ECE 445 Senior Design Laboratory

Design Document

Auto-Adjusted Smart Desk Lamp for Healthy Lighting

Team 15

TEAM MEMBERS:
Howard Li [zl114]
Jihyun Seo [jihyun4]
Kevin Chen [kdchen2]

TA
Zhuoer Zhang

Table of Contents

1. Introduction

1.1 Problem

Prolonged desk work under poor or inconsistent lighting can cause eye strain, headaches, and fatigue. Most desk lamps today are static; they require manual adjustment and do not adapt to changes in daylight or different tasks. This often leaves users with lighting that is either too dim or too bright for their task.

Research shows that digital eye strain is a growing issue, especially for people spending hours on screens. Poor lighting around the desk only makes this problem worse, leading to discomfort and reduced productivity[1]. A desk lamp that can automatically adjust brightness and color temperature would help create a healthier and more comfortable environment for studying, gaming, or working.

1.2 Solution

We plan to build a smart desk lamp that adjusts itself based on the lighting around the desk. The lamp will use two sensors: one to measure overall brightness and one to measure color temperature. These readings will go into a microcontroller (ESP32), which processes the data and decides how the lamp should respond.

The light itself will come from two LED sources, one warm white (2700–3000K) and one cool white (6000–6500K). By mixing these two channels with PWM control, we can cover a wide range of color temperatures, from warm evening light to cooler daylight. A pair of LED drivers will supply a stable current to the LEDs.

The ESP32 will run the control logic. It will filter out noise from the sensors, calculate the target brightness and color temperature, and then adjust the LEDs gradually. Gamma correction will make sure the brightness changes look smooth to the eye, and a rate limiter will prevent sudden jumps.

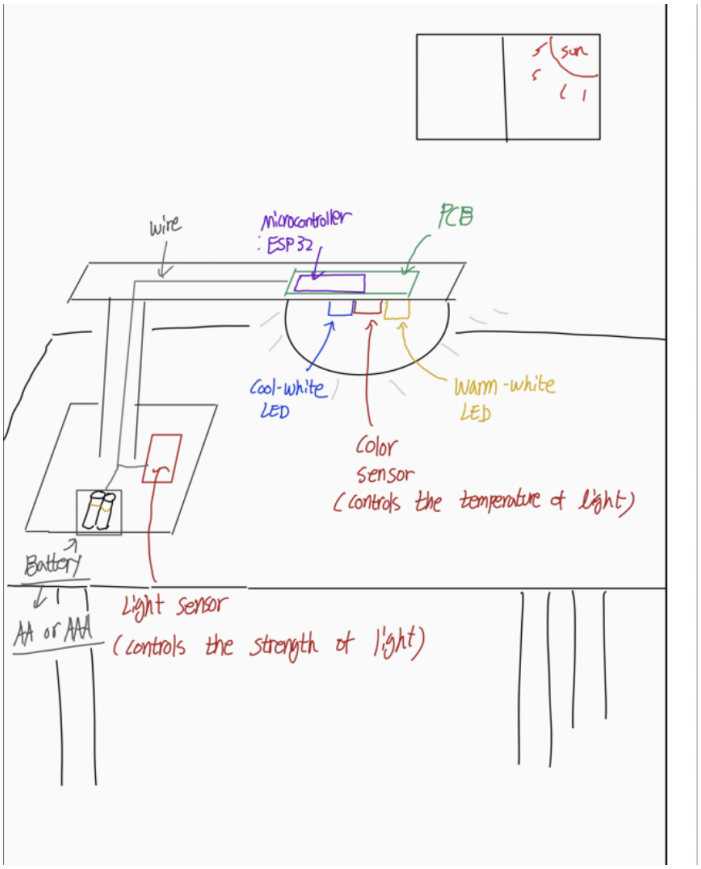
For the user, we plan to include a simple button to switch between modes like Study, Relax, or Gaming. Each mode will set a different brightness and color target, but the lamp will still adapt to the room lighting in real time. An optional knob can allow manual override if someone wants full control.

This design keeps the system straightforward: sensors read the environment, the microcontroller decides the output, and the LEDs adjust to maintain healthy and comfortable lighting while saving energy when daylight is available.

1.3 Visual Aid

The smart desk lamp will sit on a normal study desk and face the work area where a person reads, studies, or uses a computer. A light sensor near the lamp will measure the brightness on the desk, and a color sensor will check the tone of the surrounding light. The lamp will then adjust its own output to keep the desk at the right brightness and color temperature.

When there is strong daylight from a window, the lamp will dim automatically to save energy. At night or in darker conditions, it will brighten and shift to a warmer tone for comfort. A simple button on the lamp lets the user switch between modes like Study, Relax, or Gaming. An optional knob gives full manual control if needed.



1.4 High-level requirements list

-The lamp must keep the desk surface within ±10% of the target brightness level, with about 500 lux in Study Mode and 300 lux in Relax Mode, even when the surrounding light changes.

-The lamp must adjust brightness and color temperature gradually, with changes limited to no more than 2% per second, so the user does not notice sudden jumps or flicker.

