ECE 445 Senior Design Laboratory

Final Paper

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..... 4
1.1 Problem..... 4
1.2 Solution..... 4
1.3 Visual Aid..... 5
1.4 High-level requirements list..... 5
2. Design..... 6
2.1 Physical Design..... 6
2.2 Block Diagram..... 7
2.3 Subsystem Overview and Requirements..... 7
2.3.1 Board System..... 7
2.3.2 LED Output Control System..... 7
2.3.3 Subsystem Requirements..... 8
2.4 Hardware Design..... 8
2.4.1 Operation Voltage and Regulation..... 8
2.4.2 MicroController..... 10
2.4.5 USB Programming..... 11
2.5 Software Design..... 11
2.6 Tolerance Analysis..... 15
3. Design Verification..... 17
3.1 Brightness Regulation Testing..... 17
3.2 Power Consumption and Efficiency..... 18
3.3 Smooth Transition and Gamma Correction..... 18
3.4 Component-Level Verification..... 19
4. Cost and Schedule..... 19
4.1 Cost Analysis..... 19
4.1.1 Part Costs..... 19
4.1.2 Labor Cost..... 20
4.1.3 Grand Total..... 20
4.2 Schedule..... 20
5. Conclusion..... 21
5.1 Accomplishments..... 21
5.2 Uncertainties and Challenges..... 21
6. Ethics and Safety..... 22
6.1 Ethical Analysis..... 22
6.2 Safety Considerations..... 23
7. References and Datasheets:..... 23
7.1 References..... 23
7.2 Datasheets..... 24

Appendix A: Requirements and Verification..... 25
A.1 Requirement and Verification Table..... 25
Appendix B: Design Details..... 28
B.1 Schematics and PCB..... 28
B.2 Software..... 29

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 creates a healthier and more comfortable environment for studying, gaming, or working.

1.2 Solution

We designed and built a smart desk lamp that automatically adjusts based on the lighting around the desk. The lamp uses two sensors: a photoresistor to measure overall brightness and a color sensor to measure correlated color temperature (CCT). These readings are processed by an ESP32-S3 microcontroller, which determines the appropriate lamp response.

The light source consists of two LED strips: one warm white (2700–3000K) and one cool white (6000–6500K). By mixing these two channels with PWM control, the system covers a wide range of color temperatures, from warm evening light to cooler daylight. A custom power subsystem supplies stable current to the LEDs.

The ESP32-S3 runs the control logic, filtering sensor noise and calculating the target brightness and color temperature. A gamma correction algorithm ensures brightness changes appear smooth to the human eye, while a rate limiter prevents sudden jumps or flickering.

For the user, we included buttons to adjust brightness and color temperature as they wish. Adjustments can be made to set custom brightness and color targets, but the lamp will still adapt to the room lighting in real time based on sensor data to meet the targets.

This design keeps the system straightforward: sensors sample the environment, then the microcontroller takes the sensor data and adjusts PWM driving the LEDs. This way healthy and comfortable lighting is maintained while saving energy when daylight is available.

1.3 Visual Aid

The smart desk lamp sits on a standard desk, facing the work area. Sensors integrated into the base measure the brightness and color tone of the surrounding environment. The lamp automatically dims when strong daylight is detected to conserve energy and shifts to a warmer, brighter tone in darker conditions to maintain user comfort.

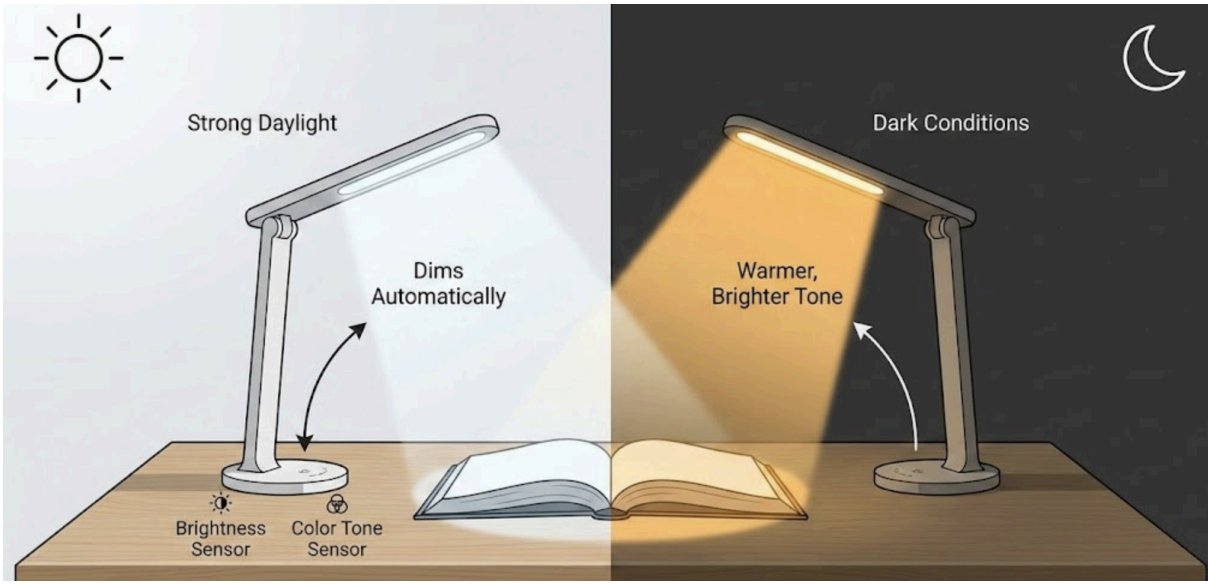


Figure 1: Visual Aid

1.4 High-level requirements list

-The lamp must keep the desk surface within ±10% of the target brightness level, with about 450lux 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

