Frequency Spectrum Analyzer & Displayer - Tulbă-Lecu Theodor-Gabriel 331CA

Introducere

Acest proiect reprezintă implementarea unui analizator de spectru ce captează sunetul din mediul încojurător. Asupra sample-urilor de sunet captate se realizează o transformare Fourier și se obține spectrul de frecvențe. Datele sunt afișate codificând spectograma obtinută pe un “ecran” de 24×8 LED-uri (alcatuită din 3 matrici LED 8×8) în funcție de frecvență (coloane) și intensitatea sunetului (rânduri). Matricile sunt organizate în așa fel încât frecvențele joase sunt reprezentate cu roșu, cele medii cu verde și cele înalte cu albastru.

Descriere generală

Principiul de funcționare al proiectului este următorul:

```
1. Prin intermediul unui microfon sunt preluate sunetele sub formă de date
analog;
2. Asupra acestor date se aplică o transformare Fourier și se obține
spectograma;
3. Spectograma este afisată pe ecranul de LED-uri.
```

Schemă bloc

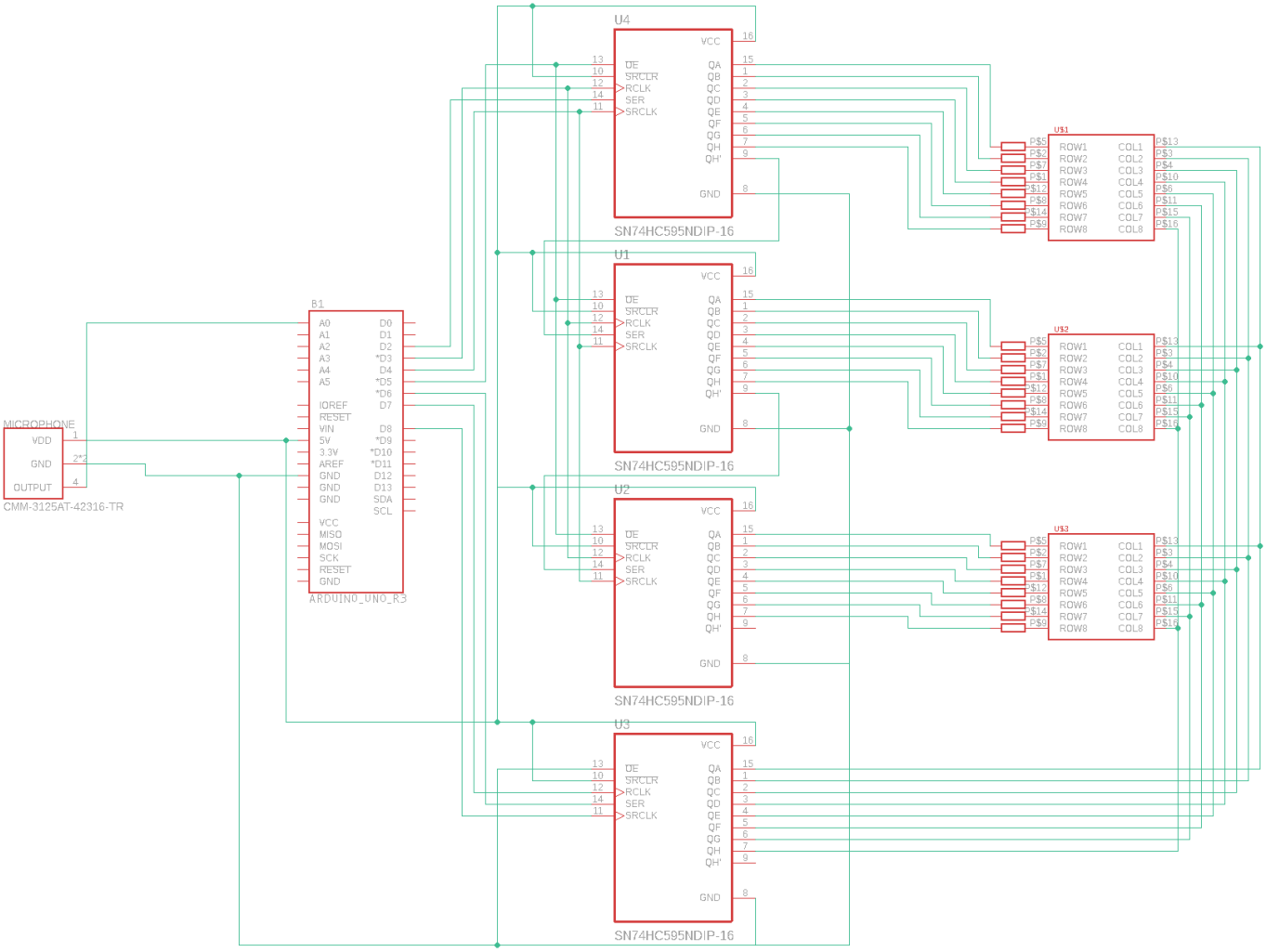
Hardware Design

Piese

- 1x Arduino Uno R3 ATmega328p;
- 1x Modul microfon High Sensitivity Sound Detection;
- 1x Matrice LED 8x8 3mm rosu (38mm);

