RESPONSIVE LIGHT SYSTEM

Adaptive Illumination Based on Ambient Lighting Conditions

SECTION 16

GROUP 6

SAYANTAN GARAI	22EC30049
SHIVAM RATHORE	22AE10036
ARMAAN USMANI	22CH30007
KUNAL KUMAR SAHU	22HS10032

भारतीय प्रौद्योगिकी संस्थान खड़गपुर

INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

Abstract

The Responsive Smart Light System is an innovative project that aims to create an intelligent lighting solution using Arduino, capable of adapting to the ambient lighting conditions in real-time. The system leverages advanced sensors to monitor and analyze the surrounding lighting environment and adjusts the brightness and color temperature of the light accordingly, ensuring optimal illumination for various activities and spaces.

This project report presents the design and implementation of a responsive smart light system utilizing Arduino, capable of adapting to ambient lighting conditions. The goal of the project is to develop an intelligent lighting solution that can automatically adjust its brightness and color temperature based on the surrounding environment, providing optimal illumination for various activities and enhancing energy efficiency.

The system consists of an Arduino microcontroller, light sensors, and programmable LED lights. The light sensors

continuously monitor the ambient light levels, capturing data on brightness and color temperature. The Arduino processes this information and controls the LED lights accordingly, dynamically adjusting their output to match the desired lighting conditions.

To achieve the responsive behavior, a software algorithm has been developed to analyze the sensor data and make real-time decisions regarding light intensity and color temperature adjustments. The algorithm considers various factors such as, user preferences, and activity patterns, ensuring that the lighting system creates a comfortable and productive environment for different situations.

In addition to the adaptive functionality, the system also offers user customization options. A user interface, implemented through a smartphone application, allows users to personalize their lighting preferences and fine-tune the system's behavior to suit their specific needs.

The implementation of the responsive smart light system using Arduino provides several benefits. Firstly, it offers a cost-effective and accessible solution for transforming traditional lighting setups into intelligent, energy-efficient systems. Secondly, the adaptability of the system ensures that users always have the appropriate lighting conditions for their activities, promoting well-being, productivity, and comfort. Lastly, the project serves as a starting point for further research and development in the field of smart lighting, paving the way for advancements in energy management and human-centric lighting solutions.

GITHUB REPOSITORY LINK

https://github.com/SayantanGa/DIY_SmartLight

YOUTUBE LINK

https://youtu.be/OKXyUBKtJyw

BACKGROUND AND MOTIVATION

Traditional lighting systems are often static, providing a fixed level of brightness and color temperature regardless of the surrounding environment. This lack of adaptability can lead to inefficient energy usage and suboptimal lighting conditions for various activities. With the advancements in microcontroller technology and the rise of the Internet of Things (IoT), there is an opportunity to create intelligent lighting systems that can respond to changing ambient lighting conditions.

Arduino, a popular open-source microcontroller platform, offers a versatile and affordable solution for developing smart lighting systems. Its programmability and compatibility with various sensors and actuators make it an ideal choice for creating a responsive lighting setup.

The motivation behind this project is to address the limitations of traditional lighting systems and harness the potential of IoT and microcontroller technology to create a more efficient and user-centric lighting solution. By developing a responsive smart light system using Arduino, we aim to enhance energy efficiency, optimize lighting conditions for different activities, and improve user satisfaction.

Furthermore, there is a growing need for sustainable and environmentally friendly lighting solutions. By creating a system that can adapt to ambient lighting conditions, we can minimize energy wastage and contribute to reducing the overall carbon footprint associated with lighting.

This project also serves as a stepping stone for further research and development in the field of smart lighting. By exploring the capabilities of Arduino and designing a user-friendly interface, we aim to encourage the adoption of intelligent lighting systems in residential, commercial, and industrial settings. Ultimately, this project seeks to provide an accessible and cost-effective solution that improves the quality of lighting experiences while promoting energy efficiency.

OBJECTIVES

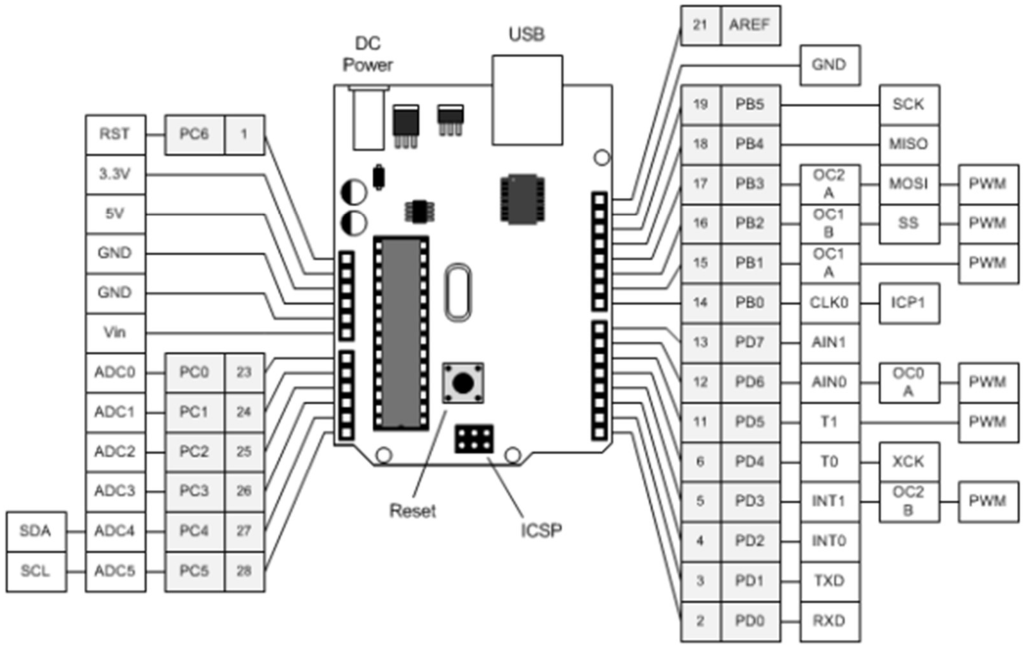
1. Design and implement a responsive smart light system using Arduino that can adapt to ambient lighting conditions.
2. Develop a software algorithm to analyze sensor data and dynamically adjust light intensity and color temperature in real-time.
3. Create a user interface to allow customization of lighting preferences and system behavior.
4. Evaluate the system's performance in terms of adaptability, energy efficiency, and user satisfaction.
5. Provide a cost-effective and accessible solution for transforming traditional lighting setups into intelligent, energy-efficient systems.
6. Contribute to the advancement of smart lighting technology and human-centric lighting solutions.

