for it to have prevalent use in more developed countries with AC power. As a result, location plays an important role in the potential users of the technology.

ix. The project could potentially cause harm to the customers and their surroundings if handled improperly. As an example, the project could catch fire, produce sparks, or electrocute a person if damaged. As a result, the project must contain extra safety precautions built into it.

x. This project does not hinder other professionals in their endeavors; on the contrary, the device can increase their knowledge on the subject and improve their own progress.

Utilitarianism

During design, engineers benefit because of increased job opportunities. During manufacturing, benefits accrue due to increased productivity for the companies involved, such as the companies building the components, the shipping companies that move goods around, and the companies that build the manufacturing equipment. However, because of using machinery to decrease manufacturing costs and thus decrease purchase costs, workers that could build the project lose potential jobs. Costs also occur due to energy usage and material usage from manufacturing. However, because the project's operation reduces energy usage, it counteracts the increase during manufacturing. Any greenhouse gas reduction improves the world's environmental stability. Because the project decreases energy usage and thus greenhouse gas emissions, it improves the environment, benefitting all people, animals, and plants. On the other hand, however, if customers do not properly dispose of the device after its usage, it could cause environmental harm by increasing landfills. These landfills can produce greenhouse gases and small amounts of more immediately dangerous compounds. These factors could counteract much of the environmental benefits from decreased energy usage.

9. Health and Safety

When manufacturing the project, health concerns arise due to the increased productivity requiring more energy investments. This means an increase in the usage of fossil fuels and other energy resources, increasing safety concerns for greenhouse gases. As a whole, though, because the device saves energy through its operation, the project should save energy usage

and decrease health concerns. During usage, because the device uses high voltages, without the necessary precautions, any problems with the device could cause dangerous situations, like fires and electrocutions.

10. Social and Political

The project's design process could potentially help other engineers in their designs, specifically those working with DC power. Also, because the project uses DC power, it could help show some benefits of DC power. The manufacturing process makes use of silicon, plastics, and electrical materials, uses energy resources, and uses manufacturing machinery. Each of these must follow their own government dictated regulations. The silicon, plastics and other electrical materials would have to meet safety and performance standards before use. Power companies would need to monitor the large scale energy usage to minimize safety or economical problems. The equipment used would have to pass safety regulations before usage. During the product's usage cycle, safety problems could occur if the product malfunctions. Otherwise, the product helps decreases energy usage, helping the power companies and decreasing environmental burdens.

The project directly impacts engineers that design and upgrade it by giving them a job. Other jobs created by the product, such as marketing jobs, would also benefit similarly. It would also directly impact customers, since they pay for the product. Indirectly, the project impacts power companies by decreasing energy usage; it helps IC manufacturing companies, silicon growing companies, companies that create the manufacturing tools, and shipping companies by giving them more demand for their services and products; and by decreasing greenhouse gases, the project indirectly benefits all other life on Earth.

Stakeholders do not benefit equally. The employees working on the project would benefit the most, as it provides them a job. Customers would benefit due to decreases in their electricity bills. These customers must live in houses that use DC power, so they would likely not live in the United States or other places with established AC power companies. The companies involved in manufacturing would benefit because of the increase in their productivity. However, other people would benefit mainly from the environmental implications. Since the employees, customers, and companies would benefit from the environmental improvements as well, they would benefit more than any person not directly involved with the project. The companies that use materials would likely pay the most, since they would lose materials needed for their continued operation.

11. Development

We learned better simulation techniques to test the design properly before physically building it. Otherwise, testing costs would have reached values we could not cover. Additionally, we

needed to completely understand each component's operation, specifically the ambient light sensor, the microcontroller, LDO, and the LED driver, before we can optimize the project's design. This required reading the datasheet and forming testing techniques to characterize the components. Lastly, an understanding of PCB design and PCB software was needed to produce a functioning final circuit.

See References for the literature search.