Lampă Inteligentă - Maria-Ana Drăghici

Introducere

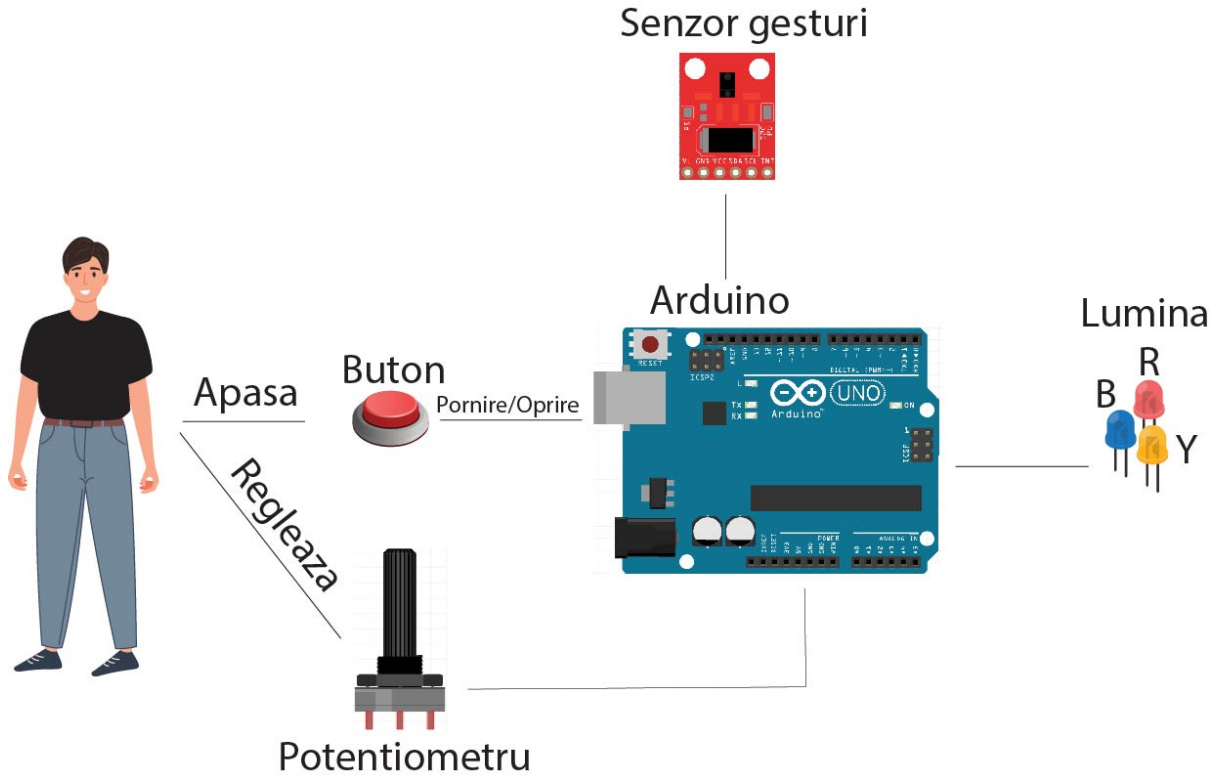
Scopul acestui proiect este realizarea unei lămpi inteligente care reflectă starea utilizatorului. Ideea a provenit de la dorința de a face un proiect interactiv care poate fi folosit zilnic de utilizator. Lampa intelgientă se poate dovedi utilă pentru a crea un ambient în concordanță cu starea utilizatorului sau pentru a-i oferi starea dorită (exemplu: albastru este o culoare ce calmează).

Descriere generală

Culoarea lămpii se va schimba în funcție de gesturile utilizatorului cu ajutorul unui senzor de gesturi astfel:

- dreapta = entuziasmat, agitat - lumină roșie
- stânga = calm, relaxat - lumină albastră
- jos = creativ, fericit - lumină galbenă
- sus = optimist - lumină mov

Lampa se va opri automat după o perioadă de 40 de secunde de la ultima schimbare a culorii luminii. Apăsând pe buton, utilizatorul va putea opri/porni lampa. De asemenea, utilizatorul va putea seta și intensitatea luminii cu ajutorul unui potențiometru.