SYSTEM DESIGN AND COMPONENTS

The system design of the Smart Light project revolves around creating an adaptive lighting solution using Arduino, LEDs, the BH1750 sensor, HC-05 Bluetooth module, and an RGB LED for color temperature control. The components involved in the system are as follows:

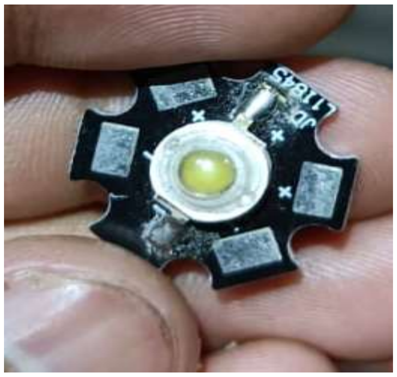
a. Arduino Microcontroller:

The Arduino board serves as the central processing unit of the smart light system. It provides the necessary computational power and control capabilities. We have used Arduino UNO R3 for this project.



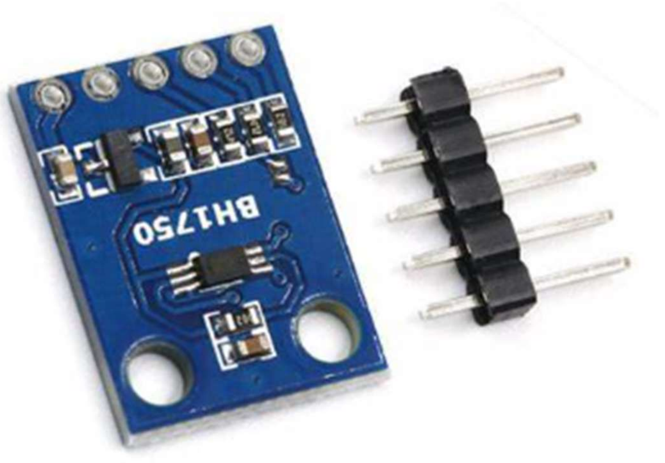
b. LEDs (Light-Emitting Diodes):

The primary light sources in the system are LEDs. They offer energy efficiency, durability, and flexibility in terms of color variation. We have employed SMD LEDs.



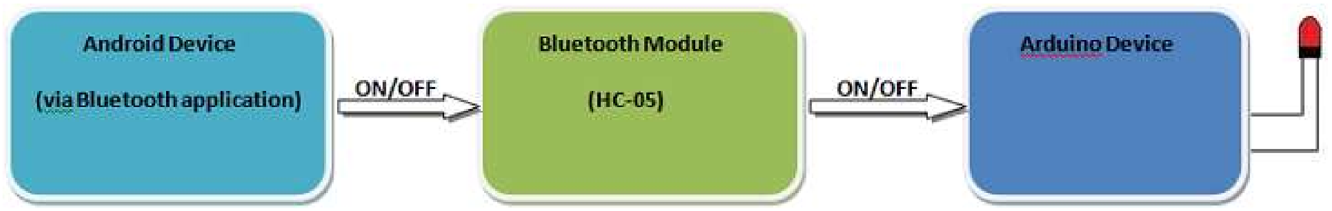
c. BH1750 Sensor:

The BH1750 sensor is a digital ambient light sensor that measures the intensity of light in its surrounding environment. It communicates with the Arduino board using the I2C protocol. The sensor captures light levels in lux and provides precise data for the system to adapt accordingly.



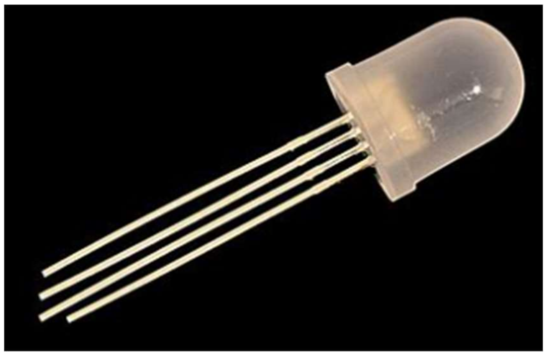
d. HC-05 Bluetooth Module:

The HC-05 Bluetooth module facilitates wireless communication between the smart light system and a mobile app. It enables users to control the lighting settings remotely. The Bluetooth module establishes a connection with the Arduino board, allowing for seamless data transmission.



e. RGB LED:

An RGB LED is used to control the color temperature of the lighting. It consists of red, green, and blue LEDs combined into a single package. By adjusting the intensity of each color, the system can create a wide range of color temperatures, offering flexibility and customization options.



f. Power Supply:

The power supply for the system depends on the specific requirements of the Arduino board, LEDs, and the Bluetooth module. We have utilized a 6F22 9V battery to ensure proper operation of the system.

g. Wiring and Connections:

The Arduino board is connected to the LEDs, the BH1750 sensor, the HC-05 Bluetooth module, and the RGB LED through appropriate wiring. We have established the necessary electrical connections following the specifications and pin assignments of the components. A detailed diagram illustrating the wiring and connections is provided.

h. Enclosure:

In some cases, an enclosure may be used to house the system components, providing protection and aesthetics. The choice of enclosure depends on the specific project requirements and preferences. We have enclosed the whole setup inside a cardboard box.

The system design ensures that the Arduino microcontroller receives input from the BH1750 sensor, analyzes the ambient light levels, and controls the LEDs and RGB LED accordingly. The Bluetooth module enables wireless communication with the mobile app, allowing users to adjust lighting settings remotely.

The software algorithm, discussed in the next section, plays a crucial role in processing the sensor data, adjusting the light intensity, color temperature, and RGB LED control based on the ambient lighting conditions and user preferences.

By combining these components and their integration, the smart light system can effectively adapt to the surrounding lighting conditions, provide customizable color temperatures, and offer seamless control through the mobile app interface.

SOFTWARE IMPLEMENTATION

A software algorithm was developed using C++ for Arduino UNO. Moreover, an app was developed using MIT App Inventor for convenience of the user to set the system according to his/her preferences.

