

Project Skylight

Project Report

Eric Brauer



Subject: Technology Project - TPJ655

Program: Computer Engineering Technology

Instructor: B. Shefler, rm. A4022, ext. 26429

Date: 4/4/2017

	1
Executive Summary	2
Introduction	3
Functional Features	3
Product Specifications	3
Operating Instructions	3
Design	4
Hardware Block Diagram	4
Software Flow Chart	5
Description of Components	6
Graphical User Interface Mockups	7
Theory of Operation	7
Maintenance Requirements	10
Conclusion	10
Further Steps	11
Appendix	12
Schematic	12
Bill Of Materials	13
Datasheets	13
Contact Information	13
On the CD:	14

Executive Summary

Project Skylight is an Internet-connected light that simulates daylight cycles. It's meant for use in places where there are no external windows, and it can be used to get a quick visual reference for the time of day. Project skylight is composed of a small microcontroller with wifi capabilities, and 16 RGB LEDs. The device is meant to run with very little set-up, but a web-based user interface can be used for further configuration. In the report, the operation and specifications of Project Skylight are covered in more detail.

Introduction

In Toronto and in many densely-populated areas, people often and work in rooms without exterior windows. With no visual reference to the time of day, it can be easy to lose track of time: whether it's daytime, evening, or night. This leads to feelings of disorientation and can exacerbate sleep disturbances.

Project skylight is meant to address this problem. The device being proposed is an Internet-connected light that simulates the time of day. The time of day is simulated with an appropriate colour and brightness of light. Daytime, nighttime, and transitional periods ie. dawn and dusk would have different characteristics which would give people the same quick visual reference that one usually gets by glancing out the window.

Functional Features

The product requirements, as envisioned, are as follows:

- Project Skylight (PS) asks for WiFi credentials, and connects to a WiFi router.
- PS serves a webpage which is used to configure the device and to provide the user's city of residence.
- PS uses the OpenWeatherMap API to get the time of dawn and dusk. The device also uses an NTP server to synchronise the current time.
- PS changes the colour and brightness of LEDs based on current time in reference to sunrise and sunset times.
- PS monitors a potentiometer, which will be used to dim or turn off the LED light.
- Further configuration is possible from the web interface. This includes setting manual daylight times.

Product Specifications

- Powered from micro USB port
- Operates at nominal indoor temperatures, not for outdoor use
- Luminosity of around 200 - 400 Lumens
- WiFi range 20m
- Current consumption no higher than 1.5A

Operating Instructions

Before proceeding, please observe the following safety instructions:

For indoor use only. Keep away from water or heat sources. Use only power adapters approved by the CSA.

Initial Setup:

- Project Skylight can be used with any micro-USB power adapter that supplies 5V. It should also be able to supply 1A. Plug the Skylight into the wall, and hang it or place it wherever you like.

- On startup, the light will flash yellow. This means that the skylight was unable to detect a wifi station in memory, and is in station mode so that wifi information can be entered.
- Use a smartphone or laptop to connect to the Access Point called 'SKYLIGHT.' Once connected, use the browser to go to **skylight.local**.
- Select your home router, and enter the password. The skylight will now attempt to connect to the internet. If it is successful, you should see it simulate the time of day. If there is a problem, the light will blink red. Please check your wifi information again, or contact customer support.

Changing brightness:

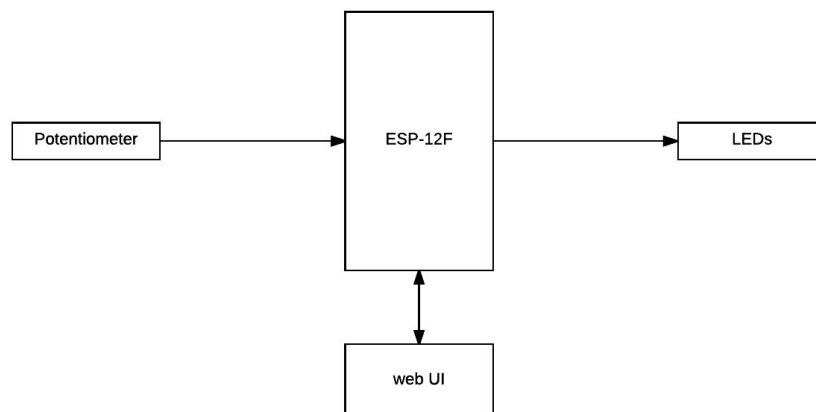
- Use the dial or adjust desired brightness. These settings will be remembered until you change them. *If the skylight doesn't seem to be responding, check first that the light hasn't been turned off.*

Changing transition times:

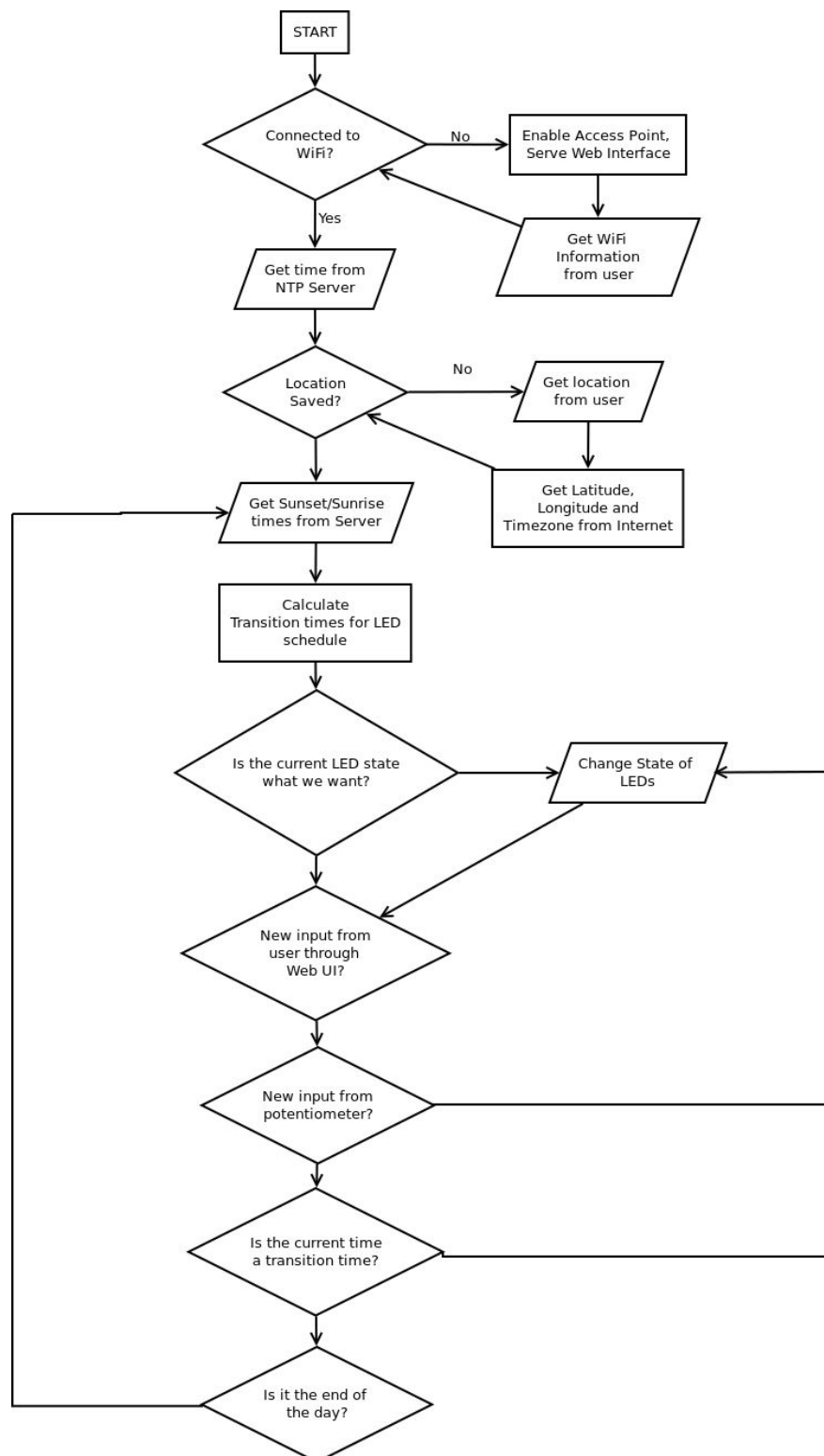
- If you wish to change the time at which sunrise or sunset are initiated, use a device connected to your home router. Open a browser window and go to **skylight.local**.
- Set a time for sunrise or sunset, and press the 'save' button. If the time entered is valid, the skylight will transition 30 minutes before the specified time, and the transition will be completed 30 minutes after. *Manual times do not repeat. After the specified time, the skylight will revert to simulating the actual time of day.*

