Device emulating real time peak wavelength of sunlight.

Report

By DIY Team 7 “CAD buddies”:

Parag Karanjekar

Sourodeep Datta

Dhanshri Gawale

Bhoumik Mhatre

Akshat Dilip Lade

Abstract:

To create a device that establishes a user-friendly environment with respect to the light of the room, so as to obtain optimistic wavelength of light according to the time of the day by using the research data for the respective user to obtain mental peace and create an environment that is closest to the natural light. To also allow the user to control the mode of light in order to be able to focus on specific tasks or relax. Useful in the modern day due to many people spending most of their time indoors during the day.

Contents

Introduction:

In the modern era it has become more difficult for people to spend time outdoor in the sunlight. This has resulted in our circadian rhythm (an internal body clock) being disrupted. Two effects of light have been interrogated extensively in human circadian and sleep research:

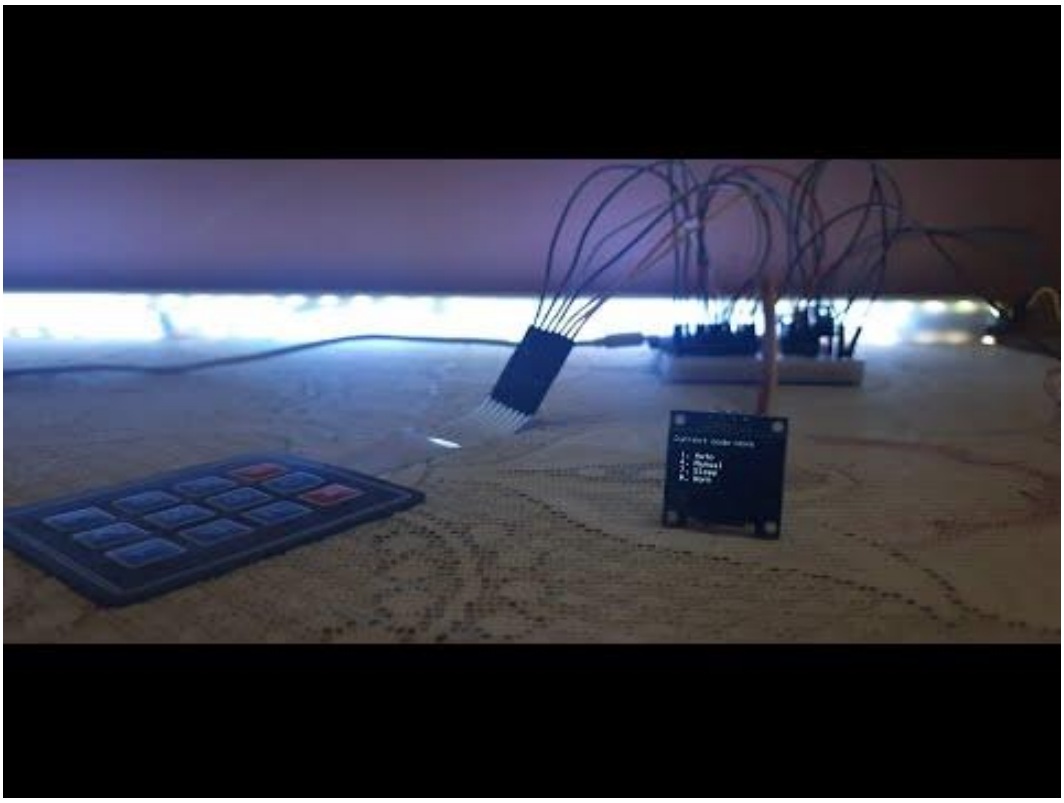
1. The acute suppression of melatonin in response to light exposure.
2. The ability of light exposure to shift circadian phase [1].

Both melatonin suppression and circadian phase shifts are modulated by the "photic history", i.e., the amount of light seen during the day [2]. Light can affect mood in several ways: by directly modulating the availability of neurotransmitters such as serotonin, which is involved in mood regulation, and by entraining and stabilising circadian rhythms, thereby addressing circadian desynchronisation and sleep disorders [1]. Because of the importance of light, specifically sunlight, in the circadian rhythm, we have built a device that emulates the peak wavelength of sunlight at any point during the day.

Firstly, we accumulated research data containing wavelengths of light that promoted required responses i.e., either promoting work or relaxation. Along with this we accumulated data related to sunlight and its variation throughout the day. This data was used in our calculations and then used to create four operating modes for the device, which are:

- Auto (Default)
- Manual
- Work
- Sleep

Video Demonstration:



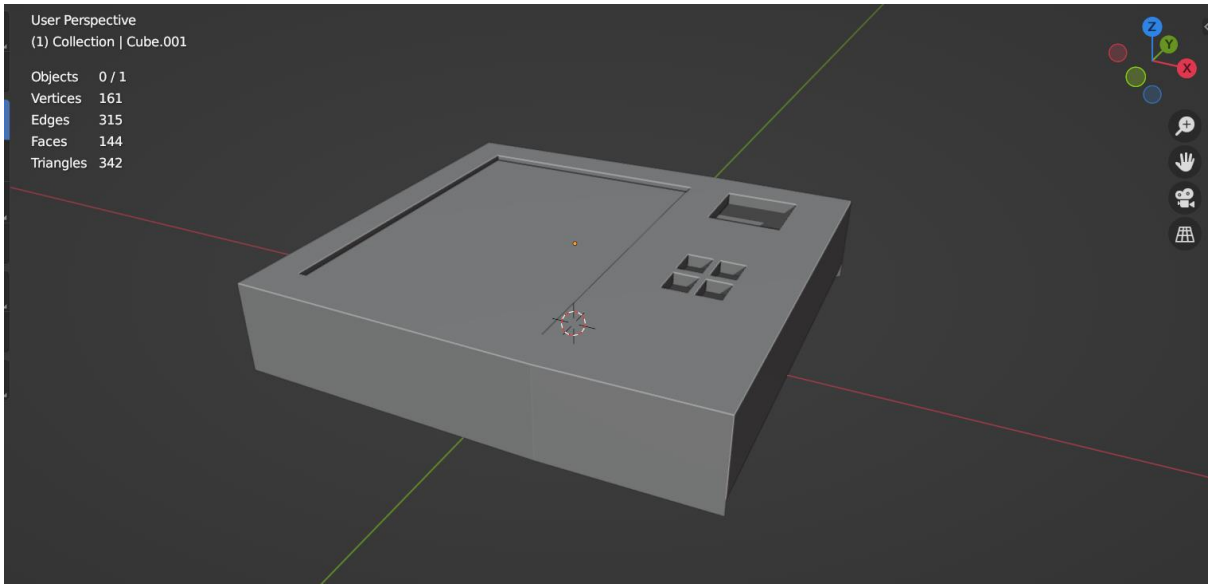
Materials and Cost:

Products List	Cost
DC-005 DC Power Jack Female Adapter PCB Mount-2.1 × 5.5mm	₹8.50
Male to Male Jumper Wires (20cm) – 40 pieces pack	₹68.00
12V 2A DC Power Supply Adapter	₹132.00
NODEMCU-ESP8266 WiFi Development Board	₹218.00
4×3 Matrix Membrane Type Keypad -12 Keys	₹75.00
TIP120 NPN Power Darlington Transistor 60V 5A TO-220 Package	₹72.00
Tactile Push Button Switch 6×6×5 -Pack of 10	₹48.00
Breadboard 400 Points for Solderless Prototyping	₹49.00
12V RGB LED Strip -4.5meter SMD 5050	₹299.00
Total Cost	**₹1300 (Including all extra charges)**

The cost of our project is significantly cheaper than similar devices available on the market, while providing much more functionality.