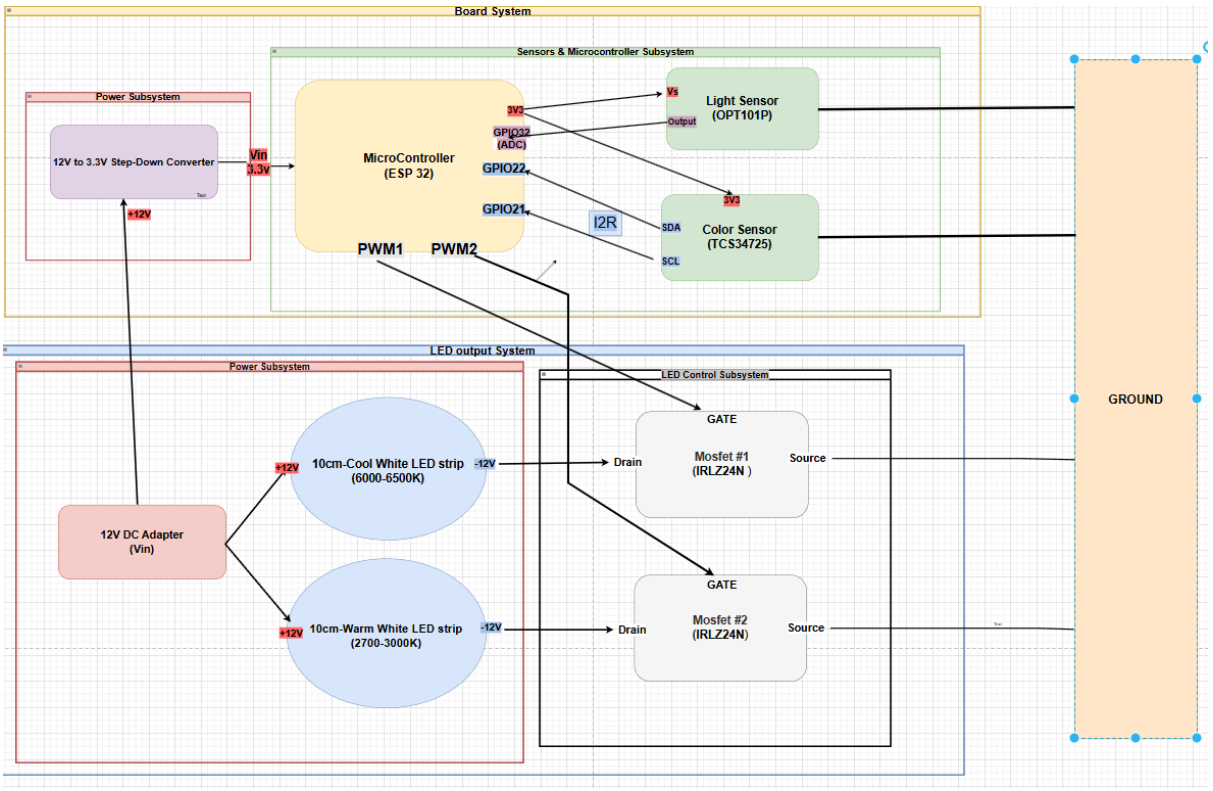
-The lamp must lower its power use by at least 20% compared to full brightness when there is enough daylight in the room.

2. Design

2.1 Physical Design

The physical design of our lamp follows a simple and modern structure. A long LED strip is mounted along the top arm to provide wide and even light coverage across the desk. The base of the lamp houses all electronic components, including the ESP32 controller, sensors, power circuitry, and wiring, all neatly enclosed to maintain a clean appearance. The sensors are positioned on the base to accurately measure surrounding light conditions without being visually noticeable. This layout keeps the lamp compact and user-friendly while supporting automatic brightness and color adjustments.

2.2 Block Diagram



2.3 Subsystem Overview and Requirements

2.3.1 Board System

The first major subsystem is the board system, which includes the low-power electronics. A 3.3V supply, from the voltage regulator LM723cn, which steps down from 12V, provides input to the ESP32 microcontroller. The ESP32 is the control unit of the lamp and runs all the logic. Two sensors are connected to it: the TCS34725 color sensor through the I²C bus (GPIO22 as SCL and GPIO21 as SDA), and the OPT101P light sensor, which outputs an analog signal to GPIO32 (ADC). Both sensors are powered from the ESP32's 3.3V output. This setup lets the ESP32 read both brightness and color information from the desk environment and compute how the lamp should respond.

2.3.2 LED Output Control System

The second major subsystem is the LED output system, which is powered separately by a 12V DC adapter. This 12V supply drives two LED strips: a cool white strip (6000–6500K) and a warm white strip (2700–3000K). Each LED strip is connected through a MOSFET(IRLZ24). The drains of the MOSFETs connect to the LED strips, the sources are tied to ground, and the gates are driven by PWM signals (PWM1 and PWM2) from the ESP32. By adjusting the PWM duty cycles, the ESP32 can control both brightness and color temperature smoothly, blending the two strips to reach the desired lighting conditions.

Together, these two subsystems make the smart desk lamp. The board system handles sensing and decision-making, while the LED output system produces the actual light. With real-time feedback from the sensors, the ESP32 can adjust the lamp's output to maintain comfortable brightness and color temperature, giving the user a stable and adaptive lighting experience.