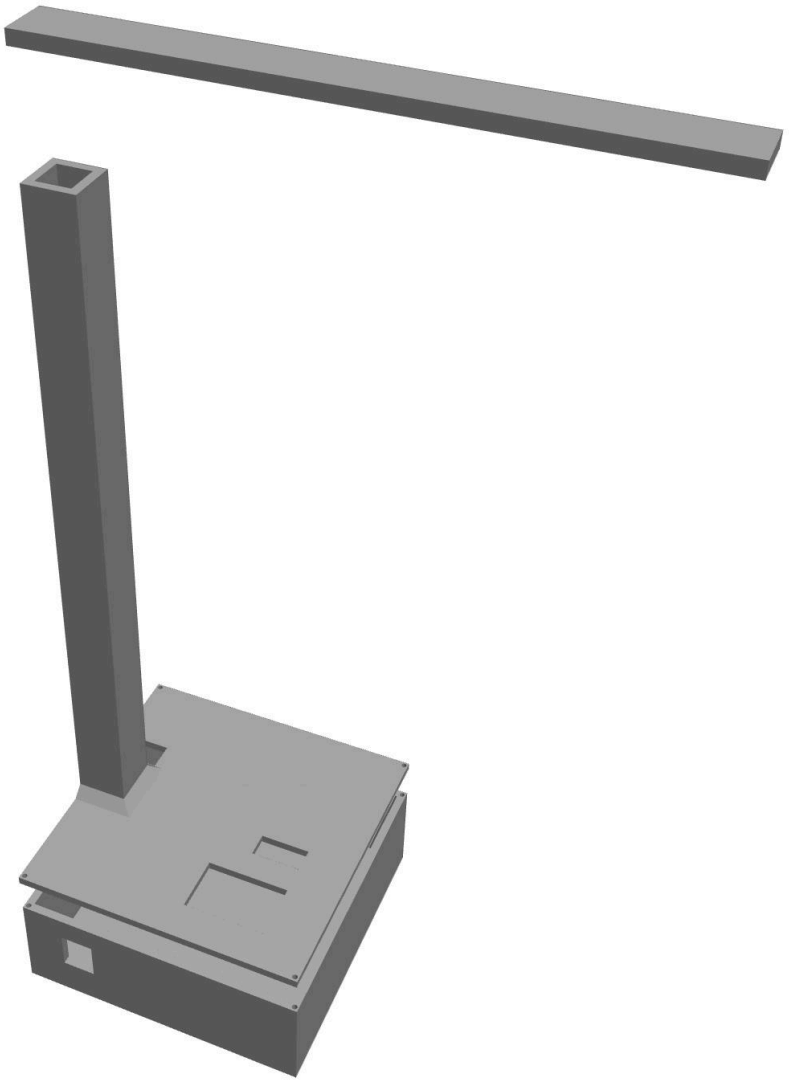


Figure 2: 3-D Printing Design

The physical design of the lamp follows a simple and modern structure. A high-density LED strip was mounted along the top arm to provide wide and even light coverage across the desk surface. The base of the lamp houses all electronic components, including the ESP32-S3 micro controller, sensors, power circuitry, and wiring, all enclosed to maintain a clean aesthetic. The sensors were positioned on the surface of the base by default for visual cleanliness, but are soldered to wires so that the user may extend and maneuver them for better performance depending on setting. This layout maintained a compact footprint while supporting the necessary hardware for automatic brightness and color adjustments.

2.2 Block Diagram

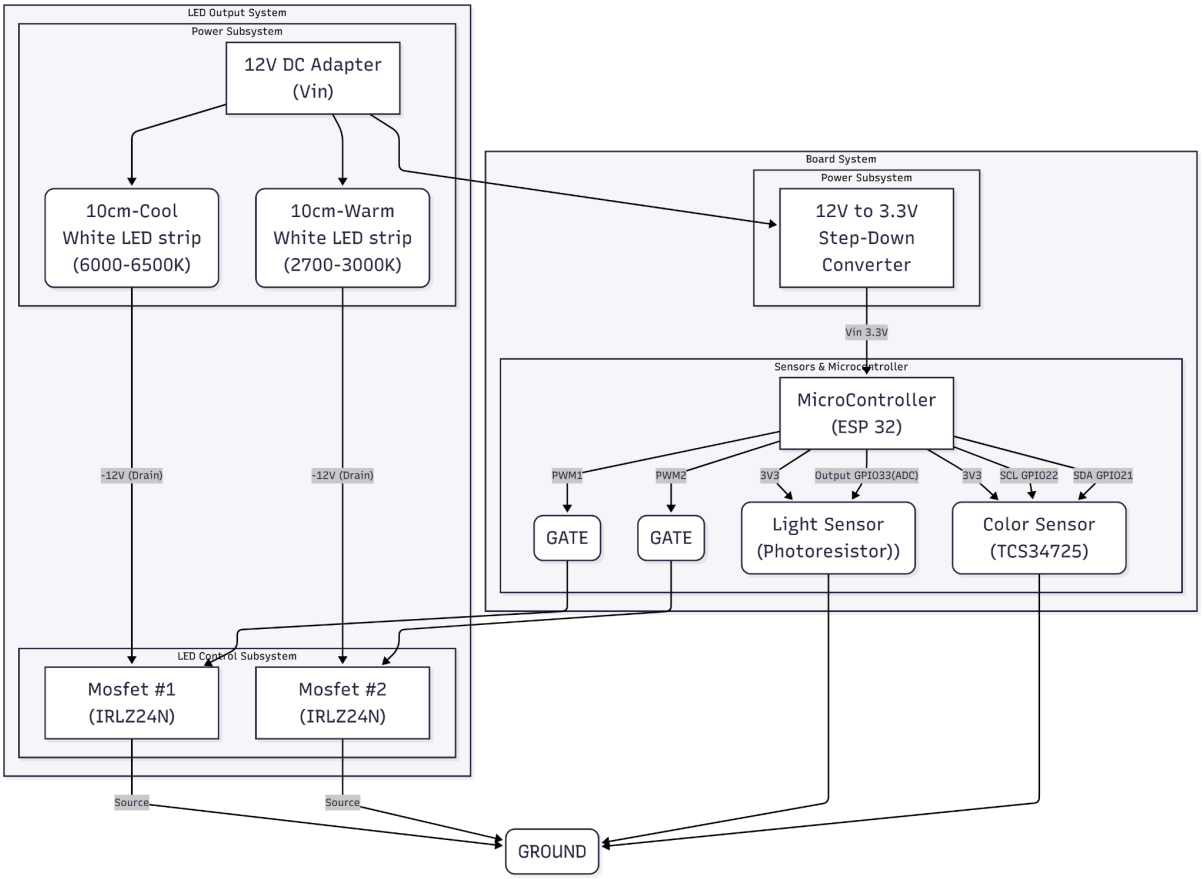


Figure 3: Block Diagram

2.3 Subsystem Overview and Requirements

2.3.1 Board System

The board system manages the low-power electronics and decision logic. A 3.3 V supply is generated from the main 12 V input using an LM723CN voltage regulator. This provides power to the ESP32-S3 microcontroller and the sensors. The ESP32-S3 serves as the central control unit. It interfaces with two sensors: a TCS34725 color sensor via the I²C bus (using GPIO22 as SCL and GPIO21 as SDA) and a photoresistor (light dependent resistor) configured as a voltage divider connected to an Analog-to-Digital Converter (ADC) pin (GPIO32). This setup allows the ESP32-S3 to read both ambient brightness and color temperature to compute the necessary lighting adjustments.