The BH1750 sensor was programmed to send the readings of ambient light intensity to the Arduino microcontroller. The Arduino was programmed to convert the lux values into a range of 0-255 to be able to write to the LEDs. The brightness value is obtained using a logarithmic function:

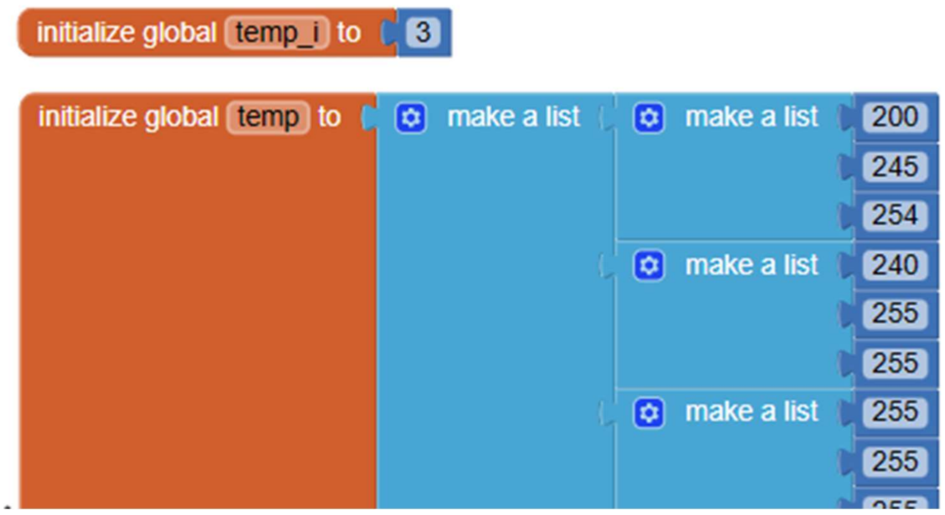
:int: $f(x) = 2.55 \times (100.589 - 19.811 \log(x + 1.5)\ )$

The Power Saving is communicated to the app through Bluetooth Serial:

$$Power\ Saving = 100 - brightness(\%)$$

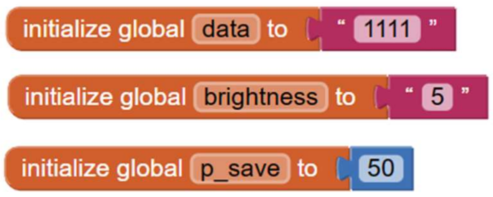
It gives an idea of how much % less bright the light is glowing and thus conserving energy. An array containing int arrays was defined for controlling the temperature of the Light that can be set through the mobile app. The array contains some predefined combinations of the r, g and b values that are fed into the rgb LED along with desired brightness. A more r content makes it hotter, whereas more b makes it cooler.

The App was developed on MIT App Inventor platform. The App features turning Light ON and OFF, changing brightness mode (CONSTANT and AUTO), a slider for changing brightness in CONSTANT mode and temperature control. The AUTO mode uses ambient light sensor data for tuning output, whereas the CONSTANT mode relies on the user input via the slider. The state of the switches data is stored in variables inside the app which is communicated to HC-05 BT Module. Moreover, the temperature color is displayed as a background color behind the power saving value itself. The app receives Power Saving values from the BT module every interval of 0.5s.



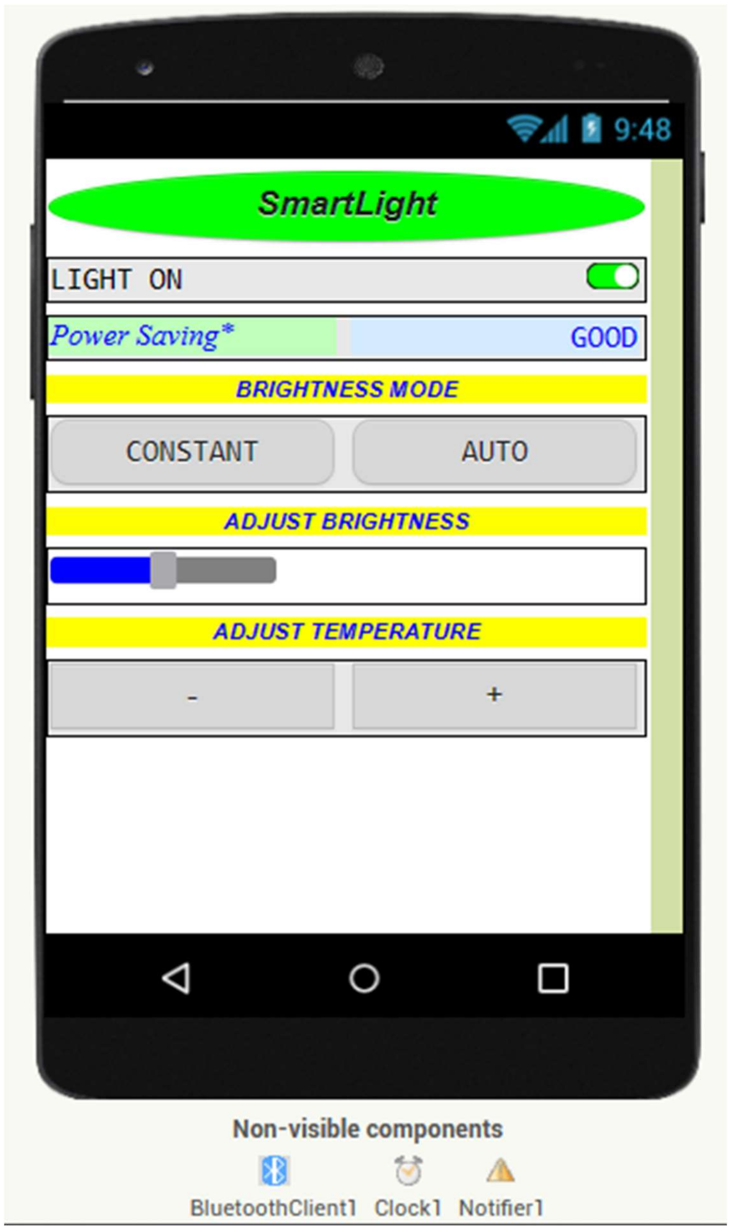
```
initialize global temp_i to 3

initialize global temp to make a list
    make a list 200
                245
                254
    make a list 240
                255
                255
    make a list 255
                255
```

