Pian electronic

Introducere

Scopul acestui proiect este de a implementa un pian electric cu 8 butoane, in 3 octave, care sa ii ajute pe cei pasionati sa invete sa cante.

Descriere generală

- 8 butoane: folosite pentru a reprezenta clapele pianului
- 2 butoane: folosite pentru a schimba octavele
- Buzzer activ 5V Rasberry OKY0151: folosit pentru a reda sunetul produs de clape
- Arduino Uno R3 cu sistemul CH340: Placa centrala a proiectului, ce va gazdui toate componentele si care va coordona functinarea pianului.
- Fire si rezistoare: Folosite pentru trecerea curentului prin componente, si protejarea acestora
- Led RGB: Folosit pentru a reprezenta octava la care pianul canta la momentul curent

Hardware Design

- 10 butoane
- 13 1k resistori
- Fire
- Buzzer activ 5V Rasberry OKY0151
- Arduino Uno R3
- Led RGB

Software Design

Descrierea codului aplicaţiei (firmware):

- mediu de dezvoltare (if any) (e.g. AVR Studio, CodeVisionAVR)
- librării şi surse 3rd-party (e.g. Procyon AVRlib)
- algoritmi şi structuri pe care plănuiţi să le implementaţi
- (etapa 3) surse şi funcţii implementate

Mediul de dezvoltare:

- Codul a fost scris in Arduino IDE.
- Schema electrica a fost realizata in Tinkercad

Functia setup : Initializeaza pinii digitali si pe cei de intrerupere:

```
Serial.begin(9600);
// SETEAZA PINUL 5 CA OUTPUT (BUZZER)
DDRD |= (1 << DDD5);
// SETEAZA PINUL 4 CA OUTPUT (LED)
DDRD |= (1 << DDD4);
// SETEAZA PINUL 6 SI 7 CA INPUS
DDRD &= ~((1 << DDD6) | (1 << DDD7));
PORTD |= ((1 << PORTD6) | (1 << PORTD7));
// SETEAZA PINUL 8-13 CA INPUS
DDRB &= ~((1 << DDB0) | (1 << DDB1) | (1 << DDB2) | (1 << DDB3) | (1 << DDB4)
| (1 << DDB5));
PORTB |= ((1 << PORTB0) | (1 << PORTB1) | (1 << PORTB2) | (1 << PORTB3) | (1
<< PORTB4) | (1 << PORTB5));
// SETEAZA PINUL 2 SI 3 CA INPUT PENTRU INTRERUPERI
DDRD &= ~((1 << DDD2) | (1 << DDD3));
PORTD |= ((1 << PORTD2) | (1 << PORTD3));
// CONFIGURARE INTRERUPERI
EICRA |= (1 << ISC01); // PD2
EICRA |= (1 << ISC11); // PD3
EIMSK |= (1 << INT0) | (1 << INT1); //ACTIVAM INTRERUPERILE
sei();
```

Functia loop : Verifica daca s-a apasat vreo clapa, si face reprezentarea unei octave, in cazul schimbarii acesteia

```
  if (octave >= 0 && octave <= 2) {
  if (!(PINB & (1 << PINB5))) {
    tone(5, f0[octave * 8], 100); // PINUL 13
    Serial.println("btn1 apasat.");
    _delay_ms(100);
  }
  if (!(PINB & (1 << PINB4))) {
    tone(5, f0[octave * 8 + 1], 100); // PINUL 12
```

```
    Serial.println("btn2 apasat.");
    _delay_ms(100);
  }
```

```
  if (!(PINB & (1 << PINB3))) {
    tone(5, f0[octave * 8 + 2], 100); // PINUL 11
    Serial.println("btn3 apasat.");
    _delay_ms(100);
  }
  if (!(PINB & (1 << PINB2))) {
    tone(5, f0[octave * 8 + 3], 100); // PINUL 10
    Serial.println("btn4 apasat.");
    _delay_ms(100);
  }
  if (!(PINB & (1 << PINB1))) {
    tone(5, f0[octave * 8 + 4], 100); // PINUL 9
    Serial.println("btn5 apasat.");
    _delay_ms(100);
  }
  if (!(PINB & (1 << PINB0))) {
    tone(5, f0[octave * 8 + 5], 100); // PINUL 8
    Serial.println("btn6 apasat.");
    _delay_ms(100);
  }
  if (!(PIND & (1 << PIND7))) {
    tone(5, f0[octave * 8 + 6], 100); // PINUL 7
    Serial.println("btn7 apasat.");
    _delay_ms(100);
  }
  if (!(PIND & (1 << PIND6))) {
    tone(5, f0[octave * 8 + 7], 100); // PINUL 6
    Serial.println("btn8 apasat.");
    _delay_ms(100);
  }
  //REPREZENTARE OCTAVA
  if(state == true){
    if(octave == 0){
      PORTD |= (1 << PORTD4); //APRINDEM LEDUL
      _delay_ms(100);
      PORTD &= ~(1 << PORTD4); //STINGEM LEDUL
    }
    else if (octave == 1){
      PORTD |= (1 << PORTD4);
      _delay_ms(100);
      PORTD &= ~(1 << PORTD4);
      _delay_ms(100);
      PORTD |= (1 << PORTD4);
      _delay_ms(100);
      PORTD &= ~(1 << PORTD4);
    }
    else if (octave == 2){
```

```
      PORTD |= (1 << PORTD4);
      _delay_ms(100);
      PORTD &= ~(1 << PORTD4);
      _delay_ms(100);
      PORTD |= (1 << PORTD4);
      _delay_ms(100);
      PORTD &= ~(1 << PORTD4);
      _delay_ms(100);
      PORTD |= (1 << PORTD4);
      _delay_ms(100);
      PORTD &= ~(1 << PORTD4);
    }
    Serial.println(octave);
    state = false;
  }
}
_delay_ms(10);
```

Intreruperile : Acestea sunt realizate pe butoanele ce schimba octavele, avand rolul de a creste performanta codului.

```
ISR(INT0_vect) {
unsigned long currentTime = millis();
if (currentTime - lastDebounceTimeUp > debounceDelay) {
octaveUp();
lastDebounceTimeUp = currentTime;
}
}
ISR(INT1_vect) {
unsigned long currentTime = millis();
if (currentTime - lastDebounceTimeDown > debounceDelay) {
  octaveDown();
  lastDebounceTimeDown = currentTime;
}
}
void octaveDown(void) {
Serial.println("Interrupt for Octave Down triggered");
octave = max(0, octave - 1);
Serial.println("OCT1 apasat.");
state = true;
_delay_ms(200);
}
void octaveUp(void) {
Serial.println("Interrupt for Octave Up triggered");
octave = min(2, octave + 1);
Serial.println("OCT2 apasat.");
state = true;
_delay_ms(200);
```

```
}
```

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