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-S3 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-S3 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-S3 and sensors. If the voltage drops below this range, the microcontroller or sensors could fail to start up, otherwise if the voltage is too high the components can burn.
2. 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.
3. 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.
4. 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 a higher voltage to maintain stable brightness and color temperature. To power the ESP32-S3 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

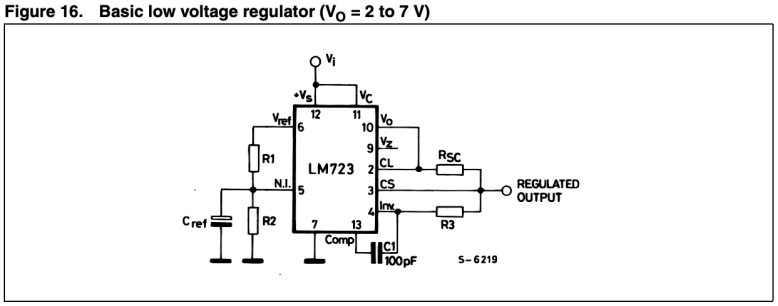
Table 6-2. Recommended Operating Conditions

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	

Figure 4:According to ESP32-S3 datasheet, we need 3.0-3.6V to power esp32

The LM723 provides a stable low-voltage output with good line and load regulation, ensuring consistent operation of the ESP32-S3 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-S3 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 (Δ V_I *= 3 V).....0.5 mV*
Load regulation (Δ I_O *= 50 mA)...1.5 mV*

Figure 5: According to the LM723 data sheet, we designed the voltage step-down

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.