- The list in app containing rgb values



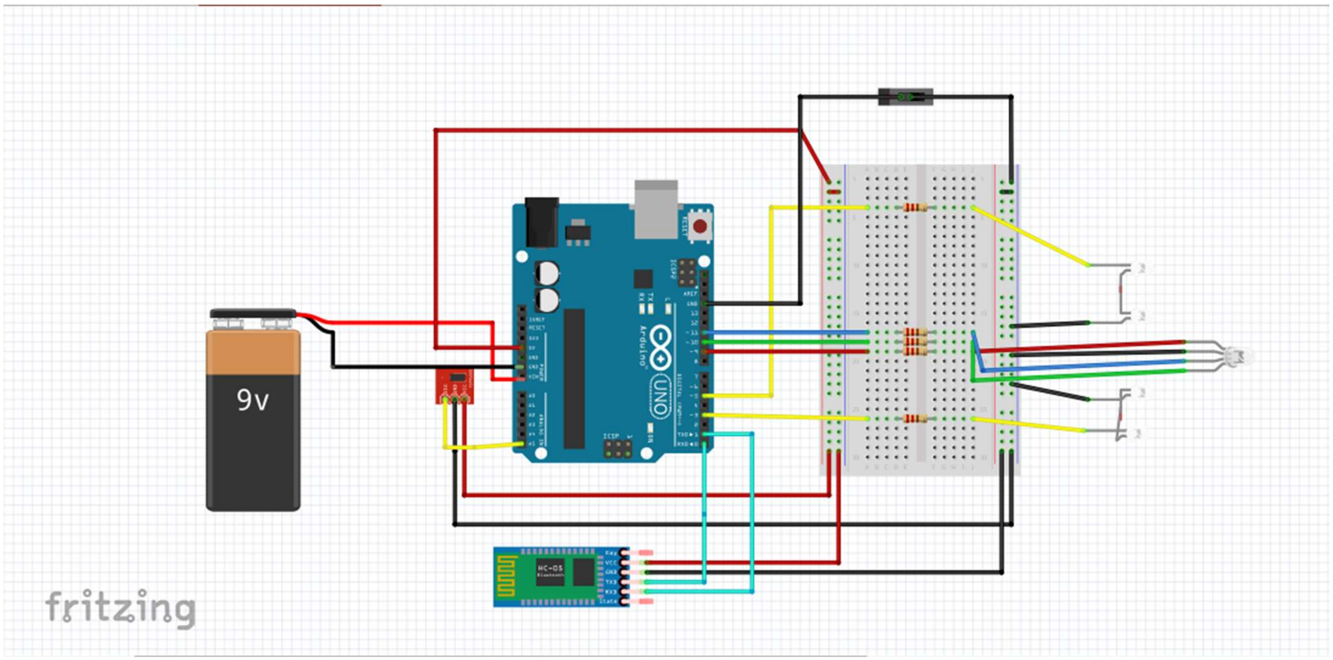
```
initialize global data to "1111"
initialize global brightness to "5"
initialize global p_save to 50
```