2.3.3 Subsystem Requirements

1. Power Subsystem (12V and voltage step down): Must supply 3.3 ±0.3V at ≥500 mA to power the ESP32 and sensors. If the voltage drops below this range, the microcontroller or sensors could fail.
2. Light Sensor (OPT101P): Must measure ambient brightness from 0–65,000 lux with ±5% accuracy. This data is needed to keep desk lighting within ±10% of the target lux level.
3. Color Sensor (TCS-34725): Must detect correlated color temperature (CCT) in the range of 2700K–6500K with ±10% accuracy. This allows the system to adjust the LED mix correctly.
4. Power Subsystem (12V DC Adapter): Must supply 12V ±0.2V at ≥1A to the two LED drivers. If the voltage drops, the LEDs will not reach full brightness.
5. LEDs (Strips of Warm white and white): Must output enough light to reach at least 1000 lux at 30 cm from the desk surface. By mixing the two, the lamp must cover the full color temperature range from 2700K–6500K.

2.4 Hardware Design

2.4.1 Operation Voltage and regulation

Our system operates with two voltage levels: 12 V and 3.3 V. The 12 V DC input serves as the main power source for the LED strips, which require higher voltage to maintain stable brightness and color temperature. To power the ESP32 microcontroller and sensors, we step down the 12 V supply to 3.3 V using a LM723CN voltage regulator circuit.

6.2 Recommended Operating Conditions

Table 6-2. Recommended Operating Conditions

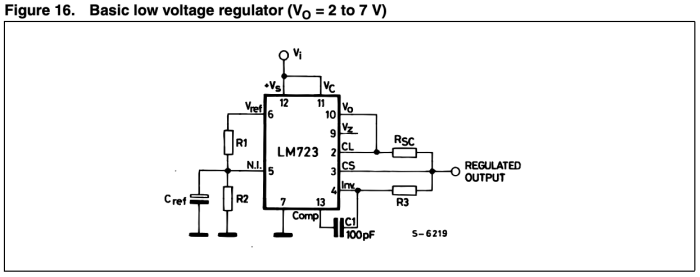
Symbol	Parameter		Min	Typ	Max	Unit
VDD33	Power supply voltage		3.0	3.3	3.6	V
I_{VDD}	Current delivered by external power supply		0.5	—	—	A
T_A	Operating ambient temperature	65 °C version	−40	—	65	°C
		85 °C version			85	
		105 °C version			105	

According to esp32 datasheet we need 3.0-3.6V to power esp32

The LM723 provides stable low-voltage output with good line and load regulation, ensuring consistent operation of the ESP32 and optical sensors. In our design, resistors R1, R2, and

R3 set the output voltage, and we tuned these values to achieve a measured output of approximately 3.1 V–3.3 V, which is within the acceptable range for reliable ESP32 logic levels. A filter capacitor at the output helps smooth transient variations and maintain voltage stability during sudden current changes from the microcontroller.

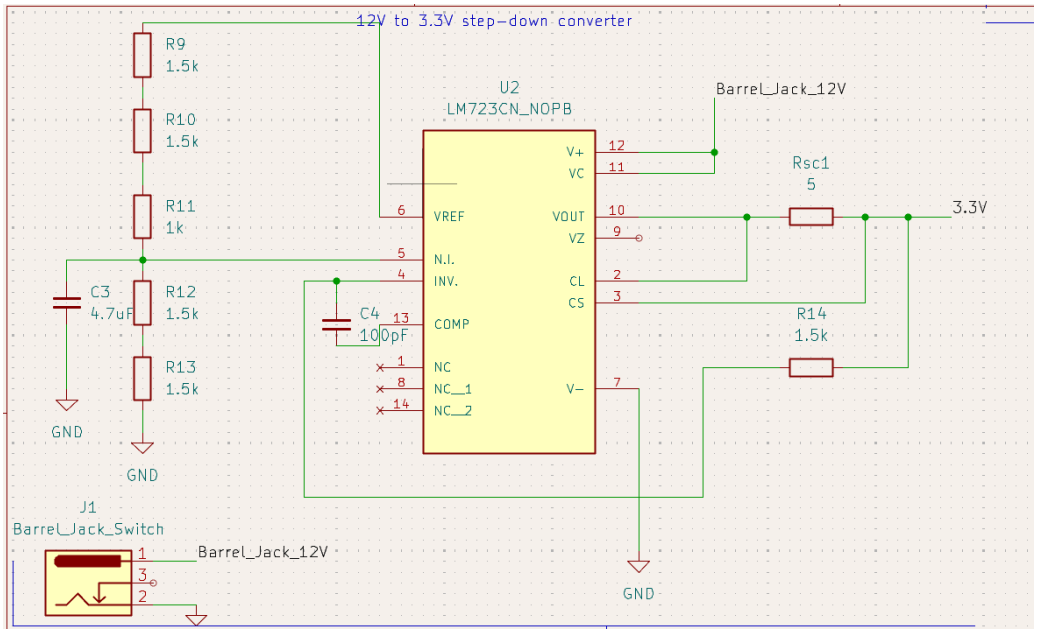
Figure 16. Basic low voltage regulator (V_O = 2 to 7 V)



