Ćwiczenie 30

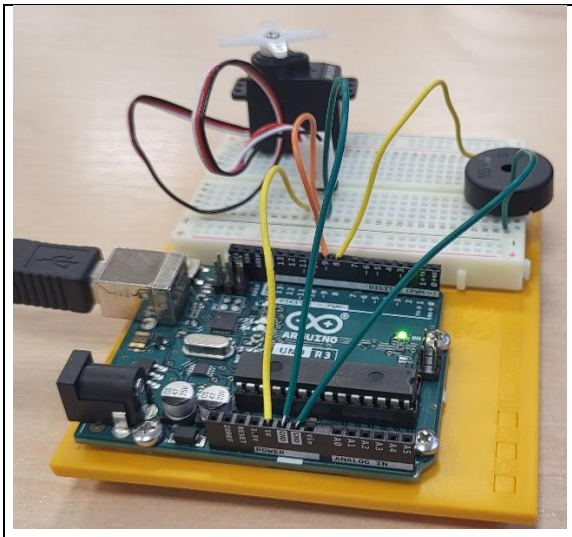
Temat: Dźwięk i ruch – Serwomechanizmy i biblioteki

Cel ćwiczenia: Nauczenie się, jak sterować bardziej złożonymi komponentami, takimi jak serwomechanizm, poprzez wykorzystanie gotowych **bibliotek**. Wprowadzenie do generowania dźwięku.

Potrzebne komponenty:

- Płytka Arduino UNO i płytka stykowa
- 1x Serwomechanizm (SG90)
- 1x Brzęczyk (Piezo Buzzer)
- 1x Potencjometr
- Przewody połączeniowe

Część 1: Podstawowy obwód i kod



Zaprogramujemy serwomechanizm tak, aby poruszał się wahadłowo w zakresie od 0 do 180 stopni.

Obwód

1. **Serwomechanizm:** Posiada trzy przewody:
   - **Czerwony** → podłącz do **5V**.
   - **Brązowy/Czarny** → podłącz do **GND**.
   - **Pomarańczowy/Biały (sygnałowy)** → podłącz do pinu cyfrowego **~9**.
2. **Brzęczyk (Buzzer):**
   - jedną nóżkę (+) podłącz do pinu cyfrowego 8.
   - drugą nóżkę (-) podłącz do GND.

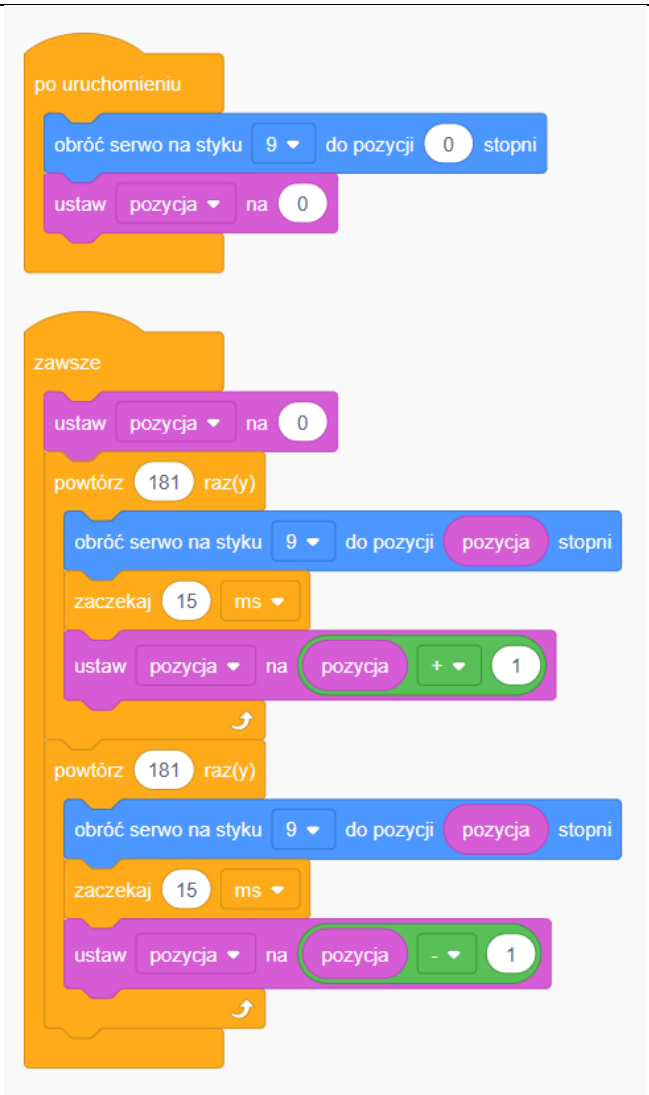
Kod

Do sterowania serwem użyjemy biblioteki **Servo.h**, która jest wbudowana w Arduino IDE. Dzięki niej nie musimy sami generować skomplikowanych sygnałów **PWM**.

