Podstawy Systemów Wbudowanych Laboratorium

Laboratorium 4: Aktuatory i układ sterowania — serwomechanizm i czujnik odległości

Autorzy: Angelika Tefelska, Dariusz Tefelski

Spis treści

1 Wprowadzenie **1**

2 Wyposażenie stanowiska **1**

3 Zadanie 1: Sterowanie serwomechanizmem **2**
3.1 Jak sterowany jest serwomechanizm? 2
3.2 Biblioteka Servo 3
3.3 Joystick jako element sterujący 4

4 Zadanie 2: Pomiar odległości czujnikiem HC-SR04 **4**
4.1 Zasada pomiaru czasu przelotu 4

5 Zadanie 3: Automatyczna bariera parkingowa **6**

6 Podsumowanie **6**

1 Wprowadzenie

Cel ćwiczenia

Celem ćwiczenia jest poznanie sposobu sterowania aktuatorami oraz zbudowanie prostego układu reagującego na informacje z czujnika. W pierwszej części wykorzystamy serwomechanizm i joystick, a następnie czujnik ultradźwiękowy HC-SR04. Na końcu połączymy oba elementy w prosty automatyczny system.

W poprzednich laboratoriach mikrokontroler przede wszystkim **odczytywał informacje**: stan przycisku, napięcie z czujnika analogowego, dane z GPS, czujników I^2C oraz czytnika RFID. W praktycznym systemie wbudowanym równie ważne jest oddziaływanie na otoczenie.

Element wykonawczy, który zamienia sygnał sterujący na działanie fizyczne, nazywamy **aktuatorem**. Przykładami są silniki, serwomechanizmy, przekaźniki, zawory, grzałki czy źródła światła.

Czujnik i aktuator

	Czujnik	Aktuator
Rola	dostarcza informacji	wykonuje działanie
Przykład	HC-SR04, BME280	serwomechanizm
Kierunek informacji	otoczenie → mikrokontroler	mikrokontroler → otoczenie

W końcowej części laboratorium wykorzystamy oba typy elementów jednocześnie: czujnik będzie dostarczał informacji o odległości, a serwomechanizm będzie reagował na wynik pomiaru.

2 Wyposażenie stanowiska

Do wykonania ćwiczenia potrzebne są:

- Arduino UNO R4 WiFi,
- przewód USB,
- serwomechanizm Redox S90,
- joystick analogowy,
- ultradźwiękowy czujnik odległości HC-SR04,
- Analog Discovery II z przewodami pomiarowymi,
- płytka stykowa i przewody połączeniowe,

- opcjonalnie dioda LED z rezystorem do sygnalizacji stanu układu.