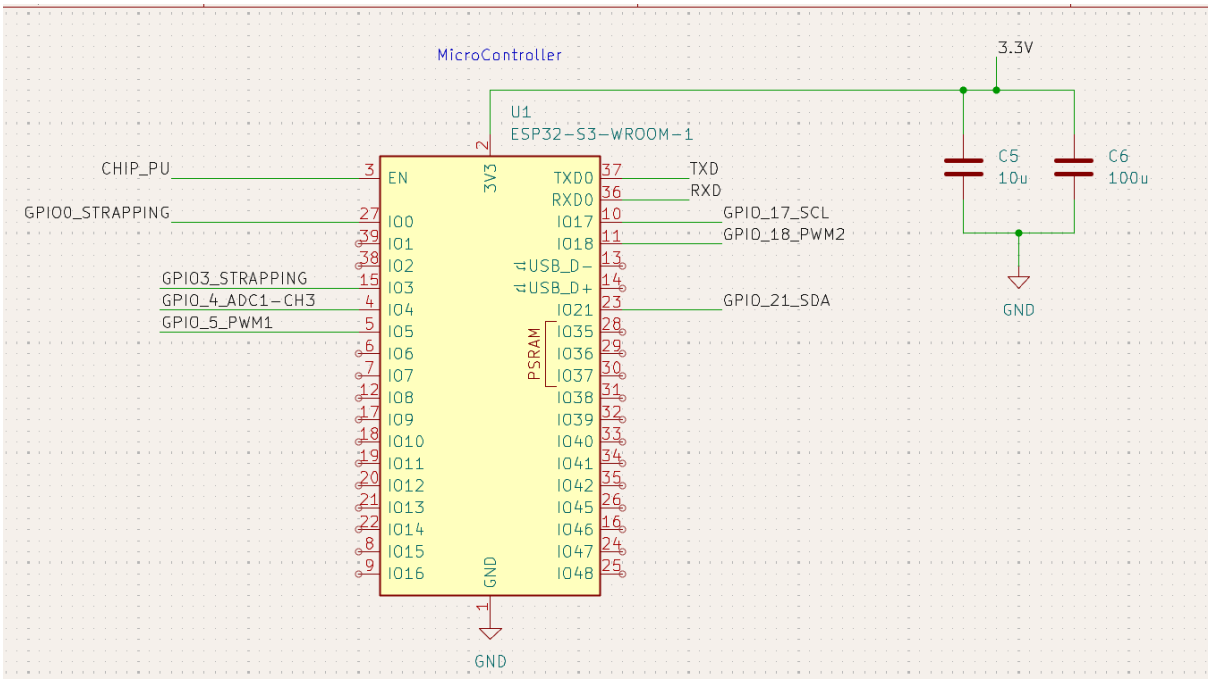
Note: *$R_3 = (R_1 x R_2)/(R_1+R_2)$ for minimum temperature drift.*
R_3 may be eliminated for minimum component count.
Typical performance
Regulated output voltage.....5 V
Line regulation (ΔV_I = 3 V).....0.5 mV
Load regulation (ΔI_O = 50 mA)...1.5 mV

According to LM723 data sheet we designed the voltage stepdown

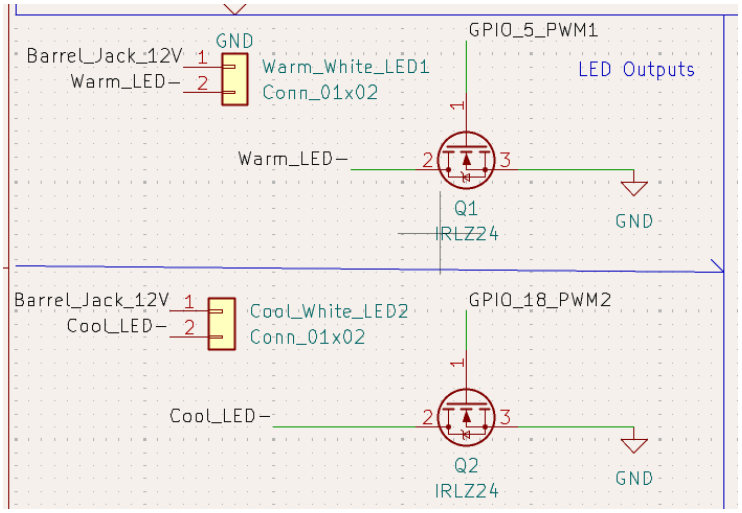
This dual-voltage setup allows the system to efficiently power both the high-power LED strips and the low-power control circuitry from a single 12 V source, simplifying wiring and ensuring overall system reliability.