Hardware Design

- Senzor gesturi APDS-9960
- Arduino Uno
- 3 LED-uri: R,Y,B
- Potențiometru
- Buton
- Rezistente

Software Design

Mediul de dezvoltare pentru acest proiect a fost Arduino IDE.

Biblioteci utilizate:

- SparkFun_APDS9960 - pentru senzorul de gesturi.

Implementare:

- Întreruperile, citirea valorii potentiometrului și timerul au fost implementate cu ajutorul regiștrilor.
- Variabila counter este folosită pentru a ști când lampa este pornită și când este oprită. Această variabilă contorizează numărul de apăsări de buton. Counter este inițial 0, deci: când counter este număr par, lampa va fi oprită (toate LED-urile sunt oprite), iar când counter este număr impar, înseamnă că lampa este pronită.

- După verificarea de pornire/oprire a lămpii, se va mai verifica si variabila timer, care ține evidența secundelor trecute de la ultima setare a culorii. Dacă au trecut 40 de secunde de la ultima schimbare a culorii luminii lămpii, aceasta se va stinge.
- Pentru o funcționare corectă există următoarele cazuri:

1. Dacă lampa se oprește din apăsarea butonului, atunci counter va fi incrementat și seconds va fi setat pe 0
2. Dacă lampa se stinge automat (au trecut 40 de secunde), atunci counter va fi setat pe valoare pară (pentru a ști ca lampa este închisă).
3. Când lampa este pornită, atunci când este înregistrat un gest (deci se schimbă culoarea luminii), seconds va fi resetat la 0.

```
#define BUTTON PD4
int counter = 0;
int buttonState = HIGH;
int lastButtonState = HIGH;
unsigned long lastDebounceTime = 0;
unsigned long debounceDelay = 100;
int seconds = 0;
// Other initializations
// .......

void setup() {
  // Initialize Serial port
  Serial.begin(9600);

  // Initialize interrupts
  setup_interrupts();

  // Setup code here
  // .....
}

void setup_interrupts() {
  // Enable pin change interrupt (for button)
  PCMSK2 |= (1 << PCINT20);
  // Set PCIE2 to enable PCMSK2 scan
  PCICR |= (1 << PCIE2);

  TIMSK1 |= (1 << OCIE1A); // enable interrupts TIMER1_COMPA

  // ......
}

void loop() {
  // Potentiometer code
  // ......
  /* If counter odd number, lamp is on
  if the seconds limit has not been reached
  execute tasks */
  if (counter % 2 == 1 && seconds < TIME_LIMIT) {
```

```
    // Handle gesture sensor
    // ......
  } else {
    // Turn off lamp
    // ......
  }
}

ISR(TIMER1_COMPA_vect) {
  // Interrupt code for TIMER1
  get_time();
}

void get_time() {
  // Increments number of seconds
  seconds++;
  // If the limit is reached
  if (seconds >= TIME_LIMIT) {
    // Turn off all LEDs
    // .....
    // Give counter even value for knowing
    // that the lamp is off
    counter = 0;
  }
}

// ......

ISR(PCINT2_vect) {
  // Debounce code for button
  // .......
    // If valid button press has occurred
    if (buttonState == HIGH && digitalRead(4) == LOW) {
      // Increment counter
      counter++;
      // Reset timer
      time = 0;
    }
    // Updates
    // .......
  }
}

// ......
```

- Citirea valorii potențiometrului se face constant în funcția de loop. Valoarea citită este salvată, iar intensitatea luminii setată în funcție de aceasta.

```
void loop() {
  /* Start conversion */
  ADCSRA |= (1 << ADSC);
```

```
  /* Wait until conversion is complete */
  while(!(ADCSRA & ( 1 << ADIF)));
  /* Save result */
  uint16_t result = ADC;
  /* Get brightness */
  int brightness = ADC / 4;

  // ......
}
```