Scrapped Ideas:

Control Box:



3D Model

Dimensions: 98.2 * 80.9 * 20 mm.

Printing time duration: 7 hr 30 min.

This 3D model is made using blender software. It was supposed to be used as manual control box. It has different sections which can fit our display screen, keypad etc. and it has space inside it for the NodeMCU and other wire connections.

It was scrapped due to high 3D printing cost and time requirement.

Research Data:

Data Points	Mode
6000K [3]	Work
Dim Red Light (1600K) [4]	Sleep
1900K (Sunrise) - 6600K (Mid- Day) - 1900K (Sunset) [5]	Auto and Manual

Some info about these data points:

- White light is useful for working purposes so as to boost alertness. We know that the color of the light affects the circadian rhythms, of which blue light has the strongest impact. Exposure to blue light (and white light, which contains blue light) during sensitive periods can make it difficult for you to fall asleep. Exposure to white light during the day can have positive effects, including boosting alertness and mood.
- Creativity was better under warm light (3000 K) than under colder light (4500 K, 6000 K). Concentration was best under cold light (6000 K) [3].
- Red light has no effect on the circadian clock, so should dim red light should be used at night [4].

Calculations:

Calculation of parameters for Wavelength vs Time graph:

Based on the data collected from various research papers, we know that the range of colour temperatures of the sun throughout the day varies from:

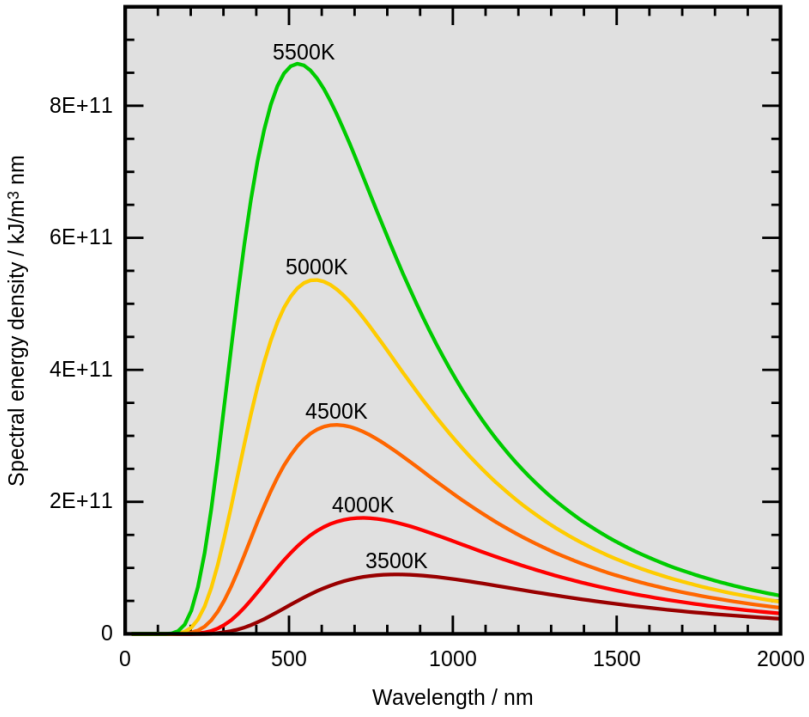
1900K(Sunrise) → 6600K(Mid-day) → 1900K(Sunset)

Usually colour temperatures are indicative of a range of wavelength of light, however given our current project is limited to a single wavelength at any given point, we will be using peak wavelength corresponding to the maximum intensity.

Using Wiens Law, we can calculate the peak wavelength corresponding to the colour temperature at that time.

Wien's Displacement Law:

Wien's displacement law states that the black-body radiation curve for different temperatures will peak at different wavelengths that are inversely proportional to the temperature.



$$\lambda_{peak} = \frac{b}{T}$$

Here λ_{peak} is peak wavelength, T is absolute temperature in Kelvin and b is constant of proportionality called *Wien's displacement constant*, equal to 2.897719… × 10^{-3} m·K.

Using this Law, we can calculate the endpoint values of the wavelength distribution which are:

1900K: 1525.2632 nm

6600K: 439.0909 nm

The peak wavelength corresponding to 1900K lies beyond the visible spectrum, so we take our max value to corresponding to 1900K as 750nm. Thus, our range is wavelengths(nm) is $(439.0909, 750)$.

Due to lack of proper documentation and for sake of simplicity, we took this variation to be linear in nature. Thus, the final curve of Wavelength vs time came to be a set of two lines between sunrise and sunset. Along with this, usually we would begin to reduce the intensity to zero after sunset, however having a lamp that turns off after sunset defeats the purpose, so we kept the wavelength as a constant of 750nm for anytime before sunrise and after sunset.

Now in order to calculate the magnitude of the slope of the lines (which are $\pm$ accordingly), we use $\frac{750-439.0909}{Midday\ time-Sunrise\ TIme}$. Midday time can be approximated to be $\frac{Sunrise\ Time+Sunset\ Time}{2}$. Thus, the final slope of the line is $\frac{621.8182}{Sunrise\ Time-Sunrise\ TIme}$.

Manual Mode:

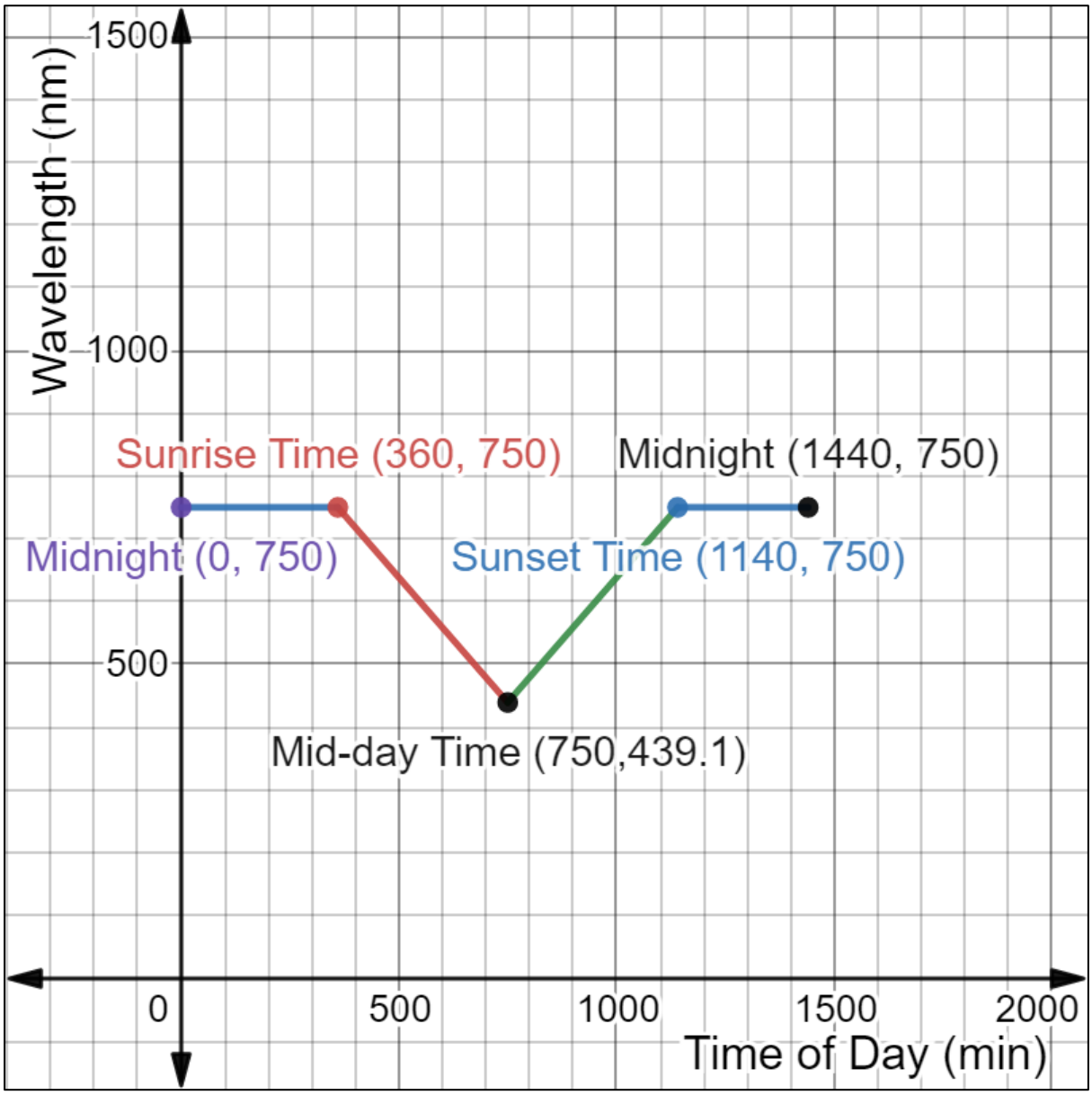
In manual mode the time for sunrise and sunset is pre-set and taken to be 6AM and 7PM respectively. **All the values we use in calculations are in terms of minutes from midnight**, so the values are 360 and 1140 respectively.