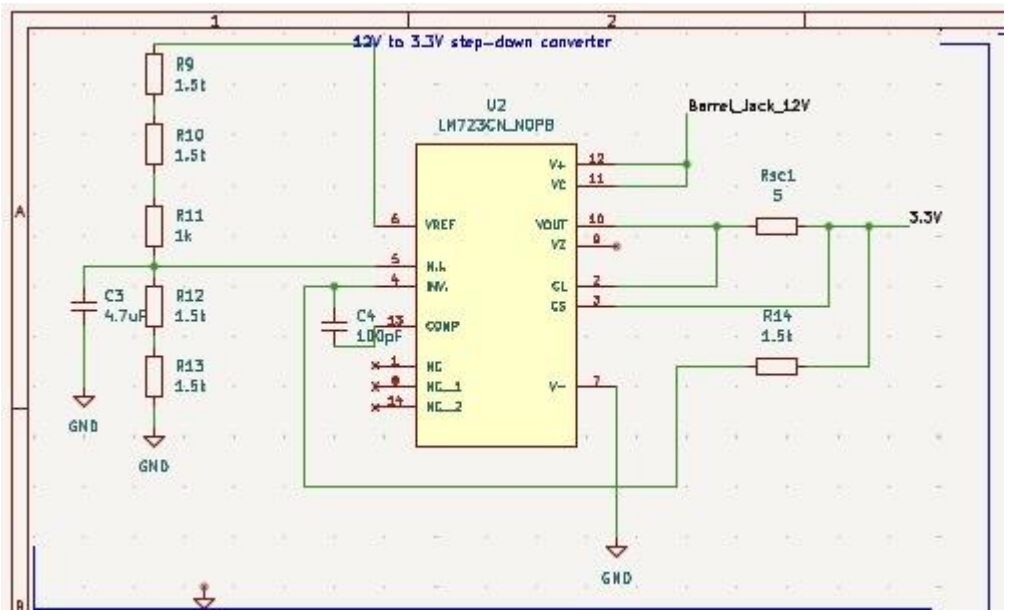


Figure 6: Schematic of voltage step down

2.4.2 MicroController

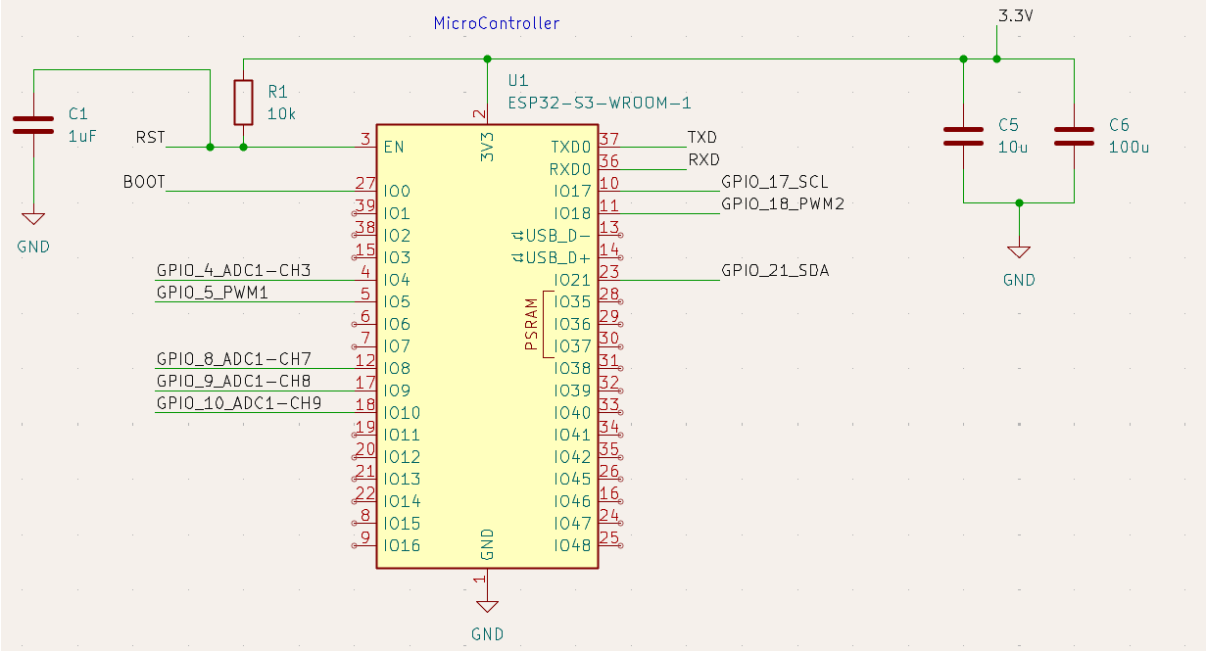


Figure 7: Schematic of MicroController

2.4.3 LED Outputs

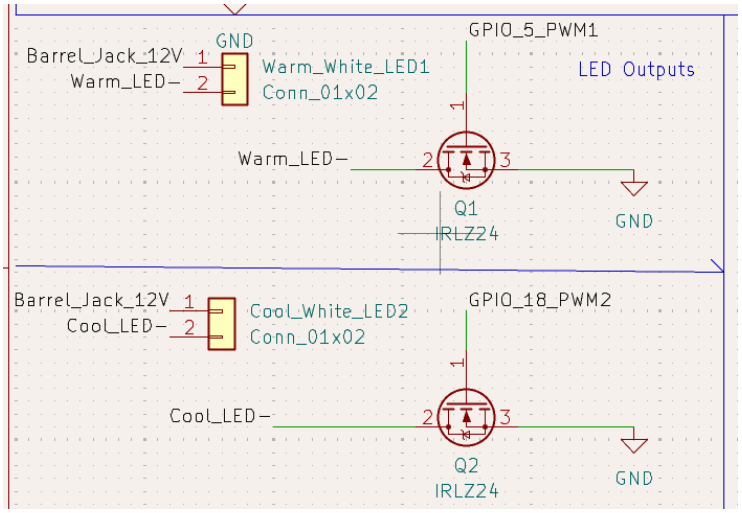
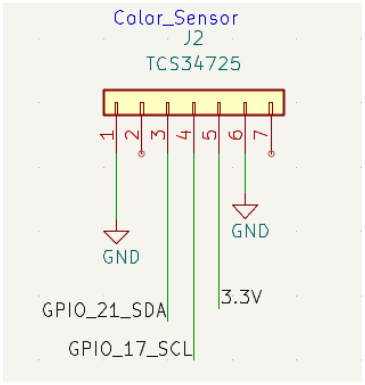
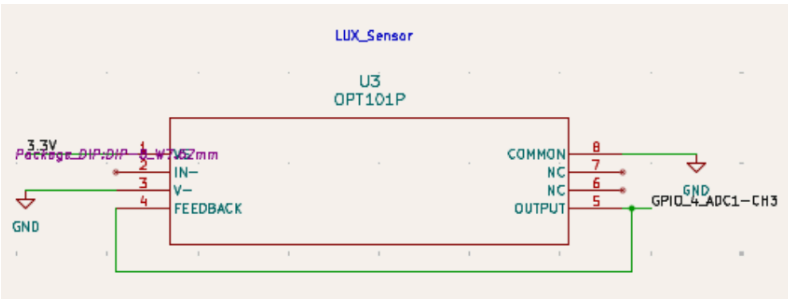


Figure 8: Schematic of LEDs

2.4.4 Sensors





Figures 9 and 10: Schematic of sensors

We kept the OPT101 through-holes on our PCB design because we could repurpose them for our photoresistor-voltage-divider lux sensor. The option to use either sensor was open to us during assembly.

2.4.5 USB Programming

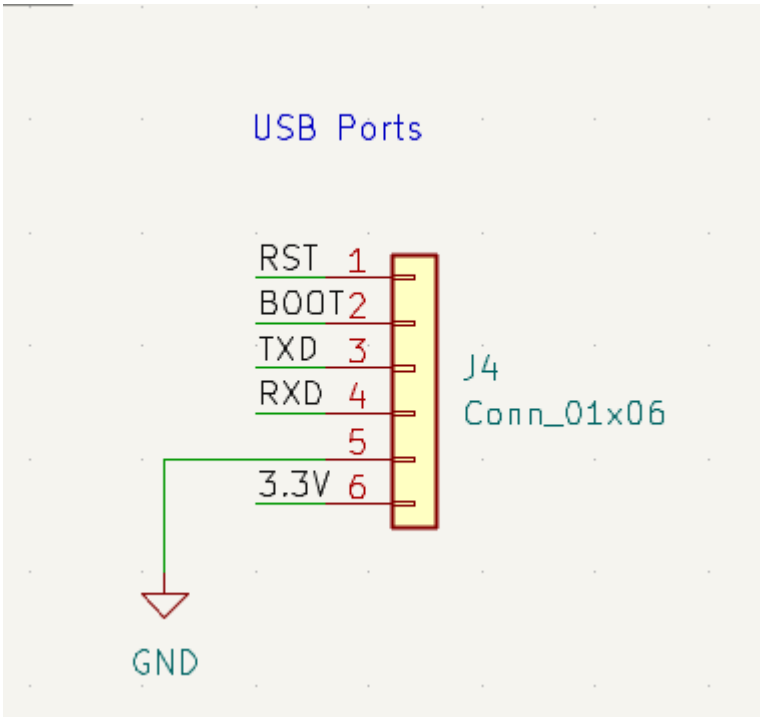


Figure 11: Schematic of Programming through holes

This was used to program out ESP32-S3 via external USB programmer (CH340K).

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). See figure 12 below. This results in larger changes as total environmental brightness increases[7].