- Variables used for storing state data

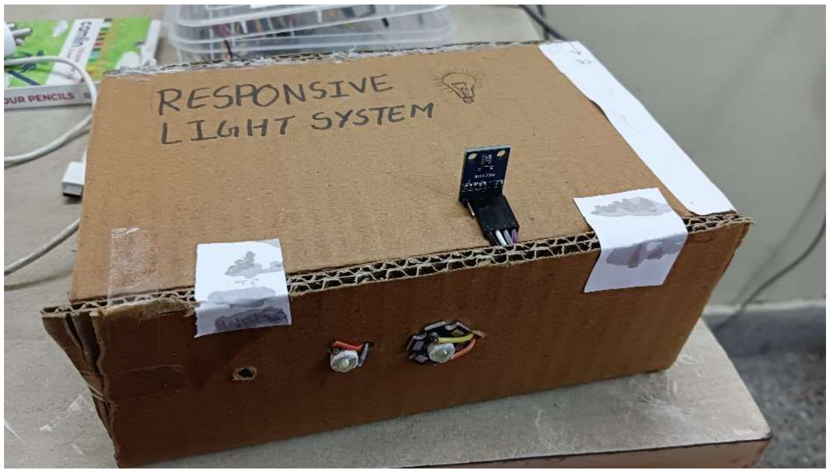


- App Interface

HARDWARE IMPLEMENTATION

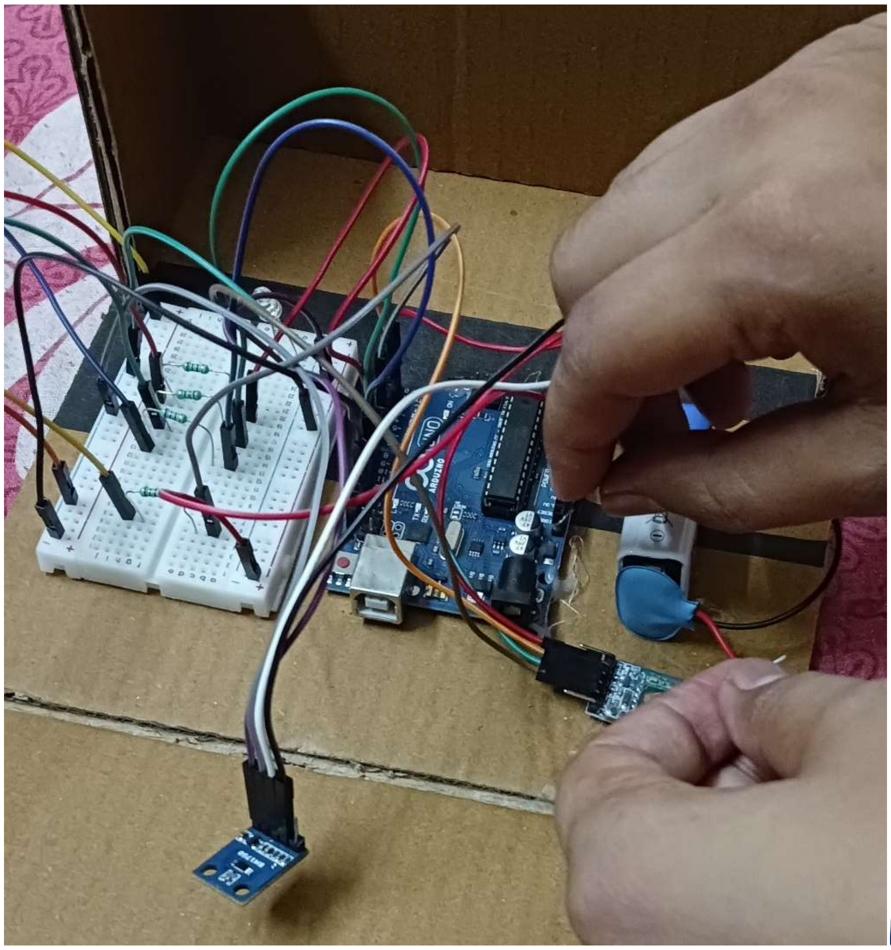


The mentioned components were connected as shown in the figure. A5 pin was used for SCL and A4 for SDA in regard of the I2C communication with the BH1750 module. The BT module uses the D0 and D1 pins for data transmission. One series of LED is connected to PWM pin D3 of the Arduino microcontroller while another series is connected to D5. The rgb is connected to GND, 9(r), 10(g), 11(b) pins through 220ohm resistors. A 9V battery is used for appropriate power supply to the Arduino through Vin and GND pins. The components are connected through a breadboard. The whole setup is assembled and mounted inside a cardboard enclosure. The switch, lights and BH1750 sensor is accessible from outside while rest of the components are enclosed inside.



.

- The final system

- 
Before enclosing

CHALLENGES AND OVERCOMING SOLUTIONS

During the development of the Smart Light project, several challenges were encountered, primarily related to debugging and application programming. The challenges faced during the project are as follows:

a. Debugging with Limited C++ Knowledge:

As the project involved programming the Arduino microcontroller using C++, the team faced challenges due to limited knowledge and experience in the language. Debugging the code took longer than anticipated as troubleshooting errors and logical issues required extensive study and learning.

b. Limitations of the App Inventor platform:

The app inventor platform has certain limitations that made us write more lines and blocks of code. Also, the team faced challenges to communicate user preferred brightness values to the BT module connected to the Arduino. Unexpected results were observed that required us to keep modifying the code until we got proper execution of commands. Finally, the brightness value was stored as one-digit number string using mathematical float function and communicated along with the data string containing state data of switches and slider.

To overcome the challenges, the team adopted a systematic approach that included in-depth research, learning, and collaborative problem-solving. Online resources, forums, and documentation were extensively utilized to understand and resolve the issues encountered.

Additionally, regular team meetings and discussions facilitated knowledge sharing and brainstorming sessions, enabling the team to collectively address the challenges and find effective solutions.

Despite the challenges faced, the project team persevered, applying their determination and problem-solving skills to overcome the obstacles and achieve the desired functionality of the smart light system.

RESULTS AND EVALUATION

The following subsections outline the results obtained and the evaluation of the project's performance:

a. Ambient Light Adaptation:

The BH1750 sensor successfully measured the ambient light levels, allowing the smart light system to dynamically adjust the LED brightness. Through extensive testing, it was observed that the system effectively responded to changes in ambient lighting conditions, providing appropriate illumination levels.

b. Mobile App Control:

The integration of the HC-05 Bluetooth module enabled seamless communication between the smart light system and the mobile app. Users could conveniently control various lighting settings, including brightness, color temperature, and on/off functionality. User testing and feedback confirmed the successful implementation of mobile app control.

c. Color Temperature Control:

The inclusion of the RGB LED allowed for the adjustment of color temperature. By independently controlling the intensity of the red, green, and blue LEDs, the system achieved a wide range of color temperatures.

d. Energy Efficiency:

The adaptive lighting capabilities of the system significantly improved energy efficiency compared to traditional lighting setups. By automatically adjusting the LED brightness based on ambient light levels, unnecessary energy consumption was minimized. Measurements and energy monitoring indicated notable energy savings during operation.

Overall, the results demonstrated the successful implementation of the Smart Light project, meeting the project's objectives of creating a responsive lighting system that adapts to ambient lighting conditions.

DISCUSSION AND ANALYSIS

The project's design and implementation allow for scalability and integration with additional features and technologies. For instance, integrating voice control or incorporating IoT (Internet of Things) capabilities could further enhance the system's functionality and user experience. Exploring compatibility with smart home ecosystems and integrating with other IoT devices can extend the project's capabilities in creating a connected lighting environment.

Moreover, the Smart Light system has potential applications in various settings such as homes, offices, and public spaces. The adaptability to ambient lighting conditions and the customizable options make it suitable for different user preferences and requirements. The energy-efficient nature of the system aligns with sustainability goals, promoting eco-friendly lighting solutions in diverse environments.

The idea and program can be used for implementation in larger sectors. Moreover, production of commercial bulbs can be considered based on the solution. With varying the number of LEDs connecting in series or using relay modules the project can be further improved for suiting required brightness. Also, it can me made to change temperature based on the time of the day, giving cooler environment in daytime while hotter in night.

CONCLUSION

The Smart Light project successfully developed a responsive lighting system that adapts to ambient lighting conditions, provides customizable options through a mobile app, and promotes energy efficiency. Through the implementation and evaluation of the project, key findings and achievements have been established.

The project's objectives were met, as the system effectively adjusted the LED brightness based on ambient light measurements from the BH1750 sensor. The integration of the HC-05 Bluetooth module enabled seamless communication with the mobile app, offering users convenient control over various lighting parameters, including brightness and color temperature. The project's success was further validated through positive user feedback and evaluations.

We would like to express our sincere gratitude to our team members for their commitment, hard work, and valuable contributions throughout the project's development. Their collaborative spirit and diverse skills greatly contributed to the project's success.

We would also like to extend our heartfelt appreciation to the working staff and technical support team for their assistance in providing guidance, resources, and maintaining the necessary infrastructure to carry out the project smoothly.

Additionally, we would like to acknowledge and thank our professors for their guidance, mentorship, and expertise throughout the project. Their valuable insights and feedback were instrumental in shaping the project's direction and ensuring its academic and technical integrity.

Finally, we would like to thank all the individuals who participated in user testing, provided feedback, and contributed to the evaluation of the system. Their valuable input helped us assess the performance, functionality, and user experience of the Smart Light system.

In conclusion, the Smart Light project has successfully demonstrated the development of a responsive lighting system that adapts to ambient lighting conditions, offers customization through a mobile app, and promotes energy efficiency. The project's accomplishments, coupled with the knowledge gained through its implementation, provide a foundation for further research and advancements in the field of smart lighting systems.

