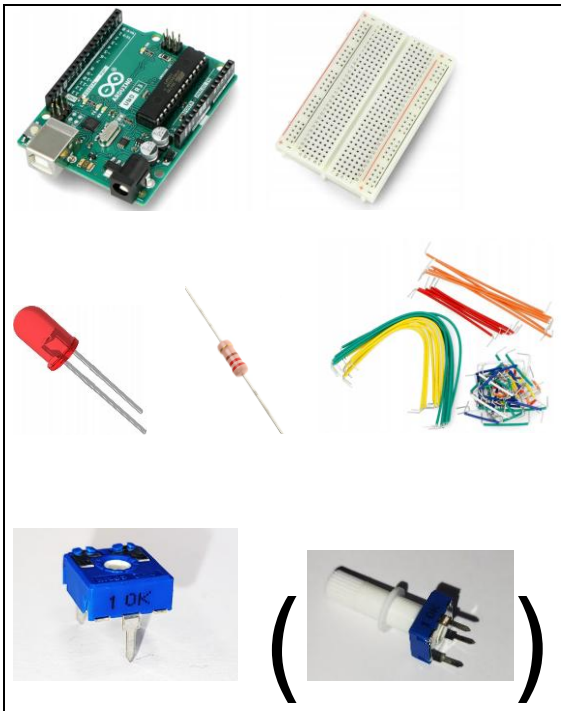
Ćwiczenie 28

Temat: Świat analogowy – Potencjometr i płynna zmiana jasności (PWM)

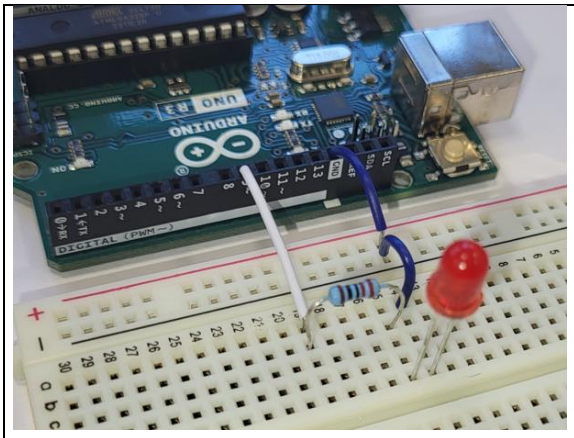
Cel ćwiczenia: Zrozumienie różnicy między sygnałem cyfrowym a analogowym. Nauczenie się, jak odczytywać wartości z potencjometru za pomocą wejścia analogowego (analogRead()) oraz jak sterować jasnością diody za pomocą sygnału PWM (analogWrite()).



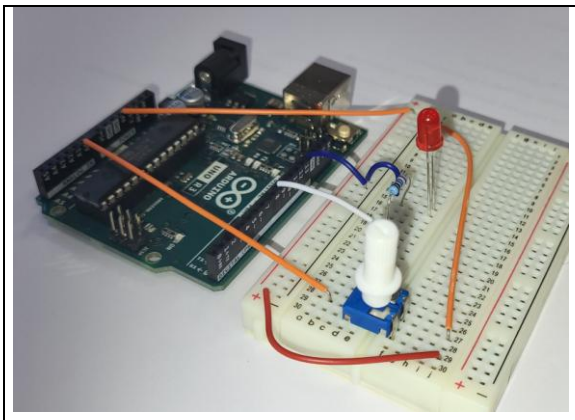
Potrzebne komponenty:

- Płytka Arduino UNO i płytka stykowa
- 1x Dioda LED
- 1x Rezystor 220Ω
- 1x Potencjometr 10kΩ **(z pokrętełkiem)**
- Przewody połączeniowe

Część 1: Podstawowy obwód i kod



- **Dioda LED:** Podłącz diodę LED do płytki stykowej wraz z rezystorem 220Ω. Katodę (krótsza nóżka) podłącz do **GND**. Anodę (dłuższa nóżka) przez rezystor podłącz do pinu cyfrowego **~9** na Arduino. Zwróć uwagę na symbol tyldy (**~**), który oznacza, że pin obsługuje PWM.