Using this we get slope of lines to be **0.79718**.

Therefore, the equation of each line is:

- $y = 750 - 0.79718 \times (x - 360)$
- $y = 750 - 0.79718 \times (1140 - x)$

Where y is wavelength (nm) and x is time of day (min, 24h).



Auto Mode:

In manual mode the time for sunrise and sunset is obtained from the internet and set accordingly.

All the values we use in calculations are in terms of minutes from midnight. The slope of the lines remains a variable and can be taken as m.

Therefore, the equation of each line is:

- $y = 750 - m \times (x - sunrise\_time)$
- $y = 750 - m \times (sunset\_time - x)$

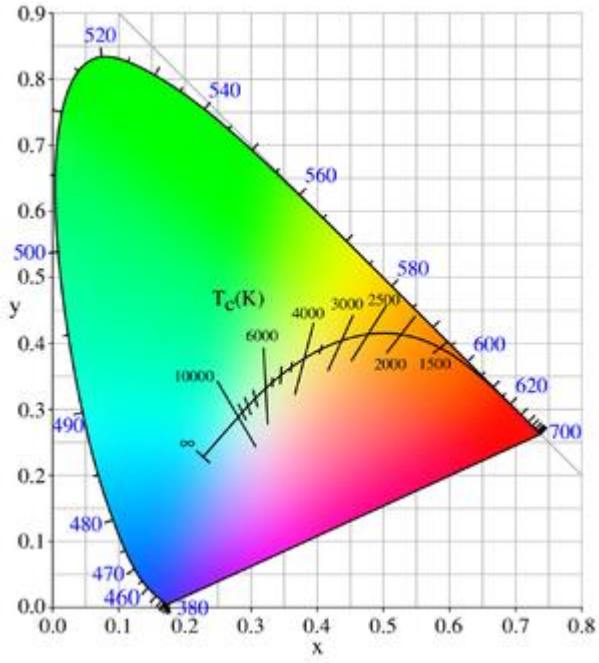
Where y is wavelength (nm) and x is time of day (min, 24h).

Graph for these lines is similar to that of manual mode, however the Sunrise, Sunset and Mid-day Times are variable.

Conversion of colour temperature to 8-bit RGB form for the pre-set modes (Sleep and Work):

For the conversion of the temperature to the 8-bit RGB we used the Planckian locus or a blackbody locus. It is the path or locus that the colour of an incandescent black body would take in a particular chromaticity space as the blackbody temperature changes. It goes from deep red at low temperatures through orange, yellowish white, white, and finally bluish white at very high temperatures.

In the CIE XYZ colour space, the three coordinates defining a colour are given by X, Y, and Z. A colour space is a three-dimensional space that is, a colour is specified by a set of three numbers (the CIE coordinates X, Y, and Z, for example, or other values such as hue, colourfulness, and luminance) which specify the colour and brightness of a particular homogeneous visual stimulus.



Planckian locus in the CIE 1931 chromaticity diagram [6]

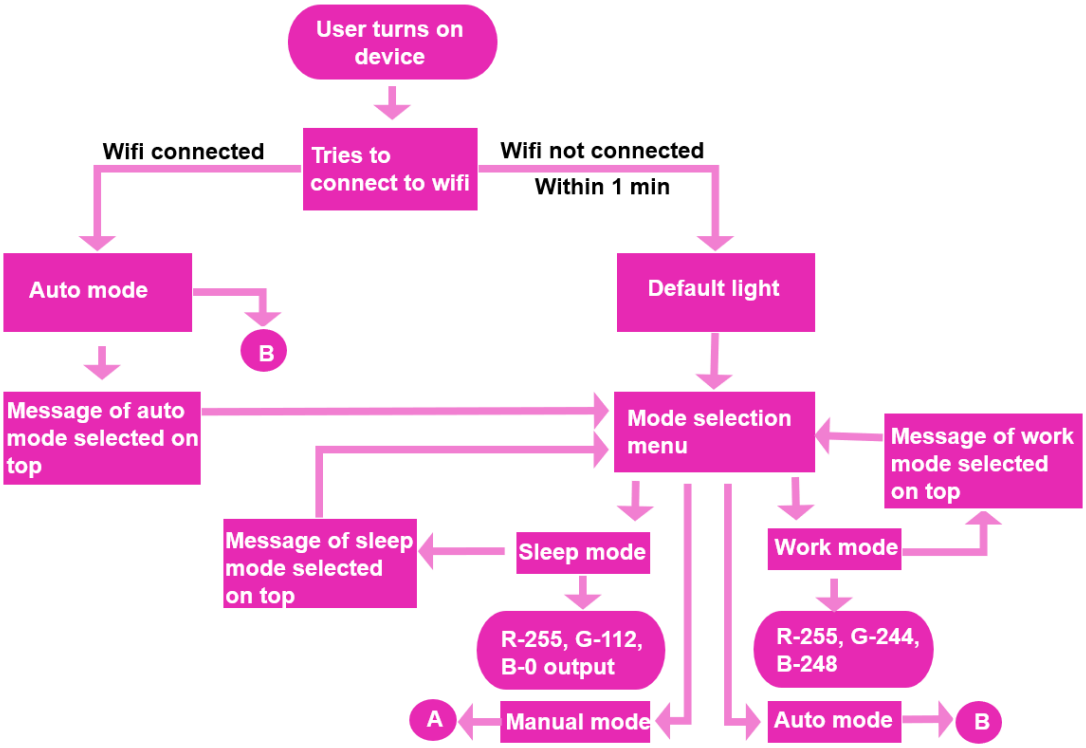
For the **sleep mode** which makes us feel relaxed under reddish color of low temperature light. temperature and 8-bit color are **1600K: (255, 112, 0)** [7]

For the **work mode** which helps to get better concentration and alertness due to colder light color. temperature and 8-bit color are **6000K: (255, 244, 248)** [7]

Flowcharts of Important Actions of Device:

User Interface:

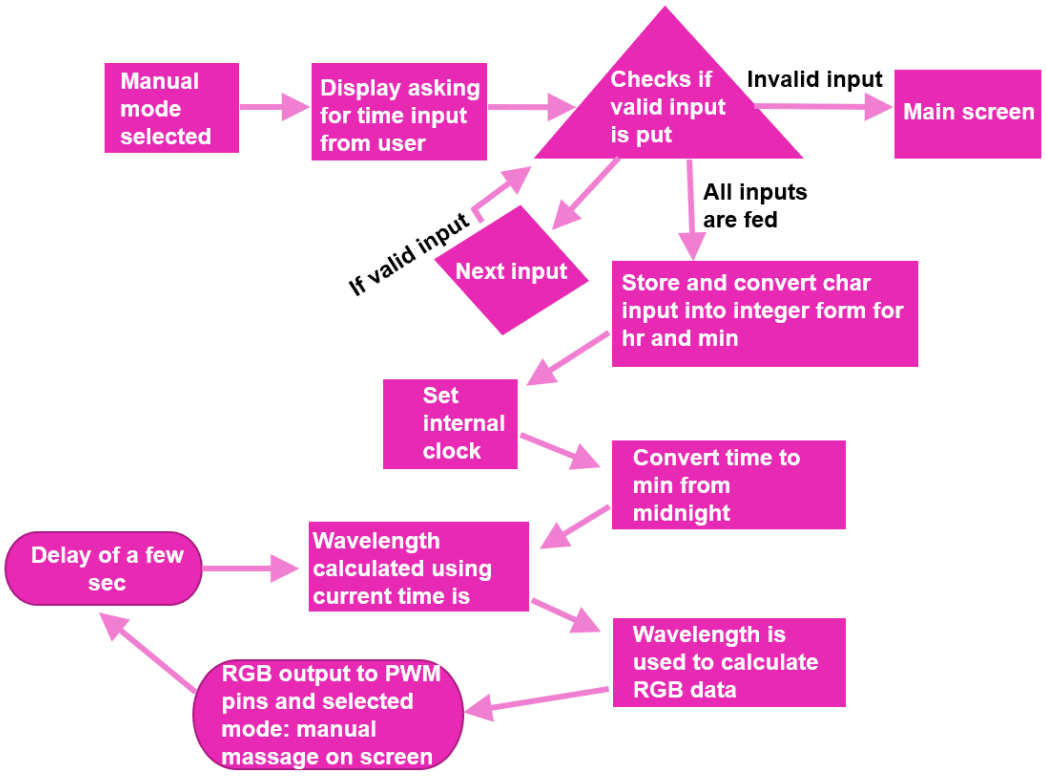
- User Interface



Explaining the process in a simple manner, at first when the device is powered and turned on, it shows the mode selection interface on the screen which has 4 modes to choose from: Manual, Auto, Sleep and Work. It defaults to Auto mode on startup and connects to the WiFi. If it is unable to connect to the WiFi in under a minute, it switches to a simple white light, and the user can either fix the WiFi or choose one of the 3 remaining options.

Manual Mode:

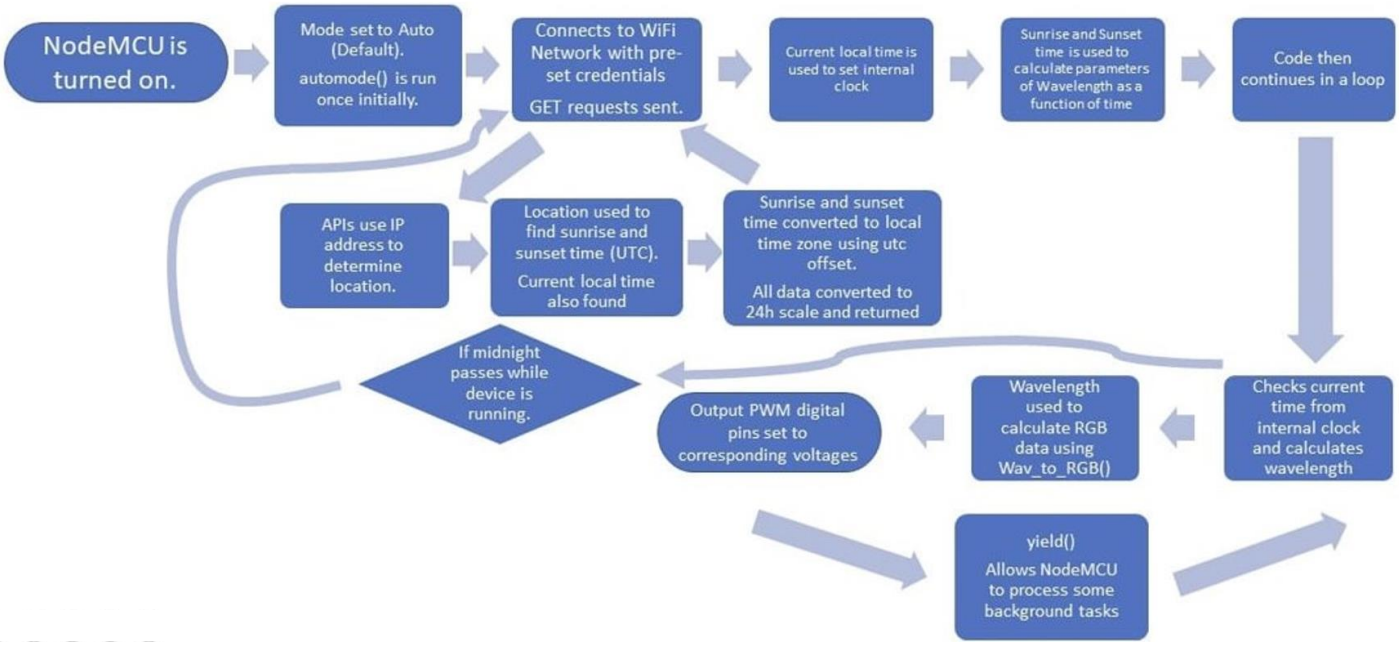
- Manual Mode (A)



On selecting the manual mode and entering the required timing in the 24-hour scale, it converts the valid input into minutes from midnight and then calculates the wavelength at that period of time. Wavelength is then used to calculate the RGB values which are used to set the PWM pins.

Auto Mode:

- Auto Mode (B)

