

ECPS 216 | IOT Systems & Software

ASSIGNMENT 1

Light Up Your Life

Student: Lan Hoang Pham 48326518

Instructor: Prof. Quoc-Viet Dang

Professional Master of Embedded and Cyber-Physical Systems

University of California, Irvine

Oct. 10th, 2024

TABLE OF CONTENTS

LIST OF FIGURES AND TABLES	2
EQUIPMENT LIST	3
PART 1	4
Requirements	4
Wiring Schematic	4
Code Explanation	4
PART 2	6
Requirements	6
Wiring Schematic	6
Code Explanation	6
PART 3	7
Requirements	7
Wiring Schematic	7
Code Explanation	8
OPTIONAL PART	8
Requirements	8
Wiring Schematic	9
Code Explanation	9

LIST OF FIGURES AND TABLES

Figure 1. Assembly of all equipment.....	3
Figure 2. ESP8266 12-E NodeMCU pinout diagram	3
Figure 3. Wiring schematic for Part 1	4
Figure 4. Wiring schematic for Part 2	6
Figure 5. Wiring schematic for Part 3.....	7
Figure 6. Wiring schematic for optional part.....	9
Table 1. Equipment list.....	3
Table 2. Wiring connections between TCS34725 RGB sensor and ESP8266	4
Table 3. Wiring connections between 3 external LEDs and ESP8266.....	6
Table 4. Wiring connections between I2C LCD, rotary encoder, push button and ESP8266	9

EQUIPMENT LIST

Items	Model	Quantity	Power Supply
NodeMCU ESP8266	ESP8266-12E	1	4.5 to 9 VDC
RGB Color Sensor	TCS34725	1	3.3 to 5 VDC
Light Emitting Diode	5mm Red/Yellow/Green LEDs	3	With resistors
Resistor	220 Ω	3	
Push Button	Bread-board Mount	1	
Liquid-Crystal Display	I2C 1602 Serial LCD Module	1	5VDC
Rotary Encoder	KY-040	1	5VDC