- **Potencjometr:** Wepnij potencjometr w płytkę stykową. Ma on trzy nóżki:

Jedną skrajną nóżkę podłącz do **5V** na Arduino.

Drugą skrajną nóżkę podłącz do **GND**.

Środkową nóżkę podłącz do pinu analogowego **A0** na Arduino. Dzięki takiemu podłączeniu, obracając gałką, na środkowej nóżce będzie pojawiać się napięcie od 0V do 5V, które Arduino odczyta jako wartość od 0 do 1023.

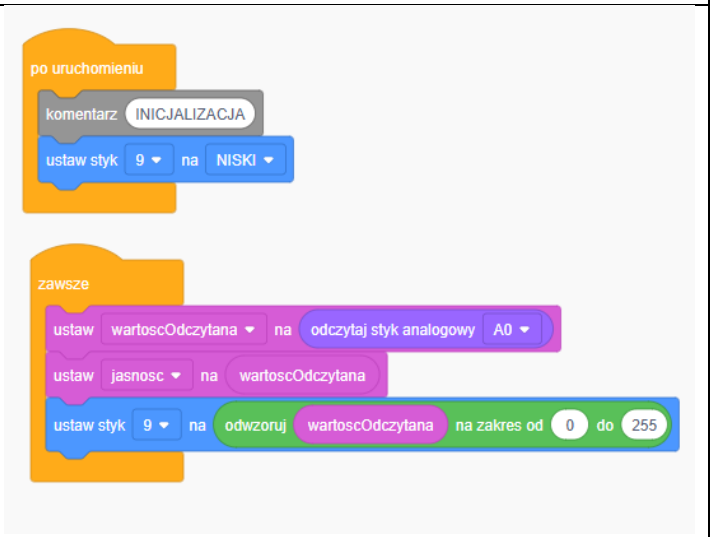
Kod

Program będzie w pętli odczytywał wartość z pinu A0, a następnie używał jej do ustawienia "mocy" świecenia diody na pinie ~9.

C++

```cpp
const int pinDioda = 9;       // Pin PWM dla diody
const int pinPotencjometr = A0; // Pin analogowy dla potencjometru

void setup() {
  pinMode(pinDioda, OUTPUT);
}

void loop() {
  // Odczytaj wartość z potencjometru (zakres 0-1023)
  int wartoscOdczytana = analogRead(pinPotencjometr);

  // Przeskaluj wartość na zakres PWM (0-255)
  // Funkcja map() robi to za nas: map(wartość, od_min, od_max, do_min, do_max)
  int jasnosc = map(wartoscOdczytana, 0, 1023, 0, 255);

  // Ustaw jasność diody
  analogWrite(pinDioda, jasnosc);

  delay(10); // Małe opóźnienie dla stabilności
}
```

Thinkercad Scratch



Część 2: Samodzielna rozbudowa

Twoim celem jest rozbudowanie automatu o przycisk dla pieszych, który zmienia cykl świateł.

Zadanie 28.1: Zmodyfikuj kod tak, aby dioda działała odwrotnie – kręcenie gałką w prawo będzie ją ściemniać, a w lewo rozjaśniać.
Zadanie 28.2: Dodaj do obwodu **drugą diodę LED** na pinie **~10**. Napisz program, w którym obie diody działają jak "przenikanie" (crossfade): gdy jedna gaśnie, druga się rozjaśnia i na odwrót.
Zadanie 28.3: Zamiast jasności, steruj **częstotliwością mrugania** diody. Przekształć odczytaną wartość z potencjometru tak, aby sterowała czasem w funkcji delay().
Zadanie 28.4: Stwórz prosty **tester refleksu**. Program ma w losowym momencie włączyć diodę. Zadaniem użytkownika jest jak najszybsze wciśnięcie przycisku (dodaj go do obwodu). Potencjometr służy do ustawiania poziomu trudności (np. maksymalnego czasu na reakcję).

