

Graficzne programowanie mikrokontrolerów

M5Stack / UIFlow

Laboratorium 2

Wejścia cyfrowe i analogowe

Czas trwania: 3 × 45 min**Tryb:** studia niestacjonarne**Jednostka:** Katedra Elektrotechniki i Technologii Nadprzewodnikowych**Prowadzący:** dr inż. Michał Łanczont

Cel ćwiczenia

Celem ćwiczenia jest obsługa wejść i wyjść cyfrowych oraz przetwarzanie sygnałów analogowych na platformie M5Stack. Student odczytuje przycisk i kanały kwadraturowe enkodera, wykrywa zbocza i eliminuje drgania styków, rozpoznaje kierunek oraz zlicza kroki. Obie osie joysticka odczytuje bezpośrednio przez wewnętrzny ADC ESP32, przeprowadza kalibrację, normalizację, filtrację i dobór martwej strefy. Przygotowane sygnały wykorzystuje do sterowania interfejsem graficznym i wyjściami przekąźnikowymi.

1. Wprowadzenie

Wyposażenie: Core + Demo Board, enkoder z przyciskiem, joystick, jedno wyjście przekąźnikowe i przewody według schematu stanowiska.

Organizacja pracy

Zapoznaj się z opisem stosowanych elementów, wykonaj cztery zadania cząstkowe, a następnie połącz przygotowane funkcje w jednym zadaniu końcowym. Polecenia, szczegółowe opisy, algorytmy i testy znajdują się bezpośrednio przy zadaniach. Zachowuj działające wersje projektu i wykorzystuj je w kolejnych etapach.

Obowiązuje zakres określony w poleceniach zadań cząstkowych i zadania końcowego. Opisane możliwości rozbudowy pozwalają pogłębić materiał. Wyniki testów i zdjęcia stanowiska zbierz podczas zajęć, a sprawozdanie opracuj po zajęciach.

Wejścia i wyjścia cyfrowe

Sygnał cyfrowy

Wejście cyfrowe rozróżnia dwa poziomy logiczne. W programie są one zwykle interpretowane jako 0/1, False/True albo LOW/HIGH. Wyjście cyfrowe pozwala ustawić odpowiedni poziom na pinie lub sterować elementem pośredniczącym, np. przekąźnikiem.

Stan wejścia nie powinien pozostawać nieokreślony. W praktyce stosuje się rezystor *pull-up* lub *pull-down*. W przypadku elementów mechanicznych trzeba dodatkowo uwzględnić drgania styków, przez które jedno fizyczne naciśnięcie przycisku lub pojedyncza zmiana stanu enkodera może zostać zinterpretowana wielokrotnie.

W *UIFlow* wejścia i wyjścia cyfrowe można obsługiwać za pomocą obiektów sprzętowych odpowiadających pinom GPIO. Przed użyciem należy skonfigurować odpowiedni pin jako wejście albo wyjście oraz, jeżeli jest to potrzebne, włączyć podciąganie. Numery GPIO wykorzystywane na stanowisku należy przyjąć zgodnie ze schematem połączeń przygotowanym dla laboratorium.

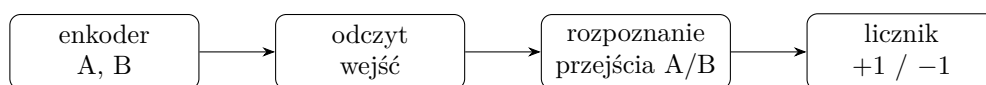
Enkoder inkrementalny z przyciskiem

Enkoder na płytce **DEMO BOARD** jest elementem cyfrowym. Udostępnia trzy niezależne sygnały: dwa kanały enkodera **A** i **B** oraz sygnał **BUTTON** pochodzący od przycisku umieszczonego w osi pokrętła. Na złączu enkodera płytki są one opisane jako **IN_A**, **IN_B**, **BUTTON** oraz **GND**.

Połączenie stanowiska

Kanałów A i B nie należy traktować jako wejść analogowych. Do ćwiczenia połącz **IN_A**, **IN_B** i **BUTTON** z trzema wejściami cyfrowymi wskazanymi w schemacie stanowiska oraz połącz wspólną masę. Przed uruchomieniem programu sprawdź, czy wybrane wejścia mogą pracować z wymaganiem podciąganiem.

Enkoder inkrementalny nie podaje bezpośrednio bezwzględnego położenia pokrętła. Podczas obrotu generuje dwa przebiegi cyfrowe przesunięte w fazie. Na podstawie kolejności zmian A i B można określić kierunek obrotu oraz zliczać kolejne kroki.