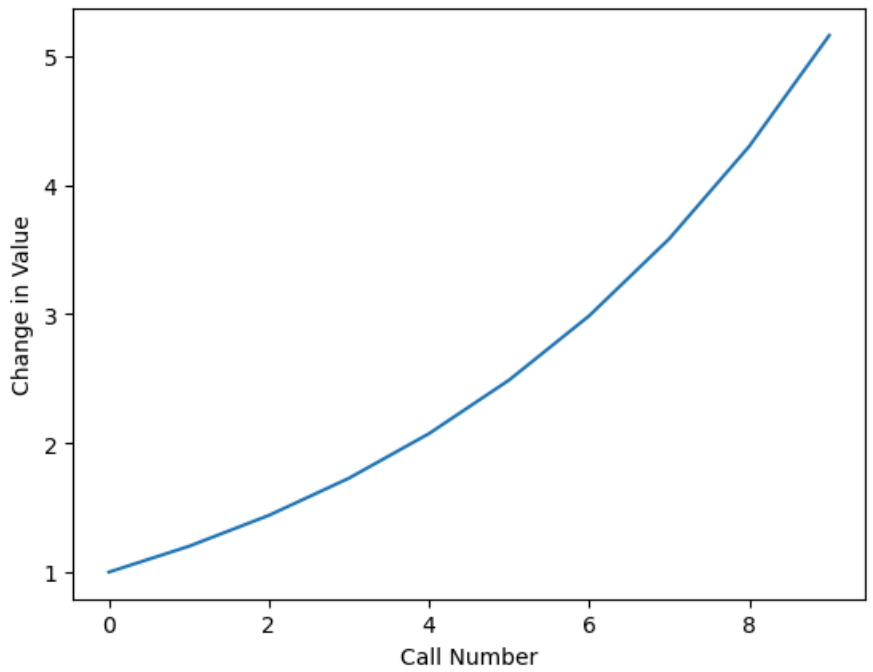


Figure 12: Filtering Algorithm being applied to an arbitrary data array
Each call takes a new value greater than the average of the data array, notice how the average of the data array changes non-linearly.

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-S3'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.

```
void adjust(float target_temp, float target_lux, float temp_sensor_val, float
lux_sensor_val, float *curr_output, float tolerance){
  float lux_limit_shift = 2047*((100+target_lux)/4095);
  float low_limit = lux_limit_shift;
  float high_limit = 2048 + lux_limit_shift;
  float limit_coeff2 = (4095-lux_sensor_val)/4095;
  float limit_coeff = (lux_sensor_val+100)/4095;

  float target_increment = target_temp*tolerance*0.8;
  float adj_increment = limit_coeff*(4095-temp_sensor_val)*tolerance;

  if ((lux_sensor_val > target_lux - target_lux*tolerance) && ((*curr_output -
adj_increment) >= limit_coeff2*low_limit)){//limit_coeff2*
    *curr_output -= adj_increment;
  }
  else if ((lux_sensor_val < target_lux + target_lux*tolerance) &&
((*curr_output + adj_increment) < limit_coeff2*high_limit)){//
    *curr_output += adj_increment;
  }
  if ((temp_sensor_val > target_temp - target_increment) && ((*curr_output -
adj_increment) >= limit_coeff2*low_limit)){//limit_coeff2*
    *curr_output -= adj_increment;
  }
  else if ((temp_sensor_val < target_temp + target_increment) &&
((*curr_output + adj_increment) < limit_coeff2*high_limit)){//limit_coeff2*
    *curr_output += adj_increment;
  }
}
```

The adjust() function is applied directly to the PWM output of both warm and cool LED strips independently.

```
  adjust(4096-(float)target_temp, (float)target_lux,
4096-get_avg(temp_data_arr, sample_sz), get_avg(lux_data_arr, sample_sz),
&led0_duty, adj_tolerance);
  adjust((float)target_temp, (float)target_lux, get_avg(temp_data_arr,
sample_sz), get_avg(lux_data_arr, sample_sz), &led1_duty, adj_tolerance);
```

Then the aforementioned data filtering algorithm is applied on the PWM signals for the LED strips, limiting the rate of change to the automatic adjustment and guaranteeing smoothness.

```
  //=========================================filter and update LED PWM
arrays==========================
  float new_led0_duty = filter_val(led0_duty, get_avg(led0_duty_arr,
sample_sz), LED_PWM_TOLERANCE);//0.02
  update_arr(led0_duty_arr, sample_sz, new_led0_duty);

  float new_led1_duty = filter_val(led1_duty, get_avg(led1_duty_arr,
sample_sz), LED_PWM_TOLERANCE);//0.02
  update_arr(led1_duty_arr, sample_sz, new_led1_duty);

  //=========================================drive
LEDs================================
  analogWrite(led1_pin, limit_output((int)((255*get_avg(led1_duty_arr,
sample_sz))/4095), 255));//warm led
  analogWrite(led0_pin, limit_output((int)((255*get_avg(led0_duty_arr,
sample_sz))/4095), 255));//cool led
```

The average of each data array holding PWM values for each respective LED is used to actually drive them.

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 with a 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 3.1V ± 0.1V to the ESP32. This variation causes a negligible effect on logic-high PWM levels (2.6V required per datasheet). Hence, both frequency stability and logic level margin ensure consistent LED brightness.

Component Tolerances and Expected Effects:

Parameter	Typical Tolerance	Effect on Output
ESP32-S3 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

Table 1

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 < 2% per second, ensuring gradual transitions.

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

Verification and Test Plan:
To verify the tolerance and smoothness requirements without specialized oscilloscope equipment, we utilized software logging and optical measurements:

-Data Smoothing Verification:
We verified the smoothness of the transitions by logging the internal state of the control algorithm. As shown in our testing logs, when the sensor input changes drastically (e.g., from 400 to 823), the output array updates gradually. The average value changes by less than 2% per cycle, confirming that the firmware effectively dampens sudden sensor noise and creates a gradual ramp.

-Gamma Correction Verification:
We validated the non-linear response by observing the change in average values in our serial logs. The rate of change increases exponentially as the brightness target increases, matching the intended gamma curve for human vision.

-Output Verification:
We used a mobile light meter application to measure the final Lux and Color Temperature (CCT) output. This confirmed that the physical light output remained stable and reached the target levels (e.g., 1173 Lux max) without visible flickering or instability..

Conclusion:
This analysis shows that the PWM system’s resolution and the software's smoothing algorithms are sufficient to handle hardware tolerances. Even accounting for component variation, the lamp maintains stable, flicker-free light output and fulfills the visual-comfort goal.