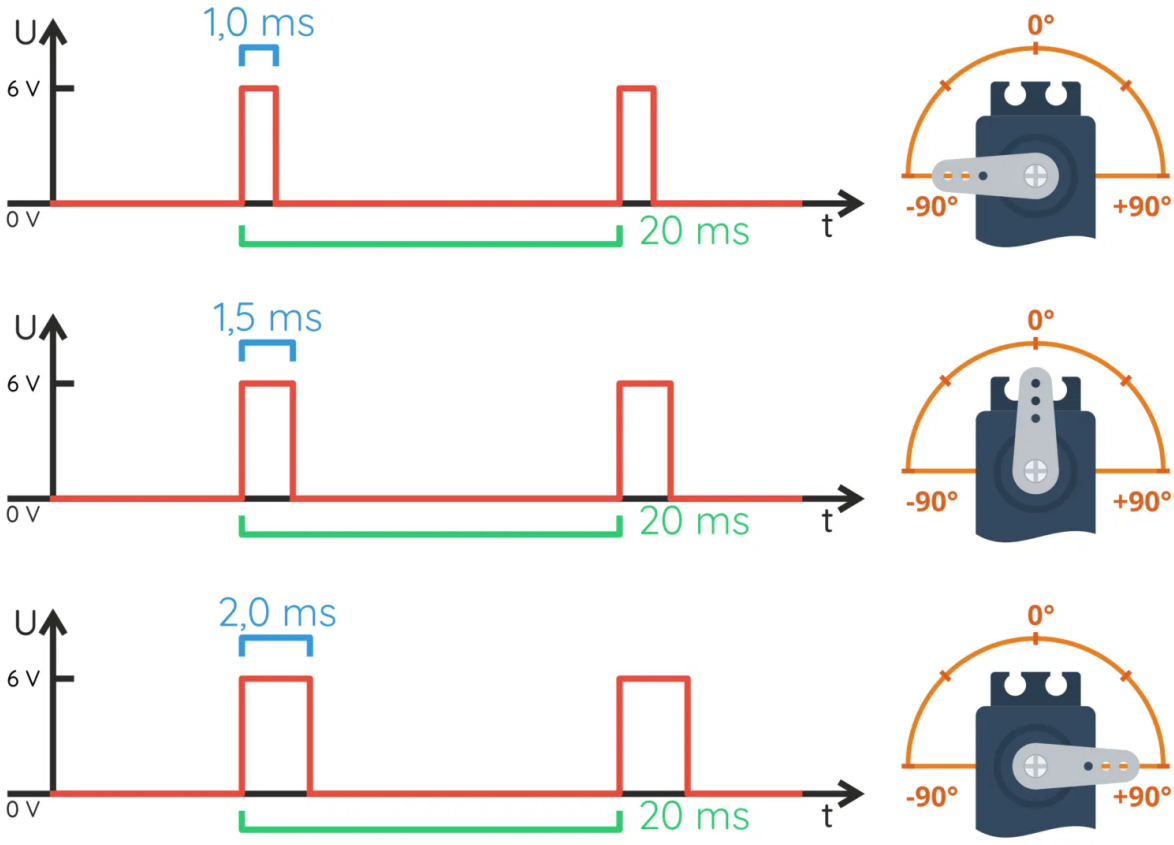
Co należy zapamiętać?

Połączenia elementów wykonuj przy odłączonym zasilaniu Arduino. Serwomechanizm może pobierać znacznie większy prąd niż typowy układ logiczny. Jeżeli podczas pracy serwa Arduino resetuje się lub serwo pracuje niestabilnie, zgłoś to prowadzącemu — nie próbuj zasilać serwa bezpośrednio z pinu GPIO.

3 Zadanie 1: Sterowanie serwomechanizmem

3.1 Jak sterowany jest serwomechanizm?

Typowy serwomechanizm modelarski zawiera silnik, przekładnię, czujnik położenia oraz wewnętrzny układ regulacji. Mikrokontroler nie steruje bezpośrednio silnikiem znajdującym się w serwie. Zamiast tego wysyła okresowy impuls sterujący, którego szerokość odpowiada żądanemu położeniu wału.



Rysunek 1: Idea sterowania położeniem serwomechanizmu (na obrazku zaznaczono U=6V, natomiast w naszym przypadku U=3,3V). Źródło obrazka: https://forbot.pl/blog/leksykon/serwomechanizm.

Dla wielu serwomechanizmów impulsy są powtarzane mniej więcej co 20 ms. Typowo krótszy impuls odpowiada jednemu skrajnemu położeniu, impuls około 1,5 ms położeniu środkowemu, a dłuższy impuls drugiemu skrajnemu położeniu. Dokładne wartości zależą od konkretnego modelu serwomechanizmu.

3.2 Biblioteka Servo

Biblioteka `Servo` należy do standardowych bibliotek Arduino i zawiera przykłady pokazujące podstawowe użycie serwomechanizmu.

Zadanie 1.1 — Pierwsze sterowanie serwomechanizmem

Przy odłączonym zasilaniu podłącz serwomechanizm:

Serwomechanizm	**Arduino UNO R4 WiFi**
GND / przewód czarny lub brązowy	GND
VCC / przewód czerwony	5V
sygnał / przewód biały, żółty lub pomarańczowy	`D6`

W Arduino IDE wybierz *Plik* → *Przykłady* → *Servo* i otwórz przykład `Sweep`. Przeanalizuj program i znajdź:

- dołączenie biblioteki,
- utworzenie obiektu reprezentującego serwomechanizm,
- funkcję przypisującą pin sterujący,
- funkcję ustawiającą żądany kąt.

