

Graficzne programowanie mikrokontrolerów

M5Stack / UIFlow

Laboratorium 2

GUI i programowanie zdarzeniowe

Czas trwania: 2 × 45 min**Tryb:** studia stacjonarne**Jednostka:** Katedra Elektrotechniki i Technologii Nadprzewodnikowych**Prowadzący:** dr inż. Michał Łanczont

Cel ćwiczenia

Celem ćwiczenia jest rozwinięcie umiejętności tworzenia interfejsu użytkownika w UIFlow oraz poznanie zasad programowania zdarzeniowego. Student projektuje prosty panel operatorski, powiązuje elementy GUI ze zmiennymi programu i zdarzeniami oraz przygotowuje interfejs reagujący na przyciski M5Stack Core i dotyk ekranu Core2.

Wstęp

Interfejs graficzny w systemie wbudowanym pełni dwie podstawowe funkcje: przedstawia stan procesu oraz umożliwia użytkownikowi wydawanie poleceń. W kolejnych laboratoriach ekran będzie wykorzystywany jako panel HMI, dlatego należy rozdzielić **logikę sterowania** od **logiki prezentacji**.

Przy projektowaniu interfejsu należy przyjąć prostą zależność:

zdarzenie → zmiana stanu → aktualizacja GUI

Elementy interfejsu nie powinny samodzielnie przechowywać informacji o stanie procesu. Stan powinien być zapisany w zmiennych programu, a GUI jedynie go prezentować.

Programowanie zdarzeniowe

Zdarzeniem może być m.in. naciśnięcie przycisku, dotknięcie obszaru ekranu, upływ czasu lub zmiana wartości wejścia. Obsługa zdarzenia powinna być możliwie krótka: zmieniać odpowiednią zmienną, wywoływać funkcję i aktualizować interfejs.

Uwaga

Na stanowisku z M5Stack Core używaj przycisków A/B/C. Core2 traktuj jako oddzielne urządzenie z ekranem dotykowym. Nie osadzaj Core2 mechanicznie na Demo Board.

Przebieg ćwiczenia

Przepływ informacji w panelu operatorskim

przycisk / dotyk → zdarzenie → zmiana stan/nastawa → aktualizuj_gui() →
ekran

Zadanie 1. Budowa ekranu operatorskiego

Do wykonania

Zaprojektuj ekran zawierający: tytuł aplikacji, pole wartości zadanej, pole aktualnego stanu, wskaźnik pracy oraz dwa elementy sterujące odpowiadające funkcjom URUCHOM i ZATRZYMANIE.

Przyjmij trzy stany aplikacji: ZATRZYMANIE, PRACA oraz ALARM. Zadbaj o czytelny podział ekranu i stałe położenie najważniejszych informacji.

Sposób realizacji i weryfikacja

1. Najpierw rozmieść elementy statyczne, potem wartości dynamiczne i na końcu elementy sterujące; unikniesz późniejszego przesuwania całego interfejsu.
2. Przyjmij stałą konwencję prezentacji: nazwa parametru, wartość, jednostka; stan procesu powinien być widoczny bez wchodzenia do dodatkowego ekranu.
3. Przed programowaniem logiki wykonaj test czytelności: z odległości około 0,5–1 m powinny być rozpoznawalne stan i wartość zadana.

Warunek odbioru: Ekran zawiera komplet informacji, a URUCHOM/ZATRZYMANIE są jednoznacznie odróżnione od elementów tylko informacyjnych.

Zadanie 2. Powiązanie GUI ze zmiennymi

Utwórz zmienne `stan`, `nastawa` i `licznik`. Przygotuj funkcję:

```
aktualizuj_gui()
```

Funkcja powinna wyświetlać bieżące wartości zmiennych i zmieniać wygląd kontrolki stanu zależnie od `stan`.