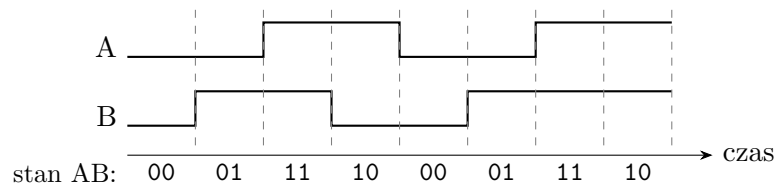
Dla jednego kierunku obrotu poprawna sekwencja stanów może mieć postać:

00 → 01 → 11 → 10 → 00

a dla kierunku przeciwnego:

00 → 10 → 11 → 01 → 00.

To, która sekwencja oznacza obrót w prawo, zależy od sposobu podłączenia kanałów i przyjętej konwencji. Po uruchomieniu programu należy to sprawdzić doświadczalnie.



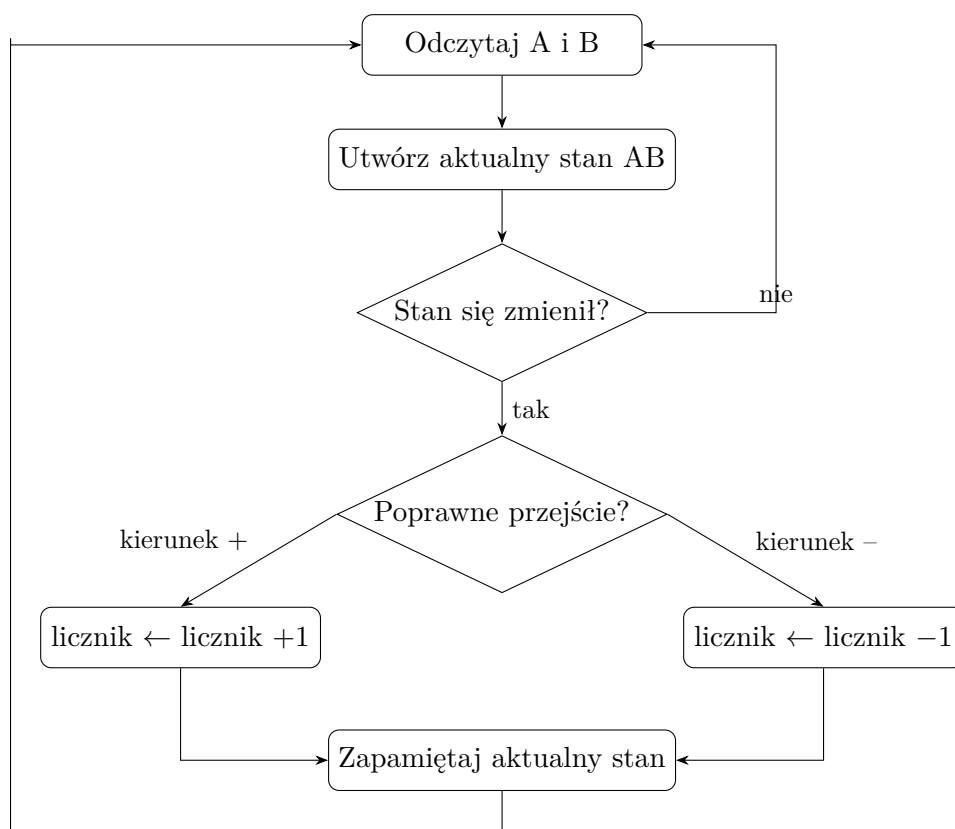
Co oznacza jeden krok?

W jednym pełnym cyklu kwadraturowym występują cztery poprawne przejścia pomiędzy stanami A/B. Program może zliczać tylko jedno wybrane zbocze, dwa zbocza albo wszystkie cztery przejścia. W ćwiczeniu należy przyjąć jedną konsekwentną definicję kroku, tak aby jeden wyczuwalny skok pokrętła powodował przewidywalną zmianę wartości użytkowej.

Najprostsza metoda programowa polega na zapamiętaniu poprzedniego stanu A/B i sprawdzaniu, czy kolejne przejście należy do sekwencji dla kierunku dodatniego lub ujemnego.

Stan poprzedni	Stan aktualny	Zmiana licznika
00	01	+1
01	11	+1
11	10	+1
10	00	+1
00	10	-1
10	11	-1
11	01	-1
01	00	-1

Pozostałe przejścia, np. 00 → 11, oznaczają pominięcie stanu albo zakłócenie i w prostym algorytmie można je zignorować.



Pseudokod dekodowania:

```

poprzedni_stan = odczytaj_AB()
licznik = 0

petla:
    biezacy_stan = odczytaj_AB()
    if biezacy_stan != poprzedni_stan:
        if przejście_w_przód(poprzedni_stan, biezacy_stan):
            licznik = licznik + 1
        else if przejście_w_tył(poprzedni_stan, biezacy_stan):
            licznik = licznik - 1
        poprzedni_stan = biezacy_stan
  
```

Przycisk enkodera jest niezależnym wejściem cyfrowym. Powinien być obsługiwany z detekcją zbocza i eliminacją drgań styków, dzięki czemu pojedyncze naciśnięcie wywołuje dokładnie jedno zdarzenie.