Design


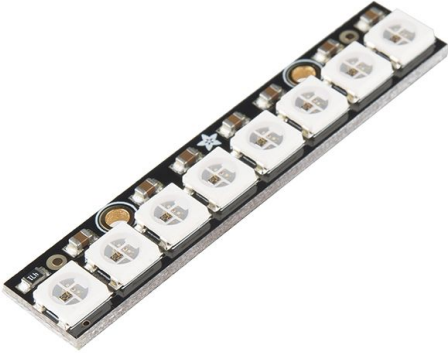
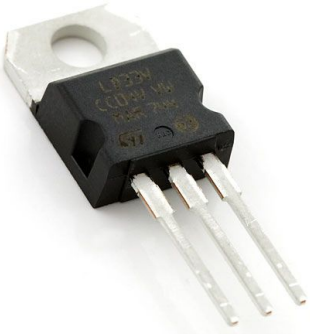
Hardware Block Diagram



Software Flow Chart



Description of Components

#	Component Name	Specifications
1	Espressif ESP-12f 	<ul style="list-style-type: none"> • Operating Voltage: 3.0~3.6V • Operating Current Average value: 80mA • SDIO 2.0, (H) SPI, UART, I2C, I2S, IRDA, PWM, GPIO • 802.11 b/g/n • Integrated low power 32-bit MCU, 80 MHz clock speed • 36 kB RAM, 2 MB flash
2	Adafruit Neopixel Strip 	<ul style="list-style-type: none"> • Provides 8 WS2812b Addressable RGB LEDs • Voltage VCC: +6.0~+7.0V • Operating current: 60mA • 256 brightness levels per colour
3	LD1117V33 Voltage Regulator 	<ul style="list-style-type: none"> • Output Voltage: 3.3V • Output Current: 800mA • Dropout Voltage: 1V

Graphical User Interface Mockups

Skylight

Time is now: 10:12:48

Set Sunrise:

Hour: Minute: AM

Set Sunset:

Hour: Minute: PM

Theory of Operation

Power Concerns

The ESP-12f requires a typical voltage of 3.3 Volts. The Neopixel array require a voltage of 4 - 7 Volts. Thus a voltage regulation stage is required. USB provides 5 Volts, and this power is used for the Neopixels. Power for the ESP-12f is supplied by an LD1117.

Power consumption is based heavily on several factors: transmission and reception by the WiFi module on the ESP-12f, signal strength of the same, and brightness levels of the Neopixels. If a single Neopixel is set to white with maximum brightness, then each of its three RGB LEDs requires 20 mA. There is never a situation when the Skylight is generating this state, however. Full brightness of the daytime SkyState would represent the maximum current consumption.

