Dokumentacja Projektu

Piotr Ratajczak Alekasander Staszewski
Wojciech Tomaszewski Krzysztof Zaporowski

11 maja 2025

1 Wprowadzenie i specyfikacja wymagań

1.1 Cel projektu

Celem projektu jest stworzenie systemu robotycznego generującego iluzję optyczną poprzez wykorzystanie zjawiska Persistence of Vision (POV). System ma za zadanie wyświetlać tekst na obracającej się listwie

1.2 Wymagania funkcjonalne

- Wyświetlanie tekstu
- Precyzyjna synchronizacja wyświetlania diod LED z pozycją obrotową
- Stabilne utrzymanie iluzji optycznej

1.3 Wymagania techniczne

- Silnik o odpowiednim momencie obrotowym
- Pasek LED NeoPixel o wysokiej częstotliwości odświeżania
- Czujnik Halla do precyzyjnego określania pozycji kątowej
- Mikrokontroler z wystarczającą mocą obliczeniową do obsługi diod LED
- Stabilna konstrukcja mechaniczna zapewniająca płynny ruch

1.4 Ograniczenia i wymagania

- Minimalna prędkość obrotowa zapewniająca efekt POV
- Maksymalna masa komponentów na listwie
- Wymagana stabilność mechaniczna konstrukcji
- Odpowiednie zasilanie dla wszystkich komponentów

- Bezpieczeństwo użytkowania (ochrona przed dostępem do ruchomych części)

1.5 Opis pomysłu rozwiązania systemu komputerowego

System komputerowy składa się z mikrokontrolera ATtiny85, który pełni rolę głównego elementu sterującego. Jego zadaniem jest:

- Synchronizacja wyświetlania diod LED z pozycją obrotową listwy
- Przetwarzanie sygnałów z czujnika Halla
- Generowanie odpowiednich sekwencji świetlnych na pasku NeoPixel

2 Rozwiązania techniczne i opis konstrukcji

System opiera się na precyzyjnej synchronizacji między komponentami:

- Czujnik Halla wykrywa magnes zamontowany pod listwą, generując sygnał referencyjny dla każdego pełnego obrotu
- Mikrokontroler ATtiny85 wykorzystuje ten sygnał do synchronizacji wyświetlania diod LED
- Pasek NeoPixel, dzięki wysokiej częstotliwości odświeżania, tworzy płynny obraz
- Silnik krokowy zapewnia stałą prędkość obrotową, niezbędną dla stabilności iluzji

Techniczny aspekt iluzji optycznej:

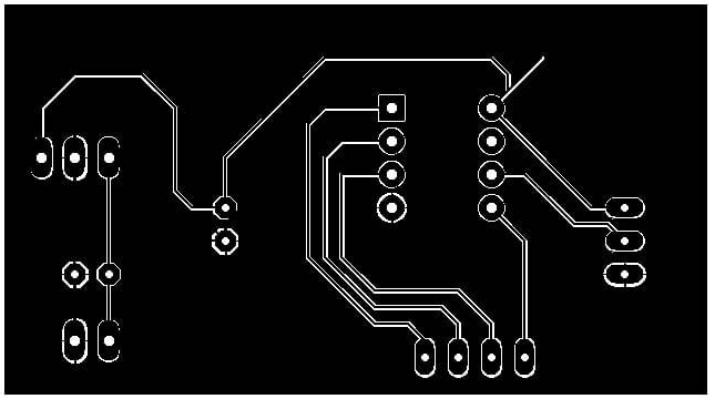
- Obraz jest generowany przez sekwencyjne włączanie diod LED w odpowiednich momentach obrotu
- Częstotliwość odświeżania diod jest dostosowana do prędkości obrotowej
- Precyzyjne obliczenia czasowe zapewniają prawidłowe wyświetlanie obrazu
- Efekt POV (Persistence of Vision) wykorzystuje bezwładność ludzkiego oka do tworzenia pozornie statycznego obrazu

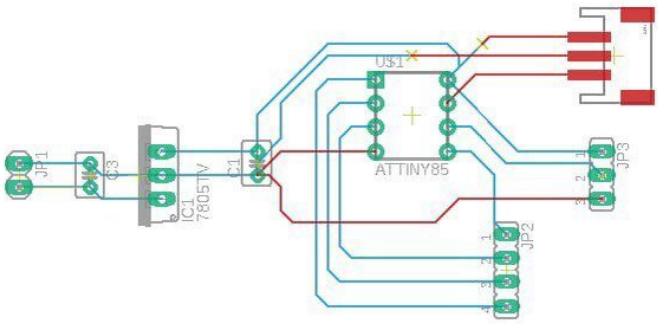
2.1 Rysunek płytki do trawienia

3 Oprogramowanie

3.1 Wykorzystane zasoby mikrokontrolera

W projekcie wykorzystano następujące zasoby mikrokontrolera:





- Pin cyfrowy 4 (DATA_PIN) - do sterowania paskiem LED NeoPixel
- Pin cyfrowy 3 (SENSOR_PIN) - do odczytu sygnału z czujnika Halla
- Pamięć RAM - do przechowywania bufora wyświetlania i danych fontu
- Pamięć Flash - do przechowywania programu i definicji znaków
- Timer - do precyzyjnego sterowania czasem wyświetlania

3.2 Kod źródłowy

3.2.1 Algorytm

Główny algorytm działania systemu opiera się na następujących elementach:

- Synchronizacja z czujnikiem Halla w funkcji `loop()`
- Wyświetlanie tekstu poprzez sekwencyjne odświetlanie kolumn znaków
- Obsługa fontu 5x8 pikseli zdefiniowanego w tablicy `font[]`

3.2.2 Obsługa przerwań

System wykorzystuje przerwania do:

- Wykrywania sygnału z czujnika Halla
- Synchronizacji wyświetlania z pozycją obrotową
- Precyzyjnego sterowania czasem świecenia diod LED

3.2.3 Obsługa czujników

- Czujnik Halla podłączony do pinu 3
- Ciągłe monitorowanie stanu czujnika w pętli głównej
- Synchronizacja wyświetlania z każdym pełnym obrotem

3.2.4 Obsługa elementów wykonawczych

- Sterowanie paskiem NeoPixel (8 diod) poprzez pin 4
- Biblioteka Adafruit_NeoPixel do obsługi diod LED
- Precyzyjne sterowanie kolorem i jasnością diod
- Optymalizacja czasu odświeżania (750 mikrosekund na kolumnę)

3.2.5 Przykładowy kod

```
// Definicje pinów i parametrów
#define NUM_PIXELS  8      // Liczba diod LED w pasku
#define DATA_PIN    4      // Pin do sterowania paskiem LED
#define SENSOR_PIN  3      // Pin do odczytu czujnika Halla

// Zakres obsługiwanych znaków (tylko wielkie litery)
#define ASCII_START 'A'    // Pierwszy obsługiwany znak
#define ASCII_END   'Z'    // Ostatni obsługiwany znak

// Inicjalizacja paska LED NeoPixel
Adafruit_NeoPixel strip(NUM_PIXELS, DATA_PIN, NEO_GRB + NEO_KHZ800);

// Parametry fontu
const int charHeight = 8;  // Wysokość znaku w pikselach
const int charWidth = 5;   // Szerokość znaku w pikselach

// Funkcja inicjalizująca
void setup() {
    pinMode(SENSOR_PIN, INPUT);    // Konfiguracja pinu czujnika jako wejście
    strip.begin();                 // Inicjalizacja paska LED
```

```
    strip.show();                   // Wyłączenie wszystkich diod
}

// Główna pętla programu
void loop() {
    // Oczekiwanie na sygnał z czujnika Halla
    while (digitalRead(SENSOR_PIN) != HIGH) {
        strip.clear();              // Czyszczenie bufora diod
        strip.show();               // Aktualizacja stanu diod
    }
    displayText("HELLO");           // Wyświetlenie tekstu
}

// Funkcja wyświetlająca tekst
void displayText(const char* text) {
    for (int i = 0; i < strlen(text); i++) {
        char ch = text[i];
        // Sprawdzenie czy znak jest w obsługiwanym zakresie
        if (ch >= ASCII_START && ch <= ASCII_END) {
            // Wyświetlenie każdej kolumny znaku
            for (int j = 0; j < charWidth; j++) {
                displayColumn(font[ch - ASCII_START][j]);
            }
            displayColumn(0x00);    // Dodanie odstępu między literami
        }
    }
}

// Funkcja wyświetlająca pojedynczą kolumnę znaku
void displayColumn(byte column, bool flipped = false) {
    strip.clear();                  // Czyszczenie bufora diod
    // Iteracja po wszystkich pikselach w kolumnie
    for (int i = 0; i < charHeight; i++) {
        // Odczyt stanu piksela (z uwzględnieniem odwrócenia)
        bool pixelOn = bitRead(column, flipped ? i : (7 - i));
        if (pixelOn) {
            // Ustawienie koloru diody (zielony)
            strip.setPixelColor(i, strip.Color(0, 150, 0));
        }
    }
    strip.show();                   // Aktualizacja stanu diod
    delayMicroseconds(750);         // Opóźnienie dla stabilności wyświetlania
}
```

4 Kosztorys

Część	Ilość sztuk	Miejsce zakupu	Cena
Silnik RS-550	1	Amazon	25.26 zł
Kondensator 0.33nF	1	Zasoby własne	0.00 zł
Kondensator 0.1mF	1	Zasoby własne	0.00 zł
Włącznik przelotowy	1	Botland	9.40 zł
Czujnik Halla	1	Botland	4.50 zł
Przewody	-	Zasoby własne	0.00 zł
Łącznik wału silnika	1	Botland	15.50 zł
Stabilizator napięcia L7805ABV	1	Botland	1.90 zł
Mikrokontroler ATtiny85	1	Botland	19.90 zł
NeoPixel 8	1	Botland	14.90 zł

Tabela 1: Kosztorys projektu