Pomiar i przetwarzanie sygnałów analogowych

Budowa joysticka

Joystick na płytce **DEMO BOARD** zawiera dwa potencjometry ustawione prostopadle. Wychylenie drążka zmienia napięcie na wyjściach osi X i Y. Są to sygnały analogowe, dlatego przed wykorzystaniem w programie muszą zostać przetworzone na liczby przez przetwornik ADC. Naciśnięcie drążka uruchamia dodatkowy przycisk Z, który jest sygnałem cyfrowym i należy traktować go niezależnie od obu osi analogowych.

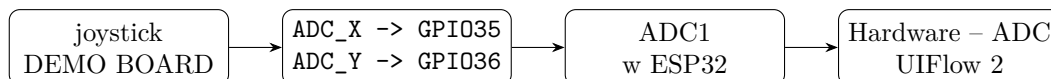
Na złączu joysticka płytki sygnały są opisane jako ADC_X, ADC_Y, BTN1 i GND. W tym ćwiczeniu sygnały osi X i Y należy doprowadzić przewodami do **wejść ADC mikrokontrolera**. Zalecane jest wykorzystanie GPIO35 i GPIO36, ponieważ oba piny należą do bloku ADC1 ESP32 i są dostępne na

złącza M5-Bus zarówno w Core1, jak i Core2. Przykładowe przypisanie: ADC_X -> GPIO35, ADC_Y -> GPIO36. Masa joysticka musi być połączona z masą mikrokontrolera. Sygnał BTN1 – jeżeli jest wykorzystywany – należy podłączyć do wolnego wejścia cyfrowego zgodnie ze schematem stanowiska.



Wewnętrzny przetwornik ADC mikrokontrolera

W mikrokontrolerze ESP32 wejścia analogowe są obsługiwane przez wbudowane bloki ADC. W UIFlow 2 należy wykorzystać elementy z grupy **Hardware – ADC**, utworzyć osobny obiekt ADC dla każdej osi i wskazać odpowiedni pin. W ćwiczeniu zaleca się stosowanie pinów GPIO35 i GPIO36, należących do ADC1. Unika się dzięki temu ograniczeń ADC2, który w ESP32 współdzieli zasoby z modulem Wi-Fi.



Przetwornik ADC zamienia napięcie wejściowe na liczbę. Wartość surowa zależy od rozdzielczości, tłumienia wejścia, napięcia oraz rzeczywistych parametrów potencjometru joysticka. Nie należy więc zakładać idealnych wartości skrajnych. Wartości minimum, środek i maksimum należy wyznaczyć eksperymentalnie dla każdego stanowiska.

W UIFlow 2 funkcja `read()` zwraca surowy wynik ADC, natomiast `read_u16()` zwraca wynik przeskalowany do zakresu 0–65535. W całym ćwiczeniu należy konsekwentnie stosować ten sam rodzaj odczytu dla obu osi.

Typowy tor przetwarzania ma postać:

GPIO ADC → ADC ESP32 → kalibracja → filtracja → normalizacja → martwa strefa → GUI/sterowanie

Dla każdej osi należy wyznaczyć co najmniej trzy wartości: minimum, środek i maksimum. Ponieważ położenie środkowe nie musi znajdować się dokładnie w połowie zakresu, korzystniejsze jest osobne skalowanie obu połówek charakterystyki:

```

if surowa >= srodek:
    wartosc = mapuj(surowa, srodek, maksimum, 0, 100)
else:
    wartosc = mapuj(surowa, minimum, srodek, -100, 0)
wartosc = ogranicz(wartosc, -100, 100)
  
```

Takie przeliczenie pozwala uzyskać wartość bliską 0 w położeniu spoczynkowym nawet wtedy, gdy rzeczywiste zakresy po obu stronach środka nie są symetryczne.

Martwa strefa i filtracja

Wartość joysticka w spoczynku może nieznacznie zmieniać się z powodu tolerancji potencjometrów, szumu pomiarowego i ograniczonej rozdzielczości układu. Z tego powodu wokół środka stosuje się **martwą strefę**. Jeżeli wartość znormalizowana znajduje się np. w zakresie $-10 \dots +10$, można przyjąć wartość 0.

```

if abs(wartosc) <= martwa_strefa:
    wartosc = 0
  
```

Drugim sposobem ograniczenia wahań jest filtracja. W tym ćwiczeniu wystarczy prosta średnia ruchoma z kilku ostatnich próbek. Większa liczba próbek poprawia wygładzenie, ale zwiększa opóźnienie reakcji.

Wskazówki do wariantu realizowanego na zajęciach

Zadania 2 i 4 wykorzystaj w zadaniu końcowym w zakresie jednego przełącznika i kursora prędkościowego. Pełny wybór trzech przełączników oraz oba tryby kursora pozostają opisanymi rozwinięciami. Pełne dekodowanie kwadraturowe i wszystkie metody przetwarzania ADC pozostają w materiale.