Table 1. Equipment list

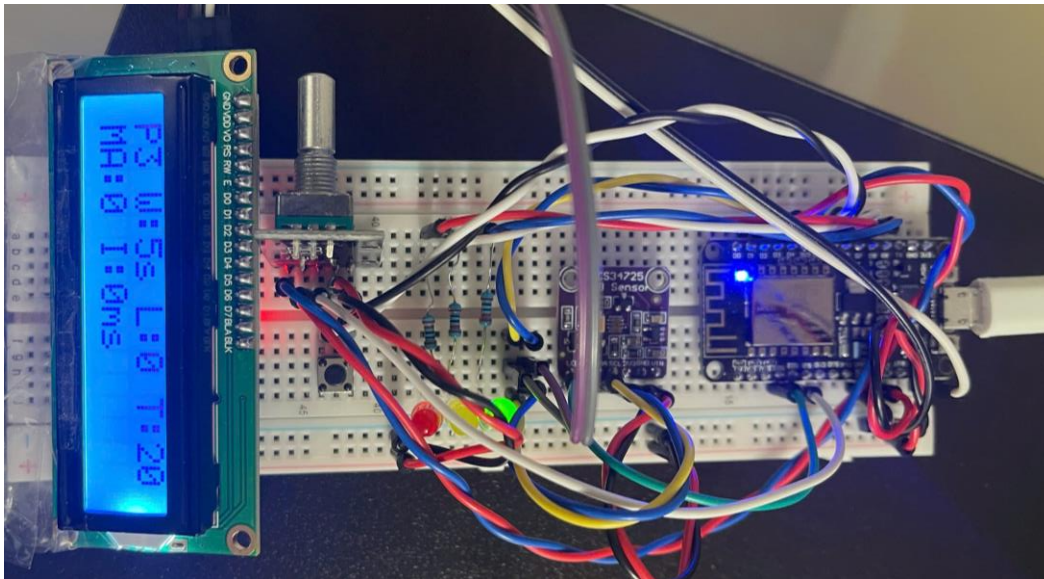


Figure 1. Assembly of all equipment

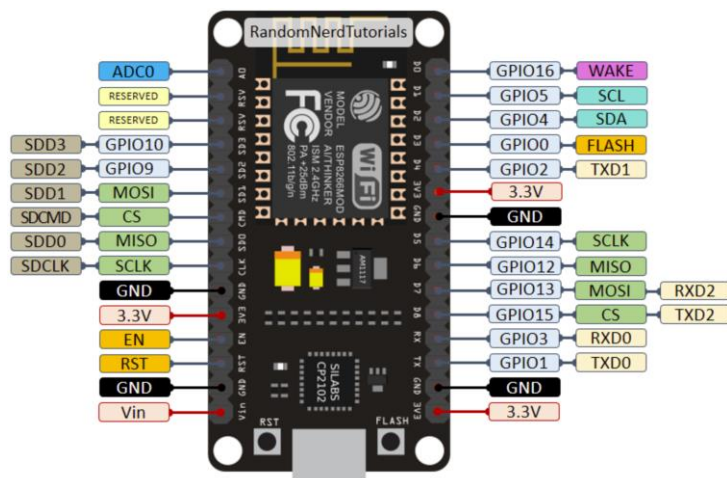


Figure 2. ESP8266 12-E NodeMCU pinout diagram

PART 1

Requirements

- Connect the RGB Light Sensor to ESP8266.
- Read sensor values once every second.
- Display the values on the IDE console window.
- Turn the onboard LED ON if the average value in a window of X seconds is higher than a threshold of Y.

Wiring Schematic

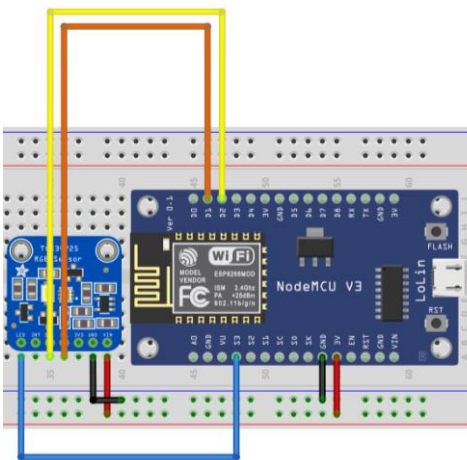


Figure 3. Wiring schematic for Part 1

RGB Sensor	ESP8266
GND	GND
Vin	3.3V
SCL	D1 GPIO5 SCL
SDA	D2 GPIO4 SDA
LED	GPIO10

Table 2. Wiring connections between TCS34725 RGB sensor and ESP8266

Code Explanation

- Include necessary libraries and define connected pins.

```

1  #include "Adafruit_TCS34725.h"
2  #include "Adafruit_I2CDevice.h"
3
4  const int TCS_SCL = 5; // Yellow wire
5  const int TCS_SDA = 4; // Blue wire
6  const int TCS_LED = 10;

```

- Define variables and intervals for lux sampling and averaging, and set a 40-lux threshold (measured at a reasonable distance from the phone's flash to the sensor) for LED activation.

```

8  uint16_t r, g, b, c, colorTemp, lux;
9
10 /* Initialise with default values (int time = 2.4ms, gain = 1x) */
11 Adafruit_TCS34725 tcs = Adafruit_TCS34725();
12
13 int lux_values = 0;
14 unsigned long lux_previousMillis = 0; // To store the last time a lux reading was taken
15 int lux_interval = 1000;           // Read the lux value once every second
16
17 int lux_numSamples = 0;
18 int avg_lux = 0;
19 unsigned long luxAvg_previousMillis = 0; // Timer for sampling
20 int luxAvg_interval = 5000;           // Turn the onboard LED ON if the average value in a
21 int Y_threshold = 40;

```

- Initialize serial communication; check sensor connection and print status messages, if the sensor is not found, enter an infinite loop to avoid unexpected operations.

```

23 void setup(void) {
24     Serial.begin(115200);
25
26     pinMode(TCS_LED, OUTPUT);
27     digitalWrite(TCS_LED, LOW);
28
29     if (tcs.begin()) {
30         Serial.println("TCS RGB Sensor connected");
31     }
32     else {
33         Serial.println("TCS RGB Sensor not found -> check connections");
34         while (1); // enter an infinite loop and stop further execution
35     }
36 }