C++

```cpp
#include <Servo.h> // Dołącz bibliotekę do obsługi serwomechanizmów

Servo mojeServwo;  // Stwórz obiekt serwa, którym będziemy sterować
int pozycja = 0;  // Zmienna do przechowywania aktualnej pozycji serwa

void setup() {
  mojeServwo.attach(9); // Przypisz serwo do pinu 9
}

void loop() {
  // Ruch od 0 do 180 stopni
  for (pozycja = 0; pozycja <= 180; pozycja += 1) {
    mojeServwo.write(pozycja); // Ustaw serwo w danej pozycji
    delay(15);                 // Poczekaj chwilę, aby serwo zdążyło się obrócić
  }

  // Ruch od 180 do 0 stopni
  for (pozycja = 180; pozycja >= 0; pozycja -= 1) {
    mojeServwo.write(pozycja);
    delay(15);
  }
}
```

Thinkercad Scratch



Część 2: Samodzielna rozbudowa

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

Zadanie 30.1: Dodaj do obwodu **potencjometr** (jak w ćwiczeniu 4). Napisz kod, który pozwoli na **ręczne sterowanie pozycją serwomechanizmu** za pomocą obracania gałki.
Zadanie 30.2: Stwórz prosty **instrument muzyczny**. Dodaj do obwodu 3..4 przyciski. Każdy przycisk po naciśnięciu ma odtwarzać inną nutę na brzęczyku za pomocą funkcji **tone()** **(przyciski + R=10kΩ)**
Zadanie 30.3: Zbuduj **miniaturowy szlaban parkingowy**. Po naciśnięciu przycisku, serwo ma powoli podnieść się do pozycji 90 stopni, odczekać 5 sekund, a następnie powoli opaść
Zadanie 30.4: Skonstruuj **wizualno-dźwiękowy radar**. Serwo porusza się wahadłowo. Dołącz czujnik odległości (jeśli jest w zestawie) lub symuluj wykrycie obiektu potencjometrem. Gdy "obiekt" jest blisko, dioda LED ma szybko mrugać, a brzęczyk wydawać wysokie, piskliwe dźwięki. Im dalej obiekt, tym wolniejsze mruganie i niższy dźwięk.

Pliki: [zad30-1.INO, zad30-2.INO, zad30-3.INO, zad30-4.INO] przesłać na haukeprojekt@gmail.com ,
w tytule podać „**3XY(Z)_cwiczenie30**” gdzie X=klasa, Y=grupa, Z=numer_z_dziennika

30.4 – Rozszerzona instrukcja:

Zadanie 30.4: Wizualno-dźwiękowy radar sonarowy

Cel ćwiczenia: Zbudowanie w pełni funkcjonalnego prototypu sonaru. Urządzenie będzie używać serwomechanizmu do "skanowania" otoczenia, czujnika ultradźwiękowego do pomiaru odległości od przeszkód, a diody LED i brzęczyka do sygnalizowania, jak blisko znajduje się wykryty obiekt. To ćwiczenie integruje całą zdobytą dotychczas wiedzę.

Potrzebne komponenty:

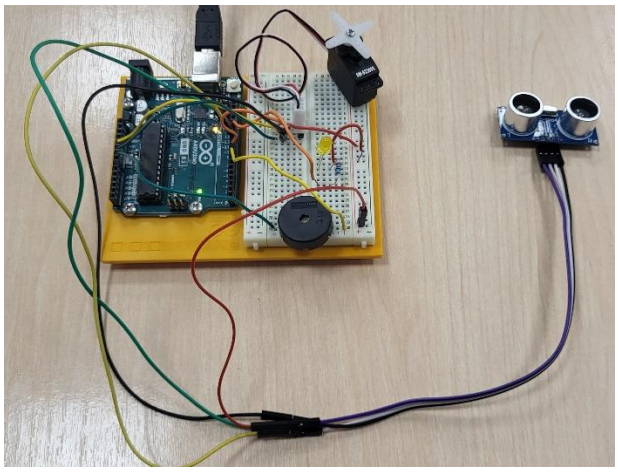
- Płytka Arduino UNO i płytka stykowa
- 1x Serwomechanizm (SG90)
- 1x Ultradźwiękowy czujnik odległości (HC-SR04)
- 1x Dioda LED (dowolny kolor)
- 1x Rezystor 220Ω
- 1x Brzęczyk (Piezo Buzzer)
- Przewody połączeniowe