Sygnały cyfrowe i analogowe

Wejście GPIO rozróżnia poziomy logiczne 0 i 1. Dla przycisku trzeba ustalić poziom aktywny oraz sposób podciągania. Drgania styków mogą wywołać kilka pozornych naciśnieć. Enkoder ma cyfrowe kanały A/B, a joystick dostarcza napięcie zależnych od wychylenia X/Y.

ADC zamienia napięcie na liczbę. Parametry minimum, środek i maksimum wyznacz dla rzeczywistego joysticka, osobno dla każdej osi. Nie zakładaj, że środek stanowi dokładnie połowę skali.

Połączenia i przygotowanie

IN_A, IN_B i BUTTON enkodera podłącz do wejść cyfrowych wskazanych w schemacie. ADC_X joysticka połącz z GPIO35, ADC_Y z GPIO36 oraz połącz masy. Sprawdź dostępność pinów na stanowisku. GPIO35/36 są wejściami; nie używaj ich jako wyjść przełącznika.

Przycisk i enkoder

Akceptuj zmianę przycisku dopiero po utrzymaniu nowego poziomu przez około 30 ms. Akcję wykonuj na zboczu stabilnego sygnału, a nie w każdej iteracji pętli. Nie blokuj przy tym odczytu kanałów A/B.

Przykładowy cykl enkodera: 00 → 01 → 11 → 10 → 00; odwrotna kolejność oznacza przeciwny kierunek. Zapamiętuj poprzedni stan, dodawaj +1 dla przejścia zgodnego z cyklem i -1 dla odwrotnego. Brak zmiany daje 0; przejście z jednoczesną zmianą obu bitów pomiń. Ustal, ile przejść przypada na wyczuwalny skok pokrętle.

Kalibracja i normalizacja joysticka

Odczytuj obie osie w ten sam sposób, np. surowy wynik ADC. Dla próbki v i kalibracji $\min < \text{środek} < \max$ użyj poniższych zależności. Nie wykonuj dzielenia, jeżeli kalibracja jest niepoprawna.

```
gdy v >= srodek: u = 100*(v-srodek)/(max-srodek)
gdy v < srodek: u = 100*(v-srodek)/(srodek-min)
u = ogranicz(u, -100, 100)
gdy abs(u) <= martwa_strefa: u = 0
```

Filtruj wartości surowe średnią z ostatnich czterech próbek, a następnie normalizuj i stosuj martwą strefę, początkowo 10. Na początku dziel sumę przez faktyczną liczbę próbek. Większe okno filtru wygładza wskazania, lecz opóźnia reakcję.

Przełącznik

Przełącznik jest wyjściem dwustanowym. Zmienna programu przechowuje żądany stan, a jedna funkcja ustawia wyjście i GUI. Po starcie oraz błędzie wyjście ma być wyłączone.

Bezpieczeństwo

Połączenia zmieniaj przy wyłączonym zasilaniu. Do przełącznika dołącz wyłącznie obwód SELV wskazany przez prowadzącego; nie podłączaj napięcia sieciowego.

2. Zadania cząstkowe

Wykonuj zadania w podanej kolejności, rozwijając ten sam projekt. Każde zadanie zawiera polecenie, opis wykonania, przykłady i sposób sprawdzenia wyniku. Przykłady pokazują również dodatkowe warianty działania; zakres wymagany określa polecenie na początku zadania. Zapisuj działającą wersję po każdym etapie.

Tor przetwarzania sygnałów cyfrowych

A/B + przycisk → detekcja zmian → dekodowanie kierunku → licznik / wybór → wyjścia + GUI

Zadanie 1. Pojedyncze naciśnięcie

Polecenie zadania

Odczytaj BUTTON enkodera w spoczynku i po naciśnięciu. Wyznacz poziom aktywny. Dodaj filtr stabilności 30 ms oraz licznik akceptowanych naciśnień. Po starcie zapamiętaj aktualny poziom wejścia.

Warunek odbioru: Dziesięć naciśnień daje dziesięć zdarzeń; przytrzymanie przez 2 s daje jedno.

Sposób wykonania i przykłady

Wykorzystaj poniższe opisy i algorytmy do realizacji polecenia. Szczegółowe warianty wykraczające poza polecenie są możliwościami rozbudowy tego samego zadania.

Odczyt przycisku enkodera

Odczytaj stan przycisku zintegrowanego z enkoderem i wyświetl na ekranie zarówno wartość logiczną wejścia, jak i jej interpretację: ZWOLNIONY albo WCIŚNIĘTY. Ustal doświadczalnie, czy przycisk jest aktywny stanem wysokim, czy niskim.

Sposób realizacji i weryfikacja

1. Odczytaj stan wejścia przed naciśnięciem i podczas naciśnięcia.
2. Nie zakładaj z góry logiki aktywnej HIGH lub LOW – określ ją na podstawie pomiaru.
3. Sprawdź działanie dla krótkiego naciśnięcia i dłuższego przytrzymania.