4 LEDs displaying white = $60\text{mA} \times 4$	240mA
12 LEDs displaying cyan = $40\text{mA} \times 12$	480mA
Total Current for Neopixels:	720mA

In addition, the ESP-12f requires a maximum of 100mA. Meaning that a USB adapter rated at 1A should be sufficient for proper function and also provide a buffer of almost 20%.

Initial State / Connectivity

The ESP-12f is a popular IC among the maker community, and much work has been done to implement a thorough Arduino library which exposes much of the connectivity capability of the chip. The Skylight in particular makes use of several libraries to implement the initial Access Point state (used to enter WiFi credentials), to communicate to OpenWeatherMap, and to serve a web-based user interface.

ESP8266WiFi exposes functions for connecting to a WiFi access point and communicating over TCP/IP.

ESP8266WebServer allows the ESP to also act as a web server, and serve HTTP code to clients.

ESP8266mDNS provides a 'micro DNS' server which will associate a URL with whatever IP address is assigned to the ESP's web server.

WifiManager checks flash for a WiFi SSID and password that was previously used to connect. If these don't exist, WifiManager will set the ESP-12 to station mode with an SSID set in software. The user can connect to this access point, and then navigate to the url specified by the micro-DNS server. From this webpage, the user has the option to scan for available access points, and then enter a password to connect to this access point. From this point, the ESP exits station mode and attempts to connect to the access point as a client. If it is successful, these credentials will be saved in flash memory to be used on startup.

The Web Interface

The ESP handles web requests directed at the url created by the micro DNS server. On the Skylight, the url is skylight.local. The main interface is located at the root. Other webpages are used to acknowledge user requests, and exist at /submit, /demo and /reset. All other requests will result in a HTTP code of 404.

A typical user request is to manually change the time at which a sunset event occurs. The time desired is typed into text boxes, and the 'save' button is pressed. Information inside the text boxes is saved as arguments. The arguments are validated first. The arguments are checked to ensure that they are numerical and in a proper 12-hour time format. A bad request will result in a 'bad request' message and a link back to the root page. A good request will be accepted. Finally, the browser will navigate to skylight.local/submit, where the user is notified that the alarm has been set.

Adafruit NeoPixel Array

Neopixels are addressable RGB LEDs. Control circuitry, LEDs, and internal oscillators are all included in one package. Data can be received from a microcontroller at speeds of 800kbps, and outputted to the next neopixel. This allows an array of neopixels to be controlled from one data pin.

Each neopixel requires 24bits of information. The first neopixel uses 24 bits to set the values of discrete red, green and blue LEDs. Each of these LEDs takes a PWM 8-bit signal to set its intensity, resulting in 255 possible states. Once the state of neopixel has been set, subsequent data is amplified and sent through a 'Data Out' line to program the next neopixel. The advantage of neopixels is their flexibility, but larger arrays will require far more memory and processing than some microcontrollers can provide. Since the Skylight uses only 16 neopixels, performance requirements are relatively light.

Adafruit provides an Arduino Library for communicating with the Neopixel array. The Neopixels are powered from a 5V VCC line and the data line is connected to the ESP's GPIO pin 14.

Brightness Control

A potentiometer is connected to the ESP-12's single Analog-to-Digital converter. A standard Arduino library function is used to convert this number to a 10-bit value, which is then to an 8-bit value using bit shift. The 8-bit value is used as an argument for the `setBrightness` function which is part of the Adafruit Neopixel library. Using the ADC with interrupts proved to be problematic, and so the ADC is polled in `loop()` and if its value changes, the `setBrightness` function is called. The potentiometer is connected to the ADC pin identified as 0.

Setting dusk_time and dawn_time From OpenWeatherMap.org

OpenWeatherMap provides an API that can be used for free with registration for a smaller number of requests. For the Skylight prototype, this is sufficient. Documentation for the API is located at <http://openweathermap.org/current>. The API can be used with several different formats of location data; for simplicity's sake a GET request is used encoded with the city and country code are used along with an API key.

