Lumino-X

Introducere

“Lumino-X” este un sistem avansat de iluminare care folosește tehnologie LED RGB pentru a crea un mediu adaptabil și confortabil în orice încăpere. Proiectul integrează un senzor de mișcare PIR și un senzor de lumină, alături de un ceas real-time (RTC) pentru a ajusta culoarea și intensitatea luminii LED-ului RGB în funcție de ora din zi și de prezența umană. Scopul “Lumino-X” este de a optimiza consumul de energie în timp ce oferă o iluminare adecvată pentru sănătatea și confortul utilizatorilor.

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

Lumino-X este un sistem automatizat de iluminare care utilizează LED-uri RGB pentru a ajusta lumina dintr-o încăpere în funcție de prezența umană și ora din zi. Proiectul integrează un senzor de mișcare PIR și un senzor de lumină ambientală, alături de un modul RTC DS3231, pentru a oferi iluminare adaptată nevoilor utilizatorilor și pentru a promova economia de energie. Cu “Lumino-X”, lumina se ajustează dinamic: stimulează trezirea dimineața, sprijină productivitatea pe timpul zilei și favorizează relaxarea seara, închizându-se automat în timpul nopții sau în absența activității. Aceasta tehnologie este ideală pentru locuințe, birouri și alte spații, combinând confortul cu eficiența energetică într-o soluție de iluminare inteligentă și accesibilă.

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

Componente Hardware Arduino Uno - Unitatea centrală de control care execută codul firmware pentru gestionarea diferitelor funcții ale sistemului.

Senzor de lumină - Utilizat pentru a măsura intensitatea luminii ambientale. Acesta permite sistemului să ajusteze automat intensitatea și culoarea LED-urilor RGB în funcție de condițiile de iluminat existente.

Senzor PIR (Passive Infrared Sensor) - Detectează mișcarea în încăpere, permițând sistemului să activeze sau să regleze iluminarea când detectează prezența umană.

Modul RTC DS3231 - Un ceas real-time care păstrează timpul exact. Este utilizat pentru a regla iluminarea în funcție de ora din zi, facilitând astfel schimbări automate ale iluminării de la lumina albastră dimineața la lumina caldă seara.

LED RGB - Oferă iluminat cu capacitatea de a schimba culoarea și intensitatea. Folosit pentru a crea efecte de iluminat dinamic și personalizat în funcție de setările sistemului și de mediul ambiental.

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

Descrierea codului aplicaţiei (firmware):

- Mediu de dezvoltare: Arduino IDE
- Librării şi surse 3rd-party: AVRlib, Arduino standard libraries
- Algoritmi şi structuri:
  1. Comunicarea I2C cu un modul RTC (DS3231) pentru a obține timpul curent.
  2. Utilizarea unui senzor PIR pentru detectarea mișcării.
  3. Utilizarea unui senzor de lumină pentru a citi intensitatea luminii ambientale.
  4. Controlul unui LED RGB pentru a schimba culorile în funcție de timpul zilei și intensitatea luminii.
- (etapa 3) Surse şi funcţii implementate:
  1. Inițializarea și controlul magistralei I2C.
  2. Citirea datelor de la modulul RTC.
  3. Configurarea și citirea senzorilor de lumină și PIR.
  4. Controlul culorii LED-ului RGB pe baza timpului și luminii detectate.