Detekcja zbocza i eliminacja drgań styków

Zmień program tak, aby zmienna `licznik_nacisniec` zwiększała się dokładnie raz przy każdym naciśnięciu przycisku, niezależnie od czasu jego przytrzymania.

```
biezacy = odczytaj_przycisk()
if biezacy == AKTYWNY and poprzedni == NIEAKTYWNY:
    czekaj_ms(czas_eliminacji_drgan)
    if odczytaj_przycisk() == AKTYWNY:
        licznik_nacisniec += 1
poprzedni = biezacy
```

Weryfikacja

Wykonaj co najmniej dziesięć naciśnień o różnym czasie trwania. Licznik ma zwiększyć się dokładnie o liczbę rzeczywistych naciśnień.

Zadanie 2. Enkoder i nastawa

Polecenie zadania

Wyświetl kanały A/B. Przy wolnym obrocie sprawdź kolejność stanów. Wprowadź dekodowanie przejść opisane we wstępie. Obrót zmienia nastawę o 5 na przyjęty krok użytkowy; ogranicz nastawę do 0–100. Naciśnięcie przycisku zeruje nastawę.

Warunek odbioru: Obrót w przeciwnych kierunkach daje przeciwne zmiany. Zakres 0–100 i zerowanie działają. Zanotuj przyjętą definicję kroku.

Organizacja pętli

Obsługuj enkoder często; GUI odświeżaj np. co 100 ms. Powolne odświeżanie ekranu nie może wymuszać równie rzadkiego odczytu wejść.

Sposób wykonania i przykłady

Wykorzystaj poniższe opisy i algorytmy do realizacji polecenia. Szczegółowe warianty wykraczające poza polecenie są możliwościami rozbudowy tego samego zadania.

Obserwacja stanów A/B enkodera

Odczytuj jednocześnie kanały A i B. Wyświetlaj je na ekranie jako dwubitowy stan AB. Powoli obracaj enkoder w obu kierunkach i zanotuj kolejność pojawiających się stanów.

Sposób realizacji i weryfikacja

1. Wyświetlaj osobno stan A, stan B oraz połączony zapis 00, 01, 10 lub 11.
2. Przy bardzo wolnym obrocie sprawdź, czy obserwowana kolejność odpowiada jednej z sekwencji opisanych we wstępie.
3. Powtórz próbę dla przeciwnego kierunku i potwierdź odwrócenie kolejności stanów.

Zliczanie obrotu i rozpoznawanie kierunku

Zaimplementuj licznik położenia względnego enkodera. Obrót w prawo powinien zwiększać wartość, a obrót w lewo ją zmniejszać. Dozwolone są dwie metody: odczyt kanału B na wybranym zboczu kanału A albo analiza kolejnych stanów A/B na podstawie tabeli przejść.

Sposób realizacji i weryfikacja

1. Przyjmij i zapisz, co w Twoim programie oznacza jeden krok użytkowy.
2. Ogranicz wartość licznika do zakresu, np. 0...100.
3. Jeżeli obrót w prawo zmniejsza licznik, odwróć interpretację kierunku – nie zmieniaj przypadkowo całego algorytmu.
4. Sprawdź szybki obrót w obu kierunkach i zwróć uwagę na ewentualne pominięte lub błędne przejścia.

Enkoder jako element nastawczy

Wykorzystaj licznik z poprzedniego zadania jako wartość **nastawa** w zakresie 0...100. Obrót zmienia nastawę, natomiast krótkie naciśnięcie przycisku ustawia wartość domyślną 50. Na ekranie przedstaw nastawę liczbowo i w postaci paska.

```
if krok == +1: nastawa += 1
if krok == -1: nastawa -= 1
nastawa = ogranicz(nastawa, 0, 100)
```

```
if wykryto_nacisniecie:
    nastawa = 50
```

Sterowanie wyjściami za pomocą enkodera

Wykorzystaj enkoder jako prosty interfejs operatora do sterowania trzema wyjściami przełącznikowymi. Obrót wybiera element R1, R2 lub R3, a krótkie naciśnięcie przycisku przełącza stan aktualnie wybranego wyjścia. Po uruchomieniu programu wszystkie przełączniki muszą być wyłączone.

Sposób realizacji i weryfikacja

1. Zmienna **wybrane_wyjscie** przyjmuje wartości 1–3 i jest zmieniana obrotem enkodera.
2. Naciśnięcie przycisku zmienia wyłącznie stan aktualnie wskazanego przełącznika.
3. Ekran prezentuje numer wybranego wyjścia i rzeczywiste stany R1–R3.
4. Długie przytrzymanie przycisku, np. ponad 1,5 s, wykorzystaj jako polecenie **WYŁĄCZ WSZYSTKO**.
5. Do prób używaj wyłącznie bezpiecznych obwodów SELV wskazanych przez prowadzącego; nie podłączaj napięcia sieciowego.