Sposób realizacji i weryfikacja

1. Ustal wartości początkowe `stan=ZATRZYMANIE`, `nastawa=0`, `licznik=0` i dopiero potem wywołaj `aktualizuj_gui()`.
2. W funkcji prezentacji zamień dane na tekst tylko w jednym miejscu; nie przechowuj osobnych kopii tych samych wartości wyłącznie na potrzeby GUI.
3. Dodaj wizualne rozróżnienie ZATRZYMANIE/PRACA/ALARM, ale nie używaj samego koloru jako jedynej informacji - wyświetl także nazwę stanu.

Pseudokod / zapis algorytmu:

```
funkcja aktualizuj_gui():  
    pokaz(nastawa)  
    pokaz(licznik)  
    pokaz(stan)  
    ustaw_wyglad_kontrolki(stan)
```

Zadanie 3. Obsługa przycisków Core

Przypisz:

- A – zwiększenie wartości zadanej,
- B – zmniejszenie wartości zadanej,
- C – przełączenie ZATRZYMANIE/PRACA.

Ogranicz wartość zadaną do zakresu 0–100. Po każdej zmianie wywołaj funkcję aktualizującą GUI.

Sposób realizacji i weryfikacja

1. Dla A i B wykonuj zmianę nastawy krokowo, np. o 1 lub 5, a następnie ogranicz wynik do 0–100.
2. Dla C zaimplementuj świadome przejście ZATRZYMANIE/PRACA; nie przełączaj do PRACA, jeśli aktywny jest ALARM.
3. Przetestuj wartości graniczne 0 i 100 oraz wielokrotne naciskanie przycisków poza zakresem.

Pseudokod / zapis algorytmu:

```
nastawa = ogranicz(nastawa + krok, 0, 100)
if C and stan != ALARM:
    stan = PRACA if stan == ZATRZYMANIE else ZATRZYMANIE
aktualizuj_gui()
```

Zadanie 4. GUI dotykowe na Core2

Na Core2 przygotuj analogiczny panel sterujący, wykorzystując zdarzenia dotykowe. Zaimplementuj co najmniej trzy pola aktywne: URUCHOM, ZATRZYMANIE i ZERUJ. Zdarzenie dotykowe nie może być realizowane przez ciągle sprawdzanie współrzędnych bez kontroli stanu naciśnięcia.

Sposób realizacji i weryfikacja

1. Zdefiniuj obszary dotykowe tak, aby nie nakładały się i były większe niż sam napis; ułatwia to obsługę palcem.
2. Reaguj na zdarzenie naciśnięcia/zwolnienia lub zmianę stanu dotyku, a nie na każdą iterację pętli z palcem przyłożonym do ekranu.
3. Dodaj krótką informację zwrotną po dotknięciu, np. zmianę wyglądu przycisku lub komunikat wykonanej akcji.

Pseudokod / zapis algorytmu:

```
if zdarzenie_dotyku == WCISNIETY and trafiono(URUCHOM):
    zadanie_uruchomienia()
if zdarzenie_dotyku == WCISNIETY and trafiono(ZATRZYMAJ):
    zadanie_zatrzymania()
```

Zadanie 5. Kontrolka stanu i alarm

Dodaj warunek alarmowy: gdy **nastawa** przekroczy 80, aplikacja przechodzi do stanu **ALARM**. W stanie alarmowym:

- kontrolka powinna zmienić kolor,
- na ekranie powinien pojawić się komunikat,
- URUCHOM nie może uruchomić programu,
- ZERUJ powinien skasować alarm po obniżeniu wartości zadanej.

Sposób realizacji i weryfikacja

1. Oddziel warunek wykrycia alarmu od procedury jego kasowania; samo obniżenie nastawy nie powinno automatycznie uruchamiać systemu.
2. Po wejściu do ALARM zapamiętaj ten stan, zablokuj URUCHOM i pokaż przyczynę alarmu.
3. ZERUJ może przywrócić ZATRZYMANIE tylko wtedy, gdy przyczyna alarmu już nie występuje; po resecie wymagaj osobnego URUCHOM.