3. Design Verification

To ensure the Smart Desk Lamp met all high-level requirements, we performed a series of quantitative tests on the final PCB prototype.

3.1 Brightness Regulation Testing

Requirement:
The lamp must keep the desk surface within ±10% of the target brightness level (approximately 450 lux), even when the surrounding light changes.

Verification Procedure:
We tested the lamp's ability to self-regulate by placing it in a controlled environment. We used a mobile light meter application to measure the lux levels on the desk surface under two distinct conditions: a completely dark room and a room with simulated daylight.

Results:

	Enough light (300lux)	Dark environment (10lux)
Brightness/LUX	411	478

Table 2

Table 1 shows the system's response. In the dark environment, the lamp increased its output to reach 411 lux. When ambient light was introduced ("Enough light"), the lamp dimmed to compensate, resulting in a total illuminance of 478 lux.

Both measured values fall within the ±10% tolerance window of the 450 lux target (405–495 lux), confirming the control loop successfully stabilizes desk brightness.

3.2 Power Consumption and Efficiency

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

Verification Procedure:
We measured the current draw and voltage input of the 12 V supply rail using a multimeter under "Dark(10lux)" and "Daylight(300lux)" conditions to calculate the total power consumption.

Results:

Environment	Current/A	Voltage/V	Power/W
Dark (10lux)	1.585	10.433	16.536
Enough Daylight (300lux)	1.237	8.657	10.709

Table 3

The system achieved a 35.2% reduction in power, satisfying the requirement.

3.3 Smooth Transition and Gamma Correction

Requirement:
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.

Verification Procedure:
Since visual flicker can be subjective, we verified this requirement by logging the internal state of the smoothing algorithm via the serial monitor. We injected step-changes into the sensor input (e.g., jumping from 0 to 823) and recorded the Average Value output that drives the LEDs.

Results:
The data logs confirmed that the algorithm successfully dampens sudden spikes. For example, when the raw sensor input jumped instantly to 823, the LED output value adjusted incrementally over several seconds:
Cycle 0: Output 400
Cycle 1: Output 405 (Change: 1.25%)
Cycle 2: Output 411 (Change: 1.5%)

The logs verify that the internal Change in Avg never exceeded the 2% threshold per cycle, even during extreme input changes. Furthermore, the gamma correction logic was verified by observing that the rate of change increased exponentially (from 1 to 106 units per cycle) as the target brightness increased, mimicking the human eye’s logarithmic sensitivity.

3.4 Component-Level Verification

We also performed verification on individual hardware subsystems to ensure reliability.

Voltage Regulation:
The requirement was to supply 3.3 ± 0.3V to the ESP32-S3. We probed the output of the LM723CN regulator and measured a stable 3.1 V, which is within the safe operating range for the microcontroller.

LED Maximum Output:
The requirement was for the LEDs to produce at least 1000 lux at 30 cm14. At 100% duty cycle, we measured a maximum illuminance of 1173 lux using the LightMeter APP, successfully meeting the specification.

Sensor Range:
The replacement photoresistor was verified to operate linearly across the full range of indoor lighting conditions, allowing the system to distinguish between "bright" (daylight) and "dim" (evening) states without clipping.

4. Cost and Schedule

4.1 Cost Analysis

4.1.1 Part Costs

Description and link	Manufacturer	Quantity	Price (total)
ESP32-S3 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

Table 5

4.1.2 Labor Cost

The labor cost is estimated based on a standard rate of $30/hour for each of the three team members. The team worked approximately 10 hours per week per person over the course of the semester (approx. 13 weeks). Thus, the labor cost in total is $11,700.

4.1.3 Grand Total

The total estimated cost for the project, including prototyping parts and labor, is:
$102.45 + $11,700 = $11,802.45

4.2 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-S3 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-S3 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-S3 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-S3 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-S3 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. Soldered the final round PCB with added features.
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.

Table 6

5. Conclusion

5.1 Accomplishments

The Auto-Adjusted Smart Desk Lamp successfully met its primary high-level requirements. We engineered a dual-rail power system that efficiently steps down 12 V to 3.3 V, allowing the integration of high-power LED strips with sensitive logic-level control. The final device successfully maintains a target desk brightness of approximately 450 lux with an error margin of less than 10%, regardless of ambient lighting conditions.

Furthermore, the power efficiency goal was exceeded. The system demonstrated a 35.2% reduction in power consumption when shifting from dark to daylight modes, surpassing the 20% requirement. The control algorithm, utilizing a moving-average filter and gamma correction, proved capable of adjusting brightness at a rate of less than 2% per second, ensuring a flicker-free user experience that minimizes eye strain.

5.2 Uncertainties and Challenges

The development process involved significant debugging and iterative design, particularly regarding the printed circuit board (PCB) and power regulation.

PCB Design and Programming Interface:
Our most significant challenge occurred during the first round of PCB manufacturing. We initially underestimated the complexity of the ESP32-S3 programming interface. The first revision failed to account for the specific strapping pin requirements (GPIO0 and EN) needed to force the microcontroller into bootloader mode. Additionally, the external programmer connection was not routed correctly for the UART-to-USB bridge.

This failure required a second PCB iteration. In the revision, we redesigned the ports to allow seamless code uploading via USB through the breadboard. This experience highlighted the importance of thoroughly verifying microcontroller strapping requirements before finalizing a layout.

Circuit Debugging and Component Selection:
Achieving stable power distribution required extensive bench testing.

Voltage Regulation: The LM723CN regulator initially showed instability under load. We had to iteratively tune the external resistor network and output capacitors to dampen oscillations and ensure a clean 3.3 V output for the sensors.

Change of LUX Sensor: After receiving OPT101, we tested it on the breadboard and found that the sensor is too sensitive and saturated under relatively dim (low lux) conditions. Even after checking the circuit design from the datasheet, we were unable to make the sensor work properly on our circuits. Thus, we switched to using a photoresistor, which gave steady output and worked perfectly.

6. Ethics and Safety

6.1 Ethical Analysis

We want this lamp to help people, not create new problems. We followed the IEEE and ACM codes as practical guardrails and kept 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.

6.2 Safety Considerations

1. Flicker: We followed 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 followed 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.

