Podstawy Systemów Mikroprocesorowych Laboratorium

Laboratorium 3: ADC i PWM — pomiar i sterowanie

Autorzy: Dariusz Tefelski, Angelika Tefelska

Spis treści

1 Wprowadzenie **1**

2 Wyposażenie stanowiska **1**

3 Zadanie 1: Przetwornik analogowo-cyfrowy ADC **2**
3.1 Pomiar napięcia z potencjometru 2

4 Zadanie 2: Powtarzalność pomiaru i bufor cykliczny **3**

5 Zadanie 3: Modulacja PWM **4**

6 Zadanie 4: Sterowanie serwomechanizmem **6**

7 Zadanie 5: PWM jako źródło napięcia quasi-analogowego **7**

8 Podsumowanie **9**

1 Wprowadzenie

Cel ćwiczenia

Celem ćwiczenia jest dokładniejsze poznanie pomiaru sygnałów analogowych oraz wykorzystania modulacji PWM do sterowania urządzeniami. W czasie zajęć wykorzystasz wbudowany przetwornik analogowo-cyfrowy Raspberry Pi Pico 2 W, sprzętowy generator PWM, diodę LED, serwomechanizm oraz Analog Discovery II.

Na Laboratorium 1 przetwornik ADC był wykorzystywany głównie jako gotowe narzędzie do odczytu napięcia z fotorezystora. Tym razem interesuje nas również to, **jak** wykonywany jest pomiar: jaka jest rozdzielczość przetwornika, dlaczego kolejne wyniki mogą się nieznacznie różnić oraz jak można wygładzić zmieniający się sygnał za pomocą średniej ruchomej.

Druga część laboratorium dotyczy modulacji PWM. Najpierw zaobserwujemy przebieg PWM za pomocą Analog Discovery II i wykorzystamy go do regulacji jasności diody LED. Następnie zastosujemy PWM do sterowania serwomechanizmem oraz sprawdzimy, w jaki sposób filtr RC może zamienić szybkozmienny przebieg cyfrowy na napięcie zbliżone do analogowego.

ADC i PWM — dwa różne kierunki przetwarzania

- **ADC** (*Analog-to-Digital Converter*) zamienia napięcie analogowe na wartość liczbową, którą może przetwarzać program.
- **PWM** (*Pulse Width Modulation*) generuje przebieg cyfrowy o zmiennym współczynniku wypełnienia. Sygnał nadal przyjmuje tylko dwa poziomy napięcia, ale poprzez zmianę czasu trwania stanu wysokiego można sterować niektórymi komponentami elektronicznymi.

2 Wyposażenie stanowiska

Do wykonania ćwiczenia potrzebne są:

- płytka dydaktyczna z Raspberry Pi Pico 2 W,
- przewód USB i przewody połączeniowe,
- jeden z potencjometrów 10 kΩ znajdujących się na płytce dydaktycznej,
- jedna z diod LED dostępnych na płytce dydaktycznej,
- serwomechanizm S90,
- rezystor 1 kΩ oraz kondensator około 10 μF,
- Analog Discovery II z programem WaveForms.

Przygotowanie Raspberry Pi Pico 2 W

Na kolejnych laboratoriach MicroPython pozostaje zainstalowany na płytce. Jeżeli z jakiegoś powodu interpreter nie jest dostępny, wgraj go ponownie zgodnie z procedurą poznaną na Laboratorium 0.

Uwaga

Wejścia ADC Raspberry Pi Pico 2 W pracują w zakresie od 0 do około 3,3 V. Potencjometry na płytce dydaktycznej są zasilane napięciem 3,3 V, dlatego ich wyjścia mogą być bezpiecznie podłączone do wejścia ADC Pico.
Serwomechanizmu **nie zasilaj z wyjścia 3,3 V Raspberry Pi Pico**. Do jego zasilania wykorzystaj linię 5 V stanowiska i pamiętaj o połączeniu wspólnej masy układu.

3 Zadanie 1: Przetwornik analogowo-cyfrowy ADC

Raspberry Pi Pico 2 W udostępnia trzy wejścia ADC na pinach `GP26`, `GP27` i `GP28`. Wbudowany w RP2350 przetwornik ma rozdzielczość 12 bitów. Oznacza to $2^{12} = 4096$ możliwych poziomów wyniku pomiaru.

Dla zakresu 0–3,3 V idealny krok kwantyzacji wynosi w przybliżeniu:

$$\Delta U = \frac{3{,}3\,\mathrm{V}}{4096} \approx 0{,}806\,\mathrm{mV}. \tag{1}$$

Dlaczego `read_u16()` zwraca aż 0–65535?