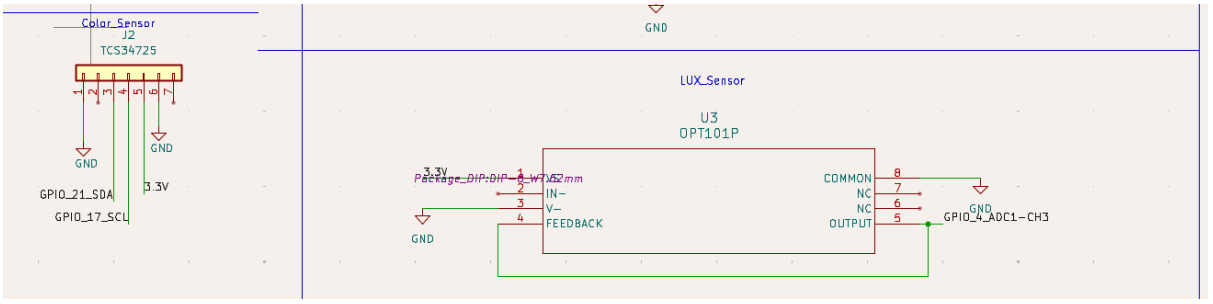
2.4.2 MicroController



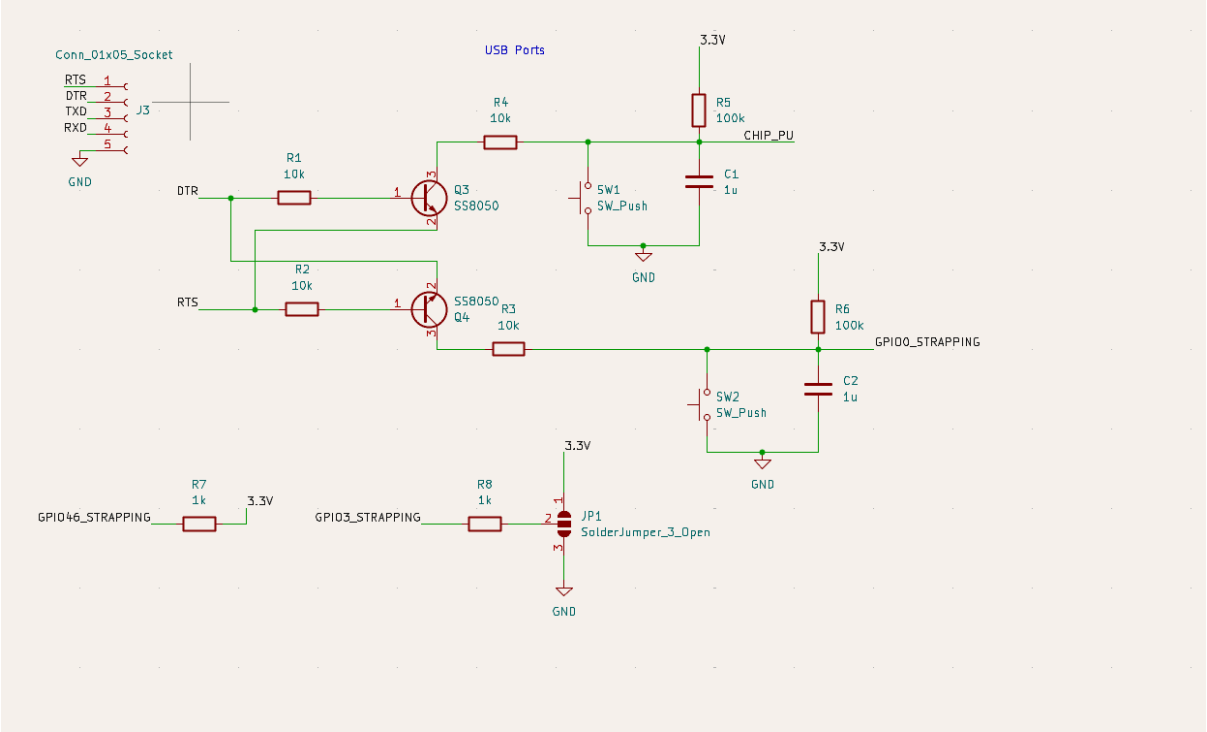
2.4.3 LED Outputs



2.4.4 Sensors



2.4.5 USB and Download



2.5 Software Design

Software is responsible for tying together sensor data and light output via PWM. To meet the design goals of the project the software must meet some basic requirements:

1. Ignore abrupt changes from sensor readings (oversensitivity)
2. Gamma correct PWM output
3. Adjust lighting in response to ambient lux and color levels smoothly

To address requirement (1), we will use a moving-average-like filtering algorithm when processing sensor data.

We will start with storing initial filler values in an array.

```
const int sample_sz = 15;
int lux_adc[sample_sz] =
{250,250,250,250,250,250,250,250,250,250,250,250,250,250,250};
```

During the operation of the lamp we will first take an average of all the values in the array using the following function.

```
float get_avg(int input_arr[], int length){
  long sum = 0L;
  for (byte i = 0; i < length; i++) {
```

```
    sum += input_arr[i];
  }
  return ((float)sum)/length;
}
```

Then when processing sensor readings, for each reading, we will take a percent (tolerance) of the average of our array and create a low and high value by subtracting or adding the tolerance. We then compare the raw sensor reading to low and high values: if it is higher than the high, then the high will be shifted into the sensor data array; if it is lower than the low, then the low will be shifted into the sensor data array. This is done in the follow functions:

```
float filter_val(int num, float avg, float tolerance){
  float rtn_val = (float)num;
  float increment = tolerance*avg;
  float high = avg + increment;
  float low = avg - increment;
  if (num > high){
    rtn_val = high;
  }
  else if (num < low){
    rtn_val = low;
  }

  return rtn_val;
}
```

```
void update_arr(int input_arr[], int length, int new_val) {
  for (byte i = 0; i < length-1; i++) {
    input_arr[i + 1] = input_arr[i];
  }
  if (new_val < 100){
    new_val = 100;
  }
  input_arr[0] = new_val;
}
```

From the algorithm it is clear that the average of the sensor data array will change very gradually and is resistant to sudden changes in sensor readings. The average will be used to calculate the PWM level during the operation of the lamp to ensure smooth lighting changes.