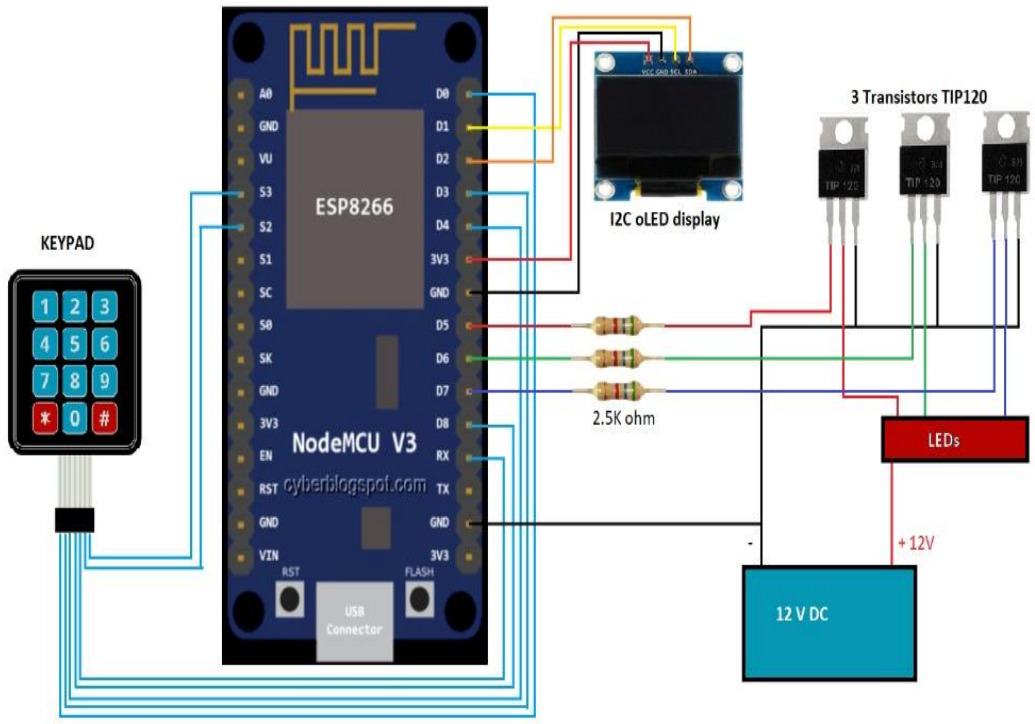


On selection the auto mode after the WiFi gets connected with pre-set credentials, after which sunrise, sunset and current time data is obtained. Calculations are done using this data to get wavelength at any instant and it is used to get RGB data which is outputted to the PWM pins.

Electronics:

Components used:

- Node MCU ESP 8266 Wi-Fi development board
- 4 x 3 matrix keypad
- I2C oLED display
- Transistors TIP 120 (3)
- 2.5k ohm resistors (3)
- 12V SMD 5050 RGB LED strip (270 LEDs)
- 12V 2A DC power adapter
- Jumper wires
- Breadboard
- DC Female power jack



Pin connections of Node MCU:

Keypad:

Column Pins of key pad are connected to:

- S2 (GPIO 9)
- S3 (GPIO 10)
- D0 (GPIO 16)

Row Pins of keypad are connected to:

- D3 (GPIO 0)
- D4 (GPIO 2)
- D8 (GPIO 15)
- RX (GPIO 3)

I2C oLED Display:

- VCC at 3V3
- GND at Ground (GND)
- SCL at D1 (Serial Clock)
- SDA at D2 (Serial Data)

RGB Strip analog data output from Node MCU

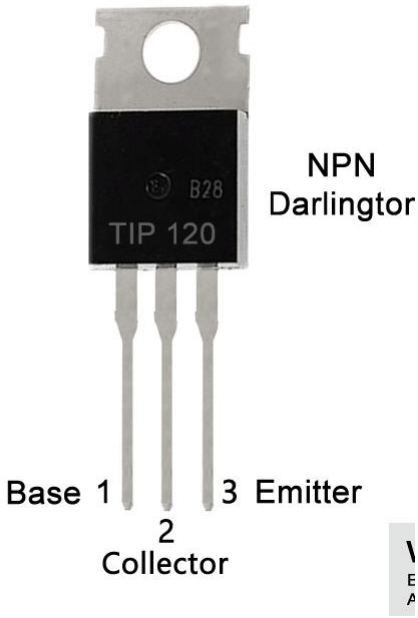
- Red D5 (GPIO 14)
- Green D6 (GPIO 12)
- Blue D7 (GPIO 13)

DC 12V input common ground has been connected to Node MCU ground.

Use of the TIP 120 transistor:

A transistor is a semiconductor device used to amplify or switch electrical signals and power. It is composed of semiconductor material, usually with at least three terminals for connection to an electronic circuit. Since the max voltage output from the Node MCU is 3.3V but we need 12V to power our RGB LED strip, therefore an external 12V 2A DC power source is used to power the RGB strip along with the TIP 120 transistor to amplify the analog signal from the Node MCU.

The base pin of transistor is connected to the Node MCU, collector pin is connected to the RGB strip and the emitter pin is connected to the common ground of the 12V DC power supply.



I2C oLED display:

It works on the I2C communication protocol. I2C only uses two wires to transmit data between devices:

SDA (Serial Data) – The line for the master and slave to send and receive data.

SCL (Serial Clock) – The line that carries the clock signal.

I2C is a serial communication protocol, so data is transferred bit by bit along a single wire (the SDA line).

Like SPI, I2C is synchronous, so the output of bits is synchronized to the sampling of bits by a clock signal shared between the master and the slave. The clock signal is always controlled by the master.

Code

1. Headers

```
#include <Arduino.h>
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>
#include <Keypad.h>
#include <TimeLib.h>
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
```

- Arduino.h: Default Arduino Library.
- ESP8266WiFi.h and ESP8266HTTPClient.h: Libraries for ESP8266 WiFi Module in NodeMCU.
- Keypad.h: Library for Keypad.
- TimeLib.h: Library by Paul Stoffregen. Gives Timekeeping functionality.
- SPI.h: Serial Peripheral Interface (SPI) is a synchronous serial data protocol used by microcontrollers for communicating with one or more peripheral devices.
- Wire.h: This library allows us to communicate with I2C / TWI devices. For display we are using the I2C protocol over the serial data and the serial clock lines.
- Adafruit_GFX.h: Contains all the functions required to display graphical output on an oLED or LCD display.
- Adafruit_SSD1306.h: It is a display driver used for oLED displays it defines the dimensions and the reset pin along with the address of the display.

2. Declarations:

```
#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 64 // OLED display height, in pixels
#define REDPIN 14        //GPIO pins on which RGB are connected
#define GREENPIN 12
#define BLUEPIN 13
#define D 5               //delay for rgb output
#define OLED_RESET     -1 // Reset pin # (-1 since sharing Node MCU reset pin)
#define SCREEN_ADDRESS 0x3c //Address of i2c display
```

Defining some constants such as Screen width, height and Pin names to correspond to specific values.

```
WiFiClient client;

const byte ROWS = 4;
const byte COLS = 3;
char Time[4];               //user time input (manual mode)
int oppMode = 0;    //Mode device is currently opperating in (0: Nothing, 1:
Manual, 2: Auto)
int min_day_span, min_mid_day;  //Dayspan time and mid day time in min (auto mode)
bool connectionFail = false;
bool incorrecKeyInput = false;
char key;                   //main keypad input variable for selecting modes
int rgb_manual[3];          //RGB values for manual output
int rgb_auto[3];            //RGB values for auto output
double colour_wav_slope, colour_wav_auto; //wavelength slope and value at time t
(auto mode)
double colour_wav_manual;
byte rowPins[ROWS] = {10, 9, 16, 0};        //pins on which keypad is connected
byte colPins[COLS] = {2, 15, 3};
char keys[ROWS][COLS] =
{
  {'1','2','3'},
  {'4','5','6'},
  {'7','8','9'},
  {'*','0','#'}
};

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);
Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );
```

Initializing a few variables, notable ones being:

- Array Time of size 4: Stores time input in terms of char for manual mode.
- Int oppMode: States the mode device is running in.
- min_day_span and min_mid_day: Stores time of day span and mid-day in terms of minutes.
- Bool connectionFail: Used when connecting to WiFi.
- Bool IncorrectKeyInput: Used when incorrect input is put in manual mode.
- Char key: Stores key input when changing modes.
- Int arrays rgb_manaual and rgb_auto: Store final rgb values for their respective modes.
- Double colour_wav_slope, colour_wav_auto and colour_wav_manual: Stores slope of wavelength vs time, and final wavelength in auto and manual modes.
- Adafruit_SSD1306 object called display: Used to control display with its member functions.