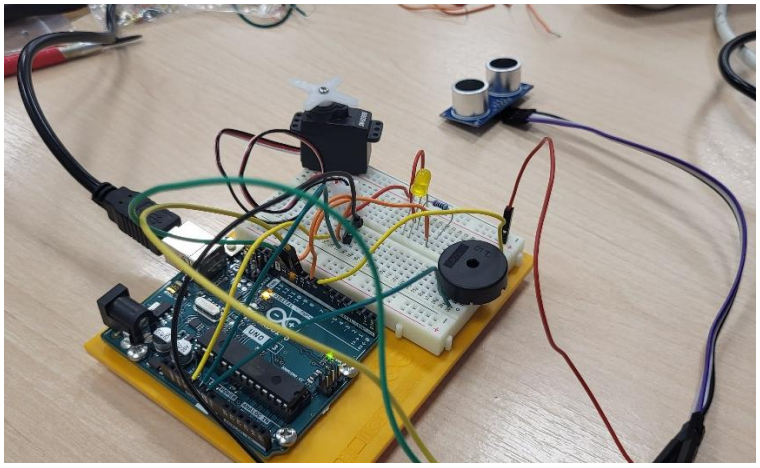
Część 1: Obwód – montujemy nasz sonar

Zbudujemy teraz kompletny układ. Zwróć uwagę na prawidłowe podłączenie każdego z elementów.

1. **Serwomechanizm:** Podłącz go tak jak w ćwiczeniu 30:
    - **Czerwony** przewód (zasilanie) -> pin **5V** na Arduino.
    - **Brązowy/Czarny** przewód (masa) -> pin **GND** na Arduino.
    - **Pomarańczowy/Żółty** przewód (sygnał) -> pin cyfrowy **~9**.
2. **Ultradźwiękowy czujnik odległości (HC-SR04):** Ten czujnik ma cztery piny:
    - **VCC** (zasilanie) -> pin **5V** na Arduino.
    - **GND** (masa) -> pin **GND** na Arduino.
    - **Trig** (Trigger - "spust") -> pin cyfrowy **12**. Ten pin służy do wysyłania impulsu dźwiękowego.
    - **Echo** (Echo) -> pin cyfrowy **11**. Ten pin "nasłuchuje" powrotu odbitego echa.
3. **Dioda LED (sygnalizator wizualny):**
    - Podłącz dłuższą nóżkę (anodę) przez rezystor **220Ω** do pinu cyfrowego **13**.
    - Krótszą nóżkę (katodę) podłącz do pinu **GND**.
4. **Brzęczyk (sygnalizator dźwiękowy):**
    - Dłuższą nóżkę (+) podłącz do pinu cyfrowego **8**.
    - Krótszą nóżkę (-) podłącz do **GND**.

Wskazówka: Dobrym pomysłem jest fizyczne przymocowanie czujnika ultradźwiękowego do orczyka (ramienia) serwomechanizmu, np. za pomocą taśmy dwustronnej. Dzięki temu czujnik będzie obracał się razem z serwem, skanując otoczenie.





Część 2: Jak to działa? – Koncepcja radaru

1. **Skanowanie:** Serwomechanizm będzie powoli obracał się w zakresie od 0 do 180 stopni i z powrotem. To symuluje ruch obrotowej anteny radaru.
2. **Pomiar:** W każdym punkcie obrotu (lub co kilka stopni) czujnik HC-SR04 wyśle krótką, niesłyszalną dla człowieka falę dźwiękową. Następnie będzie czekał, aż fala odbije się od przeszkody i wróci.
3. **Obliczenia:** Arduino zmierzy czas, jaki upłynął od wysłania fali do jej powrotu. Znając prędkość dźwięku w powietrzu (ok. 343 m/s), program będzie w stanie precyzyjnie obliczyć odległość od przeszkody.
4. **Sygnalizacja:** Tutaj dzieje się cała magia. Zmierzona odległość zostanie przekształcona na sygnały dla użytkownika:
    - **Im bliżej jest obiekt**, tym **szybciej będzie mrugać dioda LED** i tym **wyższy (bardziej piskliwy) dźwięk** wyda brzęczyk.
    - **Im dalej jest obiekt**, tym **wolniej będzie mrugać dioda** i tym **niższy dźwięk** usłyszymy.

Część 3: Logika programu – plan działania

Zanim zaczniesz pisać kod, przemyśl jego strukturę. Oto plan, który możesz zrealizować krok po kroku.