An additional benefit of this data filtering algorithm is that it addresses requirement (2) for gamma correction. Human vision has a logarithmic curve in that it is more sensitive to light

changes in darker settings than brighter settings. In this data filtering algorithm, as the average value in the data array increases, so does the actual size of the tolerance increment (average * tolerance). This results in larger changes as total environmental brightness increases[7].

Finally to address requirement (3) for smooth lighting changes in response to ambient lux and color levels, our design will use two independent PWM signals to drive a warm and a cool LED strip.

We will use an analog sensor for lux, which will output a value from 0 to 4095 on one of the ESP32's ADC pins and the adafruit tcs34725 color sensor to acquire RGB values via I2C through adafruit's libraries. The lux sensor data will be fed directly into the filtering algorithm while the color sensor data will be converted to a temperature value between 0 and 4095 before being fed into the filtering algorithm, 0 being cool and 4095 being warm.

```
float get_temp(int r, int g, int b){
  float temp = 2048;
  float offset = 2047*(float)(r-b)/(r+b+1);
  return temp + offset;
}
```

The averages of the lux and temperature data arrays will then be fed into the function below as sensor_val.

```
float adjust(float target_output, float sensor_val, float curr_output, float
tolerance){
  float output_inc = 0;
  float increment = target_output*tolerance;
  float adj_increment = sensor_val*tolerance;
  if ((sensor_val < target_output-increment) && (curr_output += adj_increment
< 4096)){
    output_inc = adj_increment;
  }
  else if ((sensor_val > target_output+increment) && (curr_output -=
adj_increment >= 0)){
    output_inc = -adj_increment;
  }

  return output_inc;
}
```

The adjust function will return a positive or negative increment that will be added to the PWM output value for each LED strip. This will make sure the sensor value falls within some tolerance of a target value to prevent oscillation and also that the adjustment won't overshoot the allowable limits for a PWM output.

Each LED PWM value will be adjusted like such:

```
led_out0 = led_out0 + adjust(...) + adjust(...);
```

One adjust term for overall lux and another for color temperature. The cool LED's temperature adjust will need (4095-target_temp) for its target_output parameter.

2.6 Tolerance Analysis

Critical Feature:
The most critical performance requirement in our design is that the desk lamp dims and brightens smoothly, without visible flicker or discrete steps. This is achieved by controlling the LED brightness through the PWM output of the ESP32. The PWM duty cycle determines the average current through the LEDs, so precise and stable control is essential to maintain comfortable lighting.

Feasibility and Quantitative Analysis:
The ESP32-S3 PWM generator (LEDC) operates up to a 40 MHz base clock with 16-bit resolution. At our chosen PWM frequency of 1 kHz (period = 1 ms), we retain full 16-bit precision, giving 65,536 steps of duty control. Each step corresponds to roughly 0.0015 % of the full scale. Even if the internal oscillator drifts by ±0.1 %, the resulting duty error would be <0.002 %, far below the human eye's perception threshold (~1 %).

The voltage regulator section provides 3.1 V ± 0.1 V to the ESP32, measured on the oscilloscope. This variation causes a negligible effect on logic-high PWM levels (≥ 2.6 V required per datasheet). MOSFET switching delay (≈ 20–30 ns) is also insignificant compared to the 1 ms PWM period. Hence, both frequency stability and logic level margin ensure consistent LED brightness.

Component Tolerances and Expected Effects:

Parameter	Typical Tolerance	Effect on Output
ESP32 PWM clock drift	±0.1 %	±0.1 % brightness error
Supply voltage (3.3 V rail)	±0.3 V	No change in PWM logic
LED forward voltage	±0.2 V	Minor color shift (< 1 %)
Resistor tolerance	±1 %	Negligible current change
Sensor noise	±2 % lux	Filtered by EMA algorithm

Even under the combined worst-case variation, the total perceived brightness change remains below 10%, well within the human comfort limit. The firmware also caps the brightness change rate at ≤ 10% per second, ensuring gradual transitions.

Non-Idealities and Compensation:
Because human brightness perception is nonlinear, the control algorithm applies a gamma-correction curve so equal PWM steps appear visually uniform. A moving-average filter on the OPT101 light sensor smooths transient noise. Rate limiting prevents the feedback loop from overshooting or oscillating under rapidly changing ambient light.

Verification and Test Plan:
Step Test: Command 1 %, 2 %, 5 % duty changes; record LED current to verify smooth analog transitions.

Ramp Test: Sweep 50 → 75 % duty over 5–10 s and measure lux; expect monotonic response without jumps.

Flicker Test: Use a phone's high-speed camera to confirm no visible flicker at 1 kHz.

Closed-Loop Test: Vary ambient light; confirm output tracks within ±10 % lux of target and transitions remain gradual.

Conclusion:
This analysis shows that the PWM system's timing, voltage stability, and component tolerances all stay well within limits required for smooth dimming. Even accounting for ±10 % hardware variation, the lamp maintains stable, flicker-free light output and fulfills the project's visual-comfort goal.