- Keypad object called keypad: Contains data about keys in keypad, used as reference.

```
class Time_of_Day                                                    //Declares class
containing time values
{
 public:
    int minutes;

    void set(int h, int m)
    {
      minutes = h*60 + m;
    }
};

Time_of_Day sunrise, sunset;                //Instances of Time_of_Day to store
time values.
```

- Class Time_of_Day which is used to store time of sunrise and sunset in terms of minutes. Has objects called sunrise and sunset.
- It has member function set() used to set time.

3. Setup function (Runs once at start of device)

```
void setup()
{
    Serial.begin(115200);
    pinMode(REDPIN, OUTPUT);
    pinMode(GREENPIN, OUTPUT);
    pinMode(BLUEPIN, OUTPUT);

    if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3c))    // SSD1306_SWITCHCAPVCC =
generate display voltage from 3.3V internally
    {
        Serial.println(F("SSD1306 allocation failed"));
        for(;;); // Don't proceed, loop forever
    }

    modesInterface_TextStyle();               //main screen
    display.setCursor(0,0);                   //coordinates of point from which
printing on display starts total pixels -> (128,64)
    display.println(F("DIY TEAM 7"));         //prints any variable or text
    display.display();                        //prints all the things in buffer
    modesInterface();

    oppMode = 2;                              //Defaults to automode at startup
    Serial.println("auto");
    automode();
}
```

- Sets baud rate (rate of information transfer) to 115200, which is the required value for ESP8266, in order to allow serial communication between the microcontroller and serial monitor.
- Sets 3 pins to output.
- Ensures display is working.
- Prints on display “DIY Team 7” followed by mode choosing interface.
- Sets operating mode to auto (2), which is default mode.
- Calls the initial setup function for automode.

4. Loop function (Runs in a loop in NodeMCU)

```
void loop()
{


    key = keypad.getKey();          //takes input from keypad

    if (key)                        //won't enter the if unless a key is pressed
    {
        if (key == '7')             //Sleep mode
        {
            String s = "Sleep";
            modergb(s);
            oppMode = 0;
        }
        if (key == '*')             //Work mode
        {
            String s = "Work";
            modergb(s);
            oppMode = 0;
        }
        if (key == '1')             //Auto mode
        {
            oppMode = 2;
            Serial.println("auto");
            automode();
        }
        if (key == '4')             //Manual mode
        {
            oppMode = 1;
            modesInterface_TextStyle();
            display.setCursor(0,0);
            display.println(F("Enter time in HH : MM"));
            display.display();
            usertimeinput();
            while (incorrecKeyInput == true) {
                   modesInterface_TextStyle();
                   display.setCursor(0,0);
                   display.println(F("Enter time in HH : MM"));
                   display.display();
                   usertimeinput();
            }
        }
    }
```

- Gets key pressed via keypad.getKey().
- Checks which device should run on according to value of key.
- If mode selected is either auto or manual, then an initial setup function is run. If mode is “Work” or “Sleep” then a one-time function is called to set LEDs to a permanent value. Also sets oppMode to corresponding mode.
- In case of manual mode, there is a while loop used in case the user inputted an invalid time.

```
  if (day() == 1 && oppMode == 2) {           //Check incase device passes mid-
night.
      delay(60000);
      automode();
  }
```

- In case device is running in auto mode and it passes through midnight (local time), then it has to run automode() function again to get the new sunrise and sunset times.

```
   if (oppMode == 1)           //If oppMode is in Manual
    {
        int Min = current_time();                          //time in minutes

        if ( Min <= 750 && Min >= 360 )               //gives wavelength accord-
ing to the time of day the sunrise(6am) and sunset(7pm) time for manual input are
perdefined
          {
              colour_wav_manual = 750 - 0.79718*(Min-360);
          }
        else if ( Min >= 750 && Min <= 1140 )
          {
              colour_wav_manual = 750 - 0.79718*(1140-Min);
          }
        else
          {
              colour_wav_manual = 750;
          }

        delay(1000);
        Wav_to_RGB(colour_wav_manual, rgb_manual);   //calls the wav to rgb func-
tion to get rgb data according to the wavelength
        Serial.println(colour_wav_manual);
        String s = "Manual";
        modergb(s);                                  //give required output on the
display and to the rgb pins
    }
```

- During each loop, if oppMode is 1 (Manual), it calculates the wavelength corresponding to the current time (using current_time()). This wavelength is used to get the RGB values (using Wav_to_RGB) which is then used to the LED pin outputs (using modergb()).

```
    else if (oppMode == 2)
    {
        if ( current_time() <= min_mid_day && current_time() >= sunrise.minutes )
//Calculate wavelength as at a particular time
          {
              colour_wav_auto = 750 - colour_wav_slope*(current_time()-sun-
rise.minutes);
          }
        else if ( current_time() >= min_mid_day && current_time() <= sun-
set.minutes )
          {
              colour_wav_auto = 750 - colour_wav_slope*(sunset.minutes-cur-
rent_time());
          }
        else
          {
              colour_wav_auto = 750;
          }
        delay(1000);
        Wav_to_RGB(colour_wav_auto, rgb_auto);
//Call function to get RGB value from wavelength
        Serial.println(colour_wav_auto);
        String s = "Auto";
        modergb(s);                                  //give required output on the
display and to the rgb pins
    }
    yield();
}
```

- During each loop, if oppMode is 2 (Auto), it calculates the wavelength corresponding to the current time (using current_time()). This wavelength is used to get the RGB values (using Wav_to_RGB) which is then used to the LED pin outputs (using modergb()).
- yield() stops the code temporarily to allow the NodeMCU to perform some background tasks.

5. Important Functions

A) EnableWiFi

```
void EnableWiFi()                                          // Connects to WiFi
{
  WiFi.mode(WIFI_STA);
  WiFi.begin("SSID", "password");                    //Enter ssid, password as
strings)
  Serial.print("Connecting to WiFi");
  int wifi_count = 0;                                      //Check iterations
of wifi connecting loop
  while(WiFi.status() != WL_CONNECTED)
  {
    Serial.print('.');
    delay(200);
    wifi_count++;
    if (wifi_count == 300) {                               //Incase connecting
takes too long
        Serial.println("Connection Failed");
        connectionFail = true;
        yield();
        return;
    }
  }

  Serial.print("IP Address =");
  Serial.print(WiFi.localIP());                            //Prints IP Adress
  yield();                                                 //Yield
}
```

- Sets device as a station and then connects to WiFi using SSID and password.
- While loop continues until WiFi is connected. In case it takes more than 60 sec, it stops and returns.

B) GetData

Returns data for sunrise, sunset and current time.

```
void GetData(int arr1[2], int arr2[2], int arr3[2])            //Get arrays for
sunrise, sunset and current time in h,min from APIs
{
  HTTPClient http;                                             //Declares HTTP
clients
  HTTPClient http1;
  HTTPClient http2;
  delay(500);

  http.begin(client, "http://ip-api.com/json/");

  int httpCode = http.GET();
  if(httpCode < 0)
   {
    Serial.println("I don't feel so good, httpCode < 0");
    while(true);
  }
  String payload1 = http.getString();                          //String containing
lat and long data of location of device
  int lat_index = payload1.indexOf("lat");
  int lon_index = payload1.indexOf("lon");
  String lat = payload1.substring(lat_index+5, lat_index+12);
  String lon = payload1.substring(lon_index+5, lon_index+12);
  delay(500);
  http.end();
```