Rezultat przykładu

Program poprawnie rozpoznaje naciśnięcie przycisku, kierunek obrotu enkodera i jego kolejne kroki. Enkoder może być używany jako element nastawczy i jako interfejs wyboru wyjścia, a po restarcie wszystkie wyjścia pozostają w stanie bezpiecznym.

Tor przetwarzania joysticka

X/Y → GPIO35/36 → ADC ESP32 → kalibracja → filtr → normalizacja → martwa strefa → sterowanie

Zadanie 3. Joystick i kalibracja

Polecenie zadania

Utwórz dwa obiekty ADC. Zmierz min, środek i max dla X oraz Y; zapisz sześć wartości. Znormalizuj osie do -100...100 według wzorów ze wstępu. W razie potrzeby odwróć znak, aby ruch w prawo i w górę dawał wartości dodatnie.

Warunek odbioru: Ekran pokazuje surowe i znormalizowane X/Y. Środek daje wartość bliską zeru, skrajne wychylenia około ± 100 .

Sposób wykonania i przykłady

Wykorzystaj poniższe opisy i algorytmy do realizacji polecenia. Szczegółowe warianty wykraczające poza polecenie są możliwościami rozbudowy tego samego zadania.

Odczyt surowych wartości joysticka

Połącz ADC_X joysticka z GPIO35, ADC_Y z GPIO36 oraz wspólną masę. W UIFlow 2 utwórz dwa obiekty **Hardware** – **ADC** i odczytaj obie osie joysticka. Wyświetl ich surowe wartości na LCD. Zanotuj wskazania dla położenia środkowego oraz dla skrajnych wychyleń w lewo, prawo, górę i dół. Jeżeli w dalszej części wykorzystujesz przycisk Z, odczytuj go osobnym wejściem cyfrowym.

Sposób realizacji i weryfikacja

1. Sprawdź, czy oś X jest rzeczywiście odczytywana z GPIO35, a oś Y z GPIO36.
2. Dla każdej osi zanotuj **minimum**, **srodek** i **maksimum**.
3. Sprawdź, czy wzrost wartości odpowiada ruchowi w prawo i w górę; jeżeli nie, zanotuj konieczność odwrócenia znaku danej osi.
4. Odczyt przycisku Z prezentuj oddzielnie od wartości analogowych.

Sprawdzenie: W sprawozdaniu powinny znaleźć się rzeczywiście zmierzone wartości kalibracyjne obu osi.

Kalibracja i normalizacja zakresu

Przelicz obie osie na zakres od -100 do $+100$, wykorzystując osobno minimum, środek i maksimum każdej osi. Położenie spoczynkowe powinno odpowiadać wartości możliwie bliskiej 0.

Sposób realizacji i weryfikacja

1. Zastosuj osobne parametry kalibracji dla X i Y.
2. Dla wartości poniżej środka mapuj zakres **minimum-srodek** na $-100 \dots 0$.
3. Dla wartości powyżej środka mapuj zakres **srodek-maksimum** na $0 \dots 100$.
4. Po przeliczeniu ogranicz wynik do zakresu $-100 \dots 100$.

Zadanie 4. Filtracja i stabilność**Polecenie zadania**

Dodaj średnią z czterech próbek pobieranych co około 20 ms. Porównaj $N=1$ i $N=4$. Zastosuj martwą strefę 10; zwiększ ją tylko wtedy, gdy po puszczeniu joysticka pozostaje niezerowe sterowanie. Wyznacz intensywność jako $\max(\text{abs}(X), \text{abs}(Y))$.

Warunek odbioru: Po zwolnieniu joysticka X i Y wracają do zera. Pokaż i krótko wyjaśnij wpływ filtru na szum oraz szybkość reakcji.

Zachowaj funkcje odczytu, filtracji i normalizacji do ćwiczeń 3 i 4. Nie kalibruj ponownie bez potrzeby, jeżeli pracujesz na tym samym stanowisku.

Sposób wykonania i przykłady

Wykorzystaj poniższe opisy i algorytmy do realizacji polecenia. Szczegółowe warianty wykraczające poza polecenie są możliwościami rozbudowy tego samego zadania.

Martwa strefa

Dodaj martwą strefę wokół środka. Porównaj wartości 0, 5, 10 i 15 w skali znormalizowanej i wybierz najmniejszą wartość, dla której po zwolnieniu joysticka obie osie stabilnie wracają do zera.

```
if abs(x) <= martwa_strefa: x = 0
if abs(y) <= martwa_strefa: y = 0
```

Filtracja pomiarów

Zaimplementuj średnią ruchomą dla obu osi. Porównaj działanie dla liczby próbek $N = 1$, $N = 4$ i $N = 8$. Obserwuj zarówno poziom wahań w spoczynku, jak i opóźnienie przy szybkim wychyleniu joysticka.

```

bufor.append(probka)
if len(bufor) > N: usun_najstarsza()
filtrowana = sum(bufor) / len(bufor)

```

Weryfikacja

Wybierz jedną wartość N do dalszej części ćwiczenia i krótko uzasadnij wybór: filtr powinien zmniejszać wahania, ale nie może powodować wyraźnie uciążliwego opóźnienia sterowania.