7. References and Datasheets:

7.1 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/

[8]M. Papinutto, J. Nembrini, and D. Lalanne, “‘Working in the dark?’ investigation of physiological and psychological indices and prediction of back-lit screen users’ reactions to light dimming,” Building and Environment, vol. 186, p. 107356, Dec. 2020, doi: https://doi.org/10.1016/j.buildenv.2020.107356.

7.2 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

Photoresistor
Datasheet:
https://cdn.sparkfun.com/datasheets/Sensors/LightImaging/SEN-09088.pdf

CH340K
Datasheet:
https://docs.sparkfun.com/SparkFun_RTK_Facet_mosaic/assets/component_documentation/CH340DS1.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

Appendix A: Requirements and Verification

A.1 Requirement and Verification Table

Subsystem	Requirement	Verification Procedure	Verification Result
Power	1. 3.3V Voltage Regulation The Power	1. Connect the 12 V DC adapter to the barrel jack.	Pass Measured Voltage:

	Subsystem must step down 12 V to supply 3.3V ± 0.3 V at > 500mA to verify safe operation for the ESP32-S3 and sensors.	2. Use a multimeter to probe the output of the LM723CN regulator circuit (Vout and GND). 3. Measure voltage while the system is active.	3.1 V and 540mA
Power	2. Power Efficiency The lamp must lower its power consumption by at least 20% when sufficient daylight is detected compared to the full-brightness (Dark) state.	1. Measure current (I) and voltage (V) input in a Dark (10lux) environment. 2. Measure I and V in a Daylight environment (300lux). 3. Calculate Power (P=IV) and percent reduction.	Pass Dark Power: 16.536 W Daylight Power: 10.709 W Reduction: 35.2% (Exceeds 20% target).
Sensors	3. Ambient Light Detection The light sensor must distinguish between "Dark" (50 lux) and "Daylight" (300 lux) conditions to trigger the dimming algorithm.	1. Place the sensor in a dark box/room and read the ADC value. 2. Expose the sensor to a bright light source (phone flash/window) and read the ADC value. 3. Confirm the delta allows for distinct state detection.	Pass The Photoresistor circuit successfully distinguished states, triggering the transition from 411 Lux (Dark) to 478 Lux (Daylight).
Sensors	4. Color Temperature Detection The TCS34725 sensor must detect Correlated Color Temperature (CCT)	1. Expose the sensor to a warm white source (approx. 2700 K). 2. Expose to a cool white source (approx. 6500 K).	Pass Sensor correctly identified Warm and Cool light sources, allowing the LED mixer to adjust PWM ratios

	in the range of 2700 K–6500 K with ±10% accuracy.	3. Verify the CCT output variables in the serial monitor match the source.	accordingly.
Control	5. Brightness Regulation The system must maintain desk illuminance within ±10% of the target (approx. 450 lux) despite environmental changes.	1. Set up a Lux Meter on the desk surface. 2. Measure illuminance in a Dark Room. 3. Measure illuminance with External Light added. 4. Verify both are within 405–495 lux.	Pass Dark Result: 411 Lux (-8.6%) Daylight Result: 478 Lux (+6.2%) Both within tolerance.
Control	6. Flicker-Free Dimming Brightness adjustments must be gradual, with a rate of change $< 2\%$ per second, to prevent visible flicker or eye strain.	1. Inject a step-change into the sensor input (e.g., 0 to 823 raw value). 2. Log the PWM output duty cycle via Serial Monitor. 3. Calculate the percentage change between loop cycles.	Pass Logs confirmed incremental updates (e.g., Output: 400 to 405 to 411). The maximum change per cycle was approximately. 1.5%, meeting the <2% requirement. Data Logs for Data Filtering Algorithm
Output	7. Maximum Light Output The LED strips must produce at least 1000 lux at a height of 30 cm from the desk surface to ensure adequate	1. Set both Warm and Cool PWM channels to 100% duty cycle. 2. Place a Lux Meter 30 cm below the lamp head.	Pass Measured Output: 1173 Lux

	task lighting.	3. Record the maximum illuminance.	