Pliki: [zad28-1.INO, zad28-2.INO, zad28-3.INO, zad28-4.INO] przesłać na haukeprojekt@gmail.com , w tytule podać „**3XY(Z)_cwiczenie26**” gdzie X=klasa, Y=grupa, Z=numer_z_dziennika

28.4 – Rozszerzona instrukcja:

Jasne, wyjaśnijmy dokładnie, na czym polega pomysł na "tester refleksu". To świetne ćwiczenie, ponieważ łączy w sobie wszystko, czego uczniowie nauczyli się do tej pory: losowość, odczyt cyfrowy (przycisk), odczyt analogowy (potencjometr) i sterowanie wyjściem (dioda).

Idea gry 🎮

Celem jest stworzenie prostej gry zręcznościowej. Komputer (Arduino) testuje, jak szybko uczeń jest w stanie zareagować na sygnał wzrokowy.

1. **Czekanie:** Program czeka przez losowy okres czasu (np. od 2 do 7 sekund). To kluczowe, aby gracz nie mógł przewidzieć, kiedy dokładnie zapali się dioda.
2. **Sygnał:** Nagle zapala się dioda LED. To jest sygnał "START!" dla gracza. W tym samym momencie Arduino uruchamia wewnętrzny stoper.
3. **Reakcja:** Gracz musi jak najszybciej nacisnąć przycisk.
4. **Wynik:** Gdy tylko przycisk zostanie wciśnięty, Arduino zatrzymuje stoper. Następnie porównuje zmierzony czas reakcji z "poziomem trudności" ustawionym za pomocą potencjometru.
5. **Informacja zwrotna:** Program informuje gracza o wyniku, np. zapalając inną diodę na zielono (sukces) lub czerwono (porażka).

Rola poszczególnych komponentów

- **Dioda LED (np. żółta):** To główny sygnał dla gracza. Jej zapalenie się rozpoczyna pomiar czasu.
- **Przycisk:** To "buzzer" gracza. Jego naciśnięcie kończy pomiar czasu.

- **Potencjometr:** Działa jak **pokrętło do regulacji poziomu trudności**.
    - Gdy jest skręcony w jedną stronę (np. w lewo), gracz ma dużo czasu na reakcję (np. całą sekundę, czyli 1000 milisekund). To jest **poziom łatwy**.
    - Gdy jest skręcony w drugą stronę (np. w prawo), czas na reakcję jest bardzo krótki (np. ćwierć sekundy, czyli 250 milisekund). To jest **poziom trudny**.

Jak to działa w kodzie? (Logika programu)

Program można podzielić na kilka etapów (stanów):

1. **Stan PRZYGOTOWANIE:**
    - Program losuje czas oczekiwania (np. liczbę milisekund od 2000 do 7000).
    - Czeka w miejscu przez wylosowany czas. W tym czasie dioda LED jest zgaszona.
2. **Stan POMIAR_CZASU:**
    - Po odczekaniu, program zapala diodę LED.
    - Natychmiast zapisuje aktualny czas z wewnętrznego zegara Arduino za pomocą funkcji millis(). To jest nasz "czas startu".
    - Program wchodzi w pętlę i czeka, aż stan przycisku zmieni się na "wciśnięty".
3. **Stan WYNIK:**
    - Gdy tylko program wykryje wciśnięcie przycisku, ponownie zapisuje aktualny czas z millis(). To jest "czas stopu".
    - Oblicza czas reakcji: czasReakcji = czasStopu - czasStartu.
    - Odczytuje wartość z potencjometru (0-1023) i **odwzorowuje (mapuje)** ją na zakres czasu, np. od 1000 ms (łatwy) do 250 ms (trudny). Nazwijmy to maksymalnyCzas.
    - Porównuje wyniki:
        - **JEŚLI (czasReakcji <= maksymalnyCzas)**: Gracz zdążył! Program może np. na 2 sekundy zapalić zieloną diodę.
        - **W PRZECIWNYM RAZIE**: Gracz się spóźnił. Program może na 2 sekundy zapalić czerwoną diodę.
    - Po wyświetleniu wyniku program wraca do stanu PRZYGOTOWANIE i gra zaczyna się od nowa.

To ćwiczenie jest doskonałym podsumowaniem dotychczasowej wiedzy i wprowadza uczniów w myślenie o programowaniu jako o zarządzaniu różnymi "stanami" automatu.