The HTTP response is in a JSON format. First, the HTTP header of the response is discarded. Then `ArduinoJson` library is used to parse the response. Most information is related to weather conditions and is not used. The sunrise and sunset times are sent in epoch time based on UTC. In order to set this to local time, a timezone offset is added to these times and the result is stored in memory.

Using NTP to Sync Local Time

`NTPClient` is another library used to communicate with NTP servers. This is used with `TimeLib` to set the ESP's time. A timezone offset is required by the library.

State Machine

Assuming that NTP and GET Requests have succeeded, the Skylight will compare current time with the next scheduled event. Once current time is within forty minutes of a sunrise or sunset, the Skylight enters a DUSK or DAWN event. Both events are identical, but DUSK occurs in reverse order.

There are three parts to a transition, and each takes 20 minutes. Two of the parts of the transition occur before the event time, and one occurs after. Each part of the transition is handled by a function, and each function takes as an argument an 8-bit number.

SkyState1(0-255)	SkyState2(0-255)	SkyState3(0-255)
<i>Duration: 1200 seconds</i>	<i>Duration: 1200 seconds</i>	<i>Duration: 1200 seconds</i>

In the case of DAWN, the actual `sunrise_time` is the time between `SkyState2` and `SkyState3`. In order to set the argument for each `SkyState` function, a simple equation derived from the Radiometric Equation is used.

$$y = (x - 2400) / (-3600 / 765)$$

Where x is equal to seconds until event and y is equal to the skyState argument. What this means is that approximately every 7 seconds, SkyState functions will increment the levels of the Neopixel LEDs which results in a simulated dawn or dusk event. Once the current time is over twenty minutes past the last event, the state machine will return to a state where it decides if it is waiting for dawn or waiting for dusk. If both dawn and dusk events are earlier than the current time, the state machine will attempt to request new times from OpenWeatherMap.

Maintenance Requirements

The skylight is designed to run without any interference whatsoever. The LEDs used should last for years, and there should be few if any problems. The exception to this relates to internet connectivity.

If the skylight flashes red, there could be problems connecting to the server required for time synchronisation.

- Unplug the skylight, then plug it back in. This should cause the system to reboot, and to check connectivity. If the device flashes red after a reboot, check that your home router is connected to the internet.
- It is possible that a software update will be required to ensure proper function. To initiate an OTA update check, reboot the skylight. **Note: This is not yet implemented.**

If the device seems to be responsive but there are no lights, first check that the brightness dial is not turned off. If lights fail to engage, check that the power adapter is not faulty.

Conclusion

Project Skylight fulfills the features as they were laid at the outset of the project. The device syncs with NTP, performs GET requests, servers a web UI and does a good job of simulating time of day. In addition, it is easy to set up and runs well without interference. The cost of materials for this project was under \$50 and could easily be dropped down, possibly under \$20 on a larger scale.

The question of what would need to change in order for this to be mass produced is an interesting one. The price of materials would obviously drop significantly, but several new challenges would arise.

- Security / Privacy: The Skylight is quite secure by default. It does not require any personal information to be logged apart from WiFi credentials, and the web interface is only accessible to people who are connected to the same WiFi. While the information sent to OpenWeatherMap is not protected, it only includes the user's city of residence and so is not sensitive. There is one possible vector of attack: during initial setup, the Skylight sets up a station and gets the user's wifi credentials. This station could easily be spoofed and the attacker would gain the user's WiFi credentials. If this were to be commercial product, this and other possible vectors of attack would need to be addressed.
- API: The Skylight uses a free API from OpenWeatherMap, but this wouldn't be available to a commercial product without service fees. It might make more sense for

a business producing Skylights to set up their own server for requests. Of course, if the business went bankrupt then shutting down the server would make the Skylight useless. Such is the reality of IoT devices.