Kierunek i intensywność wychylenia

Na podstawie znormalizowanych wartości X i Y określ kierunek wychylenia joysticka. Jeżeli obie osie są równe 0 po zastosowaniu martwej strefy, wyświetl ŚRODEK. W przeciwnym przypadku wykorzystaj oś o większej wartości bezwzględnej do rozpoznania kierunku LEWO, PRAWO, GÓRA lub DÓŁ. Dodatkowo wyznacz intensywność wychylenia w zakresie 0...100.

```

if x == 0 and y == 0:
    kierunek = SRODEK
else if abs(x) >= abs(y):
    if x > 0: kierunek = PRAWO
    else:    kierunek = LEWO
else:
    if y > 0: kierunek = GORA
    else:    kierunek = DOL

intensywnosc = max(abs(x), abs(y))

```

Kursor sterowany joystickiem

Narysuj na ekranie punkt lub mały prostokąt. Położenie albo prędkość ruchu obiektu ma zależeć od obu osi joysticka. Ogranicz współrzędne tak, aby obiekt nie wychodził poza ekran.

Sposób realizacji i weryfikacja

1. Używaj wartości po kalibracji, filtracji i zastosowaniu martwej strefy – nie steruj kursorem bezpośrednio surowym odczytem ADC.
2. Przelicz $-100 \dots 100$ na zmianę współrzędnych kursora albo bezpośrednią pozycję na ekranie.
3. Ogranicz współrzędne do wymiarów ekranu pomniejszonych o rozmiar obiektu.
4. Po zwolnieniu joysticka kursor w trybie prędkościowym nie powinien samoczynnie dryfować.

```

kursor_x += x / skala
kursor_y += y / skala
kursor_x = ogranicz(kursor_x, 0, szerokosc-szerokosc_obiektu)
kursor_y = ogranicz(kursor_y, 0, wysokosc-wysokosc_obiektu)

```

Rezultat przykładu

Joystick dostarcza stabilnych, skalibrowanych wartości X i Y w zakresie $-100 \dots 100$. Program poprawnie wykrywa położenie środkowe, kierunek i intensywność wychylenia, a obiekt graficzny reaguje płynnie i nie dryfuje po zwolnieniu joysticka.

3. Zadanie końcowe

Polecenie zadania końcowego

Manipulator z progiem wyjścia

Połącz zadania w jedną aplikację. Joystick przemieszcza kursor w dwóch osiach, enkoder ustawia próg 0–100, a przełącznik sygnalizuje przekroczenie progu intensywności wychylenia.

- Co około 50 ms dodawaj do położenia kursora $X/20$ i $-Y/20$. Znak minus dla Y uwzględnia ekranowy kierunek w dół. Ogranicz położenie do obszaru roboczego pomniejszonego o rozmiar kursora.
- Przełącznik włączaj tylko gdy intensywność $>$ próg; przy równości wyłączaj. Zamiast zerowania nastawy przycisk enkodera ma teraz przełączać BLOKADA/AKTYWNY. Start zawsze w BLOKADA, wyjście wyłączone.
- W BLOKADA kursor pozostaje nieruchomy i przełącznik jest wyłączony. Wyświetl próg, intensywność, tryb oraz stan wyjścia.

Testy i odbiór

Próba	Oczekiwana reakcja
Joystick puszczoney	Brak dryfu kursora; intensywność 0.
Ruch przy brzegu ekranu	Kursor pozostaje w obszarze roboczym.
Intensywność poniżej/równa/powyżej progu	Wyjście kolejno 0/0/1 w AKTYWNY.
BLOKADA i restart	Przełącznik wyłączony; brak ruchu kursora.

Kryteria zaliczenia

Jedno zdarzenie na naciśnięcie, odczyt kierunku enkodera, kalibracja obu osi, filtr i martwa strefa, działający kursor oraz sterowanie wyjściem z blokadą.

Sprawozdanie

Oddaj projekt UIFlow i PDF zawierający połączenia, sześć wartości kalibracji, przyjęty krok enkodera, porównanie $N=1$ i $N=4$, zdjęcie aplikacji oraz wyniki testów. Dane zbierz na zajęciach; sprawozdanie zredaguj po zajęciach.

Możliwość rozbudowy: wybór trzech przełączników enkoderem lub przełączanie kursora między trybem pozycji i prędkości.