- Pentru senzorul de gesturi a fost folosită biblioteca SparkFun_APDS9960. Acest senzor folosește I2C-ul. Pinul de întrerupere este conectat la pinul digital 2, deci se va folosi întreruperea externă INT0.

```
SparkFun_APDS9960 apds = SparkFun_APDS9960();
int isr_flag = 0;
// ....

voide setup() {
  // Other setups
  // ....

  setup_interrupts();

  // ....

  // Initialise gesture sensor
  if ( apds.init() ) {
    Serial.println(F("APDS-9960 initialization complete"));
  } else {
    Serial.println(F("Something went wrong during APDS-9960 init!"));
  }

  if ( apds.enableGestureSensor(true) ) {
    Serial.println(F("Gesture sensor is now running"));
  } else {
    Serial.println(F("Something went wrong during gesture sensor init!"));
  }

  // ....
}

void setup_interrupts() {
  // ....

  // Extern interrupt
  EIMSK |= (1 << INT0);

  // ....
```

```
}

void loop() {
  // ....

  // If lamp on
  // ....
  if( isr_flag == 1 ) {
      lights = handleGesture();
      isr_flag = 0;
    }

  switch (lights) {
        case 1:
          // Turn on all LEDs
          // ....
          break;
        case 2:
          // Turn on yellow LED
          // ....
          break;
        case 3:
          // Turn on blue LED
          // ....
        case 4:
          // Turn on red LED
          // ....
          break;
    }
  } else {
    // Turn off all LEDs
    // ....
  }

  // ....
}
int handleGesture() {
    if ( apds.isGestureAvailable() ) {
      // Return by case of gesture
    switch ( apds.readGesture() ) {
      case DIR_UP:
        // ....
        return 1;
      case DIR_DOWN:
        // ....
        return 2;
      case DIR_LEFT:
        // ....
        return 3;
      case DIR_RIGHT:
        // ....
```

```
        return 4;
      default:
        // ....
    }
  }
}
```

Rezultate Obţinute

Concluzii

Prin intermediul acestei lămpi inteligente, utilizatorul poate beneficia de un control mai mare asupra mediului înconjurător și poate experimenta beneficiile unei iluminări personalizate în funcție de starea sa. Căutarea informatiilor despre cum trebuie conectate componentele și crearea circuitului au fost interesante. De asemenea, a fost foarte util datasheet-ul unde am găsit regiștrii ce trebuie modificați pentru fiecare componentă. Macheta pentru proiect a avut o dificultate nu așa de ridicată ca și software-ul și hardware-ul. Mi-a plăcut să lucrez la acest proiect întrucât m-a interesat foarte mult tema aleasă.

Download

draghici_maria_proiect-pm.zip

Bibliografie/Resurse

Surse Hardware:

- https://ocw.cs.pub.ro/courses/pm/lab/lab0-2022
- https://ocw.cs.pub.ro/courses/pm/lab/lab1-2022
- https://ocw.cs.pub.ro/courses/pm/lab/lab2-2023
- https://ocw.cs.pub.ro/courses/pm/lab/lab3-2023
- https://ocw.cs.pub.ro/courses/pm/lab/lab4-2022
- https://how2electronics.com/interfacing-apds9960-gesture-rgb-color-sensor-with-arduino/

Surse Software:

- https://ocw.cs.pub.ro/courses/pm/lab/lab0-2022
- https://ocw.cs.pub.ro/courses/pm/lab/lab1-2022
- https://ocw.cs.pub.ro/courses/pm/lab/lab2-2023

- https://ocw.cs.pub.ro/courses/pm/lab/lab3-2023
- https://ocw.cs.pub.ro/courses/pm/lab/lab4-2022
- https://how2electronics.com/interfacing-apds9960-gesture-rgb-color-sensor-with-arduino/
- https://ocw.cs.pub.ro/courses/_media/pm/atmel-7810-automotive-microcontrollers-atmega328p_datasheet.pdf

Export to PDF