Sprzętowy ADC ma rozdzielczość 12 bitów, ale MicroPython funkcją `read_u16()` zwraca wynik przeskalowany do 16-bitowego zakresu od 0 do 65535. Większa liczba możliwych wartości zwracanych przez funkcję **nie zwiększa fizycznej rozdzielczości przetwornika**.

3.1 Pomiar napięcia z potencjometru

Na płytce dydaktycznej znajdują się dwa potencjometry oznaczone **POT1** i **POT2**. Są one już podłączone pomiędzy linię 3,3 V i GND, a napięcie na ich wyjściu zmienia się w przybliżeniu w zakresie od 0 do 3,3 V.

Połącz wyjście **POT1** z pinem `GP26` (ADC0).

Zadanie 1.1 — Odczyt ADC i przeliczenie na napięcie

Uruchom program:

```
1 from machine import ADC, Pin
2 from utime import sleep_ms
```



```
adc = ADC(Pin(26))

while True:
    raw = adc.read_u16()
    voltage = raw * 3.3 / 65535
    print(raw, voltage)
    sleep_ms(200)
```

Ustaw potencjometr kolejno w co najmniej pięciu położeniach obejmujących możliwie szeroki zakres napięcia. Dla każdego położenia zanotuj:

- wynik `raw`,
- napięcie obliczone w programie,
- napięcie zmierzone za pomocą Analog Discovery II.

Porównaj obie wartości napięcia. Czy różnica jest jednakowa w całym zakresie?

Pomiar 1.1 — Weryfikacja za pomocą Analog Discovery II

W WaveForms uruchom narzędzie **Scope**. Kanał 1 podłącz do środkowego wyprowadzenia potencjometru, a masę Analog Discovery II do GND płytki. Ustaw sprzężenie DC.
Sprawdź również napięcia bliskie 0 V, połowie zakresu oraz 3,3 V. Nie przekraczaj napięcia 3,3 V na wejściu ADC.

4 Zadanie 2: Powtarzalność pomiaru i bufor cykliczny

Nawet przy nieruchomym potencjometrze kolejne wyniki ADC nie muszą być identyczne. Wpływają na to między innymi szumy czy zakłócenia zasilania.

Zadanie 2.1 — Seria pomiarów



Ustaw potencjometr tak, aby napięcie wejściowe wynosiło około 1,5–2 V. Następnie wykonaj serię 200 pomiarów:

```
values = []

for i in range(200):
    values.append(adc.read_u16())

print("min =", min(values))
print("max =", max(values))
print("avg =", sum(values) / len(values))
```

Uruchom program kilka razy bez zmiany położenia potencjometru. Sprawdź:

- czy wszystkie wyniki są jednakowe,



- jaka jest różnica pomiędzy wartością maksymalną i minimalną,
- czy wartość średnia zmienia się pomiędzy kolejnymi seriami.

Bufor cykliczny i średnia ruchoma



W systemie pomiarowym często interesuje nas nie średnia ze wszystkich pomiarów wykonanych od początku programu, lecz średnia z **ostatnich** N **próbek**. Można ją obliczać przy użyciu bufora cyklicznego.
Bufor ma stałą długość. Każdy nowy pomiar zastępuje najstarszą próbkę, a indeks zapisu po dojściu do końca tablicy wraca na jej początek. Taki mechanizm pozwala przechowywać stale tę samą liczbę najnowszych wyników bez ciągłego powiększania listy.

Zadanie 2.2 — Bufor cykliczny i średnia z ostatnich pomiarów



Utwórz bufor przechowujący 16 ostatnich wyników ADC. Na początku wypełnij go 16 pomiarami.
Następnie napisz program, który w pętli:

1. wykonuje nowy pomiar ADC,
2. zapisuje go w miejscu najstarszej wartości w buforze,
3. przesuwa indeks zapisu na kolejną pozycję,
4. oblicza średnią z wszystkich 16 elementów bufora,
5. wypisuje aktualny pomiar oraz średnią.

Do cyklicznego przesuwania indeksu możesz wykorzystać operator reszty z dzielenia:

```
index = (index + 1) % BUFFER_SIZE
```

Powoli obracaj potencjometrem i obserwuj jednocześnie wartość pojedynczego pomiaru oraz średnią z bufora. Jak zmienia się zachowanie układu, gdy zwiększysz rozmiar bufora z 16 do 32 lub 64 próbek?
Zastanów się nad kompromisem: większy bufor lepiej wygładza przypadkowe zmiany wyniku, ale powoduje wolniejszą reakcję na rzeczywistą zmianę napięcia.