2.7 Cost Analysis

Description and link	Manufacturer	Quantity	Price (total)
ESP32 development board	ELEGOO	3	16.99
OPT1010P	Texas Instruments	1	11.16
CP2102 USB 2.0 to TTL	ATNSINC	1	9.99
COB LED Strip warm	ADLEDQYGD	1	16.99
COB LED Strip cold	ADLEDQYGD	1	16.99
TCS35725	Tetleten	3	12.88
IRLZ24N	ALLECIN	10	8.99

The total price of components adding the 9% sales tax is $102.45. We expect $30/hr salary for each teammates, we work 10hr per week per person, the labor cost would be $ 11700. Thus the total cost would be $118102.45.

2.8 Schedule

Week	Task
September 15 – September 22	Team meetings with TAs. Submit project proposal and team contract. Start planning the schematic and power design. Research ESP32 pinout and LED control interface. Ordered all the components.
September 22 – September 29	Pass proposal review. Build the first breadboard prototype for the color sensor and LED control. Test software for ESP32 communication and LED PWM. Verify 12 V → 3.3 V voltage step-down circuit. Tune control logic for smooth PWM transitions.
September 29 – October 6	Complete breadboard demonstration. Finish the voltage regulator and light control circuit. Connect the ESP32 to the sensors and verify stable control of brightness and color. Begin schematic capture and PCB layout design for the final board. Prepare design document for submission.
October 6 – October 13	Receive the OPT101 light sensor and begin testing the sensor response. Integrate OPT101 light sensor readings with the ESP32 code. Review PCB design with TA. Submit the first round PCB order.
October 13 – October 20	Submit design document. Order components for PCB and start testing the 3D printed lamp housing concept. Implement firmware for the weighted sensor control algorithm.
October 20 – October 27	Prepare the Second breadboard demo and final verification of the control loop. Finalize PCB layout and pass audit for first PCB order. Start assembling a partial prototype. Test the PCB from the first round.
October 27 – November 3	Debug any issues with the ESP32 and LED control. Modify the breadboard and PCB design. Breadboard Demo2.
November 3 – November 10	Order PCB in the third round with the complete design. Finish the individual progress reports.
November 10 – November 17	Integrate sensors and LED circuits on the PCB. Validate real-time adjustment algorithm using both color and brightness data. Prepare for the mock demo.
November 17 – November 24	Conduct a mock demo with the TA. Fix circuit issues. Begin assembling the full lamp housing and wiring.
November 24 – December 1	During Fall break, prepare for the Final demo.
December 1 – December 8	Present final demo and submit presentation slides, report, and lab notebook. Finalizing final paper.

December 8-11	Final presentation, submit the final paper, and do the lab check out.

3. Ethics and Safety

3.1 Ethical Analysis

We want this lamp to help people, not create new problems. We'll follow the IEEE and ACM codes as practical guardrails and keep our choices simple and transparent.

1. Safety first, honest claims. IEEE I-1 and I-5 ask us to protect public safety and be honest about limits, to seek/accept criticism, and credit others [2]. We will publish measured lux/CCT ranges, flicker limits, and power data from repeatable tests. If we find errors, we'll correct them in our docs and code commits before the demo.

2. Avoid harm. ACM 1.2 and 1.1 say to minimize negative consequences and support well-being [3]. We'll cap maximum output and blue-heavy settings (upper CCT limit ≈6500 K), ship warm defaults at night, and keep flicker outside risky bands (see "Standards" below).

3. Be fair and respectful. IEEE II-7/8/9 and ACM 1.4 require respect and non-discrimination [2][3]. Team roles, code reviews, and decisions are shared; feedback is invited in stand-ups so everyone is heard.

4. Competence and review. IEEE I-6 and ACM 2.6/2.4 call for working within competence and using peer review [2][3]. Power and thermal design will be led by the teammate with the most experience; all safety-critical PRs require a second reviewer before merge.

5. Privacy by design. IEEE I-1 (privacy) and ACM 1.6/1.7 emphasize respecting privacy [2][3]. Our lamp does not log or transmit personal data. If we later add BLE for settings, it will be local-only and opt-in; no cloud.

3.2 Safety Considerations

1. Flicker: follow IEEE Std 1789-2015 recommendations; use high-frequency PWM (≥1 kHz) or DC dimming and limit modulation depth at low frequencies to avoid headache/eye-strain risks [4].

2. Light exposure: check against IEC 62471 (photobiological safety of lamps/LEDs). Our power level and diffusers keep us in exempt/low-risk categories; we will verify during testing [5].

3. Lab Safety: We will follow the University of Illinois lab safety guide[6]: Never work alone, at least two people present; Complete the online safety training and submit the certificate before lab work; Extra training if any high voltage is involved; Follow battery and current-through-body guidelines exactly (we are not sending current through a person).

4. Thermal: heatsink the LED board, add NTC-based derating, and shut down on over-temp.

5. Batteries (if we go portable later). We will default to bench supplies for development. If a battery is introduced for demos, we will follow the ECE 445 battery safety rules: avoid batteries unless needed, cover terminals, use protection circuits, charge only in approved bags, and get TA approval before any test. Emergency steps (swelling/heat/noise → isolate, notify TA) will be posted at the bench.

4. References and Datasheets:

4.1 References

[1]K. Kaur et al., “Digital Eye Strain- A Comprehensive Review,” Ophthalmology and Therapy, vol. 11, no. 5, pp. 1655–1680, Jul. 2022, doi: https://doi.org/10.1007/s40123-022-00540-9.