Pseudokod / zapis algorytmu:

```
if nastawa > 80:
    stan = ALARM

if ZERUJ and nastawa <= 80:
    stan = ZATRZYMANIE
```

Zadanie 6. Rozdzielenie logiki i prezentacji

Uporządkuj program tak, aby zmiana stanu była realizowana w oddzielnej funkcji, a wygląd ekranu w funkcji `aktualizuj_gui()`. Prowadzący powinien móc wskazać w programie część odpowiedzialną za logikę procesu i część odpowiedzialną za prezentację.

Sposób realizacji i weryfikacja

1. Utwórz funkcje typu `zadanie_uruchomienia()`, `zadanie_zatrzymania()`, `sprawdz_alarm()` oraz `aktualizuj_gui()`.
2. Funkcje zdarzeń przycisków/dotyku powinny jedynie zgłaszać żądanie lub zmieniać dane wejściowe; decyzję o dozwolonym przejściu wykonuje logika stanu.
3. Wprowadź jedno miejsce odpowiedzialne za okresowe zwiększanie licznika pracy w PRACA.

Pseudokod / zapis algorytmu:

```
petla:
    sprawdz_alarm()
    if stan == PRACA and minela_1s:
        licznik += 1
    aktualizuj_gui_gdy_zmiana()
```

Wymagany rezultat

Panel operatorski reaguje na zdarzenia, poprawnie prezentuje stan programu i nie zawiera powielonej logiki aktualizacji elementów GUI w wielu miejscach programu.

Zadanie końcowe**Integracja zadań etapowych**

Zadanie końcowe jest scaleniem efektów zadań 1–6. Użyj ekranu z zadania 1, zmiennych programu z zadania 2, obsługi przycisków Core z zadania 3, obsługi dotyku Core2 z zadania 4, alarmu z zadania 5 oraz rozdzielenia logiki i prezentacji z zadania 6. Wszystkie elementy mają pracować na wspólnych zmiennych i być odświeżane przez jedną funkcję interfejsu.

Rozbuduj projekt do panelu sterowania symulowanym procesem. Panel ma zawierać pola **URUCHOM**, **ZATRZYMAJ** i **ZERUJ**, nastawę 0–100, licznik czasu pracy oraz kontrolkę stanu. W

stanie **PRACA** licznik czasu zwiększa się, w **ZATRZYMANIE** pozostaje zatrzymany, a w **ALARM** polecenie **URUCHOM** jest blokowane. Na Core2 sterowanie wykonaj z użyciem ekranu dotykowego.

Rozszerzenie: dodaj komunikat wyjaśniający przyczynę blokady uruchomienia oraz licznik wystąpień alarmu.

Kryteria zaliczenia

Ćwiczenie uznaje się za zaliczone, jeżeli student:

- przygotuje czytelny panel GUI,
- zastosuje obsługę zdarzeń,
- powiąże GUI ze zmiennymi programu,
- utworzy funkcję aktualizującą interfejs,
- zaimplementuje co najmniej trzy stany pracy,
- potrafi wyjaśnić różnicę pomiędzy stanem programu a jego prezentacją.

Sprawozdanie

Podstawą zaliczenia ćwiczenia jest sprawozdanie dotyczące wykonania zadania końcowego. Zadania etapowe są zaliczane na zasadzie demonstracji działania programu na stanowisku. Sprawozdanie należy przesłać w formie pliku **.pdf* oraz pliku projektu programu do weryfikacji. W dokumencie należy przedstawić:

- kod programu w UIFlow,
- zdjęcia działającego stanowiska lub czytelne zrzuty ekranu,
- krótki opis zastosowanego algorytmu wraz ze schematem blokowym, diagramem stanów lub pseudokodem,
- opis sposobu testowania programu,
- wnioski.