5 Zadanie 3: Modulacja PWM

PWM (*Pulse Width Modulation*) polega na szybkim przełączaniu wyjścia pomiędzy stanem niskim i wysokim. W każdym okresie T część czasu t_{ON} sygnał znajduje się w stanie wysokim, a pozostałą część w stanie niskim.

Współczynnik wypełnienia definiujemy jako:

$$D = \frac{t_{\mathrm{ON}}}{T} \cdot 100\%. \tag{2}$$

Częstotliwość przebiegu wynosi:

$$f = \frac{1}{T}. \tag{3}$$

Zmiana współczynnika wypełnienia nie zmienia poziomu logicznego wyjścia — pin nadal przyjmuje tylko około 0 V lub 3,3 V. Zmienia się natomiast część okresu, przez którą wyjście pozostaje w stanie wysokim.

Zadanie 3.1 — Pierwszy sygnał PWM

Podłącz jedną z diod LED płytki do `GP15`. Uruchom program:

```
from machine import Pin, PWM

pwm = PWM(Pin(15))
pwm.freq(1000)
pwm.duty_u16(32768)
```

Za pomocą Analog Discovery II zaobserwuj sygnał na `GP15`. Zmierz:

- częstotliwość,
- okres,
- współczynnik wypełnienia,
- poziom niski i wysoki.

Następnie ustaw wypełnienie odpowiadające w przybliżeniu 10%, 25%, 50%, 75% i 90% i porównaj przebiegi w WaveForms.

LED na płytce jest aktywna stanem niskim

Dioda LED używana w ćwiczeniu jest włączana stanem logicznym 0. Dlatego dla tej konkretnej diody zwiększanie wartości `duty_u16()` powoduje **zmniejszanie** jasności, a nie jej zwiększanie. Jest to skutek sposobu podłączenia LED, a nie właściwość samego PWM.

Zadanie 3.2 — Płynna zmiana jasności LED

Napisz program, który płynnie zmienia jasność diody:

1. od wartości minimalnej do maksymalnej,
2. następnie od maksymalnej z powrotem do minimalnej,
3. cały cykl powtarza się w nieskończoność.

Wykorzystaj pętle `for`, niewielki krok zmiany wartości PWM oraz krótkie opóźnienie pomiędzy kolejnymi krokami.

Pamiętaj o aktywnej stanem niskim diodzie. Wygodnie jest wprowadzić zmienną `brightness` opisującą jasność w sposób intuicyjny i dopiero podczas ustawiania PWM zastosować:

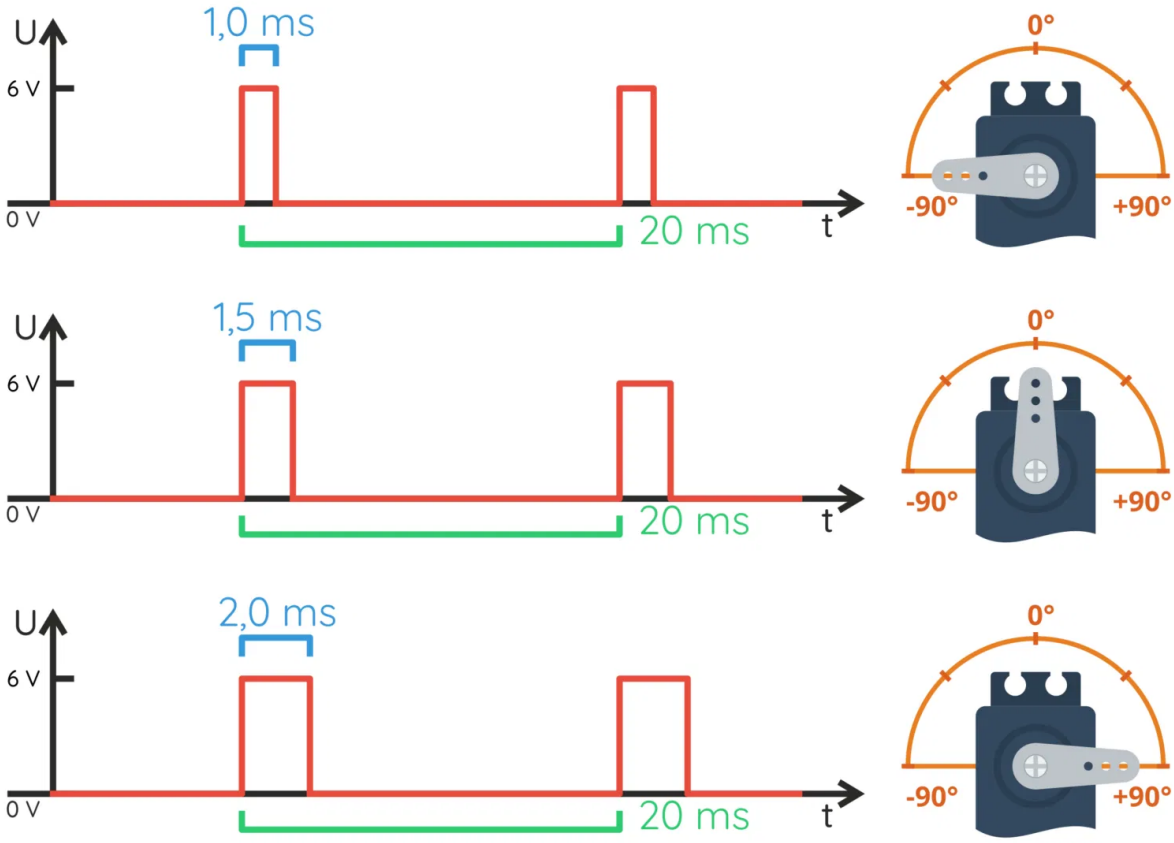
```
pwm.duty_u16(65535 - brightness)
```

