Light Dress

- Keren Ioana Boingiu - 332 CB

Introducere

Inspirată de designurile emotionally-responsive din filmele SF, vreau să realizez o rochie cu un inel de leduri ce va lumina în mod idferit în funcție de pulsul persoanei care o poartă, temperatură, intensitate luminoasă și sunet. Scopul este pur de divertisment, menit să creeze o legătură între mediul înconjurător și persoana care poartă rochia.

Prezentarea pe scurt a proiectului vostru:

- ce face
- care este scopul lui
- care a fost ideea de la care aţi pornit
- de ce credeţi că este util pentru alţii şi pentru voi

Descriere generală

Prezentarea pe scurt a proiectului:

- Constă într-un articol vestimentar ce luminează în funcție de contextul în care este purtat.
- Poate reprezenta un bun ice-breaker prin designul atrăgător și inedit.
- **PRIMA ȘI PRIMA** sursă de inspirație a reprezentat-o un colier dintr-o carte pe care am citit-o care lumina în funcție de sentimentele pe care purtatorul le are pentru persoana care le-a oferit bijuteria. Cu toate acestea, ***nu trăim intr-o lume fantasy***, așa că din noțiunea de “sentiment” am derivat în zona de numărare a bătăilor inimii. Păentru complexitate, vor fi implementate și restul de funcționalități.
- Utilitatea în plan personal - mă voi familiariza cu noțiunile predate, tocmai prin aplicarea lor într-un proiect mai complex. Proiectul se dorește a avea scop decorativ/social, deci utilitatea în plan extins va fi pentru **divertisment**.

O schemă bloc cu toate modulele proiectului vostru, atât software cât şi hardware însoţită de o

descriere a acestora precum şi a modului în care interacţionează.

Exemplu de schemă bloc: http://www.robs-projects.com/mp3proj/newplayer.html

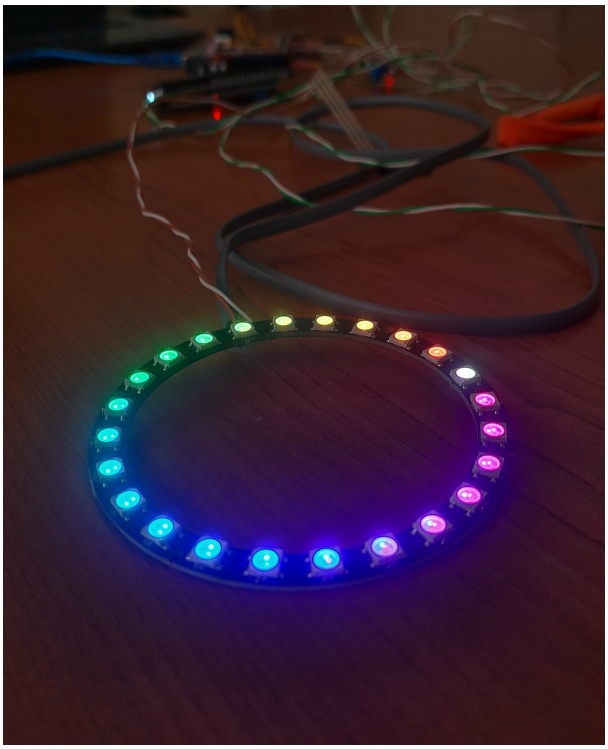
Hardware Design

Listă de piese

- 1 x Placa de dezvoltare compatibila cu Arduino UNO (ATmega328p si CH340) Plusivo
- 1 x Inel cu 24 LED-uri RGB
- 1 x modul senzor de puls si pulsoximetru ~~+ temperatura~~

update 26.05 - nu am mai implementat deloc funcționalitatea de temperatură

- 1 x modul intensitate luminoasa
- 1 x modul sunet
- 1 x buton
- 1 x buzzer
- LCD 2 x 16



Aici puneţi tot ce ţine de hardware design:

- listă de piese

- scheme electrice (se pot lua şi de pe Internet şi din datasheet-uri, e.g. http://www.captain.at/electronic-atmega16-mmc-schematic.png)
- diagrame de semnal
- rezultatele simulării