```

- Main loop that reads lux values from the sensor once every second, calculates average lux over a window of 5 seconds, and activates the LED if the average lux exceeds Y_threshold.

```

38 void loop(void) {
39
40     unsigned long lux_currentMillis = millis();
41
42     // Check if it's time to take a new reading (every one second)
43     if (lux_currentMillis - lux_previousMillis >= lux_interval) {
44         tcs.getRawData(&r, &g, &b);
45         lux = tcs.calculateLux(r, g, b);
46
47         unsigned long luxAvg_currentMillis = millis(); // Track current time
48
49         // Check if it's time to calculate lux average
50         if (luxAvg_currentMillis - luxAvg_previousMillis >= luxAvg_interval) {
51             // Calculate average lux over the window
52             avg_lux = lux_values / lux_numSamples;
53             Serial.print("Avg Lux: "); Serial.print(avg_lux, DEC);
54             Serial.println(" ");
55             // Check if the average lux exceeds the threshold
56             if (avg_lux >= Y_threshold) {
57                 digitalWrite(TCS_LED, HIGH); // Turn on the LED
58             }
59             else {
60                 digitalWrite(TCS_LED, LOW); // Turn off the LED
61             }
62             luxAvg_previousMillis = luxAvg_currentMillis;
63             lux_numSamples = 0;
64             lux_values = 0;
65         }
66         else {
67             lux_values += lux;
68             lux_numSamples++;
69         }
70
71         Serial.print("Lux: "); Serial.print(lux, DEC);
72         Serial.println(" ");
73         lux_previousMillis = lux_currentMillis;
74     }
75 }

```

PART 2

Requirements

- Connect 3 external LEDs to ESP8266.
- Display the intensity of light from the RGB Sensor on 3 external LEDs using various patterns (e.g. low -> green LED on, medium -> yellow LED on, high -> red LED on). Document your value thresholds and justify.

Wiring Schematic

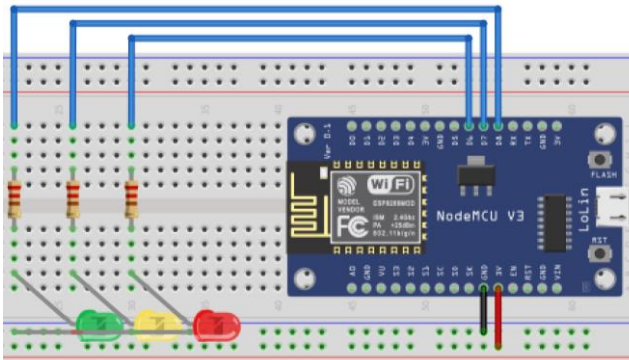


Figure 4. Wiring schematic for Part 2

LEDs	ESP8266
GND	GND
Red +	D6 GPIO12
Yellow +	D7 GPIO13
Green +	D8 GPIO15
Use 220Ω resistor for each LED	

Table 3. Wiring connections between 3 external LEDs and ESP8266

Code Explanation

- Define variables for controlling 3 external LEDs (red, yellow, and green), representing different ranges of lux level, and define thresholds for green (below 20 - low), yellow (from 20 to 70 - medium), and red (above 70 - high) (thresholds are defined based on experimental measurements with the phone's flash).

```

13 const int LED_red = 12; // Red LED
14 const int LED_yellow = 13; // Yellow LED
15 const int LED_green = 15; // Green LED
16
17 // low -> green LED on, medium -> yellow LED on, high -> red LED on
18 int upper_threshold = 70;
19 int lower_threshold = 20;

```


- Controls the green, yellow, and red LEDs based on lux levels collected once every second: green for lux below the lower threshold, yellow for values between thresholds, and red for lux above the upper threshold.

```
// Check if it's time to take a new reading (every one second)
if (lux_currentMillis - lux_previousMillis >= lux_interval) {
  tcs.getRawData(&r, &g, &b, &c);
  colorTemp = tcs.calculateColorTemperature(r, g, b);
  lux = tcs.calculateLux(r, g, b);

  // Check if the average lux is low or medium or high
  // low -> green LED on, medium -> yellow LED on, high -> red
  if (lux <= lower_threshold) {
    Serial.println("LOW");
    digitalWrite(LED_red, LOW);
    digitalWrite(LED_yellow, LOW);
    digitalWrite(LED_green, HIGH);
  }
  else if (lux >= upper_threshold) {
    Serial.println("HIGH");
    digitalWrite(LED_red, HIGH);
    digitalWrite(LED_yellow, LOW);
    digitalWrite(LED_green, LOW);
  }
  else {
    Serial.println("MEDIUM");
    digitalWrite(LED_red, LOW);
    digitalWrite(LED_yellow, HIGH);
    digitalWrite(LED_green, LOW);
  }
}
```

PART 3

Requirements

- Flash (instead of just ON) onboard LED if the average value in a sliding window of X seconds is higher than a threshold of Y.
- The higher the readings, the faster it flashes.