[2]IEEE, “IEEE Code of Ethics | IEEE,” Ieee.org. https://www.ieee.org/about/corporate/governance/p7-8 (accessed Sep. 18, 2025).

[3]Association for Computing Machinery, “ACM Code of Ethics and Professional Conduct,” Association for Computing Machinery, Jun. 22, 2018. https://www.acm.org/code-of-ethics (accessed Sep. 18, 2025).

[4]“IEEE Standards Association,” IEEE Standards Association. https://standards.ieee.org/ieee/1789/4479/ (accessed Sep. 19, 2025).

[5]“IEC 62471-7:2023,” Webstore.iec.ch, 2023. https://webstore.iec.ch/en/publication/68810 (accessed Sep. 18, 2025).

[6]University of Illinois Urbana-Champaign, “Laboratory Safety Guide.” Accessed: Sep. 18, 2025. [Online]. Available: https://drs.illinois.edu/site-documents/LaboratorySafetyGuide.pdf

[7]“The problem with driving LEDs with PWM,” codeinsecurity, Jul. 17, 2023. https://codeinsecurity.wordpress.com/2023/07/17/the-problem-with-driving-leds-with-pwm/

4.2 Datasheets

OPT101P
Information:
https://www.digikey.com/en/products/detail/texas-instruments/OPT101P/251177?gclsrc=aw.ds&gad_source=1&gad_campaignid=120565755&gbraid=0AAAADrbLliWQ1ouMUvblb-SaVDuzeTuJ&gclid=Cj0KCQjw_rPGBhCbARIsABjq9ccN2LmBRx2S3Xc2wu6SPwWX1pKaTdl6Dy6j1Usr1ENnvJ4gfj3UqjgaAgfGEALw_wcB
Datasheet:
https://www.ti.com/lit/ds/symlink/opt101.pdf?HQS=dis-dk-null-digikeymode-dsf-pf-null-wwe&ts=1758304012910&ref_url=https%253A%252F%252Fwww.ti.com%252Fgeneral%252Fdocs%252Fsuppproductinfo.tsp%253FdistId%253D10%2526gotoUrl%253Dhttps%253A%252F%252Fwww.ti.com%252Flit%252Fgpn%252Fopt101

LM723CN
Information:
https://www.mouser.com/ProductDetail/Texas-Instruments/LM723CN?qs=QbsRYf82W3GG7X%252BEiVmMbg%3D%3D&utm_id=22380207736&utm_source=google&utm_medium=cpc&utm_marketing_tactic=amercorp&gad_source=1&gad_campaignid=22376567996&gbraid=0AAAADn_wf0OaRlikXwe1CbzSbgCnzBeO&gclid=Cj0KCQjwovPGBhDxARIsAFhgkwTGbfRDWAdjY6NRQiLcC6UaQ3CWk8Ev1ajGGHmqhBUiomtkcB7iE1oaAtbIEALw_wcB
Datasheet:
https://www.mpja.com/download/lm723.pdf

TCS-34725
Information:
https://www.amazon.com/Teyleten-Robot-TCS-34725-TCS34725-Recognition/dp/B087Z3K6P5/ref=sr_1_1_sspa?dib=eyJ2IjoiMSJ9.XPYcZF8Ig8Nnz-kBMROzM9CXqq8_9xSdNBImpSFcXNiArBzxLe4SkRmNtmWSfeTsLHJYDfXTeYGg2PkzhCn9ZEv8UnZTCU8noaNOALHAMFBW7QSUB2_Av6dUMwM5SZBm26Ez6EsZ7CsKBAdKjDT-ZjWbatOeAGxDSlryIVlKJXtmONQLFa4SVnLqoHbBt2Dg56NpXGToUkfcEjYOvoGLUc2_s7XXq_wtqAXR7ctMo1c.DKjlTpdsINJ1AtBmpRp08JFZuJWpg5GSMJowTVQA2Vw&dib_tag=se&keywords=tcs34725&qid=1758305817&sr=8-1-spons&sp_csd=d2lkZ2V0TmFtZT1zcF9hdGY&psc=1
Datasheet:
https://cdn-shop.adafruit.com/datasheets/TCS34725.pdf

COB LED Strip Light - 12V LED COB Strip 16.4ft/5m 320LED/M Warm White 3000K - CRI90+ 8W/M
Information:

https://www.amazon.com/gp/product/B0FDKTL2F5/ref=ox_sc_act_title_1?smid=A1KTRSYCZ04XQJ&th=1

COB LED Strip Lights 12V 8W/M 16.4ft/5m 320LED/M White 6500K CRI90+
Information:
https://www.amazon.com/gp/product/B0FDQW4MR3/ref=ox_sc_act_title_2?smid=A1KTRSYCZ04XQJ&th=1

IRLZ24N
Datasheet:
https://us.rs-online.com/m/d/82394c9f060c5a2115b97d919bd6aa96.pdf?srsltid=AfmBOorB1NXt5MtWPc7m8OVIDu7y-q-hdVszpKKrdi3QbvyRAAKQCZ4k

ESP 32
Information:
https://www.youtube.com/watch?v=lMaDJlYp29s
Datasheet:
https://cdn.sparkfun.com/datasheets/IoT/esp32_datasheet_en.pdf