Software Design

Nu știu cum să dau fold la cod, deci scuze că e atât de lung și trebuie să dai scroll

```
#include <Wire.h>
#include "MAX30105.h"
#include "heartRate.h"
#include <LiquidCrystal_I2C.h>
#include <FastLED.h>

MAX30105 particleSensor;
const byte RATE_SIZE = 8;
byte rates[RATE_SIZE];
byte rateSpot = 0;
long lastBeat = 0;
float beatsPerMinute;
int beatAvg;
int color;

const int buzzerPin = 9;
const int soundSensorPinDigital = 12;
const int soundSensorPinAnalog = A1;

LiquidCrystal_I2C lcd(0x27, 16, 2);
const int buttonPin = 5;
int buttonPressCounter = 0; // Contor pentru apăsările butonului
const int REST = 4; // Update to include the new state
bool isNight = false;
bool isDay = false;
unsigned long lastLightUpdate = 0;
const unsigned long lightUpdateInterval = 1000;
bool lastButtonState = HIGH; // Starea anterioară a butonului

#define LED_PIN     7
#define NUM_LEDS    24
#define BRIGHTNESS  69  // Ajustează nivelul de luminozitate (0-255)

CRGB leds[NUM_LEDS];

enum DisplayState { LIGHT_INTENSITY, PULSE_RATE, LED_COLOR, SOUND_LEVEL };
DisplayState currentState = LIGHT_INTENSITY;
```

```
void setup() {
  Serial.begin(115200);
  Serial.println("Initializing...");

  // Configurare LED-uri
  FastLED.addLeds<WS2812, LED_PIN, GRB>(leds,
NUM_LEDS).setCorrection(TypicalLEDStrip);
  FastLED.setBrightness(BRIGHTNESS); // Setează luminozitatea generală

  // Configurare senzor puls
  if (!particleSensor.begin(Wire, I2C_SPEED_FAST)) {
    Serial.println("MAX30102 was not found. Please check wiring/power. ");
    while (1);
  }
  Serial.println("Place your index finger on the sensor with steady
pressure.");

  particleSensor.setup();
  particleSensor.setPulseAmplitudeRed(0x0A);
  particleSensor.setPulseAmplitudeGreen(0);

  // Inițializare LCD
  lcd.init();
  lcd.backlight();
  pinMode(buttonPin, INPUT_PULLUP);
  displayLightIntensity(); // Afișează starea inițială a intensității
luminii

  pinMode(buzzerPin, OUTPUT);
  digitalWrite(buzzerPin, LOW);
}

void loop() {
    bool buttonState = digitalRead(buttonPin);

  if (buttonState != lastButtonState) {
    lastButtonState = buttonState;
    if (buttonState == LOW) {
      buttonPressCounter++;
      currentState = static_cast<DisplayState>(buttonPressCounter % REST);
      updateDisplay();
      buzzBuzzer(100);
    }
  }

  // Măsurare puls și actualizare valori
  long irValue = particleSensor.getIR();

  if (checkForBeat(irValue) == true) {
    long delta = millis() - lastBeat;
```

```
    lastBeat = millis();

    beatsPerMinute = 60 / (delta / 1000.0);

    if (beatsPerMinute < 255 && beatsPerMinute > 20) {
      rates[rateSpot++] = (byte)beatsPerMinute;
      rateSpot %= RATE_SIZE;

      beatAvg = 0;
      for (byte x = 0; x < RATE_SIZE; x++)
        beatAvg += rates[x];
      beatAvg /= RATE_SIZE;
    }
  }

  // Afișare valori în serial monitor
  Serial.print("IR=");
  Serial.print(irValue);
  Serial.print(", BPM=");
  Serial.print(beatsPerMinute);
  Serial.print(", Avg BPM=");
  Serial.print(beatAvg);

  if (irValue < 50000)
    Serial.print(" No finger?");

  Serial.println();

  // Actualizează afișajul corespunzător în timp real
  updateDisplay();
}

// Actualizează afișajul LCD în funcție de starea curentă
void updateDisplay() {
  static DisplayState lastState = currentState;
  static unsigned long lastUpdate = 0;
  unsigned long currentMillis = millis();

  // Actualizează afișajul la fiecare 500 ms pentru a evita refresh-ul
excesiv
  if (currentMillis - lastUpdate >= 500 || lastState != currentState) {
    lastUpdate = currentMillis;
    lastState = currentState;

    switch (currentState) {
      case LIGHT_INTENSITY:
        displayLightIntensity();
        break;
      case PULSE_RATE:
        displayPulseRate();
        break;
```

```
      case LED_COLOR:
        displayLEDColor();
        break;
      case SOUND_LEVEL:
        displaySoundLevel();
        break;
    }
  }
}

// Afișează pulsul mediu pe LCD
void displayPulseRate() {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Puls mediu:      ");
  lcd.setCursor(0, 1);
  lcd.print(beatAvg);
  lcd.print(" BPM");

  Serial.print("Puls mediu: ");
  Serial.print(beatAvg);
  Serial.println(" BPM");
}

// Afișează intensitatea luminii pe LCD
void displayLightIntensity() {
  float volts = analogRead(A0) * 5.0 / 1024.0;
  float amps = volts / 10000.0;
  float microamps = amps * 1000000;
  float lux = microamps * 2.0;

  if (lux < 60 && !isNight) {
    isNight = true;
    isDay = false;
  } else if (lux >= 60 && !isDay) {
    isNight = false;
    isDay = true;
  }

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Lux Intensity:   ");
  lcd.setCursor(0, 1);
  lcd.print("Lux: ");
  lcd.print(lux, 1);

  lcd.setCursor(11, 1);
  if (isNight) {
    lcd.print("Night ");
  } else if (isDay) {
```

```
    lcd.print("Day   ");
  }
}

// Afișează culoarea LED pe LCD și pulsul mediu
void displayLEDColor() {
  // Determinăm culoarea în funcție de valoarea BPM
  if (beatAvg < 55) {
    color = 9; // indigo
  } else if (beatAvg >= 55 && beatAvg < 60) {
    color = 8; // Violet
  } else if (beatAvg >= 60 && beatAvg < 65) {
    color = 7; // albastru
  } else if (beatAvg >= 65 && beatAvg < 70) {
    color = 6; // cyan
  } else if (beatAvg >= 70 && beatAvg < 75) {
    color = 5; // bleu
  } else if (beatAvg >= 75 && beatAvg < 80) {
    color = 4; // verde
  } else if (beatAvg >= 80 && beatAvg < 85) {
    color = 3; // galben
  } else if (beatAvg >= 85 && beatAvg < 90) {
    color = 2; // potocaliu
  } else if (beatAvg >= 90 && beatAvg < 95) {
    color = 1; // rosu
  } else {
    color = 0; // Roșu
  }

  Serial.println(color); // Trimite valoarea culorii prin portul serial
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Culoare:");
  switch(color) {
    case 0:
      lcd.print(" rosu!!");
      break;
    case 1:
      lcd.print(" rosu");
      break;
    case 2:
      lcd.print(" portocaliu");
      break;
    case 3:
      lcd.print(" galben");
      break;
    case 4:
      lcd.print(" verde");
      break;
    case 5:
      lcd.print(" bleu");
```

```
      break;
    case 6:
      lcd.print(" cyan");
      break;
    case 7:
      lcd.print(" albastru");
      break;
    case 8:
      lcd.print(" violet");
      break;
    case 9:
      lcd.print(" indigo");
      break;
  }

  lcd.setCursor(0, 1);
  lcd.print(beatAvg);
  lcd.print(" BPM");



  setColor(color); // Setează culorile LED-urilor
}

// Afișează nivelul sunetului pe LCD
void displaySoundLevel() {
  int soundLevel = analogRead(soundSensorPinAnalog);

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Sound Level:     ");
  lcd.setCursor(0, 1);

  if (soundLevel > 800)
    lcd.print("zgomot!!");
  else
    lcd.print("liniste...");
  //lcd.print(soundLevel);

  Serial.print("Sound Level: ");
  Serial.println(soundLevel);
}

// Setează toate LED-urile la culoarea determinată în funcție de valoarea
lui 'color'
void setColor(int color) {
  CRGB colorValue;
  switch(color) {
    case 0: // rosu!!!
      colorValue = CRGB::Red;
```

```
      break;
    case 1: // rosu
      colorValue = CRGB::Red;
      break;
    case 2: // portocaliu
      colorValue = CRGB::Orange;
      break;
    case 3: // galben
      colorValue = CRGB::Yellow;
      break;
    case 4: // verde
      colorValue = CRGB::Green;
      break;
    case 5: // bleu
      colorValue = CRGB::LightBlue;
      break;
    case 6: // cyan
      colorValue = CRGB::Cyan;
      break;
    case 7: // albastru
      colorValue = CRGB::Blue;
      break;
    case 8: // violet
      colorValue = CRGB::Violet;
      break;
    case 9: // indigo
      colorValue = CRGB::Indigo;
      break;
    default:
      colorValue = CRGB::White; // Setează implicit la alb dacă 'color' este
în afara intervalului
      break;
  }

  for (int i = 0; i < NUM_LEDS; i++) {
    leds[i] = colorValue;
  }
  FastLED.show();
  Serial.println("LED color updated"); // Mesaj de debug pentru confirmare
}

void buzzBuzzer(int duration) {
  digitalWrite(buzzerPin, HIGH); // buzzer on
  delay(duration);
  digitalWrite(buzzerPin, LOW);  // buzzer off
}
```