Wiring Schematic

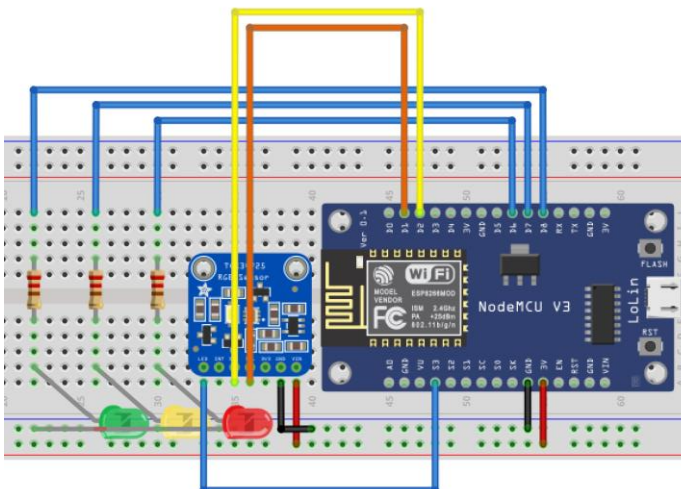


Figure 5. Wiring schematic for Part 3

Code Explanation

- Calculate the moving average of lux values read every one second over a sliding window of 5 seconds by creating a circular buffer.

```
71 void cal_movingAvg (int lux_value){
72     lux_total = lux_total - lux_readings[lux_index]; // Subtract the last reading from the lux_total
73     lux_readings[lux_index] = lux_value; // Take a new sensor reading
74     lux_total = lux_total + lux_readings[lux_index]; // Add the new reading to the lux_total
75     lux_index = (lux_index + 1) % lux_numReadings; // Advance to the next lux_index (circular buffer)
76     lux_movingAvg = lux_total / lux_numReadings; // Calculate the moving average
77 }
```

- Calculate the slope and intercept for a linear relationship between the lux value and the LED flashing interval, ensuring faster flashes at higher lux values.

```
void getSlopeIntercept(int x1, int y1, int x2, int y2, int *a, int *b) {
    // Calculate slope (a)
    *a = (y2 - y1) / (x2 - x1);
    // Calculate intercept (b)
    *b = y1 - (*a) * x1;
}

// Get slope and intercept for the linear equation of flashing interval
getSlopeIntercept(Y_threshold, Y_interval, Z_threshold, Z_interval, &slope, &intercept);
```

- Flash the onboard LED faster as the lux value increases, calculated using a linear equation, with a minimum flash interval of 10 ms.

```
86 void ledFlash(int ledPin, int lux_value, int a, int b){
87     if (lux_movingAvg > Y_threshold){
88         // We have:
89         // interval at Y_threshold: x = 20 -> y = 1050 (ms)
90         // interval at: x = 150 -> y = 10 (ms)
91         // => Linear function to calculate led_interval:
92         led_interval = a*lux_value + b;
93         if (led_interval <= 10){ led_interval = 10;}
94
95         unsigned long led_currentMillis = millis();
96         if (led_currentMillis - led_previousMillis >= led_interval) {
97             led_previousMillis = led_currentMillis;
98             if (led_state == LOW){ led_state = HIGH; }
99             else{ led_state = LOW; }
100             digitalWrite(ledPin, led_state);
101         }
102     }
103     else {
104         digitalWrite(ledPin, LOW);
105         led_interval = 0;
106     }
107 }
```

OPTIONAL PART

Requirements

- Add/use a button to switch between parts 1, 2, and 3 so you do not have to reprogram for each part.
- Display on the IDE/LED panel which part of the program is running.
- Integrate a tool to change the Y threshold externally.