- **Time / Location:** The difficulties of timezones and locations are myriad and would present some challenges for a commercial product. In the present prototype, location and timezone offset are hard-coded. A commercial product would need to get this information from the user, or otherwise get it from the device. Inputting the city of residence from the user is not a good solution, since it excludes people living outside of cities. In addition, if there are multiple cities with the same name, this would also cause a problem. The best solution would be city ID codes, but users don't know these. A separate API call would be required: users would enter their city, be presented with a list of possible cities, and then the city ID code of their selection would be used for all future requests. Timezones require an additional request, since timezone information is not part of OpenWeatherMap's API. These challenges would easily be addressed by a custom server and API.
- **Enclosure:** The Skylight includes an array of Neopixels placed behind a diffusion panel held in a 4" x 6" picture frame. This solution is unwieldy, and is different from the proposed enclosure, which was envisioned to hang from a wall. Placing the LEDs closer to the frame didn't produce the desired effect. A more appropriate enclosure would need to be developed for a viable commercial product.

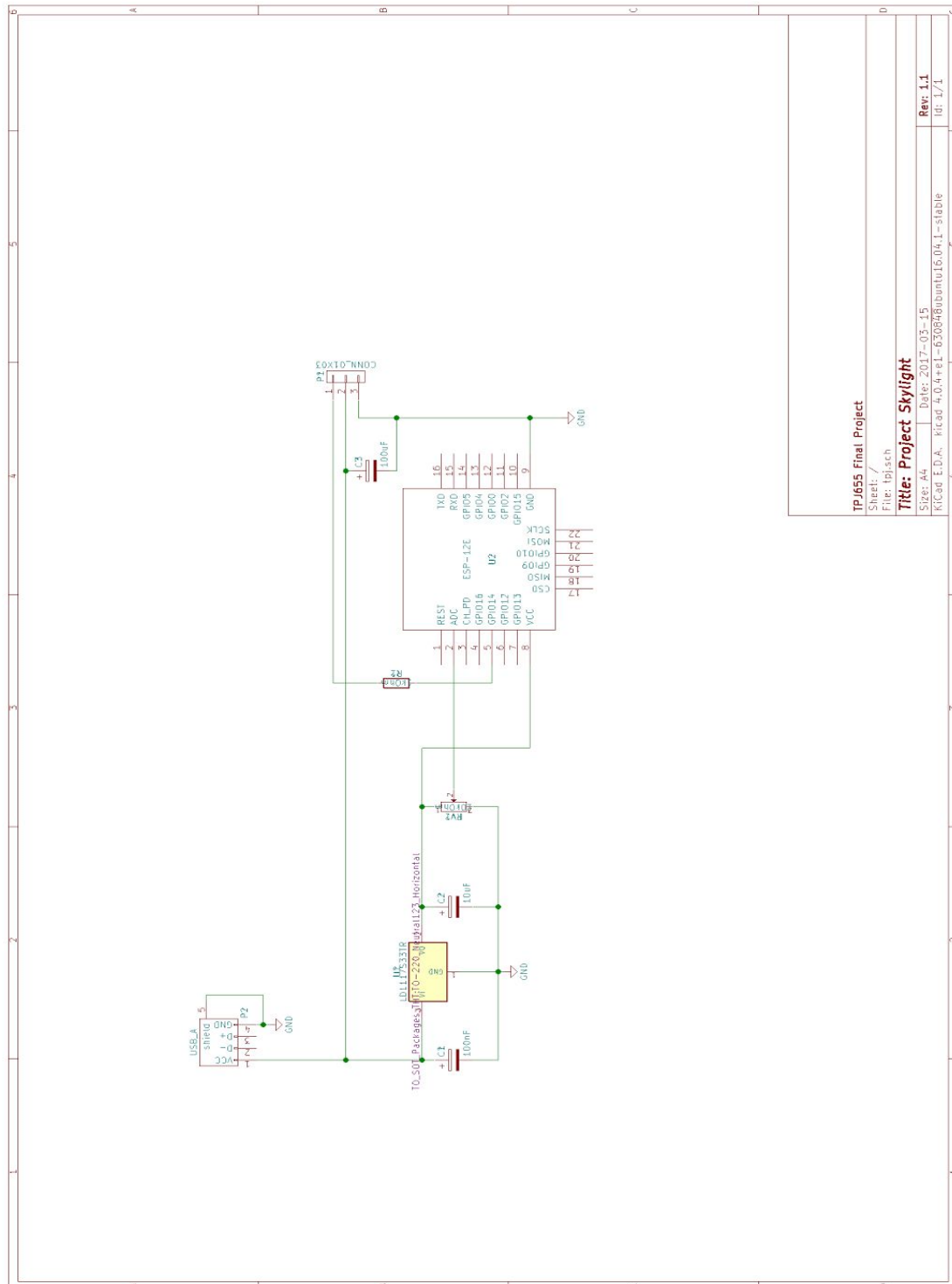
Further Steps

The following features were not defined as requirements, but would be further enhancements that could be implemented:

- **Themes:** Instead of time of day, the user could select themes from the user interface. Some possible themes include: fireplace, aquarium, lava lamp, and possibly others.
- **Light timers:** People often buy light timers to protect their homes when they are on vacation. The timers turn on lights at set times, in order to fool potential burglars. PS would have the same functionality.