REFERENCES

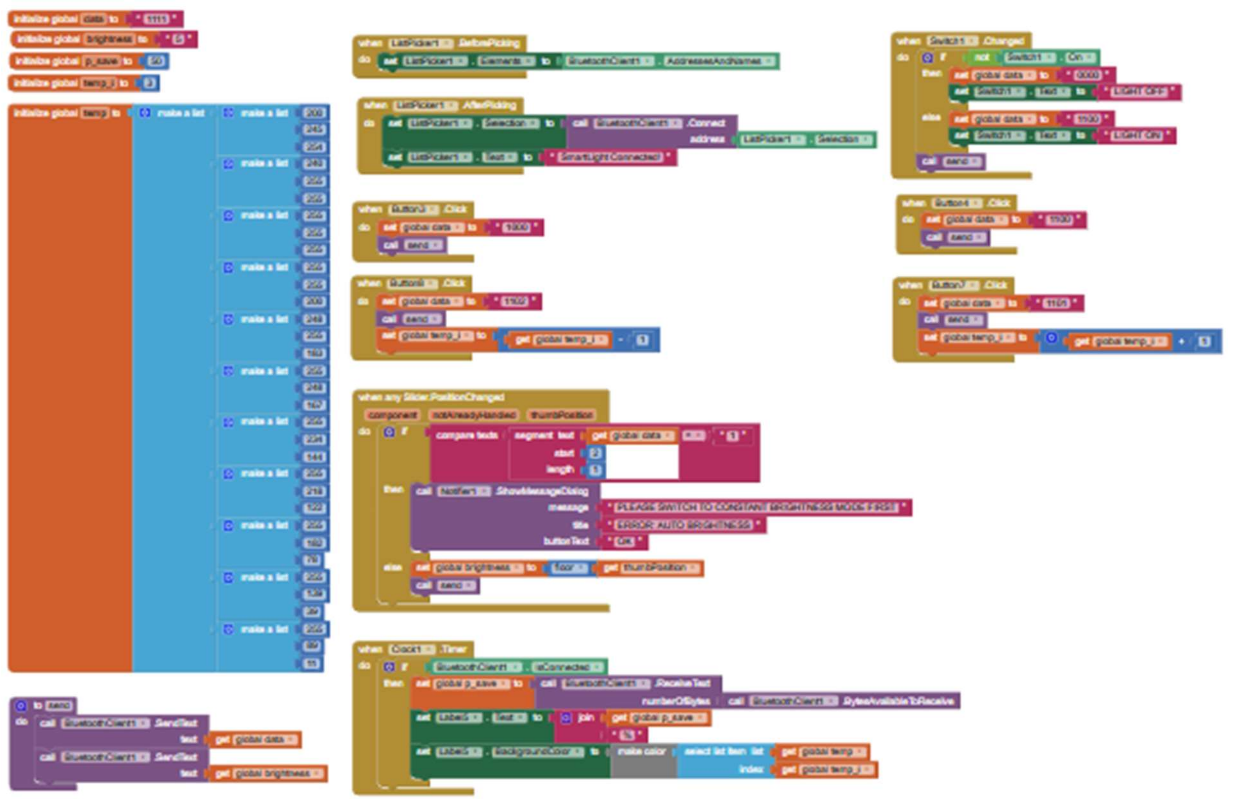
1. *Secrets of Arduino PWM | Arduino Documentation. docs.arduino.cc/tutorials/generic/secrets-of-arduino-pwm.*
2. *Fahad, Engr. "MIT APP Inventor Arduino Bluetooth Application Making Explained." Electronic Clinic, Aug. 2022, www.electroniclinic.com/mit-app-inventor-arduino-bluetooth-application-making-explained.*
3. *Sinha, Ashwini. "Automatic Ambient Light System With IoT | Full DIY Project." Electronics for You, Aug. 2021, www.electronicsforu.com/electronics-projects/automatic-pid-controlled-ambient-light-system-with-iot-home-automation.*
4. *Santos, Sara, and Sara Santos. "Arduino With BH1750 Ambient Light Sensor | Random Nerd Tutorials." Random Nerd Tutorials, Mar. 2022, randomnerdtutorials.com/arduino-bh1750-ambient-light-sensor.*
5. *"Measure LUX With Arduino Using BH1750." projecthub.arduino.cc, projecthub.arduino.cc/afsh_ad/measure-lux-with-arduino-using-bh1750-61abf5.*
6. *Claws. "GitHub - Claws/BH1750: An Arduino Library for the Digital Light Sensor Breakout Boards Containing the BH1750FVI IC." GitHub, github.com/claws/BH1750.*