Codul aplicației:

```
#include <avr/io.h>
#include <util/twi.h>
#include <Arduino.h>

#define DS3231_ADDRESS 0x68

#define light A0 // Pin pentru senzorul de lumină
#define PIR_PIN 2 // Pinul D2 pentru PIR

// Pini pentru LED-ul RGB pe pini PWM
#define RED_PIN 3
#define GREEN_PIN 5
#define BLUE_PIN 6

struct DateTime {
  uint8_t second;
  uint8_t minute;
  uint8_t hour;
  uint8_t day;
  uint8_t date;
  uint8_t month;
  uint8_t year;
};

void i2c_init() {
  TWSR = 0x00; // Set prescaler to 1
  TWBR = 0x48; // Set bit rate register for 100kHz with 16MHz clock
}

void i2c_start() {
  TWCR = (1 << TWSTA) | (1 << TWEN) | (1 << TWINT);
  while (!(TWCR & (1 << TWINT)));
}

void i2c_stop() {
  TWCR = (1 << TWSTO) | (1 << TWEN) | (1 << TWINT);
}

void i2c_write(uint8_t data) {
  TWDR = data;
  TWCR = (1 << TWEN) | (1 << TWINT);
  while (!(TWCR & (1 << TWINT)));
}

uint8_t i2c_read_ack() {
  TWCR = (1 << TWEN) | (1 << TWINT) | (1 << TWEA);
  while (!(TWCR & (1 << TWINT)));
  return TWDR;
}
```

```
uint8_t i2c_read_nack() {
  TWCR = (1 << TWEN) | (1 << TWINT);
  while (!(TWCR & (1 << TWINT)));
  return TWDR;
}

uint8_t bcdToDec(uint8_t val) {
  return ((val / 16 * 10) + (val % 16));
}

DateTime readRTC() {
  DateTime now;

  i2c_start();
  i2c_write((DS3231_ADDRESS << 1) | TW_WRITE);
  i2c_write(0x00); // Start from the register 0x00
  i2c_start();
  i2c_write((DS3231_ADDRESS << 1) | TW_READ);

  now.second = bcdToDec(i2c_read_ack());
  now.minute = bcdToDec(i2c_read_ack());
  now.hour = bcdToDec(i2c_read_ack());
  now.day = bcdToDec(i2c_read_ack());
  now.date = bcdToDec(i2c_read_ack());
  now.month = bcdToDec(i2c_read_ack());
  now.year = bcdToDec(i2c_read_nack());

  i2c_stop();

  return now;
}

void setRGBColor(int red, int green, int blue) {
  analogWrite(RED_PIN, red);
  analogWrite(GREEN_PIN, green);
  analogWrite(BLUE_PIN, blue);
}

void setRGBColorForTimeOfDay(int hour, int mVolt) {
  int baseIntensity = map(mVolt, 0, 5000, 0, 255);
  if (hour >= 6 && hour < 10) {
    setRGBColor(baseIntensity, baseIntensity, 255);
  } else if (hour >= 10 && hour < 16) {
    setRGBColor(baseIntensity, baseIntensity, baseIntensity);
  } else if (hour >= 16 && hour < 19) {
    setRGBColor(255, map(baseIntensity, 0, 255, 165, 255), 0);
  } else {
    setRGBColor(255, map(baseIntensity, 0, 255, 0, 255), 0);
  }
}
```

```
void setup() {
  Serial.begin(9600);
  i2c_init();

  // Setează pinii LED-ului RGB ca output utilizând registre
  DDRD |= (1 << DDD3); // RED_PIN
  DDRD |= (1 << DDD5); // GREEN_PIN
  DDRD |= (1 << DDD6); // BLUE_PIN

  // Setează pinul PIR ca input utilizând registre
  DDRD &= ~(1 << DDD2); // PIR_PIN
}

void loop() {
  DateTime now = readRTC();

  int pirState = PIND & (1 << PIND2); // Citește starea senzorului PIR utilizând registre
  int lightValue = analogRead(light); // Citește valoarea senzorului de lumină
  int mVolt = map(lightValue, 0, 1023, 0, 5000); // Mapează valoarea luminii la mVolt

  if (pirState && !(now.hour >= 1 && now.hour < 8)) {
    setRGBColorForTimeOfDay(now.hour, mVolt); // Setează culoarea LED-ului în funcție de ora zilei și luminozitate
  } else {
    setRGBColor(0, 0, 0); // Stinge LED-ul RGB
  }

  delay(1000); // Așteaptă o secundă înainte de a citi din nou
}
```

Rezultate Obţinute

Care au fost rezultatele obţinute în urma realizării proiectului vostru.

Concluzii

Download

O arhivă (sau mai multe dacă este cazul) cu fişierele obţinute în urma realizării proiectului: surse, scheme, etc. Un fişier README, un ChangeLog, un script de compilare şi copiere automată pe uC crează întotdeauna o impresie bună .

Fişierele se încarcă pe wiki folosind facilitatea **Add Images or other files**. Namespace-ul în care se încarcă fişierele este de tipul **:pm:prj20??:c?** sau **:pm:prj20??:c?:nume_student** (dacă este cazul). **Exemplu:** Dumitru Alin, 331CC → **:pm:prj2009:cc:dumitru_alin**.

Jurnal

Puteți avea și o secțiune de jurnal în care să poată urmări asistentul de proiect progresul proiectului.

Bibliografie/Resurse

Listă cu documente, datasheet-uri, resurse Internet folosite, eventual grupate pe **Resurse Software** şi **Resurse Hardware**.

Export to PDF