- First connects to http://ip-api.com/json/ to get latitude and longitude data of device from IP address.
  {"status":"success","country":"India","countryCode":"IN","region":"DL","regionName":"National Capital Territory of Delhi","city":"New Delhi","zip":"110001","lat":28.6328,"lon":77.2204,"timezone":"Asia/Kolkata","isp":"","org":"","as":"","query":""}
- lat and lon data is parsed by using the index of “lat” and “lon” within the payload and stored as strings.

```
http1.begin(client, "http://api.sunrise-sunset.org/json?lat="+lat+"&lng="+lon);

  int httpCode1 = http1.GET();
  if(httpCode < 0)
  {
    Serial.println("I don't feel so good, httpCode < 0");
    while(true);
  }
  String payload2 = http1.getString();                          //String containing
sunrise, sunset data
  int Sunrise_index = payload2.indexOf("sunrise");
  int Sunset_index = payload2.indexOf("sunset");
  int colon_index = payload2.indexOf(':', Sunrise_index+10);

  String Sunrise_time1 = payload2.substring(Sunrise_index+10, colon_index);
  String Sunrise_time2 = payload2.substring(colon_index+1, colon_index+3);
  arr1[0] = Sunrise_time1.toInt(); arr1[1] = Sunrise_time2.toInt();
    if( payload2.substring(colon_index+7, colon_index+9) == "PM" && arr1[0] != 12)
//Convert PM to 24h data
    {
      arr1[0] += 12;
    }

  colon_index = payload2.indexOf(':', Sunset_index+9);
  String Sunset_time1 = payload2.substring(Sunset_index+9, colon_index);
  String Sunset_time2 = payload2.substring(colon_index+1, colon_index+3);
  arr2[0] = Sunset_time1.toInt(); arr2[1] = Sunset_time2.toInt();
   if( payload2.substring(colon_index+7, colon_index+9) == "PM"  && arr2[0] != 12)
//Convert PM to 24h data
    {
      arr2[0] += 12;
    }

  delay(500);
  http1.end();
```

- It then connects to https://api.sunrise-sunset.org/
  {"results":{"sunrise":"6:56:01 AM","sunset":"6:06:15 PM","solar_noon":"12:31:08 PM","day_length":"11:10:14","civil_twilight_begin":"6:31:12 AM","civil_twilight_end":"6:31:04 PM","nautical_twilight_begin":"6:01:06 AM","nautical_twilight_end":"7:01:10 PM","astronomical_twilight_begin":"5:31:08 AM","astronomical_twilight_end":"7:31:08 PM"},"status":"OK"}
- Uses previously obtained lat and lon data to append the url and get the data for the correct location.
- sunrise and sunset data is parsed by using the index of “sunrise”, “sunset” and “:” within the payload and stored as strings.
- These strings are converted to integers by using the String.toInt() member function.
- These integers are then converted into 24h scale.

```
http2.begin(client, "http://worldtimeapi.org/api/ip");

  httpCode = http2.GET();
  if(httpCode < 0)
  {
    Serial.println("I don't feel so good, httpCode < 0");
    while(true);
  }
  String payload3 = http2.getString();
//Contains IST current time and utc offset
  int Time_index = payload3.indexOf("datetime");
  int utc_offset_index = payload3.indexOf("utc_offset");
  String New_time1 = payload3.substring(Time_index+22, Time_index+24);
  String New_time2 = payload3.substring(Time_index+25, Time_index+27);
  arr3[0] = New_time1.toInt(); arr3[1] = New_time2.toInt();

  String utc_offset_h = payload3.substring(utc_offset_index+14, utc_offset_in-
dex+16);
  String utc_offset_m = payload3.substring(utc_offset_index+17, utc_offset_in-
dex+19);
  int utc_h = utc_offset_h.toInt();
  int utc_m = utc_offset_m.toInt();

  delay(500);
  http2.end();


  arr1[0] += utc_h; arr1[1] += utc_m;
  arr2[0] += utc_h; arr2[1] += utc_m;
  if( arr1[1] >= 60)
//Checks if min value is greater than 60
  {
    arr1[0]++; arr1[1] -= 60;
  }
  delay(250);
  if( arr2[1] >= 60)
//Checks if min value is greater than 60
  {
    arr2[0]++; arr2[1] -= 60;
  }
  delay(250);
  if( arr1[0] >= 24)
//Checks if hour values is greater than 24
  {
    arr1[0] -= 24;
  }
  delay(250);
  if( arr2[0] >= 24)
//Checks if hour values is greater than 24
  {
    arr2[0] -= 24;
  }

  delay(500);
  yield();
//Yield
}
```

- Finally connects to http://worldtimeapi.org/api/ip to get current local time along with utc offset of the region.
  {"abbreviation":"IST","client_ip":"180.151.18.120","datetime":"2022-02-22T16:48:55.918086+05:30","day_of_week":2,"day_of_year":53,"dst":false,"dst_from":null,"dst_offset":0,"dst_until":null,"raw_offset":19800,"timezone":"Asia/Kolkata","unixtime":1645528735,"utc_datetime":"2022-02-22T11:18:55.918086+00:00","utc_offset":"+05:30","week_number":8}
- Parses this to get current time and utc offset, which is used to convert sunrise, sunset times to local time zone format.

- Once converted to the local time zone, data is checked to ensure it is in the proper format (hours cannot be greater than 24, min cannot be greater than 60). Finally returns the obtained data.

C) Wav_to_RGB

```
void Wav_to_RGB (double wavelength, int arr[3])                         //Takes wavelength, pointer
to an array. Converts wavelength to RGB values and edits array to that. Source : Prof Dan Bruton,
http://www.physics.sfasu.edu/astro/color/spectra.html
{                                                                       // Original code written in
FORTRAN, last updated 20Feb 1996
    double gamma = 0.8;                                                 //Input wavelength b/w 380
and 750nm, visible spectrum of light
    double R, G, B, attenuation;

    if (wavelength >= 380 && wavelength <= 440)
    {
        attenuation = 0.3 + 0.7 * (wavelength - 380) / 60.0;
        R = pow(((-(wavelength - 440) / 60.0 ) * attenuation), gamma);
        G = 0.0;
        B =pow((1.0 * attenuation), gamma);
    }
    else if (wavelength >= 440 && wavelength <= 490)
    {
        R = 0.0;
        G = pow(((wavelength - 440) / 50.0 ), gamma);
        B = 1.0;
    }
    else if (wavelength >= 490 && wavelength <= 510)
    {
        R = 0.0;
        G = 1.0;
        B = pow((-(wavelength - 510) / 20.0 ), gamma);
    }
    else if (wavelength >= 510 && wavelength <= 580)
    {
        R = pow(((wavelength - 510) / 70.0 ), gamma);
        G = 1.0;
        B = 0.0;
    }
    else if (wavelength >= 580 && wavelength <= 645)
    {
        R = 1.0;
        G = pow((-(wavelength - 645) / 65.0 ), gamma);
        B = 0.0;
    }
    else if (wavelength >= 645 && wavelength <= 750)
    {
        attenuation = 0.3 + 0.7 * (750 - wavelength) / 105.0 ;
        R = pow((1.0 * attenuation), gamma);
        G = 0.0;
        B = 0.0;
    }
    else
    {
        R = 0.0;
        G = 0.0;
        B = 0.0;
    }

    int R1 = R * 255; //Gives RGB values in range of 0-255
    if ( R1 > 255)    //Checks if RGB value is greater than 255
    R1 = 255;
    int G1 = G * 255;
    if ( G1 > 255)
    G1 = 255;
    int b1 = B * 255; //B1 is some pre-existing arduino object, used b1 instead
    if ( b1 > 255)
    b1 = 255;
    arr[0]=R1; arr[1]=G1; arr[2]=b1;
    delay(1000);
    yield();          //Yield
}
```

- Convert wavelength (nm, in visible spectrum) to RGB values.
- Original code written by Dan Bruton in Fortran [8].