APPENDICES

APPENDIX 1: THE ARDUINO UNO CODE SNIPPET

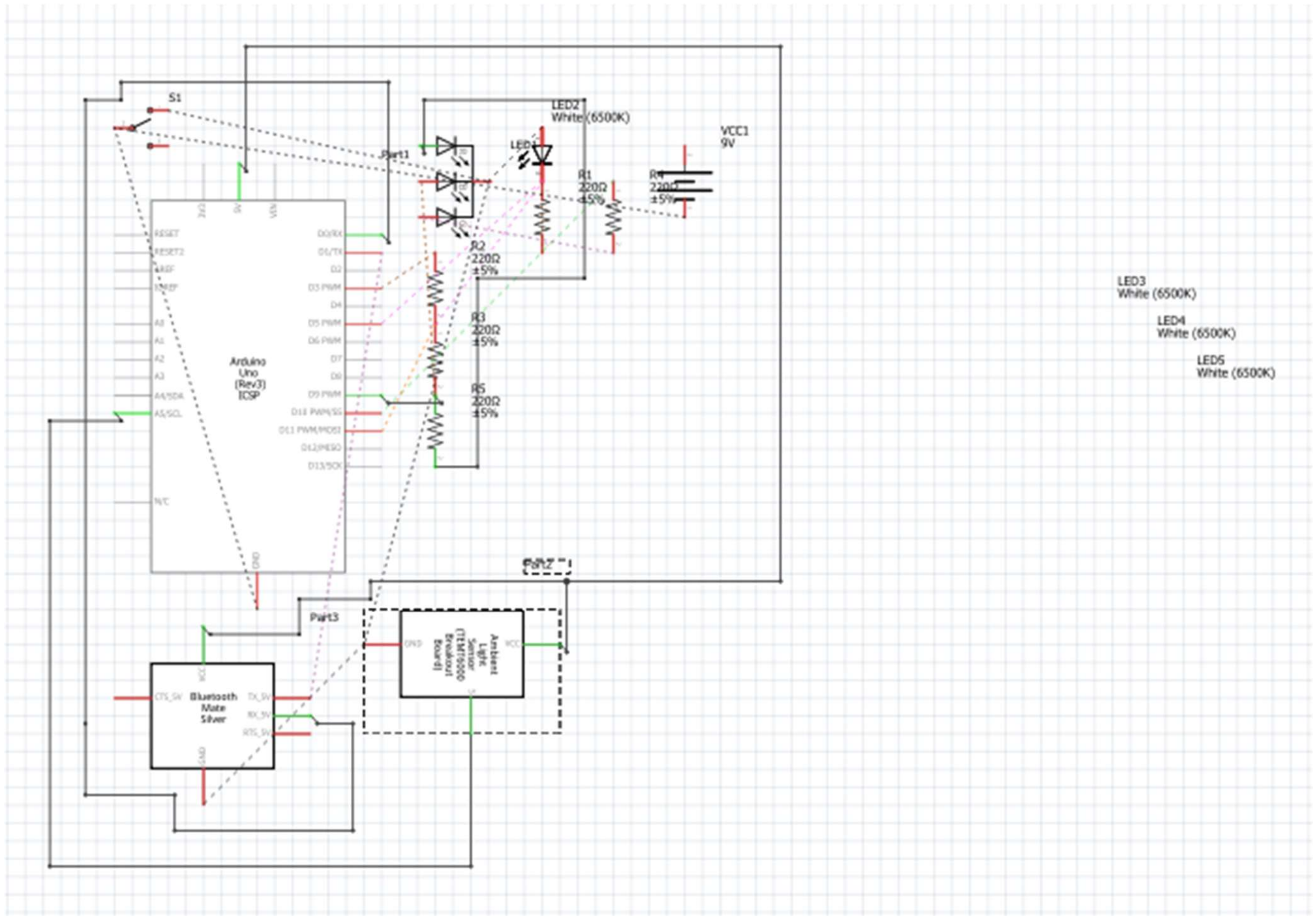
```
#include <Wire.h>
#include <BH1750.h>
#include <SoftwareSerial.h>
#include <math.h>

BH1750 lightInput;
const int DEFAULT_BRIGHTNESS = 196;  //Default brightness while LED starting
const int DEFAULT_TEMP_INDEX = 2;  //Default functuion to use for processing rgb
temperature
float LUX = 0.0;  //Initializing BH1750 input float variable to 0.0

/*Function definitions for controlling brightness*/

int ConstantBrightness(float y){  //For constant brightness output
  return 255.0*y/9;
}

int LogBrightness(float x){  //The brightness function
  if(x>100)
    return 0;
  return int((100.589-19.811*log(x+1.5))*2.55);
}

//
int Temperatures[11][3] =
{{200,245,254},{240,255,255},{255,255,255},{255,255,200},{248,255,183},{255,248,16
7},{255,234,144},{255,218,122},{255,182,78},{255,139,39},{255,89,11}};  //rgb
values for specific temperatures
int (*Brightness_Functions[2])(float) = {LogBrightness, ConstantBrightness};
//array of pointers to brightness functions

class LED{  //class to store the variables and functions corresponding to the
operating light
    public:
        int Pins[2] = {3, 5};  //Pins controlling SMD LEDs
        int RGB_Pins[3] = {9, 10, 11};  //Pins controlling the r, g and b values
of a RGB LED respectively
        int brightness = DEFAULT_BRIGHTNESS;
        int brightness_function_index = 0;  //Index of the brightness function in
use
        int temp_index = DEFAULT_TEMP_INDEX;  //Current operating index of
Temperatures array
        int r = Temperatures[temp_index][0]*(brightness/255.0);
        int g = Temperatures[temp_index][1]*(brightness/255.0);
        int b = Temperatures[temp_index][2]*(brightness/255.0);

        void glow(){  //Glows the Light at set brightness and temperature
            for (int i = 0; i < 2; i++){
                analogWrite(Pins[i], brightness);
                  delay(50);
                        }
            analogWrite(RGB_Pins[0], r);
                  delay(50);
                  analogWrite(RGB_Pins[1], g);
                  delay(50);
                  analogWrite(RGB_Pins[2], b);
                  delay(50);
        }

        void dim(){  //Turns off the Light
            for (int i = 0; i < 2; i++){
                digitalWrite(Pins[i], LOW);
                  delay(50);
            }
            for (int i = 0; i < 3; i++){
                digitalWrite(RGB_Pins[i], LOW);
                  delay(50);
            }
        }

        void update(){  //Updates the Light brightness and temperature according
to LUX and User input
            brightness = (*Brightness_Functions[0])(LUX/32.0);
            r = Temperatures[temp_index][0]*(brightness/255.0);
            g = Temperatures[temp_index][1]*(brightness/255.0);
            b = Temperatures[temp_index][2]*(brightness/255.0);
            glow();
        }

                void update_with_const_brightness(int level){  //Updates the
brightness according to user input
                        brightness_function_index = 1;
                      brightness =
(*Brightness_Functions[brightness_function_index])(level);
                      r = Temperatures[temp_index][0]*(brightness/255.0);
            g = Temperatures[temp_index][1]*(brightness/255.0);
            b = Temperatures[temp_index][2]*(brightness/255.0);
            glow();
            }
        char Hotter(){  //Increases light temperature
            if(temp_index < 10){
                temp_index++;
            }
            update();
                        return '0';
        }

        char Cooler(){  //Decreases light temperature
            if(temp_index > 0){
                temp_index--;
            }
        update();
                    return '0';
        }
};

char data[5] = {'1', '1','0', '0', '7'};  //array to store transmitted information
const int data_size = 5;
LED Light;


void setup(){
    Serial.begin(9600);
    Wire.begin();
    lightInput.begin(BH1750::ONE_TIME_HIGH_RES_MODE);
    pinMode(3, OUTPUT);
    pinMode(5, OUTPUT);
    pinMode(9, OUTPUT);
    pinMode(10, OUTPUT);
    pinMode(11, OUTPUT);

    Light.brightness = DEFAULT_BRIGHTNESS;
    LUX = lightInput.readLightLevel();
    Light.glow();
}

void loop(){
    lightInput.configure(BH1750::ONE_TIME_HIGH_RES_MODE);
    LUX = lightInput.readLightLevel();
    if(Serial.available())
      Serial.readBytes(data, data_size);
    if(data[0]=='1'){
      Light.glow();
      if(data[1]=='0')
        Light.update_with_const_brightness(data[4]-'0');
      if(data[1]=='1')
        Light.update();
      if(data[3]=='1')
        data[3] = Light.Hotter();
      if(data[3]=='2')
        data[3] = Light.Cooler();
    }
    else if(data[0]=='0')
      Light.dim();
    Serial.println(100-Light.brightness/2.55);
    delay(500);
}
```