Descrierea codului aplicaţiei (firmware)

Mediu de dezvoltare:

- **Arduino IDE**

Librării şi surse 3rd-party:

- **Wire.h**: Comunicație I2C.
- **MAX30105.h**: Senzor puls MAX30105.
- **heartRate.h**: Calcul puls.
- **LiquidCrystal_I2C.h**: Afișaj LCD I2C.
- **FastLED.h**: Control benzi LED.

Algoritmi şi structuri:

- **Detecție puls**: Citire și calcul ritm cardiac folosind MAX30105.
- **Mediere valori**: Calcul medie puls.
- **Stări afișaj**: Gestionare stări pentru afișarea informațiilor pe LCD.
- **Control LED-uri**: Schimbare culoare LED-uri în funcție de puls.

(Etapa 3) Surse şi funcţii implementate: 1. **Setup**:

1. Configurare LED-uri.
2. Configurare senzor puls.
3. Inițializare afișaj LCD.
4. Configurare buton.

2. **Loop**:

1. Detectare apăsări buton.
2. Citire și calcul puls.
3. Actualizare afișaj.

3. **Funcții de afișare**:

1. **updateDisplay**: Actualizare afișaj.
2. **displayPulseRate**: Afișare puls mediu.
3. **displayLightIntensity**: Afișare intensitate lumină.
4. **displayLEDColor**: Afișare și setare culoare LED.
5. **displaySoundLevel**: Afișare nivel zgomot.

4. **Funcții auxiliare**:

1. **setColor**: Setare culoare LED-uri.
2. **buzzBuzzer**: Activare buzzer pentru feedback.

Rezultate Obţinute

YouTube Video Embed