D) modesInterface_TextStyle

```
void modesInterface_TextStyle()   //sets text style to size 1 and clears the ex-
isting display output along with setting cursor at (0,0)
{
    display.clearDisplay();
    display.display();
    display.setTextSize(1);             // Normal 1:1 pixel scale
    display.setTextColor(SSD1306_WHITE);
    display.setCursor(0,0);
}
```

- This function sets the text format for the display text to size 1, white colour and sets the initial printing coordinate to (0,0) as well as clears the existing output on the display.

E) modesInterface

```
void modesInterface()  //displays the main modes interface, won't work without
modesInterface_TextStyle both functions are seprate since it allows to print any
other message than main interface
{
    display.setCursor(10,10);
    display.println(F("# to turnoff"));
    display.setCursor(10,20);
    display.println(F("1. Auto"));
    display.setCursor(10,30);
    display.println(F("4. Manual"));
    display.setCursor(10,40);
    display.println(F("7. Sleep"));
    display.setCursor(10,50);
    display.println(F("*. Work"));
    display.display();
    display.display();
}
```

- This function displays the user interface on different modes as well as the main screen.
- The dispay.setCursor is a library function that input of coordinates and from these coordinates the printing on the screen starts.
- Display.println prints the input to the function on new line, here ‘ln’ can be removed so that output is on the same line.
- Display.display prints all the print statements above it which were stored in buffer.

F) usertimeinput

```
void usertimeinput()
{
    int a, b, c, d;
    char keytime = '\0';
    for (int data_count=0; data_count <= 3; data_count++)   //loop to get time input from
user
    {
        while (keytime == '\0')    //won't proceed ahead unless a input from keypad is
given within 4 second for each input, if while was not present user won't get any time to
give input
        {
            keytime = keypad.getKey();
        }

        Time[data_count] = keytime;   //stores time input in array

        display.setTextSize(2);             // Draw 2X-scale text
        display.setTextColor(SSD1306_WHITE);
        display.setCursor(data_count*20+20,27);    //cursor position according to index of
array
        display.print(Time[data_count]);
        if(data_count == 1)
        {
          display.setCursor(50,27);
          display.println(':');
        }
        display.display();
        keytime = '\0';
    }

    String temp12 = String(Time[0]);
    a = temp12.toInt();         //converts the char input from the keypad to int form
    temp12 = String(Time[1]);
    b = temp12.toInt();
    temp12 = String(Time[2]);
    c = temp12.toInt();
    temp12 = String(Time[3]);
    d = temp12.toInt();

    int manual_hr = a*10+b;
    int manual_min = c*10+d;

    if (manual_hr > 24 || manual_min > 59) {
        incorrecKeyInput = true;
        entervalidtime();
        yield();
        return;
    } else {
      incorrecKeyInput = false;
    }
    delay(500);
    display.clearDisplay();
    modesInterface_TextStyle();
    modesInterface();
    display.setCursor(20,0);
    display.println(F("Time set!"));    //if complete input is given by user displays time
set message
    display.display();
    display.clearDisplay();

    setTime(manual_hr, manual_min, 0, 0, 0, 0);    //sets clock according to input time
    yield();                                       //Yield
}
```

- In manual mode, gets user input using keypad, converts into integers and checks for valid input.
- A while loop is present in this function that would make sure that code does not proceeds with null value in the keypad input that is stored in 'keytime' variable. 'keypad.getKey()' is a library function of keypad.h library that takes input from user.
- Along with taking the input this function displays the input on screen. To determine the coordinates of each digit on the screen a simple formula is used (data_count * 20 + 20, 27), this sets the coordinates of the digits at (20,27), (40,27) etc. This is required to prevent overlapping of the digits on the display.

- It then sets internal time of device to given input.

G) automode

```
void automode()                                 //Sets time and calculates wavelength
parameters for auto mode
{
    modesInterface_TextStyle();
    display.println(F("Selected mode Auto"));
    modesInterface();

    EnableWiFi();                               //Enables Wifi
    if (connectionFail == true) {
        oppMode = 0;
        String s = "Fail";
        modergb(s);
        connectionFail = false;
        return;
    }
    int a[2], b[2], c[2];                       //Arrays for h, min. a: Sunrise, b:
Sunset, c: Current time IST)
    GetData(a, b, c);                           //Gives arrays h and min values.
    sunrise.set(a[0], a[1]); sunset.set(b[0], b[1]);
    min_day_span = sunset.minutes - sunrise.minutes;
    min_mid_day = (sunset.minutes+sunrise.minutes)/2;
    colour_wav_slope = 621.80/min_day_span;
    setTime(c[0], c[1], 0, 0, 0, 0);          //Set time
}
```

- Starting function for Auto Mode.
- Connects to Internet, gets data and uses it to set internal clock and find parameters which are used to calculate wavelength as a function of time.
- If connection fails, then it returns and light is set to default of white.

H) modergb

```
void modergb(String c)                //displays the interface in a particular mode along with giving appropriate analog output for RGB input string is the name of mode
{
    int r,g,b;                        //RGB output values
    display.clearDisplay();
    modesInterface_TextStyle();
    display.print(F("Current mode:"));
    display.println(c);
    modesInterface();
    if (c == "Sleep")
    {
          r = 255, g = 112, b = 0;
    }
    if (c == "Work")
    {
          r = 255, g = 244, b = 248;
    }
    if (c == "Manual")
    {
          r = rgb_manual[0], g = rgb_manual[1], b = rgb_manual[2];
    }
    if (c == "Turnoff")
    {
          r = 0, g = 0, b = 0;
    }
    if (c == "Auto")
    {
          r = rgb_auto[0], g = rgb_auto[1], b = rgb_auto[2];
    }
    if (c == "Fail")
    {
          r = 255, g = 255, b = 255;
    }
    analogWrite(REDPIN, r);
    analogWrite(GREENPIN, g);
    analogWrite(BLUEPIN, b);
    Serial.println(r);
    Serial.println(g);
    Serial.println(b);
    delay(D);
}
```

- Sets RGB values according to the selected mode and calculated values (for manual and auto mode).
- Uses this to set the PWM pins accordingly.

6. Modes:

1. Sleep (oppMode = 0)

Sets device to best wavelength for encouraging relaxation/sleep.

2. Work (oppMode = 0)

Sets device to best wavelength for encouraging productivity.

3. Manual (oppMode = 1)

Makes user input current time, sets internal clock and gives corresponding wavelength.

4. Auto (oppMode = 2)

Connects to internet, gets sunrise, sunset and current time, sets internal clock, and calculates wavelength according to that time.

Bibliography

[1] G. C. S. M. Blume C, 20 August 2019. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6751071/.

[2] S. M. F. R. A. M. P. S. Webler FS, 30 December 2019. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6701986/.

[3] B. H. L. A. J. S. Weitbrecht WU, 22 June 2015. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/26098084/.

[4] [Online]. Available: https://www.cdc.gov/niosh/emres/longhourstraining/color.html.

[5] J. D. X. H. C. P. Y. C. L. L. Q. Z. X. X. H. &. W. X. Lin, 17 May 2019. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6525242/.

[6] [Online]. Available: http://hyperphysics.phy-astr.gsu.edu/hbase/vision/cie.html#c2.

[7] [Online]. Available: https://www.wolframalpha.com/widgets/view.jsp?id=5072e9b72faacd73c9a4e4cb36ad08d.

[8] D. Bruton. [Online]. Available: http://www.physics.sfasu.edu/astro/color/spectra.html.

[9] B. P. Myers BL, July 1993. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/8327605/.

[10] T. H. Morita T, 17 May 1998. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/9682518/.

[11] T. H. Morita T, 15 September 1996. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/8979406/.