Wiring Schematic

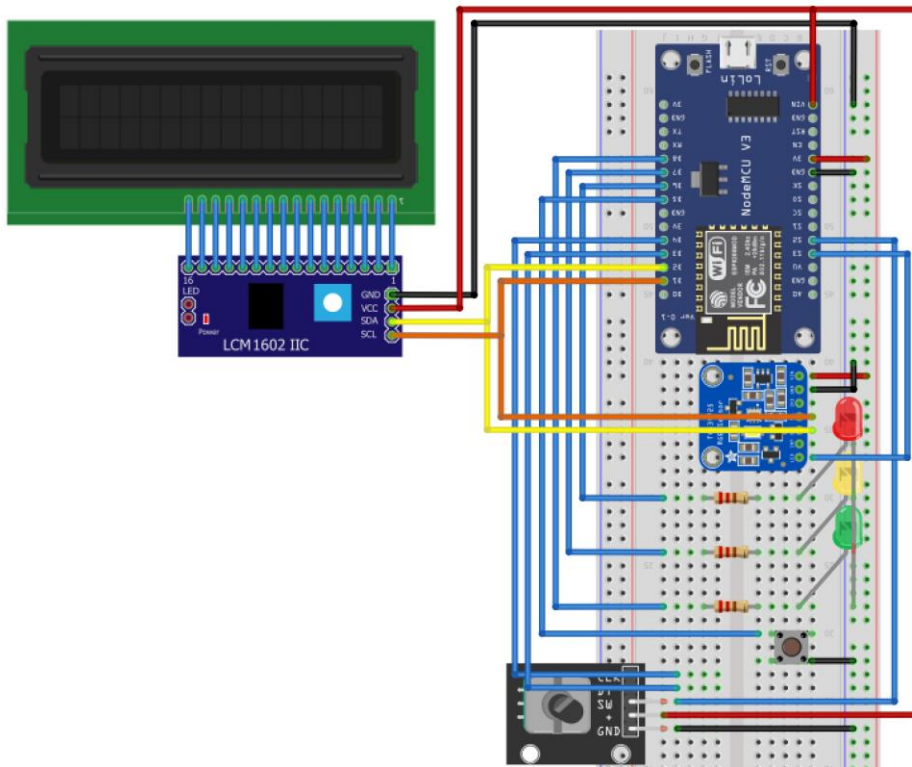


Figure 6. Wiring schematic for optional part

I2C LCD	ESP8266	Encoder	ESP8266	Button	ESP8266
GND	GND	GND	GND	NO 1	GND
VCC	Vin (USB Power)	+	3	NO 2	D5 GPIO14
SCL	D1 GPIO5 SCL	SW	GPIO9		
SDA	D2 GPIO4 SDA	DT	D3 GPIO0		
LED	GPIO10	CLK	D4 GPIO2		

Table 4. Wiring connections between I2C LCD, rotary encoder, push button and ESP8266

Code Explanation

- Include I2C communication library for LCD and RGB sensor (Wire.h), and control the LCD (LiquidCrystal_I2C).

```
4 #include "LiquidCrystal_I2C.h"
5 #include "Wire.h"
```

- Check if the main button was pressed and released. If it was, the part variable is incremented cyclically to switch between 3 different parts using Switch-Case logic.

```
200 // Check if the button is pressed
201 if(lastButtonState == LOW && buttonState == HIGH) {
202     part = (part % 3) + 1; // Button is pressed, increment part from 1 to 3 cyclicly
203 }
204 lastButtonState = buttonState;
```

- If the rotary encoder's state changes, the code checks which direction the knob was rotated (by comparing `encod_DT` and `encod_CLK`). Based on the rotation, the `Y_threshold` is increased or decreased within a range of 0 to 100 (self-defined). And if the encoder button is pressed, the `Y_threshold` is reset to its initial value (`Y_threshold_base = 20`).

```

200 // Check if the button is pressed
201 if(lastButtonState == LOW && buttonState == HIGH) {
202     part = (part % 3) + 1; // Button is pressed, increment part from 1 to 3 cyclically
203 }
204 lastButtonState = buttonState;
205
206 // Check if the state has changed
207 if (encod_currentState_CLK != encod_lastState_CLK) {
208     // Check the direction of rotation
209     if (digitalRead(encod_DT) != encod_currentState_CLK) {
210         Y_threshold++;
211     } else {
212         Y_threshold--;
213     }
214
215     Y_threshold = constrain(Y_threshold, 0, 100); // Constrain the Y_threshold between 0 and 100
216 }
217
218 encod_lastState_CLK = encod_currentState_CLK; // Update encod_lastState_CLK with the current state
219
220 // Check if the button is pressed
221 if(encod_lastButtonState == LOW && encod_buttonState == HIGH) {
222     Y_threshold = Y_threshold_base;
223 }
224 encod_lastButtonState = encod_buttonState;

```