Dobierz krok i opóźnienie tak, aby zmiana jasności wyglądała płynnie. Następ-

nie sprawdź, jak zmienia się efekt dla częstotliwości PWM 100 Hz, 1 kHz i 10 kHz.

6 Zadanie 4: Sterowanie serwomechanizmem

W typowym serwomechanizmie położeniu jest zakodowana w **czasie trwania impulsu** powtarzanego zwykle co około 20 ms, czyli z częstotliwością około 50 Hz (patrz rysunek 1).



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 używanego na stanowisku serwomechanizmu rozpocznij od bezpiecznego zakresu około:

1,0 ms — jedno skrajne położenie,
1,5 ms — położenie środkowe,
2,0 ms — drugie skrajne położenie.

Dokładny zakres zależy od konkretnego egzemplarza serwomechanizmu, dlatego nie próbuj wymuszać ruchu poza jego mechanicznym zakresem.

Uwaga

Serwomechanizm zasil z linii 5 V stanowiska, a nie z pinu 3,3 V Raspberry Pi Pico. Masa zasilania serwa i masa Raspberry Pi Pico muszą być połączone.

Zadanie 4.1 — Trzy położenia serwomechanizmu

Podłącz przewód sygnałowy serwomechanizmu do `GP16`. Utwórz sygnał PWM o częstotliwości 50 Hz.
Przy okresie $T = 20\,\text{ms}$ wartość dla funkcji `duty_u16()` można obliczyć z czasu impulsu:

$$\text{duty} = \frac{t_{\text{impulsu}}}{20000\,\mu\text{s}} \cdot 65535. \tag{4}$$

Napisz funkcję:

```
def set_pulse_us(pulse_us):
    duty = int(pulse_us * 65535 / 20000)
    servo.duty_u16(duty)
```

Sprawdź kolejno impulsy 1000, 1500 i 2000 μs. Dla każdego przypadku:

- obserwuj położenie serwa,
- zmierz czas impulsu za pomocą Analog Discovery II,
- porównaj wynik z wartością zadaną w programie.

Zadanie 4.2 — Potencjometr jako regulator położenia

Połącz wiedzę z części dotyczącej ADC i PWM. Napisz program, w którym:

1. potencjometr **POT1** jest odczytywany przez ADC,
2. wynik jest wygładzany za pomocą bufora cyklicznego z zadania 2.2,
3. średnia wartość z ADC jest przeliczana na czas impulsu z zakresu 1000–2000 μs,
4. serwomechanizm ustawia położenie odpowiadające położeniu potencjometru.