- 1x Matrice LED 8x8 3mm verde (32mm);
- 1x Matrice LED 8x8 3mm albastru(32mm);
- 4x Shift register SN74HC595N;
- 24x Rezistor 220Ω;
- 2x Breadboard 830 puncte;
- 3x Placă prototipare 5x7 cm
- 1x Carcasă printată 3D

Schema Electrică



Software Design

Biblioteci Externe

- fix_fft - folosit pentr a aplica transformata Fourier
- TimerOne - folosit pentru interrupt-uri

Implementare

Setup

În setup se initializează pinii, se testează că funcționează matricile LED și se initializează un timer. Timer-ul este folosit pentru a crea întreruperi în mod regulat ce va declanșa randarea următoarei coloane de pixeli ai “ecranului”.

```
void setup() {
  // Initialize pins
  pinMode(SER1, OUTPUT);
  pinMode(RCLK1, OUTPUT);
  pinMode(SRCLK1, OUTPUT);
  pinMode(OE, OUTPUT);
  pinMode(SER2, OUTPUT);
  pinMode(RCLK2, OUTPUT);
  pinMode(SRCLK2, OUTPUT);

  digitalWrite(OE, HIGH);

  // Code for checking if the LED lights work
  for(int i = 1; i <= 8; i++) {
    for(int j = 0; j < 8; j++) {
      write_mat((1<<i) - 1, (1<<i) - 1, (1<<i) - 1, 1<<j);
      delay(50);
    }
  }
  write_mat(255, 255, 255, 255);
  delay(2000);

  // Initialize Timer for the interrupts
  Timer1.initialize(1000); //microseconds
  Timer1.attachInterrupt(draw_column);
}
```

Comunicare cu Shift Register

Conform datasheet-ului, se transmit date seriale ce sunt reținute în shift register. cel mai simplu mod de interacționare este folosirea funcției `shiftOut` care transmite un întreg byte de date.

```
// Send one byte of data to a shift register
void write_to_register(byte data, int rclk, int ser, int srclk) {
  digitalWrite(rclk, LOW);
  shiftOut(ser, srclk, LSBFIRST, data);
  digitalWrite(rclk, HIGH);
}
```

Randarea Matricelor LED

Randarea unei matrici se face pe coloane. Refresh rate-ul este de 1000 * 24 µs = 24ms.

```
// Write to the 24x8 LED matrix using the shift registers
void write_mat(byte r, byte g, byte b, byte row) {
  digitalWrite(OE, HIGH);

  write_to_register(b, RCLK1, SER1, SRCLK1);
  write_to_register(g, RCLK1, SER1, SRCLK1);
  write_to_register(r, RCLK1, SER1, SRCLK1);

  write_to_register(row ^ 255, RCLK2, SER2, SRCLK2);

  digitalWrite(OE, LOW);
}

// Renders the next column of the matrix
void draw_column() {
  if(mat_line < 8) { // Draw a red column
    write_mat((1 << min(8, histogram[mat_line])) - 1, 0, 0, 1 << mat_line);
  } else if(mat_line < 16) { // Draw a green column
    write_mat(0, (1 << min(8, histogram[mat_line])) - 1, 0, 1 << (mat_line &
7));
  } else { // Draw a blue column
    write_mat(0, 0, (1 << min(8, histogram[mat_line])) - 1, 1 << (mat_line &
7));
  }

  mat_line++;
  if(mat_line == 24) { // Reset the rendering process
    mat_line = 0;
  }
}
```

Main Loop

În `loop` se citesc datele analog de la microfon și sunt transformate in spectograma utilizând funcția `fix_fft`.

```
void loop()
{
  static int i, j, step;
  int val;

  // get audio data
  for(i = 0; i < SAMPLES; i++) {
    val = analogRead(AUDIO); // 0-1023

    real[i] = (char)(val/4 - 128); // store the result on 8 bits
    imag[i] = 0; // set all imaginary parts to 0
  }
```

```
  // run FFT
  fix_fft(real, imag, LOG_SAMPLES, 0);

  histogram[0] = 0; // noise, so we ignore it

  // extract absolute value of data only, for half the results
  for(i = 1; i < SAMPLES/2; i++) {
    histogram[i] = (int)sqrt(real[i] * real[i] + imag[i] * imag[i]);
  }

  // compress the histogram to 24 values so it can be printed on the LED
matrices
  for(int i = 0; i < LED_COLS; i++) {
    // magic to make the spectogram look good and ignore low frequency
signals
    histogram[i] = (histogram[i * 2 + 6] + histogram[i * 2 + 7] + histogram[
i * 2 + 8]) * 1.33f;
  }
}
```

Rezultate Obţinute

Demo funcționalitate: link

Concluzii

Proiectul a fost finalizat cu succes.

Download

Codul sursa al proiectului poate fi găsit aici: Spectrum Analyser.

Bibliografie

Datasheets

- Matrice LED 8×8 3mm rosu: Datasheet;
- Matrice LED 8×8 3mm verde: Nu am gasit, dar are acelasi circuit ca celelalte doua, doar ca diodele sunt inversate;
- Matrice LED 8×8 3mm albastru: Datasheet;
- Shift register SN74HC595N: Datasheet;
- Rezistor 220Ω: Datasheet.

Resurse

- Setup Shift Registers: link;
- Spectrum Analyzer: link;
- Timer Interrupts: link.

Export to PDF