Na podstawie przykładu przygotuj program, który kolejno ustawia serwomechanizm w trzech położeniach: 0° → 90° → 180° → 90° → 0°. Pomiędzy zmianami pozostaw około 1 s przerwy.

Sprawdź, czy wszystkie trzy położenia są osiągane poprawnie. Nie wymuszaj mechanicznie ruchu poza zakresem serwa.

Pin serwa nie musi być oznaczony symbolem PWM

Biblioteka `Servo` wykorzystuje zasoby sprzętowe mikrokontrolera do generowania przebiegu sterującego. Dlatego sygnału serwa nie trzeba koniecznie podłączać do pinu oznaczonego na płytce symbolem `~`.

Pomiar 1 — Sygnał sterujący serwomechanizmem

Podłącz wejście `1+` Analog Discovery II do pinu sygnałowego serwa, a wejście `1-` do GND Arduino. Uruchom *WaveForms* → *Scope*.

Dla położeń 0°, 90° i 180°:

1. zmierz okres powtarzania impulsów,
2. zmierz czas trwania stanu wysokiego,

3. porównaj wyniki dla trzech kątów.

Czy wraz ze zmianą zadanego kąta wyraźniej zmienia się okres przebiegu, czy szerokość pojedynczego impulsu?

3.3 Joystick jako element sterujący

Joystick znajdujący się na stanowisku zawiera dwa potencjometry. Napięcia na wyjściach `VRX` oraz `VRY` zależą od wychylenia drążka w dwóch prostopadłych kierunkach. Dla Arduino są więc zwykłymi sygnałami analogowymi.

Zadanie 1.2 — Sterowanie położeniem serwa joystickiem

Przy odłączonym zasilaniu podłącz joystick:

Joystick	Arduino UNO R4 WiFi
VCC	5V
GND	GND
VRX	`A0`
VRY	`A1`

Najpierw napisz krótki program, który wyświetla w *Serial Monitor* wartości odczytane z `A0` i `A1`. Określ w przybliżeniu wartości minimalne, środkowe i maksymalne dla obu osi.

Następnie wykorzystaj jedną wybraną oś joysticka do ustawiania położenia serwomechanizmu. Przeskaluj wartość ADC na bezpieczny zakres kąta, np. od 0° do 180°.

Możesz wykorzystać funkcję:

```
int angle = map(raw, rawMin, rawMax, 0, 180);
angle = constrain(angle, 0, 180);
```

Program powinien na bieżąco wyświetlać w monitorze portu szeregowego wartość ADC oraz wyliczony kąt.

4 Zadanie 2: Pomiar odległości czujnikiem HC-SR04

4.1 Zasada pomiaru czasu przelotu

Czujnik HC-SR04 mierzy odległość na podstawie czasu przelotu fali ultradźwiękowej. Arduino wysyła krótki impuls na wejście `TRIG`. Czujnik emituje falę ultradźwiękową, a następnie ustawia wyjście `ECHO` w stan wysoki na czas odpowiadający drodze fali do przeszkody i z powrotem.

Jeżeli t jest czasem trwania impulsu `ECHO`, a v prędkością dźwięku, odległość od przeszkody wynosi:

$$d = \frac{v\,t}{2}. \tag{1}$$

Dla prędkości dźwięku około 0,0343 cm/μs otrzymujemy:

$$d[\text{cm}] \approx \frac{0{,}0343\, t[\mu\text{s}]}{2}. \tag{2}$$

Dzielimy przez 2, ponieważ fala pokonuje drogę od czujnika do przeszkody, a następnie wraca do czujnika.