Możliwości rozbudowy zadania końcowego

Zakres rozszerzeń

Zadanie końcowe możesz rozbudować o opisane niżej funkcje. Wykorzystaj działające moduły przygotowane w zadaniach cząstkowych. Możliwości rozbudowy nie zwiększają obowiązkowego zakresu zadania końcowego. Dobierz testy oraz dokumentację do faktycznie zrealizowanych funkcji.

Wybór i obsługa trzech wyjść przekaźnikowych

Wykorzystanie przygotowanych funkcji

Rozszerzenie projektu ma być bezpośrednim rozwinięciem programów z zadań 1 i 2. Obsługa przycisku wykorzystuje detekcję zbocza i eliminację drgań, obrót enkodera korzysta z jednego wspólnego mechanizmu dekodowania A/B, a stany przekaźników są przechowywane w zmiennych programu i prezentowane na ekranie.

Zbuduj **panel sterowania trzema wyjściami**. Obrót enkodera wybiera R1, R2 lub R3. Krótkie naciśnięcie przełącza stan wybranego wyjścia. Długie przytrzymanie przycisku realizuje polecenie **WYŁĄCZ WSZYSTKO**. Na ekranie pokaż bieżące stany A/B, wartość licznika enkodera, wybrane wyjście oraz stany R1–R3. Program po każdym uruchomieniu musi jawnie ustawić wszystkie wyjścia w stan wyłączony.

Rozszerzenie: dodaj drugi tryb pracy, w którym enkoder nie wybiera przekaźnika, lecz ustawia wartość **nastawa** 0...100; krótkie naciśnięcie przełącza pomiędzy trybem **WYBÓR WYJŚCIA** i **NASTAWA**.

Manipulator z trybem pozycji i prędkości

Wykorzystanie przygotowanych funkcji

Rozszerzenie projektu wykonaj przez połączenie wszystkich etapów toru pomiarowego. Surowe wartości z zadania 3 przechodzą przez kalibrację z zadania 3, filtrację z zadania 4 oraz martwą strefę z zadania 4. Dopiero tak przygotowane wartości mają być użyte do rozpoznawania kierunku i sterowania kursorem.

Zbuduj **graficzny manipulator dwuwymiarowy**. Na ekranie wyświetlaj surowe wartości ADC, wartości po filtracji, wartości znormalizowane X/Y, rozpoznany kierunek oraz intensywność wychyleń. Joystick steruje kursorem. Naciśnięcie przycisku Z przełącza pomiędzy dwoma trybami:

- **POŁOŻENIE** – wychylenie joysticka wyznacza bezpośrednio pozycję kursora względem środka ekranu,
- **PRĘDKOŚĆ** – wychylenie określa kierunek i szybkość przesuwania kursora.

Po zwolnieniu joysticka w trybie **PRĘDKOŚĆ** kursor ma pozostać nieruchomy.

Rozszerzenie: dodaj procedurę ponownej kalibracji środka uruchamianą dłuższym przytrzymaniem przycisku Z.

Dodatkowe testy przy rozbudowie zadania końcowego

W zależności od wybranych funkcji sprawdź poniższe wymagania. Dotyczą one odpowiednich rozszerzeń projektu.

- poprawny odczyt przycisku enkodera i identyfikacja aktywnego poziomu logicznego,
- detekcja pojedynczego naciśnięcia z eliminacją drgań styków,
- poprawny odczyt kanałów A i B oraz rozpoznawanie kierunku obrotu,
- konsekwentne zliczanie kroków enkodera,
- wykorzystanie enkodera i przycisku jako interfejsu operatora,
- bezpieczne sterowanie co najmniej trzema wyjściami cyfrowymi.
- poprawne podłączenie osi joysticka do GPIO35 i GPIO36 oraz odczyt przez wewnętrzny ADC ESP32,

- wyznaczenie i zastosowanie parametrów kalibracji,
- poprawna normalizacja zakresu do $-100 \dots 100$,
- zastosowanie martwej strefy i prostej filtracji,
- określenie kierunku i intensywności wychylenia,
- płynne sterowanie obiektem graficznym bez dryfu w położeniu spoczynkowym.

Dokumentowanie dodatkowych funkcji

Przygotuj sprawozdanie PDF dotyczące wykonanego rozszerzenia i dołącz projekt UIFlow. Udokumentuj zastosowane rozwiązania, wybierając odpowiednie elementy poniższej listy.

- kod programu w UIFlow,
- zdjęcia działającego stanowiska lub czytelne zrzuty ekranu,
- opis sposobu odczytu i dekodowania sygnałów A/B,
- schemat blokowy, diagram stanów lub pseudokod zastosowanego algorytmu,
- opis sposobu testowania programu, w tym test obrotu w obu kierunkach i test przycisku,
- wnioski.
- tabelę wartości kalibracyjnych dla osi X i Y,
- porównanie działania dla wybranych wartości martwej strefy i długości filtru,
- krótki opis toru przetwarzania wraz ze schematem blokowym lub pseudokodem,
- opis sposobu testowania programu,