Do liniowego przeliczenia możesz wykorzystać zależność:

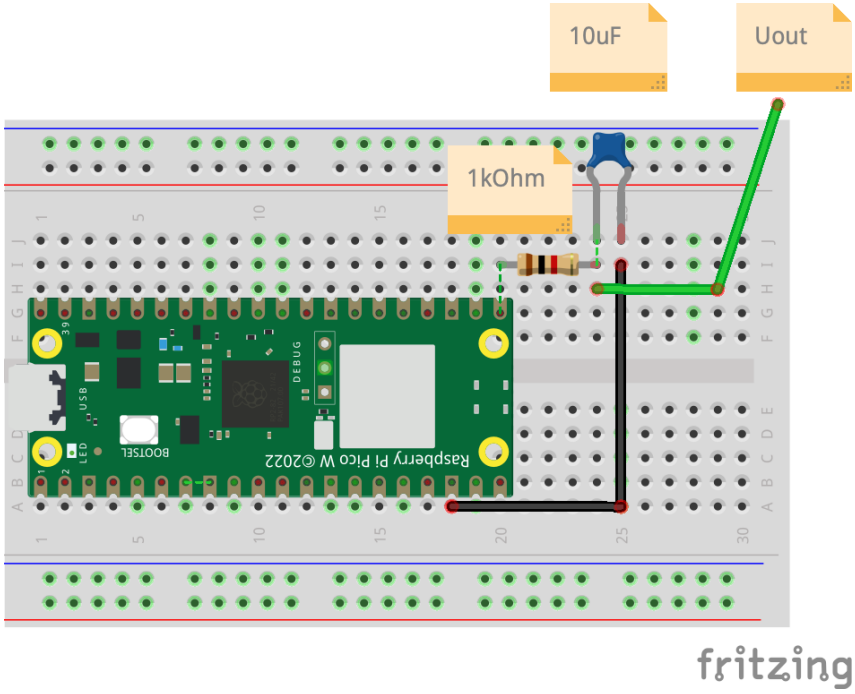
```
pulse_us = 1000 + average * 1000 // 65535
```

Porównaj działanie układu z uśrednianiem i bez uśredniania. Czy drgania lub przypadkowe zmiany położenia serwa są zauważalne?

7 Zadanie 5: PWM jako źródło napięcia quasi-analogowego

Szybkozmienny sygnał PWM można wygładzić za pomocą filtru dolnoprzepustowego. Zbudujemy najprostszy filtr RC:

Jeżeli używasz kondensatora elektrolitycznego, jego biegun dodatni podłącz do punktu `Uout`, a ujemny do GND.



Zadanie 5.1 — Filtracja PWM

Utwórz sygnał PWM na `GP16` o częstotliwości 1 kHz. Kanał 1 Analog Discovery II podłącz bezpośrednio do `GP17`, a kanał 2 do punktu `Uout` za filtrem RC.
Dla wypełnienia około 25%:

1. zaobserwuj jednocześnie przebieg PWM i napięcie za filtrem,
2. zmierz średnie napięcie na wyjściu filtru,
3. porównaj wynik z wartością $D \cdot 3{,}3\,\mathrm{V}$.

Powtórz kroki 1-3 dla wypełnień: 50% i 75%.
Dla wypełnienia 50% porównaj zachowanie filtru przy częstotliwości 100 Hz, 1 kHz i 10 kHz. Zwróć uwagę na amplitudę tętnień napięcia za filtrem.

Czy PWM jest przetwornikiem DAC?

Nie. Na pinie PWM nadal występuje sygnał cyfrowy, który przełącza się pomiędzy dwoma poziomami. Filtr RC pozwala otrzymać napięcie zbliżone do wartości średniej, ale pozostają tętnienia, a szybkość zmian napięcia zależy od parametrów filtru. Prawdziwy DAC generuje bezpośrednio określony poziom napięcia analogowego. Poznasz go na kolejnych laboratoriach.

8 Podsumowanie

Po tych zajęciach powinieneś umieć



- wyjaśnić różnicę pomiędzy fizyczną rozdzielczością ADC a zakresem zwracanym przez `read_u16()`,
- przeliczyć wynik ADC na napięcie i oszacować krok kwantyzacji,
- ocenić powtarzalność pomiaru,
- zbudować bufor cykliczny i obliczać średnią z ostatnich próbek,
- wyjaśnić pojęcia okresu, częstotliwości i współczynnika wypełnienia PWM,
- wykorzystać PWM do płynnej regulacji jasności diody LED,
- wygenerować sygnał sterujący serwomechanizmem,
- połączyć pomiar ADC, uśrednianie i sterowanie PWM w jeden układ,
- wyjaśnić, dlaczego przefiltrowany PWM może przypominać napięcie analogowe, ale nie jest klasycznym wyjściem DAC.

Uwaga



Przed opuszczeniem stanowiska zatrzymaj program i usuń z pamięci Raspberry Pi Pico pliki utworzone podczas tego laboratorium.