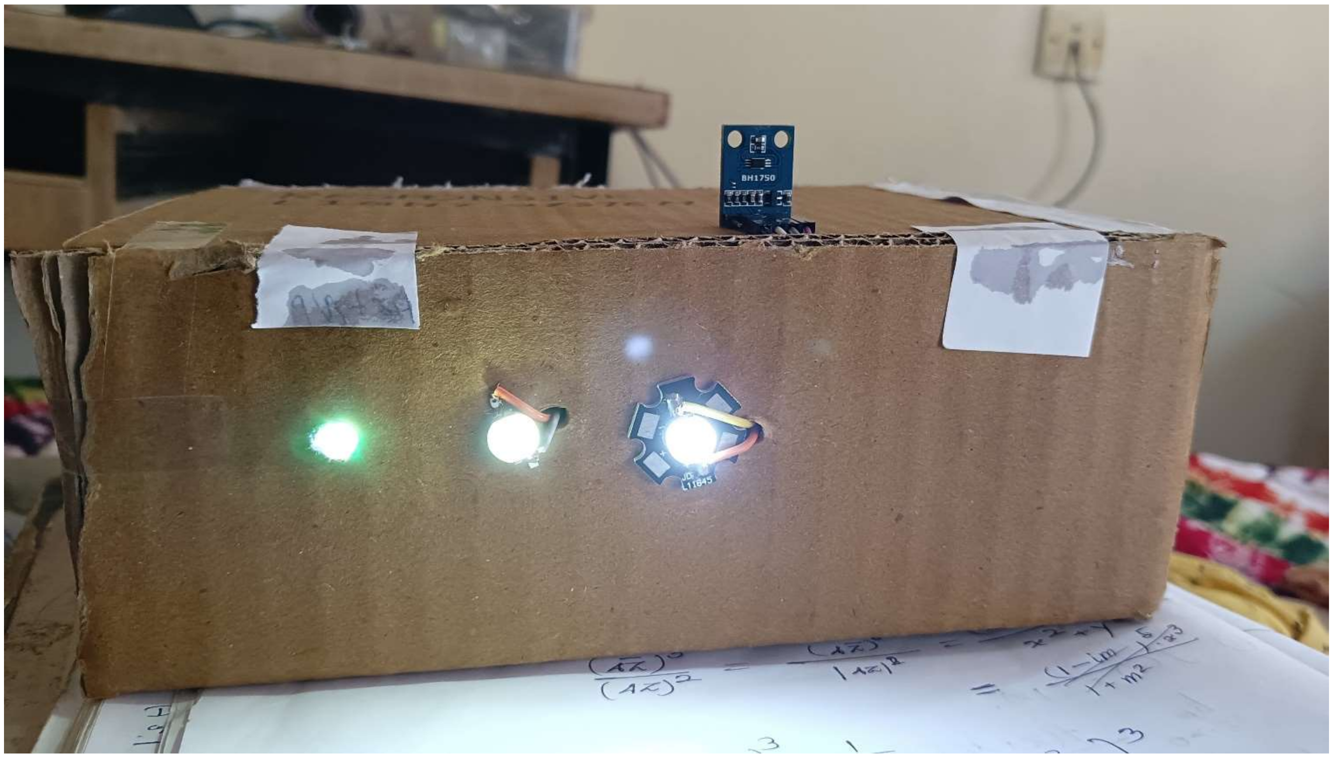
APPENDIX 2: APPLICATION BLOCK CODE



```
initialize global data to "1111"
initialize global brightness to "5"
initialize global p_save to 50
initialize global temp_i to 3

initialize global temp to make a list
    make a list 200 245 254
    make a list 240 255 255
    make a list 255 255 255
    make a list 255 255 200
    make a list 248 255 183
    make a list 255 248 167
    make a list 255 234 144
    make a list 255 218 122
    make a list 255 183 78
    make a list 255 139 39
    make a list 255 89 15

to send
do  call BluetoothClient1.SendText
        text  get global data
    call BluetoothClient1.SendText
        text  get global brightness

when ListPicker1.BeforePicking
do  set ListPicker1.Elements to BluetoothClient1.AddressesAndNames

when ListPicker1.AfterPicking
do  set ListPicker1.Selection to call BluetoothClient1.Connect
                                     address  ListPicker1.Selection
    set ListPicker1.Text to "SmartLight Connected"

when Button3.Click
do  set global data to "1000"
    call send

when Button5.Click
do  set global data to "1102"
    call send
    set global temp_i to get global temp_i - 1

when any Slider.PositionChanged
    component  notAlreadyHandled  thumbPosition
do  if  compare texts  segment text  get global data
                                start  2
                                length 1
                       =  "1"
    then  call Notifier1.ShowMessageDialog
              message    "PLEASE SWITCH TO CONSTANT BRIGHTNESS MODE FIRST"
              title      "ERROR: AUTO BRIGHTNESS"
              buttonText "OK"
    else  set global brightness to floor get thumbPosition
          call send

when Clock1.Timer
do  if  BluetoothClient1.IsConnected
    then  set global p_save to call BluetoothClient1.ReceiveText
                                   numberOfBytes  call BluetoothClient1.BytesAvailableToReceive
          set Label5.Text to join  get global p_save
                                   "%"
          set Label5.BackgroundColor to make color  select list item  list   get global temp
                                                                      index  get global temp_i

when Switch1.Changed
do  if  not Switch1.On
    then  set global data to "0000"
          set Switch1.Text to "LIGHT OFF"
    else  set global data to "1100"
          set Switch1.Text to "LIGHT ON"
    call send

when Button4.Click
do  set global data to "1100"
    call send

when Button7.Click
do  set global data to "1101"
    call send
    set global temp_i to get global temp_i + 1
```

APPENDIX 3: CIRCUIT SCHEMATIC




BH1750