

Graficzne programowanie mikrokontrolerów

M5Stack / UIFlow

Laboratorium 10

Projekt integracyjny

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

Cel ćwiczenia

Celem ćwiczenia jest integracja poznanych elementów w jeden niewielki system wbudowany. Student planuje strukturę programu, łączy GUI, wejścia, czujniki, elementy wykonawcze i komunikację, implementuje diagnostykę oraz prezentuje i uzasadnia przyjęte rozwiązania.

Ostatnie laboratorium nie wprowadza nowego rodzaju sprzętu. Zadaniem zespołu jest połączenie elementów poznanych wcześniej w system posiadający czytelnie zdefiniowane wejścia, wyjścia, stany i reakcję na błędy.

Minimalna architektura projektu

Projekt powinien zawierać co najmniej cztery warstwy:

1. wejścia lub czujniki,
2. logikę programu i automat stanów,
3. GUI operatorskie,
4. wyjścia, napęd lub komunikację z drugim urządzeniem.

Dodatkowo należy przewidzieć diagnostykę i bezpieczny stan ZATRZYMANIE/BLAD.

Organizacja pracy

Przed uruchomieniem projektu z napędami lub przekaźnikami zespół musi przedstawić prowadzącemu schemat działania i sposób zatrzymania układu. Nie wolno pomijać ograniczeń bezpieczeństwa tylko po to, aby skrócić program.

Przebieg ćwiczenia

Architektura projektu integracyjnego

wejścia → akwizycja → automat stanów → wyjścia/napęd
GUI ↔ stan systemu ↔ komunikacja/diagnostyka

Zadanie 1. Wybór projektu

Wybierz jeden wariant lub uzgodnij własny z prowadzącym:

- A – kontrola dostępu: RFID + GUI + przekaźnik + ESP-NOW,

- B – sterownik napędu: joystick + enkoder + silnik + zabezpieczenia,
- C – stacja IoT: I2C + GUI + ESP-NOW + Core2 jako terminal,
- D – mini-system przemysłowy: czujniki + automat + napęd/przełącznik + RS485; zalecane rozwinięcie: Modbus RTU Core-Core2.

Sposób realizacji i weryfikacja

1. W jednym zdaniu sformułuj funkcję użytkową systemu: co mierzy/steruje, kto wydaje polecenia i jaki ma być bezpieczny stan.
2. Wybierz minimum trzy grupy funkcjonalne poznane wcześniej oraz wskaż zależności między nimi.
3. Nie rozpoczynaj implementacji, dopóki nie da się narysować przepływu wejście-decyzja-wyjście i wskazać sposobu diagnozowania błędu.

Warunek odbioru: Prowadzący akceptuje zakres projektu przed rozpoczęciem integracji.

Zadanie 2. Specyfikacja wejść i wyjść

Przygotuj tabelę sygnałów. Dla każdego sygnału określ nazwę, źródło, typ danych, zakres oraz sposób wykorzystania. Zdefiniuj także wyjścia i warunki ich aktywacji.

Sposób realizacji i weryfikacja

1. Dla każdego wejścia podaj źródło, typ, zakres, okres odczytu i zachowanie przy błędzie.
2. Dla każdego wyjścia podaj stan bezpieczny oraz warunek jego aktywacji.
3. Dla komunikacji dodaj adres/peer, kierunek, najważniejsze pola danych oraz przekroczenie czasu oczekiwania.

Warunek odbioru: Tabela sygnałów pozwala zaimplementować system bez zgadywania nazw, jednostek i zakresów.

Zadanie 3. Automat stanów

Zdefiniuj co najmniej cztery stany systemu, np. INICJALIZACJA, ZATRZYMANIE, PRACA, BLAD. Określ warunki przejść i działanie systemu w każdym stanie.

Sposób realizacji i weryfikacja

1. Narysuj diagram stanów przed implementacją i opisz każde przejście warunkiem logicznym.
2. BLAD powinien określać działania bezpieczne; samo wyświetlenie napisu BLAD nie jest pełną obsługą błędu.
3. Ustal sposób powrotu z BLAD: automatyczny po ustaniu przyczyny albo ręczny ZERUJ po spełnieniu warunku.

Pseudokod / zapis algorytmu:

```
INICJALIZACJA -> ZATRZYMANIE
ZATRZYMANIE --URUCHOM--> PRACA
PRACA --blad--> BLAD
BLAD --ZERUJ and blad_usuniety--> ZATRZYMANIE
```

Zadanie 4. Implementacja modułowa

Podziel program na funkcje odpowiedzialne za: wejścia, aktualizację GUI, logikę procesu, wyjścia i komunikację. Unikaj jednej bardzo długiej pętli zawierającej całą logikę programu.