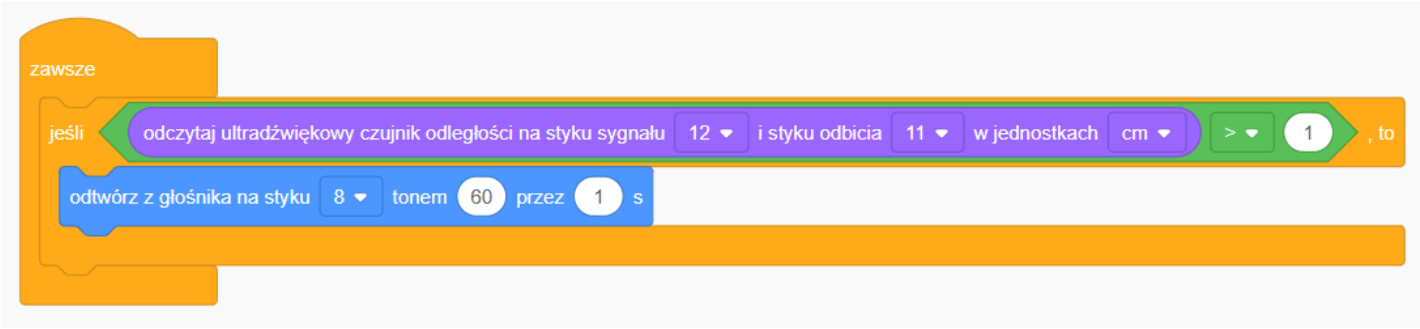
1. **Setup (Konfiguracja):**
    - Dołącz bibliotekę <Servo.h>.
    - Zdefiniuj stałe dla wszystkich pinów (serwo, trig, echo, dioda, brzęczyk).
    - Stwórz obiekt serwa.
    - W funkcji setup():
        - Przypisz serwo do pinu 9 (mojeSerwo.attach(9)).
        - Ustaw tryby pinów: Trig jako OUTPUT, Echo jako INPUT, a piny diody i brzęczyka jako OUTPUT.
        - Opcjonalnie uruchom Monitor Szeregowy (Serial.begin(9600)), aby móc wyświetlać zmierzone odległości na komputerze.
2. **Loop (Główna pętla):**
    - Stwórz pętlę for, która będzie przesuwać serwo od 0 do 180 stopni.
    - **Wewnątrz tej pętli** dla każdego kroku (lub co 5 kroków, żeby nie było zbyt gęsto):
        - Ustaw serwo w aktualnej pozycji (mojeSerwo.write(pozycja)).
        - **Wykonaj pomiar odległości:**
            - Wyślij krótki impuls na pin Trig (ustaw go na HIGH na 10 mikrosekund, a potem na LOW).
            - Użyj funkcji pulseIn(pinEcho, HIGH), aby zmierzyć czas powrotu echa w mikrosekundach.

        - Przelicz ten czas na centymetry. **Wzór: odleglosc = czas * 0.034 / 2;** (mnożymy przez prędkość dźwięku w cm/µs i dzielimy przez 2, bo fala pokonuje drogę w obie strony).
    - **Przekształć odległość na sygnały:**
        - Użyj funkcji map(), aby **odwzorować** zmierzoną odległość (np. z zakresu 2-200 cm) na **opóźnienie mrugania** diody (np. z zakresu 1000-50 ms).
        - Użyj map() ponownie, aby **odwzorować** tę samą odległość na **częstotliwość dźwięku** (np. z zakresu 100-1500 Hz).
    - **Wygeneruj sygnały:**
        - Włącz diodę, odczekaj obliczone opóźnienie, wyłącz diodę.
        - Użyj funkcji tone(pinBuzzer, obliczonaCzestotliwosc, 100), aby brzęczyk wydał krótki dźwięk o odpowiedniej wysokości.
    - Dodaj małe opóźnienie na końcu pętli (delay(50)), aby cały ruch nie był zbyt gwałtowny.
  - Stwórz **drugą pętlę for**, która będzie robić to samo, ale przesuwając serwo z powrotem od 180 do 0 stopni.

Podpowiedź do kodu (C++):

```
// Fragment kodu do pomiaru odległości
digitalWrite(pinTrig, LOW);
delayMicroseconds(2);
digitalWrite(pinTrig, HIGH);
delayMicroseconds(10);
digitalWrite(pinTrig, LOW);

long czas = pulseIn(pinEcho, HIGH);
int odleglosc = czas * 0.034 / 2;
```

Podpowiedź do kodu (Bloki Tinkercad): W kategorii **Wejście (Input)** znajdziesz gotowy fioletowy blok **odczytaj czujnik odległości ultradźwiękowej...**, który sam wykonuje wszystkie obliczenia i od razu zwraca odległość w centymetrach! To znacznie upraszcza programowanie.



Powodzenia! Ten projekt daje ogromną satysfakcję i jest świetnym przykładem, jak działają prawdziwe systemy sensoryczne w robotyce.