Zadanie 2.1 — Pomiar odległości

Przy odłączonym zasilaniu podłącz HC-SR04:

HC-SR04	Arduino UNO R4 WiFi
VCC	5V
GND	GND
TRIG	D7
ECHO	D8

Napisz funkcję:

```
float measureDistanceCm() {
    // 1. wygeneruj impuls TRIG o czasie 10 us
    // 2. zmierz czas stanu HIGH na ECHO funkcja pulseIn()
    // 3. przelicz czas na odleglosc w cm
    // 4. zwroc wynik
}
```

W funkcji pamiętaj, aby przed impulsem wyzwalającym ustawić `TRIG` na chwilę w stan niski. Funkcja `pulseIn(pin, HIGH)` zwraca czas trwania stanu wysokiego w mikrosekundach.
Wyświetlaj zmierzoną odległość w *Serial Monitor*. Sprawdź wskazania dla kilku odległości.

Pomiar nie zawsze musi się udać

Jeżeli czujnik nie odbierze echa, funkcja `pulseIn()` może oczekiwać bardzo długo. W praktycznym programie warto zastosować limit czasu, np. `pulseIn(ECHO_PIN, HIGH, 30000)`, i odpowiednio obsłużyć wynik równy 0.

Pomiar 2 — Impulsy TRIG i ECHO

Jeżeli czas na to pozwala, wykorzystaj dwa kanały Analog Discovery II:

- kanał 1 do obserwacji sygnału `TRIG`,
- kanał 2 do obserwacji sygnału `ECHO`.

Ustaw przeszkodę w wybranej odległości i zmierz szerokość impulsu `ECHO`. Na podstawie zmierzonego czasu oblicz odległość ze wzoru podanego wcześniej i porównaj ją z wynikiem programu Arduino.

5 Zadanie 3: Automatyczna bariera parkingowa

W poprzednich zadaniach serwomechanizm i czujnik odległości pracowały niezależnie. Teraz połączymy je w prosty system reagujący na zmianę otoczenia.

Zadanie 3.1 — Bariera sterowana odległością

Zbuduj program realizujący automatyczną barierę parkingową.
Przyjmij:

- położenie `CLOSED` — serwo około 20°,
- położenie `OPEN` — serwo około 90°,
- próg otwarcia — przeszkoda bliżej niż 20 cm,
- próg zamknięcia — przeszkoda dalej niż 30 cm.

Program powinien:

1. wykonywać pomiar odległości co około 100 ms,
2. otworzyć barierę, gdy obiekt znajdzie się bliżej niż 20 cm,
3. pozostawić barierę otwartą, dopóki obiekt znajduje się w pobliżu,
4. zamknąć barierę dopiero po zwiększeniu odległości powyżej 30 cm,
5. wyświetlać aktualną odległość i stan `OPEN`/`CLOSED` w *Serial Monitor*.

Do okresowego wykonywania pomiarów wykorzystaj `millis()` poznane w Laboratorium 1. Nie stosuj długich opóźnień `delay()` w głównej logice programu. Krótkie `delayMicroseconds()` potrzebne do wygenerowania impulsu TRIG są dozwolone.

6 Podsumowanie

Po tych zajęciach powinieneś umieć

- wyjaśnić różnicę pomiędzy czujnikiem a aktuatorem,
- odnaleźć i przeanalizować przykład biblioteki `Servo`,
- ustawić położenie serwomechanizmu i sterować nim za pomocą joysticka,
- wykorzystać Analog Discovery II do obserwacji sygnału sterującego serwem,
- wygenerować impuls wyzwalający dla HC-SR04,
- zmierzyć czas impulsu funkcją `pulseIn()` i przeliczyć go na odległość,
- zastosować limit czasu w funkcji `pulseIn()`,
- połączyć czujnik i aktuator w prostym układzie automatycznego sterowania,
- zastosować histerezę, aby ograniczyć niepożądane przełączanie stanu.

Co należy zapamiętać?

Przed opuszczeniem stanowiska odłącz zasilanie Arduino, odłącz serwomechanizm i czujnik HC-SR04 oraz uporządkuj przewody i elementy.