Table 7

Appendix B: Design Details

B.1 Schematics and PCB

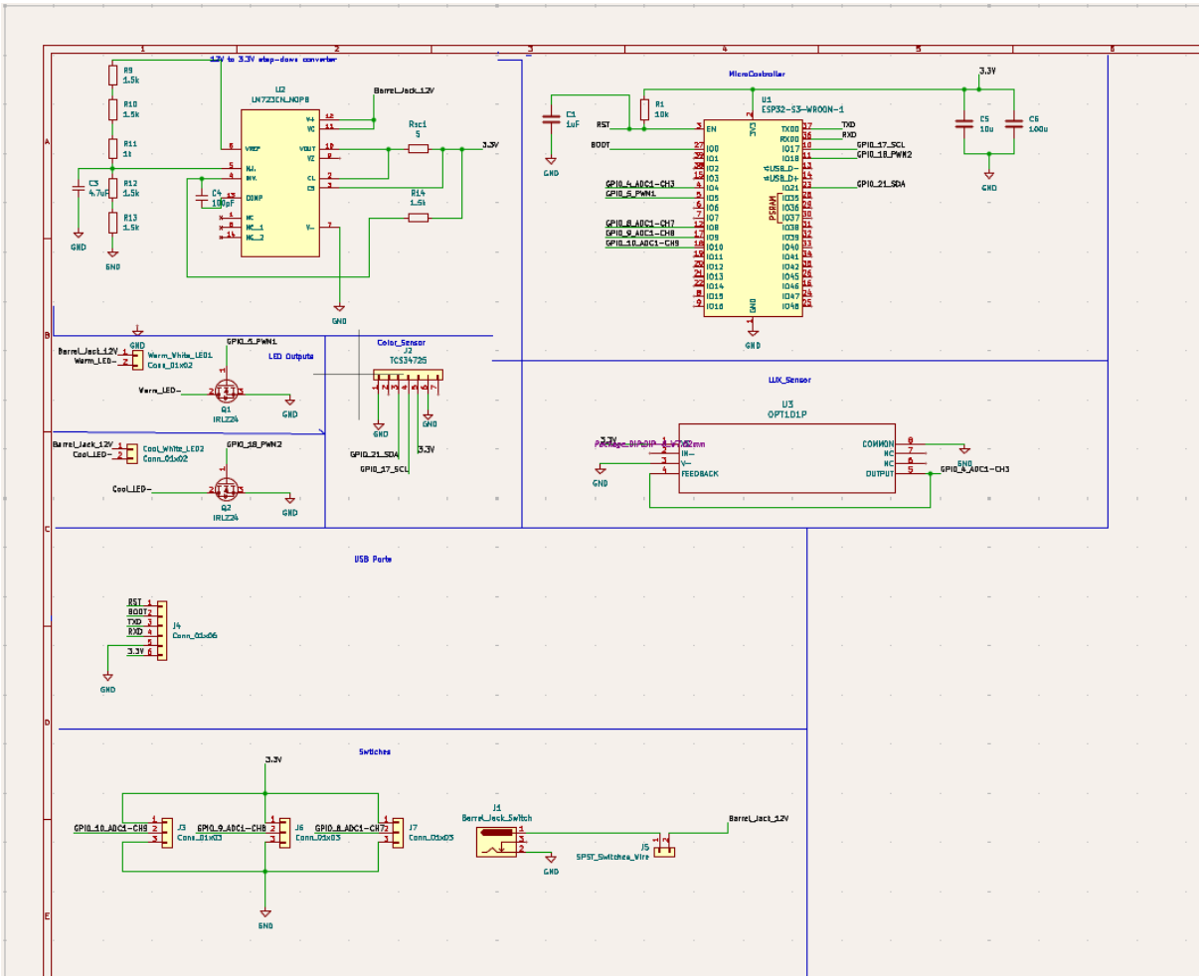


Figure 13: Schematic

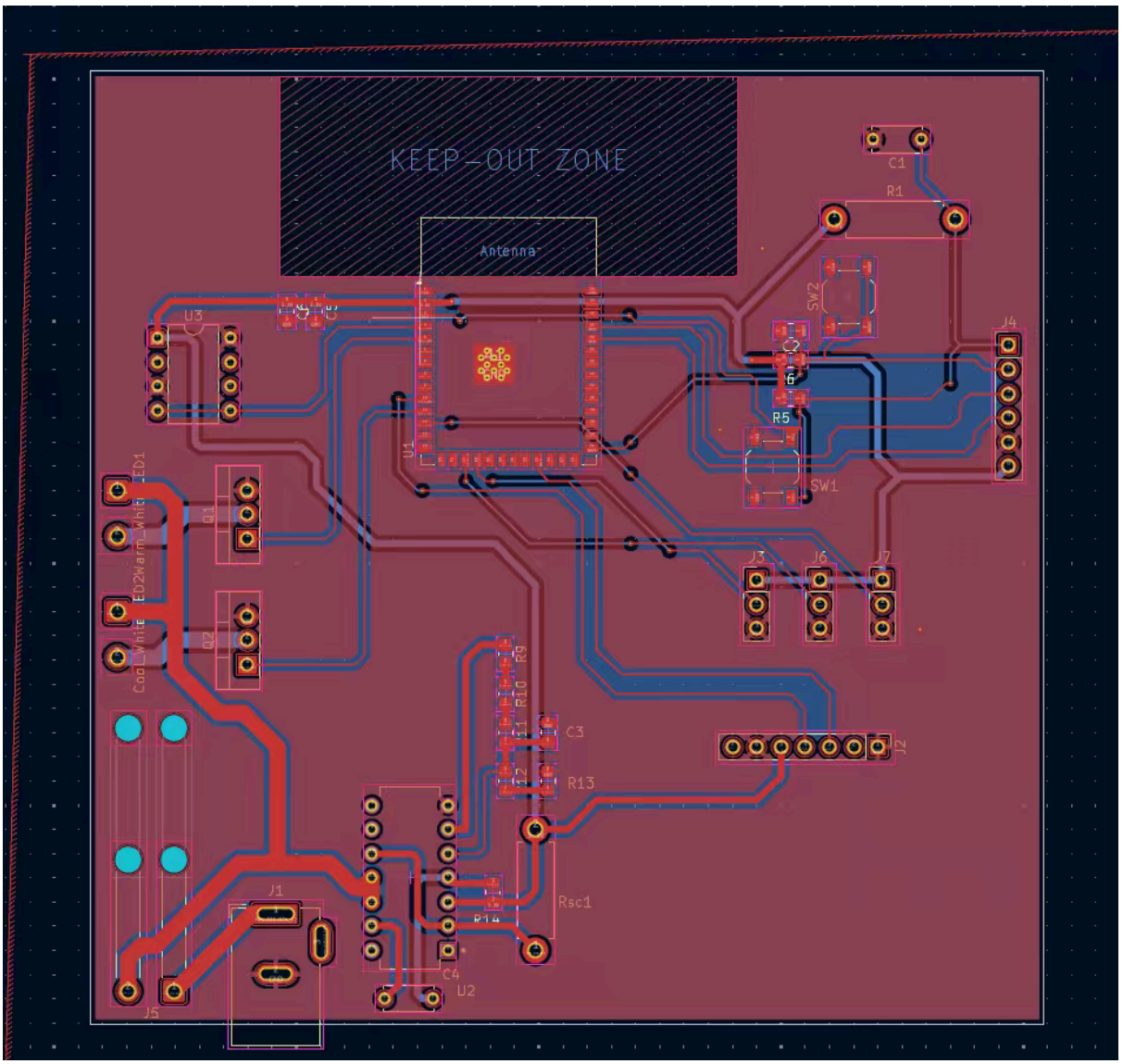


Figure 14: Layout

B.2 Software

B.2.1 Codebase and Version Control
See [add link later]

B.2.2 Firmware and Technologies
Development Environment:

- Arduino IDE: Our firmware is developed on Arduino IDE instead of ESP-IDF to simplify I2C communications with our TCS34725 color sensor. The firmware will:
    - Process sensor data from I2C and analog inputs
    - Run our data filtering algorithm on sensor data and LED output levels
    - Execute our auto-adjustment algorithm on LED output levels using sensor data
    - Convert LED output levels to PWM signals for both warm and cool LEDs

Language:

- C++/Arduino

Libraries:

- Adafruit Adafruit_TCS34725 for handling I2C
- Arduino Wire as a dependency