Sposób realizacji i weryfikacja

1. Przygotuj osobne funkcje: `odczytaj_wejscia()`, `aktualizuj_stan()`, `ustaw_wyjscia()`, `aktualizuj_gui()`, `obsluz_komunikacje()`.
2. W pętli głównej wywołuj te funkcje w stałej kolejności; unikaj bezpośrednich zmian wyjść w wielu miejscach programu.
3. Wartości globalne ogranicz do rzeczywistego stanu systemu i danych współdzielonych między modułami.

Pseudokod / zapis algorytmu:

```

petla:
    odczytaj_wejscia()
    obsluz_komunikacje()
    aktualizuj_stan()
    ustaw_wyjscia()
    aktualizuj_gui()

```

Zadanie 5. Testy błędów

Przeprowadź co najmniej dwa testy awaryjne właściwe dla projektu, np. utrata ESP-NOW, błędna karta RFID, przekroczenie progu czujnika, brak ramki RS485 lub próba ruchu poza zadany zakres.

Sposób realizacji i weryfikacja

1. Dla każdego testu zapisz: stan początkowy, wprowadzony błąd, oczekiwaną reakcję, wynik rzeczywisty i sposób powrotu.
2. Co najmniej jeden test powinien dotyczyć utraty/niepoprawności danych, a drugi bezpieczeństwa wyjścia lub napędu.
3. Po teście sprawdź nie tylko komunikat GUI, ale również fizyczny stan wyjść.

Warunek odbioru: System przechodzi do przewidzianego stanu i nie pozostawia aktywnego niebezpiecznego polecenia po błędzie.

Zadanie 6. Prezentacja

Przygotuj krótką prezentację działania: cel systemu, schemat blokowy, kluczowe elementy programu, demonstracja normalnej pracy i demonstracja co najmniej jednego błędu.

Sposób realizacji i weryfikacja

1. Pokaż diagram blokowy i automat stanów przed demonstracją sprzętu, aby obserwator znał oczekiwany przepływ informacji.
2. Zademonstruj scenariusz normalny oraz co najmniej jeden błąd w sposób powtarzalny.
3. Wskaż na projekcie UIFlow funkcje odpowiadające za wejścia, logikę, wyjścia, GUI i komunikację.

Warunek odbioru: Każdy członek zespołu potrafi wyjaśnić co najmniej jeden moduł oraz reakcję systemu na błąd.

Wymagany rezultat

Powstaje kompletny mini-system wbudowany, którego działanie można wyjaśnić poprzez przepływ: wejście → decyzja → wyjście oraz poprzez zdefiniowane stany programu.

Zadanie końcowe

Integracja zadań etapowych

Projekt integracyjny jest końcową wersją rozwiązania rozwijanego w zadaniach 1–6 tego laboratorium. Nie należy po specyfikacji rozpoczynać nowej implementacji: lista wejść i wyjść z zadania 2 ma zostać bezpośrednio wykorzystana w automacie z zadania 3, funkcje z zadania 4 realizują jego działanie, a przypadki z zadania 5 stają się obowiązkowymi testami odbiorowymi. Zadanie 6 jest prezentacją dokładnie tego samego projektu.

Zespół prezentuje kompletny mini-system wykorzystujący minimum trzy różne grupy funkcjonalne poznane na wcześniejszych zajęciach, w tym co najmniej jedną formę komunikacji albo identyfikacji. Projekt musi posiadać czytelny interfejs, automat stanów, diagnostykę oraz jednoznaczny stan bezpieczny. Nazwy zmiennych, funkcji tworzonych przez studentów, stanów programu i tekstów wyświetlanych użytkownikowi powinny być polskie i spójne w całym projekcie.

Rozszerzenie: przygotuj ekran diagnostyczny zawierający bieżący stan systemu, ostatni błąd, liczniki zdarzeń oraz informację o sprawności wykorzystywanych kanałów komunikacji.

Kryteria zaliczenia

Ocena projektu powinna uwzględniać:

- poprawność działania i kompletność funkcji,
- czytelność GUI,
- strukturę i modularność programu,
- poprawną obsługę sprzętu,
- reakcję na błędy i stan bezpieczny,
- poprawność komunikacji,
- umiejętność wyjaśnienia rozwiązania podczas obrony.

Sprawozdanie

Sprawozdanie z projektu powinno zawierać: opis celu, schemat blokowy systemu, tabelę wejść i wyjść, diagram stanów, zrzuty programu UIFlow, zdjęcia stanowiska, opis testów normalnych i awaryjnych, wyniki oraz wnioski. Do sprawozdania należy dołączyć plik projektu.