- **Brightness monitor:** By including a brightness monitor, the PS could detect when the user has turned off their other bedroom lights. From this information the PS could monitor how late the user is staying up past 'sunset.' This information could be uploaded to a database, and made accessible to the user for perusal.
- **Integration with MQTT notifications:** MQTT is a technology where IoT devices can generate push notifications to users' mobile devices. The PS could be used to generate gentle light notifications: possibly blink an LED to get the user's attention.

Appendix

Schematic



Bill Of Materials

Part #	Model	Designation	Description	Source	Cost per Unit	Quantity	Subtotal
1	ESP-12F	U2	WiFi/MCU	Ebay	\$2.45	1	\$2.45
2	NeoPixel Strip	N/A	LEDs	Creatron	\$6.95	2	\$13.90
3.	10kOhm log	RV1	Potentiometer	Home Hardware	\$1.00	1	\$1.00
4.	USB B female port	P2	9V DC Jack / USB	Creatron	\$2.35	1	\$2.35
5.	LD-1117	U1	Regulator: 3.3V out	Creatron	\$2.45	1	\$2.45
6.	10uF, 100uF Electrolytic Caps.	C2, C3	Capacitor	Home Hardware	\$0.45	2	\$0.90
7.	0.1uF ceramic Cap.	C1	Capacitor	Creatron	\$0.80	1	\$0.80
8.	JST connector	P1	Cable for LEDs	Creatron	\$1.18	1	\$1.18
9.	PCB	N/A	PCB	Oshpark	\$7.45	1	\$7.45
10..	Mylar sheet	N/A	Diffusion panel	TBD	\$5.00	1	\$5.00
11.	4" x 6"	N/A	Picture Frame	Dollarama	\$2.00	1	\$2.00
Subtotal							\$39.48
Total with HST							\$44.61

Datasheets

ESP-12F: <https://www.elecrow.com/download/ESP-12F.pdf>

WS2812B: <https://cdn-shop.adafruit.com/datasheets/WS2812B.pdf>

LD1117V33: <https://www.sparkfun.com/datasheets/Components/LD1117V33.pdf>

Contact Information

Eric Brauer

647-235-6609

eric.m.brauer@gmail.com

On the CD:

The CD includes the following:

- This file (docx, pdf)
- Project Proposal (docx, pdf)
- Presentation slides (pptx, pdf)
- Gantt chart (mpp)
- Schematic (pdf)
- Source code (main.cpp)
- Platformio.ini (contains links to third-party libraries)
- Datasheets for ESP-12f and WS2812b
- Project folder (includes all the above)

Files will also be available from:

<https://github.com/ericbrauer/TPJ>